Robust and Efficient
Deep Visual Learning

Robust and Efficient Deep Visual Learning

Dissertation

der Mathematisch-Naturwissenschaftlichen Fakultät

der Eberhard Karls Universität Tübingen

zur Erlangung des Grades eines

Doktors der Naturwissenschaften

(Dr. rer. nat.)

vorgelegt von

Sergey Prokudin
aus Moskau, Russland

Tübingen
2020

Tag der mündlichen Qualifikation:	02.12.2020
Dekan:	Prof. Dr. Wolfgang Rosenstiel
1. Berichterstatter:	Dr. Peter Vincent Gehler
2. Berichterstatter:	Prof. Dr. Zeynep Akata

To my family

Abstract

The past decade was marked by significant progress in the field of artificial intelligence and statistical learning. Efficient new algorithms, coupled with the availability of large datasets and the dramatic increase in computing power, led to solutions that match or exceed human performance in perception tasks such as image and speech recognition, 3D shape analysis and various types of generative modeling.

However, the most impressive of modern models come in the form of computationally expensive black boxes, with the majority of them lacking the ability to reason about the confidence of their predictions robustly. Being capable of quantifying model uncertainty and recognizing failure scenarios is crucial when it comes to incorporating them into complex decision-making pipelines, e.g. autonomous driving or medical image analysis systems. It is also important to maintain a low computational cost of these models - the model that can be deployed on a mobile phone or an average PC rather than a GPU cluster will have a much higher potential social impact.

In the present thesis, the aforementioned desired properties of robustness and efficiency of deep learning models are studied and developed in the three specific realms of computer vision. First, we investigate deep probabilistic models that allow uncertainty quantification, i.e. the models that "know what they do not know". Here, we propose a novel model for the task of angular regression that allows probabilistic object pose estimation from 2D images. We also showcase how the general deep density estimation paradigm can be adapted and utilized in two other real-world applications, ball trajectory prediction and brain imaging.

Next, we turn to the field of 3D shape analysis and rendering. We propose a method for efficient encoding of 3D point clouds, the type of data that is hard to handle with conventional learning algorithms due to its unordered nature. We show that simple neural networks that use the developed encoding as input can match the performance of state-of-the-art methods on various point cloud processing tasks while using orders of magnitude less floating point operations.

Finally, we explore the emerging field of neural rendering and develop the framework that connects classic deformable 3D body models with modern image-to-image translation neural networks. This combination allows efficient photorealistic human avatar rendering in a controlled manner, with the possibility to control the camera flexibly and to change the body pose and shape appearance.

The thesis concludes with the discussion of the presented methods, including current limitations and future research directions.

Zusammenfassung

Das vergangene Jahrzehnt war von bedeutenden Fortschritten auf dem Gebiet der künstlichen Intelligenz und des statistischen Lernens geprägt. Effiziente neue Algorithmen, gepaart mit der Verfügbarkeit großer Datensätze und einem drastischen Anstieg der Rechenleistung, erlaubten es, die menschliche Wahrnehmung im Bereich der Bild- und Spracherkennung, der 3D-Formanalyse und bei verschiedenen Arten der generativen Modellierung zu imitieren, diese in vielen Hinblicken sogar zu übertreffen.

Die beeindruckendsten modernen Modelle sind jedoch rechenintensive Blackboxen, denen meist die Fähigkeit fehlt, robust über die Zuverlässigkeit ihrer Vorhersagen zu urteilen. Modellunsicherheiten zu quantifizieren und Fehlerszenarien zu erkennen, ist von entscheidender Bedeutung, wenn es darum geht, sie in komplexe Entscheidungsprozesse, zu implementieren, z.B. in autonome Fahrsysteme oder medizinische Bildanalysesysteme. Ebenso wichtig ist es, die benötigte Rechenleistung dieser Modelle niedrig zu halten - ein Modell, welches auf einem Mobiltelefon oder einem durchschnittlichen PC und nicht nur einem GPU-Clusters eingesetzt werden kann, wird potentiell eine viel größere soziale Auswirkung haben.

In der vorliegenden Arbeit werden die oben genannten, erwünschten Eigenschaften der Robustheit und Effizienz in Modellen für das tiefe Lernen in drei verschiedenen Bereichen der Computer Vision untersucht und weiterentwickelt.

Zum einen untersuchen wir tiefe probabilistische Modelle, die eine Quantifizierung der Unsicherheit erlauben, d.h. solche Modelle, die ”wissen, was sie nicht wissen”. Hier demonstrieren wir, dass ein neuartiges, von uns entworfenes Modell für die Aufgabe der Winkelregression, eine probabilistische Abschätzung der Objektposition aus 2D-Bildern ermöglicht. Wir zeigen auch, wie das allgemeine Paradigma der Schätzung der tiefen Dichte angepasst und in zwei anderen realen Anwendungen, der Vorhersage der Flugbahn von Kugeln und der Bildgebung des Gehirns, verwendet werden kann.

Zum anderen untersuchen wir 3D-Formanalyse und 3D-Rendering. Wir haben eine effiziente Methode zur Kodierung von 3D-Punktewolken entwickelt, der Art von Daten, die aufgrund ihrer ungeordneten Natur mit herkömmlichen Lernalgorithmen schwer zu handhaben sind. Wir zeigen, dass einfache neuronale Netze, die die entwickelte Kodierung als Input verwenden, die Leistung modernster Methoden für verschiedene Aufgaben der Punktwolkenverarbeitung erreichen können, während sie mit um Zehnerpotenzen geringeren Fließkommaoperationen arbeiten.

Desweiteren kombinieren wir klassische verformbare 3D-Körpermodelle mit modernen neuronalen Netzwerken zur Bild-zu-Bild-Translation im jungen Gebiet des neuronalen Renderings.Diese Kombination ermöglicht äußerst effizient ein fotorealistisches,

menschliches Avatar-Rendering, mit der Möglichkeit, die Kamera flexibel zu steuern und die Körperhaltung und das Aussehen des Körpers zu verändern.

Die Dissertation schließt mit der Diskussion der vorgestellten Methoden, einschließlich der aktuellen Grenzen und zukünftigen Forschungsrichtungen.

Acknowledgments

I would like to thank the wonderful institutions I have been lucky to be part of and collaborate with: Max Planck Institute for Intelligent Systems, University of Tübingen, Amazon, as well as specific people there: Melanie Feldhofer, Prof. Dr. Zeynep Akata. This work was partially supported by Microsoft Research through its PhD Scholarship Programme.

This work would be simply impossible without the help and guidance of my supervisors and senior collaborators: Dr. Peter Vincent Gehler, Dr. Sebastian Nowozin, Dr. Javier Romero, Dr. Michael J. Black. I would also like to thank my co-authors for their work, discussions and interdisciplinary research ideas: Dr. Christoph Lassner, Dr. Sebastian Gomez, Prof. Dr. Moritz Zaiss.

During these years, I have learned a tremendous amount of things from my friends, from the core computer vision and machine learning knowledge to the art of bicycle maintenance: Fatma Güney, Laura Sevilla-Lara, Varun Jampani, Thomas Nestmeyer, Luigi Gresele, Sabrina Klotz, Eric Lacosse, Ilya Tolstikhin.

Finally, I would like to thank my whole family and especially my mother Firaya Prokudina for supporting me throughout life in all my best aspirations.

Contents

Chapter 1

Introduction

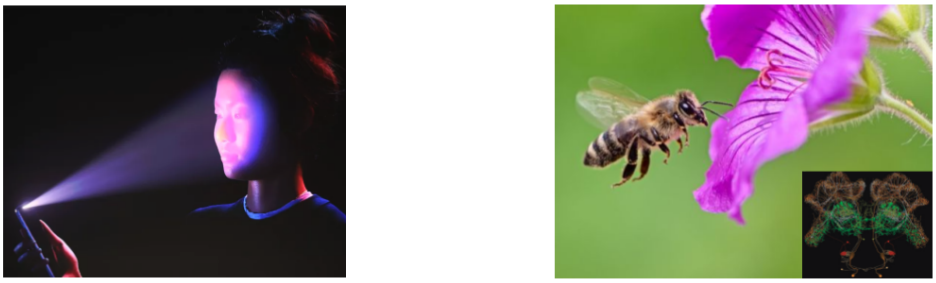
1.1 Motivation

The progress in computer vision was truly remarkable in the past decade. Powered by the increased availability of 2D and 3D data, computational resources and efficient deep learning algorithms, visual learning pipelines become a part of our daily life. This includes algorithms for face identification, image search, character recognition, computational photography, driving assistance systems and many more.

With the increased role of computer vision in critical applications, two considerations become of great importance. First, deployed models should be robust to novel input data, i.e. the one that significantly differs from those seen during the training phase. At the very least, the model should be able to detect these scenarios and *correctly quantify its uncertainty* about the produced results. If uncertainty is not well-calibrated, or—even worse—is not taken into account at all, then the consequences of decisions made by the system cannot be accurately assessed, resulting in poor decisions at best, and dangerous actions at worst. Examples of such systems include autonomous driving systems, medical imaging and robotics.

Another desired feature of any real-world system is *computational efficiency*. The best of deep learning systems regularly need days and weeks of training time, as well as powerful GPUs during test time inference. At the same time, there are multiple reasons why the efficiency of our developed algorithms should be a concern. First, a system that can be deployed on an average smartphone rather than a GPU cluster will have a much more social impact. One can think of an analogy with digital photography here: the ability to take a photo with a smartphone and upload it to the Internet made datasets like ImagenNet [64] possible, and consequently, marked up the whole new era of data-driven visual learning. Another rising concern is the carbon footprint of deep learning research. Recently, it has been shown that training some of the models for NLP tasks results in the amount of CO2 emissions that are comparable to a lifetime of five average cars [252].

Finally, efficiency and robustness of a visual learning system become essential when both inference and learning is made on an isolated compact computing device. One successful example of such a system is Apple Face ID [2] which allows user to unlock a smartphone via RGB-D image taken with front-face camera sensors. The system learns



(a) (b)

Figure 1.1: *Robust and efficient visual systems in technology and biology*: **(a)** Apple FaceID technology for image-based authorization running directly on a user device [2, 5]; **(b)** with only 10^6 neurons, *honeybees* are capable of performing large variety of tasks, including navigation, abstract concept learning, communication and uncertainty reasoning [178, 3].

a model of an owner's face from a series of calibration shots. During authorization, this model is compared to the one obtained from the authorization image. The whole process of user-dependent model adjustment and test-time inference is made directly on the device, with no data leaving the protected storage. The system is also continuously learning to adapt to novel face appearances like changing hairstyles, glasses, etc.

As with many other technological designs, including neural networks themselves, one can seek inspiration in biological systems. Here, a honeybee and similar species form a perfect example of a truly efficient perceiving system [178]. The brain of a bee consists of only $\approx 10^6$ neurons (0.001% that of a human brain). Yet, a single honeybee have a surprising portfolio of successfully executed perception and behavioral tasks, including pattern recognition [20], optical flow based navigation [71], communicating via symbolic language [269] and abstract reasoning [19]. Strikingly, they also learn to quantify the reliability of the obtained information about the environment and learn to pay less attention to noisy signals [214]. While we do not necessarily want to copy the biological design, this example showcases that a complex pattern analysis can be done within a relatively miniature system.

1.2 Organization and Contributions

In this thesis, we focus on three specific topics that contribute to the overall goal of creating truly robust and efficient visual learning algorithms.

First, we investigate the models that infer object pose from 2D images (Figure 1.2a). Modern deep learning systems successfully solve this task when the input image is of high quality. However, in challenging imaging conditions such as on low-resolution

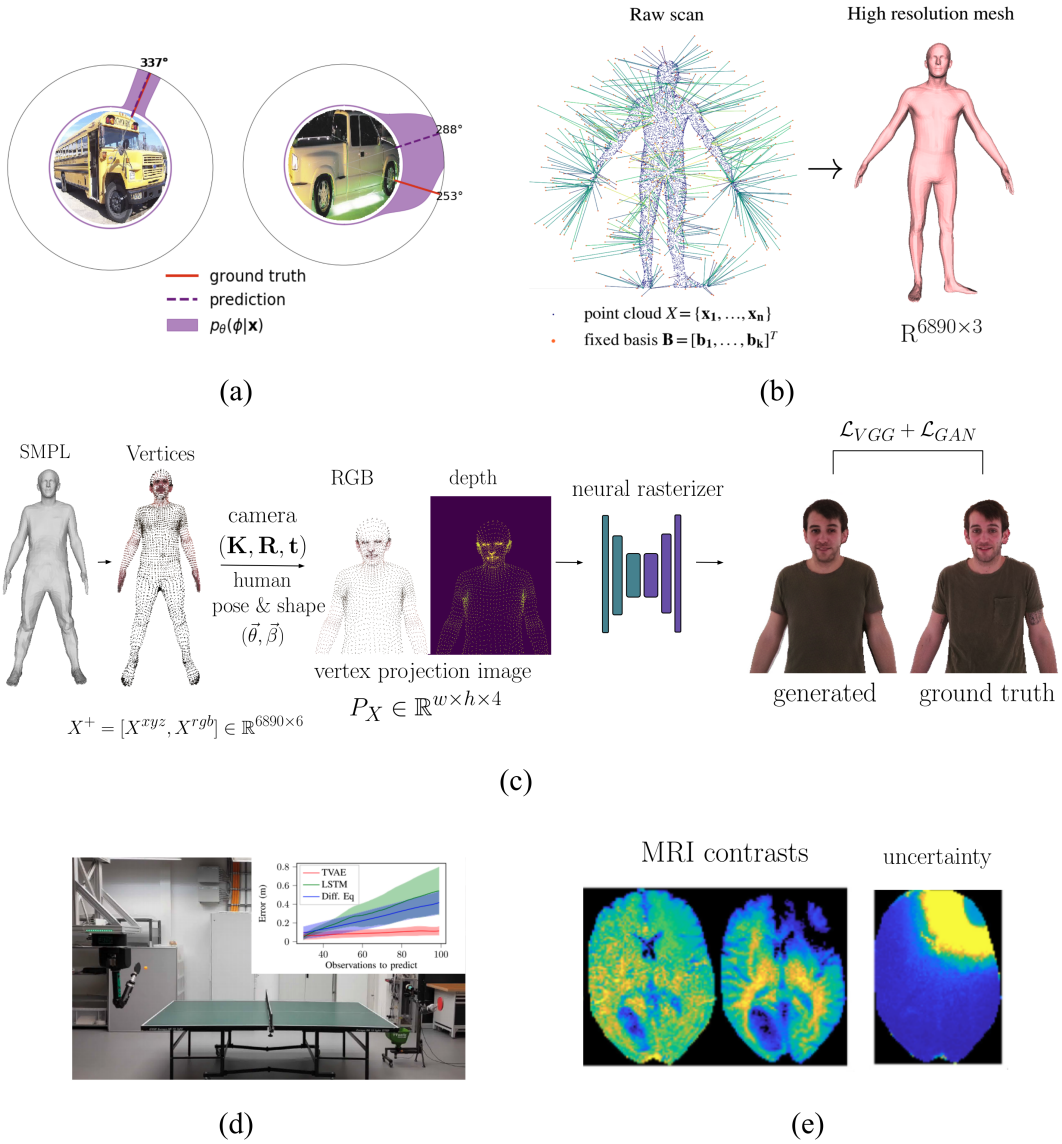


Figure 1.2: *Visual learning systems developed in the thesis:* **(a)** probabilistic object pose estimation framework [215]; **(b)** basis point set encoding for 3D shape analysis [217]; **(c)** neural human rendering pipeline (Chapter 7); *work done in collaboration*: **(d)** real-time ball trajectory prediction for robotic table tennis arm [100]; **(e)** robust MRI parameter determination and uncertainty quantification [93].

images or when the image is corrupted by imaging artifacts, current systems degrade considerably in accuracy. While a loss in performance is unavoidable, we would like our models to quantify their uncertainty in order to achieve robustness against images of varying quality. We, therefore, propose a novel probabilistic deep learning model for

the task of angular regression that combines the expressive power of deep learning with uncertainty quantification. Our model uses mixtures of von Mises distributions to predict a rich distribution over object pose angle. We show how to learn a deep mixture model using a finite and infinite number of mixture components. We demonstrate on several challenging pose estimation datasets that our model produces well-calibrated probability predictions and competitive or superior point estimates compared to the current state-of-the-art.

The deep probabilistic density estimation framework is, in fact, quite general and applicable beyond circular regression tasks. We consider two other applications of it, CEST contrast predictions in magnetic resonance imaging (Figure 1.2e) and probabilistic ball trajectory prediction in robotics (Figure 1.2d).

Next, we turn to the field of 3D shape analysis. With the increased availability of 3D scanning technology, 3D data in the form of point clouds is moving into the focus of computer vision as a rich representation of everyday scenes. However, point clouds are hard to handle for machine learning algorithms due to their unordered structure, and the deep neural networks working with point clouds directly often use a large number of parameters. In this thesis, we propose *basis point sets (BPS)* as a highly efficient and fully general way to process point clouds with machine learning algorithms (Figure 1.2b). The basis point set representation is a residual representation that can be computed efficiently and can be used with standard neural network architectures and other machine learning algorithms. Using the proposed representation in combination with simple neural networks allows us to match the performance of state-of-the-art deep frameworks on a shape classification task while using three orders of magnitude less floating-point operations. We also show how the proposed representation can be used for registering high-resolution meshes to noisy 3D scans. Here, we present the first method for single-pass high-resolution mesh registration, avoiding time-consuming per-scan optimization and allowing real-time execution.

Finally, we address the task of photorealistic human rendering (Figure 1.2c). Recent advances in deep generative models have led to an unprecedented level of realism for synthetically generated images of humans. However, one of the remaining fundamental limitations of these models is the ability to control the generative process flexibly, e.g. change the camera and human pose while retaining the subject identity. At the same time, deformable human body models like SMPL [165] and its successors provide full control over pose and shape but rely on classic computer graphics pipelines for rendering. Such rendering pipelines require explicit mesh rasterization that (a) does not have the potential to fix artifacts or lack of realism in the original 3D geometry and (b) until recently, were not fully incorporated into deep learning frameworks. We propose to bridge the gap between classic geometry-based rendering and the latest generative networks operating in pixel space by introducing a *neural rasterizer*, a trainable neural network module that directly "renders" a sparse set of 3D mesh vertices as photorealistic images, avoiding any hardwired logic in pixel colouring and occlusion reasoning. We train our model on a large corpus of human 3D models and corresponding real photos, and show the advantage

over conventional differentiable renderers both in terms of the level of photorealism and rendering efficiency.

In summary, we propose three novel techniques that lead to improved robustness and efficiency of deep visual learning algorithms. Introduced deep probabilistic framework for angular regression provides a scalable and robust alternative to purely deterministic regression methods. We show that it allows efficient uncertainty quantification, the property crucial for real-world applications. Developed basis point set encoding for point clouds can serve as an efficient descriptor in a variety of 3D shape analysis tasks, improving by orders of magnitude computational efficiency of state-of-the-art models. We show how it can be combined with virtually any machine learning algorithm to meet computational constraints for specific applications. With the neural human rendering pipeline, we investigate the efficient ways to combine classic and novel approaches in computer graphics in order to create flexible and photorealistic rendering engines.

1.3 List of Publications

The contributions in this thesis mainly comprise of work from the following publications:

- Prokudin, S., Gehler, P., and Nowozin, S. (2018). Deep directional statistics: pose estimation with uncertainty quantification. In *Proceedings of the European Conference on Computer Vision (ECCV) (pp. 534-551)* [215];

- Prokudin, S., Lassner, C., and Romero, J. (2019). Efficient learning on point clouds with basis point sets. In *Proceedings of the IEEE International Conference on Computer Vision (pp. 4332-4341)* [217];

- Prokudin, S., Black, M.J., and Romero, J. (2020). SMPLpix: Neural pixels from 3D Human Models. *Submitted to the European Conference on Computer Vision (ECCV)*;

- Gomez-Gonzalez, S., Prokudin, S., Schölkopf, B., and Peters, J. (2020). Real time trajectory prediction using deep conditional generative models. IEEE Robotics and Automation Letters, 5(2), 970-976 [100];

- Glang, F., Deshmane, A., Prokudin, S., Martin, F., Herz, K., Lindig, T., Bender, B., Scheffler, K. and Zaiss, M. (2019). DeepCEST 3T: Robust MRI parameter determination and uncertainty quantification with neural networks—application to CEST imaging of the human brain at 3T. Magnetic Resonance in Medicine [93].

The following research was conducted during the time of a PhD studentship which is not covered by this thesis:

- Prokudin, S., Kappler, D., Nowozin, S., and Gehler, P. (2017, September). Learning to filter object detections. In German Conference on Pattern Recognition (pp. 52-62). Springer, Cham. [216].

Part I

Uncertainty in Deep Visual Models

Chapter 2

Uncertainty Modeling

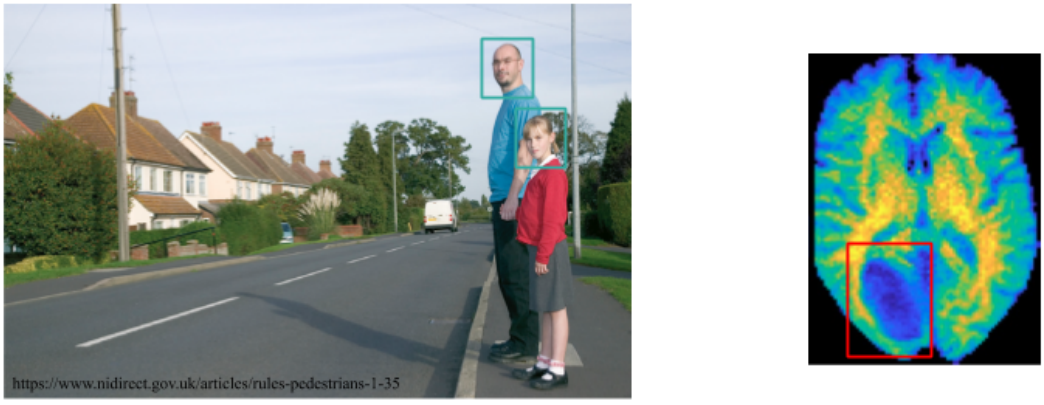
2.1 The Importance of Knowing What We Do Not Know

Self-awareness, the ability to recognize the limits of one's own understanding of the world, is a fundamental property of intelligent agents. Often referred to as *uncertainty monitoring* in behavioral studies [245], this feature is often associated with higher degree of consciousness in species, or *metacognition*. It has been now found in many species capable of non-trivial intelligent behaviour, including primates, dolphins and bees. The ability to recognize novel setups and quantify the confidence of observations and actions is indeed crucial for the agent in order to make optimal decisions, avoid risky actions and acquire additional information when possible.

Likewise, artificial intelligent systems we create should have the same property [16]. When dealing with noisy, incomplete inputs, learning system should be able to recognize failure scenarios, provide valid confidence intervals or defer prediction until more information is acquired. Figure 2.1 shows some of the real-world scenarios where correctly quantifying the degree of uncertainty might be important. Figure 2.1a shows potential use case of the developed probabilistic pose estimation framework [215] in the autonomous driving system. The system detects pedestrians on the roadside and analyzes their head pose and gaze direction in order to understand whether they are aware of approaching car. The system might continue driving or induce extreme braking, depending on the estimated gaze direction. Here, the final decision is crucially dependent on whether the system is confident about the obtained information or just guessing at random.

Another example is medical imaging and decease detection [154]. Figure 2.1b shows MRI CEST contrast image of a brain obtained with a method developed in [93]. This image can be used in order to define the boundaries of a detected tumor accurately; again, understanding the confidence of the provided information will be critical for planning potential surgery.

These are simply two examples that directly involve systems developed in this thesis. However, virtually any real-world decision-making process involves some form of probabilistic reasoning and uncertainty manipulation. Other applications include active learning, planning in robotics, out-of-distribution sample detection and adversarial ro-



(a) (b)

Figure 2.1: *Importance of uncertainty quantification.* **(a)** *Head pose estimation framework [215] in autonomous driving*: how sure we are that pedestrians are looking in the right direction and are aware of the approaching car? We need to know that in the case we want to make *decisions* based on our estimates (e.g stop the car or continue). **(b)** *MRI brain image of a tumor patient [93]*: we need to robustly detect the boundaries of a tumor in order to plan the surgery.

bustness.

2.2 Uncertainty in Machine Learning Models

General machine learning model. Throughout the course of this chapter, we will consider a machine learning system as a function $\mathbf{f}_\theta$ with parameters $\theta \in \mathbb{R}^p$ that receives an input vector $\mathbf{x} \in \mathbb{R}^n$ and maps it to an output vector $\mathbf{y} \in \mathbb{R}^k$:

$$\mathbf{f}_\theta(\mathbf{x}) : \mathbb{R}^n \times \mathbb{R}^p \rightarrow \mathbb{R}^k. \tag{2.1}$$

The learning algorithm then seeks to find the best set of parameters θ^* that minimizes some predefined loss function $\mathcal{L}$ on a given set of input-output pairs $\mathcal{D} = (\mathbf{X}, \mathbf{Y}) = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^M$:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{M} \mathcal{L}(\mathbf{f}_\theta(\mathbf{x}_i), \mathbf{y}_i) \tag{2.2}$$

The optimal value (2.2) is often found via some form of stochastic gradient decent optimization algorithm [230].

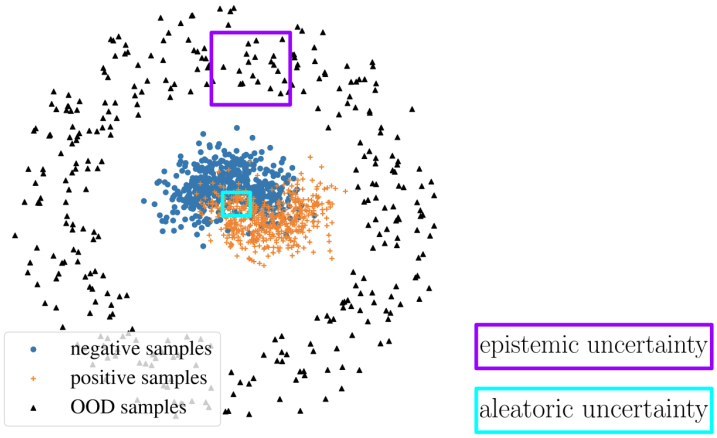


Figure 2.2: *Potential sources of uncertainty in synthetic two-dimensional, two-class classification data. Aleatoric uncertainty* stems from noisy observations, while *epistemic* occurs in the areas not covered by the training data.

One particular common example of a loss function for various regression tasks is a *mean squared error (MSE)*:

$$\mathcal{L}(\mathbf{f}_\theta(\mathbf{x}), \mathbf{y}) = ||\mathbf{y} - \mathbf{f}_\theta(\mathbf{x})||^2 \tag{2.3}$$

In the case of deep neural networks, the exact form of $\mathbf{f}_\theta$ is defined by a network architecture and parameters θ are the network weights. In the case of binary image classification, input vector $\mathbf{x} \in \mathbb{R}^{w \times h \times 3}$ represents an input image and output vector $\mathbf{y} \in \{0, 1\}$ represents predicted class label.

Once the system is trained and the optimal value θ^* is found, the function $\mathbf{f}_{\theta^*}(\mathbf{x})$ will be used to make predictions over the unseen value $\mathbf{x}_{new}$. At this stage, several potential problems might arise that we want to be able to detect and diagnose.

Noisy observations. Consider a simple example of classifying synthetic two-dimensional data visualized in Figure 2.2. First, we want to address the situation when new samples come from the region where two classes significantly overlap (cyan box). In this case, the system should signal that the noise and ambiguity in obtained input observations will not allow it to assign the class label confidently. This is the area of *aleatoric uncertainty* which can be resolved in the system by improving the measurement process. In the case of image processing, this can correspond to improving the quality of input image.

Out-of-distribution samples. While the *i.i.d* assumption is a cornerstone of machine learning algorithms, in practice we might draw $\mathbf{x}_{new}$ from the area of input space not seen during training (solid box in Figure 2.2). In this case, we talk about *out-of-distribution (OOD)* samples. Since the model haven't seen training examples in that area, its pre-

diction will likely be less reliable. The associated uncertainty of the model in this case represents one example of *epistemic uncertainty*. Epistemic uncertainty arises from the limited data and incorrect modeling assumptions. This type of uncertainty can be resolved by acquiring more data or refusing to make predictions in the areas of model's ignorance.

From point estimates to probability densities. In both situations mentioned above, we are fundamentally interested in the confidence of obtained predictions. The question of confidence can be formulated in the following way: how likely is the obtained value $\mathbf{f}_{\theta^*}(\mathbf{x}_{new})$ compared to any other possible output? The natural way to answer to this question is to utilize the language of *probabilities* [91]. Instead of modeling a single point estimate $\mathbf{f}_\theta(\mathbf{x}_{new})$, we aim to model the probability density function over all possible values $\mathbf{y}$:

$$p_\theta(\mathbf{y}|\mathbf{x}) : \mathbb{R}^n \times \mathbb{R}^p \times \mathbb{R}^k \to \mathbb{R}. \tag{2.4}$$

There are multiple ways to model $p_\theta(\mathbf{y}|\mathbf{x})$ via deep learning systems. Each of them emphasizes and reflects different types of uncertainties (i.e. aleatoric or epistemic). We will now describe the most popular approaches, discussing their theoretical foundations, empirical performance and potential limitations.

2.3 Approaches to Uncertainty Quantification

2.3.1 Maximum Likelihood Estimation

One of the classic and direct methods of finding the optimal $p_\theta(\mathbf{y}|\mathbf{x})$ is *maximum likelihood estimation principle (MLE)* [78] that aims to directly maximize the quantity (2.4) for observed data.

In the context of machine learning models discussed above, this results in the following modelling. First, we assume some parametric form of a density function (2.4), with the parameters of distribution being the outputs of our learning pipeline. For example, in the case of a neural network regression task, we can assume Gaussian distribution whose mean and variance are produced by the last layer of a network.

Next, we will fit the parameters of our model in order to maximize the likelihood of observed ground truth data samples, or, equivalently, minimize its *negative log-likelihood*:

$$\mathcal{L}_{NLL}(\theta, \mathbf{X}, \mathbf{Y}) = -\sum_{i=1}^{m} \log p_\theta(\mathbf{y}_i|\mathbf{x}_i), \tag{2.5}$$

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{m} \big[-\log p_\theta(\mathbf{y}_i|\mathbf{x}_i) \big] \tag{2.6}$$

MLE in the case of continuous regression. As was mentioned above, in the case of a regression task we can model our density function as a normal distribution:

$$p_\theta(\mathbf{y}|\mathbf{x}) = \frac{1}{\sqrt{2\pi\sigma_\theta^2(\mathbf{x})}} * \exp\Big[-\frac{||\mathbf{y} - \mu_\theta(\mathbf{x})||^2}{2\sigma_\theta^2(\mathbf{x})} \Big], \tag{2.7}$$

where $\mu_\theta, \sigma_\theta^2$ are the outputs of the network. We will use this form of a density modeling later in the thesis when we consider applications of deep probabilistic models in MRI imaging in Section 4.1.

We can gain some intuition about the induced uncertainty model by analyzing the corresponding negative log-likelihood objective of the Gaussian density. Placing the Equation (2.7) inside the logarithm and opening the brackets will result in:

$$-\log p_\theta(\mathbf{y}|\mathbf{x}) = \frac{1}{2\sigma_\theta^2(\mathbf{x})} ||\mathbf{y} - \mu_\theta(\mathbf{x})||^2 + \frac{1}{2} \log 2\pi\sigma_\theta^2(\mathbf{x}). \tag{2.8}$$

The obtained final objective consists of two competing terms, both of which are dependent on the predicted level of noise σ_θ^2: (a) mean squared error term scaled by the variance; (b) entropy of the predicted Gaussian distribution. This allows the system to properly model aleatoric uncertainty in the data: increasing predicted variance in the areas of noisy observations will result in a lower penalty for the potential deviation from the ground truth at the cost of increased entropy.

Note that we can also interpret vanilla mean squared error loss (2.3) as a case of log-likelihood model where we assume constant noise level for all observations, commonly referred to as *homoscedastic regression* and opposed to *heteroscedastic models* with the noise dependent on input [193, 140].

MLE in the case of binary classification. In the case of binary classification, a convenient and popular modeling distribution of choice is *Bernoulli*:

$$p_\theta(\mathbf{y}|\mathbf{x}) = \begin{cases} \mathbf{f}_\theta(\mathbf{x}), & \mathbf{y} = 1; \\ 1 - \mathbf{f}_\theta(\mathbf{x}), & \mathbf{y} = 0. \end{cases} \tag{2.9}$$

Again, here $\mathbf{f}_\theta(\mathbf{x}) \in [0, 1]$ is the output of the network. Negative log-likelihood objec-

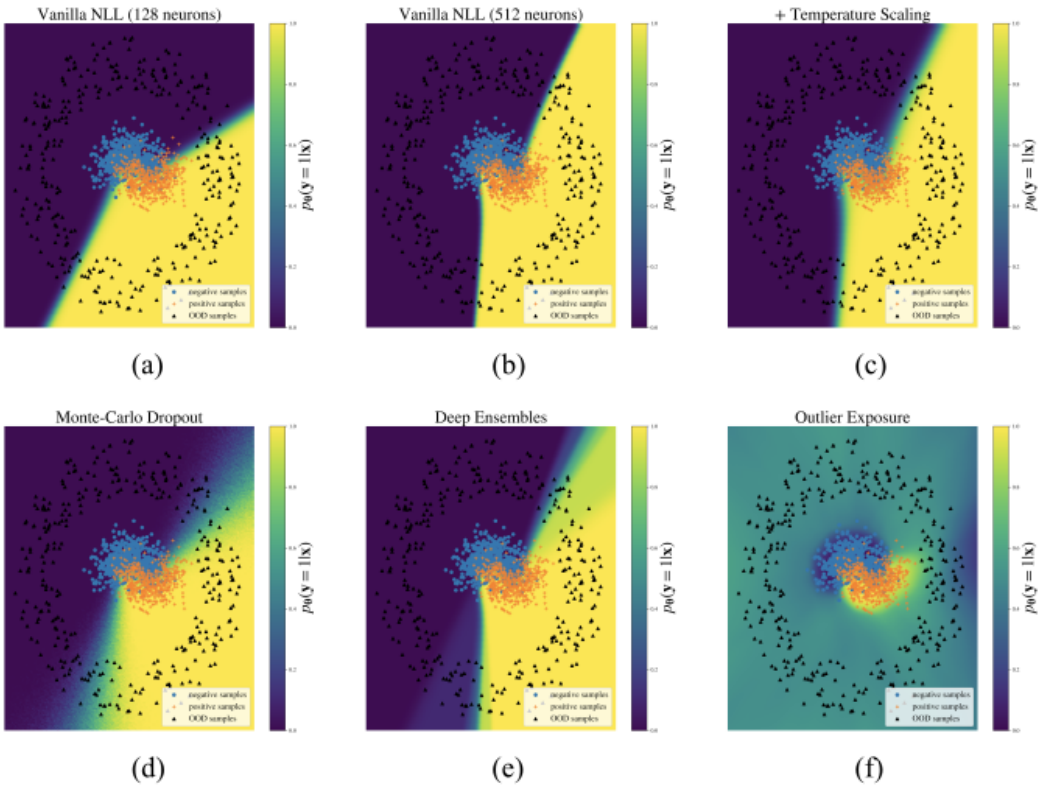


Figure 2.3: *Probability landscapes induced by various deep uncertainty models on the synthetic classification task.* **(a)** Training a multi-layer perceptron with standard log-likelihood objective; **(b)** same model with increased capacity; **(c)** temperature scaling [110]; **(d)** Monte Carlo Dropout as Bayesian inference approximation [86]; **(e)** Deep Ensembles [150] with 10 networks in ensemble; **(f)** Outlier Exposure technique [118, 117].

tive will then take the form:

$$-\log p_\theta(\mathbf{y}|\mathbf{x}) = -\Big[\mathbf{y} * \log \mathbf{f}_\theta(\mathbf{x}) + (1-\mathbf{y}) * \log\left(1-\mathbf{f}_\theta(\mathbf{x})\right)\Big]. \tag{2.10}$$

Loss function (2.10) is also often referred to as *binary cross-entropy* in the literature.

Toy example. To provide some empirical intuition and compare various methods of uncertainty modeling, we will consider a synthetic two-dimensional classification dataset visualized in Figure 2.2. Blue and orange dots denote two classes of interest, while black triangles will serve as our test-time out-of-distribution samples. We will use a simple fully connected [102] neural network with three layers as our base deep model, with ReLU activations [94] between intermediate layers and a sigmoid activation for

modelling Bernoulli distribution (2.9):

$$p_\theta(\mathbf{y}=1|\mathbf{x}) = \frac{1}{1+\exp\left[-\mathbf{g}_\theta(\mathbf{x})\right]}, \tag{2.11}$$

where $\mathbf{g}_\theta(\mathbf{x})$ is the output of the last layer of the network.

After the network is trained, we can visualize the obtained class probabilities at each point of our two-dimensional space. We train two versions of the model, with 128 and 512 neurons in the hidden layers respectively (Figures 2.3a,b).

We can can see that both models correctly assign lower confidence in the noisy boundary area where two classes significantly overlap. However, the smoothness of the boundary is significantly influenced by the model capacity, and the higher capacity model tends to generate sharper boundaries that will also result in higher number of overconfident yet incorrect predictions (see also Table 2.1 for numerical comparison of test-time likelihoods).

At the same time, both networks fail shortly in the areas where no training data was observed, simply extrapolating high confidence predictions based on the obtained decision boundaries.

2.3.2 Bayesian Neural Networks

Another way to model the density function (2.4) is via utilizing full Bayesian framework in the context of deep learning [169, 187, 65].

In the case of *Bayesian neural networks*, instead of aiming to find a single best point estimate (2.6) for the network parameters, we introduce a distribution over *all possible values of* θ. Regularly this process is initialized with a Gaussian prior over weights $\theta \sim \mathcal{N}(0, \sigma^2 I)$ that is then updated based on the training observations $\mathcal{D} = (\mathbf{X}, \mathbf{Y}) = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^m$ to obtain *posterior distribution*:

$$p(\theta|\mathbf{X}, \mathbf{Y}) = \frac{p(\mathbf{Y}|\mathbf{X}, \theta)p(\theta)}{p(\mathbf{Y}|\mathbf{X})} \tag{2.12}$$

The numerator consists of a prior over weights $p(\theta)$ and a likelihood function of training data:

$$p(\mathbf{Y}|\mathbf{X}, \theta) = \prod_{i=1}^{m} p_\theta(\mathbf{y}_i|\mathbf{x}_i), \tag{2.13}$$

where the density $p_\theta(\mathbf{y}_i|\mathbf{x}_i)$ is modeled by a neural network (e.g. (2.8) or (2.9)).

Denominator of (2.12) is computed via integrating over all possible values of θ, the process known as *marginalization*:

$$p(\mathbf{Y}|\mathbf{X}) = \int p(\mathbf{Y}|\mathbf{X},\theta)p(\theta)\mathrm{d}\theta. \tag{2.14}$$

Finally, given a novel observation $\mathbf{x}_{new}$, we can estimate the likelihood of any given possible output:

$$p(\mathbf{y}|\mathbf{x}_{new},\mathbf{X},\mathbf{Y}) = \int p_\theta(\mathbf{y}|\mathbf{x}_{new})p(\theta|\mathbf{X},\mathbf{Y})\mathrm{d}\theta. \tag{2.15}$$

While theoretically appealing and elegant in formulation, inference in Bayesian neural networks face major computational challenges when obtaining the value of a marginal distribution (2.14).

One way to deal with this problem is to use some form of Monte Carlo [87] approximation of the marginal integral (2.14). Another common solution to this problem is *variational inference* [80] that seeks to approximate the posterior weight distribution $p(\theta|\mathbf{X},\mathbf{Y})$ with some simpler distribution $q_\gamma(\theta)$. We refer to [183, 31] for a comprehensive study of Bayesian inference techniques.

In the context of modern deep learning, one method commonly used in practice is based on dropout technique [251]. It was shown in [86, 85] that training neural networks with dropout can be seen as a form of variational Bayesian approximation. Training the network with pre-defined dropout probabilities for neurons corresponds to fitting a specific approximate posterior distribution $q_\gamma(\theta)$. We can then obtain samples from this distribution by simply running the trained network with dropout applied during the test time and get corresponding induced uncertainty by running Monte Carlo integration over the model samples.

In the case of classification, this will correspond to the following final model output:

$$p(\mathbf{y}|\mathbf{x}_{new},\mathbf{X},\mathbf{Y}) = \frac{1}{S}\sum_{i=1}^{S} p_{\theta_\mathbf{i}}(\mathbf{y}|\mathbf{x}_{new}), \tag{2.16}$$

also referred to as *Monte Carlo (MC) Dropout* in the original paper [86]. However, the quality of the final approximation and the induced uncertainties will largely depend on the complexity of a model, number of samples S and hyper-parameters such as prior distribution used for weight initialization, dropout rates, etc.

We showcase the example of probabilities induced by MC Dropout method on our synthetic classification dataset in Figure 2.3d. It can be seen that using model averaging scheme defined by (2.16) results in the increased level of uncertainty in the borderline regions, even far away from the training data. However, the model still outputs high

confidence predictions in other regions where no actual training data was observed (upper left and lower right corners).

2.3.3 Deep Model Ensembling

Bayesian inference in the form of Monte-Carlo Dropout (2.16) investigated in the previous section can, in fact, be interpreted as a form of *model ensembling* [66], with elements of the ensemble sharing structure and weights. The idea of using ensembling methods for deep uncertainty quantification was also investigated in [150].

In particular, it was shown in [150] that reasonable and practically useful uncertainty estimates can be obtained by a simple averaging of the likelihoods produced by T independently trained networks:

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{T}\sum_{t=1}^{T} p_{\theta_t}(\mathbf{y}|\mathbf{x}), \tag{2.17}$$

We visualize the resulting probability landscape on our toy dataset obtained via training and averaging predicted densities of $T = 10$ baseline networks in Figure 2.3e. The overall effect is similar to Monte-Carlo Dropout technique. However, it was shown in [150] and later in [248] that, in general, likelihood estimates obtained with (2.17) tend to outperform uncertainties produced via MC Dropout technique both in terms of test set likelihoods and OOD detection capabilities.

Deep ensembles can be further improved by bringing down computational costs via ensemble *model distillation* [119], as well as combined with other methods like *adversarial training* [150] for further improvement of predictive likelihoods.

2.3.4 Post-Processing Calibration

In principle, likelihood-based training discussed in Section 2.3.1 aims to directly optimize probability scores returned by the model, and the objective (2.5) is minimized if and only if the model density $p_\theta(\mathbf{y}|\mathbf{x})$ recovers ground truth distribution $p(\mathbf{y}|\mathbf{x})$.

However, limited amount of training data and optimization challenges can result in biased probability scores produced by the network during test time. In this case, we can utilize some available validation data in order to re-adjust, or *calibrate*, our final probabilities.

In the context of deep learning, one popular and effective way to perform calibration is *Platt scaling* [210, 110]. The idea behind the method is to learn additional sigmoidal model based on the validation data $\mathcal{D}^{val} = (\mathbf{X}^{val}, \mathbf{Y}^{val}) = \{(\mathbf{x}_i^{val}, \mathbf{y}_i^{val})\}_{i=1}^{V}$:

$$p_{\theta^+}(\mathbf{y} = 1|\mathbf{x}) = \frac{1}{1 + \exp[\alpha f_{\theta^*}(\mathbf{x}) + \beta]}, \tag{2.18}$$

where $\theta^+ = [\alpha, \beta, \theta^*]$ and $f_{\theta^*}(\mathbf{x})$ is the output probability of our trained neural network model.

We then optimize the negative log-likelihood of validation data in order to obtain the optimal parameters of the scaler:

$$[\alpha^*, \beta^*] = \underset{\alpha,\beta}{\operatorname{argmin}} \sum_{i=1}^{V} \big[-\log p_{\theta^+}(\mathbf{y}_i^{val} | \mathbf{x}_i^{val}) \big]. \tag{2.19}$$

Note that the optimal weights θ^* of the original neural network are kept fixed.

A simpler version with no bias term β^* is known as *temperature scaling* and was shown to be an effective technique for calibrating probabilistic predictions [110].

We visualize the results of a calibrated 512-neuron model in Figure 2.3c. It can be seen that the calibrated model have softer decision boundaries, which also means higher uncertainty in the noisy areas.

2.3.5 Data Augmentation

Finally, we can also enforce certain behaviour of our predicted conditional densities on the unobserved data via data augmentation techniques. In this section, we will briefly review two general directions.

Boundary smoothing via adversarial training. The first set of methods aims to model the behaviour near the observed training examples. In particular, these methods set to enforce predictive likelihoods to behave smoothly in some ε-neighborhood of the training samples [150, 180]. Enforcing such behaviour along all the input dimensions might, however, be computationally prohibitive. Instead, [150] propose to utilize *adversarial training* [104] techniques in order to find the directions in input space (and corresponding synthetic samples) where the likelihood will decrease the most.

Formally, given a training pair $(\mathbf{x}^{tr}, \mathbf{y}^{tr}) \in \mathcal{D}^{tr}$, we will generate a synthetic adversarial example via *fast gradient sign* method [104]:

$$\mathbf{x}' = \mathbf{x}^{tr} + \varepsilon \operatorname{sign}\Big(\nabla_{\mathbf{x}^{tr}} [-\log p_\theta(\mathbf{y}^{tr} | \mathbf{x}^{tr})] \Big), \tag{2.20}$$

The pair $(\mathbf{x}', \mathbf{y}^{tr})$ is then added to the training set. More details, as well as experimental evaluation of this approach, can be found in [150].

Entropy maximization of the OOD samples. We can also force the model to output uniform predictions in order to reflect its ignorance on the samples outside its training domain. For the classification task, a generic procedure of *outlier exposure* was proposed in [118]. It implies the construction of the additional dataset $\mathcal{D}_{ood}^{tr} = \{\mathbf{x}_r\}_{r=1}^{R}$ of samples that lie far away from the training data, and enforcing the following *maximum entropy* auxiliary loss on the density:

$$\mathcal{L}_{OOD} = -\lambda \sum_{r=1}^{R} H\big(p_\theta(\mathbf{y}|\mathbf{x}_r)\big) = \lambda \sum_{r=1}^{R} \sum_{k=1}^{K} p_\theta(\mathbf{y}=k|\mathbf{x}_r) \log\big[p_\theta(\mathbf{y}=k|\mathbf{x}_r)\big]. \quad (2.21)$$

This loss is then added to the maximum likelihood loss (2.5). Intuitively, the loss (2.21) forces the network to output uniformly distributed probabilities on the OOD samples.

The choice of $\mathcal{D}^{tr}_{OOD}$ is the subject to a specific method and dataset. [118] proposed to use available real-world datasets that would differ significantly from the training data (e.g. using SVHN [189] images as $\mathcal{D}^{tr}_{OOD}$ while training on CIFAR [148]), while [117] advocated for a more agnostic approach via sampling a uniform distribution of the input domain $[0,1]^n$.

Importantly, it has been shown in both [118] and [117] that the network trained with the addtional loss (2.21) on $\mathcal{D}^{tr}_{ood}$ generalizes to other unseen OOD samples $\mathcal{D}^{test}_{OOD}$.

For our toy dataset, we gather outliers $\mathcal{D}^{tr}_{ood}$ by sampling the interval $[-4,4]^2$ uniformly. We also set $\lambda = 1$ in Equation (2.21). The results of the training with the additional max-entropy term are visualized in Figure 2.3f. We can see that this training mode achieves something we would intuitively expect from the model: high confidence at the centers of clusters, smooth boundaries and high uncertainty in the rest of domain.

2.4 Evaluation of Uncertainty Estimates

Once the model is trained, we will want to evaluate the quality of our probabilistic model $p_\theta(\mathbf{y}|\mathbf{x})$. Intuitively, we want the evaluation measure to encourage the network to uncover the true underlying distribution of data. This intuition us captured and formalized in the theory of *proper scoring rules* [95].

Proper scoring rules. As before, let $p_\theta = p_\theta(\mathbf{y}|\mathbf{x}) \in \mathcal{P}$ be our obtained probabilistic estimate, where $\mathcal{P}$ is a set of all probabilistic measures. A *scoring rule* is then a function that takes potential outcome $y \in \mathcal{Y}$ together with its estimated likelihood $p_\theta(\mathbf{y}|\mathbf{x})$ and assigns a real value to the tuple:

$$S(p_\theta, \mathbf{y}) : \mathcal{P} \times \mathcal{Y} \to \mathbb{R}. \quad (2.22)$$

Now, we can consider a true distribution $q = q(\mathbf{y}|\mathbf{x})$ and compute the expected score of the likelihood model under q:

$$S(p_\theta, q) = \mathbb{E}_{y\sim q} S(p_\theta, \mathbf{y}). \quad (2.23)$$

The scoring rule (2.22) is then called *proper*, if

$$S(p_\theta, q) \leq S(q, q), \quad \forall p_\theta, q \in \mathcal{P}. \tag{2.24}$$

The score is *strictly proper* if equality in (2.24) obtained if and only if $p_\theta = q$.

Aside from a general intuition of favoring the models that uncover true distribution of the data, there is also a strong theoretical justification and practical evidence that support the usage of proper scoring rules when considering uncertainty estimation. We refer to [95] for more discussion on the topic.

It turns out that the log-likelihood considered in Section 2.3.1 is a proper scoring rule [95]. Hence, in order to consistently assess the quality of our probabilistic estimates, we can measure the negative log-likelihood of the test data $\mathcal{D}^{test} = (\mathbf{X}^{test}, \mathbf{Y}^{test}) = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{Q}$:

$$NLL(p_\theta, \mathcal{D}^{test}) = -\sum_{i=1}^{Q} \log p_\theta(\mathbf{y}_i | \mathbf{x}_i). \tag{2.25}$$

However, one potential drawback of a log-likelihood score as a final evaluation protocol for certain applications is its tendency to highly penalize confident, but incorrect predictions [221, 261].

Brier score [39] is another proper scoring rule. For binary classification, it is defined as a mean squared error between the class label and positive class probability:

$$BS(p_\theta, \mathcal{D}^{test}) = \sum_{i=1}^{Q} \big(\mathbf{y}_i - p_\theta(\mathbf{y}_i = 1 | \mathbf{x}_i)\big)^2. \tag{2.26}$$

Calibration curves. We can also consider a frequentist approach [116] and think of the probabilistic model as being *well-calibrated* if the produced probability estimates reflect the actual statistics on the test data. In the case of binary classification, this would mean that among all the samples that obtained probability score 0.9 of belonging to positive class, 90% of the samples do actually have the positive label. Formally,

$$P(\mathbf{y} = 1 | p_\theta(\mathbf{y} = 1 | \mathbf{x}) = p) = p, \forall p \in [0, 1]. \tag{2.27}$$

However, we cannot compute quantity (2.27) using finite number of samples, and in practice approximations are used [192, 110].

One such approximation is a *reliability curve* (or a *reliability histogram*), an example of which is shown in Figure 2.4. The basic idea behind reliability curves is to split the interval $[0, 1]$ into L bins with the corresponding probabilities $[p_1, \ldots, p_L]$. Additionally, we set $p_0 = 0$ and $p_L = 1$. Given some test data $\mathcal{D}^{test} = (\mathbf{X}^{test}, \mathbf{Y}^{test}) = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{Q}$, we

can then compare average confidence of a model inside each bin B_l,

$$conf(B_l) = \frac{1}{|B_l|} \sum_{i=1}^{Q} I[p_{l-1} <= p_\theta(\mathbf{y} = 1|\mathbf{x}_i) < p_l] * p_\theta(\mathbf{y} = 1|\mathbf{x}_i), \tag{2.28}$$

with the actual accuracy inside this bin,

$$acc(B_l) = \frac{1}{|B_l|} \sum_{i=1}^{Q} I[p_{l-1} <= p_\theta(\mathbf{y} = 1|\mathbf{x}_i) < p_l] * I[\mathbf{y}_i = 1], \tag{2.29}$$

and compute the *calibration score* of our model p_θ:

$$s_l(p_\theta, \mathcal{D}^{test}) = |conf(B_l) - acc(B_l)|. \tag{2.30}$$

To compute the overall calibration score of a model, we can take a weighted average over bins:

$$ECE(p_\theta, \mathcal{D}^{test}) = \sum_{l=1}^{L} \frac{|B_l|}{Q} s_l(p_\theta, \mathcal{D}^{test}). \tag{2.31}$$

Expected calibration error (ECE) (2.31), however, is not a proper scoring rule. As was noticed in [248], simply setting $p_\theta(\mathbf{y}|\mathbf{x})$ to the marginal distribution $p(\mathbf{y})$ computed over training data will yield perfectly calibrated yet completely uninformative predictions. Nevertheless, ECE is often used together with reliability curve visualizations as an easy and intuitive way to diagnose probabilistic behaviour of the model.

Figure 2.4 shows reliability curves for the vanilla likelihood and temperature scaled versions of the perceptron model fitted on our toy data. We can see that, on average, the original model tends to overestimate its confidence (upper left plot) and is purely calibrated in the middle-confidence region. Fitting a single temperature parameter discussed in Section 2.3.4 significantly reduces the margin between network outputs and the perfect calibration curve (red diagonal line).

OOD entropy. There is no single widely accepted metric for assessing the quality of model behaviour on out-of-distribution samples. However, similar to the objective (2.21), we can measure the level of entropy on the test OOD samples:

$$H_{OOD} = \sum_{r=1}^{R} H\big(p_\theta(\mathbf{y}|\mathbf{x}_r)\big). \tag{2.32}$$

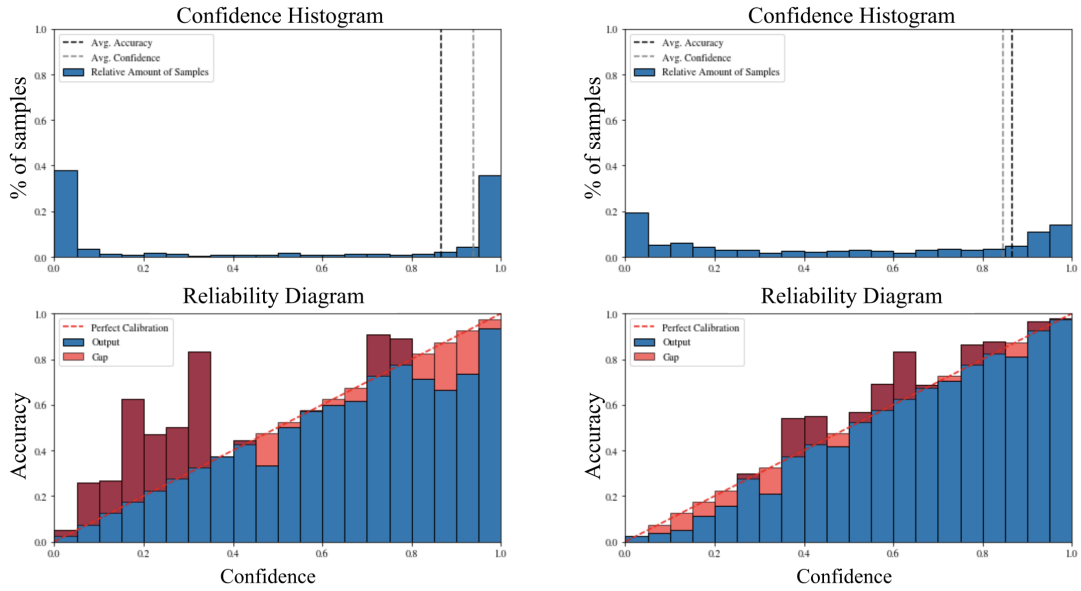


Figure 2.4: *Confidence histograms and reliability curves.* **(a)** Before and **(b)** after *temperature scaling* [110] applied for the toy moon dataset and multi-layer perceptron model. See Section 2.4 for discussion.

Higher values of H_{OOD} will indicate better model behaviour since we expect it to be less confident on the unknown domain.

Table 2.1 accumulates the results of all the discussed metrics and models. Even on this toy data, we can see that there is no uncertainty model that outperforms the others on all the metrics. Monte Carlo Dropout might be considered a method of choice if the test time likelihood is important for the task at hand; however, simple temperature scaling might be also preferred for this in practice since it adds only minor computational overhead compared to ensemble methods. Of all the discussed methods, only direct outlier exposure successfully handles OOD samples and achieves OOD-Entropy that is close to the maximum value of ≈ 0.7. Interestingly, it also achieves slightly higher test accuracy, likely due to the positive effect of the noise injection introduced by the outliers falling in the original class data regions.

2.5 Conclusion

In this chapter, we have inspected some of the popular methods of augmenting neural network outputs with a measure of uncertainty. The presented overview is, of course, far from complete. The methods not covered by this survey include modeling the full distribution $p(\mathbf{x}, \mathbf{y})$ [143, 12, 186], as well as augmenting the pipelines with separate OOD

Table 2.1: *Comparison of deep uncertainty estimation methods on differerent metrics*

Method	Accuracy↑	NLL↓	Brier↓	ECE↓	OOD-Entropy↑
Vanilla NLL (128 neurons)	85.8%	0.345	0.102	0.045	0.024
Vanilla NLL (512 neurons)	86.5%	0.437	0.107	0.081	0.009
+ Temperature Scaling [110]	86.5%	0.320	0.098	0.045	0.031
Monte Carlo Dropout [86]	86.8%	**0.311**	**0.093**	0.039	0.101
Deep Ensembles [150]	86.8%	0.372	0.102	0.072	0.070
Outlier Exposure [118, 117]	**87.5%**	0.342	0.101	**0.030**	**0.689**

detector modules and "reject" option [30, 158, 153, 23]. Uncertainty quantification in deep learning is an emerging field, and we refer to [248] for a more comprehensive recent survey. Closely related to the uncertainty quantification, there is also an extensive amount of work on constructing networks and optimization techniques robust to *adversarial examples* [170, 18, 266, 281], which could be considered particularly complex cases of OOD samples.

The resulting quality of produced uncertainty estimates significantly depends on the selected method, amount of available data and its dimensionality, as well as the network design choices and its hyper-parameters. As of today, there is no single method that will provide ideal uncertainty measures in all scenarios. Therefore, one should always carefully consider a practical task at hand. In some cases, the i.i.d. assumption about train and test data can safely be made, and a simple post-processing calibration can achieve great results. On the other hand, in dynamic real-world environments, detection and proper handling of out-of-distribution samples is critical [16].

However, we have also observed that defining network outputs as conditional densities and the usage of *proper scoring rules* [95] for optimization are the basic building blocks that allow consistent and rigorous uncertainty reasoning. At the same time, Gaussian likelihood model discussed in Section 2.3.1 is not a valid choice for all regression tasks. In particular, for the task of angular regression, when the output values lie in the interval $[-\pi, \pi]$, normal distribution will not properly model periodicity of the circular data. Therefore, in the next chapter, we will study and develop methods for deep probabilistic circular regression that allow proper modeling of this domain. Additionally, while basic adaptive noise models like (2.7) can only predict single-mode distributions, we will develop a framework that allows rich, multi-modal densities.

Chapter 3

Probabilistic Circular Regression

3.1 Introduction

Estimating object pose is an important building block in systems aiming to understand complex scenes and has a long history in computer vision [172, 185]. Whereas early systems achieved low accuracy, recent advances in deep learning and the collection of extensive data sets have led to high performing systems that can be deployed in useful applications [211, 176, 28].

However, the reliability of object pose regression depends on the quality of the image provided to the system. Key challenges are low-resolution due to distance of an object to the camera, blur due to motion of the camera or the object, and sensor noise in case of poorly lit scenes (see Figure 3.1).

We would like to predict object pose in a way that captures uncertainty. As we discussed in the previous chapter, *probability* is the right way to capture the uncertainty [26] and in this chapter we therefore propose a novel model for object pose regression whose predictions are fully probabilistic. Figure 3.1 depicts an output of the proposed system. Moreover, instead of assuming a fixed form for the predictive density, we allow for very flexible distributions, specified by a deep neural network.

The value of quantified uncertainty for the task of object orientation estimation is twofold: *first*, a high prediction uncertainty is a robust way to diagnose poor inputs to the system; *second*, given accurate probabilities we can summarize them to improved point estimates using Bayesian decision theory.

In the following sections we present our method and make the following contributions:

- We demonstrate the importance of probabilistic regression on the application of object pose estimation;
- We propose a novel efficient probabilistic deep learning model for the task of circular regression;
- We show on a number of challenging pose estimation datasets (including PASCAL 3D+ benchmark [275]) that the proposed probabilistic method outperforms purely discriminative approaches in terms of predictive likelihood and shows competitive performance in terms of angular deviation losses classically used for the tasks.

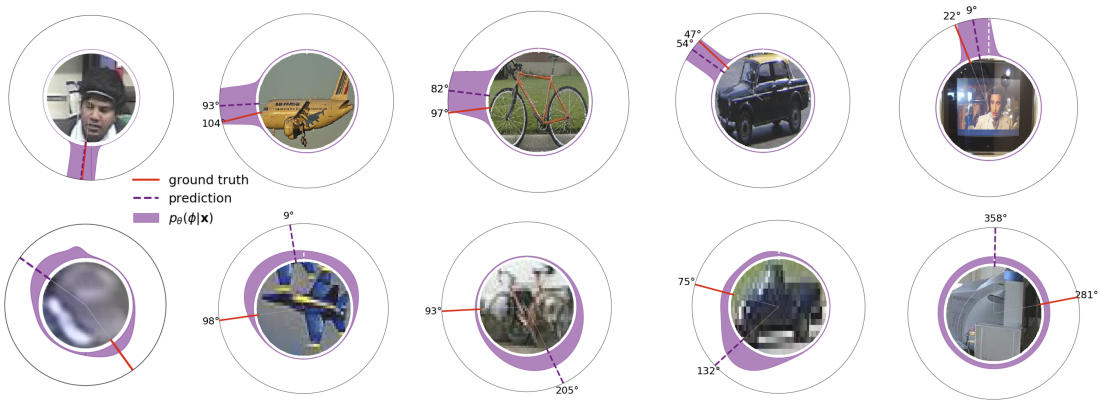


Figure 3.1: Our model predicts complex multimodal distributions on the circle (truncated by the outer circle for better viewing). For difficult and ambiguous images our model report high uncertainty (bottom row). Pose estimation predictions (pan angle) on images from IDIAP, TownCentre and PASCAL3D+ datasets.

3.2 Related Work

Estimation of object orientation arises in different applications and in this chapter we focus on the two most prominent tasks: head pose estimation and object class orientation estimation. Although those tasks are closely related, they have been studied mostly in separation, with methods applied to exclusively one of them. We will therefore discuss them separately, despite the fact that our model applies to both tasks. We present results for our methods on the standard benchmarks from both domains.

Head pose estimation has been a subject of extensive research in computer vision for a long time [242, 185] and the existing systems vary greatly in terms of feature representation and proposed classifiers. The input to pose estimation systems typically consists of 2D head images [195, 105, 62], and often one has to cope with low resolution images [184, 77, 25, 242]. Additional modalities such as depth [75] and motion [25, 45] information has been exploited and provide useful cues. However, these are not always available. Also, information about the full body image could be used for joint head and body pose prediction [47, 79, 198]. Notably, the work of [79] also promotes a probabilistic view and fuses body and head orientation within a tracking framework. Finally, the output of facial landmarks can be used as an intermediate step [61, 287].

Existing head pose estimation models are diverse and include manifold learning approaches [167, 124, 265, 22], energy-based models [198], linear regression based on HOG features [89], regression trees [75, 137] and convolutional neural networks [28]. A number of probabilistic methods for head pose analysis exist in the literature [21, 63, 79], but none of them combine probabilistic framework with learnable hierarchical feature representations from deep CNN architectures. At the same time, deep probabilistic mod-

els have shown an advantage over purely discriminative models in other computer vision tasks, e.g., depth estimation [139]. To the best of our knowledge, our work is the first to utilize deep probabilistic approach to angular orientation regression task.

An early dataset for estimating the *object rotation for general object classes* was proposed in [235] along with an early benchmark set. Over the years the complexity of data increased, from object rotation [235] and images of cars in different orientations [200] to Pascal3D [276]. The work of [276] then assigned a separate Deformable Part Model (DPM) component to a discrete set of viewpoints. The work of [206, 207] then proposed different 3D DPM extensions which allowed viewpoint esimation as integral part of the model. However, both [206] and [207] do not predict a continuous angular estimate but only a discrete number of bins.

More recent versions make use of CNN models but still do not take a probabilistic approach [211, 176]. The work of [254] investigates the use of a synthetic rendering pipeline to overcome the scarcity of detailed training data. The addition of synthetic and real examples allows them to outperform previous results. The model in [254] predicts angles and constructs a loss function that penalizes geodesic and ℓ_1 distance. In this work, we advocate the use of likelihood estimation as a principled probabilistic training objective.

Many works phrase angular prediction as a classification problem [211, 267, 254] which always limits the granularity of the prediction and also requires the design of a loss function and a means to select the number of discrete labels. The recent work of [267] draws a connection between viewpoints and object keypoints. Again, the viewpoint estimation is framed as a classification problem in terms of Euler angles to obtain a rotation matrix from a canonical viewpoint. A benefit of a classification model is that components like softmax loss can be re-used and also interpreted as an uncertainty estimate. In contrast, our model mitigate this problem: the likelihood principle suggests a direct way to train parameters, moreover ours is the only model in this class that conveys an uncertainty estimate.

3.3 Review of Biternion Networks

We build on the Biternion networks method for pose estimation from [28] and briefly review the basic ideas here. Biternion networks regress angular data and currently define the state-of-the-art model for a number of challenging head pose estimation datasets.

The key problem is to regress angular orientation which is periodic. The periodicity prevents a straight-forward application of standard regression methods, including CNN models with common loss functions. Consider a ground truth value of 0°, then both predictions 1° and 359° should result in the same absolute loss. Applying the $\bmod$ operator is no simple fix to this problem, since it results in a discontinuous loss function that complicates the optimization.

A loss function needs to be defined to cope with this discontinuity of the target value.

Biternion networks overcome this difficulty by using a different parameterization of angles and the cosine loss function between angles.

3.3.1 Biternion Representation

Beyer et al. [28] propose an alternative representation of an angle ϕ using the two-dimensional sine and cosine components $\mathbf{y} = (\cos\phi, \sin\phi)$.

This *biternion representation* is inspired by quaternions, which are popular in computer graphics systems. It is easy to predict a $(\cos, \sin)$ pair with a fully-connected layer followed by a normalization layer, that is,

$$f_{BT}(\mathbf{x}; \mathbf{W}, \mathbf{b}) = \frac{\mathbf{W}\mathbf{x} + \mathbf{b}}{||\mathbf{W}\mathbf{x} + \mathbf{b}||}. \tag{3.1}$$

A Biternion network is then a convolutional neural network with a layer (3.1) as the final operation, outputting a two-dimensional vector $\mathbf{y}_{pred}$. We use VGG-style network [241] and InceptionResNet [256] networks in our experiments and provide a detailed description of the network architecture in Section 3.6.1. Given recent developments in network architectures it is likely that different network topologies may perform better than selected backbones. We leave this for future work, as our contributions are orthogonal to the choice of the basis model.

3.3.2 Cosine Loss Function

The cosine distance is chosen in [28] as a natural candidate to measure the difference between the predicted and ground truth Biternion vectors. It reads

$$L_{cos}(\mathbf{y}_{pred}, \mathbf{y}_{true}) \quad = \quad 1 - \frac{\mathbf{y}_{pred} \cdot \mathbf{y}_{true}}{||\mathbf{y}_{pred}|| \cdot ||\mathbf{y}_{true}||} = 1 - \mathbf{y}_{pred} \cdot \mathbf{y}_{true}, \tag{3.2}$$

where the last equality is due to $||\mathbf{y}|| = \cos^2\phi + \sin^2\phi = 1$.

The combination of a Biternion angle representation and a cosine loss solves the problems of regressing angular values, allowing for a flexible deep network with angular output. We take this state-of-the-art model and generalize it into a family of probabilistic models of gradually increasing flexibility.

3.4 Probabilistic models of circular data.

The von Mises ($\mathcal{VM}$) distribution [173] is the basic building block of our probabilistic framework for circular data. We continue with a brief formal definition and in Section 3.4.1 describe a simple way to convert the output of Biternion networks into a $\mathcal{VM}$ density that does not require any network architecture change or re-training as it requires

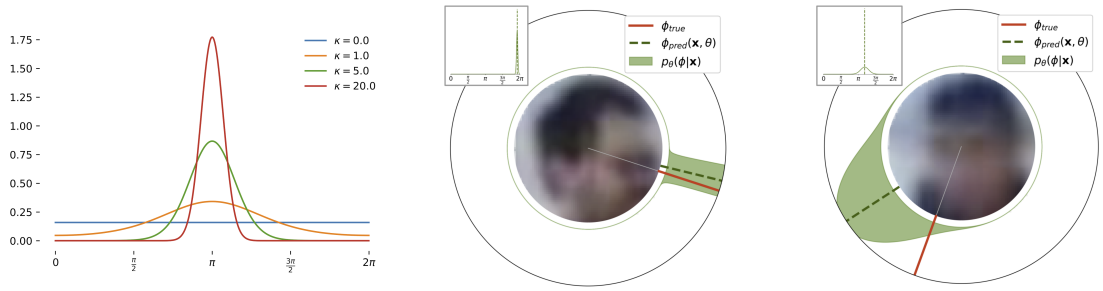


Figure 3.2: *Left*: examples of the von Mises probability density function for different concentration parameters κ. *Center, right*: predicted distributions for two images from the CAVIAR dataset. We plot the predicted density on the viewing circle. For comparison we also include the 2D plot (better visible in zoomed pdf version). The distribution on the center image is very certain, the one on the right more uncertain about the viewing angle.

only selection of the model variance. We will then use this approach as a baseline for more advanced probabilistic models. Section 3.4.2 slightly extends the original Biternion network by introducing an additional network output unit that models uncertainty of our angle estimation and allows optimization for the log-likelihood of the $\mathcal{VM}$ distribution.

The von Mises distribution $\mathcal{VM}(\mu, \kappa)$ is a close approximation of a normal distribution on the unit circle. Its probability density function is

$$p(\phi; \mu, \kappa) = \frac{\exp(\kappa \cos(\phi - \mu))}{2\pi I_0(\kappa)}, \tag{3.3}$$

where $\mu \in [0, 2\pi)$ is the mean value, $\kappa \in \mathbb{R}^+$ is a measure of concentration (a reciprocal measure of dispersion, so $1/\kappa$ is analogous to σ^2 in a normal distribution), and $I_0(\kappa)$ is the modified Bessel function of order 0. We show examples of $\mathcal{VM}$-distributions with $\mu = \pi$ and varying κ values in Figure 3.2 (left).

3.4.1 Von Mises Biternion Networks

A conceptually simple way to turn the Biternion networks from Section 3.3 into a probabilistic model is to take its predicted value as the center value of the $\mathcal{VM}$ distribution,

$$p_\theta(\phi | \mathbf{x}; \kappa) = \frac{\exp(\kappa \cos(\phi - \mu_\theta(\mathbf{x})))}{2\pi I_0(\kappa)}, \tag{3.4}$$

where $\mathbf{x}$ is an input image, θ are parameters of the network, and $\mu_\theta(\mathbf{x})$ is the network output. In order to arrive at a probability distribution we may regard $\kappa > 0$ as a hyper-

parameter. For the fixed network parameters θ we can select κ by maximizing the log-likelihood of the observed data,

$$\kappa^* = \underset{\kappa}{\operatorname{argmax}} \sum_{i=1}^{N} \log p_\theta(\phi^{(i)}|\mathbf{x}^{(i)}; \kappa). \tag{3.5}$$

The model (3.4) with κ^* will serve as the simplest probabilistic baseline in our comparisons.

3.4.2 Maximizing the von Mises Log-likelihood

Using a single scalar κ for every possible input in the model (3.4) is clearly a restrictive assumption: model certainty should depend on factors such as image quality, light conditions, etc. For example, Figure 3.2 (center, right) depicts two low resolution images from a surveillance camera that are part of the CAVIAR dataset [77]. In the left image facial features like eyes and ears are distinguishable which allows a model to be more certain when compared to the more blurry image on the right.

We therefore extend the simple model by replacing the single constant κ with a function $\kappa_\theta(\mathbf{x})$, predicted by the Biternion network,

$$p_\theta(\phi|\mathbf{x}) = \frac{\exp\left(\kappa_\theta(\mathbf{x})\cos\left(\phi - \mu_\theta(\mathbf{x})\right)\right)}{2\pi I_0(\kappa_\theta(\mathbf{x}))}. \tag{3.6}$$

We train (3.6) by maximizing the log-likelihood of the data,

$$\log \mathcal{L}(\theta|\mathbf{X}, \Phi) = \sum_{i=1}^{N} \kappa_\theta(\mathbf{x}^{(i)})(\cos\left(\phi^{(i)} - \mu_\theta(\mathbf{x}^{(i)})\right) - \sum_{i=1}^{N} \log 2\pi I_0(\kappa_\theta(\mathbf{x}^{(i)})). \tag{3.7}$$

Note that when κ is held constant in (3.7), the second sum in $\log \mathcal{L}(\theta|\mathbf{X}, \Phi)$ is constant and therefore we recover the Biternion cosine objective (3.2) up to constants C_1, C_2,

$$\log \mathcal{L}(\theta|\mathbf{X}, \Phi, \kappa) = C_1 \sum_{i=1}^{N} \cos\left(\phi^{(i)} - \mu_\theta(\mathbf{x}^{(i)})\right) + C_2.$$

The sum has the equivalent form,

$$\begin{aligned} \sum_{i=1}^{N} \cos\left(\phi^{(i)} - \mu_\theta(\mathbf{x}^{(i)})\right) &= \sum_{i=1}^{N} \left[\cos\phi^{(i)} \cos\mu_\theta(\mathbf{x}^{(i)}) + \sin\phi^{(i)} \sin\mu_\theta(\mathbf{x}^{(i)})\right] & (3.8) \\ &= \sum_{i=1}^{N} \mathbf{y}_{\phi^{(i)}} \cdot \mathbf{y}_{\mu_\theta(\mathbf{x}^{(i)})}, & (3.9) \end{aligned}$$

where $\mathbf{y}_\phi = (\cos\phi, \sin\phi)$ is a Biternion representation of an angle. Note, that the above

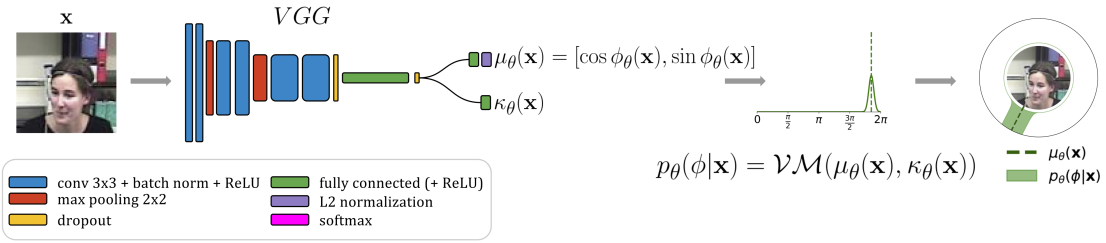


Figure 3.3: *The single mode von Mises model (VGG backbone variation).* A Biternion-VGG network regresses both mean and concentration parameter of a single vM distribution.

derivation shows that the loss function in [28] corresponds to optimizing the von Mises log-likelihood for the fixed value of $\kappa = 1$. This offers an interpretation of Biternion networks as a probabilistic model.

The additional degree of freedom to learn $\kappa_\theta(\mathbf{x})$ as a function of $\mathbf{x}$ allows us to capture the desired image-dependent uncertainty as can be seen in Figure 3.2.

However, just like the Gaussian distribution, the von Mises distribution makes a specific assumption regarding the shape of the density. We now show how to overcome this limitation by using a mixture of von Mises distributions.

3.5 Mixture of von Mises Distributions

The model described in Section 3.4.2 is only unimodal and can not capture ambiguities in the image. However, in the case of blurry images like the ones in Figure 3.2 we could be interested in distributing the mass around a few potential high probability hypotheses. For example, the model could predict that a person is looking sideways, but could not determine the direction, left or right, with certainty. In this section, we present two models that are able to capture multimodal beliefs while retaining a calibrated uncertainty measure.

3.5.1 Finite Mixture of von Mises Distributions

One common way to generate complex distributions is to sum multiple distributions into a *mixture distribution*. We introduce K different component distributions and a K-dimensional probability vector representing the mixture weights. Each component is a simple von Mises distribution. We can then represent our density function as

$$p_\theta(\phi|\mathbf{x}) = \sum_{j=1}^{K} \pi_j(\mathbf{x}, \theta)\, p_j(\phi|\mathbf{x}, \theta), \tag{3.10}$$

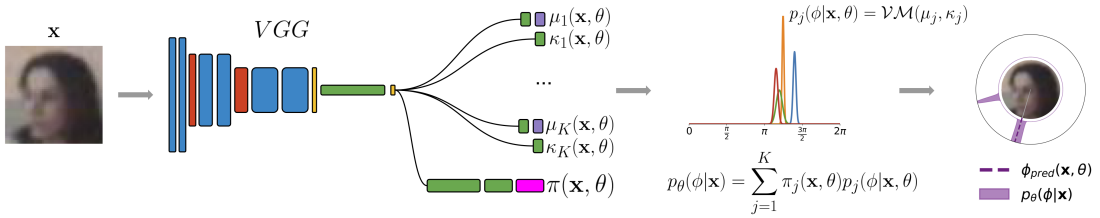


Figure 3.4: *The finite $\mathcal{VM}$ mixture model.* A VGG network predicts K mean and concentration values and the mixture coefficients π. This allows to capture multimodality in the output.

where $p_j(\phi|\mathbf{x},\theta) = \mathcal{VM}(\phi|\mu_j,\kappa_j)$ for $j = 1,\ldots,K$ are the K component distributions and the mixture weights are $\pi_j(\mathbf{x},\theta)$ so that $\sum_j \pi_j(\mathbf{x},\theta) = 1$. We denote all parameters with the vector θ, it contains component-specific parameters as well as parameters shared across all components.

To predict the mixture in a neural network framework, we need $K \times 3$ output units for modeling all von Mises component parameters (two for modeling the Biternion representation of the mean, $\mu_j(\mathbf{x},\theta)$ and one for the $\kappa_j(\mathbf{x},\theta)$ value), as well as K units for the probability vector $\pi_j(\mathbf{x},\theta)$, defined by taking the `softmax` operation to get positive mixture weights.

The finite von Mises density model then takes form

$$p_\theta(\phi|\mathbf{x}) = \sum_{j=1}^{K} \pi_j(\mathbf{x},\theta) \frac{\exp\Big(\kappa_j(\mathbf{x},\theta)\cos\big(\phi - \mu_j(\mathbf{x},\theta)\big)\Big)}{2\pi I_0\big(\kappa_j(\mathbf{x},\theta)\big)}. \tag{3.11}$$

Similarly to the single von Mises model, we can train by maximizing the log-likelihood of the observed data, $\sum_{i=1}^{N} \log p_\theta(\phi^{(i)}|\mathbf{x}^{(i)})$. We show an overview of the model in Figure 3.4.

3.5.2 Infinite Mixture (CVAE)

To extend the model from a finite to an infinite mixture model, we follow the variational autoencoder (VAE) approach [142, 223], and introduce a vector-valued latent variable $\mathbf{z}$. The resulting model is depicted in Figure 3.5. The continuous latent variable becomes the input to a decoder network $p(\phi|\mathbf{x},\mathbf{z})$ which predicts the parameters—mean and concentration—of a single von Mises component. We define our density function as the infinite sum (integral) over all latent variable choices, weighted by a learned distribution $p(\mathbf{z}|\mathbf{x})$,

$$p_\theta(\phi|\mathbf{x}) = \int p(\phi|\mathbf{x},\mathbf{z})\, p(\mathbf{z}|\mathbf{x}) d\mathbf{z}, \tag{3.12}$$

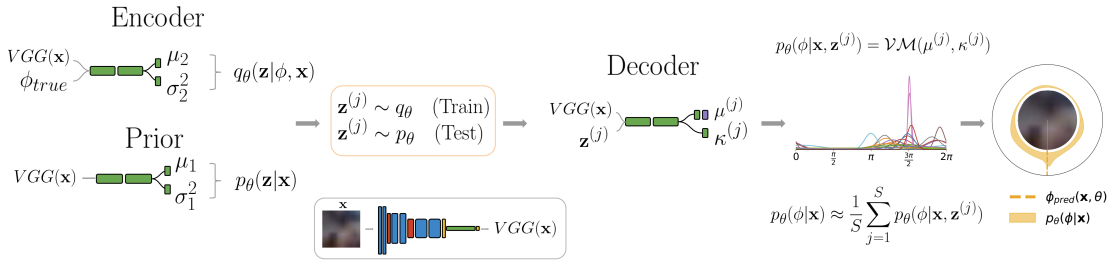


Figure 3.5: *The infinite mixture model (CVAE).* An encoder network predicts a distribution $q(\mathbf{z}|\mathbf{x})$ over latent variables $\mathbf{z}$, and a decoder network $p(\phi|\mathbf{x},\mathbf{z})$ defines individual mixture components. Integrating over $\mathbf{z}$ yields an infinite mixture of von Mises distributions. In practice we approximate this integration using a finite number of Monte Carlo samples $\mathbf{z}^{(j)} \sim q(\mathbf{z}|\mathbf{x})$.

where $p_\theta(\phi|\mathbf{x},\mathbf{z}) = \mathcal{VM}(\mu(\mathbf{x},\theta),\kappa(\mathbf{x},\theta))$, and $p_\theta(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mu_1(\mathbf{x},\theta),\sigma_1^2(\mathbf{x},\theta))$. The log-likelihood $\log p_\theta(\phi|\mathbf{x})$ for this model is no longer tractable, preventing simple maximum likelihood training. Instead, we use the variational autoencoder framework of [142, 223] in the form of the conditional VAE (CVAE) [249]. The CVAE formulation uses an auxiliary *variational* density $q_\theta(\mathbf{z}|\mathbf{x},\phi) = \mathcal{N}(\mu_2(\mathbf{x},\phi,\theta),\sigma_2^2(\mathbf{x},\phi,\theta))$ and instead of the log-likelihood optimizes a *variational lower bound*,

$$\log p_\theta(\phi|\mathbf{x}) = \log \int p_\theta(\phi|\mathbf{x},\mathbf{z}) p_\theta(\mathbf{z}|\mathbf{x}) d\mathbf{z} \tag{3.13}$$

$$= \log \int p_\theta(\phi|\mathbf{x},\mathbf{z}) \frac{q_\theta(\mathbf{z}|\mathbf{x},\phi)}{q_\theta(\mathbf{z}|\mathbf{x},\phi)} p_\theta(\mathbf{z}|\mathbf{x}) d\mathbf{z} \tag{3.14}$$

$$= \log \mathbb{E}_{\mathbf{z}\sim q_\theta(\mathbf{z}|\mathbf{x},\phi)} \left[p_\theta(\phi|\mathbf{x},\mathbf{z}) \frac{p_\theta(\mathbf{z}|\mathbf{x})}{q_\theta(\mathbf{z}|\mathbf{x},\phi)} \right] \tag{3.15}$$

$$\geq \mathbb{E}_{\mathbf{z}\sim q_\theta(\mathbf{z}|\mathbf{x},\phi)} \left[\log p_\theta(\phi|\mathbf{x},\mathbf{z}) + \log \frac{p_\theta(\mathbf{z}|\mathbf{x})}{q_\theta(\mathbf{z}|\mathbf{x},\phi)} \right] \tag{3.16}$$

$$= \mathbb{E}_{\mathbf{z}\sim q_\theta(\mathbf{z}|\mathbf{x},\phi)} \left[\log p_\theta(\phi|\mathbf{x},\mathbf{z}) \right] - KL\Big(q_\theta(\mathbf{z}|\mathbf{x},\phi) \parallel p_\theta(\mathbf{z}|\mathbf{x}) \Big), \tag{3.17}$$

where $KL(q||p)$ is the Kullback-Leibler divergence [149] and (3.16) is due to Jensen's inequality [188]. We refer to [142, 249, 67, 223] for more details on VAEs. The CVAE model is composed of multiple deep neural networks: an *encoder network* $q_\theta(\mathbf{z}|\mathbf{x},\phi)$, a *conditional prior network* $p_\theta(\mathbf{z}|\mathbf{x})$, and a *decoder network* $p_\theta(\phi|\mathbf{x},\mathbf{z})$. Like before, we use θ to denote the entirety of trainable parameters of all three model components. We show an overview of the model in Figure 3.5.

The model is trained by maximizing the variational lower bound (3.17) over the training set $(\mathbf{X},\Phi)$, where $\mathbf{X} = (\mathbf{x}^{(1)},\ldots,\mathbf{x}^{(N)})$ are the images and $\Phi = (\phi^{(1)},\ldots,\phi^{(N)})$ are the

ground truth angles. We maximize

$$\hat{\mathcal{L}}_{\text{CVAE}}(\theta|\mathbf{X},\Phi) = \frac{1}{N}\sum_{i=1}^{N}\hat{\mathcal{L}}_{\text{ELBO}}(\theta|\mathbf{x}^{(i)},\phi^{(i)}), \tag{3.18}$$

where we use $\hat{\mathcal{L}}_{\text{ELBO}}$ to denote the Monte Carlo approximation to (3.17) using S samples. We can optimize (3.18) efficiently using stochastic gradient descent.

To evaluate the log-likelihood during testing, we use the importance-weighted sampling technique proposed in [42] to derive a stronger bound on the marginal likelihood,

$$\log p_\theta(\phi|\mathbf{x}) \geq \log\frac{1}{S}\sum_{j=1}^{S}\frac{p_\theta(\phi|\mathbf{x},\mathbf{z}^{(j)})\,p_\theta(\mathbf{z}^{(j)}|\mathbf{x})}{q_\theta(\mathbf{z}^{(j)}|\mathbf{x},\phi)}, \tag{3.19}$$

$$\mathbf{z}^{(j)} \sim q_\theta(\mathbf{z}^{(j)}|\mathbf{x},\phi) \qquad j=1,\ldots,S. \tag{3.20}$$

Simplified CVAE. In our experiments we also investigate a variant of the aforementioned model where $p_\theta(\mathbf{z}|\mathbf{x}) = q_\theta(\mathbf{z}|\mathbf{x},\phi) = p(\mathbf{z}) = \mathcal{N}(0,I)$. Compared to the full CVAE framework, this model, which we refer to as *simplified CVAE* (sCVAE) in the experiments, sacrifices the adaptive input-dependent density of the hidden variable $\mathbf{z}$ for faster training and test inference, as well as optimization stability. In that case the KL-divergence $KL\big(q_\theta \parallel p_\theta\big)$ term in $\hat{\mathcal{L}}_{\text{ELBO}}$ becomes zero, and we train for a Monte Carlo estimated log-likelihood of the data:

$$\hat{\mathcal{L}}_{\text{sCVAE}}(\theta|\mathbf{X},\Phi) = \frac{1}{N}\sum_{i=1}^{N}\log\Big(\frac{1}{S}\sum_{j=1}^{S}p_\theta(\phi^{(i)}|\mathbf{x}^{(\mathbf{i})},\mathbf{z}^{(j)})\Big), \tag{3.21}$$

$$\mathbf{z}^{(j)} \sim p(\mathbf{z}) = \mathcal{N}(0,I), j=1,\ldots,S. \tag{3.22}$$

In some applications it is necessary to make a single best guess about the pose, that is, to summarize the posterior $p(\phi|\mathbf{x})$ to a single point prediction $\hat{\phi}$. We now discuss an efficient way to do that.

3.5.3 Point Prediction

To obtain an optimal single point prediction, we utilize Bayesian decision theory [26, 213, 36] and minimize the expected loss,

$$\hat{\phi}_\Delta = \underset{\phi\in[0,2\pi)}{\operatorname{argmin}}\,\mathbb{E}_{\phi'\sim p(\phi|\mathbf{x})}\left[\Delta(\phi,\phi')\right], \tag{3.23}$$

where $\Delta : [0,2\pi)\times[0,2\pi)\to\mathbb{R}^+$ is a loss function. We will use the $\Delta_{\text{AAD}}(\phi,\phi')$ loss which measures the absolute angular deviation (AAD). To approximate (3.23) we draw

S samples $\{\phi_j\}$ from $p_\theta(\phi|\mathbf{x})$ and then use the empirical approximation of [213],

$$\hat{\phi}_\Delta = \underset{j=1,\ldots,S}{\operatorname{argmin}} \frac{1}{S} \sum_{k=1}^{S} \Delta(\phi_j, \phi_k). \tag{3.24}$$

We now evaluate our models both in terms of uncertainty, as well as in terms of point prediction quality.

3.6 Experiments

This section presents the experimental results on several challenging head and object pose regression tasks. Section 3.6.1 introduces the experimental setup including used datasets, network architecture and training setup. In Section 3.6.2 we present and discuss qualitative and quantitative results on the datasets of interest.

3.6.1 Experimental Setup

Network architecture. We build our probabilistic framework on top of the Biternion network approach. Therefore, for all the experiments on the head pose regression, we use the same deep batch-normalized VGG-style network [241] architecture as in [28]. The architecture consists of six convolutional layers with 24, 24, 48, 48, 64 and 64 feature channels, respectively, followed by a fully-connected layer of a variable length. The final layer was set to match the number of required parameters of the probabilistic model. For the CVAE, the same architecture is used for both the encoder and decoder network.

For the experiments on the PASCAL3D+ dataset [276], we use InceptionResNet [256] as a backbone architecture and jointly predict distributions over three angles (azimuth, elevation and tilt). We use a separate model for each class of objects and consider constructing a single shot probabilistic object detector and pose estimator a future work. All models were implemented in Keras [53] using the TensorFlow [9] back-end. Code and data for all the experiments are available [1].

Training. We optimize using Adam [141] and perform random search proposed in [27] for finding the best values of hyper-parameters such as dropout values, batch size, fully connected layers sizes, learning rate and other optimizer parameters. For the head-pose estimation tasks, we train all networks for 1000 epochs, with an early stopping in case of no improvement of validation loss after 200 consecutive steps. For PASCAL3D+, we train for 200 epochs with an early stopping after 10 epochs of no improvement.

Head pose datasets. We evaluate all methods together with the non-probabilistic BiternionVGG baseline on three diverse (in terms of image quality and precision of pro-

[1] `https://github.com/sergeyprokudin/deep_direct_stat`

Table 3.1: Quantitative results on the IDIAP head pose estimation dataset [195] for the three head rotations pan, roll and tilt. In the situation of fixed camera pose, lightning conditions and image quality, all methods show similar performance (methods are considered to perform on par when the difference in performance is less than *standard error of the mean).*

estimated pose component	pan		tilt		roll	
	MAAD	log-likelihood	MAAD	log-likelihood	MAAD	log-likelihood
Beyer et al. ([28]), fixed κ	$\mathbf{5.8}^\circ \pm \mathbf{0.1}^*$	0.37 ± 0.01	$\mathbf{2.4}^\circ \pm \mathbf{0.1}$	1.31 ± 0.01	$\mathbf{3.1}^\circ \pm \mathbf{0.1}$	$\mathbf{1.13 \pm 0.01}$
Ours (single von Mises)	$6.3^\circ \pm 0.1$	$\mathbf{0.56 \pm 0.01}$	$\mathbf{2.3}^\circ \pm \mathbf{0.1}$	$\mathbf{1.56 \pm 0.01}$	$3.4^\circ \pm 0.1$	$\mathbf{1.13 \pm 0.01}$
Ours (mixture-CVAE)	$6.4^\circ \pm 0.1$	$\approx 0.52 \pm 0.02$	$2.9^\circ \pm 0.1$	$\approx 1.35 \pm 0.01$	$3.5^\circ \pm 0.1$	$\approx 1.05 \pm 0.02$

*standard error of the mean (SEM).

Table 3.2: Quantitative results on the CAVIAR-o [77] and TownCentre [24] coarse gaze estimation datasets. We see clear improvement in terms of quality of probabilistic predictions for both datasets when switching to mixture models that allow to output multiple hypotheses for gaze direction.

	CAVIAR-o		TownCentre	
	MAAD	log-likelihood	MAAD	log-likelihood
Beyer et al. ([28]), fixed κ	$5.74^\circ \pm 0.13$	0.262 ± 0.031	$22.8^\circ \pm 1.0$	-0.89 ± 0.06
Ours (single von Mises)	$5.53^\circ \pm 0.13$	0.700 ± 0.043	$22.9^\circ \pm 1.1$	-0.57 ± 0.05
Ours (mixture-finite)	$\mathbf{4.21}^\circ \pm \mathbf{0.16}$	$\mathbf{1.87 \pm 0.04}$	$23.5^\circ \pm 1.1$	$\mathbf{-0.50 \pm 0.04}$

vided ground truth information) headpose datasets: IDIAP head pose [195], TownCentre [24] and CAVIAR [77] coarse gaze estimation. The IDIAP head pose dataset contains 66295 head images stemmed from a video recording of a few people in a meeting room. Each image has a complete annotation of a head pose orientation in form of pan, tilt and roll angles. We take 42304, 11995 and 11996 images for training, validation, and testing, respectively. The TownCentre and CAVIAR datasets present a challenging task of a coarse gaze estimation of pedestrians based on low resolution images from surveillance camera videos. In case of the CAVIAR dataset, we focus on the part of the dataset containing occluded head instances (hence referred to as CAVIAR-o in the literature). We use (10802, 5444, 5445) and (6916, 874, 904) images for the training-validation-testing split for the CAVIAR and TownCentre datasets, respectively.

PASCAL3D+ object pose dataset. The Pascal 3D+ dataset [276] consists of images from the Pascal [74] and ImageNet [64] datasets that have been labeled with both detection and continuous pose annotations for the 12 rigid object categories that appear in Pascal VOC12 [74] train and validation set. With nearly 3000 object instances per category, this dataset provide a rich testbed to study general object pose estimation. In our experiments on this dataset we follow the same protocol as in [254, 267] for viewpoint estimation: we use ground truth detections for both training and testing, and use Pascal validation set to evaluate and compare the quality of our predictions.

Table 3.3: Results on PASCAL3D+ viewpoint estimation with ground truth bounding boxes. First two evaluation metrics are defined in [267], where $Acc_{\frac{\pi}{6}}$ measures accuracy (the higher the better) and $MedErr$ measures error (the lower the better). Additionally, we report the log-likelihood estimation $\log \mathcal{L}$ of the predicted angles (the higher the better). We can see clear improvement on all metrics when switching to probabilistic setting compared to training for a purely discriminative loss (fixed κ case).

	aero	**bike**	**boat**	**bottle**	**bus**	**car**	**chair**	**table**	**mbike**	**sofa**	**train**	**tv**	**mean**
$Acc_{\frac{\pi}{6}}$ (Tulsiani et al.[267])	0.81	0.77	0.59	0.93	**0.98**	0.89	**0.80**	0.62	0.88	0.82	0.80	0.80	0.81
$Acc_{\frac{\pi}{6}}$ (Su et al.[254])	0.80	0.82	**0.62**	0.95	0.93	0.83	0.75	**0.86**	0.86	0.85	0.82	0.89	0.83
$Acc_{\frac{\pi}{6}}$ (Ours, fixed κ)	0.83	0.75	0.54	0.95	0.92	0.90	0.77	0.71	**0.90**	0.82	0.80	0.86	0.81
$Acc_{\frac{\pi}{6}}$ (Ours, single v.Mises)	0.87	0.78	0.55	**0.97**	0.95	**0.91**	0.78	0.76	**0.90**	0.87	**0.84**	**0.91**	**0.84**
$Acc_{\frac{\pi}{6}}$ (Ours, mixture-sCVAE)	**0.89**	**0.83**	0.46	0.96	0.93	0.90	**0.80**	0.76	**0.90**	**0.90**	0.82	**0.91**	**0.84**
$MedErr$ (Tulsiani et al.[267])	13.8	17.7	21.3	12.9	5.8	9.1	14.8	15.2	14.7	13.7	8.7	15.4	13.6
$MedErr$ (Su et al.[254])	10.0	**12.5**	**20.0**	**6.7**	4.5	6.7	12.3	**8.6**	13.1	11.0	5.8	13.3	**10.4**
$MedErr$ (Ours, fixed κ)	11.4	18.1	28.1	6.9	4.0	6.6	14.6	12.1	12.9	16.4	7.0	12.9	12.6
$MedErr$ (Ours, single v.Mises)	**9.7**	17.7	26.9	**6.7**	**2.7**	4.9	12.5	8.7	13.2	10.0	4.7	**10.6**	10.7
$MedErr$ (Ours, mixture-sCVAE)	**9.7**	15.5	45.6	**5.4**	2.9	**4.5**	13.1	12.6	**11.8**	**9.1**	**4.3**	12.0	12.2
$\log \mathcal{L}$(Ours, fixed κ)	-0.89	**-0.73**	-1.21	0.18	2.09	1.43	-0.08	0.69	**-0.50**	-0.75	0.06	-1.02	-0.07 ± 0.15
$\log \mathcal{L}$(Ours, single v.Mises)	0.19	-1.12	-0.30	2.40	**4.87**	**2.85**	**0.42**	**0.79**	-0.72	**-0.54**	**2.52**	0.52	$\mathbf{1.17 \pm 0.07}$
$\log \mathcal{L}$(Ours, mixture-sCVAE)	**0.60**	**-0.73**	**-0.26**	**2.71**	4.45	2.52	-0.58	0.08	-0.62	-0.64	2.05	**1.14**	$\mathbf{1.15 \pm 0.07}$

3.6.2 Results and Discussion

Quantitative results. We evaluate our methods using both discriminative and probabilistic metrics. We use discriminative metrics that are standard for the dataset of interest in order to be able to compare our methods with previous work. For headpose tasks we use the *mean absolute angular deviation (MAAD)*, a widely used metric for angular regression tasks. For PASCAL3D+, we use the metrics advocated in [267]. Probabilistic predictions are measured in terms of log-likelihood [101, 96], a widely accepted scoring rule for assessing the quality of probabilistic predictions. We summarize the results in Tables 3.1, 3.2 and 3.3. It can be seen from results on IDIAP dataset presented in Table 3.1 that when camera pose, lightning conditions and image quality are fixed, all methods perform similarly. In contrast, for the coarse gaze estimation task on CAVIAR we can see a clear improvement in terms of quality of probabilistic predictions for both datasets when switching to mixture models that allow to output multiple hypotheses for gaze direction. Here, low resolution, pure light conditions and presence of occlusions create large diversity in the level of head pose expressions. Finally, on the challenging PASCAL3D+ dataset, we can see clear improvement on all metrics and classes when switching to a probabilistic setting compared to training for a purely discriminative loss (fixed κ case). Our methods also show competitive or superior performance compared to state-of-the-art methods on disriminative metrics advocated in [267]. Method of [254] uses large amounts of synthesized images in addition to the standard training set that was used by our method. Using this data augmentation technique can also lead to an

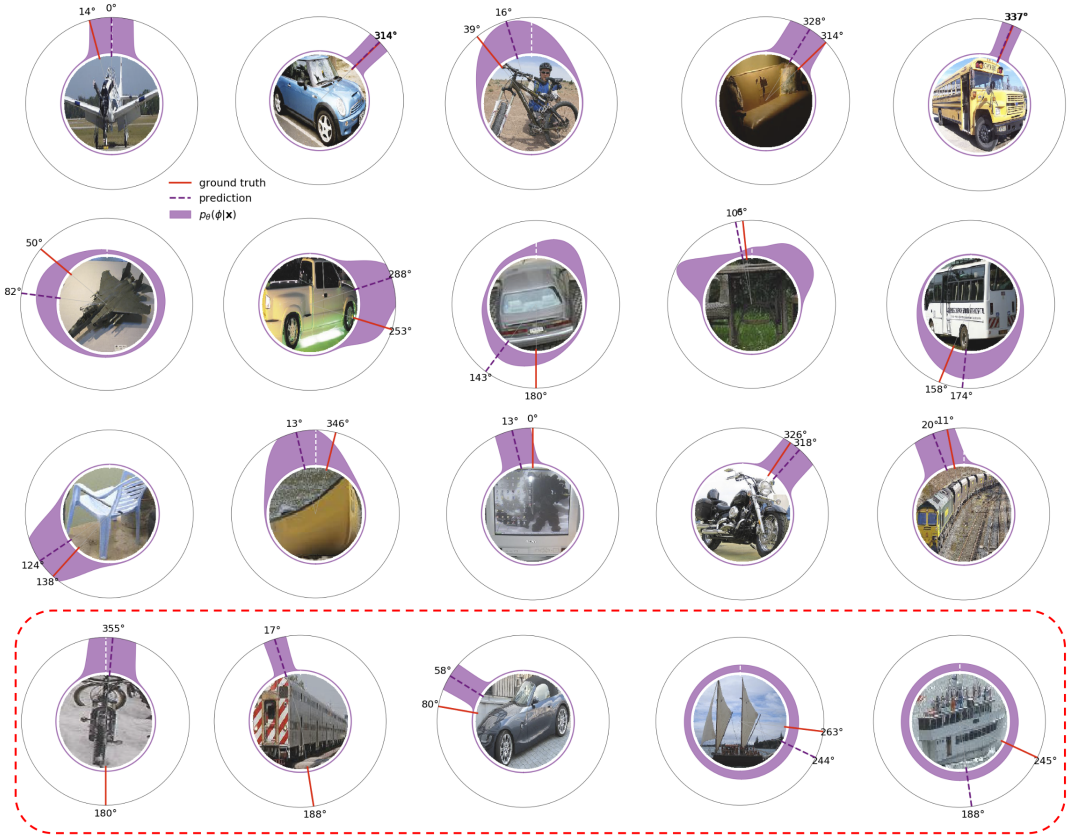


Figure 3.6: *Qualitative results of our simpified CVAE model on the PASCAL3D+ dataset.* Our model correctly quantifies the uncertainty of pose predictions and is able to model ambiguous cases by predicting complex multimodal densities (second row). Last row shows failure cases (confusing head and tail of the object with high confidence, uniform densities in hard cases).

improved performance of our method and we consider this a future work.

Qualitative results. Examples of probabilistic predictions for PASCAL3D+ dataset are shown in Figure 3.6. The first row highlights the effect we set out to achieve: to correctly quantify the level of uncertainty of the estimated pose. For easier examples we observe sharp peaks and a highly confident detection, and more spread-out densities otherwise. The examples on the second row highlight the advantage of mixture models, which allow to model complex densities with multiple peaks corresponding to more than one potential pose angle. Failure scenarios are highlighted in the last row: high confidence predictions in case if the model confuses head and tail of an object and tendency to uniform distributions ($\kappa \to 0$) for hard classes.

Uncertainty under image blur. To test whether the predicted uncertainty is calibrated, we performed an experiment where we controlled the image quality by Gaussian

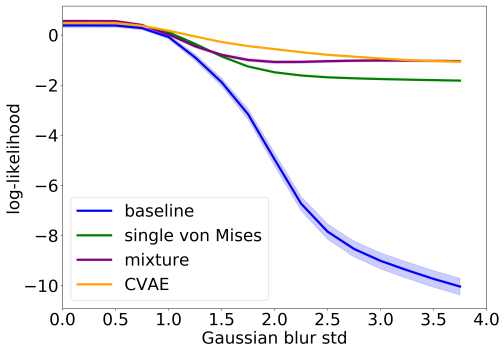


Figure 3.7: *Effect of Gaussian blur on the log-likelihood performance.* Aggregated results for 1000 images from the IDIAP test set.

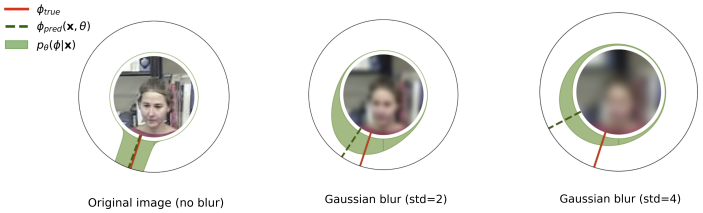


Figure 3.8: *Probabilistic predictions adapt to increased level of blur in the image.* Sample image from the IDIAP dataset.

blurring. For 1000 test images from IDIAP we applied blurs at different bandwidths and recorded log-likelihood for the ”pan” prediction of all models. The empirical result is shown in Figure 3.7 and an example image is shown in Figure 3.8. From the results we make the following observations. For all models, the performance in log-likelihood degrades with decreasing image quality. CVAE has the best likelihood estimate and is thus the best calibrated method. Both the baseline and the single vM model deteriorate the quickest in terms of log-likelihood. The simple extension of estimating κ successfully turns the baseline into a probabilistic model. The finite mixture model and the CVAE turn out to be most robust among the trained models. All probabilistic models retain a reasonable likelihood, the baseline model fails due to the fixed κ and the simple unimodal extension already performs significantly better.

3.7 Conclusion

In this chapter, we demonstrated a new probabilistic model for object pose estimation that is robust to variations in input image quality and accurately quantifies its uncertainty. More generally, our results confirm that our approach is flexible enough to accommodate different output domains such as angular data and enables rich and efficient probabilistic

deep learning models. We train all models by maximum likelihood but still find it to be competitive with other works from the literature that explicitly optimize for point estimates even under point estimate loss functions. In the future, in order to improve our predictive performance and robustness, we would also like to handle uncertainty of model parameters [139] and to use the Fisher-von Mises distribution to jointly predict a distribution of azimuth-elevation-tilt [173].

Chapter 4

Deep Probabilistic Models in Real-World Systems

As was rightly stated by George E.P. Box, *all models are wrong but some are useful [37]*. We have observed in Chapter 2 that none of the uncertainty quantification methods would work ideally in all data distribution scenarios and tasks. It is therefore important to observe how these models behave in practice; in this chapter, we will explore how variations of deep probabilistic frameworks discussed in previous sections are deployed in critical real-world systems that have high demand in computational efficiency and robustness of generated predictions.

In Section 4.1, we will investigate how deep probabilistic regression models can be efficiently utilized in brain imaging for robust determination of model contrast parameters. Substituting conventional methods of the field with neural network based approach leads to significant improvements in computational efficiency, while also providing useful and reliable uncertainty estimates.

In Section 4.2, we will use the variational auto-encoder regression framework for the robust real-time trajectory prediction of the ball in robotic table tennis environment. Again, we will show how the developed method outperforms classic approaches based on differential equations, as well as other deep network like long short-term memory approaches (LSTMs) that are regularly utilized for temporal prediction. The predicted trajectories are then used in real-time planning of robotic arm movement.

Importantly, both works were done in close collaboration with experts of the corresponding fields, with the strong focus on building reliable holistic systems. The contribution of the author of the present thesis was mainly in providing expertise in deep uncertainty quantification methods, as well as in creating early prototypes of the respective components. Therefore, in the follow up sections we will restrict ourselves to a brief formulation of the domain problem ans the essence of technical contribution, as well as the qualitative description of the most important results. We refer to original publications for the comprehensive description of these projects [93, 100].

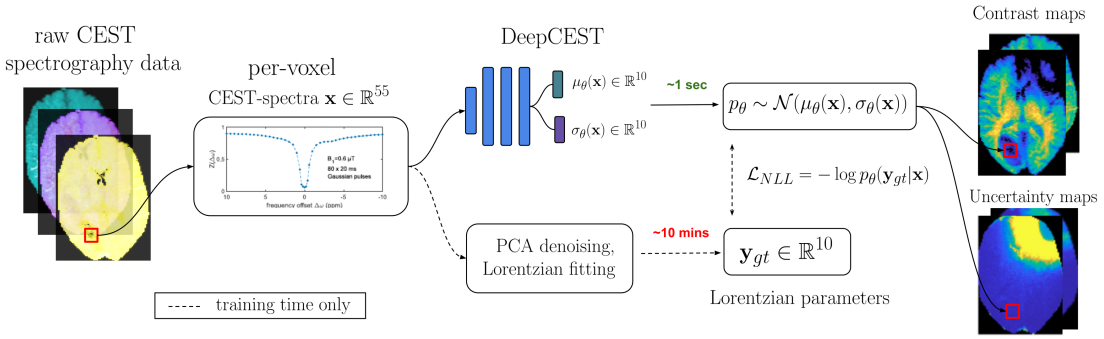


Figure 4.1: *DeepCEST framework [93].* Deep neural network is used to shortcut conventional MRI CEST-spectra model parameter determination (PCA denoising, Lorentzian model least square fitting), as well as to provide uncertainty measurements. This significantly reduces the amount of time it takes to generate final contrast maps, making an important step towards online clinical applications of CEST contrasts.

4.1 DeepCEST: Robust MRI Parameter Determination

Problem formulation. *Magnetic resonance imaging (MRI)* [159, 43] is a powerful technique that is nowadays widely used in both everyday medical diagnosis and scientific research. The imaging principle of MRI is based on polarizing atoms of hydrogen (abundant in organic tissues) with a strong magnetic field. The time it takes for the atoms to return to its normal state (so called *relaxation time*) will then provide the information on the type of the tissue it is coming from. E.g., in case of brain imaging, gray and white matter hydrogen will have different relaxation times. This information is regularly captured in the form of spectrography data provided for each tissue voxel. The obtained and processed differences, or *contrast maps*, can then be used in order to analyze the structure and detect pathologies in the tissues of interest.

Chemical exchange saturation transfer (CEST) [239, 268] is a relative novel MRI technique that, compared to conventional techniques, allows to detect chemical compounds in low concentrations. While have already being proved useful in the analysis of tumor cells [226], CEST imaging often requires complex and time-consuming mathematical modeling before contrast generation, e.g. Lorentzian models [278, 273, 98]. This modeling can be error-prone and time-consuming, because it commonly must be carried out offline and sometimes requires user interaction and expert knowledge.

Here, we introduce a probabilistic deep learning approach to shortcut conventional Lorentzian least square fitting analysis (Figure 4.1). By making use of the previously fitted data, this approach addresses the mentioned problems of complex non-linear models regarding evaluation speed and parameter fine-tuning. The objective is to produce several informative CEST contrasts simultaneously and robustly with the trained neural network, in a fraction of the normal evaluation time. At the same time, the networks are supposed to tell if predictions are trustworthy, or—more importantly—if they are not,

allowing confident interpretation of contrast changes.

Technical contribution. Abstracting away the MRI-related technical details (we refer to original publication [93] for this), the task of CEST model parameter determination can be formulated in the following way.

For every brain voxel, the noisy raw spectrography data $\mathbf{x} \in \mathbb{R}^{55}$ (so called *Z-spectra* in CEST literature) is provided as input to the system that estimates parameters of a specific spectral line model; in this case, multiple Lorentzian models [278, 273] with 10 parameters in total $\mathbf{y} \in \mathbb{R}^{10}$ were used. Conventionally, this process is done in several steps that involve denoising [38] and least square fitting of the Lorentzian parameters [98]. This overall process regularly takes approximately 10 minutes per subject.

The technical contribution of the thesis in this project is defined by substituting the least square fitting procedure described above with a simple deep probabilistic model that directly regresses the Lorentzian parameters together with the corresponding uncertainty estimates (Figure 4.1). The compact deep model is comprised of two hidden fully connected layers of size 100, each followed by an exponential linear unit function [54] that regresses mean and variance of the 10-dimensional Gaussian distribution $p_\theta(\mathbf{x}) = \mathcal{N}(\mu_\theta(\mathbf{x}), \sigma_\theta(\mathbf{x}))$. Following the approach described in Section 2.3.1, the model is then trained to minimize negative log-likelihood (equation 2.8) of the parameters obtained with the conventional pipeline.

The network was trained on brain voxel data from 3 healthy human subjects (135,752 training samples in total). Importantly, to improve the quality of provided uncertainty estimates, the training data was augmented with samples that contained additive Gaussian noise of different magnitude, as well as shifted versions of Z-spectra.

Results. While being trained on only three healthy subjects, the network was able to robustly reproduce the results of conventional fitting on the unseen healthy subject, as well as brain tumor patient (please see the original paper [93] for extensive quantitative evaluation). Areas of increased uncertainty predicted by the model were also coherent with the actual area of high deviation from the ground truth parameters. Finally and probably most importantly, the overall deepCEST framework was able to achieve these results in only a fraction of time of the conventional least square fitting (1 second instead of 10 minutes), bringing down computational costs of obtaining CEST contrast maps in clinical environments.

4.2 Real-Time Trajectory Prediction in High Speed Robotics

Problem formulation. Dynamic high speed robotics tasks [82, 40, 182] often require accurate methods to forecast the future value of a physical quantity based on previous measurements while respecting the real-time constraints of the particular application. For example, to hit or catch a flying ball with a robotic system [182], we need to predict

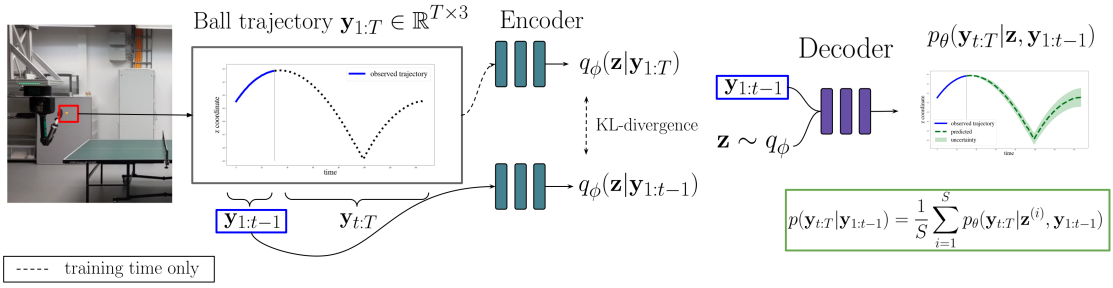


Figure 4.2: *Probabilistic ball trajectory prediction in robotic table tennis setup [100].* The observed part of trajectory is encoded with conditional variational framework (CVAE). Samples from the latent distribution are then passed through decoding network in order to generate distribution over future positions of the ball. The whole framework runs in real-time (8-10ms per prediction) and is integrated into dynamic high speed robotic system [99].

accurately and fast the trajectory of the ball based on the previous observations that are often noisy and might include outliers or missing data. Note that the time it takes to compute the predictions, called *latency*, is as important for the application as the accuracy of predictions. In our previous example, the prediction of the future ball positions are only useful if the computation time is significantly faster than the ball itself.

Both physics-based [284] and data-driven [35] models are used for trajectory forecasting. Physical models based on differential equations have been typically preferred to model and predict trajectories in high speed robotic systems [182] because they are relatively fast in prediction and are well-studied models known to provide reasonably good predictions for many problems. However, in some applications like pneumatic muscle robots [41], the best known physic-based models are not accurate enough to be useful for control. Even in cases where the physics is relatively well-known, estimating all the relevant variables to model the system can be difficult. In table tennis, for example, estimating the spin of the ball in real time from images is hard. In addition, small lens distortion on the vision system makes the position estimates not equally accurate in all the robot work space, rendering the estimation of the initial position and velocity less accurate. A data-driven approach, on the other hand, may have the potential to estimate the spin from its effect on the trajectory and ignore the lens distortion as long as it is present both at training and test time. However, popular data-driven methods for time series modeling like recurrent neural networks [197] and auto-regressive models [35] suffer from cumulative errors that render trajectory forecasting inaccurate as we predict farther into the future. In this section, we propose a novel method for trajectory prediction that mixes the power of deep learning and conditional generative models to provide a data-driven approach for accurate trajectory forecasting with the low latency required

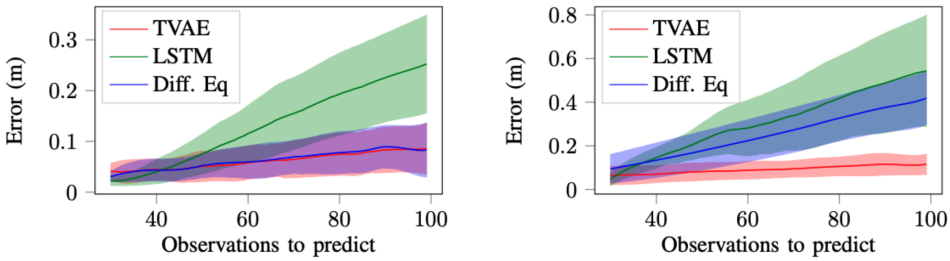


Figure 4.3: *Comparison between proposed trajectory prediction framework (TVAE), physics-based model [182] and LSTM [121] on **(a)** simulated and **(b)** real data*. Note that in the simulated data scenario our model performs as well as the differential equation-based prediction, which was the model used in simulation and therefore is the best we can get. On the real data, our model outperforms both the LSTM and differential equation models, specially as we predict farther into the future. Results from [100].

by real-time applications.

Technical contribution. The proposed model for ball trajectory prediction (developed together with the main author of [100]) is depicted in Figure 4.2. It closely resembles the conditional variational autoencoder model (CVAE) we have discussed in Section 3.5.2. Here, we provide the observed part of a trajectory $\mathbf{y}_{1:t-1}$ to a simple two-layer encoder network in order to estimate the parameters of a latent Gaussian distribution $q_\phi \sim \mathcal{N}(\mu_\theta(\mathbf{y}_{1:t-1}), \sigma_\theta(\mathbf{y}_{1:t-1}))$. During training, we also provide the full trajectory to the same encoder and aim to minimize the Kullback-Leibler divergence [149] between the two induced latent distributions $KL(q_\phi(\mathbf{z}|\mathbf{y}_{1:T})||q_\phi(\mathbf{z}|\mathbf{y}_{1:t-1}))$. Samples $\mathbf{z} \sim q_\phi$ from this distribution are then passed through a simple two-layer decoder together with the observed trajectory path in order to produce conditional density function over possible future trajectories $p_\theta(\mathbf{y}_{t:T}|\mathbf{z}, \mathbf{y}_{1:t-1})$. The final conditional density is then given by marginalizing over $\mathbf{z}$:

$$p(\mathbf{y}_{t:T}|\mathbf{y}_{1:t-1}) = \int p_\theta(\mathbf{y}_{t:T}|\mathbf{z}, \mathbf{y}_{1:t-1}) q_\phi(\mathbf{z}|\mathbf{y}_{1:t-1}) \mathrm{d}\mathbf{z}, \tag{4.1}$$

which can be approximated via Monte Carlo sampling:

$$p(\mathbf{y}_{t:T}|\mathbf{y}_{1:t-1}) = \frac{1}{S} \sum_{i=1}^{S} p_\theta(\mathbf{y}_{t:T}|\mathbf{z}^{(i)}, \mathbf{y}_{1:t-1}), \mathbf{z}^{(i)} \sim q_\phi(\mathbf{z}|\mathbf{y}_{1:t-1}). \tag{4.2}$$

Results. The proposed method to predict the trajectory of a table tennis ball was evaluated both in simulation and in the real table tennis robotic system. In case of real-

world scenario, the system was able to robustly learn the dynamics from a relatively small amount of observations (614 trajectories). The comparison with existing physics-based model [182] and alternative deep learning approaches (e.g. LSTMs [121]) was done both in terms of prediction accuracy and inference time. Overall, the developed temporal CVAE framework clearly outperformed these competing approaches (Figure 4.3). We refer to the original publication [100] for details on quantitative analysis. Most importantly, the low latency achieved by the final model (8-10ms per prediction) made it possible for the probabilistic trajectory predictor to be integrated in the real-time system that controls robotic arm movement [99].

Part II

Efficient 3D Shape Analysis

Chapter 5

Overview and Foundations

While modern physics offers multiple alternatives [255, 290], for the majority of practical purposes, the concept of representing our physical universe with the *three spatial dimensions* is still a dominating one. Sensing, analyzing and rendering 3D structures are the tasks of paramount importance in computer vision, with applications in robotics, medical imaging, fluid dynamics and computer graphics.

In this chapter, we will review some of the recent developments in the field of 3D shape analysis. Presenting a comprehensive history of the subject, however, goes beyond the scope of this thesis, and we will refer to the excellent textbooks and surveys available. Instead, we will focus on the persistent concepts and observe how recent developments in deep learning are embracing and reinterpreting classical methodologies.

This chapter is organized as follows. Section 5.1 introduces the main sources of 3D data. This will include brief description of the various acquisition techniques (lidars, infrared depth cameras), as well as stereo reconstruction pipelines. Next, we will discuss available representations, including raw point clouds and voxels, as well as implicit surface representations that recently attained renowned interest in the context of deep learning.

Finally, we will discuss 3D data visualisation methodologies and address *neural rendering*, the growing body of work that aims to combine or substitute classic rendering techniques with image generators based on neural networks.

5.1 3D Data Acquisition

There are numerous ways to extract 3D shape information of the real world structures [257]. Their applicability will significantly depend on the task at hand, available hardware and imaging conditions. Below we will simply outline several popular possibilities.

3D Reconstruction from multiple images. A large and diverse set of methods was developed to reconstruct 3D scenes from a collection of 2D images. *Mutli-view stereo (MVS)* algorithms [115, 83] take a set of photos and corresponding camera parameters in order to create a dense 3D model of the underlying scene. In order to achieve high quality reconstructions, these methods typically make certain assumptions, the crucial ones being *scene rigidity* and known camera parameters. In the most general scenario

(a) (b) (c)

Figure 5.1: *Sensing 3D world:* (a) multi-view stereo reconstruction [236]; (b) smartphone RGB-D images combined into a single 3D scene with [190]; (c) monocular depth estimation with the pre-trained neural network from [152].

of unorganized photo collections, *structure from motion (SfM)* methods aim to simultaneously find correspondences between 2D pixels, estimate cameras parameters and find the 3D coordinates in some canonical coordinate system [201]. Nowadays, these methods reached impressive scalability, where the whole cities can be reconstructed from a collection of unorganized Internet photos [246, 247, 10, 236]. The process of building a 3D map of environment together with determining the camera positions at each step is also known as *simultaneous localization and mapping (SLAM)* in robotics, where the emphasis is regularly put on the real-time performance and recurrent approach [44].

Nowadays, widely available software packages [236] combine structure from motion and multi-view stereo algorithms in one solid pipeline, allowing everyone to create high quality 3D models from a collection of smartphone photos (Figure 5.1a).

Depth sensors. Another approach for 3D reconstruction implies constructing specific hardware setups for depth sensing. One such approach is based on the *structured light* technique, where infrared pattern is being projected and captured by a dedicated module. Utilized in the camera systems like early Microsoft's Kinect, this depth sensing module allowed to augment every pixel of a standard RGB-camera images with the corresponding depth information. The set of obtained RGB-D images can then be combined into a single coherent scene via algorithms like KinectFusion [190].

One limitation of the most structured light depth sensors is a significant quality deterioration in the presence of a bright sunlight which limits their applicability and reliability in the open environments. In these scenarios, *time-of-flight* based depth sensors are regularly utilized. The measurement principle is based on emitting light signal that is, being reflected from the surface and captured back by the system, provides the information on the distance travelled via the time difference between sending and receiving. *Lidar*

(light detection and ranging) represent a popular example of this technology widely used in critical applications like autonomous driving. While being robust in the challenging open environments, the drawbacks of lidar sensors include high cost and relatively sparse depth output.

Figure 5.1b shows reconstruction of the indoor scene with the structured light of a built-in RGB-D camera of a modern smartphone.

Monocular depth and 3D estimation. Finally, with the recent advances of deep learning, there have been also a significant progress in inferring depth maps and 3D shapes directly from RGB images [29, 283]. While initial methods [97, 81, 285] have been heavily tailored to specific final tasks and datasets like KITTI [88], recent works provide a high level of generalization across wide range of natural scenes [152].

Figure 5.1c shows the depth map obtained via running a pre-trained model of [152] on the captured indoor photo.

Using various shape priors can dramatically improve the accuracy of 3D reconstruction. E.g., using deformable 3D human models [166] as a proxy allows us to accurately estimate 3D shape and pose of a human subject from a single image [131].

5.2 Shape Representations

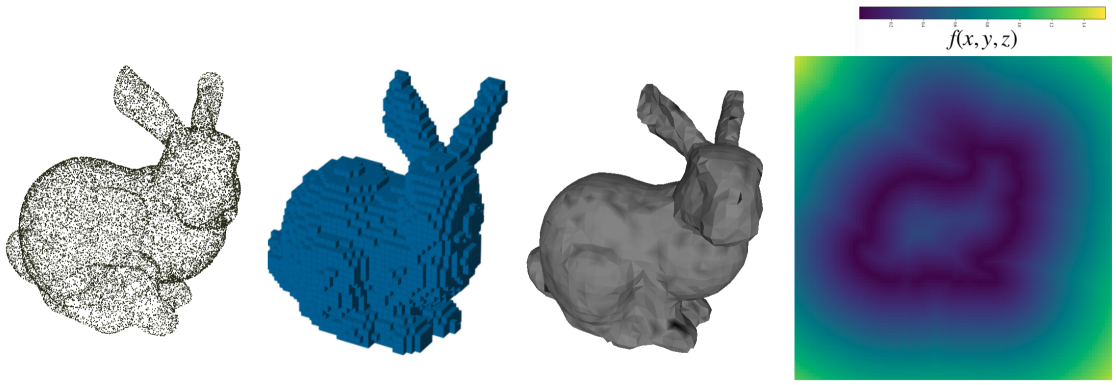
Another fundamental question arising during all the steps of 3D modeling is the representation that will best suit the application purposes. Below we will briefly review the most common 3D data structures.

Point clouds. The natural output of the most multi-view stereo reconstruction algorithms is a collection of 3D points, or a *point cloud* (Figure 5.2a). Every point in the cloud have the information regarding its 3D coordinates in some canonical common space; additionally, the information about point color, surface normal and material properties could be stored.

While providing compact yet detailed representation of a scene, point clouds are inherently unstructured data. Therefore, specific neural network designs are regularly needed that can efficiently handle order-invariant inputs [218, 220, 277]. This also motivates the need for the generic efficient point cloud encoding technique which we will address in Chapter 6. We also refer to [111] for a recent comprehensive survey on deep learning on point clouds.

Voxels. One possible solution to the problem of unorganized point clouds is the process of *voxelization*. In the simplest scenario of binary occupancy voxels, the 3D space is partitioned into regular cubes of a fixed volume, and each voxel is marked as occupied if the some point from the cloud lies inside it (Figure 5.2b). Additionally, we can store the information about the original point inside each voxel, as well as the distance from the point to the center of a voxel. In the latter case, this representation is regularly referred to as *truncated distance field (TDF)*.

Voxel representation allows us to convert unordered point clouds into regular grid data



(a) (b) (c) (d)

Figure 5.2: *Existing 3D representations:* (a) *point clouds*, a set of unordered 3D points; (b) *occupancy voxels*, regular grid data; (c) *mesh polygons*, surface as a set of surface primitives; (d) *implicit functions*, surface as the solution to equation $f(x,y,z) = 0$.

and apply 3D versions of convolutional operations, one of the key ingredients of modern deep neural networks. However, this tessellation results in a cubical growth of memory requirements. At the same time, most voxels remain empty which results in rather sparse inputs to learning algorithms.

A number of network architectures have been proposed that process voxel-based representations, with applications including shape classification, segmentation and generation [177, 219, 181, 224, 133]. One of the interesting directions is substituting simple occupancy flags and truncated distances stored inside each voxel with rich features, including trainable deep features [243, 164].

Meshes. In the field of computer graphics, probably the most utilized surface representation is *polygon mesh* which represents a shape with a set of 3D *vertices* V and surface primitives, or *mesh faces* F:

$$V = \{\mathbf{v_1}, \ldots, \mathbf{v_n}\}, \mathbf{v_k} \in \mathbb{R}^3, \tag{5.1}$$

$$F = \{(\mathbf{v_{i_1}}, \mathbf{v_{i_2}}, \mathbf{v_{i_3}})\}, \mathbf{v_{i_j}} \in V, \tag{5.2}$$

where each face $f_i \in F$ defines a triangle formed by vertices $(\mathbf{v_{i_1}}, \mathbf{v_{i_2}}, \mathbf{v_{i_3}})$.

Mesh model can be obtained from a raw point cloud with the algorithms like Poisson reconstruction [138]. It is a regular final step in 3D reconstruction pipelines which allows the usage of the obtained captured scene in various computer graphics applications.

Mesh is also a popular representation of choice in the field of deformable 3D modeling [32, 17, 129]. High-fidelity mesh-based models exist for human bodies [165], faces [157] and hands [228], as well as various animals [289, 131]. However, in order utilize

the expressive power of these models (i.e. to be able to change pose and shape of the captured body), we regularly need to bring the captured mesh in correspondence with some canonical mesh template defined by the deformable 3D model. This process is known as *registration* and is traditionally addressed with some iterative optimization procedures [33, 120]. In the next chapter, we will show how the developed novel basis point set representation can be utilized to regress human mesh model [165] directly from the noisy point clouds.

Multiple deep learning methods exist that infer meshes directly from 2D images [131, 113, 92] or build generative models of mesh data [222, 258].

Implicit 3D representations. Finally, the 3D surface S can also be represented *implicitly*, e.g. by defining a corresponding *level set function*:

$$f(x,y,z) : \mathbb{R}^3 \rightarrow \mathbb{R}, \tag{5.3}$$

$$(x,y,z) \in S \Leftrightarrow f(x,y,z) = 0. \tag{5.4}$$

Originating from physics research [199], these implicit surface representations have soon proved to be useful in many other areas, including computer vision and graphics [237]. In the recent years, they have also gained a significant new wave of interest from deep learning practitioners [179, 202, 232, 244]. The main idea beyond this growing body of work on *deep implicit functions* is to model function (5.3) (or similar) with a deep neural network $f_\theta(x,y,z)$. The capacity and architecture of the network will then define the fidelity of the obtained representation. This paradigm have already been proved useful in various 3D modeling tasks, especially the ones that include analysis and synthesis of complex topologies with large amount of high frequency details [232, 52, 107, 233].

5.3 Rendering

One of the core tasks of 3D modeling is to provide means of efficient geometry visualization. Computer graphics find its applications in diverse set of fields, including various media content generation, virtual reality, physics and robotics simulation environments and architecture design and planning.

Mesh-based rasterization. Probably the most widely applied and developed method of geometry rendering is based on polygon mesh models discussed in the previous section. In this scenario, mesh polygons (5.1-5.2) are projected into 2D camera plane and their visible areas are efficiently determined by a process commonly known as *rasterization*. Importantly, algorithms like *z-buffer* [262] and *back face culling* [70] will dramatically minimize the amount of computation by keeping track of visible surfaces. Next, *texture maps* are used that will assign RGB color values to each point in each triangle polygon. This, together with material properties and information about surface normals, will then define how the final pixel colors should change based on the light source lo-

(a) (b) (c)

Figure 5.3: *Demonstration of the state-of-the-art rendering systems:* (a) Unreal Engine 5 rendering pipeline [1]; (b) NVIDIA RTX real time ray tracing [2]; (c) synthetic face generated with StyleGAN2 network [135].

cation via deterministic shading function. Additionally, *displacement maps* and other features can be introduced to model fine details and effects of the rendered 3D scene. We refer to excellent textbooks on modern computer graphics approaches for more information [209, 11].

Nowadays, efficient rasterization algorithms and availability of powerful graphics processing units allow impressive real time visualizations of complex dynamic scenes. Figure 5.3a shows a screenshot from one of the latest versions of the popular Unreal Engine rendering framework.

Voxels and points can also be used as rendering primitives instead of polygon meshes, e.g. when computational efficiency or modeling high frequency fine details are in focus [145, 108, 57].

Ray tracing. One of the main limitations of conventional polygon-based rasterization discussed above is the shading model that assumes a single interaction between surface and light source. *Ray tracing* group of techniques aim to overcome this by physically based modeling of the full light ray path as it travels from the light source to the target camera pixel. This allows to model realistic effects like multiple light reflections (Figure 3b).

Ray tracing is considerably more demanding computationally compared to the mesh-based rasterization techniques; initially, it was mostly used to generate high quality images or video sequences in the offline regime, e.g. for the cinema production or architecture projects. However, this is also changing rapidly thanks to the increased GPU performance and improved rendering algorithms: recently, NVIDIA have released first real-time ray tracing technology with its RTX platform (Figure 5.3b).

[1] `https://youtu.be/qC5KtatMcUw`
[2] `https://youtu.be/pNmhJx8yPLk`

Neural rendering. Despite the tremendous progress over last decades, classic rendering techniques discussed above face fundamental limitations due to their heavy reliance on the quality and resolution of underlying geometry primitives, as well as deterministic rendering procedures. An example of this can be seen in the Figure 3a, where the realism of high frequency, thin 3D structures like hair of the character is hindered by the smooth geometry used for modeling.

At the same time, deep learning methods like [134] have shown impressive capabilities in generating photorealistic images with high quality textures (Figure 5.3c). While initial methods based on *generative adversarial networks (GANs)* had rather limited control over the synthesis, large corpus of recent works have explored the ways to additionally condition and constrain the process [127, 271]. This includes using scene sketches [46], segmentation maps [127, 271] and full initial mesh renders as inputs [175], as well as enforcing some form of 3D geometry consistency across multiple generated images of the same scene [191]. We refer to [260] for a comprehensive survey on the recent developments in neural rendering.

In Chapter 7, we will review the last group of methods in greater details and present a new technique that combines efficient point-based graphics with the follow-up image enhancement done by the neural renderer.

5.4 Conclusion

In the present chapter we have briefly recapped the fundamental steps of 3D modeling. We have discussed several 3D representation schemes and observed their potential advantages and limitations. In the next chapter, we will present *basis point set encoding*, simple and efficient 3D representation that combines fine geometry preservation provided by raw point clouds with the advantage of having compact fixed-length vector descriptor of a scene that can be provided to any machine learning algorithm as input.

We have also discussed state-of-the-art approaches in photorealistic rendering. In Chapter 7, we will continue this discussion with a strong focus on neural approaches for human model rendering. We will present *SMPLpix*, new rendering technique that combines classic deformable 3D models like SMPL [166] with the efficient image-to-image neural translation models, allowing us to create controllable photorealistic human avatars.

Chapter 6

Point Cloud Analysis with Basis Point Sets

6.1 Introduction

Point cloud data is becoming more ubiquitous than ever: anyone can create a point cloud from a set of photos with easy-to-use photogrammetry software or capture a point cloud directly with one of many consumer-grade depth sensors available worldwide. These sensors will soon be used in most aspects of our daily lives, with autonomous cars recording streets and city environments and VR and AR devices recording our home environment on a regular basis. The resulting data represents a great opportunity for computer vision research: it complements image data with depth information and opens up new fields of research.

However, point cloud data itself is unstructured. This leads to a variety of problems. First, point clouds have no fixed cardinality, i.e. their size varies depending on the recorded scene. They are also not 'registered' in the sense that it is not trivial to find correspondences between points across recordings of the same or of a similar scene. Point clouds have no notion of neighborhood. This means that it is not clear how convolutions, one of the critical operations in deep learning, should be performed.

In this chapter, we present a novel solution to the aforementioned problems, in particular the varying cloud cardinality. For an illustration, see Figure 7.1. We propose to encode point clouds as minimal distances to a fixed set of points, which we refer to as *basis point set*. This representation is vastly more efficient than classic extensive occupancy grids: it reduces every point cloud to a relatively small fixed-length vector. The vector length can be adjusted to meet computational constraints for specific applications and represents a trade-off between fidelity of the encoding and computational efficiency. Compared to other encodings of point clouds, the proposed representation also has an advantage in being more efficient with the number of values needed to preserve high frequency information of surfaces.

Given its fixed length, the presented encoding can be used with most of the standard machine learning techniques. In this paper, we apply mostly artificial neural networks to build models with it due to their popularity and accuracy. In particular, we analyze

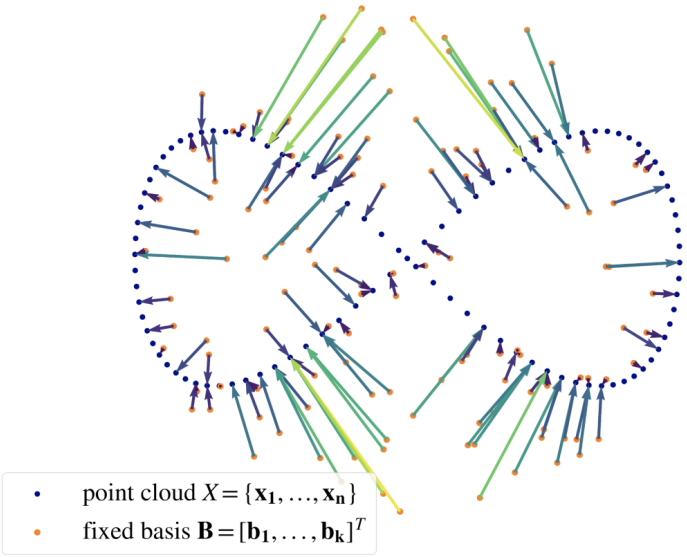


Figure 6.1: *Basis point set encoding for point clouds*. The encoding of a point cloud $X = \{\mathbf{x_1}, \ldots, \mathbf{x_n}\}$ is a fixed-length feature vector, computed as the minimal distances to a fixed set of points $\mathbf{B} = [\mathbf{b_1}, ..., \mathbf{b_k}]^T$. This representation can be used as input to arbitrary machine learning methods, in particular it can be used as input for off-the-shelf neural networks. This leads to substantial performance gains as compared to occupancy grid encoding or specialized neural network architectures without sacrificing the accuracy of predictions.

the performance of the encoding in two applications: point cloud classification and mesh registration over noisy 3D scans (Figure 6.2).

For point cloud classification, we achieve the same accuracy on the ModelNet40 [274] shape classification benchmark as PointNet [218], while using an order of magnitude less parameters and *three orders of magnitudes less floating point operations*. To demonstrate the versatility of the encoding, we show how it can be used for the task of mesh registration. We use the encoded vectors as input to a neural network that directly predicts mesh vertex positions. While showing competitive performance to the state-of-the-art methods on the FAUST dataset [33], the main advantage of our method is the ability to produce an aligned high resolution mesh from a noisy scan in a single feed-forward pass. This can be executed in real time even on a non-GPU laptop computer, requiring no additional post-processing steps. We make our code for both presented tasks available, as well as a library for usage in other projects[1].

[1] `https://github.com/sergeyprokudin/bps`

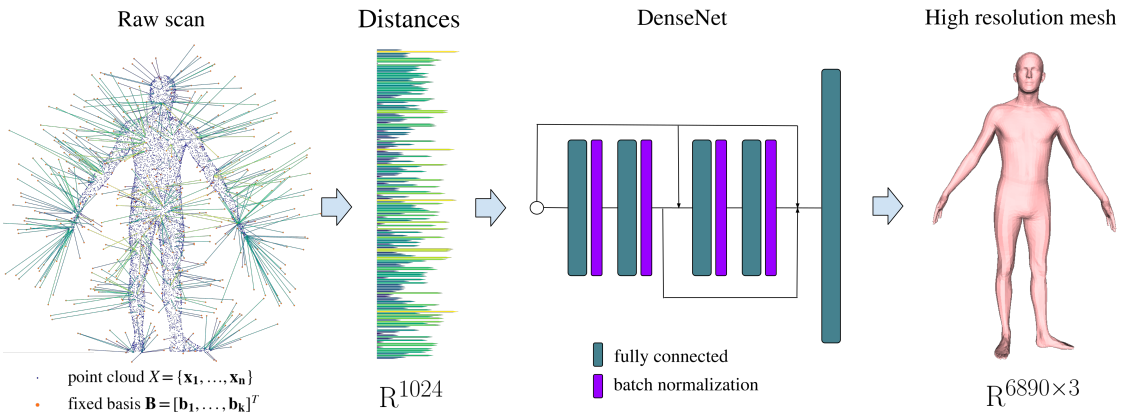


Figure 6.2: *Overview of our proposed model for the task of mesh registration to a noisy scan.* The computed minimal distances to the selected basis point set are provided as input to a simple dense network with two blocks of two fully connected layers. The model directly predicts mesh vertex positions, with a forward pass taking less than 1ms. We also propose a model for shape classification; see Section 6.5 for details.

6.2 Related Work

In this section, we continue the discussion of existing 3D data representations and models and put them in relation to the presented method. We focus on representations that are compatible with deep learning models, due to their high performance on a variety of 3D shape analysis tasks.

Point clouds. Numerous methods [218, 220, 238, 277, 156] were proposed that process 3D point clouds directly, amongst which the PointNet family of models gained the most popularity. This approach processes each point separately with a small neural network followed by an aggregation step with a pooling operation to reason about the whole point cloud. Similar pooling-based approaches for achieving feature invariance on general unordered sets were proposed in other works as well [277]. Other methods working directly on point clouds organize the data in kd-trees and other graphs [144, 84, 151]. These structures define a neighborhood and thus convolution operations can be applied. Vice versa, specific convolutional filters can be designed for sparse 3d data [259, 238].

We borrow several ideas from these works, such as using kNN-methods for searching efficiently through local neighborhoods or achieving order invariance through the use of pooling operations over computed distances to basis points. However, we believe that the proposed encoding and model architectures offer two main advantages over existing point cloud networks: (a) higher computational efficiency and (b) conceptually simpler, easy-to-implement algorithms that do not rely on a specific network architecture or require custom neural network layers.

Occupancy grids. Similar to pixels for 2D images, occupancy grid is a natural way of

encoding 3D information. Numerous deep models were proposed that work with occupancy grids as inputs [177, 219, 181]. However, the main disadvantage of this encoding is its cubic complexity. This results in a high amount of data needed to accurately represent the surface. Even relatively large grids by our current memory standards ($128^3, 256^3$) are not sufficient for an accurate representation of high frequency surfaces like human bodies. At the same time, this type of voxelization results in very sparse volumes when used to represent 3D surfaces: most of the volume measurements are zeros. This makes this representation an inefficient surface descriptor in multiple ways. A number of methods was proposed to overcome this problem [270, 224]. However, the problem of representing high frequency details remains, together with a large memory footprint and low computational efficiency for running convolutions.

Signed distance fields. Truncated signed distance fields (TSDFs) [58, 190, 225, 250, 280, 60, 202] can be viewed as a natural extension of occupancy grids: they store distance-to-surface information in grid cells instead of a simple occupancy flag. While this partially resolves the problem of representing surface information, the cubic requirement for memory and the low computational efficiency for convolutions remains. In comparison, our method can be viewed as one that uses an arbitrary subset of points from the distance field. The crucial difference is that the distance field we sample from is unsigned and non-truncated, and the number of samples is proportional to the number of points in the original cloud. We further investigate the connection between occupancy grids, TSDFs and BPS in Section 6.4.1.

2D projections. Another common strategy is to project 3D shapes to 2D surfaces and then apply standard frameworks for 2D input processing. This includes depth maps [274], height maps [234], as well as a variety of multi-view models [253, 132, 76]. Closely related are approaches that project 3D shapes into spheres and apply spherical convolutions to achieve rotational invariance [73, 55]. While projection-based approaches show high accuracy in discriminative tasks (classification, shape retrieval), they are fundamentally limited in representing shapes that have multiple 'folds', invisible from external views. In comparison, our encoding scheme can accurately preserve surface information of objects with arbitrary topology as we show in our experiments in Section 6.4.

We now describe the algorithm for constructing the proposed basis point representation from a given point cloud.

6.3 Method

Normalization. The presented encoding algorithm takes a set of point clouds as input $\mathbf{X} = \{X_i, i = 1, \ldots, p\}$. Every point cloud can have a different number of points n_i:

$$X_i = \{\mathbf{x}_{i1}, \ldots, \mathbf{x}_{in_i}\}, \mathbf{x}_{ij} \in \mathbb{R}^d, \tag{6.1}$$

where $d = 3$ for the case of 3D point clouds. In first step, we normalize all point clouds to a fit a unit ball:

$$\mathbf{x}_{ij} = \frac{\mathbf{x}_{ij} - \mathbb{E}_{\mathbf{x}_{ij} \sim X_i} \mathbf{x}_{ij}}{\max_{\mathbf{x}_{ij} \in X_i} \|\mathbf{x}_{ij} - \mathbb{E}_{\mathbf{x}_{ij} \sim X_i} \mathbf{x}_{ij}\|}, \forall i, j. \tag{6.2}$$

***BPS* construction.** Next, we form a *basis point set*. For this task, we sample k random points from a ball of a given radius r:

$$\mathbf{B} = [\mathbf{b}_1, ..., \mathbf{b}_k]^T, \mathbf{b}_{j \in} \mathbb{R}^d, \|\mathbf{b}_j\| <= r. \tag{6.3}$$

It is important to mention that this set is arbitrary but fixed for all point clouds in the dataset. r and k are hyperparameters of the method, and k can be used to determine the trade-off between computational complexity and the fidelity of the representation.

Feature calculation. Next, we form a feature vector for every point cloud in a dataset by computing the minimal distance from every basis point to the nearest point in the point cloud under consideration:

$$\mathbf{x}_i^{\mathbf{B}} = [\min_{\mathbf{x}_{ij} \in X_i} \mathrm{d}(\mathbf{b}_1, \mathbf{x}_{ij}), \ldots, \min_{\mathbf{x}_{ij} \in X_i} \mathrm{d}(\mathbf{b}_k, \mathbf{x}_{ij})]^T, \mathbf{x}_i^{\mathbf{B}} \in \mathbb{R}^k. \tag{6.4}$$

Alternatively, it is possible to store the full directional information in the form of delta vectors from each basis point to the nearest point in the original point cloud:

$$\mathbf{X}_i^{\mathbf{B}} = \left\{ \left(\operatorname*{argmin}_{\mathbf{x}_{ij} \in X_i} \mathrm{d}(\mathbf{b}_q, \mathbf{x}_{ij}) - \mathbf{b}_q \right) \right\} \in \mathbb{R}^{k \times d}. \tag{6.5}$$

Other information about nearest points (e.g., RGB values, surface normals) can be saved as part of this fixed representation. The feature computation is illustrated in Figure 7.1. The formulas (6.4) and (6.5) give us fixed-length representations of the point clouds that can be readily used as input for learning algorithms.

BPS selection strategies. We investigate a number of basis point selection strategies and provide details of these experiments in Section 6.4.2. Overall, random sampling from a uniform distribution in the unit ball provides a good trade-off between efficiency, universality of the generation process and surface reconstruction results, and we apply it throughout the experiments in this paper. Alternatively, an extensive 3D grid of basis points could be used in tandem with any existing 3D convolutional neural network in order to achieve maximum performance at the cost of increased computational complexity.

Complexity. In this work, we use Euclidean distances between points for creating our encoding, but other metrics could be used in principle. Since we are working with

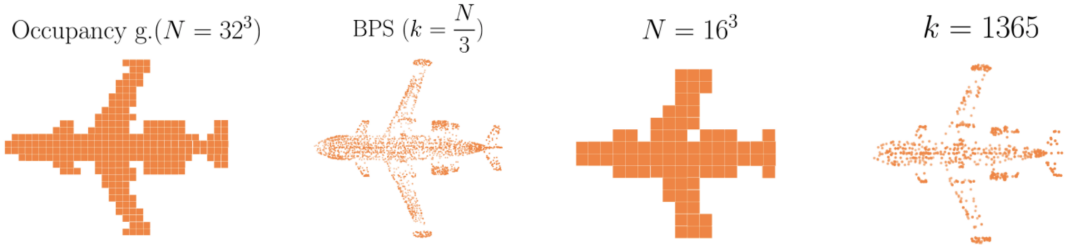


Figure 6.3: *Surface encoding with occupancy grids (a,c) and basis point sets (b,d).* With the same length of encoding N our method can capture surface details more accurately. Even when using only $k \approx 10^3$ basis points, our method can capture details of a surface (d).

3D point clouds (which corresponds to having a small value for d), the nearest neighbor search can be made efficient by using data structures like ball trees [196]. Asymptotically, $O(n \log n)$ operations are needed for constructing a ball tree from the point cloud X_i and $O(k \log n)$ operations are needed to run nearest neighbor queries for k basis points. This leads to an overall encoding complexity of $O(n \log n + k \log n)$ per point cloud. The kNN search step can be also efficiently implemented as part of an end-to-end deep learning pipeline [130]. Practically, we benchmark our encoding scheme for different values of n and k and show real-time encoding performance for values interesting for current real world applications. We will investigate practical encoding performance in Section 6.5.4.

6.4 Analysis

6.4.1 Comparison to Occupancy Grids, TSDFs and Plain Point Clouds

Informal intuition. Compared to occupancy grids and TSDFs, the efficiency and superiority of the proposed BPS encoding is based on two key observations. First, it is beneficial for both surface reconstruction and learning to *store some continuous global information* (e.g., Euclidean distance to the nearest point) in every cell of the grid instead of simple binary flags or local distances. In the latter case, most of the voxels remain empty and, moreover, the feature vector will change dramatically when slight translations or rotations are applied to an object. In comparison, every BPS cell always stores some information about the encoded object and the feature vector changes smoothly with respect to affine transformations. From this also stems the second important observation:

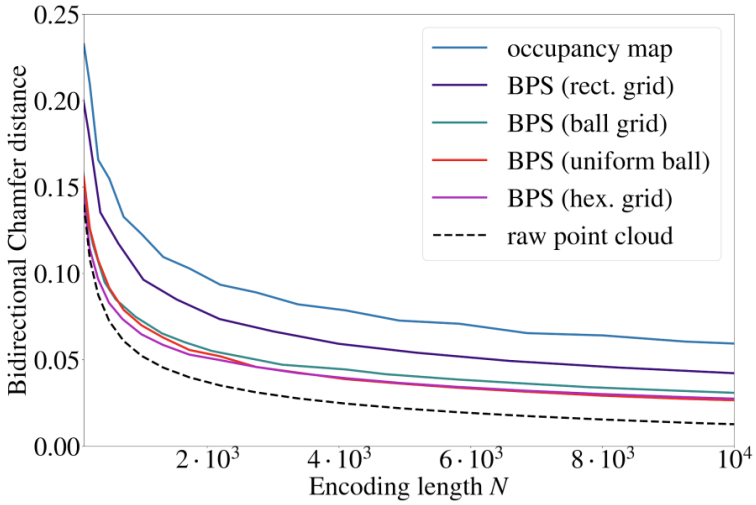


Figure 6.4: *Surface reconstruction quality vs. encoding length for different 3D data encoding methods.* We measured the Chamfer distance on 10^3 encoded and reconstructed random shapes from the ModelNet40 dataset. The suggested representation is more accurate in representing surface details than standard occupancy grid. The performance of our best basis selection methods is close to encoding the surface with subsampled unordered point clouds while being a fixed-length representation that can be directly used with a wide range of machine learning algorithms. See Section 6.4 for further details.

when every cell stores some global information, we can use a much smaller number of them in order to represent the shape accurately, thus *avoiding the cubical complexity of the extensive grid representation*. This can be seen in Figure 7.1 and Figure 6.3d, where $k \approx n$ basis points are able to capture the outline of the original cloud.

We will now validate this intuition by comparing the aforementioned representations in terms of surface reconstruction and actual learning capabilities.

Surface reconstruction experiments. Independent of a certain point cloud at hand, how well does the encoding captures details of an object? To answer this question, we take 10^3 random CAD models from the ModelNet40 [274] dataset and construct synthetic point clouds by sampling 10^4 points from each surface. We compare three approaches of encoding the resulting point clouds: storing them as is (raw point cloud), occupancy grid and the proposed encoding via basis point sets as suggested in Equation 6.5.

For all methods we define a fixed allowed description length N (as N floating point values) and compare the normalized bidirectional Chamfer distance between the original point cloud X and the reconstructed point cloud X^r for the different encodings:

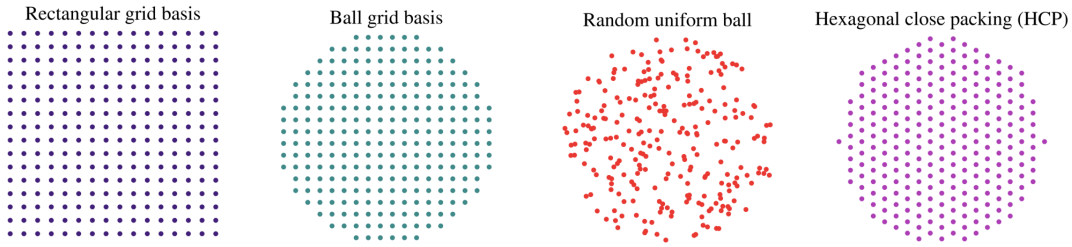


Figure 6.5: *Different basis point selection strategies.* See Section 6.4.2 for details. In this work, we mainly use random uniform ball sampling for its simplicity and efficiency, as well as rectangular grid basis that allows us to apply 3D convolutions in a straightforward manner. Different BPS arrangements allow the usage of different types of convolutions.

$$\mathrm{d}_{CD}(X, X^r) = \frac{1}{|X|} \sum_{\mathbf{x}_i \in X} \min_{\mathbf{x^r}_i \in X^r} ||\mathbf{x}_i - \mathbf{x^r}_i||^2 + \frac{1}{|X^r|} \sum_{\mathbf{x^r}_i \in X^r} \min_{\mathbf{x}_i \in X} ||\mathbf{x}_i - \mathbf{x^r}_i||^2. \tag{6.6}$$

With the same length of the description N we can either store $N/3$ points from the original point cloud, $\sqrt[3]{N} \times \sqrt[3]{N} \times \sqrt[3]{N}$ binary occupancy flags or $N/3$ basis points with the matrix $\mathbf{X}_i^{\mathbf{B}}$ defined in Equation 6.5. From this matrix, a subset of original points can be reconstructed by simply adding corresponding basis point coordinates to every delta vector. For the occupancy grid encoding, we use the centers of occupied grid cells; please note that though a full floating point representation is not necessary to store the binary flag, in reality the majority of machine learning methods will work with floating point encoded occupancy grids and we assume this representation.

Figure 6.4 shows the encoding length and the reconstruction quality measured as Chamfer distance (, Equation 6.6). The proposed encoding produces less than half of the encoding error compared to occupancy grids for point clouds up to roughly 10^4 points (see Figure 6.3 for a qualitative comparison). This is an indicator for its superiority for preserving shape information. The error curve for the basis point sets is close to the one of the subsampled point cloud representation. The basis point set representation is less accurate than the raw point cloud since the resulting extracted points are not necessarily unique. However, the basis point set is an ordered, fixed-length vector encoding well-suited to apply machine learning methods.

6.4.2 Basis Point Selection Strategies

We investigate four different variants of selecting basis points visualized in Figure 6.5.

Rectangular grid basis. A basic approach to basis set construction is to simply ar-

range points on a rectangular $[-1,1]^3$ grid. In that case, the basis point set representation resembles the truncated signed distance field [58] representation. However, one important difference is that we do not truncate the distances for far-away basis points, allowing every point in the set to store some information about the object surface. We will show in Section 6.5.1 that this small conceptual difference has an important effect on performance. We are also allowing the full directional information to be stored in the cell as defined in Equation 6.5. Finally, BPS does not require the point clouds to be converted into watertight surfaces since *unsigned* distances are used.

Ball grid basis. Since all point clouds are normalized to fit in the unit ball by the transformation defined in Equation 6.2, the basis points at the corners of the rectangular grid are located far away from the point cloud. These corner points in fact constitute 47.6% of all the samples (this can be derived by comparing the volume ratio of a unit ball to a unit cube). Hence we can potentially improve our sampling efficiency by simply trimming the corners of the grid and using more sampling locations within the unit ball.

Random uniform ball sampling. One generic simple strategy to select points lying inside a $d-$dimensional ball is uniform sampling. This can be done by either rejection sampling from a $d-$dimensional cube or other efficient methods that are summarized in [114].

Hexagonal close packing (HCP). We also experiment with *hexagonal close packing* [56] of basis points. Informal intuition behind this point selection strategy is that it will optimally cover the unit ball with equally sized balls centered at the basis points [112].

We show the comparison of reconstruction errors of 10^3 ModelNet objects using the different sampling strategies in Figure 6.4. Overall, the random uniform and HCP selection strategies provide the best reconstruction results. Using regular grids opens up possibilities for applying convolution operations and adds the possibility to learn translation and rotation invariant features.

We now evaluate the different encodings and basis point selection strategies with respect to their applicability with machine learning algorithms.

6.5 Learning with Basis Point Sets

6.5.1 3D Shape Classification

One of the classic tasks to perform on point clouds is classification. We present results for this task on the *ModelNet40* [274] dataset. We benchmark several deep learning architectures that use the proposed point cloud representation and compare them to existing methods that use alternative encodings. The dataset consists of $12 \cdot 10^3$ CAD models from 40 different categories, of which $9.8 \cdot 10^3$ are used for training. We use the same procedure for obtaining point clouds from CAD models as in [218], i.e., we

id	Method	acc.	FLOPs	params
1	VoxNet [177]	83.0%	$> 10^8$	9.0×10^5
2	Occ-MLP (32^3 grid)	$79.9\% \pm 0.3$	3.4×10^7	1.7×10^7
3	Occ-MLP (8^3 grid)	$74.5\% \pm 0.2$	1.1×10^6	5.5×10^5
4	TDF-MLP (32^3 grid)	$80.0\% \pm 0.3$	3.4×10^7	1.7×10^7
5	TDF-MLP (8^3 grid)	$75.9\% \pm 0.3$	1.1×10^6	5.5×10^5
6	BPS-MLP (32^3 grid)	$88.3\% \pm 0.2$	3.4×10^7	1.7×10^7
7	BPS-MLP (8^3 grid)	$87.6\% \pm 0.3$	1.1×10^6	5.5×10^5
8	BPS-MLP (8^3 ball)	$87.7\% \pm 0.3$	1.1×10^6	5.5×10^5
9	BPS-MLP (8^3 rand)	$88.0\% \pm 0.3$	1.1×10^6	5.5×10^5
10	BPS-MLP (8^3 HCP)	$88.1\% \pm 0.3$	1.1×10^6	5.5×10^5
11	BPS-Conv3D (32^3 grid)	$89.8\% \pm 0.2$	3.5×10^8	1.7×10^7
12	$9 \rightarrow$ direct. vect.	$86.2\% \pm 0.3$	2.2×10^6	1.1×10^6
13	$11 \rightarrow$ direct. vect.	$90.8\% \pm 0.3$	3.8×10^8	1.7×10^7
14	BPS-ERT [90] (16^3 g.)	$85.4\% \pm 0.2$	N/A	N/A
15	BPS-XGBoost (32^3 g.)	$86.1\% \pm 0.1$	N/A	N/A

Table 6.1: *Comparison between occupancy grids, truncated distance fields (TDF) and BPS as input features for 3D shape classification on the ModelNet40 [274] challenge.* We keep the model architecture fixed across experiments. Global BPS encoding significantly outperforms its local counterparts. See Section 6.5.1 for further details.

sample $n = 2048$ points from mesh faces, followed by the normalization process defined in Equation (6.2).

Comparison to occupancy grids and VoxNet. To show the superiority of BPS features and to disambiguate contributions, i.e. the BPS encoding itself and the proposed network architectures, we fix a simple generic MLP architecture with 2 blocks of [fully-connected, relu, batchnorm, dropout] layers and perform training with 32^3 rectangular grids of occupancy maps, truncated distance fields (TDFs) and BPS as inputs.

Results are summarized in Table 6.1, rows 1-7. Using global distances as features instead of occupancy flags with the same network clearly improves accuracy, outperforming an architecture that was specifically designed for processing this type of input: VoxNet [177] (row 1). TDFs store only local distances within the grid cell and suffer from the same locality problem as voxels (r. 4). It is also important to note that reducing the grid size affects these methods dramatically (rows 3 and 5, 5% drop in accuracy), while the effect on the BPS is marginal (r. 6, -0.7%).

We also compare different BPS selection strategies in the rows 7-10 of Table 6.1. In the absence of network operators exploiting the point ordering (e.g. 3D convolutions), random and HCP strategies give a slight boost in performance. When the point order in a rectangular BPS grid is exploited with 3D convolutional deep learning models like VoxNet, performance improves at the cost of increased computational complexity (ap-

Method	acc.	FLOPs	params
RotationNet 20x [132]	**97.37**%	$>10^9$	5.8×10^7
MVCNN 80x [253]	90.1%	6.2×10^{10}	9.9×10^7
VoxNet [177]	83.0%	$>10^8$	9.0×10^5
Spherical CNNs [73]	88.9%	2.9×10^7	5.0×10^5
point cloud based methods:			
KD-networks [144]	**91.8**%	$>10^9$	$>10^7$
KCNet [238]	91.0%	$>10^8$	9.0×10^5
SO-Net [156]	90.9%	$>10^8$	$>10^6$
DeepSets [277]	90.0%	1.5×10^9	2.1×10^5
PointNet++ [220]	90.7%	1.6×10^9	1.7×10^6
PointNet [218]	89.3%	4.4×10^8	3.5×10^6
PointNet(vanilla) [218]	87.2%	1.4×10^8	8.0×10^5
DeepSets (micro) [277]	82.0%	3.8×10^7	2.1×10^5
Ours (BPS-MLP)	89.0%	$\mathbf{7.6 \times 10^5}$	3.8×10^5
Ours (BPS-Conv3D)	90.8%	3.5×10^8	4.4×10^6
Ours (BPS-Conv3D, 10x)	91.6%	3.5×10^9	4.4×10^7

Table 6.2: *Results on the ModelNet40 [274] 3D shape classification challenge.* Simple fully connected network can be trained on BPS features in several minutes on a single GPU to reach the performance of PointNet.

proximately two orders of magnitude more flops, Table 6.1, r. 11).

Substituting Euclidean distances with full directional information defined by Equation (6.5) negatively affects the performance of a plain fully-connected network (Table 6.1, r.12) whereas it improves the performance of a 3D convolutional model (Table 6.1, r. 13).

To show the versatility of the proposed representation, we also use the same BPS features as input to an ensemble of extremely randomized trees (ERT [90]) and XGBoost [50] frameworks.

Comparison to other methods. Finally, we combine these findings with other enhancements (e.g., augmenting the data with few fixed rotations, improving learning schedule and regularization - please refer to the supplementary material and corresponding code repository for further details) and compare our two best-performing models to other methods in Table 6.2.

In summary, simple fully connected network, trained on BPS features in several minutes on a single GPU, is reaching the performance of PointNet [218], one of the most widely used networks for point cloud analysis. 3D-convolutional model trained on BPS

Method	Inference time (CPU)	Inference time (GPU)	Model size
KD-networks [144]	130ms	41ms	150MB
PointNet++ [220]	-	22ms	5.8MB
PointNet [218]	-	8ms	3.2MB
DeepSets [277]	16ms	0.5ms	**0.8MB**
Ours (BPS-MLP)	**0.04ms**	**0.04ms**	1.6MB
Ours (BPS-Conv3D)	56ms	2ms	18MB

Table 6.3: *Inference time for different models.* Our fully connected model achieve sub-millisecond inference time on both, CPU and GPU.

rectangular grid is matching the performance of the PointNet++[220], while still being computationally more efficient. Finally, crude ensembling of 10 such models allows us to match state-of-the-art performance [144] among methods working only on point clouds as inputs (e.g., without using surface normals that are available in CAD models but rarely in real-world scenarios).

Space and time complexity. Apart from the FLOPs comparison on the ModelNet40 task, we also provide inference times for a subset of point cloud processing networks with available implementations and compare them to our models. The results are presented in Table 6.3. Compared to other methods, our fully connected model is able to achieve sub-millisecond inference time even when being executed on a CPU.

6.5.2 Single-Pass Mesh Registration from 3D Scans

We showcase a second experiment with a different, generative task to demonstrate the versatility and performance of the encoding. For this, we pick the challenging problem of human point cloud registration. In this problem, correspondences are found between an observed, unstructured point cloud and a deformable body template. Traditionally, human point cloud registration has been approached with iterative methods [120, 288]. However, they are typically computationally expensive and require the use of a deformable model at application time. Machine learning based methods [109] remove this dependency by replacing them with a sufficiently large training corpus. However, current solutions like [109] rely on multi-stage models with complex internal representations, which makes them slow to train and test. We encourage the reader interested in human mesh registration to review the excellent summary of previous work provided in [109].

We use a simple DenseNet-like [125] architecture with two blocks (see Figure 6.2), where the input is a BPS encoding of a point cloud and the output is the *location of each*

Method	Intra (cms)	Inter (cms)
Stitched puppets [288]	1.568	3.126
3D-CODED [109]	1.985	2.878
Ours	2.327	4.529
Deep functional maps [160]	2.436	4.826
FARM [174]	2.81	4.123
Convex-Opt [49]	4.86	8.304

Table 6.4: *Results for all published methods in the intra and inter challenge for the FAUST dataset, sorted by error in the intra challenge.* Our BPS-based network has a performance comparable to other methods while allowing single pass, real-time mesh registration, with no per-scan optimizations.

vertex in the common template. Note that there is no deformable model in our system and that we do not estimate deformable model parameters or displacements; the networks learns to reproduce coherent bodies just based on its training data.

To generate this training data, we use the SMPL body model [165]. SMPL is a reshapeable, reposable model that takes as input pose parameters related to posture, and shape parameters related to the intrinsic characteristics of the underlying body (e.g., height, weights, arm length). We sample shape parameters from the CAESAR [227] dataset, which contains a wide variety of ages, body constitution and ethnicities. For sampling poses we use two sources: the CMU dataset [4] and a small set of poses inferred from a 3D scanner. Since the CMU dataset is heavily populated with walking and running sequences, we perform weighted sampling of poses with the inverse Mahalanobis distance from the sample to the CMU distribution as weight. We roughly align the CMU poses to be frontal. To increase the variation of the training data, we introduce noise sampled from the covariance of all the considered poses to half of the data points. From these meshes, a set of 10^4 points is sampled uniformly from the surface of the posed and shaped SMPL template. These point clouds are then used to compute the BPS encoding. We train the alignment network for 1000 epochs in only 4 hours and its inference time is less than 1ms on a non-GPU laptop.

To evaluate our method, we process the test set from the FAUST [33] dataset. It is used to compare mesh correspondence algorithms by using a list of scan points in correspondence. To find correspondences between two point clouds, we process each of them with our network, obtaining as a result two registered mesh templates. The templates then define the dense correspondences between the point clouds.

We obtain an average performance of 2.327cm in the intra-subject challenge and 4.529cm in the inter-subject challenge (see Table 6.4). These numbers are comparable, but higher than state-of-the-art methods like [109] or [288]. However, we note that the

Figure 6.6: *Point clouds of the FAUST dataset and the predicted meshes.* **Blue:** point cloud from a 3D scanner. **Skin color:** predicted mesh by our model through processing of its BPS representation. Note that the network produces the position of each output vertex; their coherent structure is learned solely from the training data.

two methods outperforming BPS in the FAUST intra challenge are orders of magnitude slower than our system. The two-stage procedure in [109] takes multiple minutes and the particle optimization in [288] takes hours, while our system produces alignments in 1ms (for qualitative results, see Figure 6.6). This enables real-time processing of 3D scans, which was previously impossible, or can be used as a first step for faster multi-stage systems that refine the accuracy of this single stage method.

Results on DYNAMIC FAUST. The efficiency of BPS allows us to process large datasets efficiently. We showcase this in the supplementary video[2], where thousands of point clouds from the DYNAMIC FAUST dataset [34] are aligned with BPS. We use exactly the same network used in the FAUST experiments without any retraining or fine-tunning; this shows that our network generalizes well to unseen poses and subjects. Each

[2] `https://youtu.be/kc9wRoI5JbY`

frame is processed independently and no smoothness post-processing was applied to the video. The performance is pretty consistent across the sequences, with one negative factor impacting the accuracy of the alignments: the presence of strong outliers from the floor and the scanner. While the system is robust to the presence of few spurious points around the body, it is sensitive to large amounts of points far from it. Those large chunks drastically change the representation due to the size normalization of the point cloud, which makes the point cloud very different from any cloud in the training set. We believe this could be easily alleviated by introducing similar synthetic noise in our training data or making the normalization more robust to noise.

6.5.3 Training Details

A complete description of network architectures and training strategies is available in the associated code repository. In the this section, we provide a description of the most important elements of the training.

ModelNet40 models. For the classification task, our MLP model consists of 2 fully connected hidden layers, each of size 400. Each layer is followed by a ReLU activation and dropout (with probability 0.8 and 0.4 repsectively). Class probabilities are given by a final dense layer with a softmax activation. The model is trained with the Adam optimizer for 2500 epochs with a batch size of 2048 samples, with an initial learning rate of 1.0e-4 reduced to 1.0e-5 after 2000 epochs.

Our Conv3D-model is similar to the original VoxNet [177] architecture, with 4 blocks of $3 \times 3 \times 3$ convolutions, with each pair of blocks followed by a max-pooling layer of size $2 \times 2 \times 2$. This is followed by 2 fully connected layers of size 512 (with separating dropout layers of 0.8 and 0.6). The model is trained with the Adam optimizer for 505 epochs with a batch size of 512 samples, with an initial learning rate of 1.0e-4 reduced to 1.0e-5 after 500 epochs.

FAUST model. For the FAUST model, the architecture is depicted in Figure 2 of the main manuscript. There, fully connected layers of size 1024 are used. The model is trained with SGD with momentum of 0.9 for 1000 epochs, with an initial learning rate of 1.0 and followup reductions by a factor of 0.5 every 5 epochs of no improvement on the validation set.

6.5.4 Encoding Time

Encoding performance is crucial for real world applications. For our benchmark, we take a random subset of 10^3 ModelNet40 CAD models [274] and sample $10^1 - 10^5$ points from their surfaces. We also vary the number of the basis points k used for the encoding. We investigate two implementations of the BPS encoding scheme. The first one uses the k nearest neighbor implementation available in the sklearn scientific computing package [205]. Here, we use the ball-tree version of kNN search to achieve $O(n \log n + k \log n)$ complexity of the encoding. This version of encoding can be easily

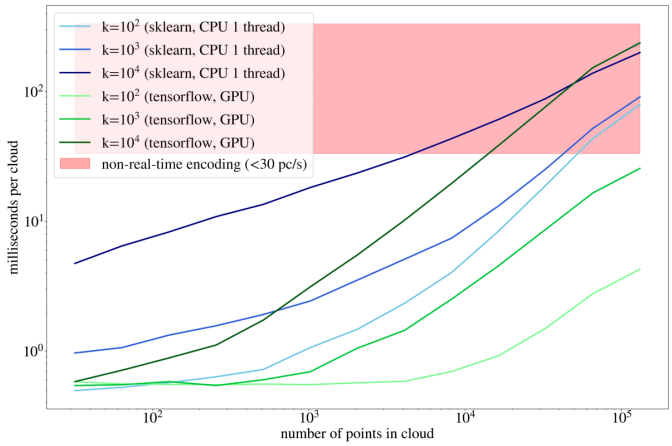


Figure 6.7: *BPS encoding time.* We measure encoding time per point cloud with respect to the number of input points, basis points k and the implementation variant. Encoding can be done in real time for the main application cases considered in this paper (point clouds with $n < 10^5$, encoded with BPS of $k < 10^4$).

parallelized across multiple CPUs for different point clouds. As an alternative, we also provide information on TensorFlow [9] implementation of a direct algorithm for computing pairwise distances. Despite its $O(kn)$ complexity, it shows remarkable performance even for large values of k and n due to the highly efficient execution on a GPU. Various hashing-based algorithms for kNN search designed specifically for 3D data could be used as well [68, 69].

Results are summarized in Figure 6.7. Both implementations show super-real-time encoding performance for the main application case considered in this paper (i.e. point clouds containing $n < 10^5$ points encoded with $k < 10^4$ points). Combined with the proposed deep network, this allows real-time mesh registration from raw scans. Overall, the BPS encoding can be tailored to a specific platform and application via the efficiency-fidelity trade-off in varying k.

6.6 Conclusion and Future Work

In this chapter, we introduced *basis point sets* for obtaining a compact fixed-length represenation of point clouds. BPS computation can be used as a pre-processing step for a variety of machine learning models. In our experiments, we demonstrated in two applications and with different models the computational superiority of our approach with orders of magnitudes advantage in processing time compared to existing methods, remaining competitive accuracy-wise. We have shown the advantage of using rectangular BPS grid in combination with standard 3D-convolutional networks. However, in

future work it would be interesting to consider other types of BPS arrangements and corresponding convolutions [122, 55, 73, 106] for improved efficiency and learning of rotation-invariant representations.

Chapter 7

SMPLpix: Neural Pixels from 3D Human Models

7.1 Introduction

Traditional graphics pipelines for human body and face synthesis benefit from explicit, parameterized, editable representations of 3D shape and the ability to control pose, lighting, material properties, and the camera, to animate 3D models in 3D scenes. While photorealism is possible with classical methods, this typically comes at the expense of complex systems to capture detailed shape and reflectance or heavy animator input. In contrast, recent developments in deep learning and the evolution of graphics processing units is rapidly bringing new tools for human modeling, animation and synthesis. Models based on generative adversarial networks [103] reach previously infeasible levels of realism in synthesizing human faces [134, 135], and various models can repose humans [46], swap identities and appearance [264], etc.

While promising, particularly in terms of their realism, these new "neural" approaches to synthesizing humans have several drawbacks relative to classical methods. Specifically, a key advantage of classical graphics methods [209] is the ability to fully and flexibly control the generative process, e.g. change the camera view, the light or even the pose or shape of the subject. These methods, however, have two main limitations relative to learning-based image synthesis. First, until recently [136, 162], rendering engines were not fully integrated into deep learning pipelines. Second, explicit mesh-based rendering methods are limited when it comes to rendering complex, high-frequency geometry (e.g. hair or fur, wrinkles on clothing, etc.) and dealing with complex, changing, topology. The future of graphics is likely a synthesis of classical and neural models, combining the best properties of both. Here we make a step in this direction by combining the parameterized control of 3D body shape and pose with neural point-based rendering, which replaces the classical rendering pipeline.

Point-based rendering has a long history in computer graphics [108, 145]. Recently, point-based rendering has been successfully coupled with the neural network pipeline via learning per-point neural descriptors that are interpreted by the neural renderer [13]. This approach produces photo-realistic novel views of a scene from a captured point

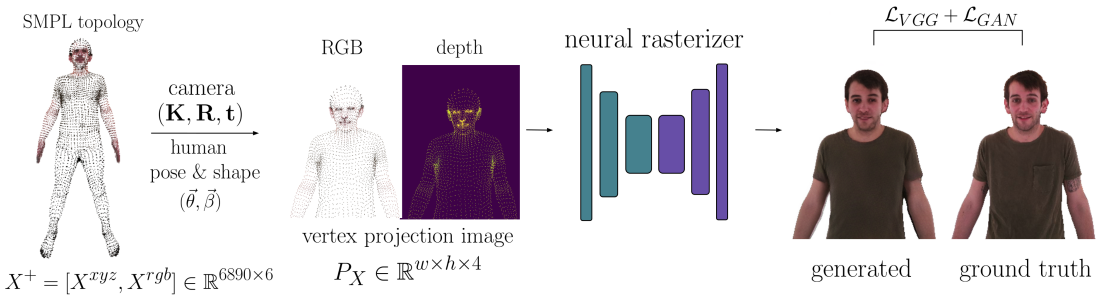


Figure 7.1: *SMPLpix neural rendering pipeline.* Training SMPLpix requires a set of 3D vertices with the corresponding RGB colors as input X^+, along with ground truth camera parameters $(\mathbf{K}, \mathbf{R}, \mathbf{t})$. Our training data is obtained by registering a SMPL model to 3D scans. Using SMPL also allows us to control the coordinates of X^+ via a small set of pose parameters θ. RGB-d training images are created by projecting the vertices, X^+ onto an image plane using a camera model. This image is then fed into a *neural rasterizer* that reconstructs surfaces from projected vertices *directly in the pixel space*. It is trained to minimize a combination of perceptual and adversarial losses wrt the ground truth image. Once trained, neural rasterization module generalizes to unseen subjects X^+, body poses θ and camera parameters $(\mathbf{K}, \mathbf{R}, \mathbf{t})$.

cloud. However, this pipeline has been demonstrated for rendering static scenes with dense point clouds as inputs, with the need of re-learning point descriptors for every novel scene.

Our approach is influenced by [13] and [175]. However, along with the technical novelties and simplifications we describe in the follow-up sections, our main aim is to extend these approaches to enable efficient rendering of human avatars *under novel subject identities and human poses*. We accomplish this by introducing SMPL [165], a deformable 3D body model, into the neural rendering loop. This provides us full control over body pose and shape variation. However, instead of relying on mesh connectivity for explicit rendering, we simply use mesh vertices and their colors projected onto image plane as inputs for the neural rendering module. This provides the benefits of a parameterized body model, while greatly improving the rendering quality, without the complexity of classical methods.

The overall pipeline, called *SMPLpix*, is outlined in Figure 7.1. During training, our framework operates on the data obtained from a commercially available 3D scanner [7]. The SMPL model is registered to the raw scans [33, 165]; other parametric models can be used in principle [129, 204]. The result of this process is a set of mesh vertices $X \in \mathbb{R}^{6890\times 3}$, the RGB color of each vertex, and the body pose parameters θ. It is important to mention that the registration process has inherent limitations like fitting hair (due to the irregularity of hair and low resolution of the SMPL model) or fitting clothing (due to the form-fitting topology of SMPL, see Figure 3). The advantage of using the

registered vertices over raw scans, however, is that we can control the pose of the vertices X by varying a small set of inferred pose parameters θ. We project the vertices of the body model using ground truth scanner camera locations (K, R, t) and obtain an RGB-d image of the projected vertices. This image is processed by a UNet-like neural rendering network to produce the rasterized output RGB image that should match the ground truth image from a scanner camera. At test time, we are given novel mesh vertices X, their colors, body poses θ and camera locations $(\mathbf{K}, \mathbf{R}, \mathbf{t})$. Note that this input can also come from the real images using methods like [14].

Intuition. Our proposed method can be seen as a middle ground between mesh-based and point-based renderers. While we use the structured nature of mesh vertices to control the generative process, we ignore the mesh connectivity and treat vertices simply as unstructured point clouds. Compared with explicit mesh rasterization, the main advantage of this vertices-as-points approach, along with its computational and conceptual simplicity, is the ability of the trained neural rasterizer to reproduce complex high frequency surfaces *directly in the pixel space*, as we will show in the experimental section. Our approach is also potentially applicable in cases when no explicit mesh connectivity information is available whatsoever and only a set of 3D anchor points is given.

Contributions. The proposed work offers the three following contributions:

- *Deep controlled human image synthesis*: apart from the classic mesh-based renderers, to the best of our knowledge, the presented approach is the first one that can render novel human subjects under novel poses and camera views. The proposed framework produces photo-realistic images with complex geometry that are hard to reproduce with these classic renderers;
- *Neural rasterization*: we show how popular image-to-image translation frameworks can be adapted to the task of translating a sparse set of 3D points to RGB images, combining several steps (geometric occlusion reasoning and image enhancement) into a single neural network module that is trained to reconstruct projected surfaces directly in the pixel space and can be easily incorporated into a larger deep learning pipelines;

7.2 Related Work

Our method is connected to several broad branches of 3D modeling and image synthesis techniques. Here we focus on the most representative work in the field.

3D Human Models. Our method is based on the idea of modeling humans bodies and their parts via deformable 3D models [32, 17, 129], and in particular SMPL [165].

Such models are controllable (essential for graphics) and interpretable (important for analysis). Extensions of SMPL exist that also model hands [228], faces [157, 204] and clothing [168]. Separate models exist for capturing and modeling clothing wrinkles and hair [286, 123]. While powerful, rendering such models requires high quality textures and accurate 3D geometry, which can be hard to acquire. Even then, resulting rendered images may look smooth and fail to model details that are not properly captured by the model or surface reconstruction algorithms.

Neural avatars. Recently, a new body of work was developed dedicated to creating high fidelity digital avatars [163, 240, 263, 272]. While these works provide a great level of photo-realism, they are mostly tailored to accurately *modeling a single subject*, and part or the whole system needs to be retrained in case of a new input. In contrast, our system is trained in a multi-person scenario and can render unseen subjects at test time. Another advantage is that it takes a relatively compact generic input (a set of 3D mesh vertices and their RGB colors) that can be also inferred from multiple sources at test time, including from real-world images [14].

Pixel-space image translation and character animation. The second part of our system, neural rasterization, is based on the recent success of pixel-to-pixel image translation techniques [127, 271, 72]. Two particular variations of this framework have the most resemblance to our model. First, [46] uses a set of sparse body keypoints (inferred from a source actor) as input to produce an animated image sequence of a target actor. However, as with the neural avatars discussed above, the system needs to be retrained in order to operate on a novel target subject. Our work also resembles the sketch-to-image translation regime, where an edge image is used in order to produce a photo-realistic image of the person's head [279] or generic objects [51]. Our approach can also be viewed as translating a sparse set of key points into an image. However, our keypoints come from a structured 3D template and therefore convey more information about the rendered subject appearance; since they exist in 3D, they can be projected to an image plane under different camera views. Finally, another advantage of using SMPL topology as input to our image translation framework is its non-uniform vertex density according to region importance (i.e. faces and hands are more densely sampled). This makes detailed rendering of these regions easier, without the need for a specific attention mechanism in the neural renderer itself.

Differentiable mesh (re-)rendering. There are several available solutions that incorporate the mesh rendering step into fully differentiable learning pipelines [165, 136, 162]. However, these methods follow a different line of work: they aim at constructing better gradients for the mesh rasterization step, while keeping the whole procedure of mesh face rendering and occlusion reasoning deterministic. This applies also to a soft rasterizer [162] that substitutes discrete rasterization step with a probabilistic alternative.

While this proved to be useful for the gradient flow, the rendering procedure still lacks full flexibility that would allow it fix artifacts of the original input geometry. One potential solution is to enhance the produced incomplete noisy renders by the additional neural re-rendering module [175, 161]. Our framework can be seen as the one that substitutes explicit rasterization step and combines it with a follow-up image enhancement into a one solid, fully differentiable, task-specific neural rasterization module. Considering the original target application of [175], another potential advantage of our framework for online conferencing is the reduced amount of data that needs to be transferred over the network channel to produce the final image.

Point-based rendering. Point-based rendering [108, 145, 155, 208, 231] offers a well-established, scalable alternative to rendering scenes that can be hard to model with surface meshing approaches. Our inspiration comes mainly from this set of methods; however, we substitute the fixed logic of rendering (e.g. surfel-based [208]) with the neural module in order to adapt to sparse point sets with highly non-uniform densities, as well as to generate photorealistic pixel-space textures.

Rendering from deep 3D descriptors. Another promising direction for geometry-aware image synthesis aims to learn some form of deep 3D descriptors from a 2D or 3D inputs [13, 164, 243, 244]. These descriptors are processed by a trainable neural renderer to generate novel views. These methods, however, are limited when it comes to controlling the generative process; shapes are represented as voxels [164, 243], unstructured point clouds [13] or neural networks weights [244]. This makes parameterized control of human pose difficult.

Neural point-based graphics. The closest work to ours is [13]. An obvious difference with respect to this work is that our input comes from a deformable model, which allows us to modify the render in a generative and intuitive way. Moreover, our model contains two additional differences. First, our inputs are considerably sparser and less uniform than the point clouds considered in [13]. Second, instead of point neural descriptors that need to be relearned for every novel scene or subject, our rendering network obtains the specific details of a subject through the RGB colors it consumes as input *at test time*. This alleviates the need for retraining the system for every novel scene.

In summary, SMPLpix fills an important gap in the literature, combining the benefits of parameterized models like SMPL with the power of neural rendering. The former gives controlability, while the latter provides realism that is difficult to obtain with classical graphics pipelines. This realism is obtained with only weak guidance (sparse points). The key novelty is that the neural rasterizer learns how to take such data and produce realistic images.

7.3 Method

As it is common in deep learning systems, our system has two key parts: the data used for training our model, and the model itself. We describe those two parts in the following sections.

7.3.1 Data

Scans. Our renderer transforms RGB-D 2D projections of SMPL [165] vertices into images. Consequently, we need pairs of images in correspondence with projected vertices to train our model. Although it would be ideal to collect such a dataset from images in the wild, the inaccuracies in methods that infer SMPL bodies from images (e.g. [131]) make this data ineffective. Instead, we use scan data collected in the lab. To that end, we collected more than a thousand scans with a commercially available 3D scanner [7]. This scanner collects two sets of 137 images of resolution 2464×3280, at two time instants separated by around 300 milliseconds. The first set is composed by images collected under a high-frequency pattern illumination created from projectors, and it is used for reconstructing the geometry of the body. The second set is collected under white illumination, and is used to create the body texture; we also use it as the source of our output training data. These images are processed with the commercially available software Agisoft Photoscan [1]. The result of this process is raw 3D point clouds (*scans*) $S \in \mathbb{R}^{M\times 6}, M \approx 10^6$ representing the body geometry, together with camera calibration $(\mathbf{K}, \mathbf{R}, \mathbf{t})$ compatible with a pinhole camera model. Note that the subjects are scanned in a neutral A-pose. Unlike most other image generation methods, this is not a problem for our system since the strong guidance provided by the input images prevents our method from overfitting to the input pose, as it can be seen in Section 7.4.2.

Registrations. While these scans could potentially undergo a rendering process like [13], it would not be possible to deform them in a generative manner, i.e. changing their shape or pose. To achieve that, we transform those unstructured point clouds into a set of points $X \in \mathbb{R}^{N\times 3}, N = 6890$ with fixed topology that correspond to a reshapeable and reposeable mode, SMPL. In its essence, SMPL is a linear blend skinning (LBS) model which aims to model the observed body vertices X as a function of identity-dependent and pose-dependent mesh deformations, driven by two corresponding compact sets of shape $\vec{\beta} \in \mathbb{R}^{10}$ and pose $\vec{\theta} \in \mathbb{R}^{72}$ parameters:

$$X = W(T_P(\vec{\beta}, \vec{\theta}), J(\vec{\beta}), \vec{\theta}, \mathcal{W}), \tag{7.1}$$

$$T_P(\vec{\beta}, \vec{\theta}) = \bar{\mathbf{T}} + B_S(\vec{\beta}) + B_P(\vec{\theta}), \tag{7.2}$$

where $T_P(\vec{\beta}, \vec{\theta})$ models shape and pose dependent deformation of the template mesh in the canonical T pose via linear functions B_S and B_P, and W corresponds to the LBS

function that takes the T-pose template T_P, set of shape-dependent K body joint locations $J(\vec{\beta}) \in \mathbb{R}^{3K}, K = 23$ and applies LBS function W with weights $\mathcal{W}$ to produce the final posed mesh. We refer to the original publication [165] for more details on the SMPL skinning function. Note that other versions of deformable 3D models [17, 129] or topologies could be used, including the ones that additionally model hands and faces [157, 204, 228], as well as clothing deformations [168]. In fact, in Section 7.4.2 we show experiments with two topologies of different cardinality.

The SMPL registration process optimizes the location of the registration vertices and the underlying model, so that the distance between the point cloud and the surface entailed by the registration is minimized, while the registration vertices remain close to the optimized model. It is inspired by the registration in [33] although the texture matching term is not used. It is worth re-emphasizing that these registrations, as registrations in [33], can contain details about the clothing of the person since their vertices are optimized as free variables. This does not prevent us from reposing those subjects after converting them into SMPL templates $\bar{\mathbf{T}}^*$ through unposing, as explained and shown in Section 7.4.3. However, these extra geometric details are far from perfect (e.g. they are visibly wrong in the case of garments with non-anthropomorphic topology, like skirts), and our model learns to make them more similar to the final images in a data-driven manner.

Color. Finally, the registered mesh is used in Agisoft Photoscan together with the original image and camera calibration to extract a high resolution texture image $I_{tex} \in \mathbb{R}^{8192\times8192\times3}$. This texture image is a flattened version of the SMPL mesh, in which every 3D triangle is SMPL corresponds to a 2D triangle in the texture image. Therefore, each triangle contains thousands of color pixels representing the appearance of that body portion. These textures can be used directly by the classic renderer to produce detailed images, as can be seen in Section 7.4.2. Although it would be possible to exploit the detail in those textures by a neural renderer, that would slow it down and make it unnecessarily complex. Instead, we propose to use the sparse set of colors $X^c \in \mathbf{R}^{6890\times3}$ sampled at the SMPL vertex locations. These colors can be easily extracted from the texture image, since they are in full correspondence with the mesh topology.

Projections. Having an input colored vertex set $X^+ = [X, X^c] \in \mathbf{R}^{6890\times6}$ and camera calibration parameters $(\mathbf{K}, \mathbf{R}, \mathbf{t})$, we obtain image plane coordinates for every vertex $x \in X$ using a standard pinhole camera model [115]:

$$\begin{pmatrix} u \\ v \\ d \end{pmatrix} = \mathbf{K}(\mathbf{R}x + \mathbf{t}). \tag{7.3}$$

Next, we form an RGB-D vertex projection image. The projection image $P_X \in \mathbb{R}^{w\times h\times4}$

is initialized to a value that can be identified as background by its depth value. Since depth values collected in the scanner have a range between 0.1 and 0.7 meters, a default value of 1 is used to initialize both RGB and depth in P_X. Then, for every vertex $x \in X$, its image plane coordinates (u, v, d) and color values $(r, g, b) \in X^c$ we assign:

$$P_X[\lfloor u \rfloor, \lfloor v \rfloor] = (r, g, b, d). \tag{7.4}$$

In order to resolve collisions during the projection phase (7.4), when different vertices from X end up sharing the same pixel-space coordinates $\lfloor u \rfloor, \lfloor v \rfloor$, we sort the vertices according to their depth and eliminate all the duplicate consecutive elements of the depth-wise sorted array of $\lfloor u \rfloor, \lfloor v \rfloor$ coordinates of X. However, we would also like to note that since the number of vertices is much smaller than the full resolution of the image plane, these collisions rarely happen in practice.

The whole vertex projection operation (7.3)-(7.4) can be easily and efficiently implemented within modern deep learning frameworks [203] and, therefore, seamlessly integrated into bigger pipelines.

Image segmentation. Apart from input projection images, our algorithm also needs example RGB output images $I_X \in \mathbb{R}^{w \times h \times 3}$. We would like to be agnostic to the backgrounds in training data, so that the model does not learn to replicate the scanner background. In order to replace the background with a constant value (e.g. white), we automatically segment the scanner images. We tried off-the-shelf segmentation algorithms like Deeplab [48], with highly inaccurate results. An alternative would be to use the projection of the scan to the image plane as a groundtruth mask; however, these projections are noisy. We found that a good compromise is to train a segmentation model with the point cloud projections. The network acts as a regularizer and denoiser, giving us better results than the previous approaches. Note that for the tests in Section 7.4.2 we test the rendered images against ground truth segmented *manually*. Manual segmentations have a better quality than our automatic ones, but are much slower and expensive to obtain for the full dataset. In this way, we avoid the metrics favouring our method because of replicating segmentation noise present in our algorithm.

7.3.2 Neural rasterization

Given our training data consisting of pairs of RGB-D projection images P_X and segmented output images I_X, we train a UNet-type [229] neural network G with parameters Θ to map inputs to outputs:

$$G_\Theta : P_X \rightarrow I_X \tag{7.5}$$

In our experiments, we use one of the publicly available UNet architecture designs [8], to which we apply only minor changes to adapt it to our types of input and output. We

also replace transposed convolutions with upsample-convolutions to avoid checkerboard artifacts [194]. In general, the particular design of this module can be further optimized and tailored to a specific target image resolution and hardware requirements; we leave this optimization and further design search for a future work.

Having the ground truth image I_{gt} for the given subject and camera pose, we optimize our rendering network G_Θ for the weighted combination of perceptual VGG-loss [128], multi-scale, patch-based GAN loss and feature matching GAN loss [271]:

$$\mathcal{L}(I_{gt}, I_X) = \alpha \mathcal{L}_{VGG}(I_{gt}, I_X) + \kappa \mathcal{L}_{GAN}(I_{gt}, I_X) + \gamma \mathcal{L}_{FM}(I_{gt}, I_X) \tag{7.6}$$

where α, κ, γ are the loss term weights. The overall setup is similar to [271]. We set $\alpha = 1, \kappa = 0, \gamma = 0$ for the first 100 epochs with the learning rate of Adam [141] optimizer set to 1.0e-4, proceeding with another 50 epochs with $\alpha = 1, \kappa = 1.0, \gamma = 0.1$ and learning rate set to 1.0e-5.

Implicitly, the network G_Θ is learning to accomplish several tasks. First, it needs to learn some form of geometric reasoning, i.e. to ignore certain projected vertices based on their depth values. In that sense, it substitutes fixed-logic differentiable mesh rendering procedures [136] with a flexible, task-specific neural equivalent. Second, it needs to learn how to synthesize realistic textures based on a sparse supervision provided by the projected vertices, as well as to hallucinate whole areas not properly captured by the 3D geometry, e.g. hair and clothing, to match the real ground truth images. Therefore, we believe that this approach could serve as potentially superior (in terms of acquired image realism), as well as easier to integrate, computationally flexible alternative to the explicit fixed differentiable mesh rendering step of [136].

7.4 Experiments

7.4.1 Data Details

Accurately captured, well-calibrated data is essential for the proposed approach in its current form. We use 3D scans of 1668 subjects in casual clothing. The subjects are diverse in gender, body shape, age, ethnicity, as well as clothing patterns and style. For each subject, we select 20 random photos from one of 137 camera positions available in the scanner camera rig. We use 1600 subjects for training and 68 subjects for test, which forms training and test sets of 32000 and 1360 images correspondingly. We use the image resolution of size 410×308 during all the experiments. Of 68 test subjects, 16 gave their explicit consent for their images to be used in the present submission. We use these test subjects for the qualitative comparison presented in this chapter, while the full test set is used for the quantitative evaluation.

Figure 7.2: *Qualitative comparison between Neural Mesh Renderer and SMPLpix (zoom in for details).*

7.4.2 Quantitative Experiments

In this section, we compare our system with other renderers that can generate images of reshapeable and reposeable bodies. This limits the other methods to be classic rendering pipelines, since, to the best of our knowledge, no other deep learning models offers this generative behaviour. It is important also for us that the renderers support automatic differentiation, since our ultimate goal includes integrating the renderer with a fully differentiable learning system. With these two constraints, we decided to choose as our plattform for comparison the neural mesh renderer introduced in [136], in its popular PyTorch re-implementation [6].

Table 7.1: *Neural mesh renderer [136] vs SMPLpix neural rasterization pipeline.*

	Method	PSNR ↑	LPIPS ↓	Input size (RGB)	Inference t.
1	NMR[136] (7k, per-verts)	23.2	0.072	$\mathbb{R}^{6890\times3}$	13ms
2	NMR[136] (7k, full textures)	23.4	0.049	$\mathbb{R}^{8192\times8192\times3}$	14ms
3	NMR[136] (27k, per-verts)	23.5	0.064	$\mathbb{R}^{27578\times3}$	48ms
4	NMR[136] (27k, full textures)	23.6	0.047	$\mathbb{R}^{8192\times8192\times3}$	50ms
5	SMPLpix (7k verts)	24.2	0.051	$\mathbb{R}^{6890\times3}$	16ms
6	SMPLpix (27k verts)	24.6	0.045	$\mathbb{R}^{27578\times3}$	17ms

Figure 7.3: *Novel view generation*.

Metrics. We compare SMPLpix against different versions of classic renders implemented with [6] according to two different quantitative metrics popular in image generation and super-resolution: peak signal-to-noise ratio (PSNR) and Learned Perceptual Image Patch Similary (LPIPS, [282]). PSNR is a classic method, while LPIPS has gained popularity in recent work for being more correlated with the perceptual differences. We should note that the field of quantitative perceptual evaluation is still an area of research, and no metric is perfect. Therefore we also provide qualitative results in the next section.

Baseline variants. For [136], we use the following rendering variants. First, we render the mesh with exactly the same information available to our SMPLpix rendering pipeline, i.e. only 1 RGB color per vertex [1]. Next, we use the much more information-dense option of texture images I_{tex}. For a fair comparison of inference times, we do not utilize the full extensive 8k textures, but rather search for the optimal downscaled version of the texture image, at which no further improvement in terms of PSNR and LPIPS were observed (Table 7.1, row 2). Since our method can be topology agnostic, we perform these comparisons for two topologies: the native SMPL topology of 6890 vertices (noted as $7k$) and a subsampled version with a higher vert count of 27578 vertices (noted as $27k$).

Results. The values for PSNR and LPIPS are compiled in Table 7.1. The first conclusion to extract from this table is that, given a fixed amount of color information (i.e.

[1] Technically, since [6] does not support per-vertex color rendering, this has to be achieved by performing linear interpolation between the vertex colors in their per-triangle texture space

Figure 7.4: *Pose generation*.

comparing per-verts NMR against SMPLpix for a fixed topology), SMPLpix clearly outperforms NMR in both PSNR and LPIPs. Limiting the color information can be useful in terms of computational and data transmission efficiency, and the use of textures makes the rendering system arguably more complex. However, we included also a comparison against NMR using full textures. Although the values are much closer, SMPLpix slightly outperforms NMR also in this case. This validates our main hypothesis, i.e. that the adaptive rendering procedure described in Section 7.3.2 can learn a valid rendering prior of the human texture and surface, and reproduce it based on a sparse input given by the colored mesh vertices. Moreover, it outperforms the conventional methods in terms of acquired level of realism since it could be trained end-to-end to reproduce the corresponding photo. In terms of efficiency, using low dimensional geometry with no anti-aliasing and full textures achieves the fastest running times, followed closely by SMPLpix, which obtained better quality metrics. Also, note that for NMR the inference time grows roughly linearly with the number of geometry elements, while for our method most of the time is being spent in the neural rasterization module that is agnostic to the number of projected points. Being a UNet-like neural network, this module can be further optimised and tailored to specific hardware requirements.

7.4.3 Qualitative experiments

Since it is well known that perceptual metrics are not perfect in capturing the quality of synthetic images, we also provide in this section examples for the reader to judge the quality of our method and experience the potential applications that its generative character bring to the field of neural rendering.

Qualitative comparison. We can see a visual comparison of ground truth and the methods previously described in Figure 7.2. First thing to note is that these images contain elements which are known to be difficult to model with SMPL topology: hair, baggy clothes or shoes, for example. We can observe that since the relation between geometry and pixel colors in NMR is very constrained, the geometry artifacts are still visible in the renders. Please note for example the unrealistic hair buns in NMR, smoothed out clothes in the first column, and the unrealistic ear shape in the sixth column due to the lack of independent hair geometry that covers the ears in the SMPL topology. In com-

Figure 7.5: *Shape variations with SMPLpix. First column shows the render of the original subject, subsequent columns explore the first three directions of the SMPL shape space, in the negative and positive directions, on top of the subject specific model.*

parison, SMPLpix learns to correlate those artifacts with specific combinations of vertex locations and shapes, and recreates loose hair, pony tails, or loose clothing (up to some extent). Another type of artifact that is corrected is wrong textures due to misalignments: as we can see in the fourth column, the hand area contain pixels of background color for this reason. SMPLpix learns to correct this artifact too. Finally, pay attention to the toes rendered on the shoes by NMR, due to the SMPL topology, which are corrected by our renderer in the next to last column. It is important to note that some of these details are reconstructed in a plausible way, though not in the exact way they are present in the ground truth. On the negative side, high frequency patterns like the striped shirt in the last column are not captured well by the sparsely sampled vertex colors, as can be seen both in NMR per-vertex renders and SMPLpix. Higher sampling resolution helps, although ensuring perfect reproducibility would be too expensive with this method. An alternative, left to study in future work, is to include per vertex deep descriptors in the projected image used as input.

Novel view generation. A first question about SMPLpix generalization capabilities is how well does it generalize to novel views. Figure 7.3 shows images generated from novel viewpoints with our algorithm. Given the ample coverage of views achieved by the scanning data, we can generate views from almost arbitrary orientations. However, we should note that the distance to the subject is not covered nearly as well in our setup, and the quality of our results degrade when the camera is too far or too close to the person. A possible way to handle this, left for future work, is to augment the data with arbitrary scalings of the input mesh and image.

Pose generation. An advantage of our method with respect to the main other point-based renderer [13] is that we can alter the renders in a generative manner, thanks to the SMPL model that generates our inputs. To that end, we take the registrations previously

Figure 7.6: *SMPLpix using textures inferred by [15] from in the-wild-photos.*

mentioned and create a subject specific model in the same spirit as in [212]. A subject specific model has a template which is obtained by reverting the effects of the estimated registration pose. More specifically, it involves applying the inverse of the LBS transformation W^{-1} and subtracting the pose-dependent deformations $B_P(\vec{\theta})$ (Equations 7.1 and 7.2) to the registration. We can repose this subject specific model to any set of poses compatible with the SMPL model. To that end, we tried some sequences from AMASS [171]. As can be seen in Figure 7.4, bodies can deviate largely from the A-pose in which most of the subjects stand in the training data. Experimentally, we have observed that this is very different for other neural renderers like [46].

Shape generation. Although [13] cannot generate people arbitrarily posed, other renderers like [46, 240] potentially can, if they have a way to generate new skeleton images. However, shape cannot change with those approaches, since skeletons only describe the length of the bones and not body structure. We can see this potential application in Figure 7.5. For this figure, we used the previously mentioned subject-specific SMPL model for two of the subjects, and modified their shape according to the first three components of the original SMPL shape space. We can see that shape variations are realistic, and details like hair or clothing remain more realistic than in the classic renders images. To our knowledge, this is the first realistic shape morphing obtained through neural rendering.

Subject color from images. A reasonable objection to SMPLpix is that the results that we have shown here use texture images extracted from high-end scanners. However, this is not a hard requirement from our method (Figure 7.6). Although there is a clear drop in image quality performance, due to the worse quality of the input colors, we see that our method can generate good images from textures provided by the authors of [15]. These textures were computed from a set of images of a person rotating in front of a camera, out of the lab. In order to eliminate the artifacts present in the images, a possible future direction would be to train our system with the outputs from the algorithm used to generate images at test time (e.g. [15]).

7.5 Conclusion and Future Work

In this work, we presented SMPLpix, a deep learning model that combines deformable 3D models with neural rendering. This combination allows SMPLpix to generate novel bodies under clothing, with the advantages that neural rendering has: better quality and data-driven results. Unlike any other body neural renderer, SMPLpix can vary the shape of the person and does not require to be retrained for each subject.

Apart from the advantages previously mentioned, one of the key characteristics of SMPLpix is that, unlike the classic renderers, it is improvable and extensible in a number of ways. We are particularly interested in integrating the renderer with systems that infer SMPL bodies from images (e.g. [131, 147, 146] to enable an end-to-end system for body image generation trained from images in the wild.

SMPLpix is a relevant step towards controllable body neural renderers, but it can obviously be improved. Rendering high frequency textures remains a challenge, although including extra information (like per-vertex image descriptors, similar to the local image descriptors pooled across views in [232]) in our input projection image is a promising approach.

Chapter 8

Conclusions and Outlook

In this thesis, we have studied and developed several techniques that contribute to the overall goal of building robust and computationally efficient vision systems. Below we summarize the main results and provide an outlook for a future work in each direction.

8.1 Deep Probabilistic Models

We have investigated efficient ways to combine modern deep learning systems with probabilistic reasoning. We have provided a brief survey of the current approaches in uncertainty reasoning, showcasing potential advantages and shortcomings of each method.

The dedicated family of methods was developed that addresses the task of probabilistic regression of circular data. We have shown that these methods can efficiently infer object orientation from images of different modalities and of varying quality and complexity, providing rich multimodal densities that reflect potential ambiguities in observations in object orientation. This allows to incorporate these systems into larger vision pipelines that involve complex decision making, e.g. autonomous driving and other robotic systems.

Via close collaboration with the experts of respective fields, we have also investigated variants of deep probabilistic framework that could be efficiently deployed in critical real-world applications. Namely, we developed a method for obtaining MRI CEST contrasts useful in medical diagnosis in a fast and robust manner, considerably reducing the computational time compared to conventional methods. Additionally, we have investigated deep trajectory predictors models for high speed dynamic robotic environments. Here, the developed temporal model proved to be superior to classic approaches based on physics modeling. Its small inference time allowed us to incorporate it into the real time control loop of a robotic arm in a table tennis environment.

Limitations and future work. In essence, all the developed frameworks are based on constructing suitable probabilistic outputs on top of existing neural network models and utilizing maximum likelihood principle. This allows them to efficiently handle noisy and ambiguous observations and indicate their lack of confidence in predicted values via increased level of uncertainty in a theoretically rigorous and consistent manner. However, as we have also observed on a simple example in Chapter 2, pure maximum

likelihood based training mostly addresses the aleatoric uncertainty in the data. It should be combined with other methods or data augmentation techniques in order to efficiently handle epistemic uncertainty. This is essential for practical applications where out-of-distributions samples can occur.

Moreover, as computer vision pipelines become a part of our daily lives, they are also becoming a target for various types of attacks and misuses, e.g. by mining adversarial examples that try to actively fool the system. Such samples could be considered a hard case of out-of-distribution data, and truly robust systems should be able to handle them properly.

Another important prospect for the future of robust systems is coming up with efficient evaluation protocols for the provided probabilistic predictions. Classic evaluation methodologies like test time likelihoods or calibration curves all have their potential shortcomings. It might be therefore desirable to link the quality of provided uncertainty estimates to the final performance of the holistic decision making system. In case of robotics applications, this will mean that we would judge the efficiency of the component by its influence on the final task performance.

8.2 Efficient Learning on Point Clouds

We have proposed a novel representation for 3D point clouds, a type of data that is gaining increasing attention in computer vision research due to the large availability of depth sensing devices and multi view reconstruction software. The developed basis point set (BPS) encoding method implies a simple and efficient way to convert unordered sets of points of variable length into fixed vector representation that can be easily used with virtually any machine learning algorithm.

We have shown the efficiency of the proposed point cloud encoding when combined with deep neural networks. Compared to other popular deep learning approaches, our pipelines were able to achieve competitive or superior 3D shape classification accuracy with orders of magnitude lower amount of computations. Additionally, we have shown how it can be efficiently utilized in the task of inferring parametric human body models from raw scans, with a potential to substitute time-consuming iterative optimization registration pipelines.

Limitations and future work. Presented approach opens up several directions for the future work. First, in the present work we have investigated mainly applications where input point clouds can be effectively downsampled to a convenient size (10^4-10^5 points). However, many real-worlds 3D captures [59] can induce point clouds that are at least an order of magnitude bigger. Encoding such point clouds will likely require additional modifications in order to maintain accuracy and efficiency of the representation.

Additionally, basis point set encoding in its current form provides a global vector descriptor of a point cloud. However, in many tasks, e.g. point cloud segmentation, it is required to provide per-point predictions of various kinds. One simple, yet not a

particularly efficient way to achieve this goal would be to simply process every point together with the global BPS vector via some additional network module.

Another limitation that BPS representation shares with the majority of 3D processing frameworks is its non-invariance to 3D rotations. In both considered applications we have dealt with the objects that were roughly aligned along the axes, something that would rarely happen if we consider dynamic 3D capture environments. Possible crude solutions to this problem include additional pose estimation framework that will bring the object to canonical pose, as well as augmenting training data with rotated versions of objects. However, building truly rotation invariant BPS shape descriptors is the most elegant way to handle rotations.

More generally, while provided method have been proved useful as an *input* representation for point clouds, the search for optimal *output* representation for 3D surfaces is an ongoing endeavour. Here, exploring the full representational power of deep implicit functions might highlight a new era of high accuracy 3D reconstructions from noisy data.

8.3 Neural Human Rendering

We have proposed a novel rendering pipeline that combines merits of classical deformable 3D models with the power of learning-based adaptive rendering. While deformable models allow us to flexibly control human avatar shape and pose, neural rendering module can fix the visual artifacts and fill in missing details directly in the pixel space. This allows us to render more realistically high frequency geometry details like hair, parts that are regularly hard to address with classic mesh rasterization rendering schemes.

We have trained our model on a large number of 3D human scans and have observed its generalization capabilities with respect to novel subjects, camera parameters, body poses and shapes. We have shown both qualitatively and quantitatively that the proposed hybrid approach to computer graphics can achieve superior results compared to classic pipelines. We believe that this opens new possibilities for the truly photorealistic computer graphics.

Limitations and future work. There are multiple exciting directions of future research in the neural rendering field. First, in order to create an avatar, our model in its current form requires a relatively expensive setup for obtaining 3D scan of a person. Exploring the ways to obtain the necessary 3D information from uncalibarated collections of photos can be a huge democratizing factor that will allow everyone to create their visual digital copy.

Second, the introduced neural rendering module in its current form assumes fixed light model. However, in order to be able to immerse created avatars into virtual environments in a realistic manner, we will need to be able to change the rendering results based on the ambient light. Potential solutions here include augmenting the initial 3D model with some features that reflect material properties, as well as conditioning the neural rasterization framework on the light source.

Finally, currently we rely on models like SMPL [165] for initial 3D modeling. While being expressive in terms of body shape and pose variation, these models are limited in terms of reflecting clothing details that deviate significantly from the minimally clothed body topology. One of the promising research directions in this regard is investigating other forms of parameterizing 3D shapes and non-rigid deformations, e.g. deformation-aware implicit functions [126].

8.4 Afterword

We are living in the time when machine learning and computer vision algorithms are influencing almost every side of our daily live, from the everyday shopping experience to the diagnosis of the most serious deceases. In order to bring a truly revolutionary change, however, these technologies should be able to act in and adapt to complex dynamic environments. They should also be easily accessible by a large audience. Practically, this means two things. First, the models we deploy should be able to make robust decisions, even under situations with a high degree of uncertainty. Second, they should be computationally- and energy-efficient in order to be deployed massively and run on compact computational devices. We hope that the algorithms considered and developed in this thesis will serve as important building blocks in the large scale computer vision technologies of tomorrow.

Symbols

x	variable
$\mathbf{x}$	vector
$\mathbb{R}$	real numbers
$\|\|\mathbf{x}\|\|$	L2 norm of a vector $\mathbf{x}$
$KL(q\|\|p)$	Kullback-Leibler divergence between two distributions

Abbreviations

API	Application programming interface
CVAE	Conditional variational autoencoder
GPU	Graphics processing unit
i.i.d	Independent and identically distributed
MLE	Maximum likelihood estimation
MSE	Mean squared error
NLL	Negative log-likelihood
OOD	Out of distribution (data)
RGB	Red-green-blue (image)
SfM	Structure from motion
SLaM	Simultaneous localization and mapping
SMPL	Skinned multi-person linear model [165]
VAE	Variational autoencoder

Bibliography

[1] Agisoft photoscan. https://www.agisoft.com/. Accessed: 2020-03-05.

[2] Apple face id security guide. https://www.apple.com/ca/business-docs/FaceID_Security_Guide.pdf. Accessed: 2020-04-15.

[3] A bee c: Scientists translate honeybee queen duets. https://www.bbc.com/news/science-environment-53029218. Accessed: 2020-04-15.

[4] Cmu graphics lab motion capture database. http://mocap.cs.cmu.edu. Accessed: 2020-06-30.

[5] The five biggest questions about apple's new facial recognition system. https://www.theverge.com/2017/9/12/16298156/apple-iphone-x-face-id-security-privacy-police-unlock. Accessed: 2020-04-15.

[6] Pytorch neural renderer. https://github.com/daniilidis-group/neural_renderer. Accessed: 2020-02-24.

[7] Tredys 3d scanner. http://www.treedys.com/. Accessed: 2020-02-24.

[8] U-net neural network in pytorch. https://github.com/milesial/Pytorch-UNet. Accessed: 2020-02-24.

[9] Martin Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv:1603.04467*, 2016.

[10] Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M Seitz, and Richard Szeliski. Building rome in a day. *Communications of the ACM*, 54(10):105–112, 2011.

[11] Tomas Akenine-Möller, Eric Haines, and Naty Hoffman. *Real-time rendering*. Crc Press, 2019.

[12] Alexander A Alemi, Ian Fischer, and Joshua V Dillon. Uncertainty in the variational information bottleneck. *arXiv preprint arXiv:1807.00906*, 2018.

[13] Kara-Ali Aliev, Dmitry Ulyanov, and Victor Lempitsky. Neural point-based graphics. *arXiv preprint arXiv:1906.08240*, 2019.

[14] Thiemo Alldieck, Marcus Magnor, Bharat Lal Bhatnagar, Christian Theobalt, and Gerard Pons-Moll. Learning to reconstruct people in clothing from a single rgb camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1175–1186, 2019.

[15] Thiemo Alldieck, Marcus Magnor, Weipeng Xu, Christian Theobalt, and Gerard Pons-Moll. Detailed human avatars from monocular video. In *International Conference on 3D Vision*, pages 98–109, Sep 2018.

[16] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

[17] Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. Scape: shape completion and animation of people. In *ACM SIGGRAPH 2005 Papers*, pages 408–416. 2005.

[18] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*, 2018.

[19] Aurore Avarguès-Weber, Adrian G Dyer, Maud Combe, and Martin Giurfa. Simultaneous mastering of two abstract concepts by the miniature brain of bees. *Proceedings of the National Academy of Sciences*, 109(19):7481–7486, 2012.

[20] Aurore Avarguès-Weber and Martin Giurfa. Conceptual learning by miniature brains. *Proceedings of the Royal Society B: Biological Sciences*, 280(1772):20131907, 2013.

[21] Sileye O Ba and Jean-Marc Odobez. A probabilistic framework for joint head tracking and pose estimation. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 4, pages 264–267. IEEE, 2004.

[22] Chiraz BenAbdelkader. Robust head pose estimation using supervised manifold learning. *Computer Vision–ECCV 2010*, pages 518–531, 2010.

[23] Abhijit Bendale and Terrance E Boult. Towards open set deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1563–1572, 2016.

[24] Ben Benfold and Ian Reid. Stable multi-target tracking in real-time surveillance video. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3457–3464. IEEE, 2011.

[25] Ben Benfold and Ian Reid. Unsupervised learning of a scene-specific coarse gaze estimator. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2344–2351. IEEE, 2011.

[26] James O Berger. *Statistical Decision Theory and Bayesian Analysis*. Springer, 1980.

[27] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

[28] Lucas Beyer, Alexander Hermans, and Bastian Leibe. Biternion nets: Continuous head pose regression from discrete training labels. In *German Conference on Pattern Recognition*, pages 157–168. Springer International Publishing, 2015.

[29] Amlaan Bhoi. Monocular depth estimation: A survey. *arXiv preprint arXiv:1901.09402*, 2019.

[30] Christopher M Bishop. Novelty detection and neural network validation. *IEE Proceedings-Vision, Image and Signal processing*, 141(4):217–222, 1994.

[31] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[32] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3d faces. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 187–194, 1999.

[33] Federica Bogo, Javier Romero, Matthew Loper, and Michael J Black. Faust: Dataset and evaluation for 3d mesh registration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3794–3801, 2014.

[34] Federica Bogo, Javier Romero, Gerard Pons-Moll, and Michael J. Black. Dynamic FAUST: Registering human bodies in motion. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2017*, Piscataway, NJ, USA, July 2017. IEEE.

[35] Anastasia Borovykh, Sander Bohte, and Cornelis W Oosterlee. Conditional time series forecasting with convolutional neural networks. *arXiv preprint arXiv:1703.04691*, 2017.

[36] Diane Bouchacourt, Pawan K Mudigonda, and Sebastian Nowozin. DISCO Nets: DISsimilarity COefficients Networks. In *Advances in Neural Information Processing Systems*, pages 352–360, 2016.

[37] George EP Box. Science and statistics. *Journal of the American Statistical Association*, 71(356):791–799, 1976.

[38] Johannes Breitling, Anagha Deshmane, Steffen Goerke, Andreas Korzowski, Kai Herz, Mark E Ladd, Klaus Scheffler, Peter Bachert, and Moritz Zaiss. Adaptive denoising for chemical exchange saturation transfer mr imaging. *NMR in Biomedicine*, 32(11):e4133, 2019.

[39] Glenn W Brier. Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1):1–3, 1950.

[40] Oliver Brock and Oussama Khatib. High-speed navigation using the global dynamic window approach. In *Proceedings 1999 ieee international conference on robotics and automation (Cat. No. 99CH36288C)*, volume 1, pages 341–346. IEEE, 1999.

[41] Dieter Büchler, Heiko Ott, and Jan Peters. A lightweight robotic arm with pneumatic muscles for robot learning. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4086–4092. IEEE, 2016.

[42] Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

[43] Richard B Buxton. *Introduction to functional magnetic resonance imaging: principles and techniques*. Cambridge university press, 2009.

[44] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on robotics*, 32(6):1309–1332, 2016.

[45] Isarun Chamveha, Yusuke Sugano, Daisuke Sugimura, Teera Siriteerakul, Takahiro Okabe, Yoichi Sato, and Akihiro Sugimoto. Head direction estimation from low resolution images with scene adaptation. *Computer Vision and Image Understanding*, 117(10):1502–1511, 2013.

[46] Caroline Chan, Shiry Ginosar, Tinghui Zhou, and Alexei A Efros. Everybody dance now. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5933–5942, 2019.

[47] Cheng Chen and Jean-Marc Odobez. We are not contortionists: Coupled adaptive learning for head and body orientation estimation in surveillance video. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1544–1551. IEEE, 2012.

[48] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[49] Qifeng Chen and Vladlen Koltun. Robust nonrigid registration by convex optimization. In *ICCV*. IEEE Computer Society, 2015.

[50] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM, 2016.

[51] Wengling Chen and James Hays. Sketchygan: Towards diverse and realistic sketch to image synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9416–9425, 2018.

[52] Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. Implicit functions in feature space for 3d shape reconstruction and completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6970–6981, 2020.

[53] François Chollet et al. Keras. `https://github.com/fchollet/keras`, 2015.

[54] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[55] Taco S Cohen, Mario Geiger, Jonas Köhler, and Max Welling. Spherical cnns. *arXiv preprint arXiv:1801.10130*, 2018.

[56] John Horton Conway and Neil James Alexander Sloane. *Sphere packings, lattices and groups*, volume 290. Springer Science & Business Media, 2013.

[57] Cyril Crassin, Fabrice Neyret, Sylvain Lefebvre, and Elmar Eisemann. Gigavoxels: Ray-guided streaming for efficient and detailed voxel rendering. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games*, pages 15–22, 2009.

[58] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. 1996.

[59] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5828–5839, 2017.

[60] Angela Dai and Matthias Nießner. Scan2mesh: From unstructured range scans to 3d meshes. *arXiv preprint arXiv:1811.10464*, 2018.

[61] Matthias Dantone, Juergen Gall, Gabriele Fanelli, and Luc Van Gool. Real-time facial feature detection using conditional regression forests. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2578–2585. IEEE, 2012.

[62] Meltem Demirkus, James J Clark, and Tal Arbel. Robust semi-automatic head pose labeling for real-world face video sequences. *Multimedia Tools and Applications*, 70(1):495–523, 2014.

[63] Meltem Demirkus, Doina Precup, James J Clark, and Tal Arbel. Probabilistic temporal head pose estimation using a hierarchical graphical model. In *European Conference on Computer Vision*, pages 328–344. Springer, 2014.

[64] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[65] John S Denker and Yann LeCun. Transforming neural-net output levels to probability distributions. In *Advances in neural information processing systems*, pages 853–859, 1991.

[66] Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.

[67] Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

[68] Bertram Drost and Slobodan Ilic. A hierarchical voxel hash for fast 3d nearest neighbor lookup. In *German Conference on Pattern Recognition*, pages 302–312. Springer, 2013.

[69] Bertram H Drost and Slobodan Ilic. Almost constant-time 3d nearest-neighbor lookup using implicit octrees. *Machine Vision and Applications*, 29(2):299–311, 2018.

[70] David Eberly. *3D game engine design: a practical approach to real-time computer graphics*. CRC Press, 2006.

[71] Harald E Esch, Shaowu Zhang, Mandyan V Srinivasan, and Juergen Tautz. Honeybee dances communicate distances measured by optic flow. *Nature*, 411(6837):581–583, 2001.

[72] Patrick Esser, Ekaterina Sutter, and Björn Ommer. A variational u-net for conditional appearance and shape generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8857–8866, 2018.

[73] Carlos Esteves, Christine Allen-Blanchette, Ameesh Makadia, and Kostas Daniilidis. Learning so (3) equivariant representations with spherical cnns. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 52–68, 2018.

[74] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[75] Gabriele Fanelli, Juergen Gall, and Luc Van Gool. Real time head pose estimation with random regression forests. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 617–624. IEEE, 2011.

[76] Yifan Feng, Zizhao Zhang, Xibin Zhao, Rongrong Ji, and Yue Gao. Gvcnn: Group-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 264–272, 2018.

[77] Robert Fisher, Jose Santos-Victor, and James Crowley. Caviar: Context aware vision using image-based active recognition, 2005.

[78] Ronald Aylmer Fisher et al. 012: A mathematical examination of the methods of determining the accuracy of an observation by the mean error, and by the mean square error. 1920.

[79] Fabian Flohr, Madalin Dumitru-Guzu, Julian F. P. Kooij, and Dariu Gavrila. A probabilistic framework for joint pedestrian head and body orientation estimation. *IEEE Transactions on Intelligent Transportation Systems*, 16:1872–1882, 2015.

[80] Charles W Fox and Stephen J Roberts. A tutorial on variational bayesian inference. *Artificial intelligence review*, 38(2):85–95, 2012.

[81] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep ordinal regression network for monocular depth estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2002–2011, 2018.

[82] Noriatsu Furukawa, Akio Namiki, Senoo Taku, and Masatoshi Ishikawa. Dynamic regrasping using a high-speed multifingered hand and a high-speed vision system. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 181–187. IEEE, 2006.

[83] Yasutaka Furukawa and Carlos Hernández. Multi-view stereo: A tutorial. *Foundations and Trends® in Computer Graphics and Vision*, 9(1-2):1–148, 2015.

[84] Matheus Gadelha, Subhransu Maji, and Rui Wang. Shape generation using spatially partitioned point clouds. *arXiv preprint arXiv:1707.06267*, 2017.

[85] Yarin Gal. Uncertainty in deep learning. *University of Cambridge*, 1:3, 2016.

[86] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016.

[87] Dani Gamerman and Hedibert F Lopes. *Markov chain Monte Carlo: stochastic simulation for Bayesian inference*. CRC Press, 2006.

[88] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE, 2012.

[89] Xin Geng and Yu Xia. Head pose estimation based on multivariate label distribution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1837–1842, 2014.

[90] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.

[91] Zoubin Ghahramani. Probabilistic machine learning and artificial intelligence. *Nature*, 521(7553):452–459, 2015.

[92] Georgia Gkioxari, Jitendra Malik, and Justin Johnson. Mesh r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9785–9795, 2019.

[93] Felix Glang, Anagha Deshmane, Sergey Prokudin, Florian Martin, Kai Herz, Tobias Lindig, Benjamin Bender, Klaus Scheffler, and Moritz Zaiss. Deepcest 3t: Robust mri parameter determination and uncertainty quantification with neural networks—application to cest imaging of the human brain at 3t. *Magnetic Resonance in Medicine*, 2019.

[94] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323, 2011.

[95] Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American statistical Association*, 102(477):359–378, 2007.

[96] Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007.

[97] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 270–279, 2017.

[98] Steffen Goerke, Yannick Soehngen, Anagha Deshmane, Moritz Zaiss, Johannes Breitling, Philip S Boyd, Kai Herz, Ferdinand Zimmermann, Karel D Klika, Heinz-Peter Schlemmer, et al. Relaxation-compensated apt and rnoe cest-mri of human brain tumors at 3 t. *Magnetic resonance in medicine*, 82(2):622–632, 2019.

[99] Sebastian Gomez-Gonzalez, Gerhard Neumann, Bernhard Schölkopf, and Jan Peters. Adaptation and robust learning of probabilistic movement primitives. *IEEE Transactions on Robotics*, 36(2):366–379, 2020.

[100] Sebastian Gomez-Gonzalez, Sergey Prokudin, Bernhard Schölkopf, and Jan Peters. Real time trajectory prediction using deep conditional generative models. *IEEE Robotics and Automation Letters*, 5(2):970–976, 2020.

[101] Irving John Good. Rational decisions. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 107–114, 1952.

[102] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[103] 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*, pages 2672–2680, 2014.

[104] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[105] Nicolas Gourier, Daniela Hall, and James L Crowley. Estimating face orientation from robust detection of salient facial structures. In *FG Net Workshop on Visual Observation of Deictic Gestures*, volume 6, 2004.

[106] Benjamin Graham and Laurens van der Maaten. Submanifold sparse convolutional networks. *arXiv preprint arXiv:1706.01307*, 2017.

[107] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. *arXiv preprint arXiv:2002.10099*, 2020.

[108] Markus Gross and Hanspeter Pfister. *Point-based graphics*. Elsevier, 2011.

[109] Thibault Groueix, Matthew Fisher, Vladimir Kim, Bryan Russell, and Mathieu Aubry. 3d-coded : 3d correspondences by deep deformation. September 07 2018.

[110] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1321–1330. JMLR. org, 2017.

[111] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3d point clouds: A survey. *arXiv preprint arXiv:1912.12033*, 2019.

[112] Thomas C Hales. A proof of the kepler conjecture. *Annals of mathematics*, pages 1065–1185, 2005.

[113] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. Meshcnn: a network with an edge. *ACM Transactions on Graphics (TOG)*, 38(4):1–12, 2019.

[114] Radoslav Harman and Vladimír Lacko. On decompositional algorithms for uniform sampling from n-spheres and n-balls. *Journal of Multivariate Analysis*, 101(10):2297–2304, 2010.

[115] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[116] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

[117] Matthias Hein, Maksym Andriushchenko, and Julian Bitterwolf. Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 41–50, 2019.

[118] Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. Deep anomaly detection with outlier exposure. *arXiv preprint arXiv:1812.04606*, 2018.

[119] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[120] David A Hirshberg, Matthew Loper, Eric Rachlin, and Michael J Black. Coregistration: Simultaneous alignment and modeling of articulated 3d shape. In *European Conference on Computer Vision*, pages 242–255. Springer, 2012.

[121] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[122] Emiel Hoogeboom, Jorn WT Peters, Taco S Cohen, and Max Welling. Hexaconv. *arXiv preprint arXiv:1803.02108*, 2018.

[123] Liwen Hu, Derek Bradley, Hao Li, and Thabo Beeler. Simulation-ready hair capture. In *Computer Graphics Forum*, volume 36, pages 281–294. Wiley Online Library, 2017.

[124] Dong Huang, Markus Storer, Fernando De la Torre, and Horst Bischof. Supervised local subspace learning for continuous head pose estimation. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2921–2928. IEEE, 2011.

[125] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[126] Zeng Huang, Yuanlu Xu, Christoph Lassner, Hao Li, and Tony Tung. Arch: Animatable reconstruction of clothed humans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3093–3102, 2020.

[127] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[128] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016.

[129] Hanbyul Joo, Tomas Simon, and Yaser Sheikh. Total capture: A 3d deformation model for tracking faces, hands, and bodies. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8320–8329, 2018.

[130] Łukasz Kaiser, Ofir Nachum, Aurko Roy, and Samy Bengio. Learning to remember rare events. *arXiv preprint arXiv:1703.03129*, 2017.

[131] Angjoo Kanazawa, Michael J Black, David W Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7122–7131, 2018.

[132] Asako Kanezaki, Yasuyuki Matsushita, and Yoshifumi Nishida. Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5010–5019, 2018.

[133] Abhishek Kar, Christian Häne, and Jitendra Malik. Learning a multi-view stereo machine. In *Advances in neural information processing systems*, pages 365–376, 2017.

[134] 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*, pages 4401–4410, 2019.

[135] 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*, 2019.

[136] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3907–3916, 2018.

[137] Vahid Kazemi and Josephine Sullivan. One millisecond face alignment with an ensemble of regression trees. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1867–1874, 2014.

[138] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7, 2006.

[139] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems*, pages 5574–5584, 2017.

[140] Kristian Kersting, Christian Plagemann, Patrick Pfaff, and Wolfram Burgard. Most likely heteroscedastic gaussian process regression. In *Proceedings of the 24th international conference on Machine learning*, pages 393–400, 2007.

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

[142] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[143] Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *Advances in neural information processing systems*, pages 3581–3589, 2014.

[144] Roman Klokov and Victor Lempitsky. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 863–872, 2017.

[145] Leif Kobbelt and Mario Botsch. A survey of point-based techniques in computer graphics. *Computers & Graphics*, 28(6):801–814, 2004.

[146] Muhammed Kocabas, Nikos Athanasiou, and Michael J Black. Vibe: Video inference for human body pose and shape estimation. *arXiv preprint arXiv:1912.05656*, 2019.

[147] Nikos Kolotouros, Georgios Pavlakos, Michael J Black, and Kostas Daniilidis. Learning to reconstruct 3d human pose and shape via model-fitting in the loop. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2252–2261, 2019.

[148] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[149] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[150] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in neural information processing systems*, pages 6402–6413, 2017.

[151] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4558–4567, 2018.

[152] Katrin Lasinger, René Ranftl, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *arXiv preprint arXiv:1907.01341*, 2019.

[153] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Advances in Neural Information Processing Systems*, pages 7167–7177, 2018.

[154] Christian Leibig, Vaneeda Allken, Murat Seçkin Ayhan, Philipp Berens, and Siegfried Wahl. Leveraging uncertainty information from deep neural networks for disease detection. *Scientific reports*, 7(1):1–14, 2017.

[155] Marc Levoy and Turner Whitted. *The use of points as a display primitive*. Citeseer, 1985.

[156] Jiaxin Li, Ben M Chen, and Gim Hee Lee. So-net: Self-organizing network for point cloud analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9397–9406, 2018.

[157] Tianye Li, Timo Bolkart, Michael J Black, Hao Li, and Javier Romero. Learning a model of facial shape and expression from 4d scans. *ACM Transactions on Graphics (ToG)*, 36(6):194, 2017.

[158] Shiyu Liang, Yixuan Li, and Rayadurgam Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690*, 2017.

[159] Zhi-Pei Liang and Paul C Lauterbur. *Principles of magnetic resonance imaging: a signal processing perspective*. SPIE Optical Engineering Press, 2000.

[160] Or Litany, Tal Remez, Emanuele Rodolà, Alexander M. Bronstein, and Michael M. Bronstein. Deep functional maps: Structured prediction for dense shape correspondence. *CoRR*, abs/1704.08686, 2017.

[161] Lingjie Liu, Weipeng Xu, Marc Habermann, Michael Zollhoefer, Florian Bernard, Hyeongwoo Kim, Wenping Wang, and Christian Theobalt. Neural human video rendering: Joint learning of dynamic textures and rendering-to-video translation. *arXiv preprint arXiv:2001.04947*, 2020.

[162] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7708–7717, 2019.

[163] Stephen Lombardi, Jason Saragih, Tomas Simon, and Yaser Sheikh. Deep appearance models for face rendering. *ACM Transactions on Graphics (TOG)*, 37(4):1–13, 2018.

[164] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *ACM Transactions on Graphics (TOG)*, 38(4):65, 2019.

[165] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. Smpl: A skinned multi-person linear model. *ACM transactions on graphics (TOG)*, 34(6):248, 2015.

[166] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. Smpl: a skinned multi-person linear model. *ACM Trans. Graph*, 34(6):248:1–248:16, 2015.

[167] Jiwen Lu and Yap-Peng Tan. Ordinary preserving manifold analysis for human age and head pose estimation. *IEEE Transactions on Human-Machine Systems*, 43(2):249–258, 2013.

[168] Qianli Ma, Jinlong Yang, Anurag Ranjan, Sergi Pujades, Gerard Pons-Moll, Siyu Tang, and Michael J Black. Learning to dress 3d people in generative clothing. *arXiv preprint arXiv:1907.13615*, 2019.

[169] David JC MacKay. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.

[170] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[171] Naureen Mahmood, Nima Ghorbani, Nikolaus F Troje, Gerard Pons-Moll, and Michael J Black. Amass: Archive of motion capture as surface shapes. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5442–5451, 2019.

[172] Eric Marchand, Hideaki Uchiyama, and Fabien Spindler. Pose estimation for augmented reality: a hands-on survey. *IEEE transactions on visualization and computer graphics*, 22(12):2633–2651, 2016.

[173] Kanti V Mardia and Peter E Jupp. *Directional statistics*, volume 494. John Wiley & Sons, 2009.

[174] Riccardo Marin, Simone Melzi, Emanuele Rodolà, and Umberto Castellani. Farm: Functional automatic registration method for 3d human bodies. *CoRR*, abs/1807.10517, 2018.

[175] Ricardo Martin-Brualla, Rohit Pandey, Shuoran Yang, Pavel Pidlypenskyi, Jonathan Taylor, Julien Valentin, Sameh Khamis, Philip Davidson, Anastasia Tkach, Peter Lincoln, et al. Lookingood: enhancing performance capture with real-time neural re-rendering. *arXiv preprint arXiv:1811.05029*, 2018.

[176] Francisco Massa, Renaud Marlet, and Mathieu Aubry. Crafting a multi-task cnn for viewpoint estimation. *arXiv preprint arXiv:1609.03894*, 2016.

[177] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928. IEEE, 2015.

[178] Randolf Menzel. The honeybee as a model for understanding the basis of cognition. *Nature Reviews Neuroscience*, 13(11):758–768, 2012.

[179] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. *arXiv preprint arXiv:1812.03828*, 2018.

[180] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993, 2018.

[181] Gyeongsik Moon, Ju Yong Chang, and Kyoung Mu Lee. V2v-posenet: Voxel-to-voxel prediction network for accurate 3d hand and human pose estimation from a single depth map. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5079–5088, 2018.

[182] Katharina Mülling, Jens Kober, and Jan Peters. A biomimetic approach to robot table tennis. *Adaptive Behavior*, 19(5):359–376, 2011.

[183] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[184] Erik Murphy-Chutorian, Anup Doshi, and Mohan Manubhai Trivedi. Head pose estimation for driver assistance systems: A robust algorithm and experimental evaluation. In *Intelligent Transportation Systems Conference, 2007. ITSC 2007. IEEE*, pages 709–714. IEEE, 2007.

[185] Erik Murphy-Chutorian and Mohan Manubhai Trivedi. Head pose estimation in computer vision: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 31(4):607–626, 2009.

[186] Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. Hybrid models with deep and invertible features. *arXiv preprint arXiv:1902.02767*, 2019.

[187] Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

[188] Tristan Needham. A visual explanation of Jensen's inequality. *The American mathematical monthly*, 100(8):768–771, 1993.

[189] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[190] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *2011 IEEE International Symposium on Mixed and Augmented Reality*, pages 127–136. IEEE, 2011.

[191] Thu Nguyen-Phuoc, Chuan Li, Lucas Theis, Christian Richardt, and Yong-Liang Yang. Hologan: Unsupervised learning of 3d representations from natural images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7588–7597, 2019.

[192] Alexandru Niculescu-Mizil and Rich Caruana. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 625–632, 2005.

[193] David A Nix and Andreas S Weigend. Estimating the mean and variance of the target probability distribution. In *Proceedings of 1994 ieee international conference on neural networks (ICNN'94)*, volume 1, pages 55–60. IEEE, 1994.

[194] Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016.

[195] Jean Marc Odobez. IDIAP Head Pose Database. `https://www.idiap.ch/dataset/headpose`.

[196] Stephen M Omohundro. *Five balltree construction algorithms*. International Computer Science Institute Berkeley, 1989.

[197] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016.

[198] Margarita Osadchy, Yann Le Cun, and Matthew L Miller. Synergistic face detection and pose estimation with energy-based models. *Journal of Machine Learning Research*, 8(May):1197–1215, 2007.

[199] Stanley Osher and James A Sethian. Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations. *Journal of computational physics*, 79(1):12–49, 1988.

[200] M. Ozuysal, V. Lepetit, and P. Fua. Pose estimation for category specific multiview object localization. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 778–785, 2009.

[201] Onur Özyeşil, Vladislav Voroninski, Ronen Basri, and Amit Singer. A survey of structure from motion*. *Acta Numerica*, 26:305–364, 2017.

[202] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. *arXiv preprint arXiv:1901.05103*, 2019.

[203] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[204] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed AA Osman, Dimitrios Tzionas, and Michael J Black. Expressive body capture: 3d hands, face, and body from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10975–10985, 2019.

[205] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[206] Bojan Pepik, Peter Gehler, Michael Stark, and Bernt Schiele. 3d2pm – 3d deformable part models. In *Proceedings of the European Conference on Computer Vision (ECCV)*, Lecture Notes in Computer Science, pages 356–370, Firenze, October 2012. Springer.

[207] Bojan Pepik, Michael Stark, Peter Gehler, and Bernt Schiele. Teaching 3d geometry to deformable part models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3362 –3369, Providence, RI, USA, June 2012. IEEE. oral presentation.

[208] Hanspeter Pfister, Matthias Zwicker, Jeroen Van Baar, and Markus Gross. Surfels: Surface elements as rendering primitives. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 335–342, 2000.

[209] Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically based rendering: From theory to implementation*. Morgan Kaufmann, 2016.

[210] John Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

[211] Patrick Poirson, Phil Ammirato, Cheng-Yang Fu, Wei Liu, Jana Kosecka, and Alexander C Berg. Fast single shot detection and pose estimation. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 676–684. IEEE, 2016.

[212] Gerard Pons-Moll, Javier Romero, Naureen Mahmood, and Michael J. Black. Dyna: A model of dynamic human shape in motion. *ACM Transactions on Graphics, (Proc. SIGGRAPH)*, 34(4):120:1–120:14, August 2015.

[213] Vittal Premachandran, Daniel Tarlow, and Dhruv Batra. Empirical minimum Bayes risk prediction: How to extract an extra few % performance from vision models with just three more parameters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1043–1050, 2014.

[214] RI'Anson Price, N Dulex, N Vial, C Vincent, and Christoph Grüter. Honeybees forage more successfully without the "dance language" in challenging environments. *Science advances*, 5(2):eaat0450, 2019.

[215] Sergey Prokudin, Peter Gehler, and Sebastian Nowozin. Deep directional statistics: Pose estimation with uncertainty quantification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 534–551, 2018.

[216] Sergey Prokudin, Daniel Kappler, Sebastian Nowozin, and Peter Gehler. Learning to filter object detections. In *German Conference on Pattern Recognition*, pages 52–62. Springer, 2017.

[217] Sergey Prokudin, Christoph Lassner, and Javier Romero. Efficient learning on point clouds with basis point sets. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4332–4341, 2019.

[218] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 1(2):4, 2017.

[219] Charles R Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5648–5656, 2016.

[220] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5099–5108, 2017.

[221] Joaquin Quinonero-Candela, Carl Edward Rasmussen, Fabian Sinz, Olivier Bousquet, and Bernhard Schölkopf. Evaluating predictive uncertainty challenge. In *Machine Learning Challenges Workshop*, pages 1–27. Springer, 2005.

[222] Anurag Ranjan, Timo Bolkart, Soubhik Sanyal, and Michael J Black. Generating 3d faces using convolutional mesh autoencoders. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 704–720, 2018.

[223] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

[224] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3577–3586, 2017.

[225] Gernot Riegler, Ali Osman Ulusoy, Horst Bischof, and Andreas Geiger. Octnetfusion: Learning depth fusion from data. In *2017 International Conference on 3D Vision (3DV)*, pages 57–66. IEEE, 2017.

[226] Michal Rivlin, Judith Horev, Ilan Tsarfaty, and Gil Navon. Molecular imaging of tumors and metastases using chemical exchange saturation transfer (cest) mri. *Scientific reports*, 3(1):1–7, 2013.

[227] Kathleen M. Robinette, Sherri Blackwell, Hein Daanen, Mark Boehmer, Scott Fleming, Tina Brill, Dãvid Hoeferlin, and Dennis Burnsides. Civilian American and European Surface Anthropometric Resource (CAESAR) final report. Technical report, US Air Force Laboratory, 2002.

[228] Javier Romero, Dimitrios Tzionas, and Michael J Black. Embodied hands: Modeling and capturing hands and bodies together. *ACM Transactions on Graphics (ToG)*, 36(6):245, 2017.

[229] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[230] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[231] Szymon Rusinkiewicz and Marc Levoy. Qsplat: A multiresolution point rendering system for large meshes. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 343–352, 2000.

[232] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[233] Shunsuke Saito, Tomas Simon, Jason Saragih, and Hanbyul Joo. Pifuhd: Multi-level pixel-aligned implicit function for high-resolution 3d human digitization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 84–93, 2020.

[234] Kripasindhu Sarkar, Basavaraj Hampiholi, Kiran Varanasi, and Didier Stricker. Learning 3d shapes as multi-layered height-maps using 2d convolutional networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 71–86, 2018.

[235] Silvio Savarese and Li Fei-Fei. 3d generic object categorization, localization and pose estimation. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[236] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[237] James Albert Sethian. *Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*, volume 3. Cambridge university press, 1999.

[238] Yiru Shen, Chen Feng, Yaoqing Yang, and Dong Tian. Mining point cloud local structures by kernel correlation and graph pooling. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4548–4557, 2018.

[239] A Dean Sherry and Mark Woods. Chemical exchange saturation transfer contrast agents for magnetic resonance imaging. *Annu. Rev. Biomed. Eng.*, 10:391–411, 2008.

[240] Aliaksandra Shysheya, Egor Zakharov, Kara-Ali Aliev, Renat Bashirov, Egor Burkov, Karim Iskakov, Aleksei Ivakhnenko, Yury Malkov, Igor Pasechnik, Dmitry Ulyanov, Alexander Vakhitov, and Victor Lempitsky. Textured neural avatars. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[241] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[242] Teera Siriteerakul. Advance in head pose estimation from low resolution images: A review. *International Journal of Computer Science Issues*, 9(2), 2012.

[243] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhofer. Deepvoxels: Learning persistent 3d feature embeddings. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2437–2446, 2019.

[244] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *Advances in Neural Information Processing Systems*, 2019.

[245] J David Smith, Wendy E Shields, and David A Washburn. The comparative psychology of uncertainty monitoring and metacognition. *Behavioral and brain sciences*, 26(3):317–339, 2003.

[246] Noah Snavely, Steven M Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3d. In *ACM Siggraph 2006 Papers*, pages 835–846. 2006.

[247] Noah Snavely, Steven M Seitz, and Richard Szeliski. Modeling the world from internet photo collections. *International journal of computer vision*, 80(2):189–210, 2008.

[248] Jasper Snoek, Yaniv Ovadia, Emily Fertig, Balaji Lakshminarayanan, Sebastian Nowozin, D Sculley, Joshua Dillon, Jie Ren, and Zachary Nado. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems*, pages 13969–13980, 2019.

[249] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems*, pages 3483–3491, 2015.

[250] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1746–1754, 2017.

[251] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[252] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*, 2019.

[253] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953, 2015.

[254] Hao Su, Charles R Qi, Yangyan Li, and Leonidas J Guibas. Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2686–2694, 2015.

[255] Leonard Susskind. The world as a hologram. *Journal of Mathematical Physics*, 36(11):6377–6396, 1995.

[256] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, volume 4, page 12.

[257] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[258] Qingyang Tan, Lin Gao, Yu-Kun Lai, and Shihong Xia. Variational autoencoders for deforming 3d mesh models. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5841–5850, 2018.

[259] Maxim Tatarchenko, Jaesik Park, Vladlen Koltun, and Qian-Yi Zhou. Tangent convolutions for dense prediction in 3d. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3887–3896, 2018.

[260] Ayush Tewari, Ohad Fried, Justus Thies, Vincent Sitzmann, Stephen Lombardi, Kalyan Sunkavalli, Ricardo Martin-Brualla, Tomas Simon, Jason Saragih, Matthias Nießner, et al. State of the art on neural rendering. *arXiv preprint arXiv:2004.03805*, 2020.

[261] Lucas Theis, Aäron van den Oord, and Matthias Bethge. A note on the evaluation of generative models. *arXiv preprint arXiv:1511.01844*, 2015.

[262] Theoharis Theoharis, Georgios Papaioannou, and Evaggelia-Aggeliki Karabassi. The magic of the z-buffer: A survey. 2001.

[263] Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures. *ACM Transactions on Graphics (TOG)*, 38(4):1–12, 2019.

[264] Justus Thies, Michael Zollhofer, Marc Stamminger, Christian Theobalt, and Matthias Nießner. Face2face: Real-time face capture and reenactment of rgb videos. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2387–2395, 2016.

[265] Diego Tosato, Mauro Spera, Marco Cristani, and Vittorio Murino. Characterizing humans on riemannian manifolds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1972–1984, 2013.

[266] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.

[267] Shubham Tulsiani and Jitendra Malik. Viewpoints and keypoints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1510–1519, 2015.

[268] Peter CM Van Zijl and Nirbhay N Yadav. Chemical exchange saturation transfer (cest): what is in a name and what isn't? *Magnetic resonance in medicine*, 65(4):927–948, 2011.

[269] Karl Von Frisch. The dance language and orientation of bees. 1967.

[270] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-cnn: Octree-based convolutional neural networks for 3d shape analysis. *ACM Transactions on Graphics (TOG)*, 36(4):72, 2017.

[271] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8798–8807, 2018.

[272] Shih-En Wei, Jason Saragih, Tomas Simon, Adam W Harley, Stephen Lombardi, Michal Perdoch, Alexander Hypes, Dawei Wang, Hernan Badino, and Yaser Sheikh. Vr facial animation via multiview image translation. *ACM Transactions on Graphics (TOG)*, 38(4):1–16, 2019.

[273] Johannes Windschuh, Moritz Zaiss, Jan-Eric Meissner, Daniel Paech, Alexander Radbruch, Mark E Ladd, and Peter Bachert. Correction of b1-inhomogeneities for relaxation-compensated cest imaging at 7 t. *NMR in biomedicine*, 28(5):529–537, 2015.

[274] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[275] Yu Xiang, Roozbeh Mottaghi, and Silvio Savarese. Beyond pascal: A benchmark for 3d object detection in the wild. In *Applications of Computer Vision (WACV), 2014 IEEE Winter Conference on*, pages 75–82. IEEE, 2014.

[276] Yu Xiang, Roozbeh Mottaghi, and Silvio Savarese. Beyond pascal: A benchmark for 3d object detection in the wild. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2014.

[277] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in neural information processing systems*, pages 3391–3401, 2017.

[278] Moritz Zaiss, Johannes Windschuh, Daniel Paech, Jan-Eric Meissner, Sina Burth, Benjamin Schmitt, Philip Kickingereder, Benedikt Wiestler, Wolfgang Wick, Martin Bendszus, et al. Relaxation-compensated cest-mri of the human brain at

7 t: unbiased insight into noe and amide signal changes in human glioblastoma. *Neuroimage*, 112:180–188, 2015.

[279] Egor Zakharov, Aliaksandra Shysheya, Egor Burkov, and Victor Lempitsky. Few-shot adversarial learning of realistic neural talking head models. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9459–9468, 2019.

[280] Andy Zeng, Shuran Song, Matthias Nießner, Matthew Fisher, Jianxiong Xiao, and Thomas Funkhouser. 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1802–1811, 2017.

[281] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P Xing, Laurent El Ghaoui, and Michael I Jordan. Theoretically principled trade-off between robustness and accuracy. *arXiv preprint arXiv:1901.08573*, 2019.

[282] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 586–595, 2018.

[283] Chaoqiang Zhao, Qiyu Sun, Chongzhen Zhang, Yang Tang, and Feng Qian. Monocular depth estimation based on deep learning: An overview. *arXiv preprint arXiv:2003.06620*, 2020.

[284] Yongsheng Zhao, Rong Xiong, and Yifeng Zhang. Model based motion state estimation and trajectory prediction of spinning ball for ping-pong robots using expectation-maximization algorithm. *Journal of Intelligent & Robotic Systems*, 87(3-4):407–423, 2017.

[285] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1851–1858, 2017.

[286] Yi Zhou, Liwen Hu, Jun Xing, Weikai Chen, Han-Wei Kung, Xin Tong, and Hao Li. Hairnet: Single-view hair reconstruction using convolutional neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 235–251, 2018.

[287] Xiangxin Zhu and Deva Ramanan. Face detection, pose estimation, and landmark localization in the wild. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2879–2886. IEEE, 2012.

[288] Silvia Zuffi and Michael J Black. The stitched puppet: A graphical model of 3d human shape and pose. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3537–3546, 2015.

[289] Silvia Zuffi, Angjoo Kanazawa, David W Jacobs, and Michael J Black. 3d menagerie: Modeling the 3d shape and pose of animals. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6365–6373, 2017.

[290] Barton Zwiebach. *A first course in string theory*. Cambridge university press, 2004.