Image Representations in Deep Neural Networks and their Applications to Neural Data Modelling

Dissertation

der Mathematisch-Naturwissenschaftlichen Fakultät

der Eberhard Karls Universität Tübingen

zur Erlangung des Grades eines

Doktors der Naturwissenschaften

(Dr. rer. nat.)

vorgelegt von

Ivan Ustyuzhaninov

aus Jekaterinburg, Russland

Tübingen

2022

Gedruckt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Eberhard Karls Universität Tübingen.

Tag der mündlichen Qualifikation:	01.12.2022
Dekan:	Prof. Dr. Thilo Stehle
1. Berichterstatter:	Prof. Dr. Matthias Bethge
2. Berichterstatterin:	Jun.-Prof. Dr. Anna Levina

Ich erkläre, dass ich die zur Promotion eingerichte Arbeit mit dem Titel “Image representations in deep neural networks and their applications to neural data modelling” selbständig verfasst, nur die angegebenen Quellen und Hilfsmittel benutzt und wörtlich oder inhaltlich übernommene Stellen als solche gekennzeichnet habe. Ich versichere an Eides statt, dass diese Angaben wahr sind und dass ich nichts verschwiegen habe. Mir ist bekannt, dass die falsche Abgabe einer Versicherung an Eides statt mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft wird.

Tübingen, den _____ _____

Datum/Date Unterschrift/Signature

Contents

Acknowledgements

During my time as graduate student I have been fortunate to be surrounded by wonderful people who not only offered scientific discussions, guidance and help but also made this time memorable. I cannot mention everyone on this page for which I want to apologise in advance — I am very thankful to everyone I met.

I had a pleasure of working with multiple supervisors whom I would like to mention first. I learnt very much from Prof. Dr. Matthias Bethge, Dr. Wieland Brendel and Prof. Dr. Alexander Ecker who were always happy to offer me scientific (and not only scientific) advice and created a great working environment for me and other fellow students. I would like to thank Prof. Dr. Fabian Sinz and Dr. Caterina De Bacco for serving on my thesis advisory board, accommodating meetings sometimes on a short notice and providing valuable feedback on my work.

I would like to thank all members of the Bethge Lab for all the scientific discussions we had over the years and the memorable moments outside the work. Santiago Cadena, Max Burg and David Klindt became my good friends, I am especially grateful to them.

I had a chance of collaborating with researchers from different institutions who were very generous to let me join their work on a number of exciting projects and learn a lot of new things. I would like to mention Dr. Neill Campbell, Dr. Carl Henrik Ek, Dr. Markus Kaiser, Dr. Erik Bodin, Jiakun Fu, Zhiwei Ding, and Dr. Andreas Tolias. Thank you very much!

A collaboration with Dr. Ieva Kazlauskaitė was especially important for me. This collaboration led to the scientific results I am most proud of, and moreover it evolved into a good friendship. I have had difficult moments during these years and Ieva's support was invaluable in these moments. I am sure that without it I would not be writing this thesis now.

Finally, I would like to express my gratitude to my parents Elena and Alexei for always being with me.

Abstract

Over the last decade, deep neural networks (DNNs) have become a standard tool in computer vision, allowing us to tackle a variety of problems from classifying objects in natural images to generating new images to predicting brain activity. Such a wide applicability of DNNs is something that these models have in common with the human vision, and exploring some of these similarities is the goal of this thesis.

DNNs much like human vision are hierarchical models that process an input scene with a series of sequential computations. It has been shown that typically only a few final computations in this hierarchy are problem-specific, while the rest of them are quite general and applicable to a number of problems. The results of intermediate computations in the DNN are often referred to as image representations and their generality is another similarity to human vision which also has general visual areas (e.g. primary visual cortex) projecting further to the specialised ones solving specific visual tasks.

We focus on studying DNN image representations with the goal of understanding what makes them so useful for a variety of visual problems. To do so, we discuss DNNs solving a number of specific computer vision problems and analyse similarities and differences of their image representations. Moreover, we discuss how to build DNNs providing image representations with specific properties which enables us to build a "digital twin" of the mouse primary visual system to be used as a tool for studying the computations in the brain.

Taking these results together, we concluded that in general we are still lacking a good understanding of DNN representations. Despite the progress on some specific problems, it still remains largely an open question how the image information is organised in these representations and how to use it for solving arbitrary visual problems. However, we also argue that thinking of DNNs as "digital twins" might be a promising framework for addressing these issues in the future DNN research as they allow us to study image representations by means of computational experiments rather than rely on a priori ideas of how these representations are structured which has proven to be quite challenging.

Zusammenfassung

In den letzten zehn Jahren haben sich tiefe neuronale Netze (Deep Neural Networks, DNNs) zu einem Standardwerkzeug in der Computer Vision entwickelt, mit dem wir eine Vielzahl von Problemen lösen können, von der Klassifizierung von Objekten in natürlichen Bildern über die Erzeugung neuer Bilder bis hin zur Vorhersage von Gehirnaktivitäten. Eine solch breite Anwendbarkeit von DNNs ist etwas, das diese Modelle mit dem menschlichen Sehen gemeinsam haben, und die Erforschung einiger dieser Ähnlichkeiten ist das Ziel dieser Arbeit.

DNNs sind, ähnlich wie das menschliche Sehen, hierarchische Modelle, die eine Eingangsszene mit einer Reihe von aufeinander folgenden Berechnungen verarbeiten. Es hat sich gezeigt, dass typischerweise nur die letzten paar Berechnungen in dieser Hierarchie problemspezifisch sind, während der Rest der Berechnungshierarchie allgemein ist und auf eine Reihe von Problemen anwendbar. Die Ergebnisse der Zwischenberechnungen im DNN werden oft als Bildrepräsentationen bezeichnet, und ihre Allgemeinheit ist eine weitere Ähnlichkeit mit dem menschlichen Sehen, bei dem ebenfalls allgemeine visuelle Areale (z. B. der primäre visuelle Rinde) vorhanden sind, die weiter zu den spezialisierten Arealen projizieren, die spezifische visuelle Probleme lösen.

Wir konzentrieren uns auf die Untersuchung von DNN-Bildrepräsentationen mit dem Ziel zu verstehen, was sie so nützlich für eine Vielzahl von visuellen Problemen macht. Zu diesem Zweck erörtern wir die Bildrepräsentationen in DNNs, die eine Reihe spezifischer Probleme in der Computer Vision lösen, und analysieren ihre Gemeinsamkeiten und Unterschiede. Außerdem erörtern wir, wie DNNs mit Bildrepräsentationen mit spezifischen Eigenschaften gebaut werden können, die es uns ermöglichen, einen "Digital Twin" des primären visuellen Systems der Maus zu bauen und als Werkzeug zur Untersuchung der Berechnungen im Gehirn zu verwenden.

Wir kommen zu dem Schluss, dass es uns im Allgemeinen noch an einem guten Verständnis der DNN-Bildrepräsentationen mangelt. Trotz der Fortschritte bei einigen spezifischen Problemen bleibt die Frage, wie die Bildinformationen in diesen Repräsentationen organisiert sind und wie man sie zur Lösung verschiedener visueller Probleme nutzen kann, weitgehend offen. Wir argumentieren jedoch, dass die Betrachtung von DNNs als "Digital Twin" ein Framework sein könnte, um diese Fragen in der zukünftigen DNN-Forschung anzugehen. "Digital Twin" erlauben es uns, Bildrepräsentationen mit Hilfe von Computerexperimenten zu untersuchen, anstatt uns auf eine vorherige Vorstellung davon zu verlassen, wie diese Repräsentationen strukturiert sind, was sich als ziemlich schwierig erwiesen hat.

1. Introduction

1.1 Why study image representations in deep neural networks?

We as humans greatly rely on vision to perceive the world around us. For example, looking at a photograph of a group of people in front of a beautiful castle, we can easily locate faces of each individual person (and maybe recognise them), infer a three-dimensional model of the castle and imagine how it would look from a different angle, maybe tell the species of the trees in the background based on the shapes on their leaves, etc. It is remarkable that such a rich visual perception ultimately arises from merely a 2D projection of the scene in front of us onto the retina. The visual system implements a series of computations which transform the light intensities of this 2D projection gauged by the retina into a conscious perception.

Since human vision is a computation, it is perhaps unsurprising that in the early days of computer vision, the aim was to build a model mimicking these computations in the human visual system. However, it quickly became clear that such a goal was too ambitious, and the field moved away from trying to model the brain to developing sophisticated, but largely not brain inspired, models for individual tasks (e.g. segmentation, 3D reconstruction, etc.). Moving forward, the last decade has witnessed another change of prevailing models in computer vision marked by a new wave of interest in connections between the human and machine vision with deep neural networks (DNNs) becoming ubiquitous and enabling remarkable progress on a variety of problems. While this connection between machine and human vision is still rather high level and DNNs are only vaguely inspired by the human visual system, the two have an important commonality: the universality of intermediate image representations, which is our main topic of interest in this thesis.

What we mean by the image representations being universal is that the they are informative for a variety of visual tasks. For example, the main visual pathway starts at the retina, goes through the thalamus (LGN) and the primary visual cortex (V1) before reaching more specialised higher visual areas (HVAs). Therefore, the representations of the input scene computed in the retina, LGN and V1 must be applicable for a variety of tasks that the visual system is solving. Deep neural networks have a similar property: representations in a network trained for one task can often be useful for a different task. Such an approach of reusing a network across different tasks is often referred to as transfer learning which has become widely used in recent years.

Unfortunately though, DNNs are similar to the visual system not only in their ability to tackle a variety of complex vision problems but also in difficulty of interpreting their computations. DNNs can approximate any reasonably regular function (which is why they are so useful), however, given a specific DNN it is hard to infer the properties of the function it approximates and why this particular function is the one solving the problem

at hand. Therefore studying DNNs themselves, the properties of the functions they implement and hence of the corresponding image representations is an important research direction, which could allow us not only to better understand and design DNN models, but also to infer computational principles of visual perception in general.

The main focus is studying the DNN image representations for various visual tasks, we aim at identifying common patterns in these representations as well as relating them to human vision. The main body of this thesis is organised into two parts. In the first part (Chapters 3–5), we concentrate on studying three specific aspects of image representations in DNNs: local or texture representations, object representations, as well as a theoretical description of a space of representations consistent with a given function implemented by the network. In the second part (Chapters 6 and 7), we use the DNN representations as a tool for studying the mouse visual system, specifically the functional diversity of neurons in the primary visual cortex.

1.2 List of publications

1.2.1 Publications constituting a part of this thesis

- Ivan Ustyuzhaninov*, Wieland Brendel*, Leon Gatys and Matthias Bethge (2017). What does it take to generate natural textures? In *International Conference on Learning Representations (ICLR)*.

  Contributions I.U., W.B. and M.B. developed the project idea and the theoretical framework with the input from L.G. I.U. conducted numerical experiments and analysed their results with the help from W.B. W.B. wrote the paper draft which was discussed and edited by all authors.

- Julius von Kügelgen*, Ivan Ustyuzhaninov*, Peter Gehler, Matthias Bethge and Bernhard Schölkopf (2020). Towards causal generative scene models via competition of experts. In *ICLR Workshop "Causal Learning for Decision Making"*.

  Contributions P.G., M.B. and B.S. proposed the project idea which was further developed by all authors. I.U. and J.K. developed the software implementation of the proposed method. J.K. developed the theoretical framework with the help from I.U. I.U. conducted numerical experiments, the results of which were analysed by I.U. and J.K. I.U. and J.K. jointly wrote the paper draft which was discussed and edited by all authors.

- Ivan Ustyuzhaninov*, Ieva Kazlauskaitė*, Markus Kaiser, Erik Bodin, Neill Campbell and Carl Henrik Ek (2020). Compositional uncertainty in deep Gaussian processes. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 480–489. PMLR.

  Contributions I.U., I.K., N.C. and C.H.E developed the project idea with the input from M.K. and E.B. I.U. and I.K. derived the theoretical results with the help from all other authors. I.U. and I.K. conducted the numerical experiments. I.U. and I.K. wrote the paper draft which was discussed and edited by all authors.

- Ivan Ustyuzhaninov, Santiago A. Cadena, Emmanouil Froudarakis, Paul G. Fahey, Edgar Y. Walker, Erick Cobos, Jacob Reimer, Fabian H. Sinz, Andreas S. Tolias, Matthias Bethge and Alexander S. Ecker (2020). Rotation-invariant clustering of neuronal responses in primary visual cortex. In *International Conference on Learning Representations (ICLR)*.

Contributions I.U. and A.S.E. developed the project idea with the input from F.H.S, A.S.T and M.B. I.U. developed the theoretical framework with the help from A.S.T. E.F., P.G.F, E.Y.W, E.C., J.R. conducted the animal experiments, collected and processed the raw data. I.U. conducted the numerical experiments with the help from S.A.C. I.U. wrote the paper draft which was discussed and edited by all authors.

- Ivan Ustyuzhaninov, Max F. Burg, Santiago A. Cadena, Jiakun Fu, Taliah Muhammad, Kayla Ponder, Emmanouil Froudarakis, Zhiwei Ding, Matthias Bethge, Andreas S. Tolias and Alexander S. Ecker (2022). Digital twin reveals combinatorial code of non-linear computations in the mouse primary visual cortex. *In Submission*.

  Contributions I.U. and A.S.E. developed the project idea with the input from M.F.B, S.A.C, A.S.T and M.B. J.F., T.M., K.P., E.F. conducted the animal experiments, collected and processed the raw data. I.U. conducted the numerical experiments with the help from M.F.B, S.A.C. and Z.D. I.U. wrote the paper draft which was discussed and edited by all authors.

1.2.2 Other publications

- Ivan Ustyuzhaninov*, Ieva Kazlauskaitė*, Carl Henrik Ek and Neill Campbell (2020). Monotonic Gaussian process flows. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 3057–3067. PMLR.

  Contributions All authors developed the project idea. I.U. and I.K. derived the theoretical results with the help from all other authors. I.U. and I.K. conducted the numerical experiments. I.U. and I.K. wrote the paper draft which was discussed and edited by all authors.

- Ivan Ustyuzhaninov*, Ieva Kazlauskaitė*, Carl Henrik Ek and Neill Campbell (2018). Sequence alignment with Dirichlet process mixtures. In *NeurIPS Workshop "All of Bayesian Nonparametrics"*.

  Contributions All authors developed the project idea. I.U. and I.K. derived the theoretical results with the help from all other authors. I.U. and I.K. conducted the numerical experiments. I.U. and I.K. wrote the paper draft which was discussed and edited by all authors.

- Ivan Ustyuzhaninov, Claudio Michaelis, Wieland Brendel and Matthias Bethge (2018). One-shot texture segmentation. *arXiv preprint arXiv:1807.02654*.

  Contributions I.U., W.B. and M.B. developed the project idea. I.U. conducted the numerical experiments. and W.B. I.U. and C.M. wrote the paper draft which was discussed and edited by all authors.

- Claudio Michaelis, Ivan Ustyuzhaninov, Matthias Bethge and Alexander S Ecker (2018). One-shot instance segmentation. *arXiv preprint arXiv:1811.11507*.

  Contributions C.M., M.B. and A.S.T. developed the project idea. C.M. conducted the numerical experiments with the input from I.U. C.M. wrote the paper draft with the input from I.U. which was discussed and edited by all authors.

- Wieland Brendel, Jonas Rauber, Matthias Kümmerer, Ivan Ustyuzhaninov and Matthias Bethge (2019). Accurate, reliable and fast robustness evaluation. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32.

Contributions W.B. and M.B. developed the project idea with the input from all other authors. W.B., J.R. and M.K. conducted the numerical experiments. W.B. wrote the paper draft was discussed and edited by all authors.

2. Background

In this chapter we briefly summarise the background knowledge necessary for understanding the main results discussed in subsequent chapters.

2.1 Neural networks

Deep neural networks (DNNs) DNNs are parametric machine learning models consisting of multiple computational units with each unit containing a linear transformation followed by a nonlinearity (Goodfellow et al., 2016). Typically, the nonlinearities are pointwise and have a simple functional form, e.g. sigmoid or ReLU (Glorot et al., 2011). That makes computations in a DNN locally simple (linear + simple nonlinearity), but on the other hand, taken together all computational units in a DNN can approximate any sufficiently regular function (Cybenko, 1989; Hornik, 1991).

The most common type of DNNs are feedforward networks, in which the computation units are applied sequentially. The simplest form of such a network can be written as a composition

$$f(\mathbf{x}) = (f^L \circ \ldots \circ f^1)(\mathbf{x}), \quad \text{where} \quad f^\ell(\mathbf{x}) = \sigma(W^\ell \mathbf{x} + \mathbf{b}^\ell). \tag{2.1}$$

Computational units f^ℓ are called layers of the network with the matrix W^ℓ being the weights of the layer ℓ, the vector $\mathbf{b}^\ell$ being the biases of the layer ℓ, and σ being a pointwise nonlinear function. There are various other DNNs architectures (i.e. the ways to arrange computational units), however, we only use feedforward networks for the projects discussed in the next chapters.

The weights and biases (as well as other trainable parameters depending on the architecture) are trained by minimising the loss function which can typically be written as a sum over the training set $\mathcal{D}$ consisting of training examples $\mathbf{x}$ and corresponding training signals y (e.g. category of an object in the image):

$$\{W^\ell\}_{\ell=1}^L, \{b^\ell\}_{\ell=1}^L = \underset{\{W^\ell\}, \{b^\ell\}}{\arg\min} \sum_{(\mathbf{x},y)\in\mathcal{D}} l(\mathbf{x}, y). \tag{2.2}$$

The loss function l corresponds to a specific task (e.g. cross-entropy for classification, or L_2 error for regression). It can also be a function of $\mathbf{x}$ only in case of unsupervised or self-supervised learning. Optimisation problem (2.2) is typically solved using iterative first-order methods (e.g. Bottou, 2010; Kingma and Ba, 2014) which take advantage of backpropagation, an efficient algorithm for computing gradients of the loss functions with respect to the DNN parameters (Rumelhart et al., 1986).

Convolutional neural networks (CNNs) CNNs are a subclass of DNNs in which linear transformations defined by matrices W^ℓ are discrete convolutions (Fukushima and Miyake, 1982; LeCun et al., 2015). The input $\mathbf{x}$ to the network can be of any dimensionality, however, CNNs are widely used for processing images (i.e. two dimensional inputs). This is because convolutions correspond to applying the same local feature extractor to every spatial location in the image, thus encoding an inductive bias that statistics of an input image are independent of the spatial location, which is a reasonable assumption for natural images (Field, 1987). Moreover, using the same computation for every spatial location corresponds to using fewer trainable parameters (so called "weight sharing") facilitating the training process.

2.2 Texture modelling

Textures are spatially stationary images which can be thought of as repeated patterns of some base structures that are sometimes referred to as "textons" (Julesz, 1981). Such images are inherently local, in the sense that knowing only a small piece of an image (a "texton") is sufficient for generating an image of an arbitrary size. In other words, there is no global structure (such as an arrangement of different objects in a photograph, for example) in texture images, making them convenient stimuli for studying visual processing of local features both in the brain and in DNNs. The goal of texture modelling is as follows: given a reference texture image, generate new images which are different on the pixel level from the reference one but visually can be described as depicting the same texture (i.e. images with the "textons" of the reference image in different spatial arrangements; Fig. 2.1).

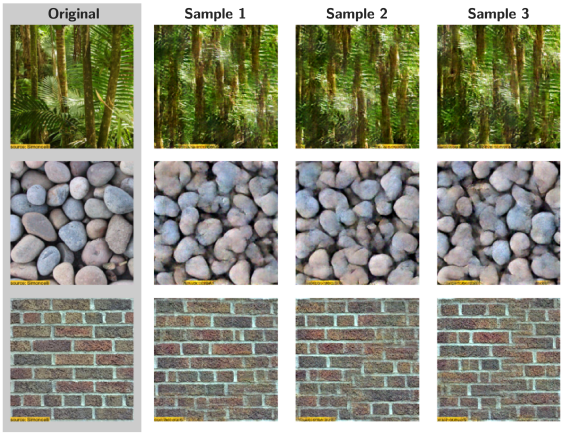


Figure 2.1: Example of texture generation. Given a reference texture (first column), the goal is to generate new images containing same texture (second to fourth columns). The figure is adopted from Ustyuzhaninov* et al. (2017).

Non-parametric modelling One approach to texture modelling aims at directly working with the pixel level "textons" of a reference image by rearranging them to create new images (e.g. Efros and Leung, 1999). Typically these methods are initialised with a piece of a reference or generated texture (of a size comparable to that of a "texton") and they sequentially extend the boundary of the generated image by resampling new pixel values from the reference texture. The main conceptual disadvantage of such methods is that they operate directly on the pixel level and thus do not extract any information about the underlying texture. For example, a human observer would probably describe the texture in the last row of Fig. 2.1 as a "brick wall" summarising the nature of the underlying "textons" and their spatial arrangement, however, non-parametric methods do not typically provide any analogue of such a description.

Parametric modelling The underlying idea of another common approach to texture modelling is more similar to human perception than non-parametric modelling. Given

a reference texture, a parametric method would first create a compact description of this texture (conceptually corresponding to a "brick wall" human description) and then generate new textures by searching for images with similar descriptions. These compact texture representations are obtained using various statistical measurements of the reference texture and they are often referred to as summary statistics.

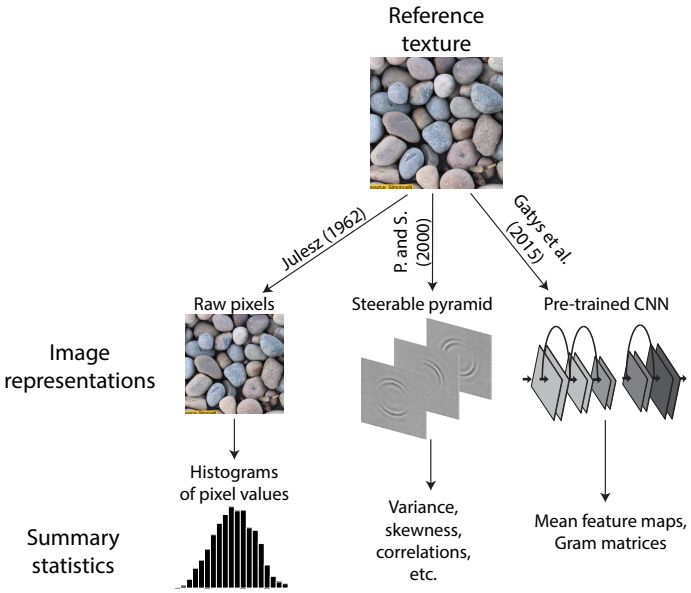


Figure 2.2: An overview of common parametric texture models. These methods compute compact texture representations by applying summary statistics in the image representation space. The illustration of steerable filters is adopted from Portilla and Simoncelli (2000).

These summary statistics could be computed on a pixel level (e.g. N-th order statistics of pixel values in Julesz (1962)), or using image representations of a reference texture obtained by processing it with a certain filter bank. The choice of a filter bank and the corresponding image representations determine the kind of statistical measurements used (Fig. 2.2). For example, Heeger and Bergen (1995) use Laplacian and Steerable pyramids as filter banks supplemented by histograms of resulting representations as summary statistics. Portilla and Simoncelli (2000) also use Steerable pyramids and a number of different summary statistics such as marginal variance, skewness, or correlations between the components of a representation vector. Gatys et al. (2015) use a filter bank of a pre-trained CNN followed by computing mean feature maps and the Gram matrices between the feature maps of the resulting CNN representations.

2.3 Object-centric representations

Generative models of natural images Textures are quite simple images and while the modelling methods based on extracting summary statistics work well for texture generation, they are insufficient for arbitrary natural images. This is because these methods exploit spatial uniformity of texture images which is in general lacking in natural images. It is very hard to find any consistent structure or symmetry in natural images based on which an explicit parametric model can be built, therefore modern state-of-the-art models of natural images are based on end-to-end trained DNNs which implicitly capture the statistical dependencies between the pixels of natural images. Specifically, the two particularly influential CNN based models that emerged in recent years are variational auto-encoders (VAEs; Kingma and Welling, 2013) and generative adversarial networks (GANs; Goodfellow et al., 2014). These models (and their numerous extensions) capture all properties (textures, objects, arrangement of objects, etc.) of images in the training set and they are capable of generating new visually similar images.

Object-centric models While the implicitness of VAEs and GANs is a powerful tool for dealing with such complex objects as arbitrary natural images, it comes with a few notable problems. One of them is that these models offer little flexibility in terms of extracting and manipulating specific aspects of the image, and it is particularly challenging when it comes to extracting representations of individual objects. However, object perception is central to human vision, often humans would describe an image in terms of constituent objects, so object representations in models of natural images are crucial for a number of practical applications and theoretical studies of vision in humans and machines.

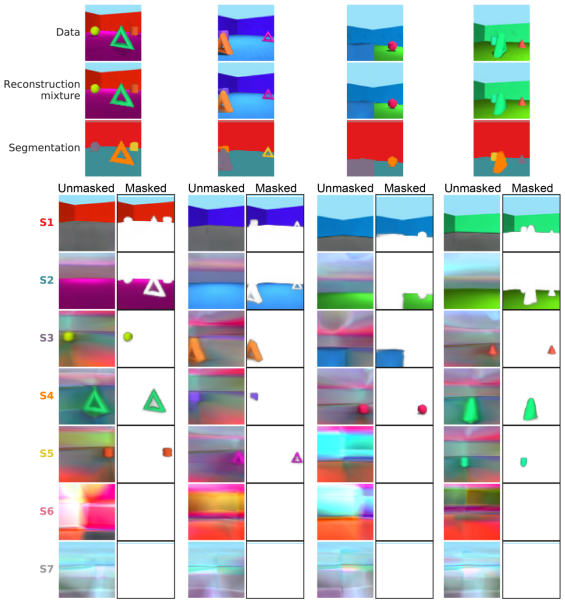


Figure 2.3: Example of an object-centric model. The model processes input images (top row) in seven sequential steps (S1-S7), at each step computing a segmentation mask for one of the objects as well as its texture (colour in this case). The figure is adopted from Burgess et al. (2019)

Multiple recent studies (e.g. Burgess et al., 2019; Engelcke et al., 2019; Greff et al., 2019) addressed this issue by focusing specifically on representations of individual objects in CNNs. Such object-centric models typically process an input image in a sequence of steps such that at every step the model extracts information about one of the objects (Fig. 2.3). They explicitly separate the compositional structure of the scene (e.g. the number of different objects, which ones are in foreground/background, etc.) from the object appearances typically defined by a texture, thus providing explicit object (i.e. global) representations.

2.4 Bayesian deep learning

DNNs fitted to a specific dataset by solving (2.2) provide one possible solution (out of potentially many) of a problem at hand. While this is not an issue in most cases when using a DNN for a specific task, it poses a problem for studying the variety of DNN representations suitable for solving a given problem which will be of interest for us as part of a discussion on DNN representations. In general, probabilistic inference, in particular Bayesian inference, is well-suited for addressing such questions as it allows us to capture the variability or multiple possible values of some quantity (e.g. DNN representations in intermediate layers) by computing an entire distribution of these values rather than a single point estimate (Bishop, 2006).

Deep Gaussian processes There is a substantial amount of research available on applying Bayesian inference to DNNs resulting in what is called Bayesian neural networks (BNNs; MacKay, 1995) which in principle allow us to compute a distribution of the network weights and hence a distribution of representations solving a given problem. However, the main issue with BNNs (as with most other Bayesian models) is that inference in such models is computationally very challenging, so various approximate inference

methods are used. These methods require us to make a decision on the trade-off between the computational efficiency and similarity of the approximate solution (i.e. approximate posterior distribution) to the true one. Approximate Bayesian inference methods for an arbitrary BNN typically have this trade-off greatly skewed towards one of the extremes, therefore we will be focusing on a multi-layer hierarchical model for which there are inference methods with a more balanced efficiency-similarity trade-off, namely deep Gaussian processes (DGPs; Damianou and Lawrence, 2013). DGPs share many of the underlying principles with BNNs making it a convenient model choice for studying the distributions of representations in intermediate layers.

DGPs are compositions of the form (2.1) where each layer f^ℓ is a Gaussian process (GP; Williams and Rasmussen, 2006). GPs, by definition, have jointly Gaussian outputs:

$$f^\ell(\mathbf{x}_1, \dots, \mathbf{x}_N) \sim \mathcal{N}\left(\mu(\mathbf{x}_1, \dots, \mathbf{x}_N), \Sigma(\mathbf{x}_1, \dots, \mathbf{x}_N)\right) \tag{2.3}$$

for some mean and covariance functions μ and Σ. In a general DNN or BNN, the outputs corresponding to different inputs $\mathbf{x}_1, \dots, \mathbf{x}_N$ are dependent in some complex way implicitly defined by their weights. In a GP, however, such a dependence is Gaussian by definition enabling analytic Bayesian inference. Compositions of GPs (i.e. DGPs) no longer have this jointly Gaussian property, but they still retain some structure of GPs enabling us to use efficient and reasonably precise approximate inference methods, one of which is variational inference.

Variational inference The goal of Bayesian inference is to compute a posterior distribution $p(\theta \mid \mathbf{y})$ of some parameter of interest θ (e.g. weights in a DNN) given the data $\mathbf{y}$. To do so, we apply the Bayes' theorem to the prior distribution $p(\theta)$ and the likelihood $p(\mathbf{y} \mid \theta)$:

$$p(\theta \mid \mathbf{y}) = \frac{p(\mathbf{y} \mid \theta)\, p(\theta)}{p(\mathbf{y})}. \tag{2.4}$$

However, computing (2.4) exactly is computationally intractable for most models of interest, including DGPs.

The main problem in (2.4) is typically the intractable denominator $p(\mathbf{y})$, and variational inference offers a method for computing an approximate posterior distribution $q(\theta)$ without the need of computing $p(\mathbf{y})$. One way to find such an approximation is to maximise the following lower bound with respect to $q(\theta)$ (Bishop, 2006):

$$p(\mathbf{y}) \geq \mathbb{E}_q\left[\log p(\mathbf{y} \mid \theta)\right] - \mathrm{KL}\left[q(\theta) \,||\, p(\theta)\right]. \tag{2.5}$$

A family of approximate distributions $q(\theta)$ should be chosen to balance the tractability of (2.5) (that is why we are using VI in the first place) and the quality of approximation of the true posterior $p(\theta \mid \mathbf{y})$. A commonly used approach of stochastic variational inference (SVI; Hoffman et al., 2013) approximates the first term in (2.5) by sampling, thus extending the space of distributions for which the lower bound can be efficiently computed.

2.5 Predictive models of primary visual cortex

Neurons in the visual system can be thought of as computational machines implementing certain functions $f(\mathbf{x})$ of the input image $\mathbf{x}$, and on the high level the goal of computational neuroscience is to understand what these functions $f(\mathbf{x})$ are. The activities of a neuron (i.e. the values of f on a finite set of different inputs $\mathbf{x}$) can be measured experimentally resulting in a dataset of brain recordings. Our goal is to use this dataset to

construct a model approximating the brain responses and by examining the model gain insights into the actual functions implemented by the neurons.

Single-neuron models A traditional approach for constructing predictive neural models is to build a separate model for each individual neuron. Such models typically include a linear filter followed by a pointwise non-linearity and a noise model reflecting the type of noise in the recorded data (often Poisson noise model is best suited for the spike counts data; Fig. 2.4).

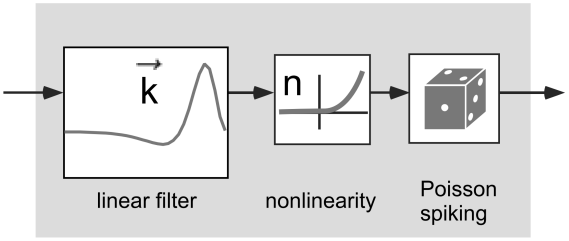


Figure 2.4: An architecture of a single neural predictive model consisting of a linear filter, pointwise non-linearity and a Poisson noise model (LNP model). The figure is adopted from Pillow et al. (2005).

Studying these models with various non-linearities led to the discovery of many interesting aspects of computations in the primary visual cortex (e.g. Adelson and Bergen, 1985; Blakemore and Tobin, 1972; DeAngelis et al., 1994; Hubel and Wiesel, 1959; Morrone et al., 1982). However, in general the range of computations which could be discovered with such single-neuron models is quite limited. One reason for that is that more complex models are necessary for studying the aspects of neural computations which cannot be described by a single non-linearity on top of a linear feature space, however, fitting such more complex models requires larger amounts of data which are challenging to collect. Another reason has to do with the fact that neurons in the primary visual cortex do not operate in isolation from each other but rather they receive information from the same brain areas (retina and LGN), they are interconnected and some of the computations implemented by the brain rely on these inter-neuron dependencies which are not captured by single-neuron models. These issues have recently been addressed with DNN based models.

DNN based models The main idea behind current state-of-the-art models is to use a DNN (typically CNN when working with images) as a feature extractor providing image representations shared across all neurons followed by neuron specific linear computations applied to this shared representation (e.g. Antolík et al., 2016; Cadena et al., 2019). This architecture allows us to simultaneously use the recordings of all neurons for fitting the model thereby increasing the amount of available training data in comparison to single-neuron models. That makes it feasible to use a non-linear CNN feature extractor with many trainable parameters capable of modelling complex neural computations. What is more, fitting all neurons simultaneously allows us to capture the dependencies between these neurons.

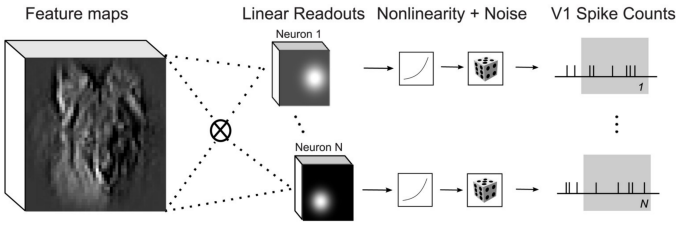


Figure 2.5: An architecture of a DNN based neural predictive model. The DNN feature maps provide a common image representation space for all neurons, on top of which the activity of each neuron is predicted by a linear readout followed by a pointwise non-linearity similar to the LNP model. The figure is adopted from Cadena et al. (2019).

Both the CNN representations and the neuron specific linear computations can be designed to account for specific biological properties of V1 neurons, such as localised receptive fields (Klindt et al., 2017) or orientation selectivity (Ecker et al., 2019), encoding useful inductive biases into the model and thus improving the accuracy of predicting neural responses.

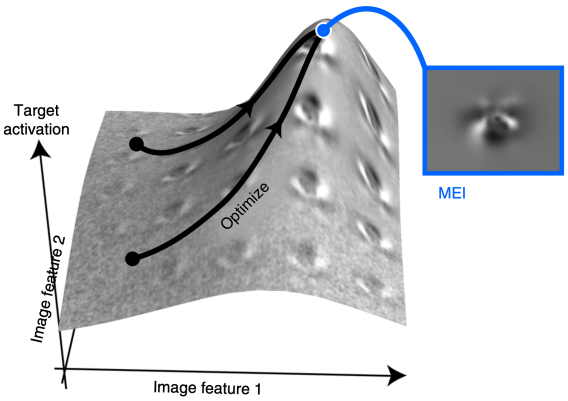


Figure 2.6: An illustration of MEI optimisation. The XY plane corresponds to the dimensions (pixel values) of the input image, the Z axis shows the predicted activity of the neuron. The goal is to find an input image maximising this predicted activity. The figure is adopted from Walker et al. (2019).

Most exciting images (MEIs) We want to use neural predictive models as a proxy for understanding brain computations and one of the main tools for that is visualisation of computations implemented by the model. In case of an LNP-like model, we can use the learnt linear filters as such a visualisation, however, for DNN models there is no such straightforward visualisation available. A common approach allowing us to visualise the computations in a DNN is called most exciting images (MEIs), the idea of which is to find an input image that maximises the predicted activity of a given neuron (Fig. 2.6). The MEIs have been shown to faithfully represent neural computations in the sense that images maximising the model predicted activities also strongly drive the corresponding neurons in the *in vivo* experiment (Bashivan et al., 2019; Walker et al., 2019).

3. What does it take to generate natural textures?

This chapter is based on the following publication:

- Ivan Ustyuzhaninov*, Wieland Brendel*, Leon Gatys and Matthias Bethge (2017). What does it take to generate natural textures? In *International Conference on Learning Representations (ICLR)*.

3.1 Motivation

A CNN based parametric texture model of Gatys et al. (2015) was a major milestone in texture modelling enabling generation of high quality natural textures which are often perceptually indistinguishable from the original photographic texture image. In particular, this model typically generates natural textures of superior quality in comparison to another widely used model of Portilla and Simoncelli (2000) which had been considered state-of-the-art parametric model before the appearance of Gatys et al. (2015). What is interesting is that despite the differences in perceptual quality of generated textures, these two models are conceptually quite similar: they both describe a reference texture image by computing a summary statistic on top of some feature representations of this image.

Gatys et al. (2015) use a VGG-19 CNN (Simonyan and Zisserman, 2015) pre-trained on ImageNet dataset (Russakovsky et al., 2015) to perform object classification as a feature extractor and Gram matrices of feature maps in intermediate layers as summary statistics. Portilla and Simoncelli (2000) develop a linear hierarchical feature extractor based on an overcomplete complex wavelet transform which they refer to as a "steerable pyramid" supplemented by the set of summary statistics adapted to the properties of the wavelet transform.

Therefore, it appears that it is the pre-trained CNN that provides feature representations well-suited for modelling and generating higher quality textures in comparison to a model based on a "steerable pyramid". Our goal in this project is to study why that is the case and what aspects of the CNN are necessary for computing useful representations for texture modelling. Specifically, we are interested in the following questions:

- Does the CNN need to be pre-trained on natural images for generation of high quality natural textures? If that is the case, what kind of pre-training should it be (e.g. should it necessarily be based on a complex visual task such as object classification)?
- What is the role of the depth of the network? The VGG-19 network is quite deep consisting of 19 layers; is such a depth necessary for a good texture model? If

so, is it because the network becomes more nonlinear with depth (since there is a nonlinearity after each layer), or alternatively, could it be that it is not nonlinearity but rather large effective receptive field sizes in the higher layers that are important?

- What is the role of the number of feature maps in each layer (i.e. the width of the network)? Feature maps in a particular layer can be thought of as a decomposition of an input image in some basis, which becomes more complete as the number of feature maps increases potentially providing more detailed descriptions of the reference texture. However, the summary statistics applied to the feature representations complicate the analysis of the effect of the basis dimensionality on the generated textures suggesting an empirical study of this question.

3.2 Results

A single layer model with random filters generates high quality textures This is an interesting result showing that a single layer random filters model provides a strong baseline while being a much simpler feature extractor than a deep CNN such as VGG-19. Moreover, we studied single-layer models with various kinds of filters obtained by unsupervised pre-training (PCA, Fourier basis, etc.) on patches of natural images, and such models often generate perceptually inferior textures to the random filters model. However, despite a random single layer model being a strong baseline, it does not completely match the quality of textures generated using the model of Gatys et al. (2015), meaning that a single layer model is lacking some essential components of a deep CNN which we investigate next.

Representations of different scales improve quality of generated textures We found that including convolutional filters of different sizes in a single layer model improves the quality of generated textures making them perceptually very similar to those generated using a VGG-19 model. Convolutional filters of different sizes in a single layer model replicate the effect of different effective receptive field sizes in intermediate layers of a deep CNN, thus suggesting that least one aspect in which the depth of a CNN model plays a role in texture generation is the computation of multi-scale representations.

Non-linearity is essential for texture generation We found that the quality of textures generated using a single layer model without a pointwise non-linearity after the convolutions (i.e. using a linear model) was substantially inferior to those generated with a non-linearity. The exact type of non-linearity does not seem to play a big role (models with both ReLU and sigmoid generated textures of similar quality) but its presence is necessary. However, since only a single non-linearity seems to be sufficient for generating high quality textures, highly nonlinear computations in deep CNNs do not appear to be essential for a good texture model.

Increasing the number of feature maps improves texture quality We considered single layer models with various types of filters (random or unsupervised pre-training) and studied how changing the number of feature maps affected the quality of generated textures. For all these models we found that increasing the number of feature maps allowed us to increase the quality of generated textures. Since increasing the number of feature maps also increases the number of parameters that need to be matched when generating a new texture, one concern for models with a large number of feature maps could be that all generated textures might be slightly modified copies of the reference texture.

However, we found it is not the case and textures generated with different random initial conditions are clearly different on the pixel level from the reference one.

3.3 Discussion

This study shows that DNN representations necessary for modelling natural textures are quite simple in the sense that they can be computed only with a single convolutional layer with random filters. Interestingly, Gatys et al. (2015) claimed the opposite, i.e. that CNN with random filters do not generate high quality natural textures. That is likely to be a consequence of texture generation using deep CNNs corresponding to a hard optimisation problem, which is illustrated by the fact that for some reference textures a texture generated with a single layer model was a better solution of this problem than the one generated using a deep CNN model (i.e. by actually solving it). This example highlights the difficulty of studying DNNs: they are complex models with many moving parts and making claims about their properties requires extensive experiments.

After publishing this work, there appeared a number of studies on DNNs with random weights (e.g. Giryes et al., 2016; He et al., 2016) highlighting various similarities in behaviour of such models to the trained ones. Perhaps, the most influential of such studies was the one of Ulyanov et al. (2018) who showed that a CNN with random weights can be used as a natural images prior achieving excellent results on a variety of computer vision problems. At a higher level, wide applicability of DNNs with random weights points to a wider problem of the interplay between the DNNs form (architecture) and function, and despite significant progress in recent years this connection is still not fully understood. Developing methods for studying this connection in DNNs might help with understanding a similar problem of relations between anatomy, morphology and functions of neurons in the brain.

4. Towards causal generative scene models via competition of experts

This chapter is based on the following publication:

- Julius von Kügelgen*, Ivan Ustyuzhaninov*, Peter Gehler, Matthias Bethge and Bernhard Schölkopf (2020). Towards causal generative scene models via competition of experts. In *ICLR Workshop "Causal Learning for Decision Making"*.

4.1 Motivation

We have seen in Chapter 3 that DNNs capable of generating high-quality natural textures can be quite simple which is, however, not the case when it comes to modelling natural images in general. For example, deep networks with complex architectures have been state-of-the-art models[1] for ImageNet (Russakovsky et al., 2015) classification for the last decade despite various attempts at achieving comparable performance with simpler shallower networks. Similarly, current state-of-the-art generative models of natural images (e.g. Child, 2021; Karras et al., 2021) are also based on complex deep architectures. While such models can compute image representations suitable for generating high-quality photorealistic images, as it is often the case with DNNs, they are hard to interpret and therefore provide limited control over the specific details of the generated image. The main objective of this project is to develop a DNN based generative model that overcomes this limitation, specifically in terms of control over the placement and appearance of individual objects in the image.

The interest for object representations and control over individual objects in generated images is driven by the importance of object perception for humans. Clearly, when describing a natural image such as a photograph, a person would pay a lot of attention to the objects in the photograph. For example, such a description is likely to provide information as to how many different objects there are, what is their spatial arrangement (foreground/background, occlusions), how these objects look like, etc. Developing generative models with components capable of replicating some aspects of human object perception has lately been an active research topic in generative modelling.

There is a number of neural models aiming at building explicit object-centric representations. Burgess et al. (2019) proposed a model performing unsupervised decomposition of multi-object scenes into the constituent objects. The model operates sequentially and keeps track of already decomposed objects so that out of remaining objects, the most foreground object is always processed at the current step. The architecture of this model includes an attention network which provides a segmentation mask for the foreground

[1] https://paperswithcode.com/sota/image-classification-on-imagenet

object and a VAE reconstructing the object in the segmentation mask. Another model is the one of Greff et al. (2019) which represents an image as a spatial mixture model and decomposes it into the individual objects by performing amortised variational inference. Its extension of Engelcke et al. (2019) enables generation of new images similar to the training data in addition to decomposition into individual objects.

The main shortcoming of these models is that they cannot be used to generate new images with arbitrary arrangements of objects extracted from a reference image, or in other words recombine the decomposed objects into new scenes. In this project we tackle this issue and aim to develop a model capable of:

- decomposing natural images into objects without any supervision;
- computing representations of individual objects allowing us to manipulate these objects independently from each other;
- generating new scenes with arbitrary numbers and compositions of extracted objects.

4.2 Results

Learning by competition of experts Similarly to Burgess et al. (2019), we designed a model sequentially decomposing images into the objects. The main novelty of this model is that for every object category it employs a separate (rather than a single network for all object categories) attention network computing segmentation masks and a VAE capturing the appearance and the shape of the object. We call such an attention-VAE module an expert. During training at each step in the decomposition process all experts reconstruct the object currently in the foreground, but only the parameters of the expert offering the best reconstruction get updated. In other words, experts compete with each other to provide the best reconstruction which makes them specialise on objects of particular categories (the best expert gets reinforced to provide even better reconstructions of a particular object category), which is an idea we adopted from causal inference literature (Parascandolo et al., 2018).

The model decomposes scenes and generalises to novel scenes We found that our model can decompose multi-object scenes into individual objects, including the scenes with partially occluded objects. Moreover, a trained model generalises to scenes with different numbers of objects. For example, we performed an experiment in which every image in the training dataset included exactly three different objects, but after training the model was able to decompose images with arbitrary numbers (including none) of objects of each category.

Specialising experts provide explicit object representations The expert VAEs capture both the appearances and the shapes of objects of the corresponding category, as well as allow sampling new instances of these objects. Therefore our architecture extracts information about every object category in the training dataset and encapsulates it into expert modules which can be used independently from each other in downstream tasks (e.g. provide priors over the extracted objects or generate object samples to be included in a different scene).

The model enables controlled generation of new scenes We can use the specialising experts to create new scenes with arbitrary numbers of objects in arbitrary depth order.

Specifically, the scenes can be generated sequentially layer by layer, such that at every step we draw a sample from an expert corresponding to an object we want to place at the current depth position and put it on top of all objects generated at previous steps.

4.3 Discussion

We proposed a neural architecture capable of computing explicit object representations by using competing expert modules specialising on distinct object categories. However, in practise the training of such a model proved to be challenging, in particular, the competition learning dynamics is prone to suboptimal local minima corresponding to some experts specialising on multiple objects while others not doing anything at all. Another shortcoming of the model is the necessity of choosing in advance the number of experts (i.e. an upper bound on the number of object categories) and the number of steps in the sequential decomposition process (i.e. an upper bound on the total number of objects in the image), both of which significantly influence the computational complexity. Due to these challenges we restricted our experiments to relatively simple synthetic scenes, and the way such a model can be modified to reliably work with more complex images (potentially natural images) remains an open question.

More generally, it seems that the main difficulty of object-centric representations is the inherent discreteness of individual objects. There is a substantial body of work on convenient disentangled representations of continuous factors of variation such as colour or location (e.g. Burgess et al., 2018; Locatello et al., 2019), but extending such representation to include a discrete component encoding the identity of the objects in the scene has proven to be challenging. We approached this issue from a different perspective of not using a joint representation space for all objects but rather a separate expert module for each object, which by construction solves the discreteness problem but introduces new challenges of learning by competition.

In this project we were dealing with static images, however, human perception also relies on various other cues such as relative motion of different objects, for example. Taking advantage of these cues in more complex datasets could help address some of the challenges of object-centric representations. A number of recent studies (e.g. Tangemann et al., 2021; Xu et al., 2019) started exploring these ideas with promising results, however, the datasets they use are still significantly simpler than arbitrary natural scenes.

Overall, there are currently no models capable of computing object-centric representations of arbitrary natural scenes that could allow us to close a significant gap between machine and human visual perception of discrete objects, but it is an active research area with many interesting approaches so it is likely there will be significant advances in the near future.

5. Compositional uncertainty in deep Gaussian processes

This chapter is based on the following publication:

- Ivan Ustyuzhaninov*, Ieva Kazlauskaitė*, Markus Kaiser, Erik Bodin, Neill Campbell and Carl Henrik Ek (2020). Compositional uncertainty in deep Gaussian processes. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 480–489. PMLR.

5.1 Motivation

In the previous chapters we discussed DNN architectures along with the corresponding image representations for texture synthesis and object-centric generative modelling. However, the specific image representations that we have studied might not be the only ones that the DNN could employ for the corresponding tasks, or in other words, the DNN architectures we used might potentially provide other solutions for the same problems. It turns out to be a general phenomenon for multi-layered models, and in this project we discuss it from a more theoretical viewpoint. Specifically, we aim at developing tools for quantifying the range of representations in the intermediate layers consistent with a given input-output function of a deep architecture.

Consider, for example, a very simple two-layer network with one-dimensional inputs and outputs of each layer. Clearly, there are multiple ways for such a network to implement a given function (e.g. the first layer implements an identity function, while the second layer implements the target function; or vice versa) leading to different outputs of the first layer (i.e. the network computes different representations in intermediate layers). The same principle applies to complex DNNs generating images or performing other tasks. Nevertheless, typically only one possible realisation of such DNNs is studied rather than a full range of intermediate representations consistent with the task solved by the DNN. We aim to address this issue by quantifying the range of such representations by means of probabilistic analysis.

Bayesian inference offers a convenient framework for studying the diversity of DNN representations. It requires us to place a prior distribution on the weights of the network, which is updated when the network is fitted to the data to obtain the posterior distribution. This distribution captures the range of network weighs (and therefore the range of the corresponding representations) that are consistent with the prior distribution and the observed data. Such a setting is different from a typical scenario in which the DNN optimisation algorithm (e.g. SGD) provides only a single solution (i.e. a point estimate of network weights) rather than an entire distribution of possible solutions.

However despite its theoretical appeal, Bayesian inference for DNNs is in general computationally intractable due to the extremely high number of weights in such networks. To circumvent this problem to some extent, we focus in this project on a more tractable hierarchical model, namely on deep Gaussian processes (DGPs). DGPs are compositions of multiple layers where each layer is a Gaussian processes (GPs), so their architecture is very similar to that of DNNs and therefore most of the results we obtain in this project could be transferred to DNNs.

Despite Bayesian inference in DGPs being more computationally tractable than in DNNs, the computation of exact posterior distribution is still extremely challenging, and therefore various approximation methods are used. A common approach employed by most DGP inference methods in the literature is to approximate the true posterior distribution with a simpler one which admits efficient inference algorithms. While there is a number of such approximate Bayesian inference methods available for DGPs (e.g. Bui et al., 2016; Dai et al., 2015; Damianou and Lawrence, 2013), they all fall short of providing a posterior distribution capturing the diversity of intermediate input representations.

Building on the work by Salimbeni and Deisenroth (2017), we aim to:

- Understand why the posterior distribution obtained using common DGP approximate inference methods typically concentrates on a single solution rather than captures different intermediate representations consistent with the overall function implemented by the model;
- Propose an approximate Bayesian inference method solving the above problem.

5.2 Results

Dependencies between the layers are crucial A common approach in the literature is to use DGP approximate posterior distributions which are factorised over the layers (i.e. different layers are marginally independent in the approximate posterior). We showed with an analytical argument and with numerical experiments that such an approximation leads to the posterior distribution being concentrated on a single solution rather than capturing the diversity the possible solutions. This fact is easy to see by revisiting an example of a two-layer network from the previous section: if the composition consisting of the two layers is set to implement a given function, the choice of the first layer must completely determine the function implemented by the second layer thus illustrating the dependence between the layers.

Approximate posterior distribution captures the range of intermediate representations We designed two approximate Bayesian inference methods for DGPs that account for the dependencies between the layers. The first one is based on approximating the posterior with a multivariate Gaussian with an inter-layer correlation structure defined by a tri-diagonal precision matrix (i.e. each layer conditionally depends only on neighbouring layers). That provided a trade-off between the computational tractability in terms of a closed-form optimisation objective and explicit inclusion of dependencies between the DGP layers.

The second method uses an approximate posterior distribution parametrised by a factorised Gaussian over the outputs of hidden layers on a small subset on input values which are treated as hyper-parameters. The idea of this approach is to model the distribution over the input-output pairs of each layer such that the output of a previous layer is

the input to the next one. This connection between inputs and outputs of layers induces an approximate posterior distribution with dependencies across the layers despite the underlying parametrisation being factorised. In comparison to the Gaussian posterior across the layers mentioned above, this approach allows us to capture more variance in intermediate layers and hence more diversity of intermediate representations. However, the disadvantage in this case is that the optimisation objective is no longer closed-form but requires estimating certain expectations by sampling.

Models accounting for the diversity of intermediate representations achieve higher marginal likelihoods Different Bayesian models can be compared by evaluating their marginal likelihoods on the same dataset (Bishop, 2006). We evaluated DGP models corresponding to the inference method of Salimbeni and Deisenroth (2017) and the two inference methods proposed above, and found that accounting for the diversity of intermediate representations allows us to achieve higher marginal likelihood values. This result provides a quantitative evidence that multi-layer models capturing the diversity of intermediate representation provide a better explanation of the data than those collapsing to a single possible solution.

5.3 Discussion

We used Bayesian inference to quantify the diversity of intermediate representations computed by a multi-layer model which arises from multiple possible solutions explaining the data. This is a relatively uncommon viewpoint on Bayesian deep learning which often focuses on uncertainty of predictions rather than on intermediate representations, however, it seems that this problem starts getting more attention in the literature. For example, concurrently with our work Ober and Aitchison (2021) developed similar ideas and applied them to both DGPs and DNNs also showing that approximate posteriors capturing the diversity of intermediate representations achieve higher likelihoods. In the next chapter we will discuss how such methods could be relevant for a specific DNN application.

Overall, we showed that Bayesian framework allows us to quantify the diversity of intermediate representations in multi-layer models as well as found the main requirement that the corresponding approximate posterior has to satisfy: it must include dependencies between the layers. However, our experiments were limited to relatively simple synthetic datasets and scaling them to real-world data seem to be highly non-trivial.

One problem is the computational complexity as including inter-layer dependencies in the approximate posterior means fewer computations available in closed form but rather requiring computationally expensive numerical approximations. Another one is that we used Gaussian approximate posterior distributions which have many advantages from a computational viewpoint, but sufficiently capturing the complexity of the true posterior in real-world problems would likely require more complex approximate posteriors as well. For example, approximate posteriors based on implicit distributions (Titsias and Ruiz, 2019) could offer an interesting direction for future research.

6. Rotation-invariant clustering of neuronal responses in primary visual cortex

This chapter is based on the following publication:

- Ivan Ustyuzhaninov, Santiago A. Cadena, Emmanouil Froudarakis, Paul G. Fahey, Edgar Y. Walker, Erick Cobos, Jacob Reimer, Fabian H. Sinz, Andreas S. Tolias, Matthias Bethge and Alexander S. Ecker (2020). Rotation-invariant clustering of neuronal responses in primary visual cortex. In *International Conference on Learning Representations (ICLR)*.

6.1 Motivation

In this project we focus on using DNN representations to understand computations in the primary visual cortex (V1) of the mouse brain. This brain area contains around 500 thousand neurons (Herculano-Houzel et al., 2013) each of which implements its own computation (also called "function"). We are interested in analysing these computations on the population level, i.e. studying the common patterns of functions, how different neurons relate to each other, etc. However, the diversity of neural functions arising from such a large number of neurons makes answering these questions highly challenging. We tackle this problem by developing a method for computing the functional similarities between the neurons which would allow us to reduce the data complexity from 500 thousand individual neurons to a few dozen groups of functionally similar neurons facilitating the population level analysis.

This idea of using functional similarities between the neurons is motivated by the concept of functional cell types, a relatively small number (typically a few dozen) of different computations such that every neuron can be described by one of them. It has been shown that mouse retina can indeed be described in terms of functional cell types (Baden et al., 2016); whether that is the case as well for the primary visual cortex is currently unknown. Baden et al. (2016) simultaneously recorded responses of a large population of retinal ganglion cells to the same stimulus and clustered these recorded responses. While for the retina it was sufficient to use a few relatively simple synthetic stimuli, the relevant stimuli for studying the V1 are natural images and the number of stimuli presented in the experiment must be quite large (at least a few thousand) to reasonably sample the huge space of possible natural images. That leads to very high-dimensional response vectors making it intractable to work directly with raw responses prompting the use of predictive models offering low-dimensional representations of neural functions, in particular DNN based models.

DNN models applied to neural recordings have been recently extensively studied in the literature and shown to achieve state-of-the-art predictive performance. Such models are typically fitted using a large-scale dataset of neural responses to natural images, they contain a CNN core shared across all neurons in the dataset and a separate linear readout for every neuron (Antolík et al., 2016). The model architecture can additionally encode prior knowledge about the V1 neurons to facilitate the training. For example, readouts can be factorised into spatial masks and feature weights vectors with the spatial masks accounting for neurons having different locations of their receptive fields (Klindt et al., 2017). Another example is the inclusion of differently rotated copies of each convolutional filter in the CNN core resulting in a so-called rotation-equivariant core which explicitly encodes an assumption that each neuron has its own preferred orientation (Ecker et al., 2019).

We use feature weights vectors in the factorised readout of the rotation-equivariant model of Ecker et al. (2019) as low-dimensional functional representations of every neuron. Using the distance between these vectors as a measure of functional similarity allows us to compute groups of functionally similar neurons. However, we want to assign the neurons differing only in receptive field locations or preferred orientations to the same group as these two properties are inherently specific to individual neurons and are generally irrelevant for population level analysis. By construction the readout feature vectors do not contain information about the receptive field locations, but it is not the case for the preferred orientations. Thereby in this project our goals are to:

- Propose a method for removing information about preferred orientation of neurons from their readout feature vectors resulting in so-called aligned feature vectors;
- Group neurons based on similarities of their aligned feature vectors and assess the functional similarities of neurons assigned to the same group.

6.2 Results

Aligning readout feature vectors by cycling shifts removes orientation information The rotation-equivariant CNN core implies that two neurons which implement the identical computations with different preferred orientations should have identical readout feature vectors up to cycling shifts of their subcomponents. Using this property we proposed an algorithm which aligns readout feature vectors by cyclically shifting them thus reducing the feature vectors of neurons differing only in preferred orientations to the same vector. To enable gradient-based optimisation of such an alignment we developed a continuous approximation of cycling shifts. We validated the proposed alignment methods using numerical experiments with various kinds of synthetic data.

Clustering the aligned readout vectors reveals groups of functionally similar neurons We clustered the aligned readout feature vectors obtained from the DNN model fitted to a dataset containing more than 6 thousand mouse V1 neurons. These clusters contained functionally similar neurons which we verified by observing (1) that maximally activating images (MEIs) of neurons within the cluster were visually similar but differed across the clusters; (2) the confusion matrix showing responses of every neuron to an MEI of every other neuron exhibited a block-diagonal structure corresponding to the obtained clusters; (3) the 2D t-SNE embedding of the aligned readout feature vectors revealed multiple areas of higher density which generally coincided with the obtained clusters. Moreover, while the MEIs of neurons in the same cluster were visually similar, they clearly showed

variability in spatial locations and orientations thus suggesting that such a clustering is invariant to these two properties.

6.3 Discussion

We proposed a method for computing low-dimensional representations of neural functions which are independent of receptive fields spatial locations and preferred orientations. Clustering such representations we obtained groups of functionally similar neurons which can be thought of as hypotheses of potential cell types that could be studied in future work. We further address this question in the next chapter.

One problem with this approach was the fact that estimating the right number of clusters when clustering the readout feature weights was challenging. We found that the DNN representations were redundant in the sense that the same neural function could be encoded in multiple different ways such that different (i.e. not related by a cycling shift) readout vectors correspond to the same neural function. Because of that, the standard methods of choosing the number of clusters (e.g. evaluating test log-likelihood for different numbers of clusters) overestimated it resulting in too granular clustering, requiring us to manually merge some clusters as a post-processing step. This is a closely related problem to the one discussed in Chapter 5, and developing methods for obtaining a posterior distribution over the readout feature vectors rather than a point estimate could potentially allow us to identify the range of feature vectors corresponding to the same function and therefore avoid a manual post-processing step.

In this work we first fitted a DNN to the neural data, and afterwards we aligned and clustered the readout vectors in this DNN. A potential extension could focus on combining these two steps into a single end-to-end trainable model which learns functional representations and assigns neurons to clusters at the same time. For example, using a mixture of Poisson distributions as the noise model might enable such end-to-end learning. Another potential direction for future work is studying the possibility of further factorisation of the readouts in which the feature vectors are decomposed into the orientation independent component describing the predictive DNN features for a given neurons and an explicit preferred orientation of the neuron. Such a factorisation would allow us to avoid aligning the feature vectors thus saving computation and not introducing approximations related to continuous cycling shifts.

7. Digital twin reveals combinatorial code of non-linear computations in mouse primary visual cortex

This chapter is based on the following publication:

- Ivan Ustyuzhaninov, Max F. Burg, Santiago A. Cadena, Jiakun Fu, Taliah Muhammad, Kayla Ponder, Emmanouil Froudarakis, Zhiwei Ding, Matthias Bethge, Andreas S. Tolias and Alexander S. Ecker (2022). Digital twin reveals combinatorial code of non-linear computations in the mouse primary visual cortex. *In Submission*.

7.1 Motivation

In Chapter 6 we developed a method for computing functional representations of neurons in the mouse primary visual cortex (V1) which capture all aspects of neural functions apart from receptive field locations and preferred orientations. In this project we apply this method to a large scale neural dataset to gain insight into the biological aspects of functional organisation of the mouse V1.

The first question we aim to tackle concerns the existence of discrete functional cell types. We have seen in Chapter 6 that clustering the DNN neural functional representations revealed groups of similar neurons which could be potential cell types. On the other hand, the existence of clusters of functionally similar neurons does not contradict the hypothesis of a continuum of neural functions as we would still expect the clustering to produce the groups of functionally similar neurons since by construction the neighbouring neurons in the representation space are functionally similar. In this project our goal is to examine these two possibilities in more detail.

Another question we are interested in deals with describing the groups of functionally similar neurons. Regardless of whether they are discrete cell types or not, these groups provide an overview of computations in the mouse V1 and we would like to study their biological implications. However, the DNN functional representations are abstract and it is not clear what mechanisms differentiate the clusters. We plan to tackle this question by using the fact that the DNN model that we fitted to obtain the neural functional representations can be thought of as a "digital twin" of the V1 in the sense that it predicts neural responses for arbitrary input stimuli. Such a model allows us to replicate classical *in vivo* electrophysiological experiments (e.g. Adelson and Bergen, 1985; Blakemore and Tobin, 1972; DeAngelis et al., 1994; Hubel and Wiesel, 1959; Morrone et al., 1982) *in silico* using virtually unlimited number of input stimuli. The results of these experiments would allow us to describe the groups of similar neurons in terms of computational mechanisms

(e.g. surround suppression, phase invariance, etc.) established in the literature.

These *in silico* experiments are interesting not only as a tool for describing the functional clusters. In the literature such experiments are typically discussed independently of each other since in the *in vivo* setting the experimental time and therefore the amount of stimuli is limited making it infeasible to conduct multiple experiments during the same recording session. In the *in silico* setting, however, we are not limited by the number of stimuli and can conduct multiple experiments on the same population of neurons. That enables us to study the dependencies between different computational mechanisms described by these experiments which is an interesting question in itself and also provides additional insight into the computations in the mouse V1.

7.2 Results

Mouse V1 is described by a non-uniform continuum of functions We analysed the functional low-dimensional representations of more than 13 thousand V1 neurons recorded in 7 different mice. The 2D t-SNE embeddings of these representations revealed that the mouse primary visual cortex appears to be organised as a non-uniform continuum of functions with around 30 modes (high density areas) corresponding to common computations; these modes generally correspond to the clusters we discussed in Chapter 6. By extending the MEIs to represent the computations of the entire clusters rather than individual neurons, we found that neighbouring clusters in the functional representation space have similar MEIs suggesting a continuous structure of functions in that space.

***In silico* experiments visualise cluster computations** We computed the MEIs, the optimal Gabors and the optimal differences of Gaussians (DoG) stimuli for every neuron and cluster, and used these stimuli to conduct *in silico* experiments. These experiments were designed to probe the orientation tuning, phase invariance, surround suppression and cross-orientation inhibition of individual neurons as well as entire clusters. Based on the tuning strengths with respect to these experiments, we were able to differentiate most of the clusters thus the *in silico* experiments provided a compact and interpretable way to describe the functional clusters.

Non-linear computations are independent of each other Further analysis of the results of *in silico* experiments revealed that phase invariance, surround suppression and cross-orientation inhibitions appear to be expressed independently of each other. Specifically, we found that tuning strengths of different clusters were uncorrelated with respect to these non-linear properties. Moreover, considering each cluster as either strongly or weakly tuned with respect to these non-linear properties, we found clusters with every possible combination of low/high tuning of these three properties; this might be a consequence of statistical independence of these non-linear computations. This result sheds light on the structure of computations in the mouse V1 suggesting that it might employ combinatorial code in the space of independent non-linearities as a means for information processing.

***In silico* tuning curves are reasonable approximations of *in vivo* ones** Our analysis is based on the *in silico* experiments, therefore, it is crucial to make sure that they offer a faithful representation of the computations in V1. In particular, there might be an issue arising from different kinds of stimuli used for training the model and for the *in silico*

experiments since we train the model using natural images, however, we use Gabors, DoGs or MEIs for *in silico* experiments.

To compare the *in vivo* and *in silico* tuning curves we recorded two scans covering the same neurons in a mouse not used for the main analysis. In one scan we recorded neural responses to natural images and in the other one we recorded neural responses to Gabor stimuli used for the surround-suppression *in silico* experiment. Such a setting allowed us to use the first scan to fit the DNN model, use the fitted model to compute the *in silico* tuning curves and directly compare them to the *in vivo* tuning curves recorded in the second scan. We found that the correlation between *in silico* and *in vivo* tuning curves was very similar to the DNN performance on predicting neural responses to natural images (i.e. the task it was trained on) thus suggesting that the predictive performance of the DNN model does not deteriorate when it is used to predict neural responses to *in silico* stimuli.

7.3 Discussion

This project is an example of using a DNN model to gain insight into the organisation of the brain. We addressed biological questions of functional organisation of a mouse primary visual cortex by studying its "digital twin" provided by the DNN model. This is a relatively new approach but it clearly has a potential for becoming one of the main instruments in the computational neuroscience toolbox.

We found that the mouse primary visual cortex is functionally organised as a non-uniform continuum which is consistent with other studies into the functional organisation of a mouse neocortex (Gouwens et al., 2020; Scala et al., 2021) but qualitatively different from the retinal ganglion cells being organised into discrete cell types (Baden et al., 2016). That raises an interesting question for future research regarding the functional organisation of the neural pathway from retina to V1 (especially LGN) aimed at understanding how the discrete retinal cell types get transformed into the continuum of V1 functions and what might be the biological reasons for that.

Another future research direction could be based on the connectomics data (e.g. Bae et al., 2021) which could allow us to study the patterns of connectivity between the V1 functional clusters as well as their projections to higher visual areas (HVAs). That might reveal if neurons within the same V1 functional cluster project to the same HVA and in general shed light on the functional organisation of HVAs which is currently unknown. Considering the discrete cell types in the retina and non-uniform continuum in V1, we might speculate that HVA functions could be even more continuous than in V1, providing a hypothesis to investigate in future research.

Future research of functional clusters connectivity to HVAs might also provide an insight into the mechanisms behind the combinatorial code of non-linear computations that we found using the *in silico* experiments. Our result suggesting the existence of such a combinatorial code in V1 is one of the few currently available results regarding the V1 computations on the population level, however, its implications are still unclear. One hypothesis might be that this combinatorial code spans a basis of non-linear computations corresponding to different functions in the downstream processing leading to specialisations of the HVAs.

8. Discussion

Throughout this thesis we discussed various properties and applications of image representations in DNNs. In this chapter we bring these discussions together and summarise the main points we learnt.

8.1 What have we learnt about DNN representations?

We have discussed a few quite different problems ranging from generating textures to predicting brain responses, and DNNs (especially CNNs) were the main tool we used to tackle them. On the one hand, it is quite remarkable that such conceptually simple architectures (albeit requiring a lot of computational power) produce image representations applicable to a variety of visual tasks and thereby operate in ways resembling the biological vision. On the other hand, these representations in some sense move the problem we aim to solve from the pixel space to abstract representation spaces but conceptually we are still confronted with the same issues as we would be in the pixel space.

Consider, for example, the texture generation project in Chapter 3 in which the key idea is to use summary statistics on top of the DNN representations. Here, using the DNN representation space is not absolutely necessary as it is also possible to use the summary statistics in the pixel space. The generated textures would be of inferior quality in that case but it would work in principle. We relied on the background knowledge that textures are stationary and invariant under certain transformations to *remove* the unnecessary information from the DNN representations by means of summary statistics rather than directly *infer* the relevant texture information from these representations. The crucial point is that our a priori understanding of the structure of texture images does not have anything to do with the DNNs.

In contrast, the next project of building object-centric representations (Chapter 4) proved to be much more challenging than texture generation for that very reason that we do not have an analogue of summary statistics for the objects. In particular, we do not have a good quantitative description of an object in a natural image, which we could apply to DNN representations. While we know that DNN representations contain information about objects (e.g. because they perform very well on object detection and classification), we do have good tools for neither *removing* the irrelevant parts of the image representations by relying on background knowledge about the objects nor for *inferring* the relevant information for object modelling by identifying what makes object an object directly from the DNN representations.

Another example of using the background knowledge in the DNN representations is the project in Chapter 6. We factorised the linear readouts to account for the spatially localised receptive fields of the V1 neurons and applied the readouts in the DNN rep-

resentation space. But again, in principle we could use the same approach even in the pixel space. Summarising these arguments, we can argue that currently we use DNN representations as a leverage to gain better results from exploiting the problem structure that we know in advance, rather than inferring the problem structure from them.

What is more, even if we had tools for inferring the relevant problem structure from the DNN representations (e.g. parts of representations corresponding to individual objects in the image), there is a more conceptual problem that we discussed in Chapter 5. The results of that project show that DNN representations are not unique, so the same object in the natural image might be represented in multiple different ways and working with such representations could be quite challenging. There might be a way to constrain the DNNs to have mostly unique representations (e.g. by strong regularisation) but it is quite likely that such constraints would have a negative impact on the model performance, and in general it seems that a capacity for over-representation is something that contributes to the remarkable performance of DNNs on a variety of tasks.

Our goal was understanding the image representations in DNNs, and summarising the above, we must admit that we are still quite far from understating DNNs sufficiently well despite the advances on solving specific visual tasks. On a positive note, currently there is a lot of interest in DNN interpretability and studying intermediate representations, so it is reasonable to expect advancements in this area in the coming years.

8.2 Digital twins

In Chapter 7 we used a DNN model of the mouse primary visual cortex as a "digital twin" of this brain area. Such an approach is a shift in perspective in comparison to other projects in this thesis in the sense that we did not attempt to understand what computations are encoded in DNN representations but rather treated the model as a "black box" only studying its outputs and not its internal structure. This is particularly well suited to computational neuroscience because it provides a large toolkit of experimental methods that could be applied to the "digital twin" virtually without any limitations on the number of experiments that could be conducted.

The "digital twin" approach could be applicable to a much wider range of problems than studying the brain, potentially allowing us if not solve but at least circumvent some of the issues discussed in the previous section. For example, revisiting an idea of the project in Chapter 4, we can consider how a "digital twin" might help us manipulate individual objects in the images:

- Having a generative model of natural images (e.g. a VAE) we can experimentally estimate changes of image representations corresponding to transformations of interest in the input images (e.g. changes in the number or locations of objects);
- This could be done by performing *in silico* experiments (similarly to those in Chapter 7) with the stimuli being sequences of natural images that differ from each other according to the relevant transformations (e.g. one of the objects changes its position across the sequences while other objects are stationary; or the number of objects changes across the sequence);
- After estimating how the image representations change, we can proceed similarly to texture generation in Chapter 3: apply the estimated changes to the representation of the reference image and solve the inverse problem of searching for the new input image corresponding to the modified representations.

This example is of course only a general outline of how digital twins might be used beyond the application in Chapter 7 rather than a detailed action plan, but hopefully it illustrates our argument that "digital twin" methodology might be a promising way for tackling visual problems with DNNs.

8.3 Conclusion

Modern computational vision models become increasingly similar to biological vision, both in terms of what they can achieve but also in terms of challenges of working with such models. Perhaps we can argue that we are close to the point when we will be talking about vision research in general without division into artificial and biological vision as the problems and methods of these two subdomains complement each other as we have seen throughout this thesis.

We studied DNN representations in different contexts but nevertheless only scratched the surface of the interplay of machine and biological vision. We found common patterns and challenges, but unfortunately have not developed universal recipes for how to build a model computing useful representations for a given problem. I hope (at least in part) it is because it is a genuinely difficult problem and future research will bring new ideas and solutions of the problems we have not managed to fully solve.

Bibliography

Edward H. Adelson and James R. Bergen (1985). Spatiotemporal energy models for the perception of motion. *J. Opt. Soc. Am. A*, 2(2):284–299.

Ján Antolík, Sonja B Hofer, James A Bednar and Thomas D Mrsic-Flogel (2016). Model constrained by visual hierarchy improves prediction of neural responses to natural scenes. *PLoS computational biology*, 12(6):e1004927.

Tom Baden, Philipp Berens, Katrin Franke, Miroslav Román Rosón, Matthias Bethge and Thomas Euler (2016). The functional diversity of retinal ganglion cells in the mouse. *Nature*, 529(7586):345–350.

J Alexander Bae, Mahaly Baptiste, Agnes L Bodor, Derrick Brittain, JoAnn Buchanan, Daniel J Bumbarger, Manuel A Castro, Brendan Celii, Erick Cobos, Forrest Collman et al. (2021). Functional connectomics spanning multiple areas of mouse visual cortex. *bioRxiv*.

Pouya Bashivan, Kohitij Kar and James J DiCarlo (2019). Neural population control via deep image synthesis. *Science*, 364(6439).

Christopher M Bishop (2006). *Pattern Recognition and Machine Learning*. Springer.

Colin Blakemore and Elisabeth A Tobin (1972). Lateral inhibition between orientation detectors in the cat's visual cortex. *Experimental brain research*, 15(4):439–440.

Léon Bottou (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer.

Wieland Brendel, Jonas Rauber, Matthias Kümmerer, Ivan Ustyuzhaninov and Matthias Bethge (2019). Accurate, reliable and fast robustness evaluation. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32.

Thang Bui, Daniel Hernández-Lobato, Jose Hernandez-Lobato, Yingzhen Li and Richard Turner (2016). Deep gaussian processes for regression using approximate expectation propagation. In *International conference on machine learning*, pages 1472–1481. PMLR.

Christopher Burgess, Loic Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matt Botvinick and Alexander Lerchner (2019). Monet: Unsupervised scene decomposition and representation. *arXiv preprint arXiv:1901.11390*.

Christopher P Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins and Alexander Lerchner (2018). Understanding disentangling in β-vae. *arXiv preprint arXiv:1804.03599*.

Santiago A Cadena, George H Denfield, Edgar Y Walker, Leon A Gatys, Andreas S Tolias, Matthias Bethge and Alexander S Ecker (2019). Deep convolutional models improve

predictions of macaque v1 responses to natural images. *PLoS computational biology*, 15(4):e1006897.

Rewon Child (2021). Very deep VAEs generalize autoregressive models and can outperform them on images. In *International Conference on Learning Representations*.

George Cybenko (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314.

Zhenwen Dai, Andreas Damianou, Javier González and Neil Lawrence (2015). Variational auto-encoded deep gaussian processes. *arXiv preprint arXiv:1511.06455*.

Andreas Damianou and Neil D Lawrence (2013). Deep gaussian processes. In *Artificial intelligence and statistics*, pages 207–215. PMLR.

Gregory C DeAngelis, Ralph D Freeman and Izumi Ohzawa (1994). Length and width tuning of neurons in the cat's primary visual cortex. *Journal of neurophysiology*, 71(1):347–374.

Alexander S. Ecker, Fabian H. Sinz, Emmanouil Froudarakis, Paul G. Fahey, Santiago A. Cadena, Edgar Y. Walker, Erick Cobos, Jacob Reimer, Andreas S. Tolias and Matthias Bethge (2019). A rotation-equivariant convolutional neural network model of primary visual cortex. In *International Conference on Learning Representations (ICLR)*.

Alexei Efros and Thomas Leung (1999). Texture synthesis by non-parametric sampling. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1033–1038. IEEE.

Martin Engelcke, Adam Kosiorek, Oiwi Parker Jones and Ingmar Posner (2019). Genesis: Generative scene inference and sampling with object-centric latent representations. *arXiv preprint arXiv:1907.13052*.

David Field (1987). Relations between the statistics of natural images and the response properties of cortical cells. *Journal of the Optical Society of America A*, 4(12):2379–2394.

Kunihiko Fukushima and Sei Miyake (1982). Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pages 267–285. Springer.

Leon Gatys, Alexander Ecker and Matthias Bethge (2015). Texture synthesis using convolutional neural networks. *Advances in neural information processing systems*, 28:262–270.

Raja Giryes, Guillermo Sapiro and Alex M Bronstein (2016). Deep neural networks with random gaussian weights: A universal classification strategy? *IEEE Transactions on Signal Processing*, 64(13):3444–3457.

Xavier Glorot, Antoine Bordes and Yoshua Bengio (2011). Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323. JMLR Workshop and Conference Proceedings.

Ian Goodfellow, Yoshua Bengio and Aaron Courville (2016). *Deep learning*. MIT Press.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville and Yoshua Bengio (2014). Generative adversarial nets. *Advances in neural information processing systems*, 27.

Nathan W Gouwens, Staci A Sorensen, Fahimeh Baftizadeh, Agata Budzillo, Brian R Lee, Tim Jarsky, Lauren Alfiler, Katherine Baker, Eliza Barkan, Kyla Berry et al. (2020).

Integrated morphoelectric and transcriptomic classification of cortical gabaergic cells. *Cell*, 183(4):935–953.

Klaus Greff, Raphaël Lopez Kaufman, Rishabh Kabra, Nick Watters, Christopher Burgess, Daniel Zoran, Loic Matthey, Matthew Botvinick and Alexander Lerchner (2019). Multi-object representation learning with iterative variational inference. In *International Conference on Machine Learning*, pages 2424–2433. PMLR.

Kun He, Yan Wang and John Hopcroft (2016). A powerful generative model using random weights for the deep image representation. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 29.

David Heeger and James Bergen (1995). Pyramid-based texture analysis/synthesis. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 229–238.

Suzana Herculano-Houzel, Charles R Watson and George Paxinos (2013). Distribution of neurons in functional areas of the mouse cerebral cortex reveals quantitatively different cortical zones. *Frontiers in Neuroanatomy*, 7:35.

Matthew D Hoffman, David M Blei, Chong Wang and John Paisley (2013). Stochastic variational inference. *Journal of Machine Learning Research*, 14(5).

Kurt Hornik (1991). Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257.

David H Hubel and Torsten N Wiesel (1959). Receptive fields of single neurones in the cat's striate cortex. *The Journal of Physiology*, 148(3):574–591.

Bela Julesz (1962). Visual pattern discrimination. *IRE transactions on Information Theory*, 8(2):84–92.

Bela Julesz (1981). Textons, the elements of texture perception, and their interactions. *Nature*, 290(5802):91–97.

Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen and Timo Aila (2021). Alias-free generative adversarial networks. In *Advances in Neural Information Processing Systems*.

Diederik Kingma and Jimmy Ba (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Diederik Kingma and Max Welling (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

David A Klindt, Alexander S Ecker, Thomas Euler and Matthias Bethge (2017). Neural system identification for large populations separating what and where. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 3509–3519.

Yann LeCun, Yoshua Bengio and Geoffrey Hinton (2015). Deep learning. *Nature*, 521(7553):436–444.

Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, Bernhard Schölkopf and Olivier Bachem (2019). Challenging common assumptions in the unsupervised learning of disentangled representations. In *International Conference on Machine Learning*, pages 4114–4124. PMLR.

David JC MacKay (1995). Bayesian neural networks and density networks. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 354(1):73–80.

Claudio Michaelis, Ivan Ustyuzhaninov, Matthias Bethge and Alexander S Ecker (2018). One-shot instance segmentation. *arXiv preprint arXiv:1811.11507*.

M Concetta Morrone, DC Burr and Lamberto Maffei (1982). Functional implications of cross-orientation inhibition of cortical visual cells. i. neurophysiological evidence. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 216(1204):335–354.

Sebastian W Ober and Laurence Aitchison (2021). Global inducing point variational posteriors for Bayesian neural networks and deep Gaussian processes. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8248–8259. PMLR.

Giambattista Parascandolo, Niki Kilbertus, Mateo Rojas-Carulla and Bernhard Schölkopf (2018). Learning independent causal mechanisms. In *International Conference on Machine Learning*, pages 4036–4044. PMLR.

Jonathan W Pillow, Liam Paninski, Valerie J Uzzell, Eero P Simoncelli and EJ Chichilnisky (2005). Prediction and decoding of retinal ganglion cell responses with a probabilistic spiking model. *Journal of Neuroscience*, 25(47):11003–11013.

Javier Portilla and Eero Simoncelli (2000). A parametric texture model based on joint statistics of complex wavelet coefficients. *International journal of computer vision*, 40(1):49–70.

David Rumelhart, Geoffrey Hinton and Ronald Williams (1986). Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein et al. (2015). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252.

Hugh Salimbeni and Marc Deisenroth (2017). Doubly stochastic variational inference for deep gaussian processes. In *Advances in Neural Information Processing Systems*, volume 30.

Federico Scala, Dmitry Kobak, Matteo Bernabucci, Yves Bernaerts, Cathryn René Cadwell, Jesus Ramon Castro, Leonard Hartmanis, Xiaolong Jiang, Sophie Laturnus, Elanine Miranda et al. (2021). Phenotypic variation of transcriptomic cell types in mouse motor cortex. *Nature*, 598(7879):144–150.

Karen Simonyan and Andrew Zisserman (2015). Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*.

Matthias Tangemann, Steffen Schneider, Julius Kügelgen, von, Francesco Locatello, Peter Gehler, Thomas Brox, Matthias Kümmerer, Matthias Bethge and Bernhard Schölkopf (2021). Unsupervised object learning via common fate. *arXiv preprint arXiv:2110.06562*.

Michalis K Titsias and Francisco Ruiz (2019). Unbiased implicit variational inference. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 167–176. PMLR.

Dmitry Ulyanov, Andrea Vedaldi and Victor Lempitsky (2018). Deep image prior. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9446–9454.

Ivan Ustyuzhaninov*, Wieland Brendel*, Leon Gatys and Matthias Bethge (2017). What does it take to generate natural textures? In *International Conference on Learning Representations (ICLR)*.

Ivan Ustyuzhaninov, Max F. Burg, Santiago A. Cadena, Jiakun Fu, Taliah Muhammad, Kayla Ponder, Emmanouil Froudarakis, Zhiwei Ding, Matthias Bethge, Andreas S. Tolias and Alexander S. Ecker (2022). Digital twin reveals combinatorial code of non-linear computations in the mouse primary visual cortex. *In Submission*.

Ivan Ustyuzhaninov, Santiago A. Cadena, Emmanouil Froudarakis, Paul G. Fahey, Edgar Y. Walker, Erick Cobos, Jacob Reimer, Fabian H. Sinz, Andreas S. Tolias, Matthias Bethge and Alexander S. Ecker (2020). Rotation-invariant clustering of neuronal responses in primary visual cortex. In *International Conference on Learning Representations (ICLR)*.

Ivan Ustyuzhaninov*, Ieva Kazlauskaitė*, Carl Henrik Ek and Neill Campbell (2018). Sequence alignment with Dirichlet process mixtures. In *NeurIPS Workshop "All of Bayesian Nonparametrics"*.

Ivan Ustyuzhaninov*, Ieva Kazlauskaitė*, Carl Henrik Ek and Neill Campbell (2020). Monotonic Gaussian process flows. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 3057–3067. PMLR.

Ivan Ustyuzhaninov*, Ieva Kazlauskaitė*, Markus Kaiser, Erik Bodin, Neill Campbell and Carl Henrik Ek (2020). Compositional uncertainty in deep Gaussian processes. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 480–489. PMLR.

Ivan Ustyuzhaninov, Claudio Michaelis, Wieland Brendel and Matthias Bethge (2018). One-shot texture segmentation. *arXiv preprint arXiv:1807.02654*.

Julius von Kügelgen*, Ivan Ustyuzhaninov*, Peter Gehler, Matthias Bethge and Bernhard Schölkopf (2020). Towards causal generative scene models via competition of experts. In *ICLR Workshop "Causal Learning for Decision Making"*.

Edgar Y Walker, Fabian H Sinz, Erick Cobos, Taliah Muhammad, Emmanouil Froudarakis, Paul G Fahey, Alexander S Ecker, Jacob Reimer, Xaq Pitkow and Andreas S Tolias (2019). Inception loops discover what excites neurons most using deep predictive models. *Nature neuroscience*, 22(12):2060–2065.

Christopher Williams and Carl Edward Rasmussen (2006). *Gaussian processes for machine learning*. MIT press.

Zhenjia Xu, Zhijian Liu, Chen Sun, Kevin Murphy, William T Freeman, Joshua B Tenenbaum and Jiajun Wu (2019). Unsupervised discovery of parts, structure, and dynamics. *arXiv preprint arXiv:1903.05136*.

Appendix

This chapter contains the publications discussed in Chapters 3–7.

WHAT DOES IT TAKE TO GENERATE NATURAL TEXTURES?

Ivan Ustyuzhaninov[*,1,2,3], **Wieland Brendel**[*,1,2], **Leon Gatys**[1,2,3], **Matthias Bethge**[1,2,3,4]

[*]contributed equally
[1]Centre for Integrative Neuroscience, University of Tübingen, Germany
[2]Bernstein Center for Computational Neuroscience, Tübingen, Germany
[3]Graduate School of Neural Information Processing, University of Tübingen, Germany
[4]Max Planck Institute for Biological Cybernetics, Tübingen, Germany

first.last@bethgelab.org

ABSTRACT

Natural image generation is currently one of the most actively explored fields in Deep Learning. Many approaches, e.g. for state-of-the-art artistic style transfer or natural texture synthesis, rely on the statistics of hierarchical representations in supervisedly trained deep neural networks. It is, however, unclear what aspects of this feature representation are crucial for natural image generation: is it the depth, the pooling or the training of the features on natural images? We here address this question for the task of natural texture synthesis and show that none of the above aspects are indispensable. Instead, we demonstrate that natural textures of high perceptual quality can be generated from networks with only a single layer, no pooling and random filters.

1 INTRODUCTION

During the last two years several different approaches towards natural image generation have been suggested, among them generative adversarial networks (Goodfellow et al., 2014; Chen et al., 2016), probabilistic generative models like the conditional PixelCNN (van den Oord et al., 2016b;a) or maximum entropy models that rely on the representations of deep neural networks (e.g. Gatys et al., 2015b; Johnson et al., 2016; Ulyanov et al., 2016). The latter approach has been particularly groundbreaking for artistic style transfer and natural texture generation (e.g. Gatys et al., 2015a;b) and has the potential to uncover the regularities that supervisedly trained deep neural networks infer from natural images.

For the sake of clarity and concreteness, this paper will focus on natural texture synthesis. Parametric texture models aim to uniquely describe each texture by a set of statistical measurements that are taken over the spatial extent of the image. Each image with the same spatial summary statistics should be perceived as the same texture. Consequently, synthesizing a texture corresponds to finding a new image that reproduces the summary statistics inferred from the reference texture. Starting from Nth-order joint histograms of the pixels by Julesz (1962), many different statistical measures have been proposed (see e.g. Heeger & Bergen, 1995; Portilla & Simoncelli, 2000). The quality of the synthesized textures is usually determined by human inspection; the synthesis is successful if a human observer cannot tell the reference texture from the synthesized ones.

The current state of the art in parametric texture modeling (Gatys et al., 2015a) employs the hierarchical image representation in a deep 19-layer convolutional network (Simonyan & Zisserman (2014); in the following referred to as VGG network) that was trained on object recognition in natural images(Russakovsky et al. (2015)). In this model textures are described by the raw correlations between feature activations in response to the texture image from a collection of network layers (see section 5 for details). Since its initial reception several papers explored which additional elements or constraints can further increase the perceptual quality of the generated textures (Berger & Memisevic, 2016; Liu et al., 2016; Aittala et al., 2016). In this work we go the opposite way and ask which elements of the original texture synthesis algorithm (Gatys et al., 2015a) are absolutely indispensable.

In particular two aspects have been deemed critical for natural texture synthesis: the hierarchical multi-layer representation of the textures, and the supervised training of the feature spaces. Here we show that neither aspect is imperative for texture modeling and that in fact a single convolutional layer with random features can synthesize textures that often rival the perceptual quality of Gatys et al. (2015a). This is in contrast to earlier reports (Gatys et al., 2015a) that suggested that networks with random weights fail to generate perceptually interesting images. We suggest that this discrepancy originates from a more elaborate tuning of the optimization procedure (see section 4).

Our main contributions are:

- We present a strong minimal baseline for parametric texture synthesis that solely relies on a single-layer network and random, data-independent filters.
- We show that textures synthesized from the baseline are of high quality and often rival state-of-the-art approaches, suggesting that the depth and the pre-training of multi-layer image representations are not as indispensable for natural image generation as has previously been thought.
- We test and compare a wide range of single-layer architectures with different filter-sizes and different types of filters (random, hand-crafted and unsupervisedly learnt filters) against the state-of-the-art texture model by Gatys et al. (2015a).
- We utilize a quantitative texture quality measure based on the synthesis loss in the VGG-based model (Gatys et al., 2015a) to replace the common-place evaluation of texture models through qualitative human inspection.
- We discuss a formal generalization of maximum entropy models to account for the natural variability of textures with limited spatial extent.

2 Convolutional Neural Network

If not mentioned otherwise, all our models employ single-layer CNNs with standard rectified linear units (ReLUs) and convolutions with stride one, no bias and padding $(f-1)/2$ where f is the filter-size. This choice ensures that the spatial dimension of the output feature maps is the same as the input. All networks except the last one employ filters of size $11 \times 11 \times 3$ (filter width $\times$ filter height $\times$ no. of input channels), but the number of feature maps as well as the selection of the filters differ:

- **Fourier-363:** Each color channel (R, G, B) is filtered separately by each element $\mathbf{B}_i \in \mathbb{R}^{11\times 11}$ of the 2D Fourier basis ($11 \times 11 = 121$ feature maps/channel), yielding $3 \cdot 121 = 363$ feature maps in total. More concretely, each filter can be described as the tensor product $\mathbf{B}_i \otimes \mathbf{e}_k$ where the elements of the unit-norm $\mathbf{e}_k \in \mathbb{R}^3$ are all zero except one.
- **Fourier-3267:** All color channels (R, G, B) are filtered simultaneously by each element $\mathbf{B}_i$ of the 2D Fourier basis but with different weighting terms $w_R, w_G, w_B \in [1, 0, -1]$, yielding $3 \cdot 3 \cdot 3 \cdot 121 = 3267$ feature maps in total. More concretely, each filter can be described by the tensor product $\mathbf{B}_i \otimes [w_R, w_G, w_B]$.
- **Kmeans-363:** We randomly sample and whiten 1e7 patches of size 11×11 from the Imagenet dataset (Russakovsky et al., 2015), partition the patches into 363 clusters using k-means (Rubinstein et al., 2009), and use the cluster means as convolutional filters.
- **Kmeans-3267:** Same as Kmeans-363 but with 3267 clusters.
- **Kmeans-NonWhite-363/3267:** Same as Kmeans-363/3267 but without whitening of the patches.
- **Kmeans-Sample-363/3267:** Same as Kmeans-363/3267, but patches are only sampled from the target texture.
- **PCA-363:** We randomly sample 1e7 patches of size 11×11 from the Imagenet dataset (Russakovsky et al., 2015), vectorize each patch, perform PCA and use the set of principal axes as convolutional filters.
- **Random-363:** Filters are drawn from a uniform distribution according to (Glorot & Bengio, 2010), 363 feature maps in total.
- **Random-3267:** Same as Random-363 but with 3267 feature maps.

Figure 1: Influence of the feature maps on texture synthesis performance. (Top) Samples synthesized from several single-layer models with 363 feature maps (see sec. 2) for three different textures (rows). Reference textures are shown in the first column. (Bottom) Samples synthesized from several single-layer models with 3267 feature maps (see sec. 2) for three different textures (rows). Additionally, the first column shows samples from the VGG model (Gatys et al., 2015a), and the last column from the multi-scale model (with 1024 feature maps).

- **Random-Multiscale** Eight different filter sizes $f \times f \times 3$ with $f = 3, 5, 7, 11, 15, 23, 37, 55$ and 128 feature maps each (1024 feature maps in total). Filters are drawn from a uniform distribution according to (Glorot & Bengio, 2010).

The networks were implemented in Lasagne (Dieleman et al., 2015; Theano Development Team, 2016). We remove the DC component of the inputs by subtracting the mean intensity in each color channel (estimated over the Imagenet dataset (Russakovsky et al., 2015)).

3 Texture Model

The texture model closely follows (Gatys et al., 2015a). In essence, to characterise a given vectorised texture $\mathbf{x} \in \mathbb{R}^M$, we first pass $\mathbf{x}$ through the convolutional layer and compute the output activations. The output can be understood as a non-linear filter bank, and thus its activations form a set of filtered images (so-called feature maps). For N distinct feature maps, the rectified output activations can be

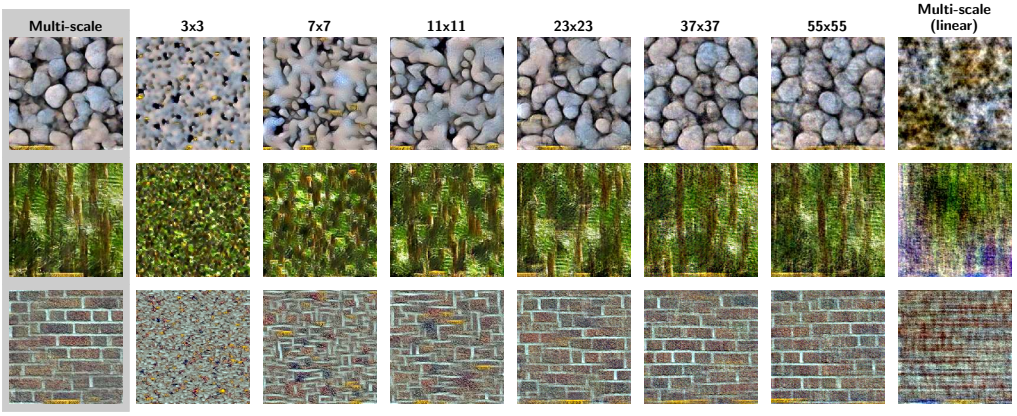


Figure 2: Influence of the scale and the non-linearity on texture synthesis performance. (1st column) Samples from the random multi-scale model for comparison (same as in Fig. 1). (2nd - 7th colum) Samples from the random single-scale model with different spatial filter sizes. (Last column) Samples from the random multi-scale model without ReLU nonlinearity.

described by a matrix $\mathbf{F} \in \mathbb{R}^{N \times M}$. To capture the stationary structure of the textures, we compute the covariances (or, more precisely, the Gramian matrix) $\mathbf{G} \in \mathbb{R}^{N \times N}$ between the feature activations $\mathbf{F}$ by averaging the outer product of the point-wise feature vectors,

$$G_{ij} = \frac{1}{M} \sum_{m=1}^{M} F_{im} F_{jm}. \tag{1}$$

We will denote $\mathbf{G}(\mathbf{x})$ as the Gram matrix of the feature activations for the input $\mathbf{x}$. To determine the relative distance between two textures $\mathbf{x}$ and $\mathbf{y}$ we compute the euclidean distance of the normalized Gram matrices,

$$d(\mathbf{x}, \mathbf{y}) = \frac{1}{\sqrt{\sum_{m,n} G_{mn}(\mathbf{x})^2} \sqrt{\sum_{m,n} G_{mn}(\mathbf{y})^2}} \sum_{i,j=1}^{N} \left(G_{ij}(\mathbf{x}) - G_{ij}(\mathbf{y}) \right)^2. \tag{2}$$

To compare with the distance in the raw pixel values, we compute

$$d_p(\mathbf{x}, \mathbf{y}) = \frac{1}{\sqrt{\sum_{m} x_m^2} \sqrt{\sum_{m} y_m^2}} \sum_{i=1}^{N} \left(x_i - y_i \right)^2. \tag{3}$$

4 Texture Synthesis

To generate a new texture we start from a uniform noise image (in the range [0, 1]) and iteratively optimize it to match the Gram matrix of the reference texture. More precisely, let $\mathbf{G}(\mathbf{x})$ be the Gram matrix of the reference texture. The goal is to find a synthesised image $\tilde{\mathbf{x}}$ such that the squared distance between $\mathbf{G}(\mathbf{x})$ and the Gram matrix $\mathbf{G}(\tilde{\mathbf{x}})$ of the synthesized image is minimized, i.e.

$$\tilde{\mathbf{x}} = \underset{\mathbf{y} \in \mathbb{R}^M}{\operatorname{argmin}} \; E(\mathbf{y}), \tag{4}$$

$$E(\mathbf{y}) = \frac{1}{\sum_{i,j=1}^{N} G_{ij}(\mathbf{x})^2} \sum_{i,j=1}^{N} \Big(G_{ij}(\mathbf{x}) - G_{ij}(\mathbf{y}) \Big)^2. \tag{5}$$

The gradient $\partial E(\mathbf{y}) / \partial \mathbf{y}$ of the reconstruction error with respect to the image can readily be computed using standard backpropagation, which we then use in conjunction with the L-BFGS-B algorithm (Jones et al., 2001–) to solve (4). We leave all parameters of the optimization algorithm at their default value except for the maximum number of iterations (2000), and add a box constraints with range [0, 1]. In addition, we scale the loss and the gradients by a factor of 10^7 in order to avoid early stopping of the optimization algorithm.

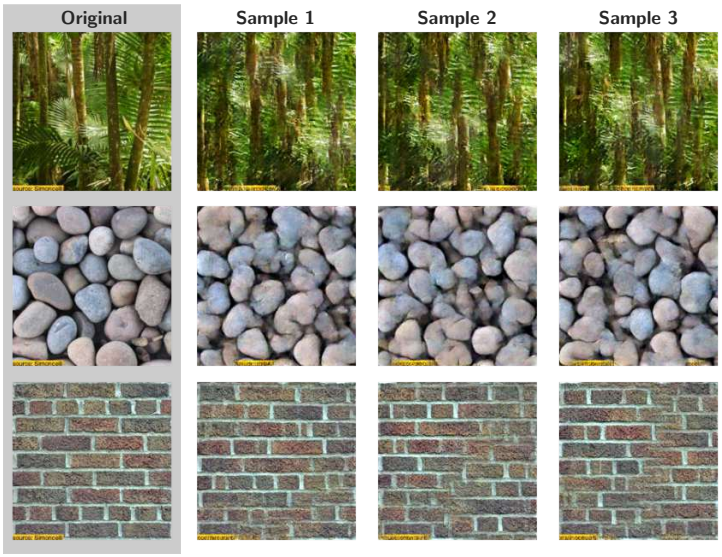


Figure 3: Each row shows the reference texture (left, gray background) and three samples that were synthesized from different (random) initial images using our multi-scale model. Most importantly, the multi-scale model generates samples that are perceptually different. All three textures are taken from Portilla & Simoncelli (2000).

5 Texture Evaluation

Evaluating the quality of the synthesized textures is traditionally performed by human inspection. Optimal texture synthesis should generate samples that humans perceive as being the same texture as the reference. The high quality of the synthesized textures by (Gatys et al., 2015a) suggests that the summary statistics from multiple layers of VGG can approximate the perceptual metric of humans. Even though the VGG texture representation is not perfect, this allows us to utilize these statistics as a more objective quantification of texture quality.

For all details of the VGG-based texture model see (Gatys et al., 2015a). Here we use the standard 19-layer VGG network (Simonyan & Zisserman, 2014) with pretrained weights and average- instead of max-pooling[1]. We compute a Gram matrix on the output of each convolutional layer that follows a pooling layer. Let $\mathbf{G}^\ell(.)$ be the Gram matrix on the activations of the ℓ-th layer and

$$E^\ell(\mathbf{y}) = \frac{1}{\sum_{i,j=1}^{N} G_{ij}^\ell(\mathbf{x})^2} \sum_{i,j=1}^{N} \Big(G_{ij}^\ell(\mathbf{x}) - G_{ij}^\ell(\mathbf{y}) \Big)^2. \tag{6}$$

the corresponding relative reconstruction cost. The total reconstruction cost is then defined as the average distance between the reference Gram matrices and the synthesized ones, i.e.

$$E(\mathbf{y}) = \frac{1}{5} \sum_{\ell=1}^{5} E^\ell(\mathbf{y}). \tag{7}$$

This cost is reported on top of each synthesised texture in Figures 4. To visually evaluate samples from our single- and multi-scale model against the VGG-based model (Gatys et al., 2015a), we additionally synthesize textures from VGG by minimizing (7) using L-BFGS-B as in section 4.

6 Results

In Fig. 1 we show textures synthesised from two random single- and multi-scale models, as well as eight other non-random single-layer models for three different source images (top left). For

[1] https://github.com/Lasagne/Recipes/blob/master/modelzoo/vgg19.py as accessed on 12.05.2016.

comparison, we also plot samples generated from the VGG model by Gatys et al. (Gatys et al., 2015a) (bottom left). There are roughly two groups of models: those with a small number of feature maps (363, top row), and those with a large number of feature maps (3267, bottom row). Only the multi-scale model employs 1024 feature maps. Within each group, we can differentiate models for which the filters are unsupervisedly trained on natural images (e.g. sparse coding filters from k-means), principally devised filter banks (e.g. 2D Fourier basis) and completely random filters (see sec. 2 for all details). All single-layer networks, except for multi-scale, feature 11×11 filters. Remarkably, despite the small spatial size of the filters, all models capture much of the small- and mid-scale structure of the textures, in particular if the number of feature maps is large. Notably, the scale of these structures extends far beyond the receptive fields of the single units (see e.g. the pebble texture). We further observe that a larger number of feature maps generally increases the perceptual quality of the generated textures. Surprisingly, however, completely random filters perform on par or better then filters that have been trained on the statistics of natural images. This is particularly true for the multi-scale model that clearly outperforms the single-scale models on all textures. The captured structures in the multi-scale model are generally much larger and often reach the full size of the texture (see e.g. the wall).

While the above results show that for natural texture synthesis one neither needs a hierarchical deep network architecture with spatial pooling nor filters that are adapted to the statistics of natural images, we now focus on the aspects that are crucial for high quality texture synthesis. First, we evaluate whether the success of the random multi-scale network arises from the combination of filters on multiple scales or whether it is simply the increased size of its largest receptive fields (55×55 vs. 11×11) that leads to the improvement compared to the single-scale model. Thus, to investigate the influence of the spatial extend of the filters and the importance of combining multiple filter sizes in one model, we generate textures from multiple single-scale models, where each model has the same number of random filters as the multi-scale model (1024) but only uses filters from a single scale of the multi-scale model (Fig. 2). We find that while 3×3 filters mainly capture the marginal distribution of the color channels, larger filters like 11×11 model small- to mid-scale structures (like small stones) but miss more long-range structures (larger stones are not well separated). Very large filters like 55×55, on the other hand, are capable of modeling long-range structures but then miss much of the small- to midscale statistics (like the texture of the stone). Therefore we conclude that the combination of different scales in the multi-scale network is important for good texture synthesis since it allows to simultaneously model small-, mid- and long-range correlations of the textures. Finally we note that a further indispensable component for good texture models are the non-linearities: textures synthesised the multi-scale model without ReLU (Fig. 2, right column) are unable to capture the statistical dependencies of the texture.

The perceptual quality of the textures generated from models with only a single layer and random filters is quite remarkable and surpasses parametric methods like Portilla & Simoncelli (2000) that have been state-of-the-art two years ago (before the use of DNNs). The multi-scale model often rivals the current state of the art (Gatys et al., 2015a) as we show in Fig. 4 where we compare samples synthesized from 20 different textures for the random single- and multi-scale model, as well as VGG. The multi-scale model generates very competitive samples in particular for textures with extremely regular structures across the whole image (e.g. for the brick wall, the grids or the scales). In part, this effect can be attributed to the more robust optimization of the single-layer model that is less prone to local minima then the optimization in deeper models. This can be seen by initializing the VGG-based synthesis with textures from the single-layer model, which consistently yields superior synthesis results (see Appendix A, Fig. 5). In addition, for a few textures such as the grid structures, the VGG-based loss is paradoxically lower for samples from the multi-scale model then for the VGG-based model (which directly optimized the VGG-based loss). This suggests that the naive synthesis performed here favors images that are perceptually similar to the reference texture and thus looses variability (see sec. 7 for further discussion). Nonetheless, samples from the single-layer model still exhibit large perceptual differences, see Fig. 3. The VGG-based loss (7) appears to generally be an acceptable approximation of the perceptual differences between the reference and the synthesized texture. Only for a few textures, especially those with very regular men-made structures (e.g. the wall or the grids), the VGG-based loss fails to capture the perceptual advantage of the multi-scale synthesis.

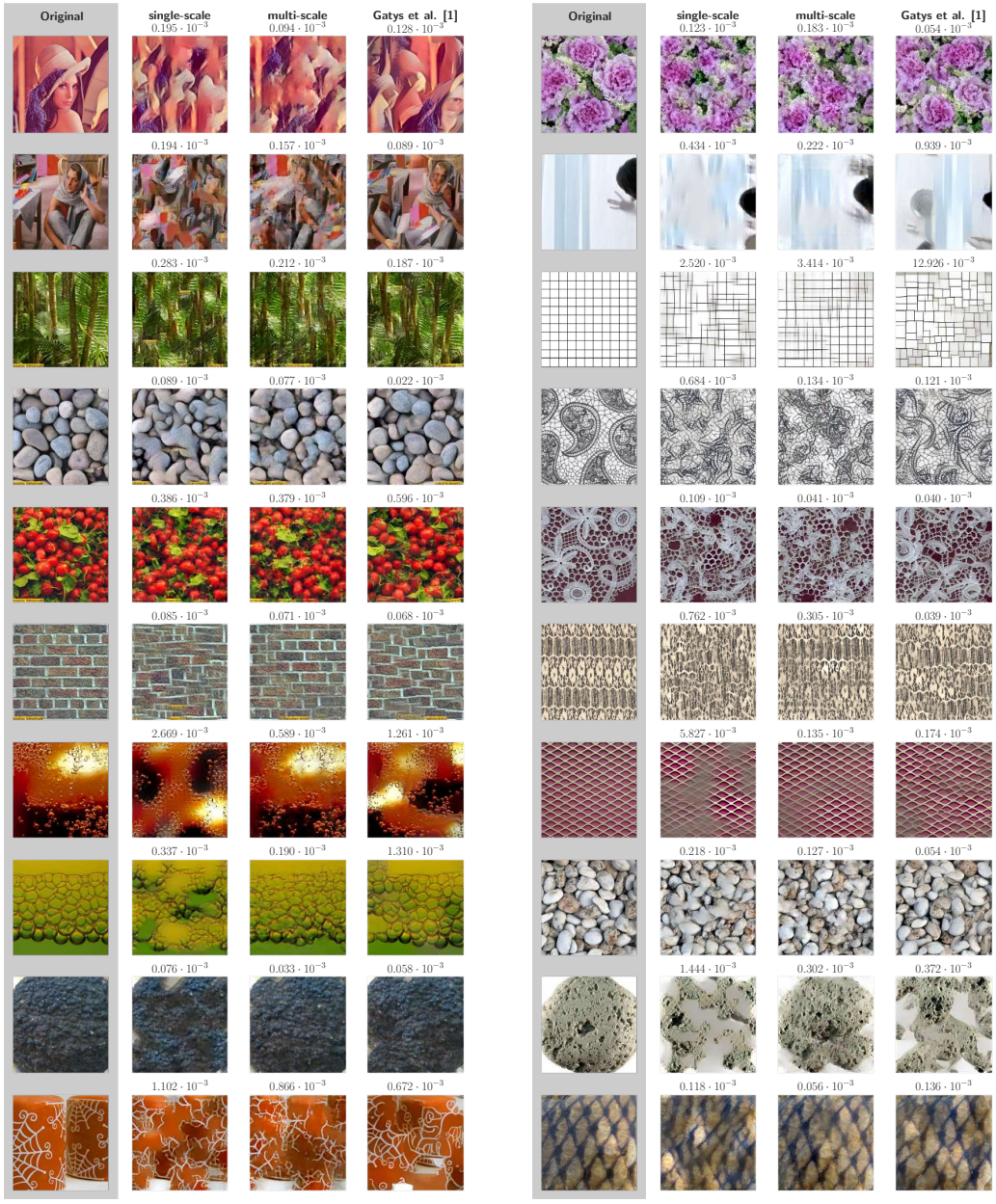


Figure 4: Each row shows the reference texture (left, gray background) and three samples that were synthesized from different (random) initial images using three different models: single-layer network with 1024 feature maps and random 11x11 filters; multi-scale single layer network with filters of sizes $f \times f$, where $f = \{3, 5, 7, 11, 15, 23, 37, 55\}$ and 128 feature maps correspond to filters of each size; and the VGG-based model (Gatys et al., 2015a). Numbers above figures show the values of the normalized VGG-loss (7) for corresponding textures.

7 DISCUSSION

We proposed a generative model of natural textures based on a single-layer convolutional neural network with completely random filters and showed that the model is able to qualitatively capture the perceptual differences between natural textures. Samples from the model often rival the current state-of-the-art (Gatys et al., 2015a) (Fig. 4, third vs fourth row), even though the latter relies on a high-performance deep neural network with features that are tuned to the statistics of natural images. Seen more broadly, this finding suggests that natural image generation does not necessarily depend on deep hierarchical representations or on the training of the feature maps. Instead, for texture synthesis, both aspects rather seem to serve as fine-tuning of the image representation.

One concern about the proposed single-layer multi-scale model is its computational inefficiency since it involves convolutions with spatially large filters (up to 55×55). A more efficient way to achieve receptive fields of similar size would be to use a hierarchical multi-layer net. We conducted extensive experiments with various hierarchical architectures and while the synthesis is indeed significantly faster, the quality of the synthesized textures does not improve compared to a single-layer model. Thus for a minimal model of natural textures, deep hierarchical representations are not necessary but they can improve the efficiency of the texture synthesis.

Our results clearly demonstrate that Gram matrices computed from the feature maps of convolutional neural networks generically lead to useful summary statistics for texture synthesis. The Gram matrix on the feature maps transforms the representations from the convolutional neural network into a stationary feature space that captures the pairwise correlations between different features. If the number of feature maps is large, then the local structures in the image are well preserved in the projected space and the overlaps of the convolutional filtering add additional constraints. At the same time, averaging out the spatial dimensions yields sufficient flexibility to generate entirely new textures that differ from the reference on a patch by patch level, but still share much of the small- and long-range statistics.

The success of shallow convolutional networks with random filters in reproducing the structure of the reference texture is remarkable and indicates that they can be useful for parametric texture synthesis. Besides reproducing the stationary correlation structure of the reference image ("perceptual similarity") another desideratum of a texture synthesis is to exhibit a large variety between different samples generated from the same given image ("variability"). Hence, synthesis algorithms need to balance perceptual similarity and variability. This balance is determined by a complex interplay between the choice of summary statistics and the optimization algorithm used. For example the stopping criterion of the optimization algorithm can be adjusted to trade perceptual similarity for larger variability.

Finding the right balance between perceptual similarity and variability is challenging because we are currently lacking robust measures of these quantities. In this work we introduced VGG-loss as a measure of perceptual similarity, and, even though, it works much better than other common measures such as Structural Similarity Index (SSIM, Wang et al., 2004, see Appendix A, Figure 6) or Euclidean distance in the pixel space (not shown), it is still not perfect (Figure 4). Measuring variability is probably even more difficult: in principle it requires measuring the entropy of generated samples, which is intractable in a high-dimensional space. A different approach could be based on a psychophysical assessment of generated samples. For example, we could use an inpainting task (illustrated in Appendix A, Figure 7) to make human observers decide between actual texture patches and inpaited ones. Performance close to a chance-level would indicate that the texture model produces variable enough samples to capture the diversity of actual patches. The further exploration of variability measures lies, however, beyond the scope of this work.

In this paper we focused on maximizing perceptual similarity only, and it is worth pointing out that additional efforts will be necessary to find an optimal trade-off between perceptual similarity and variability. For the synthesis of textures from the random models considered here, the trade-off leans more towards perceptual similarity in comparison to Gatys et al. (2015a)(due to the simpler optimization) which also explains the superior performance on some samples. In fact, we found some anecdotal evidence (not shown) in deeper multi-layer random CNNs where the reference texture was exactly reconstructed during the synthesis. From a theoretical point of view this is likely a finite size effect which does not necessarily constitute a failure of the chosen summary statistics: for finite size images it is well possible that only the reference image can exactly reproduce all the summary

statistics. Therefore, in practice, the Gram matrices are not treated as hard constraints but as soft constraints only. More generally, we do not expect a perceptual distance metric to assign exactly zero to a random pair of patches from the same texture. Instead, we expect it to assign small values for pairs from the same texture, and large values for patches from different textures. Therefore, the selection of constraints is not sufficient to characterize a texture synthesis model but only determines the exact minima of the objective function (which are sought for by the synthesis). If we additionally consider images with small but non-zero distance to the reference statistics, then the set of equivalent textures increases substantially, and the precise composition of this set becomes critically dependent on the perceptual distance metric.

Mathematically, parametric texture synthesis models are described as ergodic random fields that have maximum entropy subject to certain constraints Zhu et al. (1997); Bruna & Mallat (2013); Zhu et al. (2000) (MaxEnt framework). Practical texture synthesis algorithms, however, always deal with finite size images. As discussed above, two finite-size patches from the same ergodic random field will almost never feature the exact same summary statistics. This additional uncertainty in estimating the constraints on finite length processes is not thoroughly accounted for by the MaxEnt framework (see discussion on its "ad hockeries" by Jaynes (Jaynes (1982))). Thus, a critical difference of practical implementations of texture synthesis algorithms from the conceptual MaxEnt texture modeling framework is that they genuinely allow a small mismatch in the constraints. Accordingly, specifying the summary statistics is not sufficient but a comprehensive definition of a texture synthesis model should specify:

1. A metric $d(\mathbf{x}, \mathbf{y})$ that determines the distance between any two arbitrary textures $\mathbf{x}, \mathbf{y}$.
2. A bipartition $P_{\mathbf{x}}$ of the image space that determines which images are considered perceptually equivalent to a reference texture $\mathbf{x}$. A simple example for such a partition is the ϵ-environment $U_\epsilon(\mathbf{y}) := \{\mathbf{y} : d(\mathbf{y}, \mathbf{x}) < \epsilon\}$ and its complement.

This definition is relevant for both under- as well as over-constrained models, but its importance becomes particularly obvious for the latter. According to the Minimax entropy principle for texture modeling suggested by Zhu et al Zhu et al. (1997), as many constraints as possible should be used to reduce the (Kullback-Leibler) divergence between the true texture model and its estimate. However, for finite spatial size, the synthetic samples become exactly equivalent to the reference texture (up to shifts) in the limit of sufficiently many independent constraints. In contrast, if we explicitly allow for a small mismatch between the summary statistics of the reference image and the synthesized textures, then the set of possible textures does not constitute a low-dimensional manifold but rather a small volume within the pixel space. Alternatively, instead of introducing an ϵ-environment it is also possible to extent the MaxEnt framework to allow for variability in the summary statistics (Joan Bruna, personal communication). It will be interesting to compare in the future to what extent the difference between the two approaches can lead to differences in the perceptual appearance of the textures.

Taken together we have shown that simple single-layer CNNs with random filters can serve as the basis for excellent texture synthesis models that outperform previous hand-crafted synthesis models and sometimes even rivals the current state-of-the-art. This finding repeals previous observations that suggested a critical role for the multi-layer representations in trained deep networks for natural texture generation. On the other hand, it is not enough to just use sufficiently many constraints as one would predict from the MaxEnt framework. Instead, for the design of good texture synthesis algorithms it will be crucial to find distance measures for which the ϵ-environment around the reference texture leads to perceptually satisfying results. In this way, building better texture synthesis models is inherently related to better quantitative models of human perception.

REFERENCES

M. Aittala, T. Aila, and J. Lehtinen. Reflectance modeling by neural texture synthesis. *ACM Transactions on Graphics*, 35, 2016.

G. Berger and R. Memisevic. Incorporating long-range consistency in cnn-based texture generation. Jun 2016.

Joan Bruna and Stéphane Mallat. Audio texture synthesis with scattering moments. *CoRR*, abs/1311.0407, 2013. URL http://dblp.uni-trier.de/db/journals/corr/corr1311.html#BrunaM13.

X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. *ArXiv e-prints*, June 2016.

Sander Dieleman, Jan Schlüter, Colin Raffel, Eben Olson, Søren Kaae Sønderby, Daniel Nouri, Daniel Maturana, Martin Thoma, and other contributors. Lasagne: First release., August 2015. URL http://dx.doi.org/10.5281/zenodo.27878.

L. A. Gatys, A. S. Ecker, and M. Bethge. Texture synthesis using convolutional neural networks. In *Advances in Neural Information Processing Systems 28*, May 2015a. URL http://arxiv.org/abs/1505.07376.

L. A. Gatys, A. S. Ecker, and M. Bethge. A neural algorithm of artistic style. Aug 2015b. URL http://arxiv.org/abs/1508.06576.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS'10)*., 2010.

I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Networks. *ArXiv e-prints*, June 2014.

David J. Heeger and James R. Bergen. Pyramid-based texture analysis/synthesis. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, pp. 229–238, New York, NY, USA, 1995. ACM. ISBN 0-89791-701-4. doi: 10.1145/218380.218446. URL http://doi.acm.org/10.1145/218380.218446.

E.T. Jaynes. On the rationale of maximum-entropy methods. *Proceedings of the IEEE*, 70(9):939–952, Sept. 1982. ISSN 0018-9219.

Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, 2016.

Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. URL http://www.scipy.org/. [Online; accessed 2016-05-12].

B. Julesz. Visual pattern discrimination. *IRE Transactions on Information Theory*, 8(2):84–92, February 1962. ISSN 0096-1000. doi: 10.1109/TIT.1962.1057698.

G. Liu, Y. Gousseau, and G. Xia. Texture synthesis through convolutional neural networks and spectrum constraints. May 2016.

Javier Portilla and Eero P. Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *Int. J. Comput. Vision*, 40(1):49–70, October 2000. ISSN 0920-5691. doi: 10.1023/A:1026553619983. URL http://dx.doi.org/10.1023/A:1026553619983.

Ron Rubinstein, Michael Zibulevsky, and Michael Elad. Efficient implementation of the k-svd algorithm using batch orthogonal matching pursuit, 2009.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115 (3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. URL http://arxiv.org/abs/1409.1556.

Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016. URL http://arxiv.org/abs/1605.02688.

Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor Lempitsky. Texture Networks: Feed-forward Synthesis of Textures and Stylized Images. *arXiv:1603.03417 [cs]*, March 2016. URL http://arxiv.org/abs/1603.03417. arXiv: 1603.03417.

A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel Recurrent Neural Networks. *ArXiv e-prints*, January 2016a.

A. van den Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu. Conditional Image Generation with PixelCNN Decoders. *ArXiv e-prints*, June 2016b.

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

Song Chun Zhu, Ying Nian Wu, and David Mumford. Minimax entropy principle and its application to texture modeling. *Neural Computation*, 9(8):1627–1660, 1997.

Song Chun Zhu, Xiuwen Liu, and Ying Nian Wu. Exploring texture ensembles by efficient markov chain monte carlo-toward a 'trichromacy' theory of texture. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(6):554–569, 2000. doi: 10.1109/34.862195. URL http://dx.doi.org/10.1109/34.862195.

A APPENDIX

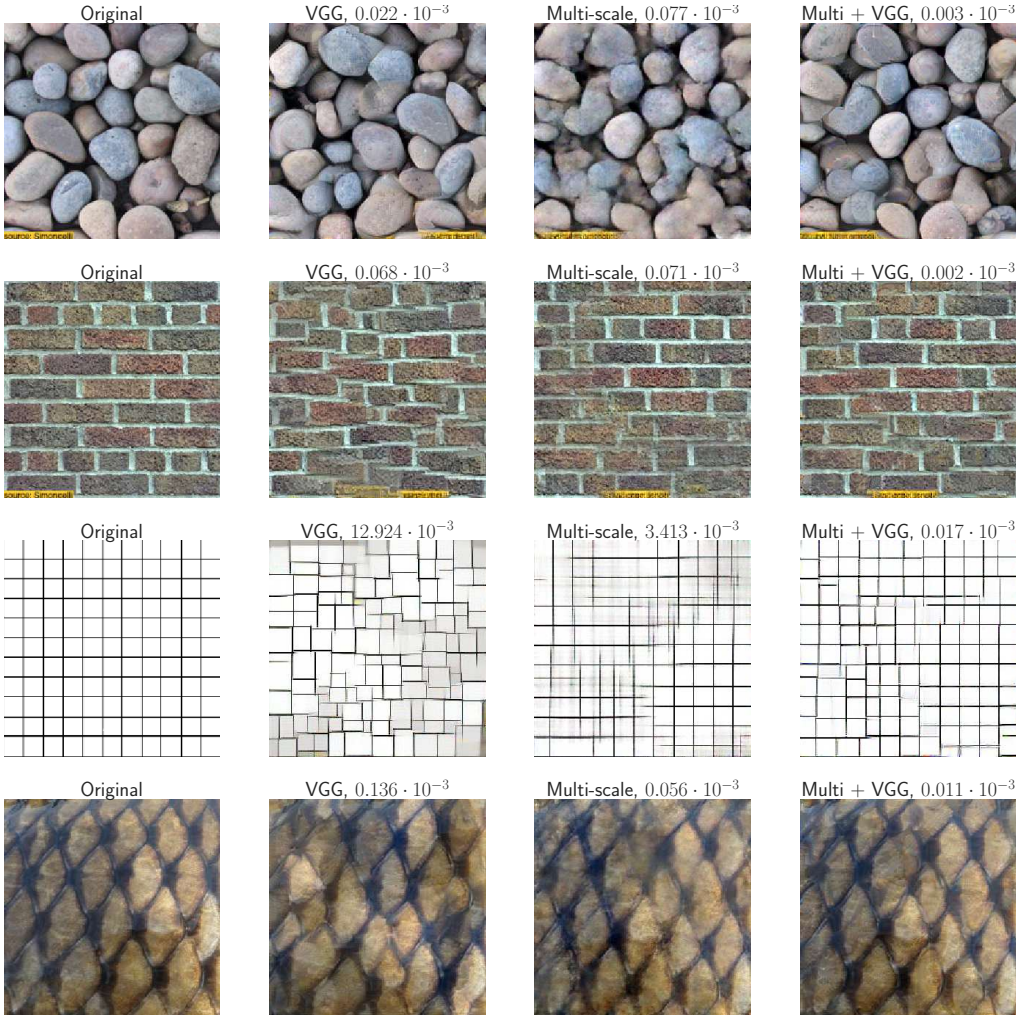


Figure 5: Initializing VGG-synthesis with a sample from the random multi-scale model. The first column shows the original textures, the second and third columns show samples from the standard VGG-based synthesis (random initialization) (Gatys et al., 2015a) and the random multi-scale model. The last column shows samples from the VGG-based model, which was initialized with samples from the random multi-scale model (from column 3). On top of all samples we report the corresponding values of the VGG-loss (7). Empirically, the VGG loss is up to two orders of magnitude lower in the last column relative to the standard VGG synthesis.

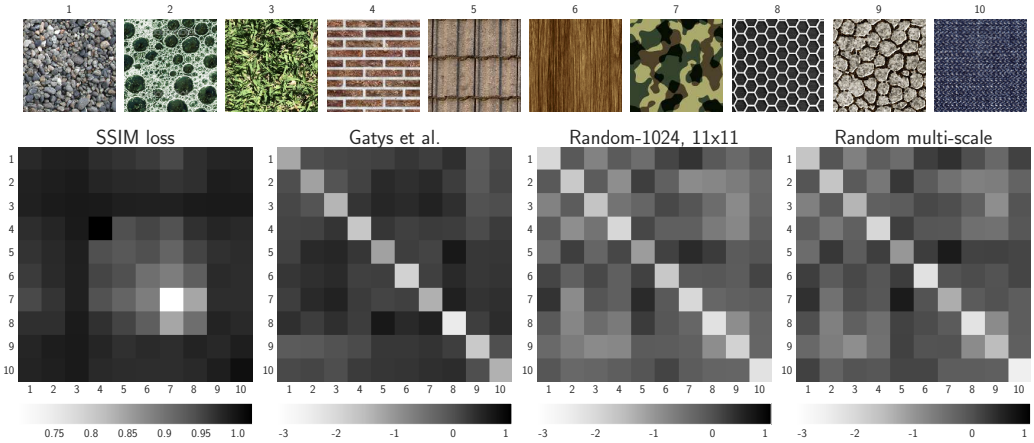


Figure 6: Similarity measures between textures computed using the structural similarity index on the pixels (SSIM, left column) and normalized euclidean distances in the feature spaces of VGG (second column) and two shallow texture models (third and fourth column). Ten random patches were extracted from each of ten different textures (examples of patches are shown in top row). The matrix element (i, j) of each similarity matrix corresponds to the median distance between patches from textures i and j. The values for all but the SSIM matrix are shown on a log-scale.

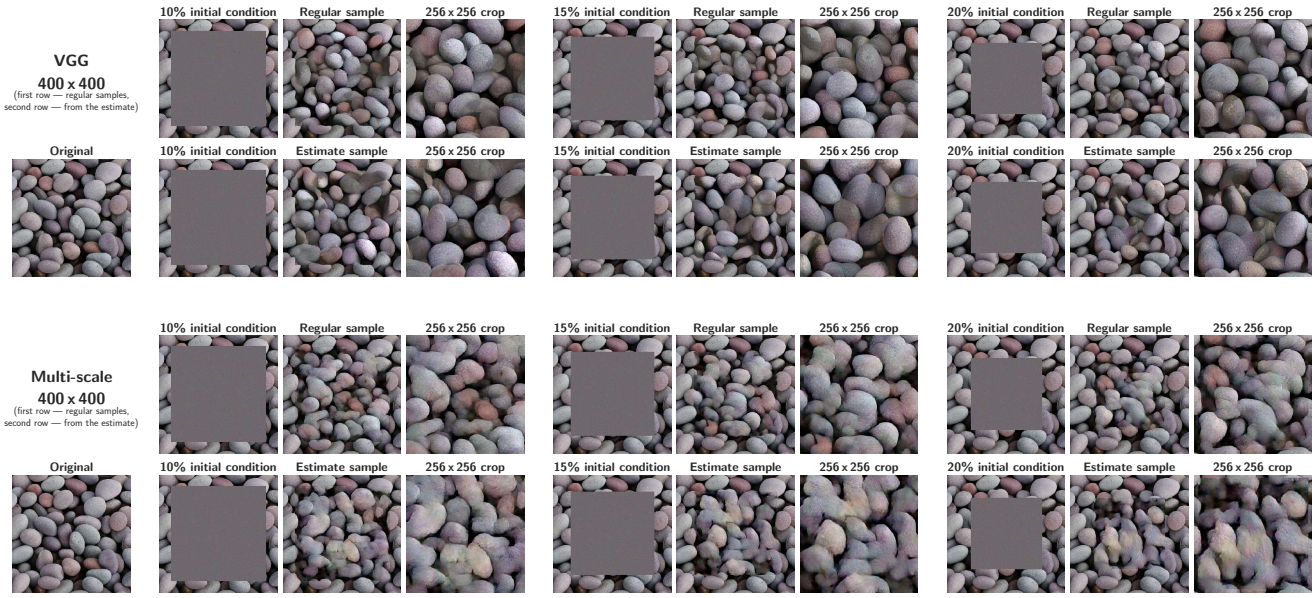


Figure 7: Examples of inpainted textures for the VGG model (Gatys et al., 2015a, , two top rows) and random multi-scale model (two bottom rows). Textures were inpainted starting from three different initial conditions (10 %, 15 %, 20 % corresponding to the width of the frame used for initialization), and for each initial condition the texture was inpainted either by matching the Gram matrix of the patch used for initializing the frame (regular sample) or by matching the Gram matrix estimated over many (500) randomly extracted patches from the texture (estimate sample).

TOWARDS CAUSAL GENERATIVE SCENE MODELS VIA COMPETITION OF EXPERTS

Julius von Kügelgen[*†1,2], **Ivan Ustyuzhaninov**[*†3],
Peter Gehler[‡4], **Matthias Bethge**[‡3,4], **Bernhard Schölkopf**[‡1,4]
[1]Max Planck Institute for Intelligent Systems Tübingen, Germany
[2]Department of Engineering, University of Cambridge, United Kingdom
[3]University of Tübingen, Germany
[4]Amazon Tübingen, Germany
{jvk,bs}@tuebingen.mpg.de,
{ivan.ustyuzhaninov,matthias.bethge}@bethgelab.org,
pgehler@amazon.com

ABSTRACT

Learning how to model complex scenes in a modular way with recombinable components is a pre-requisite for higher-order reasoning and acting in the physical world. However, current generative models lack the ability to capture the inherently compositional and layered nature of visual scenes. While recent work has made progress towards unsupervised learning of object-based scene representations, most models still maintain a global representation space (i.e., objects are not explicitly separated), and cannot generate scenes with novel object arrangement and depth ordering. Here, we present an alternative approach which uses an inductive bias encouraging modularity by training an ensemble of generative models (*experts*). During training, experts compete for explaining parts of a scene, and thus specialise on different object classes, with objects being identified as parts that re-occur across multiple scenes. Our model allows for controllable sampling of individual objects and recombination of experts in physically plausible ways. In contrast to other methods, depth layering and occlusion are handled correctly, moving this approach closer to a causal generative scene model. Experiments on simple toy data qualitatively demonstrate the conceptual advantages of the proposed approach.

1 INTRODUCTION

Proposed in the early days of computer vision Grenander (1976); Horn (1977), *analysis-by-synthesis* is an approach to the problem of visual scene understanding. The idea is conceptually elegant and appealing: build a system that is able to synthesize complex scenes (e.g., by rendering), and then understand analysis (inference) as the inverse of this process that decomposes new scenes into their constituent components. The main challenges in this approach are the need for generative models of objects (and their composition into scenes) and the need to perform tractable inference given new inputs, including the task to decompose scenes into objects in the first place. In this work, we aim to learn such as system in an unsupervised way from observations of scenes alone.

While models such as VAEs (Kingma & Welling, 2014; Rezende et al., 2014) and GANs (Goodfellow et al., 2014) constitute significant progress in generative modelling, these models still lack the ability to capture the compositional nature of reality: they typically generate entire images or scenes at once, i.e., with a single pass through a large feedforward network. While this approach works well for objects such as centred faces—and progress has been impressive on those tasks Karras et al. (2019a;b)—generating natural scenes containing several objects in non-trivial constellations gets increasingly difficult within this framework due to the combinatorial number of compositions that need to be represented and reasoned about (Bau et al., 2019).

[*]Equal contribution
[†]Work done during an internship at Amazon Research Tübingen
[‡]Joint senior author

Image formation entangles different components in highly non-linear ways, such as *occlusion*. Due to the difficulty of choosing the correct model and the complexity of inference, the task to generate complex scenes containing compositions of objects still lacks success stories. More training data certainly helps, and progress on generating visually impressive scenes has been substantial Radford et al. (2015), but we hypothesize that a satisfactory and robust solution that is not optimized to a relatively well constrained IID (independent and identically distributed) data scenario will require that our models correctly incorporate the (causal) generative nature of natural scenes.

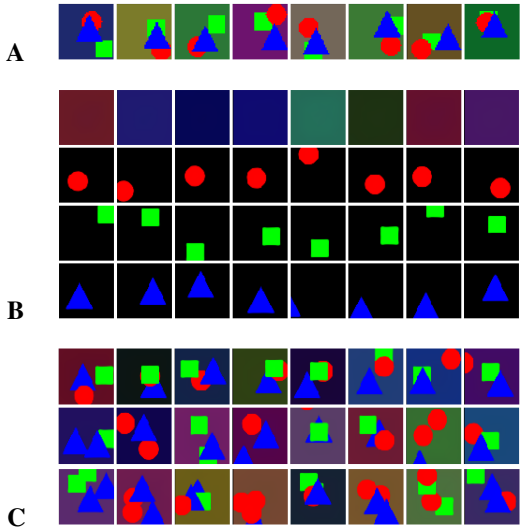


Figure 1: Our ECON model learns to decompose training scenes (A) into layers of inpainted objects. Representing object classes separately allows controllable sampling of individual objects (B: samples from different experts) which can be recombined in novel ways (C: compositions sampled by layering the experts in B in the same order as seen during training (top), or choosing three (middle) or four (bottom) objects at random).

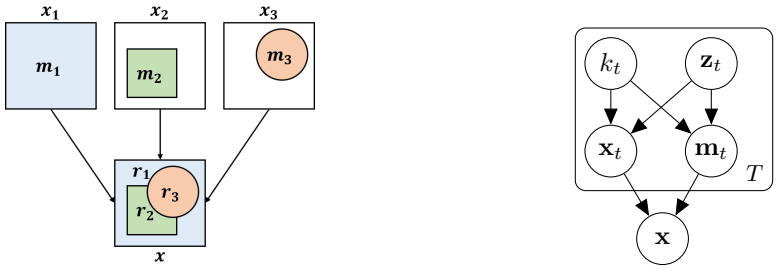
Here, we take some first small steps towards addressing the aforementioned limitations by proposing ECON, a more physically-plausible generative scene model with explicitly compositional structure. Our approach is based on two main ideas. The first is to consider scenes as *layered* compositions of (partially) depth-ordered objects. The second is to represent object classes separately using an ensemble of generative models, or *experts*.

Our generative scene model consists of a *sequential process* which places independent objects in the scene, operating from the back to the front, so that objects occurring closer to the viewer can occlude those further away. During inference, this process is reversed: at each step, experts compete for explaining part of the remaining scene, and only the winning expert is further trained on the explained part (Parascandolo et al., 2018). This competition ideally drives each expert to specialise on representing and generating instances from one, or a few related, object classes or concepts, and the notion of "objects" should automatically emerge as contiguous regions that appear in a stable way across a range of training images. By decomposing scenes in the reverse order of generation, occluded objects can be inpainted within the already explained regions so that experts can learn to generate full, unoccluded objects which can be recombined in novel ways.

Learning a modular scene representation via object-specific experts has several benefits. First, each expert only needs to solve the simpler subtask of representing and generating instances from a single object class—something which current generative models have been shown to be capable of—while the composition process is treated separately. Secondly, expert models are useful in their own right as they can be dropped or added, reused and repurposed for other tasks on an individual level.

We highlight the following contributions.

- We summarise a physically-plausible model of scene generation in §2 and use it to categorise and contrast related scene models and their shortcomings in §3.
- In §4, we present ECON, a compositional scene model, which, for a single expert, can be seen as extension of MONET (Burgess et al., 2019) into a proper generative model (§5.1).
- We introduce modular object representations through separate generators and propose a competition mechanism and objective to drive experts to specialise in §5.2.
- In experiments on synthetic data in §6 we show qualitatively that ECON is able to decompose simple scenes into objects, represent these separately, and recombine them in a layer-wise fashion into novel, coherent scenes with arbitrary numbers and depth-orderings of objects.
- We critically discuss our assumptions and propose extensions for future work in §7.



(a) layer-wise composition

(b) graphical model view

Figure 2: Assumed data generating process (dead-leaves model). Independent objects $\mathbf{x}_t$ with shapes $\mathbf{m}_t$ (drawn from class k_t with properties z_t) are placed on the canvas sequentially (reflecting depth ordering) and appear in the final composition $\mathbf{x}$ as *dependent*, partially occluded regions $\mathbf{r}_t$.

2 THE LAYER-BASED MODEL OF VISUAL SCENES

To reflect the fact that 2D images are the result of projections of richer 3D scenes, we assume that data are generated from the well-known *dead leaves model*,[1] i.e., in a layer-wise fashion, see Figure 2a for an illustration. Starting with an empty canvas $\mathbf{x} = \mathbf{0}$, an image $\mathbf{x} \in [0,1]^{D\times 3}$ is sequentially generated in T steps. At each step we sample an object from one of K different classes and place it on the canvas as follows,

$$
\begin{aligned}
\textbf{for} \quad & t = 1, ..., T: \\
& k_t \sim p(k_t), && \text{object class} \\
& \mathbf{z}_t \sim p(\mathbf{z}_t), && \text{object properties} \\
& \mathbf{m}_t \sim p(\mathbf{m}_t \mid \mathbf{z}_t, k_t), && \text{shape} \\
& \mathbf{x}_t \sim p(\mathbf{x}_t \mid \mathbf{z}_t, k_t), && \text{appearance} \\
& \mathbf{x} \leftarrow \mathbf{m}_t \odot \mathbf{x}_t + (1 - \mathbf{m}_t) \odot \mathbf{x}, && \text{place on canvas}
\end{aligned}
$$

where $k_t \in \{1, ..., K\}$ represents the object class drawn at step t; $\mathbf{z}_t \in \mathbb{R}^L$ is an abstract representation of the object's properties; $\mathbf{m}_t \in \{0,1\}^D$ is a binary image determining shape; $\mathbf{x}_t \in [0,1]^{D\times 3}$ is a full (unmasked) image containing the object; and $\odot$ denotes element-wise multiplication. The corresponding graphical model is shown in Figure 2b.[2]

This sequential generation process captures the loss of depth information when projecting from 3D to 2D and is a natural way of handling occlusion phenomena. Consequently, sampling from this model is straightforward. We therefore consider it a more truthful approach to modelling visual scenes than, e.g., spatial mixture models, in line with Le Roux et al. (2011).

On the other hand, inferring the objects composing a given image $\mathbf{x}$ is challenging. We will distinguish between *shapes* and *regions* in the following sense. The unoccluded object shapes $\mathbf{m}_t$, top row in Figure 2a, remain hidden and only appear in $\mathbf{x}$ via their corresponding, partially occluded segmentation regions $\mathbf{r}_t \in \{0,1\}^D$, see the final composition in the bottom row of Figure 2a for an illustration. In particular, a region $\mathbf{r}_t$ is always subset of the corresponding shape pixels $\mathbf{m}_t$.

In addition to the separate treatment of *shapes* $\mathbf{m}_t$ and *regions* $\mathbf{r}_t$, we also introduce a *scope* variable $\mathbf{s}_t$ to help write the above model in a convenient form. Following Burgess et al. (2019), $\mathbf{s}_t \in \{0,1\}^D$ is defined recursively as

$$
\mathbf{s}_T := \mathbf{1}, \quad \mathbf{s}_t := \mathbf{s}_{t+1} \odot (1 - \mathbf{m}_{t+1}) \quad \forall t < T. \tag{1}
$$

The scope $\mathbf{s}_t$ at time t contains those parts of the image, which have been completely generated after t steps and will not be occluded in the subsequent $T - t$ steps.

With $\mathbf{s}_t$, the regions $\mathbf{r}_t$ can be compactly defined as

$$
\mathbf{r}_t = r(\mathbf{m}_t, \ldots, \mathbf{m}_T) := \mathbf{s}_t \odot \mathbf{m}_t, \quad t = 1, \ldots, T. \tag{2}
$$

[1] the name derives from the analogy of leaves falling onto a canvas, covering whatever is beneath them,

[2] W.l.o.g., we assume that the background corresponds to $\mathbf{m}_1 \odot \mathbf{x}_1$ with $\mathbf{m}_1 = \mathbf{1}$, see Figure 2a.

Using these, we can express the final composition as

$$\mathbf{x} = \sum_{t=1}^{T} \mathbf{r}_t \odot \mathbf{x}_t. \tag{3}$$

While (3) may look like a normal spatial mixture model, it is worth noting the following important point: even though the shapes $\mathbf{m}_t$ are drawn independently, the resulting segmentation regions $\mathbf{r}_t$ become (temporally) dependent due to the layer-wise generation process, i.e., the visible part of object t depends on all objects subsequently placed on the canvas. This seems very intuitive and is evident from the fact that the RHS of (2) is a function of $\mathbf{m}_{t:T}$.

3 Related work

Clustering & spatial mixture models One line of work (Greff et al., 2016; 2017; Van Steenkiste et al., 2018) approaches the *perceptual grouping* task of decomposing scenes into components by viewing separate regions $\mathbf{r}_t$ as clusters. A scene $\mathbf{x}$ is modelled with a spatial mixture model, parametrised by deep neural networks, in which learning is performed with a procedure akin to expectation maximisation (EM; Dempster et al., 1977). The recent IODINE model of Greff et al. (2019) instead uses a refinement network (Marino et al., 2018) to perform iterative amortised variational inference over independent scene components which are separately decoded and then combined via a softmax to form the scene. While IODINE is able to *decompose* a given scene, it cannot *generate* coherent samples of new scenes because dependencies between regions $\mathbf{r}_t$ due to layering are not explicitly captured in its generative model.

This shortcoming of IODINE has also been pointed out by Engelcke et al. (2019) and addressed in their GENESIS model, which explicitly models dependencies between regions via an autoregressive prior over $\mathbf{r}_{1:T}$. While this *does* enable sampling of coherent scenes which look similar to training data, GENESIS still assumes an additive, rather than layered, model of scene composition. As a consequence, the resulting entangled component samples contain holes and partially occluded objects and cannot be easily layered and recombined as shown in Figure 1 (e.g., to generate samples with exactly two circles and one triangle).

Sequential models Our work is closely related to sequential or recurrent approaches to image decomposition and generation (Mnih et al., 2014; Gregor et al., 2015; Eslami et al., 2016; Kosiorek et al., 2018; Yuan et al., 2019). In particular, we build on the recent MONET model for scene decomposition of Burgess et al. (2019). MONET combines a recurrent attention network with a VAE which encodes and reconstructs the input within the selected attention regions $\mathbf{r}_t$ while unconstrained to inpaint occluded parts outside $\mathbf{r}_t$.

We extend this approach in two main directions. Firstly, we turn MONET into a proper *generative* model[3] which respects the layer-wise generation of scenes described in §2. Secondly, we explicitly model the discrete variable k (object class) with an *ensemble* of class-specific VAEs (the *experts*)—as opposed to within a single large encoder-decoder architecture as in IODINE, GENESIS or MONET. Such specialisation allows to control object constellations in new, but scene-consistent ways.

Competition of experts To achieve specialisation on different object classes in our model, we build on ideas from previous work using competitive training of experts (Jacobs & Jordan, 1991). More recently, these ideas have been successfully applied to tasks such as lifelong learning (Aljundi et al., 2017), learning independent causal mechanisms (Parascandolo et al., 2018), training mixtures of generative models (Locatello et al., 2018), as well as to dynamical systems via sparsely-interacting recurrent independent mechanisms (Goyal et al., 2019).

Probabilistic RBM models The work of Le Roux et al. (2011) and Heess (2012) introduced probabilistic scene models that also reason about occlusion. Le Roux et al. (2011) combine restricted Boltzmann machines (RBMs) to generate masks and shape separately for every object in the scenes into a masked RBM (M-RBM) model. Two variants are explored: one that respects a depth ordering and object occlusions, derived from similar arguments as we have put forward in the introduction;

[3] in its original form, it is a conditional model which does not admit a canonical way of sampling new scenes

Table 1: Comparison with related unsupervised scene decomposition and generation models.

	MONET	IODINE	GENESIS	M-RBM	ECON
decompose scenes into objects and reconstruct	✓	✓	✓	✓	✓
generate coherent scenes like training data	✗	✗	✓	✓	✓
controllably recombine objects in novel ways	✗	✗	✗	✓	✓
efficient (amortised) inference	✓	✓	✓	✗	✓

and a second model which uses a softmax combination akin to the spatial mixture models used in IODINE and GENESIS, although the authors argue it makes little sense from a modelling perspective. Inference is implemented as blocked Gibbs sampling with contrastive divergence as a learning objective. Inference over depth ordering is done exhaustively, that is, considering every permutation—as opposed to greedily using competition as in this work. Shortcomings of the model are mainly the limited expressiveness of RBMs (complexity and extent), as well as the cost of inference. Our work can be understood as an extension of the M-RBM formulation using VAEs in combination with attention, or segmentation, models.

Vision as inverse graphics & probabilistic programs Another way to programmatically introduce information about scene composition is through analysis-by-synthesis, see Bever & Poeppel (2010) for an overview. In this approach, the synthesis (i.e., generative) model is fully specified, e.g., through a graphics renderer, and inference becomes the inverse task, which poses a challenging optimisation problem. Probabilistic programming is often advocated as a means to automatically compile this inference task; for instance, PICTURE has been proposed by Kulkarni et al. (2015), and combinations with deep learning have been explored by Wu et al. (2017). This approach is sometimes also understood as an instance of Approximate Bayesian Computation (ABC; Dempster et al., 1977) or likelihood-free inference. While conceptually appealing, these methods require a detailed specification of the scene generation process—something that we aim to learn in an unsupervised way. Furthermore, gains achieved by a more accurate scene generation process are generally paid for by complicated inference, and most methods thus rely on variations of MCMC sampling schemes (Jampani et al., 2015; Wu et al., 2017).

Supervised approaches There is a body of work on augmenting generative models with ground-truth segmentation and other supervisory information. Turkoglu et al. (2019) proposed a layer based model to add objects onto a background, Ashual & Wolf (2019) proposed a scene-generation method allowing for fine grained user control, Karras et al. (2019a;b) have achieved impressive image generation results by exclusively training on a single class of objects. The key difference of these approaches to our work is that we exclusively focus on *unsupervised* approaches.

4 Ensemble of Competing Object Nets (ECON)

We now introduce ECON (for **E**nsemble of **C**ompeting **O**bject **N**etworks), a causal generative scene model which explicitly captures the compositional nature of visual scenes. On a high level, the proposed architecture is an ensemble of generative models, or *experts*, designed after the layer-based scene model described in §2. During training, experts compete to sequentially explain a given scene via attention over image regions, thereby specialising on different object classes. We perform variational inference (Jordan et al., 1999), amortised within the popular VAE framework (Kingma & Welling, 2014; Rezende et al., 2014), and use competition to greedily maximise a lower bound to the conditional likelihood w.r.t. object identity.

4.1 Generative Model

We adopt the generative model p described in §2, parametrise it by θ, and assume that it factorises over the graphical model in Figure 2b (i.e., assuming that objects at different time steps are drawn independently of each other). We model $p(k_t)$ with a categorical distribution,[4] and place a unit-

[4]though we will generally condition on k_t, see §5 for details,

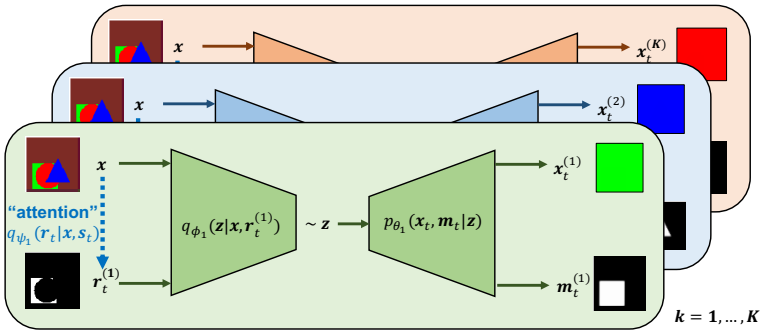


Figure 3: ECON architecture: ensemble of K competing experts. Each expert consists of (i) an attention network which selects image regions $\mathbf{r}_t$; (ii) an encoder which maps the image within the attended region to a latent code $\mathbf{z}$; and (iii) a decoder which reconstructs both an object $\mathbf{x}_t$ and its unoccluded shape $\mathbf{m}_t$. A competition mechanism determines the winning expert at each step.

variance isotropic Gaussian prior over $\mathbf{z}_t$,

$$p(\mathbf{z}_t) = p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, I) \quad t = 1, \ldots, T.$$

Next, we parametrise $p(\mathbf{m}_t \mid k, \mathbf{z})$ and $p(\mathbf{x}_t \mid k, \mathbf{z})$ using K *decoders* $f_1, \ldots, f_K : \mathbb{R}^L \to [0,1]^{D\times 3} \times [0,1]^D$ with respective parameters $\theta_1, \ldots, \theta_K$.[5] These compute object means and mask probabilities $\big(\mu_{\theta_k}(\mathbf{z}), \tilde{\mathbf{m}}_{\theta_k}(\mathbf{z})\big) = f_k(\mathbf{z}; \theta_k)$ which determine pixel-wise distributions over $\mathbf{m}_t$ and $\mathbf{x}_t$ via

$$p_\theta(\mathbf{m}_t \mid \mathbf{z}, k) = \text{Bernoulli}\left(\tilde{\mathbf{m}}_{\theta_k}(\mathbf{z})\right), \tag{4}$$

$$p_\theta(\mathbf{x}_t \mid \mathbf{z}, k) = \mathcal{N}\big(\mathbf{x}_t \mid \mu_{\theta_k}(\mathbf{z}), \sigma_x^2 I\big), \tag{5}$$

where $t = 1, \ldots, T$ and σ_x^2 is a constant variance.

We note at this point that, while other handlings of the discrete variable k are possible, we deliberately opt for K separate decoders: (i) as an inductive bias encouraging modularity; and (ii) to be able to controllably sample individual objects and recombine them in novel ways.

Finally, we need to specify a distribution over $\mathbf{x}$. Due to its layer-wise generation, this is tricky and most easily done in terms of the visible regions $\mathbf{r}_t$. From (3), (5), and linearity of Gaussians it follows that, pixel-wise,

$$p_\theta(\mathbf{x} \mid \mathbf{r}_{1:T}, \mathbf{z}_{1:T}, k_{1:T}) = \mathcal{N}\Big(\mathbf{x} \;\Big|\; \sum_{t=1}^{T} \mathbf{r}_t \odot \mu_{\theta_{k_t}}(\mathbf{z}_t), \sigma_x^2 I\Big). \tag{6}$$

Similarly, one can show from (1), (2), and (4) that $\mathbf{r}_t$ depends on $\mathbf{r}_{(t+1):T}$ only via $\mathbf{s}_t$, and that

$$p_\theta(\mathbf{r}_t \mid \mathbf{s}_t, \mathbf{z}, k) = \text{Bernoulli}\left(\mathbf{s}_t \odot \tilde{\mathbf{m}}_{\theta_k}(\mathbf{z})\right), \tag{7}$$

for $t = 1, \ldots, T$; see Appendix A for detailed derivations.

The class-conditional joint distribution then factorises as,

$$p_\theta(\mathbf{x}, \mathbf{r}_{1:T}, \mathbf{z}_{1:T} | k_{1:T}) = p_\theta(\mathbf{x} \mid \mathbf{r}_{1:T}, \mathbf{z}_{1:T}, k_{1:T}) \prod_{t=1}^{T} p_\theta(\mathbf{r}_t \mid \mathbf{s}_t, \mathbf{z}_t, k_t) p(\mathbf{z}_t). \tag{8}$$

Conditioning on $k_{1:T}$ is motivated by our inference procedure, see §5. Moreover, we express p in terms of the segmentation regions $\mathbf{r}_t$ as only these are visible in the final composition which makes is easier to specify a distribution over $\mathbf{x}$. Note, however, that while we will perform *inference over regions* $\mathbf{r}_{1:T}$, we will learn to *generate full shapes* $\mathbf{m}_{1:T}$ which are consistent with the inferred $\mathbf{r}_{1:T}$ when composed layer-wise as captured in (7), thus respecting the physical data-generating process.

[5] K is a hyperparameter ($K-1$ object classes and background) which has to be chosen domain dependently.

4.2 APPROXIMATE POSTERIOR

Since exact inference is intractable in our model, we approximate the posterior over $\mathbf{z}_{1:T}$ and $\mathbf{r}_{1:T}$ with the following variational distribution q parametrised by ϕ and ψ,

$$q_{\phi,\psi}(\mathbf{r}_{1:T}, \mathbf{z}_{1:T} \mid \mathbf{x}, k_{1:T}) = \prod_{t=1}^{T} q_\psi(\mathbf{r}_t \mid \mathbf{x}, \mathbf{s}_t, k_t) q_\phi(\mathbf{z}_t \mid \mathbf{x}, \mathbf{r}_t, k_t).$$

As for the generative distribution, we model dependence on k_t using K modules with separate parameters $\{\phi_1, \psi_1\}$, ..., $\{\phi_K, \psi_K\}$. These inference modules consist of two parts.

Attention nets $a_1, ..., a_K : [0,1]^{D\times 3} \times [0,1]^D \to [0,1]^D$ compute region probabilities $\tilde{\mathbf{r}}_{\psi_k}(\mathbf{x}, \mathbf{s}) = a_k(\mathbf{x}, \mathbf{s}; \psi_k)$ and amortise inference over regions $\mathbf{r}_t$ via

$$q_\psi(\mathbf{r}_t \mid \mathbf{x}, \mathbf{s}_t, k_t) = \text{Bernoulli}\left(\tilde{\mathbf{r}}_{\psi_k}(\mathbf{x}, \mathbf{s}_t)\right) \quad \forall t. \tag{9}$$

Encoders $g_1, ..., g_K : [0,1]^{D\times 3} \times [0,1]^D \to \mathbb{R}^{L\times 2}$ compute means and log-variances $\left(\mu_{\phi_k}(\mathbf{x}, \mathbf{r}_t), \log \sigma^2_{\phi_k}(\mathbf{x}, \mathbf{r}_t)\right) = g_k(\mathbf{x}, \mathbf{r}_t; \phi_k)$ which parametrise distributions over $\mathbf{z}_t$ via

$$q_\phi(\mathbf{z}_t \mid \mathbf{x}, \mathbf{r}_t, k) = \mathcal{N}\left(\mathbf{z}_t \mid \mu_{\phi_k}(\mathbf{x}, \mathbf{r}_t), \sigma^2_{\phi_k}(\mathbf{x}, \mathbf{r}_t) I\right) \quad \forall t.$$

We refer to the collection of $f_k(\,\cdot\,; \theta_k)$, $a_k(\,\cdot\,; \psi_k)$, and $g_k(\,\cdot\,; \phi_k)$ for a given k as an *expert* as it implements all computations (generation and inference) for a specific object class—see Figure 3 for an illustration.

5 INFERENCE

Due to the assumed sequential generative process, the natural order of inference is the reverse ($t = T, \ldots, 1$), i.e., foreground objects should be explained first and the background last. This is also captured by the dependence of $\mathbf{r}_t$ on $\mathbf{r}_{(t+1):T}$ via the scope $\mathbf{s}_t$ in q_ψ.

Such entanglement of scene components across composition steps makes inference over the entire scene intractable. We therefore choose the following greedy approach. At each inference step $t = T, \ldots, 1$, we consider explanations from all possible object-classes ($k_t = 1, \ldots, K$)—as provided by our ensemble of experts via attending, encoding and reconstructing different parts of the current scene—and then choose the best fitting one. This offers an intuitive foreground to background decomposition of an image as foreground objects should be easier to reconstruct.

Concretely, we first lower bound the marginal likelihood conditioned on $k_{1:T}$, $p_\theta(\mathbf{x} \mid k_{1:T})$, and then use a competition mechanism between experts to determine the best k. We now describe this inference procedure in more detail.

5.1 OBJECTIVE: CLASS-CONDITIONAL ELBO

First, we lower bound the class-conditional model evidence $p_\theta(\mathbf{x} \mid k_{1:T})$ using the approximate posterior q as follows (see Appendix A for a detailed derivation):

$$\log p_\theta(\mathbf{x} \mid k_{1:T}) \geq \mathcal{L}(\theta, \psi, \phi \mid k_{1:T}) := -\sum_{t=1}^{T} \mathbb{E}_{q_\psi(\mathbf{s}_t \mid \mathbf{x}, k_{(t+1):T})} \left(\mathcal{L}_{\mathbf{x},t} + \mathcal{L}_{\mathbf{z},t} + \mathcal{L}_{\mathbf{r},t}\right),$$

$$\mathcal{L}_{\mathbf{x},t} := \mathbb{E}_{q_\psi(\mathbf{r}_t \mid \mathbf{x}, \mathbf{s}_t, k_t) q_\phi(\mathbf{z}_t \mid \mathbf{x}, \mathbf{r}_t, k_t)} \left[\frac{\mathbf{r}_t}{2\sigma^2_{\mathbf{x}}} \odot \left(\mathbf{x} - \mu_{\theta_{k_t}}(\mathbf{z}_t)\right)^2\right]$$

$$\mathcal{L}_{\mathbf{z},t} := \mathbb{E}_{q_\psi(\mathbf{r}_t \mid \mathbf{x}, \mathbf{s}_t, k_t)} D_{KL}\Big(q_\phi(\mathbf{z}_t \mid \mathbf{x}, \mathbf{r}_t, k_t) \;\Big\|\; p(\mathbf{z})\Big)$$

$$\mathcal{L}_{\mathbf{r},t} := \mathbb{E}_{q_\psi(\mathbf{r}_t \mid \mathbf{x}, \mathbf{s}_t, k_t) q_\phi(\mathbf{z}_t \mid \mathbf{x}, \mathbf{r}_t, k_t)} \left[\log \frac{q_\psi(\mathbf{r}_t \mid \mathbf{x}, \mathbf{s}_t, k_t)}{p_\theta(\mathbf{r}_t \mid \mathbf{s}_t, \mathbf{z}_t, k_t)}\right]$$

Next, we use the reparametrization trick of Kingma & Welling (2014) to replace expectations w.r.t. $q_\phi(\mathbf{z}_t \mid \mathbf{x}, \mathbf{r}_t, k_t)$ by a Monte Carlo estimate using a single sample drawn as:

$$\tilde{\mathbf{z}}_t = \mu_{\phi_{k_t}}(\mathbf{x}, \mathbf{r}_t) + \sigma_{\phi_{k_t}}(\mathbf{x}, \mathbf{r}_t) \odot \epsilon, \quad \epsilon \sim \mathcal{N}(\mathbf{0}, I).$$

Finally, we approximate expectations w.r.t. $q_\psi(\mathbf{r}_t \mid \mathbf{x}, \mathbf{s}_t, k_t)$ in $\mathcal{L}_{\mathbf{x},t}$ and $\mathcal{L}_{\mathbf{z},t}$ using the Bernoulli means $\tilde{\mathbf{r}}_{\psi_{k_t}}(\mathbf{x}, \mathbf{s}_t)$. We opt for directly using a continuous approximation and against sampling discrete r's (e.g., using continuous relaxations to the Bernoulli distribution (Maddison et al., 2017; Jang et al., 2017)) as our generative model does not require the ability to directly sample regions. (Instead, we sample $\mathbf{z}$'s and decode them into unoccluded shapes which can be combined layer-wise to form scenes.)

With these approximations, we obtain the estimates

$$\hat{\mathcal{L}}_{\mathbf{x},t} := \frac{\tilde{\mathbf{r}}_{\psi_{k_t}}(\mathbf{x}, \mathbf{s}_t)}{2\sigma_{\mathbf{x}}^2} \odot \left(\mathbf{x} - \mu_{\theta_{k_t}}(\tilde{\mathbf{z}}_t)\right)^2, \tag{10}$$

$$\hat{\mathcal{L}}_{\mathbf{z},t} := D_{KL}\Big(q_\phi\left(\mathbf{z}_t \mid \mathbf{x}, \tilde{\mathbf{r}}_{\psi_{k_t}}(\mathbf{x}, \mathbf{s}_t), k_t\right) \;\Big\|\; p(\mathbf{z})\Big), \tag{11}$$

$$\hat{\mathcal{L}}_{\mathbf{r},t} := D_{KL}\Big(q_\psi(\mathbf{r}_t \mid \mathbf{x}, \mathbf{s}_t, k_t) \;\Big\|\; p_\theta(\mathbf{r}_t \mid \mathbf{s}_t, \tilde{\mathbf{z}}_t, k_t)\Big), \tag{12}$$

which we combine to form the *learning objective*

$$\hat{\mathcal{L}}(\theta, \psi, \phi \mid k_{1:T}) = -\sum_{t=1}^{T}\left(\hat{\mathcal{L}}_{\mathbf{x},t} + \beta\hat{\mathcal{L}}_{\mathbf{z},t} + \gamma\hat{\mathcal{L}}_{\mathbf{r},t}\right), \tag{13}$$

where β, γ are hyperparameters. Note that for $\beta, \gamma > 1$, (13) still approximates a valid lower bound.

Generative model extension of MONET as a special case For $K = 1$ (i.e., ignoring different object classes k_t for the moment), our derived objective (13) is similar to that used by Burgess et al. (2019). However, we note the following crucial difference in (12): in our model, reconstructed attention regions $\tilde{\mathbf{m}}_{\theta_k}(\mathbf{z})$ are multiplied by $\mathbf{s}_t$ in the p_θ term of the KL, see (7). This implies that the generated shapes $\mathbf{m}_t$ are constrained to match the attention region $\mathbf{r}_t$ only within the current scope $\mathbf{s}_t$, so that—unlike in MONET—the decoder is not penalised for generating entire unoccluded object shapes, *allowing inpainting also on the level of masks.* With just a single expert, our model can thus be understood as a generative model extension of MONET.

5.2 COMPETITION MECHANISM

For $K > 1$, i.e., when explicitly modelling object classes with separate experts, the objective (13) cannot be optimised directly because it is conditioned on the object identities $k_{1:T}$. To address this issue, we use the following competition mechanism between experts.

At each inference step $t = T, \ldots, 1$, we apply all experts $(k_t = 1, \ldots, K)$ to the current input $(\mathbf{x}, \mathbf{s}_t)$ and declare that expert the winner which yields the best competition objective (see below).[6] We then use the winning expert $\hat{k}_t$ to reconstruct the selected scene component using

$$\mathbf{x}_t = \mu_{\theta_{\hat{k}_t}}\left(\tilde{\mathbf{z}}_t(\hat{k}_t)\right),$$

where $\tilde{\mathbf{z}}_t(\hat{k}_t)$ is encoded from the region $\tilde{\mathbf{r}}_{\psi_{\hat{k}_t}}$ attended by the winning expert $\hat{k}_t$. We then compute the contribution to (13) from step t assuming fixed $\hat{k}_t$, and use it to update the winning expert with a gradient step. Finally, we update the scope using the winning expert,

$$\mathbf{s}_{t-1} = \mathbf{s}_t \odot \left(\mathbf{1} - \tilde{\mathbf{r}}_{\psi_{\hat{k}_t}}(\mathbf{x}, \mathbf{s}_t)\right), \tag{14}$$

to allow for inpainting within the explained region in the following inference (decomposition) steps.[7]

This competition process can be seen as a *greedy approximation to maximising* (13) w.r.t. $k_{1:T}$. While considering all possible object combinations would require $O(K^T)$ steps, our competition procedure is linear in the number of object classes and runs in $O(K \cdot T)$ steps. By choosing an expert at each step $t = T, \ldots, 1$, we approximate the expectation w.r.t. $q_\psi(\mathbf{s}_t \mid \mathbf{x}, k_{(t+1):T})$—which entangles the different composition steps and makes inference intractable—using $\mathbf{s}_T = \mathbf{1}$ and the updates in (14).

[6] Applying all K experts can be easily parallelised.

[7] To ensure that the entire scene is explained in T steps, we use the final scope $\mathbf{s}_1$ as attention region for all experts in the last inference step $(t = 1)$, as also done in GENESIS and MONET.

Competition objective While model parameters are updated using the learning objective (13) derived from the ELBO, the choice of competition objective is ours. Since we use competition to drive specialisation of experts on different object classes and to greedily infer k_t, (i.e., the identity of the current foreground object), the competition objective should reflect such differences between object classes. Object classes can differ in many ways (shape, color, size, etc) and to different extents, so the choice of competition objective is *data-dependent* and may be informed by prior knowledge.

For instance, in the setting depicted in Figure 1 where both color and shape are class-specific, we found that using a combination of $\hat{\mathcal{L}}_{\mathbf{x},t}$ and $\hat{\mathcal{L}}_{\mathbf{r}_t}$ worked well. However, on the same data with randomised color (as used in the experiments in §6) it did not: due to the greedy optimisation procedure, the expert which is initially best at reconstructing a particular color continues to win the competition for explaining regions of that color and thus receives gradient updates to reinforce this specialisation; such undesired specialisation corresponds to a local minimum in the optimisation landscape and can be very hard for the model to escape.

We thus found that relying solely on $\hat{\mathcal{L}}_{\mathbf{r},t}$ as the competition objective (i.e., the reconstruction of the attention region) helps to direct specialisation towards objects categories. In this case, experts are chosen based on how well they can model *shape*, and only those experts which can easily reconstruct (the shape of) a selected region within the current scope will do well at any given step, meaning that the selected region corresponds to a foreground object.

Moreover, we found that using a stochastic, rather than deterministic form of competition, (i.e., experts win the competition with the probabilities proportional to their competition objectives at a given step) helped specialisation. In particular, such approach helps prevent the collapse of the experts in the initial stages of training.

Formally, the probability of expert k winning the competition is

$$P(\hat{k}_t = k) \propto \exp\left\{-\big(\lambda\hat{\mathcal{L}}_{\mathbf{x},t}(k) + \hat{\mathcal{L}}_{\mathbf{r},t}(k)\big)\right\},$$

with $\hat{\mathcal{L}}_{\mathbf{x},t}(k)$ and $\hat{\mathcal{L}}_{\mathbf{r},t}(k)$ being the terms in (13) at step t for an expert k. The hyper-parameter λ controls the relative influence of the appearance and shape reconstruction objectives to make the *data-dependent* assumptions about the competition mechanism as discussed above.

6 EXPERIMENTAL RESULTS

To explore ECON's ability to decompose and generate new scenes, we conduct experiments on synthetic data consisting of colored 2D objects or sprites (triangles, squares and circles) in different occlusion arrangements. We refer to Appendix C for a detailed account of the used data set, model architecture, choice of hyperparameters, and experimental setting. Further experiments can be found in Appendix B.

ECON decomposes scenes and inpaints occluded objects Fig. 4 shows an example of how ECON decomposes a scene with four objects. At each inference step, the winning expert segments a region (second col.) within the unexplained part of the image (first col.), and reconstructs the selected object within the attended region (fourth col.). A distinctive feature of our model is that, despite occlusion, the full shape (rightmost col.) of every object is imputed (e.g., at step $t' = 4$). This ability to infer complete shapes is a consequence of the assumed layer-wise generative model which manifests itself in our objective via the unconstrained shape reconstruction term (12).

ECON generalizes to novel scenes Fig. 4 also illustrates that that the model is capable of decomposing scenes containing multiple objects of the same category, as well as multiple objects of the same color in separate steps. It does so for a scene with four objects, despite being trained on scenes containing only three objects, one from each class.

Single expert as generative extension of MONET We also investigate training a single expert which we claim to be akin to a generative extension of MONET. When trained on the data from Fig. 1 with ground truth masks provided, the expert learns to inpaint occluded shapes and objects as can be seen from the samples in Fig. 5. However, all object classes are represented in a shared latent space so that different classes cannot be sampled controllably.

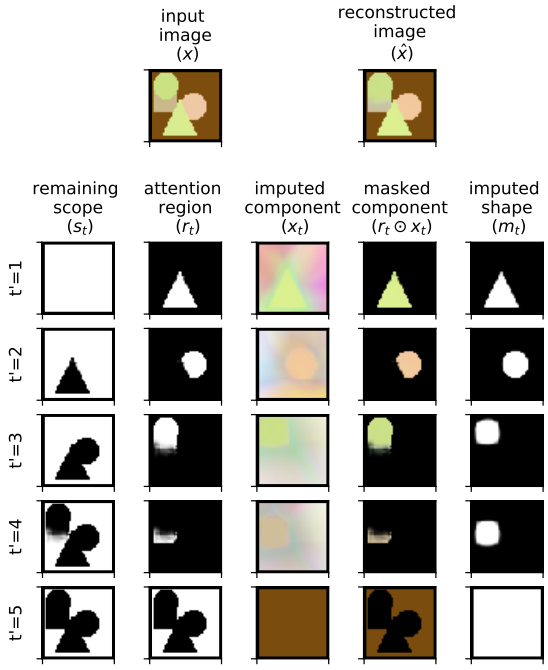


Figure 4: By explaining away a scene *from front to back*, ECON can impute occluded components $\mathbf{x}_t$ (third column) and—crucially for layered generation and recombination—their shapes $\mathbf{m}_t$ (fifth column) within the already explained regions $\mathbf{s}_t$ (first column). Each inference step $(t' = T + 1 - t)$ shows only the winning expert's output.

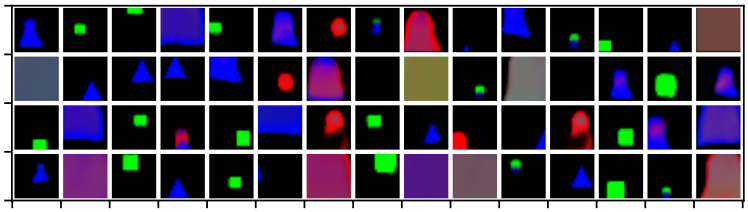


Figure 5: Random samples from a single expert (akin to a generative extension of MONET) trained on the data from Fig. 1 with ground-truth masks provided. The model learns to separately generate unoccluded objects and background, but lacks control over which object class is sampled.

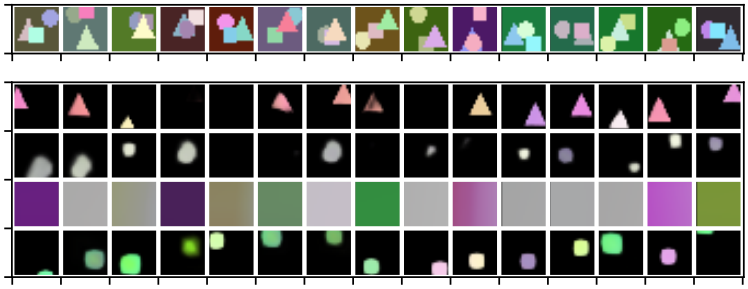


Figure 6: Samples from individual experts trained on toy data with random colors (shown in top panel). Experts (corresponding to rows in the bottom panel) specialise on triangles, circles, background, and squares, respectively, but such specialisation based-purely on shape is significantly harder when color is lost as a powerful cue. This is reflected, e.g., in the imperfect separation between squares and circles, cf. Fig. 1.

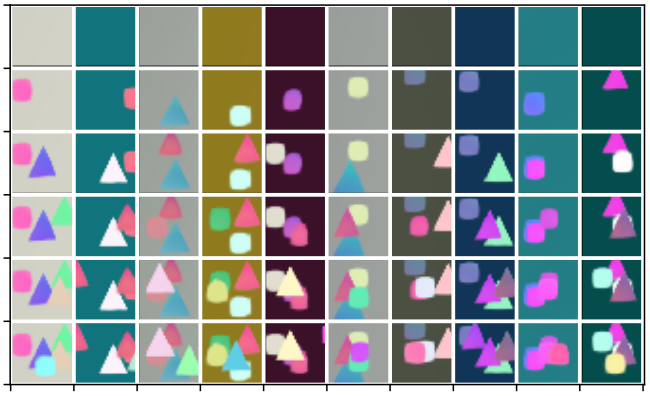


Figure 7: Illustration of layer-wise sampling from ECON after training on our toy data with random colors. Starting with a background sample, subsequent rows correspond to sampling additional objects by randomly choosing one of the specialised object experts.

Multiple experts specialise on different object classes Fig. 1B shows samples from each of the four experts trained on a dataset with uniquely colored objects (Fig. 1A). The samples from each expert contain either the same object in different spatial positions or differently coloured background, indicating that the experts specialised on the different object classes composing these scenes.

Fig. 6 shows the same plot for a model trained on scenes consisting of randomly colored objects. This setting is considerably more challenging because experts have to specialise purely based on shape while also representing color variations. Yet, experts specialise on different object classes: samples in Fig. 6 are either randomly colored background or objects from mostly one class with different colours and spatial positions, indicating that the ECON is capable of representing the scenes as compositions of distinct objects in an unsupervised way.

Controlled and layered generation of new scenes The specialisation of experts allows us to controllably generate new scenes with specific properties. To do so, we follow the sequential generation procedure described in §2 by sampling from one of the experts at each time step. The number of generation steps T, as well as the choices of experts $k_{1:T}$ allow to control the total number and categories of objects in the generated scene.

Fig. 1C shows samples generated using the experts in Fig. 1B. In Fig. 7 we show another example where more and more randomly colored objects are sequentially added. Even though the generated scenes are quite simple, we believe this result is important as the ability to generate scenes in a controlled way is a distinctive feature of our model, which current generative scene models lack.

7 DISCUSSION

Model assumptions While ECON aims at modelling scene composition in a faithful way, we make a number of assumptions for the sake of tractable inference, which need to be revisited when moving to more general environments. We assume a known (maximum) number of object classes K which may be restrictive for realistic settings, and choosing K too small may force each expert to represent multiple object classes. Other assumptions are that the pixel values are modelled as normally distributed, even though they are discrete in the range $\{0, \dots, 255\}$, and that pixels are conditionally independent given shapes and objects.

Shared vs. object-specific representations Recent work on unsupervised representation learning (Bengio et al., 2013) has largely focused on disentangling factors of variation within a single shared representation space, e.g., by training a large encoder-decoder architecture with different forms of regularization (Higgins et al.; Kim & Mnih, 2018; Chen et al., 2018; Locatello et al., 2019). This is motivated by the observation that certain (continuous) attributes such as position, size, orientation or color are general concepts which transcend object-class boundaries. However, the range of values

of these attributes, as well as other (discrete) properties such as shape, can strongly depend on object class. In this work, we investigate the other extreme of this spectrum by learning entirely object-specific representations. Exploring the more plausible middle ground combining both shared and object-specific representations is an attractive direction for further research.

Extensions and future work The goal of decomposing visual scenes into their constituents in an unsupervised manner from images alone will likely remain a long standing goal of visual representation learning. We have presented a model that recombines earlier ideas on layered scene compositions, with more recent models of larger representational power, and unsupervised attention models. The focus of this work is to establish physically plausible compositional models for an easy class of images and to propose a model that naturally captures object-specific specialization.

With ECON and other models as starting point, a number of extensions are possible. One direction of future work deals with incorporating additional information about scenes. Here, we consider static, semantically-free images. Optical flow and depth information can be cues to an attention process, facilitating segmentation and specialization. First results in the direction of video data have been shown by Xu et al. (2019). Natural images typically carry semantic meaning and objects are not ordered in arbitrary configurations. Capturing dependencies between objects (e.g., using an auto-regressive prior over depth ordering as in GENESIS), albeit challenging, could help disambiguate between scene components. Another direction of future work is to relax the unsupervised assumption, e.g., by exploring a semi-supervised approach, which might help improve stability.

On the modelling side, extensions to recurrent architectures and iterative refinement as in IODINE appear promising. Our model entirely separates experts from each other but, depending on object similarity, one can also include shared representations which will help transfer already learned knowledge to new experts in a continual learning scenario.

8 CONCLUSION

While the scenes studied here and in the recent works of Burgess et al. (2019); Greff et al. (2019); Engelcke et al. (2019) are still in stark contrast to the impressive results that holistic generative models are able to achieve, we believe it is the right time to revisit the unsupervised scene composition problem. Our goal is to build re-combineable systems, where different components can be used for new scene inference tasks. In the spirit of the analysis-by-synthesis approach, this requires the ability to re-create physically plausible visual scenes. Disentangling the scene formation process from the objects is one crucial component thereof, and the vast number of object types will require the ability of unsupervised learning from visual input alone.

ACKNOWLEDGEMENTS

The authors would like to thank Alex Smola, Anirudh Goyal, Muhammad Waleed Gondal, Chris Russel, Adrian Weller, Neil Lawrence, and the Empirical Inference "deep learning & causality" team at the MPI for Intelligent Systems for helpful discussions and feedback.

M.B. and B.S. acknowledge support from the German Science Foundation (DFG) through the CRC 1233 "Robust Vision" project number 276693517, the German Federal Ministry of Education and Research (BMBF) through the Tbingen AI Center (FKZ: 01IS18039A), and the DFG Cluster of Excellence "Machine Learning New Perspectives for Science" EXC 2064/1, project number 390727645.

REFERENCES

Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert gate: Lifelong learning with a network of experts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3366–3375, 2017.

Oron Ashual and Lior Wolf. Specifying object attributes and relations in interactive scene generation. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4561–4569, 2019.

David Bau, Jun-Yan Zhu, Jonas Wulff, William Peebles, Hendrik Strobelt, Bolei Zhou, and Antonio Torralba. Seeing what a gan cannot generate. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4502–4511, 2019.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

Thomas G. Bever and David Poeppel. Analysis by synthesis: A (re-)emerging program of research for language and vision. *Biolinguistics*, 4(2):174–200, 2010.

Christopher P Burgess, Loic Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matt Botvinick, and Alexander Lerchner. Monet: Unsupervised scene decomposition and representation. *arXiv preprint arXiv:1901.11390*, 2019.

Tian Qi Chen, Xuechen Li, Roger B Grosse, and David K Duvenaud. Isolating sources of disentanglement in variational autoencoders. In *Advances in Neural Information Processing Systems*, pp. 2610–2620, 2018.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1): 1–22, 1977.

Martin Engelcke, Adam R Kosiorek, Oiwi Parker Jones, and Ingmar Posner. Genesis: Generative scene inference and sampling with object-centric latent representations. *arXiv preprint arXiv:1907.13052*, 2019.

SM Ali Eslami, Nicolas Heess, Theophane Weber, Yuval Tassa, David Szepesvari, Geoffrey E Hinton, et al. Attend, infer, repeat: Fast scene understanding with generative models. In *Advances in Neural Information Processing Systems*, pp. 3225–3233, 2016.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Anirudh Goyal, Alex Lamb, Jordan Hoffmann, Shagun Sodhani, Sergey Levine, Yoshua Bengio, and Bernhard Schölkopf. Recurrent independent mechanisms. *arXiv preprint arXiv:1909.10893*, 2019.

Klaus Greff, Antti Rasmus, Mathias Berglund, Tele Hao, Harri Valpola, and Jürgen Schmidhuber. Tagger: Deep unsupervised perceptual grouping. In *Advances in Neural Information Processing Systems*, pp. 4484–4492, 2016.

Klaus Greff, Sjoerd Van Steenkiste, and Jürgen Schmidhuber. Neural expectation maximization. In *Advances in Neural Information Processing Systems*, pp. 6691–6701, 2017.

Klaus Greff, Raphaël Lopez Kaufman, Rishabh Kabra, Nick Watters, Christopher Burgess, Daniel Zoran, Loic Matthey, Matthew Botvinick, and Alexander Lerchner. Multi-object representation learning with iterative variational inference. In *International Conference on Machine Learning*, pp. 2424–2433, 2019.

Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. In *International Conference on Machine Learning*, pp. 1462–1471, 2015.

Ulf Grenander. Pattern synthesis – lectures in pattern theory. 1976.

Nicolas Manfred Otto Heess. *Learning generative models of mid-level structure in natural images*. PhD thesis, 2012.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework.

Berthold K. P. Horn. Understanding image intensities. *Artifical Intelligence*, 8:201–231, 1977.

Robert A Jacobs and Michael I Jordan. A competitive modular connectionist architecture. In *Advances in neural information processing systems*, pp. 767–773, 1991.

Varun Jampani, Sebastian Nowozin, Matthew Loper, and Peter V. Gehler. The informed sampler: A discriminative approach to bayesian inference in generative computer vision models. *Computer Vision and Image Understanding*, 136:32 – 44, 2015. ISSN 1077-3142.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparametrization with gumbel-softmax. In *International Conference on Learning Representations (ICLR 2017)*. OpenReview. net, 2017.

Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4401–4410, 2019a.

Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. *arXiv preprint arXiv:1912.04958*, 2019b.

Hyunjik Kim and Andriy Mnih. Disentangling by factorising. *arXiv preprint arXiv:1802.05983*, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

Adam Kosiorek, Hyunjik Kim, Yee Whye Teh, and Ingmar Posner. Sequential attend, infer, repeat: Generative modelling of moving objects. In *Advances in Neural Information Processing Systems*, pp. 8606–8616, 2018.

T. D. Kulkarni, P. Kohli, J. B. Tenenbaum, and V. Mansinghka. Picture: A probabilistic programming language for scene perception. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4390–4399, June 2015. doi: 10.1109/CVPR.2015.7299068.

Nicolas Le Roux, Nicolas Heess, Jamie Shotton, and John Winn. Learning a generative model of images by factoring appearance and shape. *Neural Computation*, 23(3):593–650, 2011.

Francesco Locatello, Damien Vincent, Ilya Tolstikhin, Gunnar Rätsch, Sylvain Gelly, and Bernhard Schölkopf. Competitive training of mixtures of independent deep generative models. *arXiv preprint arXiv:1804.11130*, 2018.

Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. In *International Conference on Machine Learning*, pp. 4114–4124, 2019.

C Maddison, A Mnih, and Y Teh. The concrete distribution: A continuous relaxation of discrete random variables. International Conference on Learning Representations, 2017.

Joe Marino, Yisong Yue, and Stephan Mandt. Iterative amortized inference. In *International Conference on Machine Learning*, pp. 3403–3412, 2018.

Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. In *Advances in neural information processing systems*, pp. 2204–2212, 2014.

Giambattista Parascandolo, Niki Kilbertus, Mateo Rojas-Carulla, and Bernhard Schölkopf. Learning independent causal mechanisms. In *International Conference on Machine Learning*, pp. 4036–4044, 2018.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dÁlché Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. 2019.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, pp. 1278–1286, 2014.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi (eds.), *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pp. 234–241, Cham, 2015. Springer International Publishing. ISBN 978-3-319-24574-4.

Mehmet Ozgur Turkoglu, William Thong, Luuk Spreeuwers, and Berkay Kicanaoglu. A layer-based sequential framework for scene generation with gans. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 8901–8908, 2019.

Sjoerd Van Steenkiste, Michael Chang, Klaus Greff, and Jürgen Schmidhuber. Relational neural expectation maximization: Unsupervised discovery of objects and their interactions. *arXiv preprint arXiv:1802.10353*, 2018.

Jiajun Wu, Joshua B Tenenbaum, and Pushmeet Kohli. Neural scene de-rendering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 699–707, 2017.

Zhenjia Xu, Zhijian Liu, Chen Sun, Kevin Murphy, William T Freeman, Joshua B Tenenbaum, and Jiajun Wu. Unsupervised discovery of parts, structure, and dynamics. *arXiv preprint arXiv:1903.05136*, 2019.

Jinyang Yuan, Bin Li, and Xiangyang Xue. Generative modeling of infinite occluded objects for compositional scene representation. In *International Conference on Machine Learning*, pp. 7222–7231, 2019.

A DERIVATIONS

A.1 DERIVATION OF ELBO

We now provide a detailed derivation of the evidence lower bound (ELBO) used in the main paper. For ease of notation we use vector notation and omit explicitly summing over pixel- and latent dimensions (as done in the implementation).

We start by writing $p_\theta(\mathbf{x}|k_{1:T})$ as an expectation w.r.t. q using importance sampling as follows:

$$\begin{aligned} p_\theta(\mathbf{x}|k_{1:T}) &= \mathbb{E}_{p_\theta(\mathbf{r}_{1:T},\mathbf{z}_{1:T}|k_{1:T})}\Big[p_\theta(\mathbf{x}|\mathbf{r}_{1:T},\mathbf{z}_{1:T},k_{1:T})\Big] \\ &= \mathbb{E}_{q_{\phi,\psi}(\mathbf{r}_{1:T},\mathbf{z}_{1:T}|\mathbf{x},k_{1:T})}\left[\frac{p_\theta(\mathbf{x},\mathbf{r}_{1:T},\mathbf{z}_{1:T}|k_{1:T})}{q_{\phi,\psi}(\mathbf{r}_{1:T},\mathbf{z}_{1:T}|\mathbf{x},k_{1:T})}\right]. \end{aligned}$$

Applying the concave function $\log(\,\cdot\,)$ and using Jensen's inequality we obtain

$$\log p_\theta(\mathbf{x}|k_{1:T}) \geq \mathbb{E}_{q_{\phi,\psi}(\mathbf{r}_{1:T},\mathbf{z}_{1:T}|\mathbf{x},k_{1:T})}\left[\log\frac{p_\theta(\mathbf{x},\mathbf{r}_{1:T},\mathbf{z}_{1:T}|k_{1:T})}{q_{\phi,\psi}(\mathbf{r}_{1:T},\mathbf{z}_{1:T}|\mathbf{x},k_{1:T})}\right]. \tag{A.1}$$

Using the chain rule of probability and properties of $\log(\,\cdot\,)$, we can rearrange the integrand on the RHS of (A.1) as

$$\log p_\theta(\mathbf{x}|\mathbf{r}_{1:T},\mathbf{z}_{1:T},k_{1:T}) - \log\frac{q_\psi(\mathbf{r}_{1:T}|\mathbf{x},k_{1:T})}{p_\theta(\mathbf{r}_{1:T}|\mathbf{z}_{1:T},k_{1:T})} - \log\frac{q_\phi(\mathbf{z}_{1:T}|\mathbf{x},\mathbf{r}_{1:T},k_{1:T})}{p_\theta(\mathbf{z}_{1:T}|k_{1:T})}. \tag{A.2}$$

We will consider the three terms in (A.2) separately and define their expectations w.r.t. the approximate posterior as

$$\begin{aligned} \mathcal{L}_\mathbf{x}(\theta,\psi,\phi|k_{1:T}) &:= \mathbb{E}_{q_{\phi,\psi}(\mathbf{r}_{1:T},\mathbf{z}_{1:T}|\mathbf{x},k_{1:T})}\Big[\log p_\theta(\mathbf{x}|\mathbf{r}_{1:T},\mathbf{z}_{1:T},k_{1:T})\Big], \\ \mathcal{L}_\mathbf{r}(\theta,\psi,\phi|k_{1:T}) &:= \mathbb{E}_{q_{\phi,\psi}(\mathbf{r}_{1:T},\mathbf{z}_{1:T}|\mathbf{x},k_{1:T})}\left[-\log\frac{q_\psi(\mathbf{r}_{1:T}|\mathbf{x},k_{1:T})}{p_\theta(\mathbf{r}_{1:T}|\mathbf{z}_{1:T},k_{1:T})}\right], \\ \mathcal{L}_\mathbf{z}(\theta,\psi,\phi|k_{1:T}) &:= \mathbb{E}_{q_{\phi,\psi}(\mathbf{r}_{1:T},\mathbf{z}_{1:T}|\mathbf{x},k_{1:T})}\left[-\log\frac{q_\phi(\mathbf{z}_{1:T}|\mathbf{x},\mathbf{r}_{1:T},k_{1:T})}{p_\theta(\mathbf{z}_{1:T}|k_{1:T})}\right]. \end{aligned}$$

Next, we use our modelling assumptions stated in the paper to simplify these terms, starting with $\mathcal{L}_\mathbf{z}$.

Using the assumed factorisation of the approximate posterior, in particular $q_\psi(\mathbf{r}_t|\mathbf{x},\mathbf{r}_{(t+1):T},k_t) = q_\psi(\mathbf{r}_t|\mathbf{x},\mathbf{s}_t,k_t)$, as well as the fact that $p_\theta(\mathbf{z}_t|k_t) = p(\mathbf{z})$, splitting the expectation into two parts, and using linearity of the expectation operator, we find that $\mathcal{L}_\mathbf{z}$ can be written as follows:

$$\begin{aligned} \mathcal{L}_\mathbf{z}(\psi,\phi|k_{1:T}) &= \mathbb{E}_{q_\psi(\mathbf{r}_{1:T}|\mathbf{x},k_{1:T})}\left[\mathbb{E}_{q_\phi(\mathbf{z}_{1:T}|\mathbf{x},\mathbf{r}_{1:T},k_{1:T})}\left[-\sum_{t=1}^{T}\log\frac{q_\phi(\mathbf{z}_t|\mathbf{x},\mathbf{r}_t,k_t)}{p(\mathbf{z})}\right]\right] \\ &= -\sum_{t=1}^{T}\mathbb{E}_{q_\psi(\mathbf{r}_{t:T}|\mathbf{x},k_{t:T})}\left[\mathbb{E}_{q_\phi(\mathbf{z}_t|\mathbf{x},\mathbf{r}_t,k_t)}\left[\log\frac{q_\phi(\mathbf{z}_t|\mathbf{x},\mathbf{r}_t,k_t)}{p(\mathbf{z})}\right]\right] \\ &= -\sum_{t=1}^{T}\mathbb{E}_{q_\psi(\mathbf{s}_t|\mathbf{x},k_{(t+1):T})}\mathbb{E}_{q_\psi(\mathbf{r}_t|\mathbf{x},\mathbf{s}_t,k_t)}\Big[D_{KL}\Big(q_\phi(\mathbf{z}_t|\mathbf{x},\mathbf{r}_t,k_t)\;\Big\|\;p(\mathbf{z})\Big)\Big]. \end{aligned}$$

Next, we consider $\mathcal{L}_\mathbf{r}$. Using a similar argument as for $\mathcal{L}_\mathbf{z}$, we find that

$$\begin{aligned} \mathcal{L}_\mathbf{r}(\theta,\psi,\phi|k_{1:T}) &= \mathbb{E}_{q_\psi(\mathbf{r}_{1:T}|\mathbf{x},k_{1:T})}\left[\mathbb{E}_{q_\phi(\mathbf{z}_{1:T}|\mathbf{x},\mathbf{r}_{1:T},k_{1:T})}\left[-\sum_{t=1}^{T}\log\frac{q_\psi(\mathbf{r}_t|\mathbf{x},\mathbf{s}_t,k_t)}{p_\theta(\mathbf{r}_t|\mathbf{s}_t,\mathbf{z}_t,k_t)}\right]\right] \\ &= -\sum_{t=1}^{T}\mathbb{E}_{q_\psi(\mathbf{s}_t|\mathbf{x},k_{(t+1):T})}\mathbb{E}_{q_\psi(\mathbf{r}_t|\mathbf{x},\mathbf{s}_t,k_t)}\left[\mathbb{E}_{q_\phi(\mathbf{z}_t|\mathbf{x},\mathbf{r}_t,k_t)}\left[\log\frac{q_\psi(\mathbf{r}_t|\mathbf{x},\mathbf{s}_t,k_t)}{p_\theta(\mathbf{r}_t|\mathbf{s}_t,\mathbf{z}_t,k_t)}\right]\right]. \end{aligned}$$

Finally, we consider $\mathcal{L}_{\mathbf{x}}$. Substituting the Gaussian likelihood for $p_\theta(\mathbf{x}|\mathbf{r}_{1:T}, \mathbf{z}_{1:T}, k_{1:T})$, ignoring constants which do not depend on any learnable parameters, and using the fact that $\mathbf{r}_t$ is binary and $\sum_{t=1}^{T} \mathbf{r}_t = \mathbf{1}$, we obtain

$$\begin{aligned}
\mathcal{L}_{\mathbf{x}}(\theta, \psi, \phi|k_{1:T}) &= \mathbb{E}_{q_\psi(\mathbf{r}_{1:T}|\mathbf{x}, k_{1:T})}\left[\mathbb{E}_{q_\phi(\mathbf{z}_{1:T}|\mathbf{x}, \mathbf{r}_{1:T}, k_{1:T})}\left[-\frac{1}{2\sigma_{\mathbf{x}}^2}\left\|\mathbf{x} - \sum_{t=1}^{T} \mathbf{r}_t \odot \mu_{\theta_{k_t}}(\mathbf{z}_t)\right\|^2\right]\right] \\
&= -\frac{1}{2\sigma_{\mathbf{x}}^2}\mathbb{E}_{q_\psi(\mathbf{r}_{1:T}|\mathbf{x}, k_{1:T})}\left[\mathbb{E}_{q_\phi(\mathbf{z}_{1:T}|\mathbf{x}, \mathbf{r}_{1:T}, k_{1:T})}\left[\sum_{t=1}^{T} \mathbf{r}_t \odot \left\|\mathbf{x} - \mu_{\theta_{k_t}}(\mathbf{z}_t)\right\|^2\right]\right] \\
&= -\sum_{t=1}^{T} \frac{1}{2\sigma_{\mathbf{x}}^2}\mathbb{E}_{q_\psi(\mathbf{s}_t|\mathbf{x}, k_{(t+1):T})}\mathbb{E}_{q_\psi(\mathbf{r}_t|\mathbf{x}, \mathbf{s}_t, k_t)}\left[\mathbf{r}_t \odot \mathbb{E}_{q_\phi(\mathbf{z}_t|\mathbf{x}, \mathbf{r}_t, k_t)}\left[\left\|\mathbf{x} - \mu_{\theta_{k_t}}(\mathbf{z}_t)\right\|^2\right]\right],
\end{aligned}$$

where $\|\cdot\|^2$ denotes the pixel-wise L2-norm between two RGB vectors. (Recall that $\mathcal{L}_{\mathbf{x}}$, $\mathcal{L}_{\mathbf{r}}$, and $\mathcal{L}_{\mathbf{z}}$ are defined as quantities in $\mathbb{R}^D$, $\mathbb{R}^D$, and $\mathbb{R}^L$, respectively, and that summation over these dimensions yields the desired scalar objective.)

We observe that $\mathcal{L}_{\mathbf{x}}$, $\mathcal{L}_{\mathbf{r}}$, and $\mathcal{L}_{\mathbf{z}}$ can all be written as sums over the T composition steps.

We thus define:

$$\begin{aligned}
\mathcal{L}_{\mathbf{x},t}(\theta, \psi, \phi|k_t) &:= \frac{1}{2\sigma_{\mathbf{x}}^2}\mathbb{E}_{q_\psi(\mathbf{r}_t|\mathbf{x}, \mathbf{s}_t, k_t)}\left[\mathbf{r}_t \odot \mathbb{E}_{q_\phi(\mathbf{z}_t|\mathbf{x}, \mathbf{r}_t, k_t)}\left[\left\|\mathbf{x} - \mu_{\theta_{k_t}}(\mathbf{z}_t)\right\|^2\right]\right], \\
\mathcal{L}_{\mathbf{r},t}(\theta, \psi, \phi|k_t) &:= \mathbb{E}_{q_\psi(\mathbf{r}_t|\mathbf{x}, \mathbf{s}_t, k_t)}\left[\mathbb{E}_{q_\phi(\mathbf{z}_t|\mathbf{x}, \mathbf{r}_t, k_t)}\left[\log \frac{q_\psi(\mathbf{r}_t|\mathbf{x}, \mathbf{s}_t, k_t)}{p_\theta(\mathbf{r}_t|\mathbf{s}_t, \mathbf{z}_t, k_t)}\right]\right], \\
\mathcal{L}_{\mathbf{z},t}(\psi, \phi|k_t) &:= \mathbb{E}_{q_\psi(\mathbf{r}_t|\mathbf{x}, \mathbf{s}_t, k_t)}\left[D_{KL}\Big(q_\phi(\mathbf{z}_t|\mathbf{x}, \mathbf{r}_t, k_t) \,\Big\|\, p(\mathbf{z})\Big)\right], \\
\mathcal{L}_t(\theta, \psi, \phi|k_t) &:= -\mathbb{E}_{q_\psi(\mathbf{s}_t|\mathbf{x}, k_{(t+1):T})}\Big(\mathcal{L}_{\mathbf{x},t}(\theta, \psi, \phi|k_t) + \mathcal{L}_{\mathbf{z},t}(\psi, \phi|k_t) + \mathcal{L}_{\mathbf{r},t}(\theta, \psi, \phi|k_t)\Big)
\end{aligned}$$

Finally, it then follows that

$$\mathcal{L}(\theta, \psi, \phi|k_{1:T}) := \sum_{t=1}^{T} \mathcal{L}_t(\theta, \psi, \phi|k_t) \leq \log p_\theta(\mathbf{x}|k_{1:T})$$

A.2 Derivation of generative region distribution

We now derive the distribution in (7). We will use the fact that $\mathbf{r}_t = \mathbf{m}_t \odot \mathbf{s}_t$, and that$\mathbf{s}_t$ can be written as $\mathbf{s}_t = \mathbf{1} - \sum_{t'=t+1}^{T} \mathbf{r}'_t$, as well as the conditional independencies implied by our model, see Figure 2b. Considering the pixel-wise distribution and marginalising over $\mathbf{m}_t$, we obtain:

$$\begin{aligned}
p_\theta\left(\mathbf{r}_t = 1|k_t, \mathbf{z}_t, \mathbf{r}_{(t+1):T}\right) &= \sum_{\mathbf{m}_t=0}^{1} p_\theta\left(\mathbf{r}_t = 1|\mathbf{m}_t, k_t, \mathbf{z}_t, \mathbf{r}_{(t+1):T}\right) p_\theta\left(\mathbf{m}_t|k_t, \mathbf{z}_t, \mathbf{r}_{(t+1):T}\right) \\
&= \sum_{\mathbf{m}_t=0}^{1} p_\theta\left(\mathbf{r}_t = 1|\mathbf{m}_t, \mathbf{s}_t\right) p_\theta\left(\mathbf{m}_t|k_t, \mathbf{z}_t\right) \\
&= 0 + p_\theta\left(\mathbf{r}_t = 1|\mathbf{m}_t = 1, \mathbf{s}_t\right) p_\theta\left(\mathbf{m}_t = 1|k_t, \mathbf{z}_t\right) \\
&= \mathbf{s}_t \odot \tilde{\mathbf{m}}_{\theta_{k_t}}(\mathbf{z}_t).
\end{aligned}$$

Since $\mathbf{r}_t$ is binary, this fully determines its distribution.

B ADDITIONAL EXPERIMENTAL RESULTS

Figure 8 shows four additional examples of ECON decomposing scenes consisting of multiple randomly coloured shapes. The model was trained on the data from Fig. 6, but is able to decompose scenes with five objects (a), multiple occluding objects from the same class (b, c), and objects of similar color to the background (d). Moreover, (b) suggests that additional timesteps ($t' = 6$) are simply ignored if they are not needed.

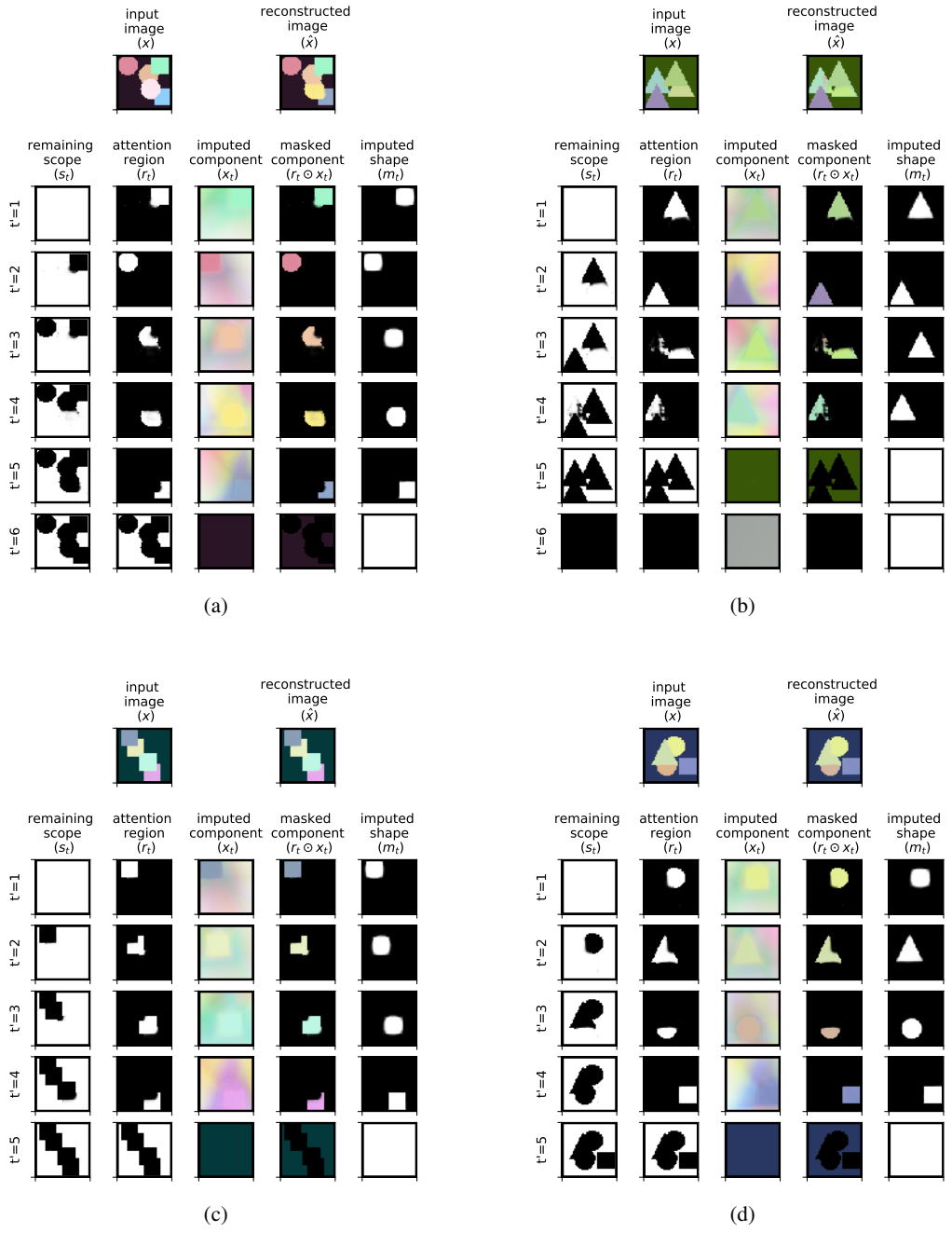


Figure 8: Additional decomposition plots for o.o.d. data. The model was trained with four experts on scenes containing three objcets (one triangle, square, and circle each) arranged in random order.

C EXPERIMENTAL DETAILS

C.1 DATASETS

Synthetic dataset: uniquely colored objects The dataset consists of images of circles, squares and triangles on a randomly and uniformly colored background, such that there is a unique correspondence between object color and class identites (red circles, green squares, blue triangles). The background color is randomly chosen to be an RGB value with each channel being a random integer between 0 and 127, while the RGB values of the object colors are (255,0,0), (0,255,0), (0,0,255) for circles, squares and triangles respectively. The spatial positions of the objects are randomly chosen such that each of the objects entirely fits into an image without crossing the image boundary.

The models shown in Fig. 1 and 5 have been trained on a version of such dataset containing images with exactly three objects per image (one of each class) in random depth orders (Fig. 1, top row). The training and validation splits include $50\,000$ and 100 such images respectively.

Synthetic dataset: randomly colored objects This dataset is the same as the one described above with the difference that the objects (circles, squares and triangles) are randomly colored with the corresponding RGB values being random integers between 128 and 255.

The models shown in Fig. 4, 6 and 8 have been trained on a version of such dataset containing images with exactly three objects per image (one of each class) in random depth orders (Fig. 6, top row). The training and validation splits include 50000 and 100 such images respectively.

C.2 ARCHITECTURE DETAILS

Each expert in our model consists of attention network computing the segmentation regions as a function of the input image and the scope at a given time step, and a VAE reconstructing the image appearance within the segmentation region and inpainting the unoccluded shape of object. Below we describe the details of architectures we used for each of the expert networks.

C.2.1 EXPERT VAES

Encoder The VAE encoder consists of multiple blocks, each of which is composed of 3×3 convolutional layer, ReLU non-linearity, and 2×2 max pooling. The output of the final block is flattened and transformed into a latent space vector by means of two fully connected layers. The output of the first fully-connected layer has 4 times the number of latent dimensions activations, which are passed through the ReLU activation, and finally linearly mapped to the latent vector by a second fully-connected layer.

Decoder Following Burgess et al. (2019), we use spatial a broadcast decoder. First, the latent vector is repeated on a spatial grid of the size of an input image, resulting in a 3D tensor with spatial dimensions being that of an input, and as many feature maps as there are dimensions in the latent space. Second, we concatenate the two coordinate grids (for $x-$ and $y-$coordinates) to this tensor. Next, this tensor is processed by a decoding network consisting of as many blocks as the encoder, with each block including a 3×3 convolutional layer and ReLU non-linearity. Finally, we apply a 1×1 convolutional layer with sigmoid activation to the output of the decoding network resulting in an output of 4 channel (RGB + shape reconstruction).

C.2.2 ATTENTION NETWORK

We use the same attention network architecture as in Burgess et al. (2019) and the implementation provided by Engelcke et al. (2019). It consists of U-Net (Ronneberger et al., 2015) with 4 down and up blocks consisting of a 3×3 convolutional layer, instance normalisation, ReLU activation and down- or up-sampling by a factor of two. The numbers of channels of the block outputs in the down part (the up part is symmetric) of the network are: 4 - 32 - 64 - 64 - 64.

C.3 TRAINING DETAILS

We implemented the model in PyTorch (Paszke et al., 2019). We use the batch size of 32, Adam optimiser (Kingma & Ba, 2014), and initial learning rate of $5 \cdot 10^{-4}$. We compute the validation loss every 100 iterations, and if the validation loss doesn't improve for 5 consecutive evaluations, we decrease the learning rate by a factor of $\sqrt{10}$. We stop the training after 5 learning rate decrease step.

C.4 CROSS-VALIDATION

Synthetic dataset: uniquely colored objects The results in Fig. 1 were obtained by cross-validating 512 randomly sampled architectures with the following ranges of parameters:

Parameter	Range
Latent dimension	2 to 3
Number of layers in encoder and decoder	2 to 4
Number of features per layer in encoder and decoder	4 to 32
β (KL term weight in (12))	0.5 to 2
γ (shape reconstruction weight in (12))	0.1 to 10
Number of experts	4 (three objects + background)
Number of time steps	4 (three objects + background)

The best performing model in terms of the validation loss (which is shown in Fig. 1) has the latent dimension of 2, 4 layers in encoder and decoder, 32 features per layer, $\beta = 9.54$ and $\gamma = 0.52$.

The results in Fig. 5 were obtained using the same model as above but with one expert.

Synthetic dataset: randomly colored objects The results in Figs. 4, 6, and 7 were obtained by cross-validating 512 randomly sampled architectures with the following ranges of parameters:

Parameter	Range
Latent dimension	4 to 5
Number of layers in encoder and decoder	3 to 4
Number of features per layer in encoder and decoder	16 to 32
β (KL term weight in (12))	1
γ (shape reconstruction weight in (12))	0.5 to 5
Number of experts	4 (three objects + background)
Number of time steps	4 (three objects + background)

The best performing model in terms of the validation loss (which is shown in Fig. 1) has the latent dimension of 5, 3 layers in encoder and decoder, 32 features per layer, $\beta = 1$ and $\gamma = 3.26$.

Compositional uncertainty in deep Gaussian processes

Ivan Ustyuzhaninov*
University of Tübingen

Ieva Kazlauskaite*
University of Bath,
Electronic Arts

Markus Kaiser
Siemens AG,
TU Munich

Erik Bodin
University of Bristol

Neill D. F. Campbell[◖]
University of Bath,
Royal Society

Carl Henrik Ek[◖]
University of Bristol

Abstract

Gaussian processes (GPs) are nonparametric priors over functions. Fitting a GP implies computing a posterior distribution of functions consistent with the observed data. Similarly, deep Gaussian processes (DGPs) should allow us to compute a posterior distribution of compositions of multiple functions giving rise to the observations. However, exact Bayesian inference is intractable for DGPs, motivating the use of various approximations. We show that the application of simplifying mean-field assumptions across the hierarchy leads to the layers of a DGP collapsing to near-deterministic transformations. We argue that such an inference scheme is suboptimal, not taking advantage of the potential of the model to discover the compositional structure in the data. To address this issue, we examine alternative variational inference schemes allowing for dependencies across different layers and discuss their advantages and limitations.

1 INTRODUCTION

Hierarchical learning studies functions represented as compositions of other functions, $f = f_L \circ \ldots \circ f_1$. Such models provide a natural way to model data generated by a hierarchical process, as each f_ℓ represents a certain part of the hierarchy, and the prior assumptions on $\{f_\ell\}_{\ell=1}^L$ reflect the corresponding prior assumptions about the data generating process. DGPs (Damianou and Lawrence, 2013), which are compositions of GPs, allow us to impose explicit prior assumptions on $\{f_\ell\}$ by choosing the corresponding kernels. Since different compositions can fit the data equally well (see an illustration in Fig. 1), DGPs are inherently unidentifiable, and this lack of identifiability should be captured by an adequate Bayesian posterior, allowing us to quantify uncertainties pertaining to each f_ℓ. We refer to this uncertainty as *compositional uncertainty*. This uncertainty can be thought of as the epistemic uncertainty (Gal, 2016) describing how the layers of the hierarchy jointly compose the observed data.

While the DGP posterior captures compositional uncertainty, exact Bayesian inference in DGPs is intractable (Damianou and Lawrence, 2013). In this work we show that the typically used approximate inference schemes (*e.g.* Salimbeni and Deisenroth, 2017) impose strong simplifying assumptions, making intermediate DGP layers collapse to deterministic transformations.[1] This corresponds to representing a DGP as a single-layer GP with a transformed kernel (Dunlop et al., 2018), similar to GPs with kernels parametrised by a deterministic function (*e.g.* a neural network). Such behaviour might not be a problem in practice if the goal is to design a model that only provides a high marginal likelihood of the data, however, it does not make full use of the capacity of DGP as it fails to describe the uncertainty that stems from the potential decomposition in the hierarchy. Distributions over compositions, and the resulting compositional uncertainty, are important for applications, *e.g.* for temporal alignment of time series data (Kaiser et al., 2018; Kazlauskaite et al., 2019), in reinforcement learning (Jin et al., 2017) as well as for building more interpretable models where each layer in the hierarchy expresses a meaningful functional prior (Sun et al., 2019).

We address the issue of collapsing compositional uncertainty by proposing variational distributions and corre-

*, ◖ Equal contributions

[1] In cases where the data has high observational noise, the noise is explained by introducing an uncertainty in one or multiple layers of the composition. We focus on the case where the data is noiseless, thus the uncertainty in each of the layers arises only due to the ambiguity in the compositional structure.

Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI), PMLR volume 124, 2020.

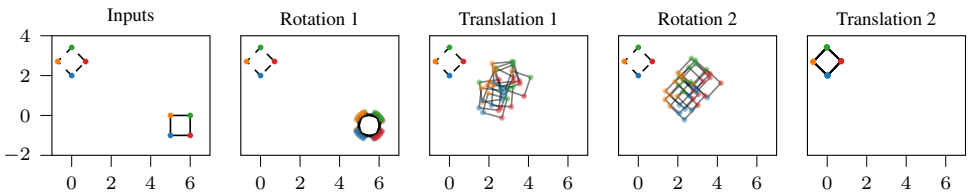


Figure 1: Compositional model (toy example): the transformation of the solid rectangle onto the dashed one is decomposed as $T_2 \circ R_2 \circ T_1 \circ R_1$ where R_i and T_i are rotations and translations. Different sampled realisations of these transformations are overlaid, showing the *compositional uncertainty*. Approximating R_i and T_i as independent transformations does not allow us to capture such uncertainty, collapsing to a single realisation of the composition.

sponding inference methods that explicitly model the dependencies between the layers, resulting in variational posteriors that capture compositional uncertainty. We highlight the limitations of existing approaches and lay the ground for future work in uncertainty quantification in DGPs. Our main contributions are:

- We demonstrate that variational distributions over the inducing points that are factorised *across layers* lead to a collapse of compositional uncertainty,
- We provide an intuitive as well as a quantitative argument for this behaviour by drawing a link between the work on mean-field variational inference for DGPs and the models of regression with noisy inputs (Girard et al., 2003);
- We propose modifications to the factorised variational distribution that incorporate the dependencies between the inducing points in different layers, and discuss the corresponding inference procedures,
- We use the proposed variational inference approaches to further illustrate how the correlations across the layers are necessary in order to argue about compositional uncertainty.

The remainder of the paper is structured as follows. We first provide a background to DGPs with an emphasis on approximate inference and discuss the method of (Salimbeni and Deisenroth, 2017) in detail, using it as the starting point for our argument on the collapse of compositional uncertainty, presented in Sec. 3. In Sec. 4 we propose variational distributions that aims to address the shortcomings of the layer-wise factorisation. In Sec. 5 we illustrate the behaviour of the proposed methods and discuss potential areas of applications.

2 BACKGROUND: MODELS OF DGPs

Previous work The hierarchical GP construction was originally motivated from the perspective of latent variable models (Lawrence, 2004) and was designed with a specific application in mind. In the early work on DGPs, Lawrence and Moore (2007) proposed a model that captured the hierarchical structure in the human skeleton, that allowed to produce interpretable generative models of human motion. However, most of the later work shifted the emphasis from uncovering specific interpretable hierarchical structures to employing a hierarchical construction to design models that are more flexible than a standard GP (in particular, by weakening the assumptions about a joint Gaussian structure in the observations). For example, Lázaro-Gredilla (2012) proposed a hierarchical (two-layer) GP model to allow for non-stationary observations. Damianou and Lawrence (2013) drew further parallels between DGPs and deep belief networks, and proposed a DGP construction beyond two layers for both supervised and unsupervised settings. Concurrently, the MAP estimation used in the early works (Lawrence and Moore, 2007) was replaced with variational inference schemes, initially proposed for the latent variable model (Titsias and Lawrence, 2010) and later adapted for the hierarchical DGP setting (Damianou and Lawrence, 2013).

However, the variational inference approach of Damianou and Lawrence (2013) was shown to be prohibitive for large data sets, motivating further research on inference schemes that scale to large data sets (Hensman et al., 2013; Hensman and Lawrence, 2014; Dai et al., 2016; Bui et al., 2016; Gal and Ghahramani, 2016; Hensman et al., 2017; Salimbeni and Deisenroth, 2017; Rudner and Sejdinovic, 2017; Cutajar, 2019). A different line of thought emerges from the work on inference using stochastic gradient Hamiltonian Monte Carlo (Havasi et al., 2018). The authors recognise the issue of compositional uncertainty, highlighting the fact that most of the existing (variational) approaches to inference are limited to estimating single modes of the posterior distributions in each layer of the hierarchy. As inference using MC is typically very costly, the authors note that it is beneficial to decouple the model in terms of the inducing points for the mean and the variance, which results in a highly non-convex optimization problem that requires careful parameterisation to improve the stability of convergence. Various issues with numerical stabil-

ity, poor convergence and underestimation of uncertainty have also been reported in the context of variational approximations (Hensman and Lawrence, 2014; Kaiser et al., 2018). Duvenaud et al. (2014) show a pathological behaviour of the concentration of density along a single dimension as the number of layers increases, and propose including direct links between the inputs and each individual layer.

Doubly stochastic variational inference (DSVI) Our work builds on the variational approximation scheme introduced by Salimbeni and Deisenroth (2017), thus here we provide a short recap of the main ideas from this work and introduce the notation that is used throughout the rest of this paper. Given a dataset[2] $\mathcal{D} = \{(x_j, y_j)\}_{j=1}^J$, with $x_j, y_j \in \mathbb{R}$, we model $y_j = (f_L \circ \ldots \circ f_1)(x_j)$, where $f_\ell \sim \mathcal{GP}(\mu_\ell(\cdot), k_\ell(\cdot,\cdot))$. We denote the inputs as $\mathbf{x} = (x_1, \ldots, x_J) \in \mathbb{R}^J$, and the evaluations of the intermediate layers at the entire vector of inputs $\mathbf{x}$ as $\mathbf{f}_\ell \sim (f_\ell \circ \ldots \circ f_1)(\mathbf{x})$ for $\ell = 2, \ldots, L$. The DGP joint distribution is

$$p(\mathbf{y}, \mathbf{f}_L, \ldots, \mathbf{f}_1 \mid \mathbf{x}) = p(\mathbf{y} \mid \mathbf{f}_L) \prod_{\ell=1}^{L} p(\mathbf{f}_\ell \mid \mathbf{f}_{\ell-1}), \quad (1)$$

where $p(\mathbf{f}_\ell \mid \mathbf{f}_{\ell-1}) \sim \mathcal{GP}(\mu_j(\mathbf{f}_{\ell-1}), k_j(\mathbf{f}_{\ell-1}, \mathbf{f}_{\ell-1}))$ is a GP prior for the ℓ-th layer, and we define $\mathbf{f}_0 = \mathbf{x}$. Integrating $\{\mathbf{f}_\ell\}$ from (1) to obtain a marginal likelihood is intractable, since that requires integrating a product of Gaussian factors, each of which contains $\mathbf{f}_\ell$ inside a non-linear kernel.

To overcome this limitation, variational inference is used to estimate the lower bound on (1). To this end, each DGP layer ℓ is augmented with M inducing locations $\mathbf{z}_{\ell-1} \in \mathbb{R}^M$ and inducing points $\mathbf{u}_\ell \in \mathbb{R}^M$, resulting in the following augmented joint distribution:

$$\begin{aligned} p(\mathbf{y}, \{\mathbf{f}_\ell\}, \{\mathbf{u}_\ell\} \mid \mathbf{x}, \{\mathbf{z}_\ell\}) &= p(\mathbf{y} \mid \mathbf{f}_L) \times \\ &\times \prod_{\ell=1}^{L} p(\mathbf{f}_\ell \mid \mathbf{f}_{\ell-1}, \mathbf{u}_\ell, \mathbf{z}_{\ell-1}) p(\mathbf{u}_\ell \mid \mathbf{z}_{\ell-1}), \end{aligned} \quad (2)$$

where $p(\mathbf{f}_\ell \mid \mathbf{f}_{\ell-1}, \mathbf{u}_\ell, \mathbf{z}_{\ell-1}) \sim \mathcal{N}(\boldsymbol{\mu}_\ell, \boldsymbol{\Sigma}_\ell)$ is a GP posterior at inputs $\mathbf{f}_{\ell-1}$ given values of $\mathbf{u}_\ell$ at $\mathbf{z}_{\ell-1}$. The specific form of $\boldsymbol{\mu}_\ell$ and $\boldsymbol{\Sigma}_\ell$ is as follows (note a slight abuse of notation: $\mu_\ell(\cdot)$ is a mean function, while $\boldsymbol{\mu}_\ell$ is a posterior mean):

$$\begin{aligned} \boldsymbol{\mu}_\ell &= \mu_\ell(\mathbf{f}_{\ell-1}) + \alpha_\ell(\mathbf{f}_{\ell-1})^T (\mathbf{u}_\ell - \mu_\ell(\mathbf{z}_{\ell-1})), \\ \boldsymbol{\Sigma}_\ell &= k_\ell(\mathbf{f}_{\ell-1}, \mathbf{f}_{\ell-1}) - \alpha_\ell(\mathbf{f}_{\ell-1})^T k_\ell(\mathbf{z}_{\ell-1}, \mathbf{z}_{\ell-1})\, \alpha_\ell(\mathbf{f}_{\ell-1}), \end{aligned}$$

where

$$\alpha_\ell(\mathbf{f}_{\ell-1}) = k_\ell(\mathbf{z}_{\ell-1}, \mathbf{z}_{\ell-1})^{-1} k_\ell(\mathbf{z}_{\ell-1}, \mathbf{f}_{\ell-1}). \quad (3)$$

[2]Throughout the paper we consider one-dimensional data but the general considerations also apply in many dimensions.

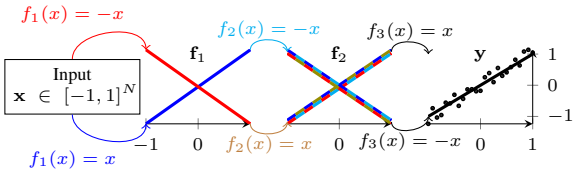


Figure 2: A toy example illustrating three layer compositions where each layer is either $f_\ell(x) = x$ or $f_\ell(x) = -x$. Multiple compositions map $\mathbf{x}$ to $\mathbf{y}$, this uncertainty is illustrated by showing the range of different values of $\mathbf{f}_1 = f_1(\mathbf{x})$ and $\mathbf{f}_2 = f_2(\mathbf{f}_1)$. If a variational distribution over $\{f_\ell\}$ is factorised, the posterior compositions collapse to a single realisation.

Introducing a factorised variational distribution over the inducing points

$$q(\{\mathbf{u}_\ell\}) = q(\mathbf{u}_1) \ldots q(\mathbf{u}_L), \quad q(\mathbf{u}_\ell) \sim \mathcal{N}(\mathbf{m}_\ell, \mathbf{S}_\ell) \quad (4)$$

the likelihood lower bound is as follows:

$$\begin{aligned} \mathcal{L}(\mathbf{y}) \geq \mathbb{E}_{q(\mathbf{f}_L)}[\log p(\mathbf{y} \mid \mathbf{f}_L)] - \\ - \sum_{\ell=1}^{L} \mathrm{KL}[q(\mathbf{u}_\ell) \,||\, p(\mathbf{u}_\ell \mid \mathbf{z}_{\ell-1})]. \end{aligned} \quad (5)$$

A key insight of Salimbeni and Deisenroth (2017) is that the expectation in (5) can be efficiently estimated by a Monte-Carlo estimator. This is possible by marginalising the inducing points $\{\mathbf{u}_\ell\}$ from the variational posterior, obtaining

$$\begin{aligned} q(\mathbf{f}_L, \ldots, \mathbf{f}_1) &= \prod_{\ell=1}^{L} \int p(\mathbf{f}_\ell \mid \mathbf{u}_\ell) q(\mathbf{u}_\ell)\, \mathrm{d}\mathbf{u}_\ell \\ &= q(\mathbf{f}_L \mid \mathbf{f}_{L-1}) \ldots q(\mathbf{f}_1 \mid \mathbf{x}), \end{aligned} \quad (6)$$

with $q(\mathbf{f}_\ell \mid \mathbf{f}_{\ell-1}) \sim \mathcal{N}(\tilde{\boldsymbol{\mu}}_\ell, \tilde{\boldsymbol{\Sigma}}_\ell)$, where

$$\tilde{\boldsymbol{\mu}}_\ell = \mu_\ell(\mathbf{f}_{\ell-1}) + \alpha_\ell(\mathbf{f}_{\ell-1})^T (\mathbf{m}_\ell - \mu_\ell(\mathbf{z}_{\ell-1})), \quad (7)$$

$$\begin{aligned} \tilde{\boldsymbol{\Sigma}}_\ell &= k_\ell(\mathbf{f}_{\ell-1}, \mathbf{f}_{\ell-1}) \\ &\quad - \alpha_\ell(\mathbf{f}_{\ell-1})^T (k_\ell(\mathbf{z}_{\ell-1}, \mathbf{z}_{\ell-1}) - \mathbf{S}_\ell)\, \alpha_\ell(\mathbf{f}_{\ell-1}). \end{aligned} \quad (8)$$

The bound in (5) can be estimated by sequentially sampling from $q(\mathbf{f}_\ell \mid \mathbf{f}_{\ell-1})$ using (7) and (8). The time complexity of this step is linear in the number of data points, since each marginal $[\mathbf{f}_\ell]_j$ can be drawn independently (we only need marginals of the final layer $\mathbf{f}_L$ in (5)).

3 MEAN-FIELD DGPs

In this section we argue that factorised variational distributions of inducing points, *e.g.* (4), imply that the layers in a DGP collapse to deterministic transformations.

3.1 INTUITION

If a DGP $f_L \circ \ldots \circ f_1$ maps fixed inputs $\mathbf{x}$ to fixed outputs $\mathbf{y}$, the functions $\{f_\ell\}$ must be dependent, because every realisation of this composition must map the *same* $\mathbf{x}$ to the *same* $\mathbf{y}$. This is illustrated in Fig. 2, which shows a composition of three layers, each of which could either be $f_\ell(x) = x$ or $f_\ell(x) = -x$. Depending on the choices of f_1 and f_2, the input is mapped by $f_2 \circ f_1$ to one of the two realisations of $\mathbf{f}_2$ (as shown by the colour code in the corresponding panel), and f_3 must be chosen in such a way that $\mathbf{f}_2$ is mapped to $\mathbf{y}$. Therefore, in this example, f_3 depends on the choice of f_1 and f_2. However, if $\{f_\ell\}$ were independent, then the only way to ensure that every realisation of the composition fits the data would be for each layer to implement a deterministic transformation (*i.e.* either $f_\ell(x) = x$ or $f_\ell(x) = -x$ such that there are zero or two instances of $f_\ell(x) = -x$). Another illustration of this idea is provided in Fig. 1, in which movement of a square is represented as a composition of correlated rotations and translations, allowing us to see a variety of possible movements. However, a model with independent transformations would converge to a single possible sequence of rotations and translations.

The same intuition holds for general DGPs. Analogously to choosing either x or $-x$ in Fig. 2, inducing locations $\mathbf{z}_\ell$ and points $\mathbf{u}_\ell$ define the transformation implemented by the corresponding layer through the predictive posterior $p(\mathbf{f}_\ell \mid \mathbf{f}_{\ell-1}, \mathbf{u}_\ell, \mathbf{z}_{\ell-1})$. Following a similar argument, the DGP layers collapse to deterministic transformations to ensure good data fits unless they are dependent to allow multiple different compositions to fit the data.

3.2 QUANTITATIVE ARGUMENT

Assume that the DGP layers $\{f_\ell\}$ are independent. Then the distribution of the outputs of layer $\ell - 1$ can be thought of as uncertain inputs[3] to the layer ℓ. Similarly to DGPs, the inference in such models is complicated by the need to propagate a distribution through a non-linear mapping. Such models have been studied in the context of GP regression (Girard et al., 2003; Mchutchon and Rasmussen, 2011; Bijl, 2018) and have also been discussed in relation to DGPs (Damianou, 2015), though not in the context of compositional uncertainty.

Assuming, for simplicity, that our dataset consists of a single point, *i.e.* $\mathcal{D} = \{(x, y)\}$, we can write $\mathbf{f}_\ell = (f_\ell \circ \ldots \circ f_1)(x) = f_\ell(\mathbf{f}_{\ell-1}) = f_\ell(\bar{\mathbf{f}}_{\ell-1} + \varepsilon_{\ell-1})$, with $\bar{\mathbf{f}}_{\ell-1}$ as the mean[4] of $\mathbf{f}_{\ell-1}$ and $\varepsilon_{\ell-1}$ as an appropriate zero-mean noise (not necessarily Gaussian, since marginals of DGP layers are not Gaussian in general (Damianou, 2015)), the variance of which we denote as $\boldsymbol{\sigma}^2_{\text{noise}} := \mathbb{V}\text{ar}\left[\varepsilon_{\ell-1}\right]$. We want to show that the variance of $\mathbf{f}_\ell$ increases with increasing variance of $\varepsilon_{\ell-1}$, which would imply that unless the layers collapse, *i.e.* $\varepsilon_{\ell-1} = 0$, there is finite variance in the final layer $\mathbf{f}_L$. That constitutes a poor fit to observations that contain low observational noise (noiseless in the limit), forcing the layers to collapse to deterministic transformations.

High observational noise might lead to the layers not collapsing despite being independent. However, such uncertainty is the observational noise spread across the layers, rather than compositional uncertainty due to multiple compositions explaining the data. To make our arguments clearer, we assume noiseless observations.

Linear approximation We can approximate $\mathbf{f}_\ell$ as

$$\mathbf{f}_\ell = f_\ell(\mathbf{f}_{\ell-1}) \approx f_\ell(\bar{\mathbf{f}}_{\ell-1}) + \varepsilon_{\ell-1}\, f'_\ell(\bar{\mathbf{f}}_{\ell-1}), \qquad (9)$$

where $f_\ell(\bar{\mathbf{f}}_{\ell-1}) \sim p(\mathbf{f}_\ell \mid \bar{\mathbf{f}}_{\ell-1}, \mathbf{u}_\ell, \mathbf{z}_{\ell-1}) = \mathcal{N}(\bar{\boldsymbol{\mu}}_\ell, \bar{\boldsymbol{\sigma}}^2_\ell)$, with $\bar{\boldsymbol{\mu}}_\ell$ and $\bar{\boldsymbol{\sigma}}^2_\ell$ given in (7) and (8). Note that both $\bar{\boldsymbol{\mu}}_\ell$ and $\bar{\boldsymbol{\sigma}}^2_\ell$ are functions of $\bar{\mathbf{f}}_{\ell-1}$, which we omit to not clutter the notation; the derivatives below are taken w.r.t. $\bar{\mathbf{f}}_{\ell-1}$.

The evaluation of a GP and its derivative are jointly distributed as follows (Rasmussen and Williams, 2005):

$$\begin{bmatrix} f_\ell(\bar{\mathbf{f}}_{\ell-1}) \\ f'_\ell(\bar{\mathbf{f}}_{\ell-1}) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \bar{\boldsymbol{\mu}}_\ell \\ \bar{\boldsymbol{\mu}}'_\ell \end{bmatrix}, \begin{bmatrix} \boldsymbol{\sigma}^2_\ell & (\boldsymbol{\sigma}^2_\ell)' \\ (\boldsymbol{\sigma}^2_\ell)' & (\boldsymbol{\sigma}^2_\ell)'' \end{bmatrix} \right). \qquad (10)$$

Similarly to Mchutchon and Rasmussen (2011), we compute a linear transformation of (10) and obtain that

$$\begin{aligned} \mathbb{E}\left[\mathbf{f}_\ell \mid \varepsilon_{\ell-1}\right] &= \bar{\boldsymbol{\mu}}_\ell + \varepsilon_{\ell-1}\bar{\boldsymbol{\mu}}'_\ell, \\ \mathbb{V}\text{ar}\left[\mathbf{f}_\ell \mid \varepsilon_{\ell-1}\right] &= \boldsymbol{\sigma}^2_\ell - 2\varepsilon_{\ell-1}(\boldsymbol{\sigma}^2_\ell)' + \varepsilon^2_{\ell-1}(\boldsymbol{\sigma}^2_\ell)''. \end{aligned}$$

Using the law of total variance we have

$$\mathbb{V}\text{ar}\left[\mathbf{f}_\ell\right] = \mathbb{E}\left[\mathbb{V}\text{ar}\left[\mathbf{f}_\ell|\varepsilon_{\ell-1}\right]\right] + \mathbb{V}\text{ar}\left[\mathbb{E}\left[\mathbf{f}_\ell|\varepsilon_{\ell-1}\right]\right],$$

where

$$\begin{aligned} \mathbb{E}\left[\mathbb{V}\text{ar}\left[\mathbf{f}_\ell|\varepsilon_{\ell-1}\right]\right] &= \boldsymbol{\sigma}^2_\ell + \boldsymbol{\sigma}^2_{\text{noise}}\,(\boldsymbol{\sigma}^2_\ell)'', \\ \mathbb{V}\text{ar}\left[\mathbb{E}\left[\mathbf{f}_\ell|\varepsilon_{\ell-1}\right]\right] &= \mathbb{V}\text{ar}\left[\bar{\boldsymbol{\mu}}_\ell + \varepsilon_{\ell-1}\,\bar{\boldsymbol{\mu}}'_\ell\right] = \boldsymbol{\sigma}^2_{\text{noise}} \cdot \left(\bar{\boldsymbol{\mu}}'_\ell\right)^2. \end{aligned}$$

Combining these results together we obtain

$$\mathbb{V}\text{ar}\left[\mathbf{f}_\ell\right] = \boldsymbol{\sigma}^2_\ell + \boldsymbol{\sigma}^2_{\text{noise}}\left[\left(\bar{\boldsymbol{\mu}}'_\ell\right)^2 + (\boldsymbol{\sigma}^2_\ell)''\right] + O(\varepsilon^2_{\ell-1}). \qquad (11)$$

The only term in (11) that can be negative is $(\boldsymbol{\sigma}^2_\ell)''$, potentially making the variance of the GP output at a noisy input smaller that the variance at a fixed input (i.e. $\boldsymbol{\sigma}^2_\ell$).

[3] Regression models that include input uncertainty can generally be formulated as: $\mathbf{y} = f(\mathbf{x} + \varepsilon_{\mathbf{x}})$, where $\mathbf{y}$ are observations, $\mathbf{x}$ are noise-free inputs and $\varepsilon_{\mathbf{x}}$ is zero-mean noise.

[4] We use bold notation for $\bar{\mathbf{f}}_{\ell-1}$, even though it refers to a scalar, to distinguish it from the notation we use for functions.

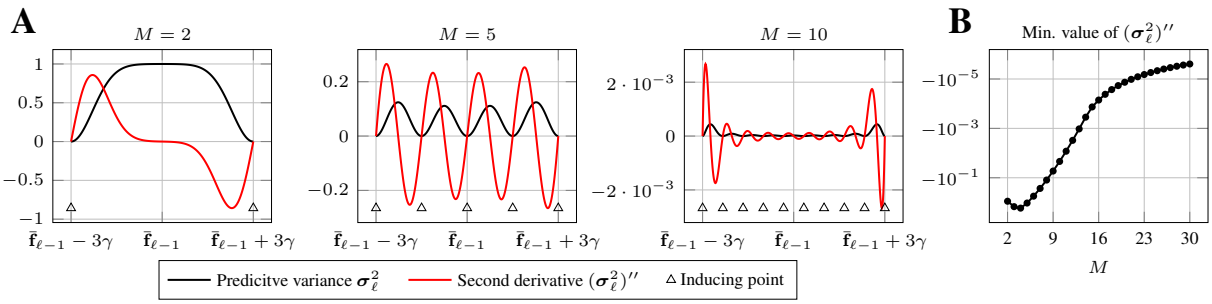


Figure 3: **A:** Predictive posterior variance $\boldsymbol{\sigma}_\ell^2$ and its second derivative $(\boldsymbol{\sigma}_\ell^2)''$ of ℓ-th layer in a $3\gamma_\ell$-neighbourhood Δ_ℓ of the noiseless input $\bar{\mathbf{f}}_{\ell-1}$ for different numbers of inducing points. **B:** Minimum value of $(\boldsymbol{\sigma}_\ell^2)''$ as a function of number of inducing points M.

Counterexample Such an example can indeed be constructed. Girard et al. (2003) study GPs with uncertain inputs and compute an exact expression for $\mathbb{V}\mathrm{ar}\left[\mathbf{f}_\ell\right]$ as a function of $\boldsymbol{\sigma}^2_{\text{noise}}$ assuming $\varepsilon_{\ell-1}$ is Gaussian. Assuming there is a single inducing point $\mathbf{u}_\ell = 0$, the derivative of $\mathbb{V}\mathrm{ar}\left[\mathbf{f}_\ell\right]$ is negative at 0 provided that $\bar{\mathbf{f}}_{\ell-1}$ is sufficiently far away from $\mathbf{u}_\ell$ (in comparison to the length scale; see the derivation in the Supplement). This means that the input noise might *reduce* the output variance. However, such an example relies on the inputs to the ℓ-th layer, $\bar{\mathbf{f}}_{\ell-1}$, appearing in the regions of the input space that are poorly covered by the inducing points. Consequently, this scenario only occurs if the inducing points are placed in a way that leads to a poor fit of the observed data.

Inducing points limit The counterexample above relies on a degenerate setting in which the inducing points are far from the observations. Here we consider a limiting case that corresponds to a more realistic situation of sufficiently many inducing points near $\mathbf{f}_\ell$ (the limit of arbitrary many inducing points is a conceptually desirable setting, complicated by the computational constraints). Specifically, in each layer we assume M linearly spaced inducing points $\mathbf{z}_\ell = \{z^1_{\ell-1}, \ldots, z^M_{\ell-1}\}$ in $\Delta_\ell := [\bar{\mathbf{f}}_{\ell-1} - 3\gamma_\ell, \bar{\mathbf{f}}_{\ell-1} + 3\gamma_\ell]$, where γ_ℓ is the kernel length scale in layer ℓ. This assumption means that the input to each layer is contained in an interval Δ_ℓ covered by the inducing points.

The behaviour of (11) under such an assumption is illustrated in Fig. 3. The minimum value of $(\boldsymbol{\sigma}_\ell^2)''$ approaches zero as M increases; this suggests that the input noise leads to increased predictive posterior variance apart from degenerate cases of inducing points not covering the input region corresponding to the observed inputs $\mathbf{x}$, and the predictive mean derivative $(\bar{\boldsymbol{\mu}}'_\ell)^2$ being sufficiently small (*i.e.* the function implemented by the ℓ-th layer being close to a constant one).

To summarise, we argue that under the assumption of $\mathbf{f}_\ell = (f_\ell \circ \ldots \circ f_1)(x)$ being contained in an interval covered by the inducing locations $\mathbf{z}_\ell$ for the next layer, the variance in $\mathbf{f}_\ell$ leads to increased variance in $\mathbf{f}_{\ell+1}$, and hence in $\mathbf{f}_L$. Therefore, for $\mathbf{f}_L$ to fit a noiseless observation y, the variance in intermediate layers has to be reduced, implying that the layers collapse to deterministic transformations.

4 BEYOND FACTORISED VARIATIONAL DISTRIBUTIONS

To further investigate the effect of the factorisation imposed by the mean-field variational inference, we propose two alternative variational inference schemes that allow for correlations between layers. By relaxing the mean-field assumption across layers, we aim to uncover a range of solutions that are consistent with the data and follow the prior belief about each of the individual layers. In Sec. 4.1 we present a natural generalisation of a factorised variational distribution (4) to capture the marginal dependencies between the layers. In Sec. 4.2 we present an alternative variational approximation that introduces dependencies between the layers by linking the inducing points and locations of the neighbouring layers.

4.1 JOINTLY GAUSSIAN INDUCING POINTS

A straightforward modification of the DSVI variational approximation (Sec. 2) allowing us to capture the dependencies between the layers is to introduce correlations between the inducing points by modelling them with a jointly Gaussian variational distribution:

$$q(\mathbf{u}_1, \ldots, \mathbf{u}_L) \sim \mathcal{N}(\mathbf{m}, \mathbf{S}), \tag{12}$$

with $\mathbf{m} \in \mathbb{R}^{LM}$, $\mathbf{S} \in \mathbb{R}^{LM \times LM}$. The variational posterior is then given by

$$q(\{\mathbf{f}_\ell\}, \{\mathbf{u}_\ell\}) = q(\mathbf{u}_1, \ldots, \mathbf{u}_L) \prod_{\ell=1}^{L} p(\mathbf{f}_\ell | \mathbf{f}_{\ell-1}, \mathbf{u}_\ell). \tag{13}$$

The corresponding likelihood lower bound has the same structure as (5) with the KL term, $\mathrm{KL}[q(\mathbf{u}_1, \ldots, \mathbf{u}_L) \,||\, p(\mathbf{u}_1) \ldots p(\mathbf{u}_L)]$, that can be computed in closed form (it involves two Gaussians). The expectation $\mathbb{E}_{q(\mathbf{f}_L)}[\log p(\mathbf{y} \mid \mathbf{f}_L)]$ is, however, harder to estimate in case of variational distribution (12). The integral (6) no longer factorises into a product of integrals, which means that we can no longer integrate $\{\mathbf{u}_\ell\}$ out from $q(\{\mathbf{f}_\ell\}, \{\mathbf{u}_\ell\})$ and draw samples from $q(\mathbf{f}_L)$ in the same way as in (Salimbeni and Deisenroth, 2017). We consider two approaches to address this issue.

Sampling $\{\mathbf{u}_\ell\}$ We start by noting that, conditioned on $\{\mathbf{u}_\ell\}$, we can draw samples from $q(\mathbf{f}_L)$ in the same way as in (Salimbeni and Deisenroth, 2017). Specifically, to estimate $\mathbb{E}_{q(\mathbf{f}_L)}[\log p(\mathbf{y} \mid \mathbf{f}_L)]$, we

1. Draw S samples $\{(\mathbf{u}_1^s, \ldots, \mathbf{u}_L^s)\}_{s=1}^S \overset{iid}{\sim} q(\mathbf{u}_1, \ldots, \mathbf{u}_L)$,
2. For each sample $(\mathbf{u}_1^s, \ldots, \mathbf{u}_L^s)$, draw $\mathbf{f}_L^s \sim q(\mathbf{f}_L \mid \mathbf{u}_1^s, \ldots, \mathbf{u}_L^s)$ by recursively drawing from $p(\mathbf{f}_\ell \mid \mathbf{f}_{\ell-1}, \mathbf{u}_\ell^s)$, which are regular GP posterior distributions conditioned on $\mathbf{u}_\ell^s$,
3. Compute a Monte Carlo estimate $\mathbb{E}_{q(\mathbf{f}_L)}[\log p(\mathbf{y} \mid \mathbf{f}_L)] \approx \frac{1}{S} \sum_s \log p(\mathbf{y} \mid \mathbf{f}_L^s)$.

This approach is easy to implement and it can be applied in a variety of settings (*e.g.* when $q(\{\mathbf{u}_i\})$ is not Gaussian, as long as we can sample from it and reparametrise the gradients). However, that comes at the cost of introducing another sampling step, resulting in $\mathbb{E}_{q(\mathbf{f}_L)}[\log p(\mathbf{y} \mid \mathbf{f}_L)]$ being estimated by two nested Monte-Carlo estimators, implying an increased overall variance of the estimator and the need to carefully choose the appropriate number of samples (Rainforth et al., 2019). Estimating the variance implied by the nested MC estimator offers a direction for future work. Moreover, drawing coherent samples from $q(\mathbf{u}_1, \ldots, \mathbf{u}_L)$ has computational complexity of $O(L^3M^3)$ leading to an overall complexity of $O(L^3M^3 + LNM^2)$ per estimation.

Analytic marginalisation To address statistical and computational limitations of the above method, we propose another approach consisting of analytically integrating $\{\mathbf{u}_\ell\}$ from (13). To do so we assume that $q(\{\mathbf{u}_\ell\})$ admits a chain-like factorisation, namely

$$q(\{\mathbf{u}_\ell\}) = q(\mathbf{u}_L \mid \mathbf{u}_{L-1}) \ldots q(\mathbf{u}_2 \mid \mathbf{u}_1) q(\mathbf{u}_1). \quad (14)$$

The precision matrix across all layers, $\Lambda = \mathbf{S}^{-1} \in \mathbb{R}^{LM \times LM}$, encodes the conditional independence assumptions, and (14) implies that such matrix is block-tridiagonal (Fig. 4). The advantage of this assumption is that the number of parameters in the unconstrained $\mathbf{S}$ scales quadratically with the number of layers, while (14) implies a linear growth.

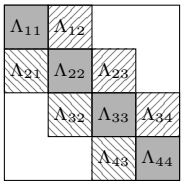


Figure 4: Precision matrix Λ induced by (14).

Assuming that the variational distribution (12) satisfies the factorisation (14), we analytically marginalise $\{\mathbf{u}_\ell\}$ from the variational posterior (13), obtaining

$$\int q(\{\mathbf{f}_\ell\}, \{\mathbf{u}_\ell\}) \, \mathrm{d}\{\mathbf{u}_\ell\} = \prod_{\ell=1}^{L} p(\mathbf{f}_\ell \mid \mathbf{f}_{\ell-1}, \ldots \mathbf{f}_1, \mathbf{x}), \quad (15)$$

where $p(\mathbf{f}_\ell \mid \mathbf{f}_{\ell-1}, \ldots \mathbf{f}_\ell, \mathbf{x}) \sim \mathcal{N}(\tilde{\boldsymbol{\mu}}_\ell, \tilde{\boldsymbol{\Sigma}}_\ell)$ with the mean and the covariance are defined recursively as follows:

$$\tilde{\boldsymbol{\mu}}_1 = \mu_1(\mathbf{x}) + \alpha_1(\mathbf{x})^T(\mathbf{m}_1 - \mu_1(\mathbf{z}_0))$$
$$\tilde{\boldsymbol{\Sigma}}_1 = k_1(\mathbf{x}, \mathbf{x}) - \alpha_1(\mathbf{x})^T(k_1(\mathbf{z}_0, \mathbf{z}_0) - \mathbf{S}_{11})\,\alpha_1(\mathbf{x})$$

and $\alpha_1(\mathbf{x})$ is defined in (3). For $i > 1$, $\tilde{\mu}_i$ and $\tilde{\Sigma}_i$ are recursively defined as

$$\begin{aligned}\tilde{\boldsymbol{\mu}}_\ell = \mu_\ell(\mathbf{f}_\ell) + \alpha_\ell(\mathbf{f}_{\ell-1})^T(\mathbf{m}_\ell + \mathbf{S}_{\ell,\ell-1}\,\alpha_{\ell-1}(\mathbf{f}_{\ell-1}) \times \\ \times \tilde{\boldsymbol{\Sigma}}_{\ell-1}^{-1}(\mathbf{f}_{\ell-1} - \tilde{\boldsymbol{\mu}}_{\ell-1} - \alpha_{\ell-1}(\mathbf{x})^T \times \\ \times (\mathbf{m}_{\ell-1} - \mu_{\ell-1}(\mathbf{z}_{\ell-2})) - \mu_\ell(\mathbf{z}_{\ell-1})),\end{aligned} \quad (16)$$

$$\begin{aligned}\tilde{\boldsymbol{\Sigma}}_\ell = k_\ell(\mathbf{f}_{\ell-1}, \mathbf{f}_{\ell-1}) - \alpha_\ell(\mathbf{f}_{\ell-1})^T(k_\ell(\mathbf{z}_{\ell-1}, \mathbf{z}_{\ell-1}) - \\ - \mathbf{S}_{\ell\ell} + \mathbf{S}_{\ell,\ell-1}\,\alpha_{\ell-1}(\mathbf{f}_{\ell-1})\tilde{\boldsymbol{\Sigma}}_{\ell-1}^{-1} \times \\ \times \alpha_{\ell-1}(\mathbf{f}_{\ell-1})^T\mathbf{S}_{\ell-1,\ell})\alpha_\ell(\mathbf{f}_{\ell-1}),\end{aligned} \quad (17)$$

where $\mathbf{S}_{ij} = \mathrm{cov}(\mathbf{u}_i, \mathbf{u}_j)$.

The derivation is provided in the Supplement. Using these results, $\mathbb{E}_{q(\mathbf{f}_L)}[\log p(\mathbf{y} \mid \mathbf{f}_L)]$ can be estimated analogously to DSVI by recursively sampling $\mathbf{f}_i$ using (15).

4.2 INDUCING POINTS AS INDUCING LOCATIONS

In this section we discuss an alternative variational approximation, that connects the inducing points and the inducing locations of the neighbouring layers. Instead of directly modelling the inducing points in every layer, we only consider the inducing inputs $\mathbf{z}$ in the first layer and variational distributions over $\{\mathbf{f}_\ell^{\mathbf{z}} \sim (f_\ell \circ \ldots \circ f_1)(\mathbf{z})\}$. The advantage of such an approach is that unlike the variational distributions of inducing points, the factorisation of a variational distribution over $\{\mathbf{f}_i^{\mathbf{z}}\}$ does not imply that the variational posterior collapses to a single realisation of a composition fitting the data. In such a setting, $\mathbf{f}_{\ell-1}^{\mathbf{z}}$ and $\mathbf{f}_\ell^{\mathbf{z}}$ can be thought of as inducing pairs of the ℓ-th layer, meaning that the inducing points of a previous layer are the inducing locations of the next one.

Intuition Let us revisit the illustration given in Fig. 2. Assuming for this example that $\mathbf{z} = \mathbf{x}$, we independently sample values of $\mathbf{f}_1$ and $\mathbf{f}_2$ from $q(\mathbf{f}_1)q(\mathbf{f}_2)$ (*i.e.* one of the two types of coloured lines in panels $\mathbf{f}_1$ and $\mathbf{f}_2$). Given such a sample, we can deduce the functions f_1, f_2, f_3. For example, the colour of $\mathbf{f}_1$ denotes the choice of f_1, the second colour of $\mathbf{f}_2$ (the first colour is that of f_1) corresponds to f_2, and f_3 is chosen to map $\mathbf{f}_2$ to the observations. Thus each sample from $q(\mathbf{f}_1)q(\mathbf{f}_2)$ corresponds to a composition mapping $\mathbf{x}$ to $\mathbf{y}$ (different samples correspond to different compositions). This is in contrast to sampling from the factorised distribution of the inducing points (which directly parametrise each $\{f_i\}$). In such case, some compositions (*e.g.* $f_1(x) = -x, f_2(x) = f_3(x) = x$) do not fit the data, making the variational posterior collapse, as argued in Sec. 3.

Inducing inputs We introduce inducing inputs $\mathbf{z} \in \mathbb{R}^M$ (with $M < N$) in the input space and denote the evaluations of intermediate layers at $\mathbf{z}$ as $\mathbf{f}_\ell^\mathbf{z} \sim (f_\ell \circ \ldots \circ f_1)(\mathbf{z})$. The augmented DGP joint distribution is

$$p(\mathbf{y}, \mathbf{f}_L, \ldots, \mathbf{f}_1, \mathbf{f}_L^\mathbf{z}, \ldots, \mathbf{f}_1^\mathbf{z} \mid \mathbf{x}, \mathbf{z}) = \tag{18}$$
$$= p(\mathbf{y} \mid \mathbf{f}_L) \prod_{\ell=1}^{L} p(\mathbf{f}_\ell \mid \mathbf{f}_\ell^\mathbf{z}, \mathbf{f}_{\ell-1}, \mathbf{f}_{\ell-1}^\mathbf{z}) p(\mathbf{f}_\ell^\mathbf{z} \mid \mathbf{f}_{\ell-1}^\mathbf{z}),$$

where $p(\mathbf{f}_\ell^\mathbf{z} \mid \mathbf{f}_{\ell-1}^\mathbf{z}) \sim \mathcal{N}(\mu_\ell(\mathbf{f}_{\ell-1}^\mathbf{z}), k_\ell(\mathbf{f}_{\ell-1}^\mathbf{z}, \mathbf{f}_{\ell-1}^\mathbf{z}))$ is an ℓ-th layer GP prior, and $p(\mathbf{f}_\ell \mid \mathbf{f}_\ell^\mathbf{z}, \mathbf{f}_{\ell-1}, \mathbf{f}_{\ell-1}^\mathbf{z})$ is an ℓ-th layer GP posterior at inputs $\mathbf{f}_{\ell-1}$ given $\mathbf{f}_\ell^\mathbf{z}$ and $\mathbf{f}_{\ell-1}^\mathbf{z}$ in ℓ-th and $(\ell-1)$-th layers respectively.

Variational lower bound We introduce the following variational distribution

$$q(\{\mathbf{f}_\ell\}, \{\mathbf{f}_\ell^\mathbf{z}\}) = \prod_{\ell=1}^{L} p(\mathbf{f}_\ell \mid \mathbf{f}_\ell^\mathbf{z}, \mathbf{f}_{\ell-1}, \mathbf{f}_{\ell-1}^\mathbf{z}) q(\mathbf{f}_\ell^\mathbf{z}), \tag{19}$$

where $q(\mathbf{f}_\ell^\mathbf{z}) \sim \mathcal{N}(\mathbf{m}_\ell, \mathbf{S}_\ell)$. The corresponding likelihood lower bound is as follows

$$\mathcal{L}(\mathbf{y}) \geq \mathbb{E}_q \left[\log \frac{p(\mathbf{y}, \{\mathbf{f}_\ell\}, \{\mathbf{f}_\ell^\mathbf{z}\})}{q(\{\mathbf{f}_\ell\}, \{\mathbf{f}_\ell^\mathbf{z}\})} \right] =$$
$$= \mathbb{E}_{q(\mathbf{f}_L)}[\log p(\mathbf{y} \mid \mathbf{f}_L)] - \tag{20}$$
$$- \sum_{\ell=1}^{L} \mathbb{E}_{q(\mathbf{f}_\ell^\mathbf{z}) q(\mathbf{f}_{\ell-1}^\mathbf{z})} \left[\log \frac{q(\mathbf{f}_\ell^\mathbf{z})}{p(\mathbf{f}_\ell^\mathbf{z} \mid \mathbf{f}_{\ell-1}^\mathbf{z})} \right]. \tag{21}$$

Estimating (20) We are estimating an expectation over the marginal $q(\mathbf{f}_L) \sim (f_L \circ \ldots \circ f_1)(\mathbf{x})$, which can be computed by marginalising the intermediate layers in the joint variational posterior (19):

$$q(\mathbf{f}_L) = \int q(\{\mathbf{f}_\ell\}, \{\mathbf{f}_\ell^\mathbf{z}\}) \, \mathrm{d}\{\mathbf{f}_\ell\}_{\ell=1}^{L-1} \, \mathrm{d}\{\mathbf{f}_i^\mathbf{z}\}_{\ell=1}^{L}$$
$$= \int p(\mathbf{f}_L \mid \mathbf{f}_L^\mathbf{z}, \mathbf{f}_{L-1}, \mathbf{f}_{L-1}^\mathbf{z}) q(\mathbf{f}_L^\mathbf{z}) \, \mathrm{d}\mathbf{f}_L^\mathbf{z} \times \tag{22}$$
$$\times \prod_{\ell=1}^{L-1} p(\mathbf{f}_\ell \mid \mathbf{f}_\ell^\mathbf{z}, \mathbf{f}_{\ell-1}, \mathbf{f}_{\ell-1}^\mathbf{z}) q(\mathbf{f}_\ell^\mathbf{z}) \, \mathrm{d}\mathbf{f}_\ell \, \mathrm{d}\mathbf{f}_\ell^\mathbf{z}.$$

The integrals in (22) are generally intractable since they require integrating the kernel matrices, thus we estimate them by sampling. Overall, the procedure is as follows:

1. Draw S samples $\{(\mathbf{f}_1^{\mathbf{z},s}, \ldots, \mathbf{f}_L^{\mathbf{z},s})\}_{s=1}^S \overset{iid}{\sim} q(\mathbf{f}_1^\mathbf{z}) \cdot \ldots \cdot q(\mathbf{f}_L^\mathbf{z})$,
2. Use the samples of $\{\mathbf{f}_\ell^\mathbf{z}\}$ to sequentially draw samples of intermediate layers $\mathbf{f}_\ell^s \sim p(\mathbf{f}_\ell \mid \mathbf{f}_\ell^{\mathbf{z},s}, \mathbf{f}_{\ell-1}^s, \mathbf{f}_{\ell-1}^{\mathbf{z},s})$ from a GP posterior given $\mathbf{f}_\ell^{\mathbf{z},s}$ and $\mathbf{f}_{\ell-1}^{\mathbf{z},s}$,
3. Use $\{\mathbf{f}_L^s\}_{s=1}^s$, the samples from $q(\mathbf{f}_L)$, to estimate the expectation in (20): $\mathbb{E}_{q(\mathbf{f}_L)}[\log p(\mathbf{y} \mid \mathbf{f}_L)] \approx \frac{1}{S} \sum_{s=1}^S \log p(\mathbf{y} \mid \mathbf{f}_L^s)$.

Estimating (21) We write the summands in (21) as

$$\mathbb{E}_{q(\mathbf{f}_\ell^\mathbf{z}) q(\mathbf{f}_{\ell-1}^\mathbf{z})} \left[\log \frac{q(\mathbf{f}_\ell^\mathbf{z})}{p(\mathbf{f}_\ell^\mathbf{z} \mid \mathbf{f}_{\ell-1}^\mathbf{z})} \right] = \tag{23}$$
$$= \mathbb{E}_{q(\mathbf{f}_{\ell-1}^\mathbf{z})} \mathrm{KL}[q(\mathbf{f}_\ell^\mathbf{z}) \,||\, p(\mathbf{f}_\ell^\mathbf{z} \mid \mathbf{f}_{\ell-1}^\mathbf{z})].$$

KL divergence between the two Gaussians $q(\mathbf{f}_\ell^\mathbf{z})$ and $p(\mathbf{f}_\ell^\mathbf{z} \mid \mathbf{f}_{\ell-1}^\mathbf{z})$ is a function of $\mathbf{f}_{\ell-1}^\mathbf{z}$ and can be computed analytically for a given value of $\mathbf{f}_{\ell-1}^\mathbf{z}$. Therefore, to estimate it, we use the draws from $\mathbf{f}_{\ell-1}^\mathbf{z}$ (which are computed for the estimate of (20) as well): for every such draw $\mathbf{f}_{\ell-1}^{\mathbf{z},s}$, we analytically compute the KL divergence $\mathrm{KL}[q(\mathbf{f}_\ell^\mathbf{z}) \,||\, p(\mathbf{f}_\ell^\mathbf{z} \,|\, \mathbf{f}_{\ell-1}^{\mathbf{z},s})]$, and then average these values to obtain a Monte-Carlo estimate of the expectation in (23).

Learning and predictions We maximise the likelihood lower bound (20-21) w.r.t. the variational parameters $\{\mathbf{m}_\ell\}$ and $\{\mathbf{S}_\ell\}$. The gradients can be obtained using a reparametrisation trick (Kingma and Welling, 2014). Given a test input $\mathbf{x}^*$, we can draw the DGP outputs $\mathbf{f}_L^* \sim (f_L \circ \ldots \circ f_1)(\mathbf{x}^*)$ by drawing from $q(\mathbf{f}_l^*)$ using the procedure for estimating (22) described above. We substitute $\mathbf{x}^*$ instead of $\mathbf{x}$ replacing $\mathbf{f}_\ell$ with $\mathbf{f}_\ell^*$ in (22), while the rest of the procedure remains the same.

Time complexity The time complexity of estimating (20) is $O(LNM^3)$. Sampling from $q(\mathbf{f}_i^\mathbf{z})$ is $O(M^3)$, while, as discussed in (Salimbeni and Deisenroth, 2017), sampling from $p(\mathbf{f}_i \,|\, \mathbf{f}_i^\mathbf{z}, \mathbf{f}_{i-1}, \mathbf{f}_{i-1}^\mathbf{z})$ can be performed separately for each element of $\mathbf{f}_i$ only requiring drawing from univariate Gaussians, which scales linearly with the

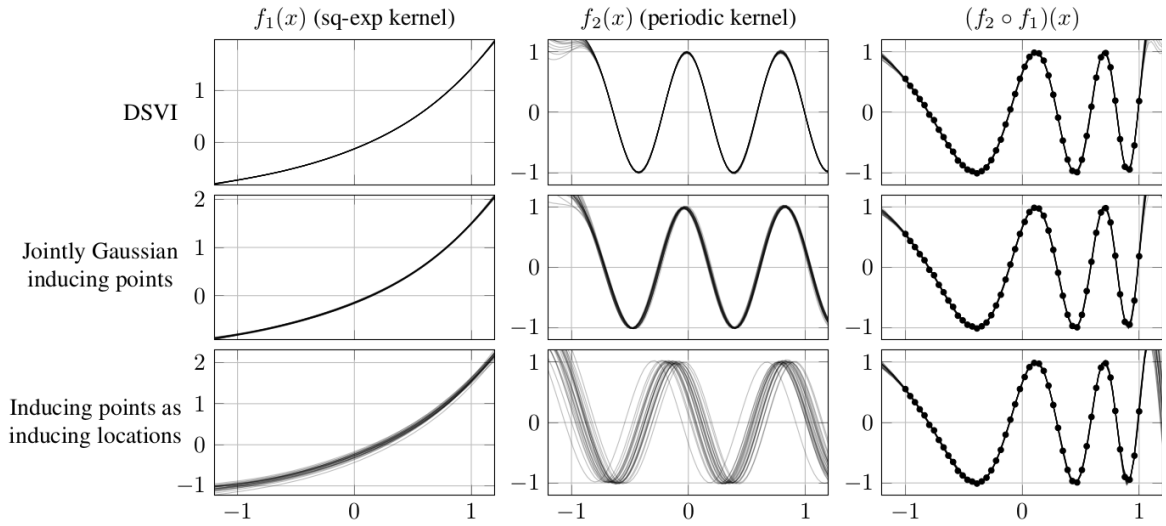


Figure 5: 25 random samples from 2-layer DGPs with squared-exponential and periodic kernels fitted to the observations in the third column (black dots) using DSVI as well as variational distributions discussed in Sec. 4. The first and second columns show samples from each of the two layers, while the third one shows samples from the entire composition (all such samples fit the data despite the variance in f_1 and f_2 because the two layers are dependent).

	ELBO	$\mathbb{V}\text{ar}\left[f_1(0)\right]$	$\mathbb{V}\text{ar}\left[f_2(0)\right]$
DSVI	13.43 ± 8.03	$1.99 \cdot 10^{-6} \pm 1.76 \cdot 10^{-7}$	$1.11 \cdot 10^{-4} \pm 1.35 \cdot 10^{-5}$
Jointly Gaussian	23.15 ± 6.80	$4.23 \cdot 10^{-5} \pm 3.17 \cdot 10^{-6}$	$3.33 \cdot 10^{-4} \pm 2.12 \cdot 10^{-5}$
Inducing points as inducing inputs	36.31 ± 3.55	$2.22 \cdot 10^{-3} \pm 2.73 \cdot 10^{-4}$	$4.98 \cdot 10^{-2} \pm 7.78 \cdot 10^{-3}$

Table 1: Evaluations of the DGPs fitted on a dataset in Fig. 5. First column shows lower bounds on marginal likelihood $p(\mathbf{y})$; the second and third ones show marginal variances of both layers at $x = 0$. The numbers are the means as well as standard deviations across 10 trials.

number of layers and training inputs. The estimate of (21) does not add additional complexity since we use the samples from $q(\mathbf{f}_i^{\mathbf{z}})$ drawn for estimating (20), while analytic computation of the KL divergence between $q(\mathbf{f}_\ell^{\mathbf{z}})$ and $p(\mathbf{f}_\ell^{\mathbf{z}} \mid \mathbf{f}_{\ell-1}^{\mathbf{z}})$ is $O(M^3)$ since it requires inversions of covariance matrices. Therefore, the overall complexity of estimating the lower bound is $O(LNM^3)$.

5 NUMERICAL SIMULATIONS

Compositional uncertainty As illustrated in Fig. 5 (first row) as well as in Table 1, the intermediate layers in a DGP with a factorised variational distribution over the inducing points collapse to nearly deterministic transformations in the range of the observed data ($[-1, 1]$). Meanwhile, the models with correlated inducing points (second and third rows) capture more uncertainty, with the approach proposed in Sec. 4.2 allowing us to capture more uncertainty than jointly Gaussian inducing points. Additional examples are provided in the Supplement.

Likelihood lower bounds In Table 1 we provide the variational lower bounds of the marginal likelihood[5], $p(\mathbf{y})$. We see that including the dependencies between the layers to the variational distribution leads to higher likelihood bounds, suggesting that factorised variational distributions are suboptimal for DGP inference.

6 APPLICATIONS

As compositions of functions, DGPs provide a natural way to represent data that is known to have a compositional structure and thus they may be used in applications to learn a more informative representation of the data.

Non-stationary time series Consider a sequence $\mathbf{y} \in \mathbb{R}^N$ that is observed at fixed time inputs $\mathbf{x} \in \mathbb{R}^N$. The

[5]The baseline estimate of the true marginal likelihood could be obtained by fitting the DGP using HMC (Havasi et al., 2018), however, we found the existing implementation of this scheme to be very unstable (as also noted by the authors) and the estimation of marginal likelihood from posterior samples to have high variance, hence we do not report such values.

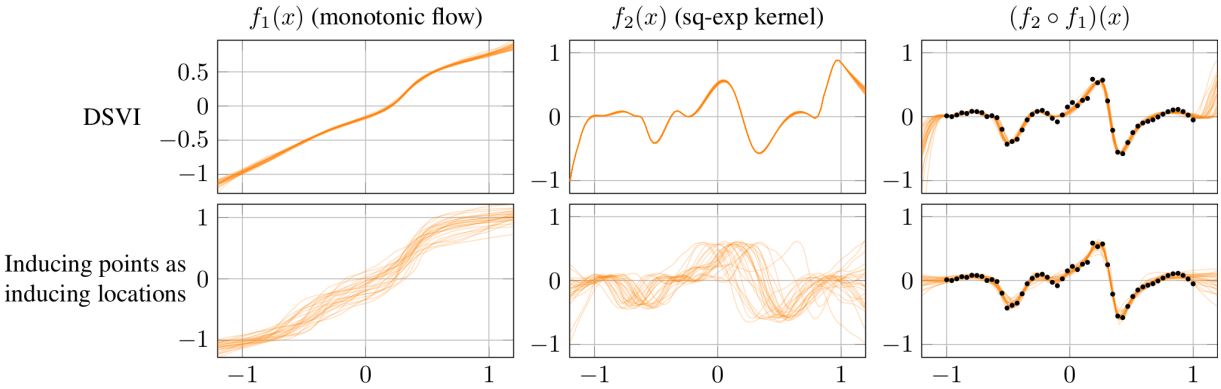


Figure 6: Compositional model of heartbeats data, comparing results without (top) and with correlations across layers.

observed sequence is assumed to be generated by temporally warping the inputs $\mathbf{x}$ as follows:

$$\mathbf{y} = f(g(\mathbf{x})) + \epsilon, \qquad \epsilon \sim \mathcal{N}(0, \sigma^2) \tag{24}$$

where $g(\cdot)$ is the temporal warping, $f(\cdot)$ is the latent function that encodes the structure of the observed sequence. The model in (24) generates non-stationary sequences, which are convenient to model with a composition of a monotonic transformation of the inputs $\mathbf{x}$ and a GP with a stationary kernel. The previous work on such models treats the temporal warping $g(\cdot)$ as a deterministic transformation (Snoek et al., 2014; Kazlauskaite et al., 2019), disregarding the fact that many different compositions may explain the observed data equally well.

To illustrate this, we consider a recording of a heartbeat (Bentley et al., 2011), and fit a two layer DGP with monotonic flow (Ustyuzhaninov et al., 2020) in the first layer. Here the prior on the warping functions $g(\cdot)$ dictates that while an identity warp is preferred, other smooth warps are plausible. The latent functions $f(\cdot)$ are modelled using a GP with a stationary squared exponential kernel. Fig. 6 shows how introducing correlations between the layers allows us to uncover a wide range of possible solutions that follow the above-defined priors and are consistent with the data. Meanwhile, the model with the same prior assumptions that uses a mean-field approximation collapses to a near-deterministic transformation, concentrating the probability mass in both layers on one of the many possible solutions. An application to sequence alignment is provided in the Supplement.

7 DISCUSSION

We have discussed the issue of compositional uncertainty in the context of DGPs. This is in contrast to much of the existing work on DGPs (as well as other Bayesian deep learning approaches (Gal, 2016)) that primarily focuses on predictive uncertainty. We argued that the uncertainty about the function implemented by each individual layer in the hierarchy provides a more informative model of the data. The inference in DGP models is typically performed using variational approximations that factorise across the layers of the hierarchy. While computationally convenient, such a factorisation implies that the distributions of the intermediate layers collapse to deterministic transformations. Such behaviour diminishes some of the other benefits offered by a compositional model of GPs, such as a systematic way to impose informative functional priors over each of the layers in the hierarchy and a way to uncover distributions over each layer.

To gain further insight into the issue of compositional uncertainty, we proposed two alternatives to the factorised variational distributions of inducing points that include some correlations between the layers. Contrary to the factorised distributions in DSVI, the proposed variational distributions uncover a range of possible solutions, reinforcing the argument that mean-field approximations are prohibitive when it comes to capturing compositional uncertainty. These consideration pose many open questions, ranging from technical considerations of more efficient ways to introduce correlations across layers and ways to represent variational distributions that are multimodal (Lawrence, 2000), to broader questions about the structures captured by each layer of the hierarchy, and the applications that may benefit from the more accurate estimates of compositional uncertainty.

Acknowledgments

This work has been supported by EPSRC CDE (EP/L016540/1), CAMERA (EP/M023281/1), EPSRC DTP, Hans Werthén Fund at The Royal Swedish Academy of Engineering Sciences, German Federal Ministry of Education and Research (project 01 IS 18049 A) and the Royal Society.

References

Bentley, P., Nordehn, G., Coimbra, M., and Mannor, S. (2011). Pascal Classifying Heart Sounds Challenge.

Bijl, H. (2018). LQG and Gaussian process techniques: For fixed-structure wind turbine control. *PhD thesis, Delft University of Technology*.

Bui, T., Hernandez-Lobato, D., Hernandez-Lobato, J., Li, Y., and Turner, R. (2016). Deep Gaussian processes for regression using approximate expectation propagation. In *International Conference on Machine Learning*.

Cutajar, K. (2019). *Broadening the scope of Gaussian processes for large-scale learning*. PhD thesis, Thesis.

Dai, Z., Damianou, A., González, J., and Lawrence, N. D. (2016). Variational auto-encoded deep Gaussian processes. In *International Conference on Learning Representations*.

Damianou, A. (2015). Deep Gaussian processes and variational propagation of uncertainty. *PhD Thesis, University of Sheffield*.

Damianou, A. and Lawrence, N. (2013). Deep Gaussian processes. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Dunlop, M. M., Girolami, M. A., Stuart, A. M., and Teckentrup, A. L. (2018). How deep are deep Gaussian processes? *Journal of Machine Learning Research*.

Duvenaud, D., Rippel, O., Adams, R. P., and Ghahramani, Z. (2014). Avoiding pathologies in very deep networks. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Gal, Y. (2016). *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge.

Gal, Y. and Ghahramani, Z. (2016). Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*.

Girard, A., Rasmussen, C. E., Candela, J. Q., and Murray-Smith, R. (2003). Gaussian process priors with uncertain inputs application to multiple-step ahead time series forecasting. In *Neural Information Processing Systems*.

Havasi, M., Hernández-Lobato, J. M., and Murillo-Fuentes, J. J. (2018). Inference in deep Gaussian processes using stochastic gradient Hamiltonian Monte Carlo. In *Neural Information Processing Systems*.

Hensman, J., Durrande, N., and Solin, A. (2017). Variational fourier features for Gaussian processes. *Journal of Machine Learning Research (JMLR)*, 18(1).

Hensman, J., Fusi, N., and Lawrence, N. D. (2013). Gaussian processes for big data. In *Conference on Uncertainty in Artificial Intelligence (UAI)*.

Hensman, J. and Lawrence, N. D. (2014). Nested variational compression in deep Gaussian processes. *arXiv preprint arXiv:1412.1370*.

Jin, M., Damianou, A., Abbeel, P., and Spanos, C. (2017). Inverse reinforcement learning via deep Gaussian process. *Conference on Uncertainty in Artificial Intelligence (UAI)*.

Kaiser, M., Otte, C., Runkler, T., and Ek, C. H. (2018). Bayesian alignments of warped multi-output Gaussian processes. In *Neural Information Processing Systems*.

Kazlauskaite, I., Ek, C. H., and Campbell, N. (2019). Gaussian process latent variable alignment learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Kingma, D. P. and Welling, M. (2014). Auto-encoding variational Bayes. In *International Conference on Learning Representations*.

Lawrence, N. D. (2000). *Variational Inference in Probabilistic Models*. PhD thesis, Cambridge University.

Lawrence, N. D. (2004). Gaussian process latent variable models for visualisation of high dimensional data. *Neural Information Processing Systems*.

Lawrence, N. D. and Moore, A. J. (2007). Hierarchical Gaussian process latent variable models. In *International Conference on Machine Learning*.

Lázaro-Gredilla, M. (2012). Bayesian warped Gaussian processes. In *Neural Information Processing Systems*.

Mchutchon, A. and Rasmussen, C. E. (2011). Gaussian process training with input noise. In *Neural Information Processing Systems*.

Rainforth, T., Cornish, R., Yang, H., Warrington, A., and Wood, F. (2019). On nesting Monte Carlo estimators. *Proceedings of Machine Learning Research*, 80.

Rasmussen, C. E. and Williams, C. K. I. (2005). *Gaussian Processes for Machine Learning*. MIT Press.

Rudner, T. G. J. and Sejdinovic, D. (2017). Inter-domain deep Gaussian processes.

Salimbeni, H. and Deisenroth, M. (2017). Doubly stochastic variational inference for deep Gaussian processes. In *Neural Information Processing Systems*.

Snoek, J., Swersky, K., Zemel, R., and Adams, R. (2014). Input warping for Bayesian optimization of non-stationary functions. In *International Conference on Machine Learning*.

Sun, S., Zhang, G., Shi, J., and Grosse, R. (2019). Functional variational Bayesian neural networks. In *International Conference on Learning Representations*.

Titsias, M. and Lawrence, N. (2010). Bayesian Gaussian process latent variable model. *Journal of Machine Learning Research (JMLR)*, 9.

Ustyuzhaninov, I., Kazlauskaite, I., Ek, C. H., and Campbell, N. D. F. (2020). Monotonic Gaussian process flow. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

ROTATION-INVARIANT CLUSTERING OF NEURONAL RESPONSES IN PRIMARY VISUAL CORTEX

Ivan Ustyuzhaninov,[1-3] Santiago A. Cadena,[1-3] Emmanouil Froudarakis,[4,5] Paul G. Fahey,[4,5] Edgar Y. Walker,[4,5] Erick Cobos,[4,5] Jacob Reimer,[4,5] Fabian H. Sinz,[4,5] Andreas S. Tolias,[1,4-6] Matthias Bethge,[1-3,5,†] Alexander S. Ecker[1-3,5,†,‡,*]

[1] Centre for Integrative Neuroscience, University of Tübingen, Germany
[2] Bernstein Center for Computational Neuroscience, University of Tübingen, Germany
[3] Institute for Theoretical Physics, University of Tübingen, Germany
[4] Department of Neuroscience, Baylor College of Medicine, Houston, TX, USA
[5] Center for Neuroscience and Artificial Intelligence, BCM, Houston, TX, USA
[6] Department of Electrical and Computer Engineering, Rice University, Houston, TX, USA

† *Authors contributed equally*
‡ *Present address: Department of Computer Science, University of Göttingen, Germany*

* ecker@cs.uni-goettingen.de

ABSTRACT

Similar to a convolutional neural network (CNN), the mammalian retina encodes visual information into several dozen nonlinear feature maps, each formed by one ganglion cell type that tiles the visual space in an approximately shift-equivariant manner. Whether such organization into distinct cell types is maintained at the level of cortical image processing is an open question. Predictive models building upon convolutional features have been shown to provide state-of-the-art performance, and have recently been extended to include rotation equivariance in order to account for the orientation selectivity of V1 neurons. However, generally no direct correspondence between CNN feature maps and groups of individual neurons emerges in these models, thus rendering it an open question whether V1 neurons form distinct functional clusters. Here we build upon the rotation-equivariant representation of a CNN-based V1 model and propose a methodology for clustering the representations of neurons in this model to find functional cell types independent of preferred orientations of the neurons. We apply this method to a dataset of 6000 neurons and visualize the preferred stimuli of the resulting clusters. Our results highlight the range of non-linear computations in mouse V1.

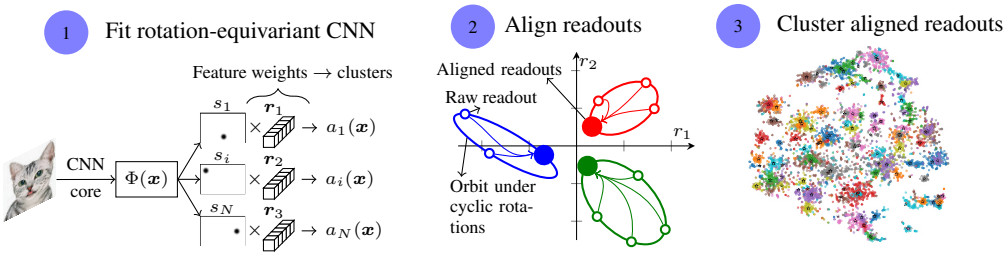


Figure 1: An overview of our approach. (1) Fit rotation-equivariant CNN to predict neural responses and use readout vectors $\boldsymbol{r}_i$ as proxies for neural computations. (2) Align readouts to account for different preferred orientations. (3) Cluster the aligned readouts.

1 INTRODUCTION

A compact description of the nonlinear computations in primary visual cortex (V1) is still elusive. Like in the retina (Baden et al., 2016; Sanes & Masland, 2015), such understanding could come from a functional classification of neurons. However, it is currently unknown if excitatory neurons in V1 are organized into functionally distinct cell types.

It has recently been proposed that predictive models of neural responses based on convolutional neural networks could help answer this question (Klindt et al., 2017; Ecker et al., 2019). These models are based on a simple principle (Fig. 1-①): learn a *core* (e.g. a convolutional network) that is shared among all neurons and provides nonlinear features $\Phi(x)$, which are turned into predictions of neural responses by a linear *readout* for each neuron (Antolík et al., 2016). Models based on this basic architecture exploit aspects of our current understanding of V1 processing. First, convolutional weight sharing allows us to characterize neurons performing the same computation but with differently located receptive fields by the same feature map (Klindt et al., 2017; McIntosh et al., 2016; Kindel et al., 2019; Cadena et al., 2019). Second, V1 neurons can extract local oriented features such as edges at different orientations, and most low-level image features can appear at arbitrary orientations. Therefore, Ecker et al. (2019) proposed a rotation-equivariant convolutional neural network model of V1 that extends the convolutional weight sharing to the orientations domain.

The basic idea of previous work (Klindt et al., 2017; Ecker et al., 2019) is that each convolutional feature map could correspond to one cell type. While this idea is conceptually appealing, it hinges on the assumption that V1 neurons are described well by individual units in the shared feature space. However, existing models do not tend to converge to such solutions. Instead, V1 neurons are better described by linearly combining units from the same spatial location in multiple different feature maps (Ecker et al., 2019). Whether or not there are distinct functional cell types in V1 is therefore still an open question.

Here, we address this question by introducing a clustering method on rotation-equivariant spaces. We treat the feature weights (Fig. 1-①) that map convolutional features to neural responses as an approximate low-dimensional vector representation of this neuron's input-output function. We then split neurons into functional types using a two-stage procedure: first, because these feature weights have a rotation-equivariant structure, we find an alignment that rotates them into a canonical orientation (Fig. 1-②); in a second step, we cluster them using standard approaches such as k-means or Gaussian mixture models (Fig. 1-③). We apply our method to the published model and data of Ecker et al. (2019) that contains recordings of around 6000 neurons in mouse V1 under stimulation with natural images. Our results suggest that V1 neurons might indeed be organized into functional clusters. The dataset is best described by a GMM with around 100 clusters, which are to some extent redundant but can be grouped into a smaller number of 10–20 groups. We analyse the resulting clusters via their maximally exciting inputs (MEIs) (Walker et al., 2018) to show that many of these functional clusters do indeed correspond to distinct computations.

2 RELATED WORK

Unsupervised functional clustering via system identification As outlined in the introduction, our work builds directly upon the methods developed by Klindt et al. (2017) and Ecker et al. (2019). While these works view the feature weights as indicators assigning each neuron to its 'cell type' (feature map), we here take a different view on the same model: rather than focusing on the convolutional features and viewing them as cell types, we treat the feature weights as a low-dimensional representation of the input-output function of each neuron and perform clustering in this space. This view on the problem has the advantage that there is no one-to-one correspondence between the number of feature maps and the number of cell types and we disentangle model fitting from its interpretation. On the other hand, our approach comes with an addition complexity: because the feature weights obey rotational equivariance and we would like our clustering to be invariant to rotations, we require a clustering algorithm that is invariant with respect to a class of (linear) transformations.

Invariant clustering A number of authors have developed clustering methods that are invariant to linear (Tarpey, 2007), affine (Brubaker & Vempala, 2008) or image transformations by rotations,

scalings and translations (Frey & Jojic, 2002). Ju et al. (2019) cluster natural images by using a CNN to represent the space of invariant transformations rather than specifying it explicitly.

Alignments Instead of using custom clustering algorithms that are invariant under certain transformations, we take a simpler approach: we first transform our features such that they are maximally aligned using the class of transformations the clustering should be invariant to. This approach has been used in other contexts before, usually by minimizing the distances between the transformed observations. Examples include alignment of shapes in $\mathbb{R}^d$ using rigid motions (Gower, 1975; Dryden & Mardia, 1998), alignment of temporal signals by finding monotonic input warps (Zhou & De la Torre, 2012), or alignment of manifolds with the distance between the observations being defined according to the metric on the manifold (Wang & Mahadevan, 2008; Cui et al., 2014). There is also work on alignment objectives beyond minimizing distances between transformed observations, examples of which include objectives based on generative models of observations (Kurtek et al., 2011; Duncker & Sahani, 2018) or probabilistic ones which are particularly suited for alignment with multiple groups of underlying observations (Kazlauskaite et al., 2019).

3 ROTATION-EQUIVARIANT CLUSTERING

Our goal is to cluster neurons in the dataset into groups performing similar computations. To do so, we use their low-dimensional representations obtained from the published rotation-equivariant CNN of Ecker et al. (2019), which predicts neural activity as a function of an external image stimulus. We briefly review this model before describing our approach to rotation-invariant clustering.

CNN model architecture The model consists of two parts (Fig. 1-①):

1. A convolutional *core* that is shared by all neurons and computes feature representations $\Phi(\boldsymbol{x}) \in \mathbb{R}^{W \times H \times K}$, where $\boldsymbol{x}$ is the input image, $W \times H$ is the spatial dimensionality and K is the number of feature maps.
2. A separate linear *readout* $\boldsymbol{w}_n = \boldsymbol{s}_n \otimes \boldsymbol{r}_n \in \mathbb{R}^{W \times H \times K}$ for each neuron $n = 1, \ldots, N$, factorized into a spatial mask $\boldsymbol{s}_n$ and a vector of feature weights $\boldsymbol{r}_n$.

The predicted activity of a neuron n for image $\boldsymbol{x}$ is

$$a_n(\boldsymbol{x}) = f(\boldsymbol{w}_n \cdot \Phi(\boldsymbol{x})) = f(\boldsymbol{r}_n \cdot \boldsymbol{s}_n \cdot \Phi(\boldsymbol{x})) \in \mathbb{R} \tag{1}$$

where $f(\cdot)$ is a non-linear activation function. Such a CNN therefore provides K-dimensional feature weights $\boldsymbol{r}_n$ characterising linear combinations of spatially weighted image features $\boldsymbol{s}_n \cdot \Phi(\boldsymbol{x})$ that are predictive of neural activity. We treat these feature weights as finite-dimensional proxies of actual computations implemented by the neurons. Because masks $\boldsymbol{s}_n$ (another component of readouts $\boldsymbol{w}_n$ defined above) are irrelevant for our analysis, we will often refer to $\boldsymbol{r}_n$ simply as readout vectors. We will be referring to the matrix having $\boldsymbol{r}_i$ as its rows as the readout matrix $\boldsymbol{R} \in \mathbb{R}^{N \times K}$.

Rotation-equivariant core Feature representations $\Phi(\boldsymbol{x})$ are rotation-equivariant, meaning that weight sharing is not only applied across space but also across rotations: for each convolutional filter there exist O rotated copies, each rotated by $2\pi/O$. Feature vectors $\phi_{ij}(\boldsymbol{x})$ at position (i, j) therefore consist of F different features, each computed in O linearly-spaced orientations (such that $F \times O = K$). We can think of $\phi_{ij}(\boldsymbol{x})$ as being reshaped into an array of size $F \times O$. Having computed $\phi_{ij}(\boldsymbol{x})$, we can compute $\phi_{ij}(\boldsymbol{x}')$ with $\boldsymbol{x}'$ being a stimulus $\boldsymbol{x}$ rotated around (i, j) by $2\pi/O$ by cyclically shifting the last axis of $\phi_{ij}(\boldsymbol{x})$ by one step (this mechanism is illustrated in Fig. 2).

Rotation-equivalent computations The linear readout adheres to the same rotation-equivariant structure as the core. As our goal is to cluster the neurons by the patterns of features they pool while being invariant to orientation, we need to account for the set of weight transformations that correspond to a rotation of the stimulus when clustering neurons. We illustrate this issue with a small toy example consisting of six neurons that fall into two cell types (Fig. 2B). Within each cell type (columns in Fig. 2B), the individual neurons differ only in their orientation. More formally, we define the computations performed by two neurons n_i and n_j to be *rotation-equivalent* if there exists a rotation ψ_{ij} such that for any input stimulus $\boldsymbol{x}$ we have $a_{n_i}(\boldsymbol{x}) = a_{n_j}(\psi_{ij}(\boldsymbol{x}))$. We will refer to such neurons as rotation-equivalent as well.

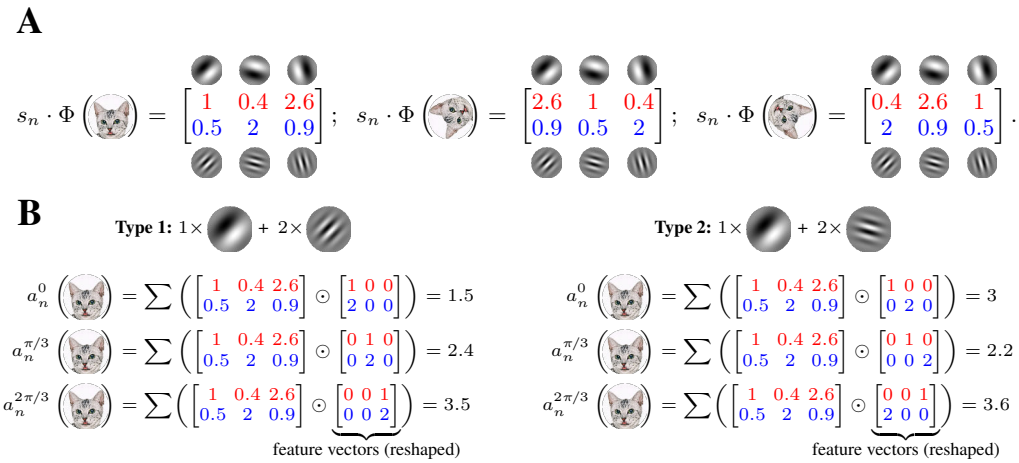


Figure 2: Toy example illustrating the computations in a rotation-equivariant CNN with two features (red and blue; cartoon feature representations are shown on top of corresponding values of $\Phi(\boldsymbol{x})$) in three orientations (0, $\pi/3$, $2\pi/3$). **A:** Output of the rotation-equivariant CNN for an input image rotated by $\pi/3$ (base orientation) can be computed by a cyclic shift. **B:** Example of two distinct types of neurons (columns) in three orientations (rows). Computations for both types consist of linear combinations of the two features computed by the CNN with the same weights, but in different relative orientations. Readouts of neurons of the same type in different orientations are cyclic shifts of each other, since they produce the same outputs on correspondingly rotated inputs.

Readouts of rotation-equivalent neurons Directly clustering the readout matrix $\boldsymbol{R}$ does not respect the rotation equivalence, because readout vectors of neurons implementing rotation-equivalent computations are not identical (Fig. 2). To address that, we first modify $\boldsymbol{R}$ to obtain a matrix $\tilde{\boldsymbol{R}}$ with the rows corresponding to rotation-equivalent neurons aligned to a canonical form, and then cluster $\tilde{\boldsymbol{R}}$ to obtain functional cell types. Rotating an input $\boldsymbol{x}$ by a multiple of $2\pi/O$ corresponds to cyclically shifting $\Phi(\boldsymbol{x})$, so the readout vectors of two rotation-equivalent neurons whose orientation difference ψ_{ij} is a multiple of $2\pi/O$ are also cyclic shifts of each other (Fig. 2). For arbitrary rotations that are not necessarily a multiple of $2\pi/O$, we assume the readout $\boldsymbol{r}_{n_j}$ of neuron n_j to be a linear interpolation of cyclic shifts of $\boldsymbol{r}_{n_i}$ corresponding to the two nearest rotations which are multiples of $2\pi/O$. Formally, we define a *cyclic rotation matrix by an angle* $\alpha \in [0, 2\pi)$ as follows (matrix has shape $O \times O$; column indices are shown above the matrix):

$$S_\alpha = \begin{array}{c} \begin{matrix} 1 & & i & i+1 & i+2 & i+3 & & O \end{matrix} \\ \begin{pmatrix} 0 & \dots & 0 & 1-\gamma & \gamma & 0 & \dots & 0 \\ 0 & \dots & 0 & 0 & 1-\gamma & \gamma & \dots & 0 \\ & \vdots & & & & & \vdots & \\ 0 & \dots & 1-\gamma & \gamma & 0 & 0 & \dots & 0 \end{pmatrix} \end{array}, \qquad \begin{aligned} i &= \left\lfloor \frac{\alpha O}{2\pi} \right\rfloor \mod O, \\ \gamma &= \frac{\alpha O}{2\pi} - i. \end{aligned} \tag{2}$$

Given a readout vector $\boldsymbol{r}_n \in \mathbb{R}^K$, we can think of it as a matrix $\boldsymbol{r}_n \in \mathbb{R}^{O \times F}$ with columns corresponding to readout coefficients for different orientations of a single feature. The *cyclic rotation of a readout* $\boldsymbol{r}_n$ *by an angle* $\alpha \in [0, 2\pi)$ can be expressed as a matrix multiplication $\boldsymbol{r}_n(\alpha) = S_\alpha \boldsymbol{r}_n$. Note, by writing $\boldsymbol{r}_n(\alpha)$ as a function of α we refer to cyclically rotated (transformed) readouts, while $\boldsymbol{r}_n$ are the fixed ones coming from a pre-trained CNN and $\boldsymbol{r}_n(0) = \boldsymbol{r}_n$.

For two rotation-equivalent neurons n_i and n_j, the readout vector $\boldsymbol{r}_{n_j}$ can be computed as a cyclic rotation of $\boldsymbol{r}_{n_i}$ by α, which is the rotation angle of ψ_{ij}. If α is a multiple of $2\pi/O$; otherwise it is only an approximation which becomes increasingly accurate as O increases.

Aligning the readouts Assuming V1 neurons form discrete functional cell types, all neurons in the dataset (and hence the readouts characterising them) can be partitioned into non-overlapping classes w.r.t. the rotation equivalence relation we introduced above. Choosing one representative of each class and replacing the rows of $\boldsymbol{R}$ with their class representatives, we can obtain $\tilde{\boldsymbol{R}}$ from $\boldsymbol{R}$. Next, we discuss an algorithm for finding such representatives of each class.

We claim that by minimising the sum of pairwise distances between the cyclically rotated readouts

$$\{\alpha_i^*\} = \arg\min_{\{\alpha_i\}} \sum_{i=1}^{N} \sum_{j=i+1}^{N} ||\boldsymbol{r}_i(\alpha_i) - \boldsymbol{r}_j(\alpha_j)||, \tag{3}$$

we can transform each readout into a representative of a corresponding equivalence class (same for all readouts of a class), i.e. $\boldsymbol{r}_i(\alpha_i^*) = \boldsymbol{r}_j(\alpha_j^*)$ if neurons i and j are rotation-equivalent. This is indeed the case because the readouts of the same equivalence class lie on the orbit obtained by cyclically rotating any representative of that class. Such angles $\{\alpha_i^*\}$ that neurons of the same class end up on the same point on the orbit (i.e. aligned to the same class representative) clearly minimise Eq. (3), and since different orbits do not intersect (they are different classes of equivalence), readouts of different equivalence classes cannot end up at the same point. Note that the resulting class representatives are not arbitrary, but those with the smallest sum of distances between each other in the Euclidean space. This mechanism is illustrated in Fig. 1-②.

Clustering aligned readouts Having obtained $\tilde{\boldsymbol{R}}$ with rows $\boldsymbol{r}_i(\alpha_i^*)$, we can cluster the rows of this matrix using any standard clustering method (e.g. K-Means, GMM, etc.) to obtain groups of neurons (cell types) performing similar computations independent of their preferred orientations.

Continuous relaxation of cyclic rotations The rotation-invariant clustering described above is based on solving the optimisation problem in Eq. (3). To do so, we would typically use a gradient-based optimisation, which is prone to local minima because of the way we define cyclic rotations in Eq. (2). According to that definition, a rotated readout is a linear combination of two nearest base rotations, or rather a linear combination of all such rotations with only two coefficients being non-zero. That means that gradients of all but two coefficients w.r.t. the angle α are zero, and the optimisation would converge to a local minimum corresponding to the best linear combination of the two base rotations initialised with non-zero coefficients (Fig. 3).

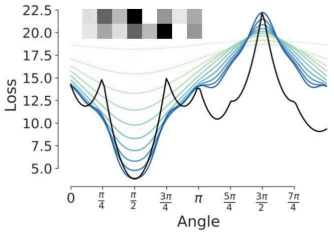


Figure 3: Distance between two vectors (top left corner) with first one fixed and second cyclically shifted by an angle on the x-axis. Continuous relaxation (shades of blue) of linearly interpolated (black) cyclic shifts smooths gradients and helps overcome local minima.

To address this issue, we propose to approximate Eq. (2), such that $\boldsymbol{r}_{n_i}(\alpha)$ is a linear combination of all base rotations with non-zero coefficients, with the coefficients for the two nearest base rotations being the largest. Specifically we compute the coefficients by sampling the von Mises density at fixed orientations to ensure cyclic boundary conditions and define $\tilde{\boldsymbol{r}}_{n_i}(\alpha)$, a continuous relaxation of $\boldsymbol{r}_{n_i}(\alpha)$, as $\tilde{\boldsymbol{r}}_{n_i}(\alpha) = \tilde{S}_\alpha \boldsymbol{r}_{n_i}$ where

$$\tilde{S}_\alpha = \begin{pmatrix} \gamma_1 & \gamma_2 & \cdots & \gamma_O \\ \gamma_O & \gamma_1 & \cdots & \gamma_{O-1} \\ \vdots & & & \vdots \\ \gamma_2 & \gamma_3 & \cdots & \gamma_1 \end{pmatrix} \quad \text{with} \quad \gamma_i = \frac{\exp(T\cos(\alpha - (i-1)\cdot 2\pi/O))}{\sum_{i=1}^{O} \exp(T\cos(\alpha - (i-1)\cdot 2\pi/O))}. \tag{4}$$

The parameter $T \geq 0$ controls the sparseness of coefficients $\{\gamma_i\}$. For small T, many of the coefficients are significantly greater than zero, allowing the optimiser to propagate the gradients and reduce the effect of initialisation. As T increases, $\tilde{\boldsymbol{r}}_{n_i}(\alpha)$ becomes more similar to $\boldsymbol{r}_{n_i}(\alpha)$, and the rotations by multiples of $2\pi/O$ are recovered. In the limit, $\tilde{\boldsymbol{r}}_{n_i}(2\pi k/O) \to \boldsymbol{r}_{n_i}(2\pi k/O)$ as $T \to \infty$ (Fig. 3). Instead of fixing T, we learn it by optimising the regularised alignment objective with additional reconstruction loss preventing trivial solutions (e.g. all coordinates of $\tilde{\boldsymbol{r}}_i(\alpha_i)$ being the same for $T = 0$):

$$\{\alpha_i^*\} = \arg\min_{\{\alpha_i\},T} \sum_{i=1}^{N} \sum_{j=i+1}^{N} ||\tilde{\boldsymbol{r}}_i(\alpha_i) - \tilde{\boldsymbol{r}}_j(\alpha_j)|| + \beta \sum_{i=1}^{N} ||\tilde{\boldsymbol{r}}_i(0) - \boldsymbol{r}_i||. \tag{5}$$

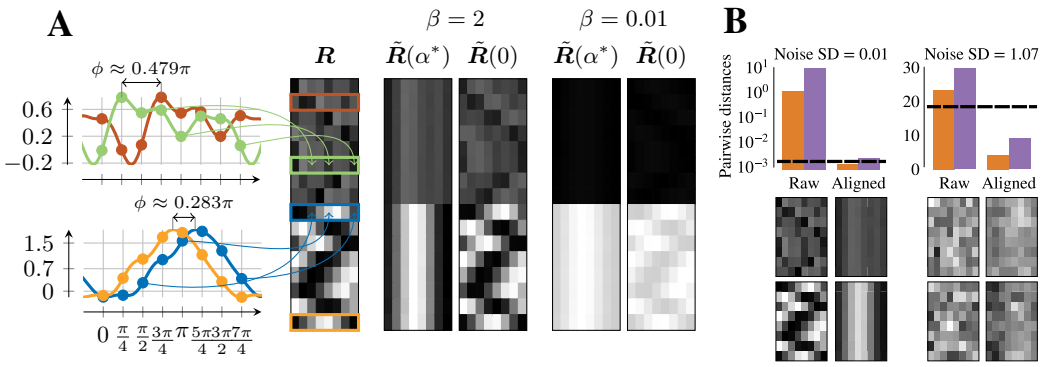


Figure 4: Synthetic data set: generation, alignment and dependence on noise. **A:** Panel $\boldsymbol{R}$ shows the unaligned synthetic data set as well as the corresponding shifted GP samples for each of the two groups of neurons (see details in the main text). Colored boxes in $\boldsymbol{R}$ correspond to the colors of corresponding GP samples. Panels $\tilde{\boldsymbol{R}}(\alpha^*)$ and $\tilde{\boldsymbol{R}}(0)$ show aligned readouts and readouts rotated by 0 using Eq. (4) respectively. $\tilde{\boldsymbol{R}}(0)$ should be similar to $\boldsymbol{R}$ for an adequate choice of β (and consequently optimised value of T). **B:** Effect of observation noise. Means of pairwise distances for each of the two groups shown in matrix $\boldsymbol{R}$ for two levels of Gaussian noise added to the dataset. Black dashed line: expected pairwise distance due to noise only (i.e. for perfectly aligned data with added noise). Raw and aligned matrices for each of the two groups are shown below the bar plots.

4 Experiments

Synthetic dataset We generate a small toy dataset consisting of 16 hypothetical neurons (readouts) of two cell types to illustrate the proposed alignment method. Each readout consists of just one feature in eight base orientations (linearly spaced between 0 and $7\pi/4$) and is generated by one of the two underlying types of readouts cyclically shifted by a random angle $\phi \in [0, 2\pi)$. To generate such a dataset , we draw two independent noiseless functions from a Gaussian process (GP) with a periodic kernel (with period 2π), then for each readout we randomly choose one of the two GP samples, shift it by an angle ϕ and evaluate the shifted function at the base orientations to obtain an 8-dimensional vector modelling the observed readout values. This process is illustrated in Fig. 4A.

Neural data We use the same dataset as in Ecker et al. (2019), consisting of simultaneous recordings of responses of 6005 excitatory neurons in mouse primary visual cortex (layers 2/3 and 4).

Model details We analyse a rotation-equivariant CNN consisting of a three-layer core with 16 features in 8 orientations in each layer (kernel sizes 13, 5, 5) and 128-dimensional readouts ($F = 16$, $O = 8$). We use the pre-trained model provided by Ecker et al. (2019). We align the readout matrix $\boldsymbol{R}$ by minimising Eq. (5) w.r.t. the rotation angles α_i and temperature T. We fit models for 20 log-spaced values of β in $[0.001, 10]$, and choose for analysis the one with the smallest alignment loss (Eq. (3)) among the models with optimised temperature $T > 5$. We use Adam (Kingma & Ba, 2015) with early stopping and initial learning rate of 0.01 decreased three times.

Clustering aligned readouts We use the Gaussian mixture model implemented in scikit-learn (Pedregosa et al., 2011) for clustering the aligned readouts $\tilde{\boldsymbol{R}}$. We use spherical covariances to reduce the number of optimised parameters. To obtain a quantitative estimate of the number of clusters in $\tilde{\boldsymbol{R}}$, we randomly split the dataset of 6005 neurons into training (4000 neurons) and test (2005 neurons) sets, fit GMMs with different numbers of clusters on the training set, and then evaluate the likelihood of the fitted model on the test set.

5 RESULTS

5.1 SYNTHETIC DATA SET ALIGNMENT

We start by demonstrating on a synthetic dataset (Sec. 4) that optimising Eq. (5) can successfully align the readouts (Fig. 4A), assuming β has been chosen appropriately. Note that readouts have been shifted by arbitrary angles (not multiples of $\pi/4$ as demonstrated for readouts in colored boxes in Fig. 4A), and they are aligned precisely via interpolation Eq. (4). We can also see the effect of the parameter β, controlling the relative weight of the reconstruction term (i.e. similarity of readouts rotated by 0 degrees to the observations). Small values of β incur a small cost for poor reconstructions resulting in small optimised values of T and over-smoothed aligned readouts.

We next ask whether the alignment procedure still works in the presence of observational noise (Fig. 4B). For small to moderate noise levels (Fig. 4B, left), alignment reduces the pairwise distances to the level expected from the observation noise (shown at the top), confirming the visual impression (shown at the bottom) that alignment works well. For high noise levels (Fig. 4B, right), alignment breaks down as expected, and we observe overfitting to the noise patterns (shown by the pairwise distances after alignment dropping below the level expected from observation noise alone).

5.2 MOUSE V1 DATASET

Clustering We evaluate the GMM used to cluster $\tilde{\boldsymbol{R}}$ for different numbers of clusters. The test likelihood starts to plateau at around 100 clusters (Fig. 5), so we use 100 clusters in the following.

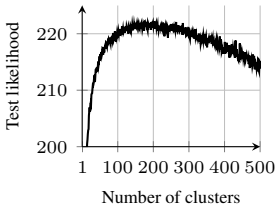


Figure 5: Test set likelihood of GMMs applied to $\tilde{\boldsymbol{R}}$ as a function of the number of clusters.

Visualization of clusters To visualize the clustering result, we compute a two-dimensional t-SNE embedding (van der Maaten & Hinton, 2008) of the matrix of aligned readouts $\tilde{\boldsymbol{R}}$, which is coloured according to the GMM clustering of $\tilde{\boldsymbol{R}}$ with 100 clusters (Fig. 6). Note that we use the embedding only for visualization, but cluster 128D aligned readouts in $\tilde{\boldsymbol{R}}$ directly. In addition to the embeddings, we also visualize the computations performed by some of the clusters by showing maximally exciting inputs (MEIs). We compute MEIs via activity maximisation (Erhan et al., 2009; Walker et al., 2018) and show the stimuli that maximally drive the 16 best-predicted neurons of each cluster. We observe that MEIs corresponding to neurons of the same cluster are generally consistent up to rotation and receptive field location, suggesting that the proposed clustering method captures the similarities in the neural computations while ignoring the nuisances as desired.

Network learned redundant features We noticed a number of clusters with similar MEIs (e.g. Block 9 and Block 13 in Fig. 6). There could be two reasons for this observation: (a) the neural computations corresponding to these clusters could be different in some other aspect, which we cannot tell by inspecting MEIs as they represent only the maximum of a complex function, or (b) the features learned by the CNN could be redundant, i.e. the hidden layers could learn to approximate the same function in multiple different ways. To answer this question, we compute a cluster confusion matrix (Fig. 7, left), which quantifies how similar the response predictions of different clusters are across images. The element (p, q) corresponds to the correlation coefficient between the predicted responses on the entire training set of hypothetical neurons with cluster means for clusters p and q used as readouts, accounting for potential differences in canonical orientation across clusters. By greedily re-arranging clusters in the matrix into blocks based on their correlations, we show that the 100 clusters in the model can be grouped into a much smaller number of functionally distinct clusters. Using a correlation threshold of 0.5 in this re-arrangement procedure, we obtain an example arrangement into 17 blocks (Fig. 7). Thus, the network has learned an internal representation that allows constructing very similar functions in multiple ways, suggesting that further pruning the learned network before clustering could lead to a more compact feature space for V1.

Finally, to quantify how consistent the resulting 17 groups of clusters are, we compute an MEI confusion matrix (Fig. 7, right panel). Its (i, j) element is the predicted activity of neuron j for the MEI of neuron i, after accounting for orientation and receptive field location (i.e. $a_j(\boldsymbol{y}_i)$, where

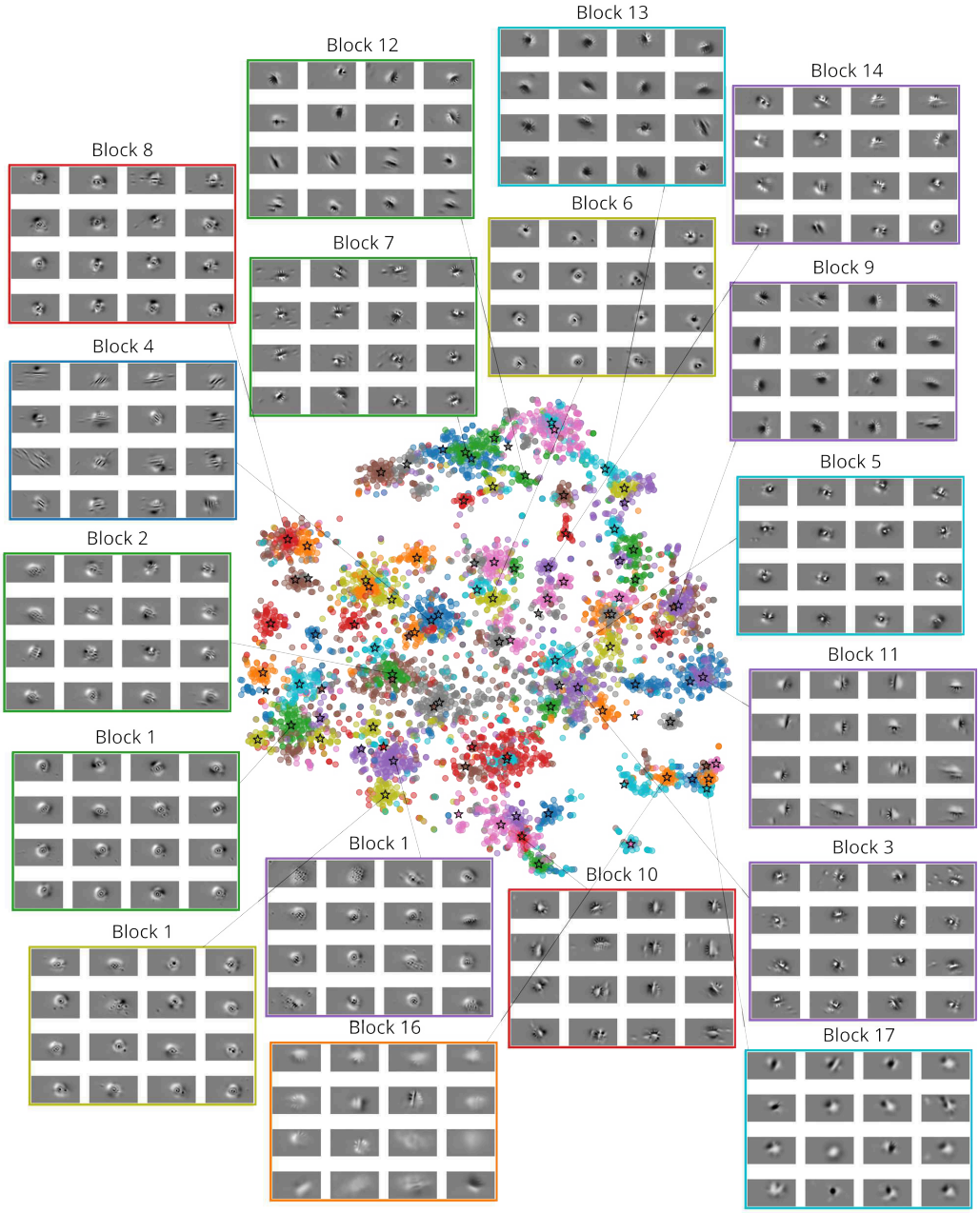


Figure 6: 2D t-SNE embedding of the aligned readouts $\tilde{\boldsymbol{R}}$, colored according to the GMM clustering with 100 components. Black stars show the locations of cluster centers. For some of the clusters, the MEIs of 16 best predicted neurons of that cluster are shown. The titles in the MEI subfigures show which matrix block in Fig. 7 (left) that cluster belongs to.

$\boldsymbol{y}_i$ is the MEI of neuron i moved and rotated such that it optimally drives neuron j). We show this matrix using the same grouping as for the cluster confusion matrix above and restrict it to the 542 (out of 6005) best predicted neurons (with test set correlation ≥ 0.7). Note that some of the blocks from the cluster confusion matrix do not appear here, indicating that those clusters include poorly predicted neurons (e.g. block 17). The MEI confusion matrix exhibits a block-diagonal structure, with most MEIs driving neurons within the same blocks most strongly, albeit with different degrees of within-block similarity.

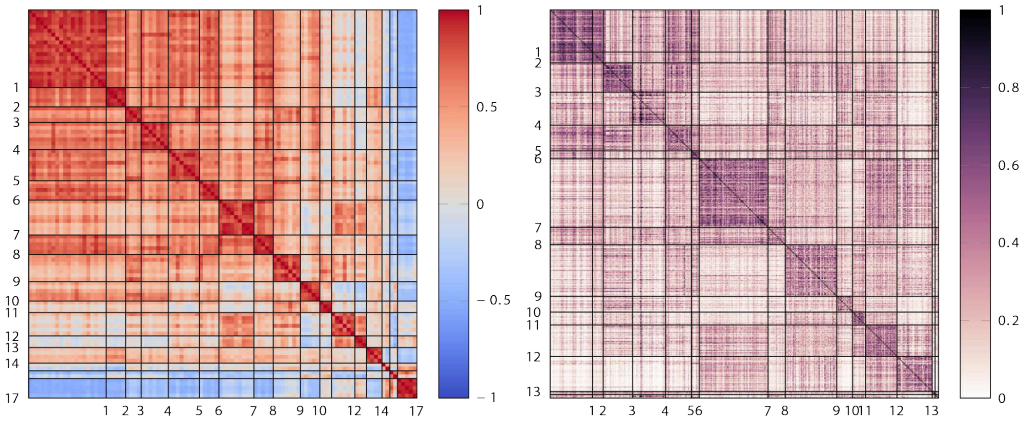


Figure 7: **Left:** Cluster confusion matrix (100×100) for 100 clusters shown in Fig. 6. Rows and columns have been arranged into 17 groups (blocks). **Right:** MEI confusion matrix for well-predicted neurons (test correlation ≥ 0.7) arranged into the same 17 blocks as on the left.

6 Conclusions and Future Work

We have presented an approach to clustering neurons into putative functional cell types invariant to location and orientation of their receptive field. We find around 10–20 functional clusters, the boundaries of some of which are not very clear-cut. To systematically classify the V1 functional cell types, these proposals need to be subsequently examined based on a variety of biological criteria reflecting the different properties of the neurons and the prior knowledge about the experiment.

REFERENCES

Ján Antolík, Sonja B. Hofer, James A. Bednar, and Thomas D. Mrsic-flogel. Model Constrained by Visual Hierarchy Improves Prediction of Neural Responses to Natural Scenes. *PLoS Comput Biol*, 2016.

Tom Baden, Philipp Berens, Katrin Franke, Miroslav Román Rosón, Matthias Bethge, and Thomas Euler. The functional diversity of retinal ganglion cells in the mouse. *Nature*, 529, 2016.

Spencer Ch. Brubaker and Santosh Vempala. Isotropic pca and affine-invariant clustering. In *Proceedings of the 2008 49th Annual IEEE Symposium on Foundations of Computer Science*, FOCS 2008, 2008.

Santiago A. Cadena, George H. Denfield, Edgar Y. Walker, Leon A. Gatys, Andreas S. Tolias, Matthias Bethge, and Alexander S. Ecker. Deep convolutional models improve predictions of macaque v1 responses to natural images. *PLoS computational biology*, 15(4), 2019.

Zhen Cui, Hong Chang, Shiguang Shan, and Xilin Chen. Generalized unsupervised manifold alignment. In *Advances in Neural Information Processing Systems (NIPS)*. 2014.

Ian L. Dryden and Kanti V. Mardia. *Statistical Shape Analysis*. Wiley, Chichester, 1998.

Lea Duncker and Maneesh Sahani. Temporal alignment and latent gaussian process factor inference in population spike trains. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems (NIPS)*. 2018.

Alexander S. Ecker, Fabian H. Sinz, Emmanouil Froudarakis, Paul G. Fahey, Santiago A. Cadena, Edgar Y. Walker, Erick Cobos, Jacob Reimer, Andreas S. Tolias, and Matthias Bethge. A rotation-equivariant convolutional neural network model of primary visual cortex. In *International Conference on Learning Representations*, 2019.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. Technical report, University of Montreal, 2009. Also presented at the ICML 2009 Workshop on Learning Feature Hierarchies, Montréal, Canada.

Brendan J Frey and Nebojsa Jojic. Fast, large-scale transformation-invariant clustering. In *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, 2002.

James C. Gower. Generalized procrustes analysis. *Psychometrika*, 40(1), 1975.

Xu Ju, João F. Henriques, and Andrea Vedaldi. Invariant information clustering for unsupervised image classification and segmentation. In *The IEEE International Conference on Computer Vision (ICCV)*, 2019.

Ieva Kazlauskaite, Carl Henrik Ek, and Neill Campbell. Gaussian process latent variable alignment learning. In *AISTATS*, volume 89. PMLR, 2019.

William F. Kindel, Elijah D. Christensen, and Joel Zylberberg. Using deep learning to probe the neural code for images in primary visual cortex. *Journal of Vision*, 19(4), 2019.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations (ICLR)*, 2015.

David Klindt, Alexander S. Ecker, Thomas Euler, and Matthias Bethge. Neural system identification for large populations separating "what" and "where". In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

Sebastian A. Kurtek, Anuj Srivastava, and Wei Wu. Signal estimation under random time-warpings and nonlinear signal alignment. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems (NIPS)*. 2011.

Lane McIntosh, Niru Maheswaranathan, Aran Nayebi, Surya Ganguli, and Stephen Baccus. Deep learning models of the retinal response to natural scenes. In *Advances in Neural Information Processing Systems (NIPS)*. 2016.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 2011.

Joshua R. Sanes and Richard H. Masland. The types of retinal ganglion cells: current status and implications for neuronal classification. *Annual review of neuroscience*, 38, 2015.

Thaddeus Tarpey. Linear Transformations and the k-Means Clustering Algorithm: Applications to Clustering Curves. *The American Statistician*, 61, 2007.

Laurens van der Maaten and Geoffrey E. Hinton. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 9, 2008.

Edgar Y. Walker, Fabian H. Sinz, Emmanouil Froudarakis, Paul G. Fahey, Taliah Muhammad, Alexander S. Ecker, Erick Cobos, Jacob Reimer, Xaq Pitkow, and Andreas S. Tolias. Inception in visual cortex: in vivo-silico loops reveal most exciting images. *bioRxiv*, 2018.

Chang Wang and Sridhar Mahadevan. Manifold alignment using procrustes analysis. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, 2008.

Feng Zhou and Fernando De la Torre. Generalized time warping for multi-modal alignment of human motion. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

A RANDOM PERMUTATIONS OF FEATURES

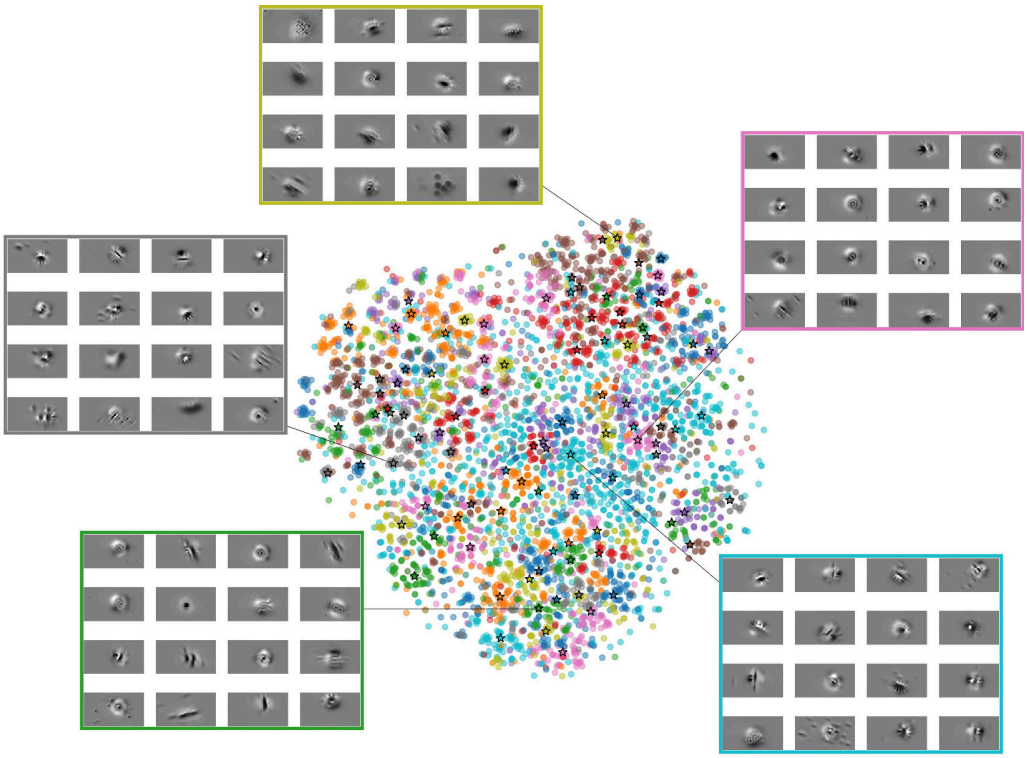


Figure A1: 2D t-SNE embedding of the aligned readouts $\tilde{\boldsymbol{R}}$ with feature weights randomly permuted for each of the neurons. The colors correspond to the GMM clustering with 100 components. Black stars show the locations of cluster centers. For some of the clusters, the MEIs of 16 best predicted neurons of that cluster are shown.

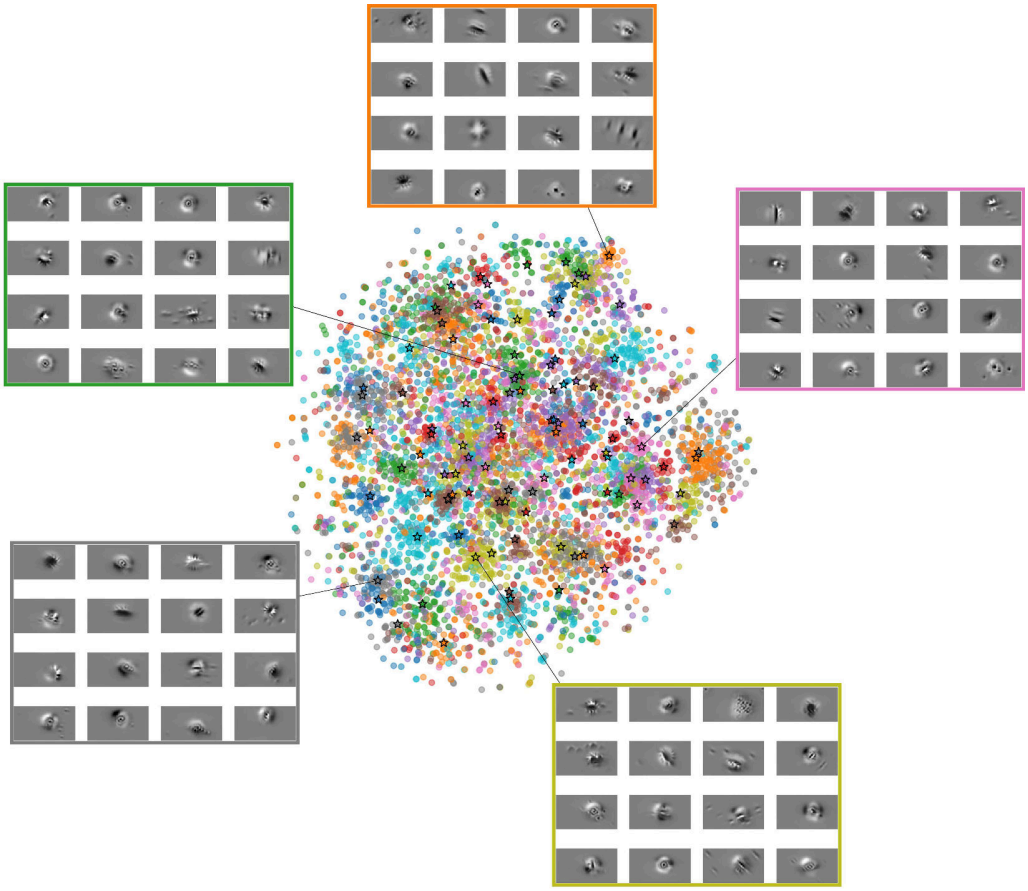


Figure A2: 2D t-SNE embedding of the aligned readouts $\tilde{\boldsymbol{R}}$ with feature weights randomly permuted across the neurons. The colors correspond to the GMM clustering with 100 components. Black stars show the locations of cluster centers. For some of the clusters, the MEIs of 16 best predicted neurons of that cluster are shown.

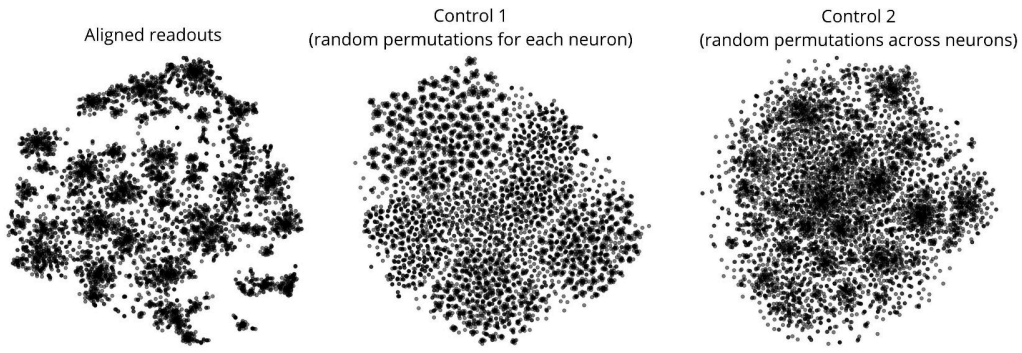


Figure A3: t-SNE embeddings for the aligned readouts (Fig. 6), and the controls with randomly permuted features for each neuron (Fig. A1) and across the neurons (Fig. A2).

B SYNTHETIC DATASET: DEPENDENCE ON NOISE

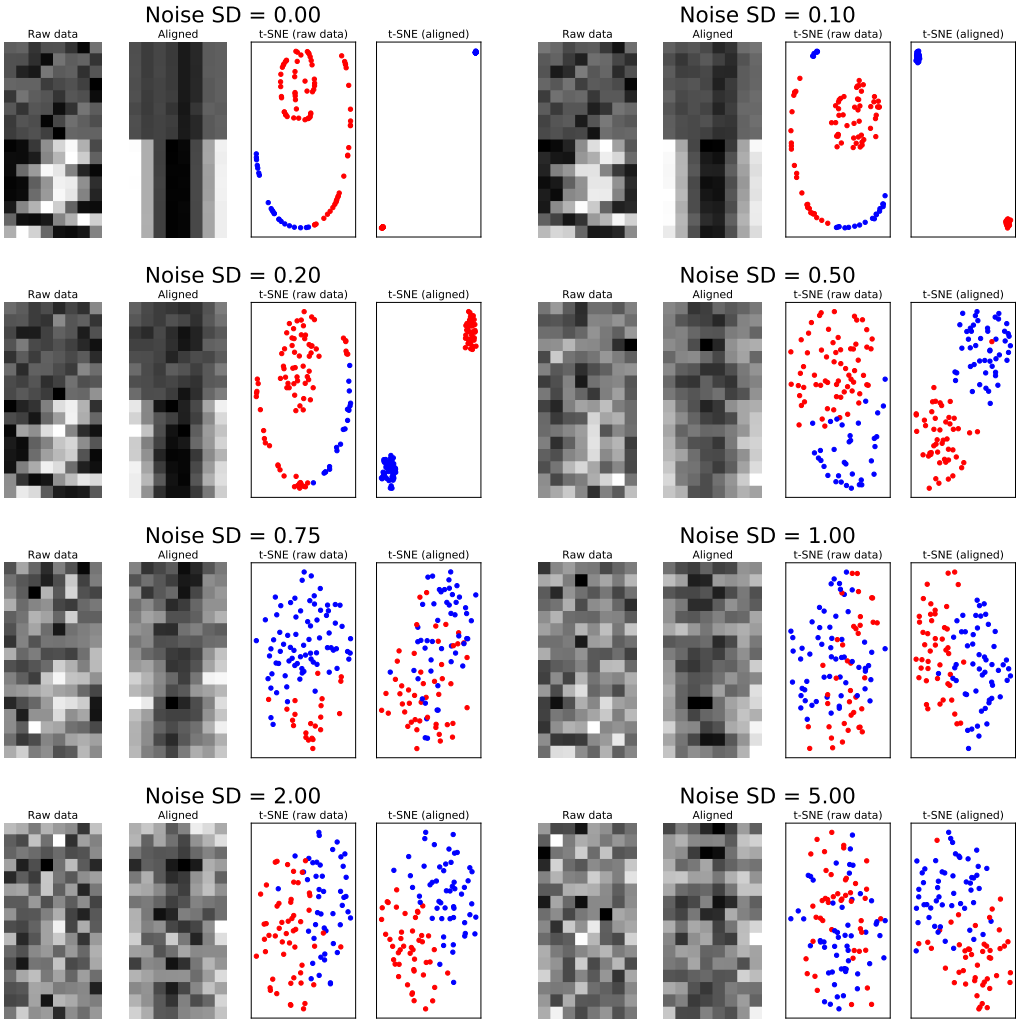


Figure B1: Alignment of a synthetic dataset of 100 observations generated using the procedure described in Sec. 4 for different amount of i.i.d. Gaussian noise added to the observations. The panels for each noise level show the 16 (out of 100) examples of the raw and aligned data as well the t-SNE embeddings of raw and aligned data coloured according to the GMM clustering with two components.

C MERGES AND SPLITS OF CLUSTER CONFUSION MATRIX BLOCKS

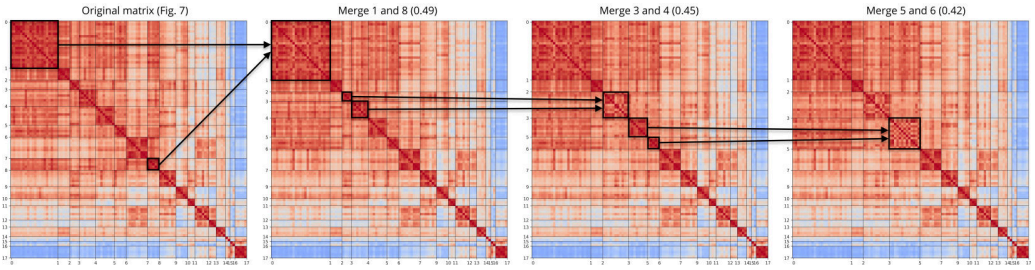


Figure C1: Sequential merges of the three pairs of blocks with the highest correlations in the cluster confusion matrix (Fig. 7, left). The merged blocks and the correlation values are shown in the titles of panels.

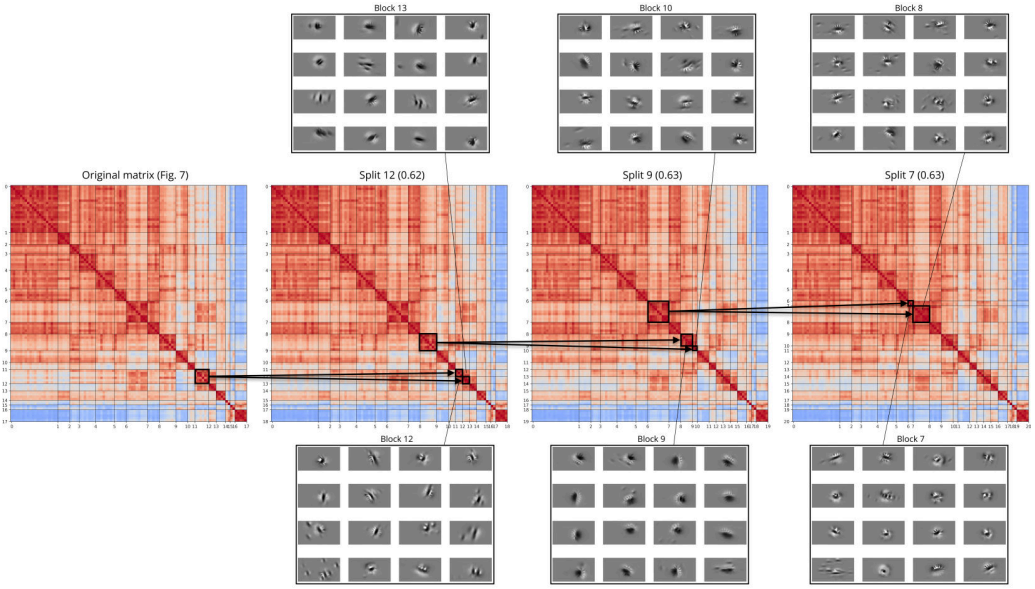


Figure C2: Sequential splits of the three pairs of blocks in the cluster confusion matrix (Fig. 7, left). The merged blocks, the correlation values, and the examples of MEIs of one of the GMM clusters in each of the splitted blocks are shown for each splitting step.

Digital twin reveals combinatorial code of non-linear computations in the mouse primary visual cortex

Ivan Ustyuzhaninov[1,7], Max F. Burg[1,2,7], Santiago A. Cadena[1,2,7], Jiakun Fu[4,5], Taliah Muhammad[4,5], Kayla Ponder[4,5], Emmanouil Froudarakis[4,5], Zhiwei Ding[4,5], Matthias Bethge[7], Andreas S. Tolias[4,5,6], and Alexander S. Ecker[2,3,✉]

[1]International Max Planck Research School for Intelligent Systems, University of Tübingen, Germany
[2]Institute of Computer Science and Campus Institute Data Science, University of Göttingen, Germany
[3]Max Planck Institute for Dynamics and Self-Organization, Göttingen, Germany
[4]Center for Neuroscience and Artificial Intelligence, Baylor College of Medicine, Houston, TX, USA
[5]Department of Neuroscience, Baylor College of Medicine, Houston, TX, USA
[6]Department of Electrical and Computer Engineering, Rice University, Houston, TX, USA
[7]Institute for Theoretical Physics, University of Tübingen, Germany

More than a dozen excitatory cell types have been identified in the mouse primary visual cortex (V1) based on transcriptomic, morphological and *in vitro* electrophysiological features. However, the functional landscape of excitatory neurons with respect to their responses to visual stimuli is currently unknown. Here, we combined large-scale two-photon imaging and deep learning neural predictive models to study the functional organization of mouse V1 using *digital twins*. Digital twins enable exhaustive *in silico* functional characterization providing a *bar code* summarizing the input-output function of each neuron. Clustering the bar codes revealed a continuum of function with around 30 modes. Each mode represented a group of neurons that exhibited a specific combination of stimulus selectivity and nonlinear response properties such as cross-orientation inhibition, size-contrast tuning and surround suppression. These non-linear properties were expressed independently spanning all possible combinations across the population. This combinatorial code provides the first large-scale, data-driven characterization of the functional organization of V1. This powerful approach based on digital twins is applicable to other brain areas and to complex non-linear systems beyond the brain.

Correspondence: ***Alexander S. Ecker, ecker@cs.uni-goettingen.de***

Introduction

Understanding the functional organization of the primary visual cortex (V1) has been a longstanding goal in neuroscience. It has long been known that V1 extracts information about local orientation Hubel & Wiesel (1959), often in a phase-invariant manner (Hubel & Wiesel, 1962). Researchers have described additional V1 nonlinearities, including direction selectivity (Adelson & Bergen, 1985) and various forms of nonlinear contextual modulation (Blakemore & Tobin, 1972; Cavanaugh et al., 2002; DeAngelis et al., 1992; Gilbert & Wiesel, 1990; Heeger, 1992; Lamme, 1995; Morrone et al., 1982). However, although we know many of the building blocks of V1 function, we do not know how they are organized at the population level.

First, we do not know whether there exists a distinct number of functional cell types, each of which implements a specific computation, or whether there is a continuum of function, where cells do not fall into discrete types. Second, independent of whether V1 functions are discrete or form a continuum, we currently do not know how the different nonlinear effects described previously are organized at the population level: are they strongly correlated – for instance because they are caused by a common computational mechanism – or are they present independently of each other within the population?

A major roadblock in revealing the functional organization of V1 has been that traditional experiments probing the well-known nonlinear mechanisms do not scale well. Large-scale population recordings are inefficient, because stimuli need to be optimized to an individual neuron's receptive field location, preferred orientation and spatial frequency. In addition, probing all nonlinear mechanisms in the same neurons is difficult because only a limited number of stimuli can be shown in an experiment.

We have overcome these limitations by combining large-scale population recordings with natural stimuli and training high-performance predictive models based on deep neural networks (Antolík et al., 2016; Batty et al., 2016; Cadena et al., 2019; Cotton et al., 2020; Klindt et al., 2017; Lurz et al., 2020; Sinz et al., 2018; Walker et al., 2019). These models are capable of jointly modeling thousands of neurons in a completely data-driven way providing a digital twin: an *in silico* approximation of the function of primary visual cortex (Fig. 1A). First, this approach allows us to quantify the similarity of neurons' response properties on the set of natural stimuli by computing a compact, low-dimensional vector representation of each neuron's function (its bar code). This representation is independent of the neuron's receptive field location and its preferred orientation and provides an unbiased metric to measure the similarity of two neurons' functions. It therefore provides a principled way to study the functional organization of V1. Second, the digital twin allows us to carry out experiments with arbitrary stimuli *in silico*, essentially with-

out limitations of experimental time to generate hypothesis which can then verified back *in vivo* using the inception loop paradigm (Bashivan et al., 2019; Walker et al., 2019). This systematic functional analysis allows us to gain interpretable insights from the model and link to existing literature.

We found that the functional organization is not entirely uniform, revealing a number of high-density modes. We therefore used the bar codes to cluster functionally similar neurons, which allowed us to analyze the neurons' functional properties at the cluster level.

Crucially, our analysis revealed that classical non-linear properties of neurons in V1 are expressed independently of each other. For instance, knowing the extent of a neuron's non-linearity along the simple-complex cell axis does not provide much information about its degree of surround suppression or cross-orientation inhibition. Moreover, there exist functional clusters expressing all combinations of nonlinear properties (including none or all), suggesting that V1 neurons might be described with a combinatorial code in the space of basic nonlinear computations.

Overall, our results suggest the following answers to the two questions posed above. The functional organization of V1 appears to form a continuum; however, it is not uniform and there are high-density modes in the space of V1 neurons' functions. This organization is consistent with recent work using transcriptomic, morphological, and electrophysiolocal properties, which showed that cortical neurons are organized in families with a continuum of properties within them rather than distinct cell types Gouwens et al. (2020); Network (2021); Scala et al. (2021). With respect to classical nonlinearities, V1 neurons can be described by a combinatorial code where each nonlinear computation is expressed along an independent axis across the population. Such factorized codes have computational advantages such as higher coding capacity (Fusi et al., 2016).

Results

Large-scale recording and predictive modeling. We recorded the activity of more than 45,000 excitatory neurons in layer 2/3 of the primary visual cortex of seven mice using a wide-field two-photon microscope (Sofroniew et al., 2016, Fig. 1A). While we imaged, the mice were head-fixed on a linear treadmill and were viewing natural images, which covered roughly $120° \times 90°$ of their visual field (Fahey et al., 2019; Walker et al., 2019). Next, we selected up to 2,000 neurons from each mouse and fitted a single predictive model for all mice (Lurz et al., 2020). The model is based on a convolutional neural network (CNN). It takes as input the image on the screen and outputs a prediction of the response of each neuron (Fig. 1C). The model achieved single trial test correlation of 0.42, and oracle correlation of 0.69. From this model, we obtained a 128-dimensional vector representation of each neuron's function. These vectors can be thought of as "bar codes" summarizing the neuron's stimulus-response function (Fig. 1D).

To describe the neurons' functional diversity, we removed two well-known factors of variation across V1 neurons: receptive field position and preferred orientation. The bar codes we obtained from our model were independent of receptive field position and preferred orientation of the neuron: if the responses of two neurons could be made identical by applying a constant shift and rotation to all images, these two neurons would obtain the same bar code. We achieved this property by using a rotation-equivariant CNN (Ecker et al., 2019; Ustyuzhaninov et al., 2020). Having bar codes that are independent of receptive field location and preferred orientation is extremely useful, because it removes two "trivial" axes of variation and allows us to focus on and visualize more subtle aspects of the neurons' selectivity or nonlinear processing.

Predictive modeling reveals functional clusters. We first asked whether V1 neurons are organized into discrete functional types or rather form a continuum. A 2D t-SNE embedding (van der Maaten & Hinton, 2008) of the bar codes (Fig. 1E) revealed several modes – or regions of high density. These modes correspond to groups of functionally similar neurons. While there is no strong evidence for discrete functional types, it is not a uniform distribution either. We performed k-means clustering (MacQueen et al., 1967) (using 50 clusters) to identify the modes of the distribution and simplify downstream analysis.

Neurons within functional clusters have similar MEIs. Given that V1 neurons can be organized into functional clusters, we aim at characterizing these clusters. We start by computing the preferred stimulus of each neuron, sometimes referred to as the most exciting image (MEI), which is optimized, using the model, to maximize the neuron's predicted activity (Fig. 2A). They have been shown to provide a faithful snapshot of neural computations (Bashivan et al., 2019; Walker et al., 2019), and therefore provide convenient single-image visualizations of a neuron's selectivity. Neurons within the same cluster had similar MEIs (up to location and rotation), while MEIs of neurons in different clusters tended to be different (Fig. 1F). This result provides a first piece of evidence that the clusters are a meaningful way of describing the functional organization of V1.

Note that our data does reproduce the large degree of heterogeneity of MEIs and their frequent striking deviations from Gabor filters (Fig. 1F) that have been reported previously for mouse V1 (Walker et al., 2019).

Cluster MEIs visualize functional similarities between the neurons. To focus more on commonalities of functionally similar neurons, we next computed optimal stimuli at the cluster level. To do so, we computed cluster MEIs: image templates that maximize average activity of neurons in

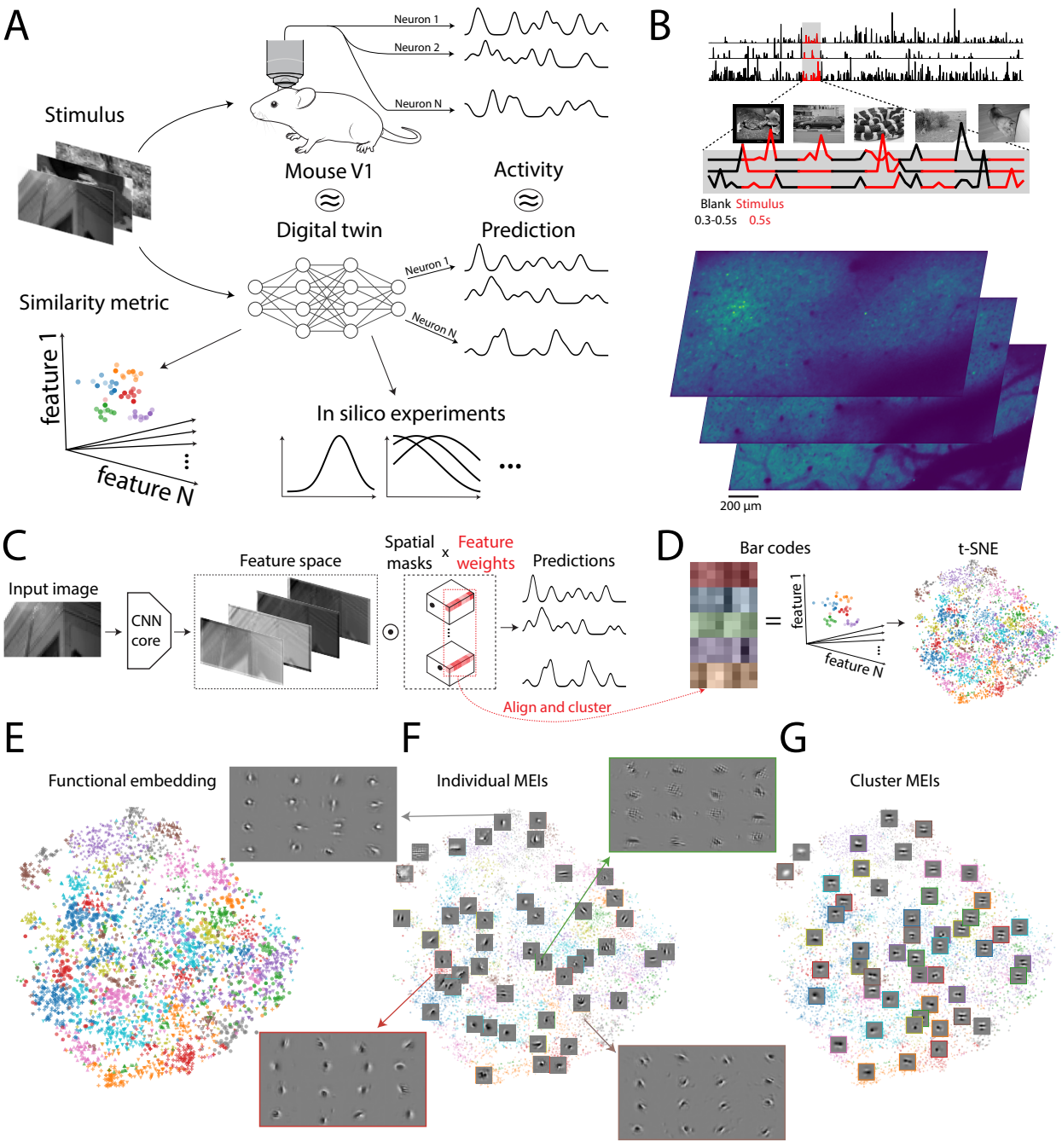


Fig. 1. A. Overview of our method. We presented natural images to a mouse and recorded corresponding responses of a large population of neurons in the primary visual cortex. This dataset allowed us to build a "digital twin" model of the mouse primary visual cortex which provided a functional similarity metric between the neurons, as well as enabled us to perform *in silico* experiments. **B.** Data recording paradigm. We presented an alternating sequence of 4692 natural and blank images to seven different mice. We showed natural images for 0.5s and blank ones for a random duration between 0.3s and 0.5s. We presented an additional test set of 100 images 10 times each. We recorded responses of 5 to 8 thousand V1 L2/3 neurons depending on the scan using a wide-range two-photon microscope. Processed calcium traces for three randomly chosen neurons are shown in the top, and raw scans at three different depths are shown in the bottom. **C.** Model fitting paradigm. We pooled the data from all 7 mice in a single dataset and fitted a rotation-equivariant CNN model to predict the recorded neural activity. The model consists of a rotation-equivariant convolutional core shared across neurons and neuron-specific linear readouts. For each neuron the readout is decomposed into a spatial mask encoding the spatial location of its receptive field and a vector of feature weights encoding predictive CNN features for this neuron. Feature weights can be thought of as "bar codes" summarizing a neuron's function. **D.** Functional clustering. We collected the feature weights for all neurons into a single matrix (row-wise) and aligned its rows by cycling shifts to remove the differences due to different preferred orientations of the neurons. We then clustered the rows of the aligned feature weights matrix into 50 clusters using the k-Means algorithm. The aligned feature weights and the clusters are visualized using a 2D t-SNE embedding. **E.** A 2D t-SNE functional emedding of the recorded neurons colored according to the cluster assignment. **F.** Examples of MEIs of 16 best predicted neurons in 4 different clusters alongside examples of MEIs of other neurons on top of a t-SNE embedding. **G.** Examples of cluster MEIs of other neurons on top of a t-SNE embedding.

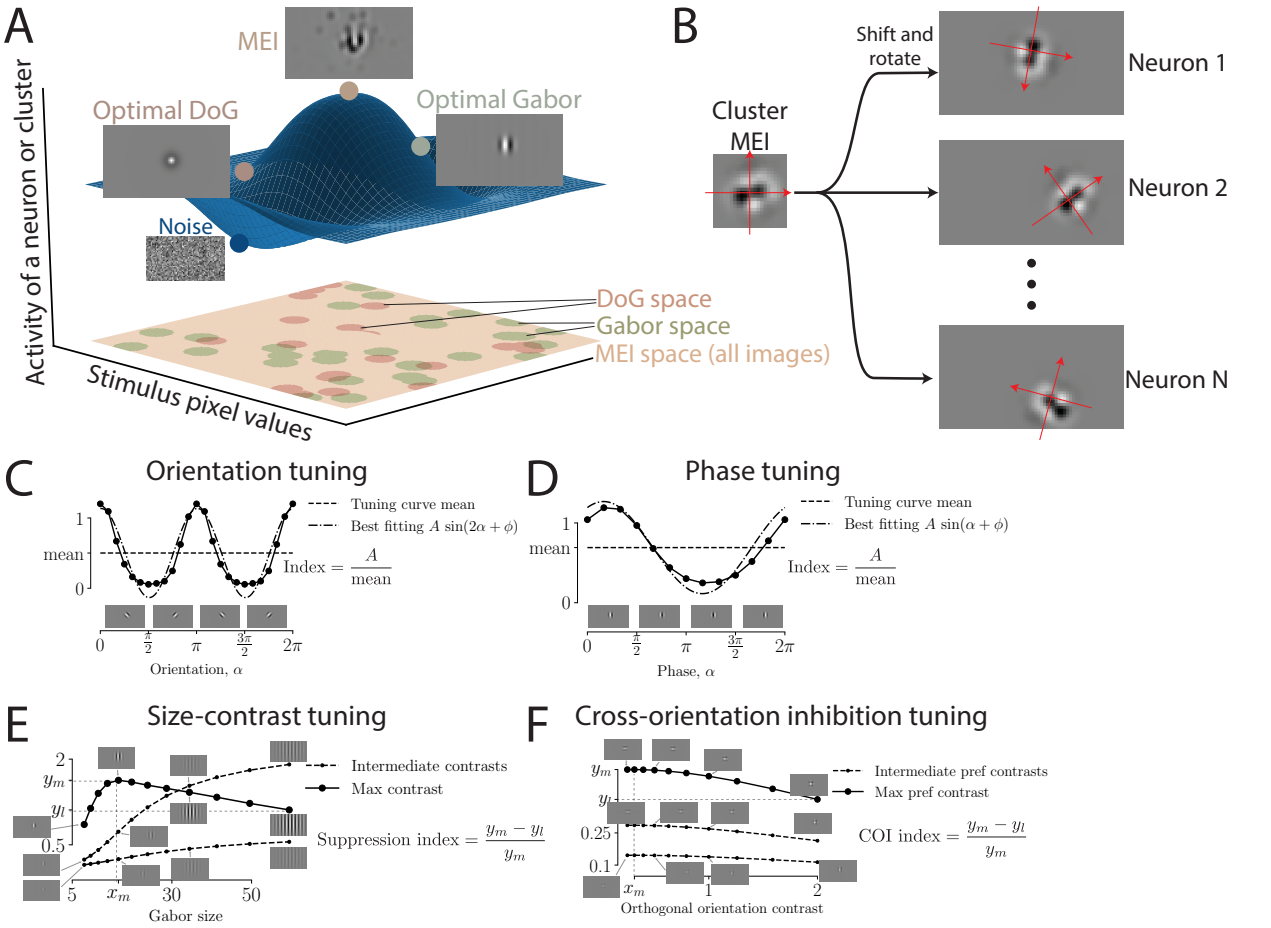


Fig. 2. A. Optimal stimuli. We maximize activity of a neuron or average activity of a cluster with respect to an input stimulus which is constrained to belong to a certain image space. An illustration shows a response surface of a neuron of a cluster and maximum values on the this surface while restricting ourselves to a specific image space shown in the XY plane. The space of possible MEIs contains all images, while the spaces of possible Gabors and DoGs are subsets of all images. **B.** Average cluster activity for a given stimulus is computed by averaging the responses of neurons in the cluster to the stimuli shifted and rotated to match the location of the receptive field and preferred orientation of each neuron in that cluster. **C-D.** Examples of orientation and phase tuning curves for a single neuron. We vary orientation and phase of an optimal Gabor while keeping all other parameters fixed to generate stimuli for orientation and phase tuning experiments. The numerical tuning indices for these tuning curves are computed by fitting a sine curve and taking the ratio of its amplitude to the mean of the tuning curve. **E.** Example of a size-contrast tuning curve for a single neuron. The stimuli for the size-tuning experiment are constructed by varying size and contrast of an optimal Gabor while keeping all other parameters fixed. The suppression tuning strength is computed for a tuning curve corresponding to the highest contrast as a relative decrease of activity when increasing the size of the Gabor beyond the size corresponding to the maximum value of the curve. The contrast tuning strength is computed analogously by transposing size and contrast, i.e. by considering a tuning curve corresponding to the largest size as a function of the contrast. **F.** Example of a cross-orientation inhibition (COI) tuning curve for a single neuron. The stimuli for this experiment are called plaids and constructed by overlaying the optimal Gabor and the Gabor orthogonal to it in different contrasts. The COI tuning strength index is computed using a tuning curve corresponding to the highest preferred contrast analogously to the suppression index for the size-contrast experiment.

each cluster when accounting for each neuron's receptive field location and preferred orientation (Fig. 2B). Cluster MEIs show a systematic variation along the different axes of the t-SNE embedding (Fig. 1G): Neighboring clusters tend to have visually similar cluster MEIs. There appears to be a global pattern in the t-SNE space with clusters on the right having oriented, Gabor-like MEIs with higher frequencies and multiple cycles within the envelope, while those in the middle having lower frequencies and fewer cycles and those towards the bottom left having more symmetric and circular MEIs.

Cluster MEIs visualize common patterns of cluster computations, and since they are designed to capture similarities rather than differences between the neurons, they exhibit less variability than individual MEIs. Quantitatively this amounts to cluster MEIs driving the neurons to around 60% of activity of their individual MEIs.

In silico experiments provide an interpretable characterization of functional clusters. While MEIs provide convenient visualizations of a neuron's or a cluster's computations, they capture only a single point – the maximum – of the tuning function (Fig. 2A). Our predictive model, however, provides a prediction for arbitrary stimuli. We used the model as an *in silico* replica of V1 to perform experiments. Unlike with experiments in the real brain, in the model we are not limited in terms of experimental time. This allowed us to replicate a number of classical experiments *in silico* and compute tuning curves with respect to a variety of different non-linear properties. Specifically, we used Gabor stimuli whose parameters were optimized for each cluster to quantify strength of orientation selectivity

(Fig. 2C), phase invariance (Fig. 2D), size-contrast tuning (Fig. 2E) and cross-orientation inhibition (Fig. 2F). Optimizing Gabors for clusters rather than individual neurons prevented the possibility of a few neurons within a cluster having very stimuli in comparison to the rest of the neurons in the cluster due to optimization instability. Furthermore, the cluster optimal Gabors drive the neurons to 87% of their individual optimal Gabor activity thus these stimuli are only mildly suboptimal for individual neurons.

Size-contrast tuning curves reveal non-linear surround suppression effects (Born & Tootell, 1991; DeAngelis et al., 1992), while cross-orientation inhibition is a nonlinear interaction that arises when two orthogonal Gabor patterns are superimposed (Morrone et al., 1982). In addition, we computed the optimal center-surround stimulus (difference of Gaussian; Fig. 2A) and quantified the degree of response nonlinearity using a generalized linear model baseline (see methods). From these in silico experiments, we obtain a sample from the joint tuning distribution for more than 10,000 neurons. This enables us to study the statistical dependencies between the different nonlinear effects, overcoming the limitations of previous *in vivo* experiments that could only study each effect in isolation.

In silico experiments reveal shared tuning properties within functional clusters. The results from the set of in silico experiments support the functional clustering (Fig. 3). Many of the modes in the t-SNE embedding are distinguishable based on one or more of the tuning properties. In contrast, neurons *within* most of the clusters exhibit similar tuning strengths to the different types of non-linarities: all 50 clusters were significantly different from the overall population tuning distribution based on at least one tuning property in Fig.3 and 15 clusters were significantly different based on all properties (two-sided Kolmogorov-Smirnov test at $\alpha = 0.01$; in the case of random cluster assignments these values are 0 and 1 respectively).

Non-linear tuning properties are independent of each other. Next, we investigated how different non-linear properties relate to each other. Qualitatively, it appears that different non-linear properties have different distributions (Fig. 3; compare the color patterns in the t-SNE plots). This qualitative impression is also confirmed by quantitative metrics (Fig. 4A). As expected, non-linearity is correlated with specific nonlinear properties like phase invariance, surround suppression and cross-orientation inhibition, but not with orientation selectivity. In addition, orientation selectivity is correlated with phase invariance and cross-orientation inhibition. Importantly, there is no correlation between the three non-linear properties phase invariance, surround suppression and cross-orientation inhibition, suggesting that these properties are independently exhibited from each other. This result is surprising, since cross-orientation inhibition and surround suppression have both been hypothesized to arise from a common mechanism: divisive normalization (Carandini & Heeger, 2012).

Because phase invariance, cross-orientation inhibition and surround suppression are tested using oriented Gabors as stimuli, we restricted our subsequent analyses to those clusters for which optimal Gabors are decent stimuli in comparison to MEIs (i.e. clusters with above average values of the Gabor vs MEI index). We binarized the tuning indices for these non-linear properties assigning each cluster to either high or low tuning category (Fig. 4B) by setting a threshold at the average population tuning strength and examined the combinations of these three tuning properties (Fig. 4C). We can see there are clusters exhibiting every possible combination of the three binary tuning properties, suggesting that these properties indeed appear to be independent from each other, and that V1 neurons might employ a combinatorial code with respect to these nonlinearities.

For the analysis of remaining clusters (i.e. clusters with below average values of the Gabor vs MEI index, Fig. 4D) we considered orientation, non-linearity, and DoG vs MEI tuning. These clusters are mostly linear tuned and unoriented with their cluster MEIs ranging from mostly center-surround shapes in the bottom-right and central parts of the t-SNE space to various complex shapes in the left part of the t-SNE space.

In silico tuning curves are good approximations of in vivo tuning. Since our analysis is based on in silico tuning curves, one concern could be that although the CNN model predicts neural activities with high accuracy, the corresponding in silico tuning curves might be different from the in vivo tuning curves we are aiming to approximate. We verified that it is not the case by directly comparing the in silico and in vivo tuning curves for the same neurons. Specifically, we recorded a dataset containing two V1 scans of the same neurons in the same mouse. We used natural images as stimuli for the first scan which allowed us to fit a rotation-equivariant CNN model. For the second scan we used Gabor stimuli allowing us to compute size tuning curves in vivo and compare them to the in silico tuning curves obtained from the model fitted to the first scan (Fig. 5A).

The stimuli for the size tuning experiment should ideally be constructed based on optimal Gabors for every neuron, however, that is infeasible in vivo for a sufficiently large population of neurons. To overcome this limitation, we investigated to what extent optimal Gabors could be replaced with suboptimal ones. We found that tuning curves obtained using Gabors driving neurons to at least 50% of their optimal Gabor activities are very similar to the tuning curves obtained with optimal Gabors (Fig. 5C), and that only 20 different Gabors can be chosen to drive 75% of the population to at least 50% of optimal Gabor activities (Fig. 5D-E). This observation allowed us to use these 20 different Gabors as stimuli for the Gabor scan, which revealed that the correlation between in vivo and in sil-

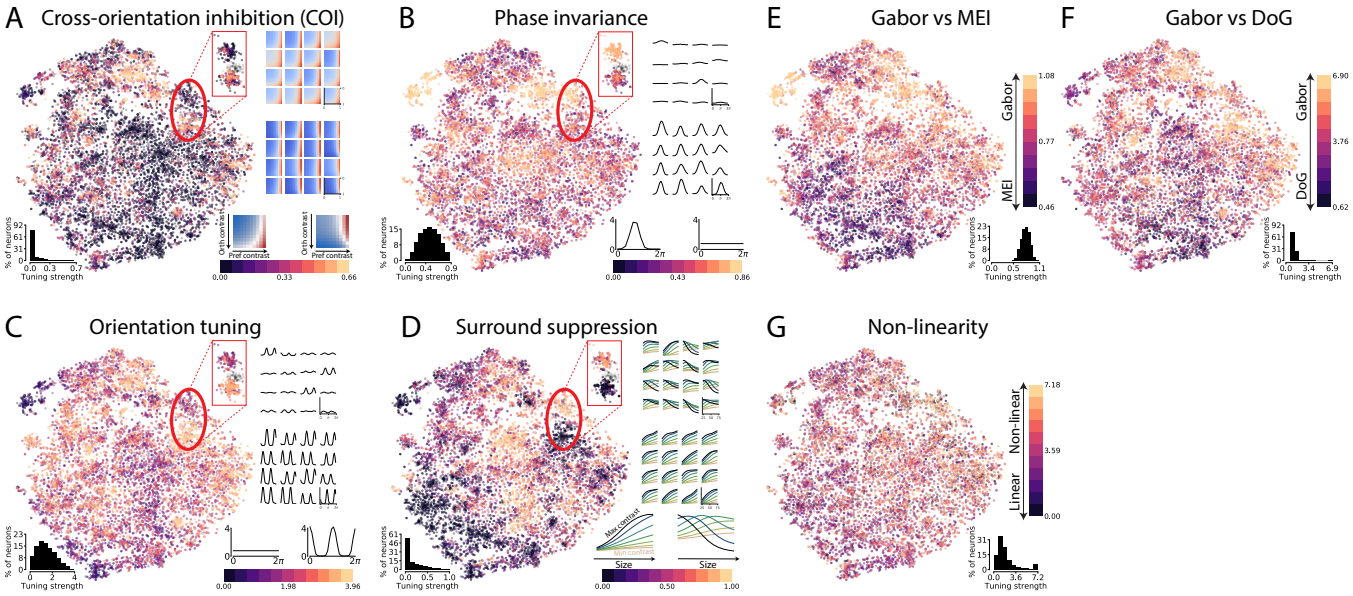


Fig. 3. Results of the in silico experiments. **A-D:** The scatter plots show t-SNE embeddings colored according to the tuning strengths of cross-orientation inhibition, phase invariance, orientation tuning and surround suppression experiments. In the bottom left corners of t-SNE embeddings we show histograms of distributions of the tuning strengths. We additionally show the tuning curves for the 16 best predicted neurons in two different clusters in a separate close-up panel, as well as examples of tuning curves of high and low tuning strengths alongside the colorbar. **E-G:** t-SNE embeddings colored according to the tuning strength of Gabor vs MEI, Gabor vs DoG and non-linearity experiments along with histograms of tuning strengths.

ico tuning curves is about 70% of the oracle correlation (Fig. 5B), which is similar to the performance of the CNN model trained and tested on natural images.

Discussion

We built a functional description of the mouse primary visual cortex based on neural representations in a high performing CNN model predicting responses of a large population of neurons on arbitrary natural images. Such an approach allows us to account for all aspects of the neuronal stimulus-response function captured by the model, instead of only a few nonlinear effects as in classical in vivo experiments with parametric stimuli. Thus, our analysis is not constrained by specific hypotheses of neural functions or the choice of parametric stimuli. An important limitation of our study is that we focus on the bottom-up aspects of stimulus processing; we did not consider how top-down, behavioral modulation of neural responses affects the responses of different neurons.

Our examination revealed that the V1 functional landscape can be described by around 30 modes of functionally similar neurons which, however, do not appear to be discrete cell types but rather high density areas in the continuous functional space. This finding is in agreement with various recent studies. For example, Scala et al. (2021) studied mouse primary motor cortex neurons based on transcriptomic and morpho-electric properties. They found that this brain area is organized into a few broad transcriptomic families with continuum of morpho-electric features in each family, making the authors question the existence of discrete transcriptomic cell types. Gouwens et al. (2020) conducted a similar study of interneurons in mouse primary visual cortex discovering both discrete and continuous variation of morpho-electric properties within the transcriptomic types. Overall, a growing body of literature suggests that mouse neocortex is organized in a complex way and cannot be adequately described by either discrete cell types or a uniform continuum of neurons. This notion of mouse neocortex organization is qualitatively different from the mouse retina, which exhibits discrete cell types (Baden et al., 2016), raising interesting directions for future research: how are other areas of neocortex organized based on various neural properties? and how do they relate to lower level brain areas projecting into the corresponding neocortical areas?

As we investigated the variability of individual neurons within the functional modes, we found visual differences between cluster and single neuron MEIs. We believe this is an expected consequence of clustering, which abstracts away individual neurons' differences and reduces the complexity of the V1 functional space by focusing on similarities between neurons. To better understand the properties of functional modes, their correspondence to cluster MEIs and relate them to previous work, we performed in silico experiments, which revealed that neurons belonging to the same mode exhibit common response patterns. This observation suggests that neurons in the same functional cluster may form computational cliques important for downstream processing, offering an interesting direction for future research. For example, integrated approaches combining functional characterization using digital twins and connectomics data can determine the connectivity of neurons within and across functional clusters and commonalties of their inputs and projection targets (Bae et al., 2021).

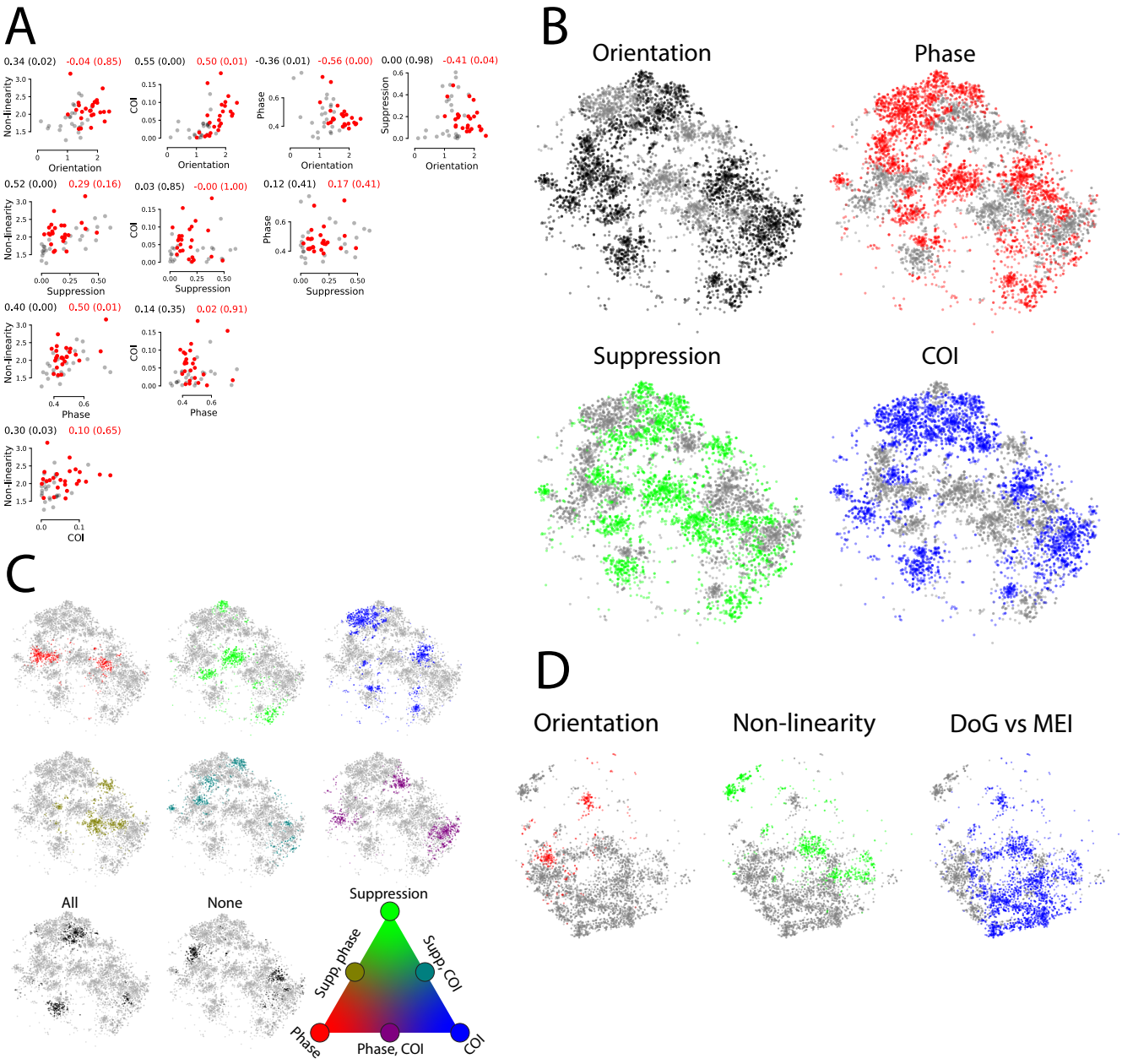


Fig. 4. A. Pairwise distributions of the in silico tuning properties. Dots in each of the plots correspond to clusters (50 dots in total) showing average tuning strength of neurons in the cluster. Red dots correspond to clusters with the above average value of the Gabor vs MEI tuning strength (i.e. those clusters for which optimal Gabors are good stimuli relative to MEIs; in the following called Gabor-like clusters), gray dots correspond to all other clusters. Correlation coefficients and the p-values (in brackets) under the null hypothesis that the correlation is zero are shown above the plots for Gabor-like clusters (red) and for all clusters (black). **B.** Subsets of t-SNE embeddings corresponding to Gabor-like clusters colored according to the binarized strengths of orientation, phase, suppression and plaids tuning. Grey clusters correspond to low tuning strength (average cluster tuning strength is less than entire population tuning strength average value), clusters of the other color correspond to high tuning strength. **C.** Subsets of t-SNE embeddings corresponding to Gabor-like clusters showing 8 possible combinations of low/high values of phase, suppression and plaids tuning. The color code is illustrated with a color triangle; clusters colored with the colors in the vertices of the triangle exhibit high value of the corresponding tuning property and low values of other two properties. Clusters colored with the colors in the edges of the triangle exhibit high values of the tuning properties in the adjacent vertices and low value of the other tuning property. **D.** Subsets of t-SNE embeddings corresponding to non Gabor-like clusters colored according to the binarized strengths of orientation, non-linearity and DoG vs MEI tuning.

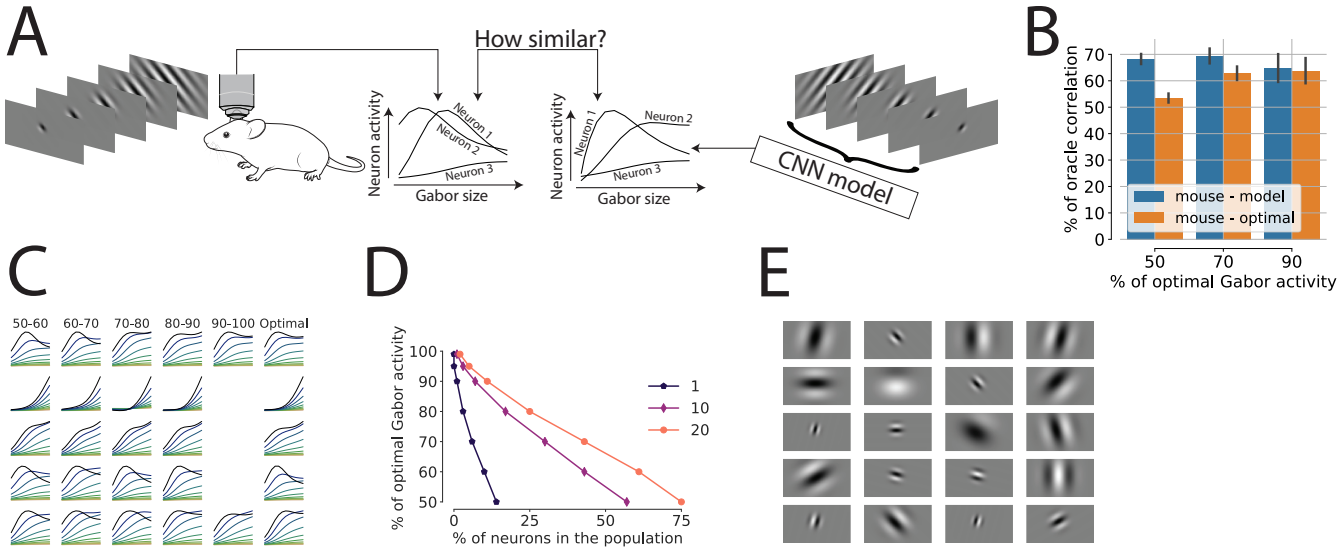


Fig. 5. A. In vivo verification experiment paradigm. We compare in vivo Gabor size tuning curves to their in silico counterparts computed for the same population of neurons. **B.** Correlations between in vivo and in silico Gabor size tuning curves measured as percentage of oracle correlation (error bars shows the standard deviation). We measure correlation between two types of tuning curves. The "mouse - model" one corresponds to the in vivo and in silico tuning curves obtained using the same experimental stimuli, which are based on suboptimal Gabors (see main text for more details). In the "mouse - optimal" case, in vivo curves are computed using suboptimal Gabors, but the in silico curves are computed using optimal Gabors for the corresponding neurons. Moreover, these correlations are computed for different subsets of neurons (x-axis) chosen such that there exists an experimental stimulus for every neruon in the subset driving this neuron to a certain percentage of the optimal Gabor activity. **C.** Examples of Gabor size-contrast in silico tuning curves for 5 randomly chosen neruons (rows) computed using suboptimal Gabors activating the neuron to a certain percentage of the optimal Gabor activity (first 5 columns; numbers above the first row show the suboptimal Gabor activity as percentage of the optimal Gabor activity) and using the optimal Gabors (last column). **D.** Percentages of neurons in the population (x-axis) that can be driven to a certain percentage of the optimal Gabor activity (y-axis) using at least one of the Gabors in the sets of 1, 10 and 20 Gabors (different curves) chosen to maximize the number of neurons activated by these stimuli. **E.** 20 Gabor stimuli corresponding the 20 Gabors curve in panel **D** and used to construct size-tuning stimuli for the in vivo experiment.

Large-scale in-silico experiments allowed us to study statistical dependencies between various nonlinear phenomena known from single-neuron in-vivo experiments. We found these effects to be independent of each other, suggesting that V1 might employ a combinatorial code between modes of functionally similar cells. The mechanisms leading to such a code in V1 and their implications for downstream processing remain unclear. A speculation that lies at hand is that there might be a basis of independent non-linear computations serving different purposes in downstream processing, thereby building a foundation for specializations in higher visual areas. As a potential verification of this hypothesis and as a question in itself, future experimental work could investigate if neurons of the same functional cluster project to the same downstream area.

Finally, we verified that in silico experiments in a high-performing CNN model provide a good approximation of the in vivo tuning curves, thus substantiating our results based on the analysis of in silico tuning curves. Our in vivo verification is consistent with the findings of Walker et al. (2019) who report that in silico model MEIs also highly activate actual neurons. Overall, these observations suggest that high-perfroming CNN models can be considered "digital twins" of real neurons, a paradigm that has started being explored relatively recently but already provided significant insights into the brain and has a potential of becoming the main tool for future research.

ACKNOWLEDGEMENTS

The research was supported by the German Federal Ministry of Education and Research (BMBF) via the Competence Center for Machine Learning (FKZ 01IS18039A); the German Research Foundation (DFG) grant EC 479/1-1 (A.S.E.), the Collaborative Research Center (SFB 1233, Robust Vision) and the Cluster of Excellence "Machine Learning – New Perspectives for Science" (EXC 2064/1, project number 390727645); the Bernstein Center for Computational Neuroscience (FKZ 01GQ1002); the National Eye Institute of the National Institutes of Health under Award Numbers U19MH114830 (A.S.T.) R01MH109556 (AST), P30EY002520, and the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior/Interior Business Center (DoI/IBC) contract number D16PC00003. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/IBC, or the U.S. Government. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

AUTHOR CONTRIBUTIONS

I.U., M.B., A.S.T and A.S.E. designed the study; J.F., T.M., K.P., E.F. and Z.D. performed imaging experiments and pre-processing of raw data; I.U., A.S.E. developed the in silico analysis framework; I.U., M.F.B., S.A.C. and A.S.E analyzed the data; I.U., M.F.B. and S.A.C. wrote the original draft; I.U., M.F.B, S.A.C. and A.S.E. reviewed and edited the manuscript with the input from M.B. and A.S.T.

Bibliography

Adelson, E. H., & Bergen, J. R. (1985). Spatiotemporal energy models for the perception of motion. *J. Opt. Soc. Am. A*, *2*(2), 284–299. URL http://www.osapublishing.org/josaa/abstract.cfm?URI=josaa-2-2-284

Antolík, J., Hofer, S. B., Bednar, J. A., & Mrsic-Flogel, T. D. (2016). Model constrained by visual hierarchy improves prediction of neural responses to natural scenes. *PLOS Computational Biology*, *12*(6), 1–22. URL https://doi.org/10.1371/journal.pcbi.1004927

Baden, T., Berens, P., Franke, K., Rosón, M. R., Bethge, M., & Euler, T. (2016). The functional diversity of retinal ganglion cells in the mouse. *Nature*, *529*(7586), 345–350.

Bae, J. A., Baptiste, M., Bodor, A. L., Brittain, D., Buchanan, J., Bumbarger, D. J., Castro, M. A., Celii, B., Cobos, E., Collman, F., et al. (2021). Functional connectomics spanning multiple areas of mouse visual cortex. *bioRxiv*.

Bashivan, P., Kar, K., & DiCarlo, J. J. (2019). Neural population control via deep image synthesis. *Science*, *364*(6439).

Batty, E., Merel, J., Brackbill, N., Heitman, A., Sher, A., Litke, A., Chichilnisky, E., & Paninski, L. (2016). Multilayer recurrent network models of primate retinal ganglion cell responses. In *International Conference on Learning Representations*.

Blakemore, C., & Tobin, E. A. (1972). Lateral inhibition between orientation detectors in the cat's visual cortex. *Experimental brain research*, *15*(4), 439–440.

Born, R. T., & Tootell, R. (1991). Single-unit and 2-deoxyglucose studies of side inhibition in macaque striate cortex. *Proceedings of the National Academy of Sciences*, *88*(16), 7071–7075.

Cadena, S. A., Denfield, G. H., Walker, E. Y., Gatys, L. A., Tolias, A. S., Bethge, M., & Ecker, A. S. (2019). Deep convolutional models improve predictions of macaque v1 responses to natural images. *PLoS computational biology*, *15*(4), e1006897.

Carandini, M., & Heeger, D. J. (2012). Normalization as a canonical neural computation. *Nature Reviews Neuroscience*, *13*(1), 51–62.

Cavanaugh, J. R., Bair, W., & Movshon, J. A. (2002). Nature and interaction of signals from the receptive field center and surround in macaque v1 neurons. *Journal of Neurophysiology*, *88*(5), 2530–2546. PMID: 12424292.
URL https://doi.org/10.1152/jn.00692.2001

Cotton, R. J., Sinz, F. H., & Tolias, A. S. (2020). Factorized neural processes for neural processes: k-shot prediction of neural responses. *arXiv preprint arXiv:2010.11810*.

DeAngelis, G. C., Robson, J. G., Ohzawa, I., & Freeman, R. D. (1992). Organization of suppression in receptive fields of neurons in cat visual cortex. *Journal of Neurophysiology*, *68*(1), 144–163. PMID: 1517820.
URL https://doi.org/10.1152/jn.1992.68.1.144

Ecker, A. S., Sinz, F. H., Froudarakis, E., Fahey, P. G., Cadena, S. A., Walker, E. Y., Cobos, E., Reimer, J., Tolias, A. S., & Bethge, M. (2019). A rotation-equivariant convolutional neural network model of primary visual cortex. In *International Conference on Learning Representations*.
URL https://openreview.net/forum?id=H1fU8iAqKX

Fahey, P. G., Muhammad, T., Smith, C., Froudarakis, E., Cobos, E., Fu, J., Walker, E. Y., Yatsenko, D., Sinz, F. H., Reimer, J., et al. (2019). A global map of orientation tuning in mouse visual cortex. *bioRxiv*, (p. 745323).

Fusi, S., Miller, E. K., & Rigotti, M. (2016). Why neurons mix: high dimensionality for higher cognition. *Current opinion in neurobiology*, *37*, 66–74.

Gilbert, C. D., & Wiesel, T. N. (1990). The influence of contextual stimuli on the orientation selectivity of cells in primary visual cortex of the cat. *Vision Research*, *30*(11), 1689–1701. Optics Physiology and Vision.
URL https://www.sciencedirect.com/science/article/pii/004269899090153C

Gouwens, N. W., Sorensen, S. A., Baftizadeh, F., Budzillo, A., Lee, B. R., Jarsky, T., Alfiler, L., Baker, K., Barkan, E., Berry, K., et al. (2020). Integrated morphoelectric and transcriptomic classification of cortical gabaergic cells. *Cell*, *183*(4), 935–953.

Heeger, D. J. (1992). Normalization of cell responses in cat striate cortex. *Visual neuroscience*, *9*(2), 181–197.

Hubel, D. H., & Wiesel, T. N. (1959). Receptive fields of single neurones in the cat's striate cortex. *The Journal of Physiology*, *148*(3), 574–591.
URL https://physoc.onlinelibrary.wiley.com/doi/abs/10.1113/jphysiol.1959.sp006308

Hubel, D. H., & Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology*, *160*(1), 106–154.
URL https://physoc.onlinelibrary.wiley.com/doi/abs/10.1113/jphysiol.1962.sp006837

Klindt, D., Ecker, A. S., Euler, T., & Bethge, M. (2017). Neural system identification for large populations separating "what"and "where". In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.) *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc.
URL https://proceedings.neurips.cc/paper/2017/file/8c249675aea6c3cbd91661bbae767ff1-Paper.pdf

Lamme, V. (1995). The neurophysiology of figure-ground segregation in primary visual cortex. *Journal of Neuroscience*, *15*(2), 1605–1615.
URL https://www.jneurosci.org/content/15/2/1605

Lurz, K.-K., Bashiri, M., Willeke, K. F., Jagadish, A. K., Wang, E., Walker, E. Y., Cadena, S., Muhammad, T., Cobos, E., Tolias, A., et al. (2020). Generalization in data-driven models of primary visual cortex. *bioRxiv*.

MacQueen, J., et al. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, (pp. 281–297). Oakland, CA, USA.

McCullagh, P., & Nelder, J. A. (2019). *Generalized linear models*. Routledge.

Morrone, M. C., Burr, D., & Maffei, L. (1982). Functional implications of cross-orientation inhibition of cortical visual cells. i. neurophysiological evidence. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, *216*(1204), 335–354.

Network, B. I. C. C. (2021). A multimodal cell census and atlas of the mammalian primary motor cortex. *Nature*, *598*(7879), 86.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision*, *115*(3), 211–252.

Scala, F., Kobak, D., Bernabucci, M., Bernaerts, Y., Cadwell, C. R., Castro, J. R., Hartmanis, L., Jiang, X., Laturnus, S., Miranda, E., et al. (2021). Phenotypic variation of transcriptomic cell types in mouse motor cortex. *Nature*, *598*(7879), 144–150.

Sinz, F. H., Ecker, A. S., Fahey, P. G., Walker, E. Y., Cobos, E., Froudarakis, E., Yatsenko, D., Pitkow, Z., Reimer, J., & Tolias, A. S. (2018). Stimulus domain transfer in recurrent models for large scale cortical population prediction on video. In *NeurIPS*.

Sofroniew, N. J., Flickinger, D., King, J., & Svoboda, K. (2016). A large field of view two-photon mesoscope with subcellular resolution for in vivo imaging. *Elife*, *5*, e14472.

Ustyuzhaninov, I., Cadena, S. A., Froudarakis, E., Fahey, P. G., Walker, E. Y., Cobos, E., Reimer, J., Sinz, F. H., Tolias, A. S., Bethge, M., & Ecker, A. S. (2020). Rotation-invariant clustering of neuronal responses in primary visual cortex. In *International Conference on Learning Representations*.
URL https://openreview.net/forum?id=rklr9kHFDB

van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-sne. *Journal of Machine Learning Research*, *9*(86), 2579–2605.
URL http://jmlr.org/papers/v9/vandermaaten08a.html

Walker, E. Y., Sinz, F. H., Cobos, E., Muhammad, T., Froudarakis, E., Fahey, P. G., Ecker, A. S., Reimer, J., Pitkow, X., & Tolias, A. S. (2019). Inception loops discover what excites neurons most using deep predictive models. *Nature neuroscience*, *22*(12), 2060–2065.

Methods

Visual stimuli. We used gray-scale ImageNet images (Russakovsky et al., 2015) as visual stimuli in the data collection experiment. The number of images varied across the scans (Tab. 1) with 4692 images in the intersection, i.e. presented to a mouse in each of the scans. For the test set we used 100 images each repeated 10 times; for some scans and some images there were fewer repeats, in which case we resampled the recording to have 10 repeats in every scan. The screen was 55 $\times$ 31 cm at a distance of 15 cm, covering roughly 120° $\times$ 90°. Each image was presented for 500 ms followed by a blank screen lasting between 300 ms and 500 ms. The response of a neuron to a given stimulus is represented as a number of spikes in the time interval between 50 ms to 350 ms following the stimulus onset.

CNN model and rotation-invariant clustering. We used the same architecture and training of the rotation-equivariant CNN model as in Ustyuzhaninov et al. (2020). For the clustering of aligned feature vectors (rotation-invariant clustering) we used the k-Means algorithm (MacQueen et al., 1967) with 50 clusters, which we empirically found to provide a good balance between clusters being small enough to contain similar neurons and the total number of clusters being relatively small.

GLM model. We also fitted a GLM model (McCullagh & Nelder, 2019) to every neuron in the recorded dataset to evaluate the non-linearity of neurons or clusters (as measured by the non-linearity index, see below). We used Poisson likelihood as the noise model and log link function to ensure the predicted neural activities are non-negative. We cross-validated the L_2 regularization coefficient for every neuron separately by considering 48 log-spaced values in $[0.1, 5]$.

Optimal stimuli. Classical experiments that we aim at replicating in silico measure changes in neural activity in response to a certain type of transformations of an input stimulus (e.g. an orientation tuning experiment might use Gabor stimuli in different orientations). Ideally the input stimuli should be optimized for every neuron separately which is infeasible in vivo, but can be achieved in silico using a CNN model. In this section we describe the stimuli we use in the in silico experiments and how we optimize them for individual neurons or clusters.

Per neuron optimal Gabors. For every neuron we compute a Gabor stimulus which maximizes the predicted activity of this neuron. We parametrize such stimuli in terms of a spatial location (r^x, r^y), size σ, spatial frequency ν, contrast a, orientation φ and phase τ. Specifically the value of the pixel in the i-th row and j-th column of an input image is defined using the following expression

$$\mathbf{x}[i,j] = \frac{a}{2}\exp\left(-\frac{1}{2}\frac{(i')^2+(j')^2}{\sigma^2/4}\right)\cos\left(2\pi\cdot i'\cdot\nu+\tau\right), \tag{1}$$

$$\text{where}\quad (i',j') = (i-r^x, j-r^y)\,[R(\varphi)]^T \tag{2}$$

with $R(\varphi)$ being a 2D rotation matrix by an angle φ.

For each neuron we iterate over a large set of Gabor stimuli parameterized according to (2) and record the parameter set corresponding to a stimulus with the highest CNN predicted activity. This process is illustrated in Fig. 2A.

Per cluster optimal Gabors. In addition to optimising Gabors for each neuron separately, we also find optimal Gabors for the entire clusters. The idea is to find a single Gabor stimulus that maximizes the average activity of neurons in the cluster and hence servres as a single image representations of a cluster computation. However, since we the clusters are explicitly constructed to contain neurons with different receptive field locations and preferred orientations, we constrain the stimuli to be identical for all neurons in the cluster apart from having neuron-specific spatial locations (i.e. receptive field centers) and orientations (see an illustration in Fig. 2A).

To find such an optimal Gabor for a specific cluster, we iterate over a large set of Gabors with locations and orientations set to a fixed value and all other parameters varying, generate stimuli for individual neurons by shifting and rotating the Gabor at the current iteration (Fig. 2A), and compute the average predicted activity of the neurons in the cluster. We call the Gabor stimulus (or rather the set of stimuli up to a spatial location and orientation) corresponding to the maximal average predicted cluster activity the cluster optimal Gabor.

Per neuron MEIs. An MEI is an input stimulus that activates the neuron the most, and as such serves as a useful visualization of the computation implemented by the neuron. It is important to keep in mind that an MEI is only a local maximum of a function mapping the input stimuli to the neural activity. While we aim at describing the entire function rather than only its maximum, MEIs nevertheless provide convenient and insightful summaries of computations performed by each of the neurons.

We compute MEI for every neuron separately by stating with a noise stimulus and iteratively optimising it to maximise the predicted activity of the neuron. Specifically, if the CNN prediction of activity of the n-th neuron when presented with a stimulus $\mathbf{x}$ is $f_n(\mathbf{x})$, we compute an MEI as a solution of the following optimisation problem:

$$\mathbf{x}_n^{\text{MEI}} = \arg\max_{\mathbf{x}}\left[f(\mathbf{x}) - \gamma\cdot\alpha(\mathbf{x})\right], \tag{3}$$

where $\alpha(\mathbf{x})$ is a regularisation function (e.g. $\alpha(\mathbf{x}) = ||\mathbf{x}||^2$) enforcing smoothness of the resulting MEI.

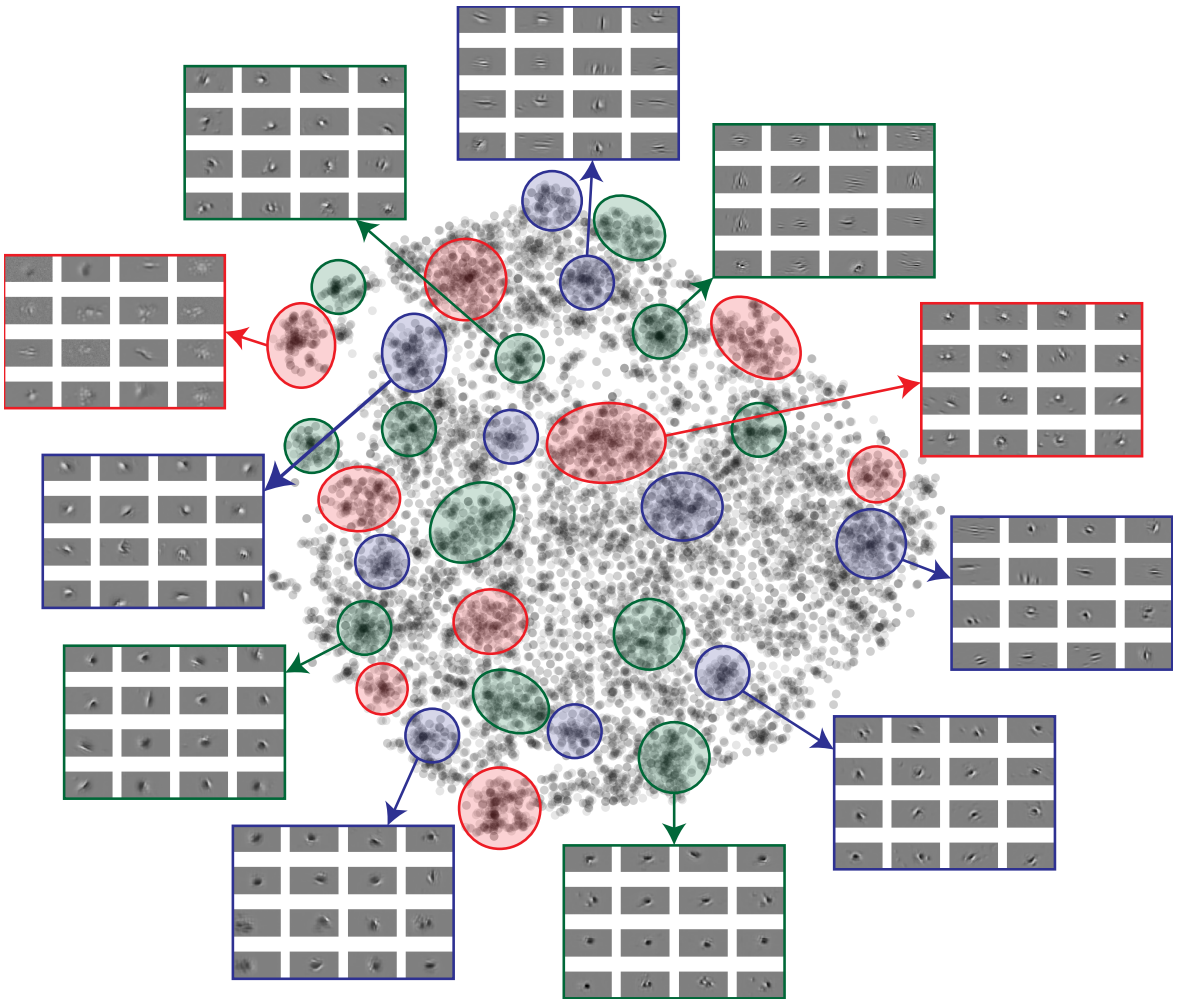


Fig. 6. Functional modes in the t-SNE space.

Animal	# of neurons in the scan	Sampled neurons	# of train images	# of test images
1	5043	1047	5998	1999
2	5984	2000	5973	1090
3	7312	2000	4926	985
4	5335	2000	4994	999
5	8367	2000	4818	964
6	6045	2000	4982	995
7	8316	2000	4991	997
	Total	13047	4692	1000

Table 1. Summary of the individual mouse scans comprising the dataset used for the cell types analysis.

Per cluster MEIs. Such MEIs are computed jointly for all neurons in the entire cluster to maximise the average predicted activity of all neurons in the cluster. Similarly to cluster optimal Gabors, these stimuli are constrained to be identical for all neurons in the cluster apart from having neuron-specific spatial locations (i.e. receptive field centers) and orientations (Fig. 2A). To implement these constraints in practice, we decompose cluster MEIs in a steerable basis and iteratively optimise the coefficients in this basis to maximize the predicted average activity of neurons in the cluster. Therefore we use a parametric model of cluster MEIs rather than a non-parametric representation of single-neuron MEIs, which reduces the spaces of possible stimuli (for single-neurons MEIs this space consists of all possible images, while in this case it consists only of the span of the steerable basis), but allows us to compute stimuli rotations as linear transformations of the coefficients.

We consider the following parametrization of the stimuli

$$\mathbf{x} = \beta_1 \cdot \boldsymbol{\psi}_1 + \ldots + \beta_k \cdot \boldsymbol{\psi}_k = \boldsymbol{\beta}^T \boldsymbol{\Psi}, \tag{4}$$

where $\boldsymbol{\Psi} = (\boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_k)^T$ is a vector of the first k basis

functions in some steerable basis (we use Hermite polynomials). Rotations of images in such a parametrization correspond to linear transformations of coefficients $\boldsymbol{\beta}$, specifically, a stimulus rotated by an angle φ can be written as

$$\text{rotate}(\mathbf{x}, \varphi) = (R(\varphi)\,\boldsymbol{\beta})^T\,\boldsymbol{\Psi} \tag{5}$$

for a corresponding rotation matrix $R(\varphi)$.

We denote a translation of an image $\mathbf{x}$ by r^x pixels along the first axis and by r^y pixels along the second axis as $\text{shift}(\mathbf{x}, r^x, r^y)$. Using center locations of optimal Gabors (r^x_n, r^y_n) as receptive fields centers for corresponding neurons, we compute the cluster MEI $\mathbf{x}^{\text{MEI}}_c$ for a cluster c containing m neurons with indices $c_1, \ldots, c_m$ as a solution of the following problem

$$\mathbf{x}^{\text{MEI}}_c = (\boldsymbol{\beta}^*)^T \boldsymbol{\Psi}, \qquad \text{where} \tag{6}$$

$$\boldsymbol{\beta}^* = \arg\max_{\boldsymbol{\beta}} \max_{\varphi_1, \ldots, \varphi_m} \left[\frac{1}{m} \sum_{i=1}^{m} A_{c_i} \right], \tag{7}$$

$$A_{c_i} = f_{c_i}(\text{shift}[\text{rotate}(\mathbf{x}, \varphi_i), r^1_{c_i}, r^2_{c_i}]) \tag{8}$$

Examples of cluster MEI are shown in Fig. 3.

Optimal differences of Gaussians (DoG). Another class of stimuli we consider are differences of Gaussians, which allow us to probe to what extent the receptive of a neuron has a center-surround structure. We parametrize such stimuli in terms of their spatial locations $\mathbf{r} = (r_1, r_2)$, sizes σ_{cen} and σ_{sur} of the center and surround Gaussians, as well as their relative contrasts a_{cen} and a_{sur} according to the following equation:

$$\mathbf{x} = a_{\text{cen}}\,(1 + a_{\text{sur}})\,g(\mathbf{r}_{\text{cen}}, \sigma_{\text{cen}}) - a_{\text{sur}}\,g(\mathbf{r}_{\text{sur}}, \sigma_{\text{sur}}), \tag{9}$$

$$\text{with} \quad g(\mathbf{r}, \sigma)[i, j] = \exp\left(-\frac{(r_1 - i)^2 + (r_2 - j)^2}{2\sigma^2} \right). \tag{10}$$

Similarly to optimal Gabors and MEIs, we find optimal DoG stimuli both for every neuron and cluster by iterating over a large set of parameter values and recording the parameter combination corresponding to the highest predicted activity of an individual neuron or average activity of all neurons in the cluster. In the case of per cluster stimuli, we constrain every neuron in the cluster to have exactly the same optimal stimulus apart from differences in spatial locations (Fig. 3A).

In silico experiments. In this section we describe specific in silico experiments that we perform using the optimal stimuli discussed in the previous section.

Orientation and phase tuning. Optimal Gabors enable us to compute standard orientation and phase tuning curves. We use both per neuron and per cluster optimal Gabors to obtain stimuli for this experiment by varying the orientation and phase parameters and keeping all other parameters fixed. Examples of such stimuli are shown in Fig. 2B.

For every neuron we compute numerical indices reflecting the tuning strength of the neuron. Specifically, we compute the F_1/F_0 summary statistics for phase tuning and F_2/F_0 statistics. These statistics are ratios of the absolute values of the first and second (reflecting periods of 2π for phase tuning and π for orientation tuning) Fourier coefficients to the mean value of the tuning curve, or alternatively the ratios of amplitudes of the est fitting sine curves to the means of the tuning curves as illustrated in Fig. 2B. If $\mathbf{r}(\mathbf{s}) = (r_1(s_1), \ldots, r_m(s_m))$ are responses of a neuron to Gabors with a parameter of interest (orientation of phase) taking values $\mathbf{s} = (s_1, \ldots, s_m)$, these indices are defined as

$$F_k/F_0 = \frac{\left| \sum_{j=1}^{m} r_j \exp(iks_j) \right|}{\sum_{j=1}^{m} r_j}. \tag{11}$$

Gabor size-contrast tuning curves. We construct stimuli for this experiment by using per neuron or per cluster optimal Gabors with all parameters fixed except for the size and contrast as illustrated in Fig. 2B. The resulting size-contrast tuning curves allow us to characterize neural computations in terms of surround or contrast suppression effects, which have been widely studied in the existing literature.

Denoting predicted activity of n-th neuron when presented a stimulus at the contrast level $c \in \{1, \ldots, C\}$ and size level $s \in \{1, \ldots, S\}$ as $g^n_{c,s}$, we compute the following suppression and contrast indices to numerically evaluate the tuning strength:

$$\text{Suppression index} = \frac{g^n_C - g^n_{C,S}}{g^n_C}, \quad g^n_C = \max_s g^n_{s,C}; \tag{12}$$

$$\text{Contrast index} = \frac{g^n_S - g^n_{C,S}}{g^n_S}, \quad g^n_S = \max_c g^n_{S,c}. \tag{13}$$

These two indices are highly correlated ($\rho = 0.95$, $p < 0.001$) which is why we use only the suppression index for the analysis. This correlation apparently stems from using Gabors as stimuli for this experiment. Indeed, increasing the size of a Gabor also increases the range of each pixel value, which is a similar effect to increasing the contrast.

Plaid stimuli. Another experiment we do with optimal Gabor stimuli is the one aimed at probing neurons or clusters for a potential effect of cross-orientation inhibition. To do some we construct plaid stimuli by superimposing two Gabors on each other, the optimal one and the one orthogonal to the optimal one, while varying the contrasts of both of these stimuli. Examples of such stimuli are shown in Fig. 2. The corresponding tuning curves allow us to see potential non-linear suppressing effect of increasing contrast of the orthogonal stimulus known as cross-orientation inhibition.

Denoting predicted activity of n-th neuron when presented a plaid stimulus with an optimal Gabor at the contrast level $p \in \{1,\dots,P\}$ and an orthogonal Gabor at the contrast level $o \in \{1,\dots,O\}$ as $g^n_{p,o}$, we compute the following numerical index to quantify the tuning strength:

$$\text{Plaids index} = \frac{g^n_O - g^n_{P,O}}{g^n_O}, \quad g^n_O = \max_p g^n_{p,C}. \tag{14}$$

Comparison of optimal Gabor, DoG, and MEI stimuli. Optimal MEIs capture a wide variety of patterns in the receptive field, while optimal Gabors and DoGs are explicitly designed to represent a particular pattern. Comparing the responses to these stimuli allows us to quantify to what extent the receptive field captured by the MEI can be modelled by oriented gratings (Gabors) or center surround patterns (DoG).

For every neuron we normalize all three (Gabor, DoG, MEI) optimal stimuli to have same energy (L_2 norm) at E different energy levels; such a normalization ensures that the stimuli have approximately the same contrast. Denoting predicted activity of n-th neuron when presented a Gabor, DoG or MEI at the energy level e as g^n_e, d^n_e, m^n_e respectively, we compute the following summary statistics for comparing the responses to these stimuli:

$$\text{Gabor vs DoG} = \frac{1}{E}\sum_{e=1}^{E}\frac{g^n_e}{d^n_e}, \tag{15}$$

$$\text{Gabor vs MEI} = \frac{1}{E}\sum_{e=1}^{E}\frac{g^n_e}{m^n_e}, \tag{16}$$

$$\text{DoG vs MEI} = \frac{1}{E}\sum_{e=1}^{E}\frac{d^n_e}{m^n_e}. \tag{17}$$

Non-linearity index. For every neuron we quantify the non-linearity computations implemented by this neuron by comparing the predictions of the GLM and the CNN models. Specifically, for the n-th neuron we denote the GLM predictive correlation on the test set as C^{GLM}_n, and the same quantity computed for the CNN model as C^{CNN}_n. Then we compute the non-linearity index as follows:

$$\text{Non-linearity index} = \frac{C^{\text{CNN}}_n}{C^{\text{GLM}}_n}. \tag{18}$$

In vivo verfification.

Experimental details. The experimental setting for the ImageNet scan in vivo verification was the same as in the main experiment (see above). The Gabor stimuli were presented in maximum contrast in 5 different sizes ($\sigma \in [8, 13.2, 21.8, 35.9, 59.3]$).

Selection of suboptimal stimuli. We find a small number of Gabor stimuli activating many neurons to a certain percentage of their optimal Gabor activities (Fig.5D-E) using the following procedure:

- We compute predicted activities for every neuron for a large selection of Gabors located in the center of the image (the same set of stimuli that we used for finding the optimal Gabor apart from the differences in spatial locations; see above),
- We greedily choose the Gabor that activated most of the neurons to a given percentage of their optimal Gabor activity until we selected the required number of stimuli (we use 20 for the experiment).

Data availability. All figures were generated from raw or processed data. The data generated and/or analyzed during the current study are available from the corresponding author upon request. No publicly available data was used in this study. All code and data will be available online upon the publication.

Code availability. Experiments and analyses were performed using custom software developed using the following tools: ScanImage 2018a (Pologruto et al., 2003), CaImAn v.1.0 (Giovannucci et al., 2019), DataJoint v.0.11.0 (Yatsenko et al., 2015, 2018), TensorFlow v.1.15.0 (Abadi et al., 2015), NumPy v.1.17.3 (Van Der Walt et al., 2011), SciPy v.1.5.4 (Virtanen et al., 2020), Docker v.19.03.12 (Merkel et al., 2014), Matplotlib v.3.1.1 (Hunter, 2007), seaborn v.0.11.1 (Waskom, 2021), pandas v.1.1.5 (McKinney et al., 2010) and Jupyter Notebook v 6.0.1 (Kluyver et al., 2016). The code will be publicly available upon the publication.

Animal research statement. All experimental procedures complied with guidelines approved by the Baylor College of Medicine Institutional Animal Care and Use Committee (IACUC).

Competing interests. A.S.T. holds equity ownership in Vathes LLC, which provides development and consulting for the framework (DataJoint) used to develop and operate the data analysis pipeline for this publication.

Bibliography

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., & Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.
URL https://www.tensorflow.org/

Giovannucci, A., Friedrich, J., Gunn, P., Kalfon, J., Brown, B. L., Koay, S. A., Taxidis, J., Najafi, F., Gauthier, J. L., Zhou, P., et al. (2019). Caiman an open source tool for scalable calcium imaging data analysis. *Elife*, *8*, e38173.

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing in science & engineering*, *9*(03), 90–95.

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B. E., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J. B., Grout, J., Corlay, S., et al. (2016). *Jupyter Notebooks – a publishing format for reproducible computational workflows.*, vol. 2016.

McKinney, W., et al. (2010). Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, vol. 445, (pp. 51–56). Austin, TX.

Merkel, D., et al. (2014). Docker: lightweight linux containers for consistent development and deployment. *Linux journal*, *2014*(239), 2.

Pologruto, T. A., Sabatini, B. L., & Svoboda, K. (2003). Scanimage: flexible software for operating laser scanning microscopes. *Biomedical engineering online*, *2*(1), 1–9.

Van Der Walt, S., Colbert, S. C., & Varoquaux, G. (2011). The numpy array: a structure for efficient numerical computation. *Computing in science & engineering*, *13*(2), 22–30.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., et al. (2020). Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, *17*(3), 261–272.

Waskom, M. L. (2021). Seaborn: statistical data visualization. *Journal of Open Source Software*, *6*(60), 3021.

Yatsenko, D., Reimer, J., Ecker, A. S., Walker, E. Y., Sinz, F., Berens, P., Hoenselaar, A., Cotton, R. J., Siapas, A. S., & Tolias, A. S. (2015). Datajoint: managing big scientific data using matlab or python. *BioRxiv*, (p. 031658).

Yatsenko, D., Walker, E. Y., & Tolias, A. S. (2018). Datajoint: a simpler relational data model. *arXiv preprint arXiv:1807.11104*.