DEGREE PROJECT IN TECHNOLOGY,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2020

Unsupervised 3D Human Pose Estimation

KTH Thesis Report
Draft for final thesis meeting

Sri Datta Budaraju

KTH ROYAL INSTITUTE OF TECHNOLOGY
ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Authors

Sri Datta Budaraju <budaraju@kth.se>
School of Electrical Engineering and Computer Science
KTH Royal Institute of Technology

Place for Project

Stockholm, Sweden
Stuttgart, Germany

Examiner

Danica Kragic Jensfelt
Stockholm, Sweden
KTH Royal Institute of Technology

Supervisor

Hedvig Kjellström
Stockholm, Sweden
KTH Royal Institute of Technology

Supervisor - Host

Arij Bouaziz

Stuttgart, Germany

Mercedes-Benz AG, Research and Development

Abstract

YET TO BE WRITTEN

This is a template for writing thesis reports for the ICT school at KTH. I do not own any of the images provided in the template and this can only be used to submit thesis work for KTH.

The report needs to be compiled using XeLaTeX as different fonts are needed for the project to look like the original report. You might have to change this manually in overleaf.

This template was created by Hannes Rabo <hannes.rabo@gmail.com or hrabo@kth.se> from the template provided by KTH. You can send me an email if you need help in making it work for you.

Write an abstract. Introduce the subject area for the project and describe the problems that are solved and described in the thesis. Present how the problems have been solved, methods used and present results for the project. Use probably one sentence for each chapter in the final report.

The presentation of the results should be the main part of the abstract. Use about ½ A4-page. English abstract

Keywords

Template, Thesis, Keywords ...

Abstract

YET TO BE WRITTEN

Svenskt abstract Svensk version av abstract – samma titel på svenska som på engelska.

Skriv samma abstract på svenska. Introducera ämnet för projektet och beskriv problemen som löses i materialet. Presentera

Nyckelord

Kandidat examensarbete, ...

Acknowledgements

YET TO BE WRITTEN

Write a short acknowledgements. Don't forget to give some credit to the examiner and supervisor.

Acronyms

AR/VR Augmented Reality/Virtual Reality

BCE Binary Cross Entropy

β-VAE Beta Variational Auto-Encoder

FC Fully Connected

FOV Field of View

GAN Generative Adversarial Network

HPE Human Pose Estimation

KLD Kullback–Leibler Divergence

MPJPE Mean Per Joint Position Estimate

POV Point of View

ReLU Rectified Linear Unit

SOTA State-of-The-Art

UMAP Uniform Manifold Approximation and Projection

VAE Variational Auto-Encoder

Contents

Chapter 1

Introduction

With rapid advancements in deep learning facilitated by the developments in computational hardware, there has been tremendous growth in computer vision research and its applications [3]. One of the major tasks of computer vision that is required for real-world applications is to perceive and understand dynamic objects and more importantly humans.

Human Pose Estimation, also referred to as Human Pose Estimation (HPE), is a fundamental computer vision task that also forms a basis for advanced tasks such as human action and gesture recognition as well as human motion prediction. HPE is defined as the localization of human joints (also known as keypoints, including head, eyes, ears, nose, etc) mainly in images and videos in either a 2D or 3D coordinate space. The widely available and used data like images and videos are 2-dimensional data and lack spatial information which is crucial for most of the applications like autonomous driving, Augmented Reality/Virtual Reality (AR/VR), social robots, etc. Hence the focus of this thesis is on 3D HPE.

1.1 Background

There has been a lot of research done in 3D human pose estimation and more advancements have been made in the past few years leveraging the power of deep learning. The current research explores various ways to solve the task using RGB/Depth image channels, 2D poses, 3D poses, multi-view, and sequential images/videos.

Most of them are supervised learning approaches that require 3D ground truth poses that can only be acquired using physical sensors. Supervised learning methods that learn 3D pose from images, follow a complex cascading approach with 2D poses in some form is an intermediate output. And other learning approaches mostly make use of images from multiple views to estimate the 3D pose.

Assuming 2D poses are obtained from the State-of-The-Art (SOTA) models specialized on 2D HPE models, some works focus on estimating the 3D pose from these 2D poses instead of images. Such networks are called *lifting* networks. These lifting networks can be simple without requiring computationally expensive convolutional layers as it only needs to learn the features of 2D poses that are low dimensional compared to images. However since 2D poses are naturally obtained by projecting 3D poses to a plane, it is an inverse problem. Also, there are multiple feasible 3D poses that when the projected result in the same 2D pose. Thus making the task of lifting 2D-to-3D, a *ill-posed inverse* problem due to its inherent ambiguity.

Non-supervised (Weakly/Self/Unsupervised) learning regimes, that are less dependent on 3D pose ground truth, have also gained traction in recent years. Weakly supervised approaches use 3D ground truth indirectly by generating more 2D poses from more views or, for training a discriminator network of a Generative Adversarial Network (GAN). While unsupervised learning (self-supervised) approaches do not use 3D ground truth poses in any shape or form. Many of the deep learning techniques that have already improved the results in other computer vision tasks are yet to be explored in 3D HPE.

1.2 Problem

How can we learn a strong visual representation of the data to tackle the task of 3D HPE? Could data as its own supervisory goal (self-supervision) resolve the ambiguities of the pose estimation?

1.3 Goal

The main aim of the thesis is to investigate unsupervised learning approaches and 2D-to-3D lifting methods that could help tackle the challenges in scaling 3D HPE to the

real-world.

Improvements in the aspects of ease of training procedure i.e requiring less data or less labor-intense labeling, inference speed, and most importantly accuracy is important and will directly impact its super tasks such as action and gesture recognition, motion prediction and intention, behavior prediction.

1.4 Benefits, Ethics and Sustainability

Human Pose Estimation plays a very important role to enable autonomous vehicles and robots to safely interact with humans. It also plays a vital role in developing higher dimensional communication platforms with AR/VR. It is crucial for surveillance systems to ensure public safety. However such important technologies are only as good as the intentions of its users. Mass surveillance of citizens by their governments is a matter of debate.

1.5 Methodology

The problem of 3D HPE has 3 aspects to be addressed and explored.

The neural network: The architecture and the kind of neural network to be used. 3D poses can be predicted using a regular linear neural network or using various other architectures like autoencoders. These models can use linear, convolutional, and graph layers to learn features. This thesis focuses on investigating the merit in using architectures like Variational Auto-Encoders (VAEs) to solve the 3D HPE within the context of leveraging probabilistic inference models, as a deterministic approach for an inherently ill-posed problem is not ideal.

The learning task: The model could either learn to directly predict the 3D coordinates of the keypoints or learn structural parameters that could model a 3D pose. The thesis only explores the former task.

The learning technique (or the cost): The model can be either trained by directly comparing the predicted 3D pose and the ground truth 3D pose thus requiring 3D annotations, or by projecting the prediction back to 2D to compare with the input

(requires only 2D annotations that could be acquired from SOTA in 2D HPE) and use a different technique to ensure the correctness of pose in 3D. Adversarial training and self-supervision techniques have also given promising results in the last couple of years. The method proposed in the thesis is designed to learn 3D from 2D poses alone in an unsupervised-adversarial learning fashion after the capabilities of the method under supervised settings being verified.

The prime motivation behind the design choices is to address the challenges in scaling up 3D HPE to real-world.

1.6 Stakeholders

Daimler's 'Environment Perception for Autonomous Driving' R&D team in Stuttgart, Germany, conducts cutting-edge research in the field of Computer vision and Deep Learning to improve the State-Of-The-Art and to make Autonomous Driving a reality. This thesis is part of the team's on-going research in understanding the human state, motion, and behavior, which would help autonomous cars better perceive, understand, and interact with humans. Daimler/Mercedes-Benz autonomous cars try to understand humans both, inside and outside the car and HPE is a critical element to accomplish this task.

The question is also of interest to the research area of Human State/Action Recognition in specific and also to areas of computer graphics to model humans in 3D space. Hence it is beneficial to various areas that try to understand and interact with humans. The scientific communities in the areas of Autonomous Driving, AR/VR, Motion Capture, Computer Graphics, and Human-Robot interaction could be interested in the contributions of this thesis.

1.7 Delimitations

This thesis focuses only on 3D pose estimation and not the intermediate 2D pose. Data collection is not part of the thesis study and uses only publicly available, widely used, and benchmarked datasets.

1.8 Outline

The theoretical knowledge required to understand the details of the thesis is presented next in chapter **??**, Theoretical Background. This entails the explanation of some preliminary concepts **??**, research area introduction **??**, where the vast literature related to HPE is summarized, and the highlights of all the works **??** that are closely related to the thesis.

The background is followed by chapter **??**, Data, providing details of the datasets used along with some visualizations. This chapter also explains the 3D projective geometry concepts required to understand the pre and post-processing steps the data undergoes.

The method, chapter 2, describes the proposed approach in detail. This covers the components of the proposed architecture, the motivation behind the choices, the training and validation procedure, and other details that help reproduction.

The results are analysed and discussed in chapter 4 and conclusions are presented in chapter 5.

Chapter 2

Method

This chapter entails the details of the neural network such as the architecture, loss/objective function along with the training and evaluation procedures of the proposed method. Other choices such as activation functions, optimizers, initialization, etc that completes the architecture are presented later in chapter 3 Experiments, along with the reasoning behind the choices and the compilation of essential components required to reproduce the entire network.

2.1 Architecture

To predict root-relative 3D pose solely using 2D poses, a hybrid network using Variational Auto-Encoder (VAE) and Generative Adversarial Network (GAN) as discussed in **??** is employed. In contrast to the VAE-GAN hybrid proposed by [9], the GAN loss gradient is propagated to both the decoder and the encoder. In other words, the VAE as a whole is considered as the generator network. The overall architecture is composed of 3 models Encoder, Decoder, and Discriminator as illustrated in Fig 2.1.2. This section elaborates on each of the models and how they are interconnected.

2.1.1 Encoder

Adding to the explanation of VAE in **??**, the role of the encoder Q, is to take a 2D pose $\mathbf{x}_i = (x_i, y_i)$ with its root located at the joint and of upper body length of c units as an input. Where $i = 1...J$ and J denote the number of joints of the pose. And output the corresponding embedding in a d dimensional latent space in the form of mean μ and

standard deviation σ of each dimension.

$$Q_{\theta_q}(\mathbf{x}) = \mu, \sigma \tag{2.1}$$

Where θ_q denotes the learned parameters of the encoder during training. This encoder is composed of an upsampling layer that scales the $2 \cdot J$ dimensional input to match the number of hidden neurons h of the encoding module. The encoding module q is made of n residual block composed of 2 Fully Connected (FC) layers following the related works, to allow comparison. This residual block structure is repeated for the decoder and the discriminator. The encoding block is followed by 2 FC (linear) layers that downsample the hidden representation of dimension h to match the latent space dimension d. The output of the two downsampling layers represents the mean and standard deviation respectively. However, in practice, the encoder is designed to predict log-variance instead of the standard deviation to have a better distribution of values and gradient. The layers of the encoder, decoder, and the discriminator models are kept similar for simplicity and as done by the related works. The residual blocks used in all the 3 models are identical and is illustrated in Fig 2.1.1.

Both FC layers of the residual block are followed by batch normalization, activation, and dropout layers of the same dimensionality. While other **FC!** (**FC!**) layers are only followed by an activation function. The linear layers in the residual blocks of the VAE contain weights but do not have the bias component.

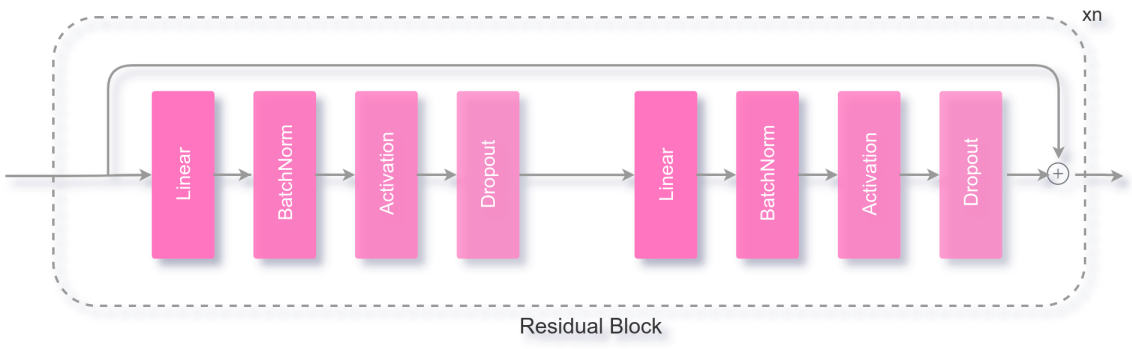

Figure 2.1.1: Illustration of the residual block made of 2 Linear layers of the same input and output dimensionality. The intermediate BatchNorm, activation and dropout layers maintain the same dimensionality. These blocks are repeated n times based on the configuration of the model.

2.1.2 Decoder

The decoder P takes the 2D pose embedding z derived from the mean and log-variance predicted by the encoder and estimates the corresponding 3D pose $\mathbf{Y}_i = (x'_i, y'_i, z'_i)$. Similar to the encoder, $i = 1...J$ and J denote the number of joints of the pose. The reparametrization trick is used to make the process differentiable and induce variance. This is done by scaling the standard deviation obtained from the log-variance with a random sample ϵ from a unit gaussian distribution $\mathcal{N}(0, 1)$. The sum of the scaled standard deviation and the mean gives the sample z.

$$z = \mu + \sigma \odot \epsilon$$
$$P_{\theta_p}(z) = \mathbf{Y} \tag{2.2}$$

Where θ_p represents the learned parameters of the decoding network. Similar to the encoder, the decoder consists of an upsampling layer that scales the sample z of dimensionality d to match the number of hidden neurons d in the decoding module. The decoding module is identical to the encoding module and consists of n residual blocks similar to that of the encoder. This decoding module is followed by a FC layer to downsample the neurons to predict the 3D pose of dimensions $3 \cdot J$. As the goal is to predict root-relative pose with the root at the origin, the output joints are considered to be relative to the origin (0, 0, 0).

Since the predicted 3D pose is to be projected to back to the 2D pose to form the reconstruction loss, the distance between the head and the pelvis joint should approximately be of unit length as explained in **??**. To achieve this, a Tanh activation function is used at the downsampling/output layer to obtain the predicted 3D pose (all joints) in the range [-1, 1].

2.1.3 Reprojection

Referring to the camera modeling section **??**, the 3D that corresponds to the pre-processed 2D pose is located at a fixed distance c from the camera. The 3D pose is considered to be predicted relative to the origin for simplicity and symmetry and is to be translated c units aways from the camera. Another required adjustment is the scale. The range of coordinates predicted is in the range [-1,1]. But the length of the lower half of the pose can be longer than the upper half and usually is the case. Hence the

predicted 3D pose is scaled by a factor of 1.3, which is the ratio of the mean length of the upper and lower halves. This is to enforce that the length of the upper half is 1 unit while covering the true range of the lower half and get the best 2D re-projection.

This scaled 3D pose prediction is directly projected from the same point of view to get a 2D projection that corresponds to the input 2D pose. This similarity is used as the reconstruction loss for constrained optimization of the VAE.

$$\begin{aligned} \mathbf{Y}' &= \mathbf{Y} * 1.3 + (0, 0, c) \\ \mathbf{y} &= PP(\mathbf{Y}') \end{aligned} \tag{2.3}$$

Where PP refers to perspective projection and $\mathbf{y}$ denotes the 2D projection that corresponds to the input 2D pose. The scaled 3D pose is also randomly rotated by uniformly sampling an azimuth angle from the range $[-\pi, \pi]$ and elevation range in the range $[-\pi/9, \pi/9]$. The 2D pose obtained from the projection of this rotated 3D pose gives a different point of view, a ***novel view*** of the 3D pose that is different from the views of the subject in the dataset. This novel view is used for unconstrained optimization of the VAE using the discriminator. The elevation angle has been followed by other works but is not observed to have any visible impact.

$$\begin{aligned} \mathbf{Y}'_{rot} &= \mathbf{R} * (\mathbf{Y} * 1.3) + (0, 0, c) \\ \tilde{\mathbf{y}} &= PP(\mathbf{Y}'_{rot}) \end{aligned} \tag{2.4}$$

Where $\mathbf{R}$ refers to the rotation matrix formed using the randomly sampled angles. $\tilde{\mathbf{y}}$ refers to novel view of the 3D pose.

2.1.4 Discriminator

The discriminator takes the novel view 2D pose of the predicted 3D pose $\tilde{\mathbf{y}}$ along with the real 2D poses $\mathbf{x}$ from the dataset as the input and classifies which 2D pose belongs to which category, real or fake (generated novel view). The real and novel views need not correspond to the same pose, the goal of the discriminator is to learn the general ability to distinguish the 2D poses from the dataset (real) from the predicted poses

(fake).

$$D_{\theta_d}(\tilde{\mathbf{y}} \cup \mathbf{x}) = \left\{p^i_{class} | i \in 1, ..n(\tilde{\mathbf{y}} \cup \mathbf{x})\right\}$$
$$where, 0 \leq p_{class} \leq 1 \tag{2.5}$$

Where θ_d refers to the learned parameters of the discriminator and p_{class} here denotes the probability of the pose's class, with 1 being real and 0 being fake. The discriminator mimicking the encoder and the decoder upsamples the $2 \cdot J$ dimensional novel 2D pose to h dimensions of the main module. The main module just as other models are composed of n residual blocks. However, the batch normalization layer is removed in the discriminator's residual blocks following standard practices. It is important to note that n need not be the same for all the 3 models. The learned features are downsampled to predict the probability of the pose being real or fake. The sigmoid activation function is used at this last layer as the required values are in the range [0, 1].

2.2 Loss Functions

The loss functions used in this VAE-GAN are similar to the ones explained in **??**. The hybrid training proposed in [9] uses the similarity between the prediction and input estimated by the discriminator *replacing* the element-wise loss. While the proposed approach exploits the element-wise loss and the similarity loss by using them on different transformations of the prediction (direct view 2D and novel view 2D). The formulation of each loss function of the hybrid with respect to the human pose data is as follows.

2.2.1 Reconstruction Loss

Since the goal of the proposed method is to learn 3D pose without requiring any of such training data, the approach uses 2D pose as its supervisory data. This self-supervision is achieved by reprojecting the 3D pose in the same view to retrieve the 2D pose that should ideally be the same as the input 2D pose as discussed in the Reprojection 2.1.3 part of the previous section. The objective of the model here is to converge the distance (element-wise) between the predicted 2D pose $\mathbf{y}$ and the input 2D pose $\mathbf{x}$. L1 loss is

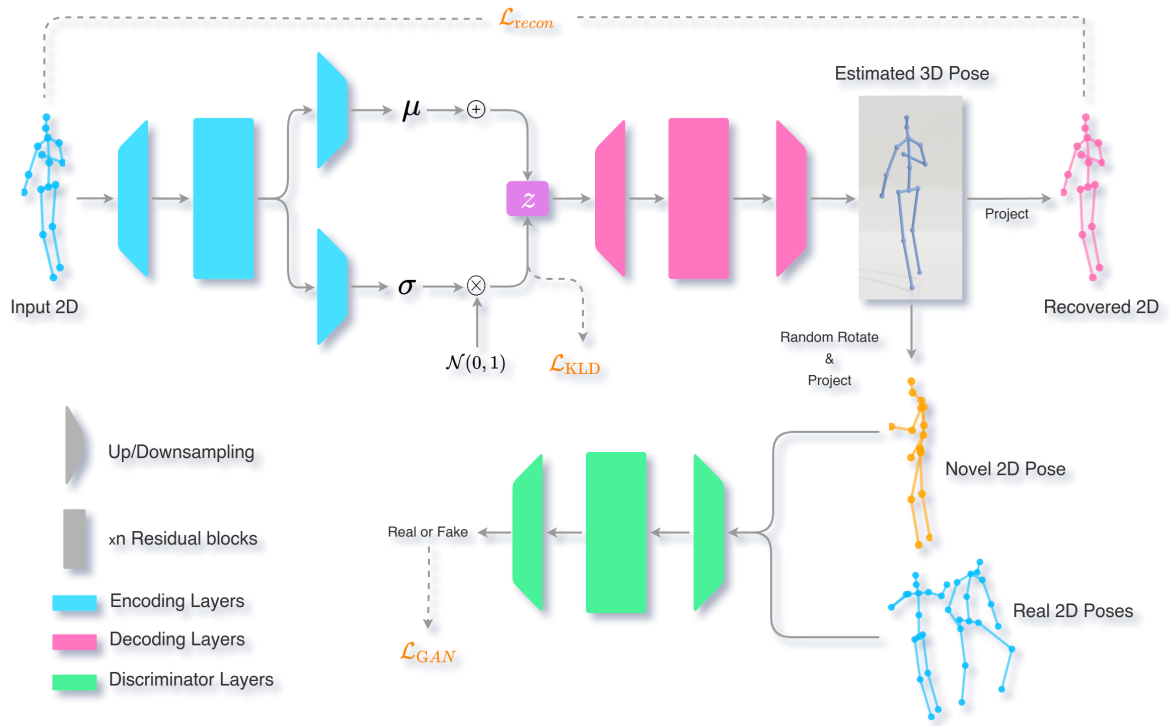

Figure 2.1.2: Illustration of the neural architecture of the proposed method. The network components in blue, encode the 2D pose to a latent representation in terms of mean μ and standard deviation σ of the distribution. While the components in pink, sample z from this latent space and decode the corresponding 3D pose. The 3D pose is projected directly to 2D space for a constrained optimization using the input 2D pose in the camera view and randomly rotated and projected for unconstrained optimization using the discriminator D in a novel angle. The data that contributes to the loss function in orange, is mapped with a dotted line.

used to measure the element-wise reconstruction error.

$$\begin{aligned} \mathcal{L}_{\text{recon}} &= L1\_loss(\mathbf{x}, \mathbf{y}) \\ &= \sum_{i=1}^{J} |\mathbf{x}_i - \mathbf{y}_i| \end{aligned} \tag{2.6}$$

The training of the hybrid network with 3 different losses is not very stable. For the reconstruction loss, there is an absolute target and metric for learning. So this is a constrained optimization and directly corresponds to the error of the 3D pose. Hence it is desired to give high priority and keep this loss relatively stable and low. All the 3 loss terms are weighted to be summed to one final object term that is used to update the network. The weight coefficient λ_{recon} is kept constant and relatively higher than

the other two that are introduced next.

2.2.2 Latent Prior Loss

The proposed hybrid though named VAE-GAN, actually uses a Beta Variational Auto-Encoder (β-VAE), it is not highlighted as the only difference is having a variable coefficient rather than fixing it to 1. Referring to the explanation of the β-VAE in **??**, the higher the weight of the Kullback–Leibler Divergence (KLD) term the better is the disentanglement or clustering of the embeddings are in the latent space. There is no magic β that gives the best reconstruction and representation. This is a trade-off and the value is chosen based on the requirements. Though both are the priorities of this work, more importance is given to the reconstruction as it is quantitative. It is important to note that the weight coefficient λ_{KLD} to balance the loss terms are separated for the β term to have better control when conducting experiments with different scheduling strategies for the β term. A cyclic beta technique similar to [5] is used to handle the vanishing KLD problem, where β is increased from 0 to true value every b epochs and observed to improve training.

$$\begin{aligned} \mathcal{L}_{\text{prior}} = \mathcal{L}_{\text{KLD}} &= -0.5 * \sum(1 + log(\sigma^2) - \mu^2 - \sigma^2) \\ &= -0.5 * \sum(1 + logvar - \mu^2 - e^{logvar}) \end{aligned} \tag{2.7}$$

The $logvar$ term here is the output of one of the downsampling layers of the encoder in practice. The terms denoted in the equation are per sample and in practice, the losses are averaged and normalized based on the dimensionality. The normalization is included in their λ coefficients.

2.2.3 Discriminator Loss

The discriminator takes both the samples from the real dataset $\mathbf{x}$ and predicted novel 2D poses $\tilde{\mathbf{y}}$ and classifies the samples. For making such training data the values of 1s and 0s are given as the labels for the real and generated 2D poses respectively. As the labels are self-generated the discriminator training is self-supervised as well. In contrast to the reconstruction error, the error in discriminator's classification is not expected to be minimized even though the models are learning to minimize this metric.

Rather, the role it plays is to help the generator produce 3D poses that give more realistic 2D novel views. That is 2D projections from random views of the 3D pose that are indistinguishable from the 2D pose from the real dataset. Fig 2.2.1 illustrates the differences in the novel views of realistic and unrealistic 3D poses that this adversarial training aims to resolve. Hence this error is used to optimize both the discriminator and generator models that produce the 3D poses.

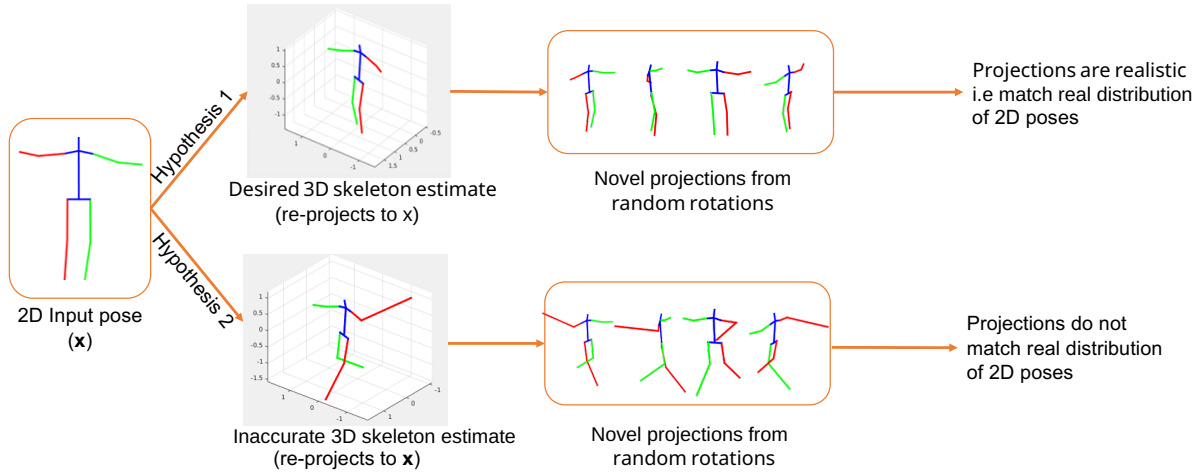

Figure 2.2.1: Illustration of the differences in accurate or realistic 3D pose predicitions from inaccurate predicitions. Modified image from [4]

The real and the fake 2D poses need not be related in any way. In practice, the loss terms of the real and fake data are computed and used to update the models separately, as suggested in [1, 7]. As the goal of the discriminator is to just predict the correct label of the 2D pose, the Binary Cross Entropy (BCE) loss function is used.

$$\begin{aligned} \mathcal{L}_{\text{Disc.}} &= BCE\_loss(y, \hat{y}) \\ &= -1 * (\hat{y} \cdot \log y + (1 - \hat{y}) \cdot \log(1 - y)) \\ &Where, \;\; y \in D_{\theta_d}(\tilde{\mathbf{y}} \cup \mathbf{x}) \;\; and \;\; \hat{y} \in (0s, 1s) \end{aligned} \tag{2.8}$$

The $\tilde{\mathbf{y}}$ here is the transformed output of the decoder. That means, to minimize this loss the decoder model is to be updated along with the discriminator. However, as discussed earlier we consider the encoder to be part of the generator. Hence all the models use this metric to update their weights. Again in contrast to the reconstruction loss, the discriminator loss does not give a direct signal to correct the 3D pose, and thus updating the models to minimize this loss is an unconstrained optimization and is not stable. Hence the weight coefficient $\lambda_{disc.}$ is not as high as that of the reconstruction

loss. As this is a min-max game where the generator tries to fool the discriminator while the discriminator tries to guess which same is from the generator, all the models cannot be updated at the same time. The training procedure to enable the models to learn and improve the generated 3D pose is discussed in the next section.

2.3 Training Procedure

In this min-max game, the generator produces samples and the discriminator verifies them, the generator uses this verification and in our method, it also uses self-supervision loss to improve itself in both aspects. The standard practice in training GANs is to start with generating the samples using the generator. As the discriminator's evaluation is required to teach the generator, the discriminator is first trained to distinguish reals from fakes given by the generator. The samples are again given to the trained discriminator to get the loss to train the generator. That is, the generator is kept constant while updating the discriminator and the discriminator is kept constant while updating the generator. This turn-wise play is important without which the models would be trying to hit a moving target as put by [6].

While the procedure seems straightforward, GAN training is finding a Nash equilibrium to a two-player non-cooperative game. There exists no feasible algorithm to find the equilibrium for a problem of such complexity[14]. As the training initiates, the VAE produces 3D poses close to the noise and thus is very easy for the discriminator to learn to classify. This overpowers the discriminator as it predicts fakes with very high confidence and the loss tends to 0. The VAE can not learn properly if the discriminator does not provide proper feedback. This happens when the discriminator does not perform well or if it performs too well where the loss is close to 0 and not helpful [19].

Other common challenges of GAN training and is explained in **??**. The widely practiced tricks to improve the training of GANs are discussed later in Bag of Tricks **??**. The most challenging part of training any GANs is the lack of a proper metric to evaluate the quality of generations. Since it is difficult to monitor intermediate states in 3D, and since 3D poses have fewer features compared to RGB images, it is even more challenging to evaluate the quality of the generations. Hence more importance is given to the quantitative aspect in this report.

2.4 Evaluation

2.4.1 Metric

3D Human Pose Estimation (HPE) and Human3.6M in particular is evaluated by Mean Per Joint Position Estimate (MPJPE) metric. MPJPE as it abbreviates is the mean of the position estimate for all the joints of a pose. Where per-joint position estimate is nothing but the euclidian distance (measured in mm) between the predicted joint to its ground truth. The reported results refer to the average of the MPJPE of all the samples in the evaluation set.

2.4.2 Protocols

Human3.6M has 11 subjects out of which 7 are publically released while the rest are kept private. All the works main use of these 7 subjects for training and test. There are 2 widely used evaluation protocols.

Protocol #1 All the 4 camera views of subjects $S1$, $S5$, $S6$, $S7$, and $S8$ are used for training. Similarly, all camera views of subjects $S9$ and $S11$ are used for validation/testing. But the test subject data is complied by sampling every 64^{th} frame of the raw data while the train subjects data is made by sampling every 5^{th} frame. The evaluation metric MPJPE calculation is done on the predicted 3D pose directly.

Protocol #2 The train/test split is the same as of Protocol #1. But MPJPE is calculated on the predicted 3D after rigid body/ Procrustes alignment.

The design choice of predicting the 3D pose in unit scale makes it impossible to evaluate using protocol #1. Hence, following [2], results are reported using only protocol #2 i.e after aligning the predicted 3D pose with the ground truth.

2.4.3 Best Hypothesis

During the evaluation phase, the discriminator network is discarded and the encoder and decoder pair alone are used. The usual practice during the evaluation phase of a VAE is to compute latent representation z using mean and zeroing out the predicted standard deviation. The 3D pose prediction using the mean only is referred to **ZV**, Zero Variance. As the trained decoder can predict realistic 3D poses from any point

in the latent space the variance can be scaled instead of zeroing to retrieve multiple 3D predictions for a 2D pose from which the mean is encoded. To evaluate the representation capability, 10 random hypotheses are produced following [11]. This is done by randomly scaling the predicted variance vector. The best hypothesis, **BH** i.e the pose with the least MPJPE for each 2D pose is selected to compute another variance of the mean MPJPE.

Chapter 3

Experiments

3.1 Supervised Baseline

To the best of the author's knowledge, the closest work using a Variational Auto-Encoder (VAE) is a cross-modal training approach in 3D hand pose estimation [16]. As there is no similar approach in Human Pose Estimation (HPE) using a VAE, a supervised model using just a VAE and 3D ground truth pose data is implemented. This is to verify the feasibility of a VAE and find a set of hyperparameters that worked well for the H3.6M dataset under the supervised setting. This served as a baseline over which an unsupervised Generative Adversarial Network (GAN) was built upon. Hyperparameters such as the number of neurons in the layers, the number of residual blocks n, dimensions of the latent space, etc are taking as reference.

3.2 Unsupervised GAN

Building upon the supervised VAE, the proposed architecture is built. The following choice are kept standard for most of the experiments and could be used to reproduce the results. All the layers are made of 1024 neurons. The encoder and the decoder is made of 1 residual block each ($n = 1$) while the discriminator is made of 2. This is to keep the capacity of the generator and the discriminator similar. Various forms of scaling the 2D poses are experimented with to make the cycle of lifting and projection smooth. The scaling mentioned in the preprocessing section where the distance between the root joint and head worked the best.

3.3 Bag of Tricks

The following techniques to improve the GAN training and the reasoning behind their importance are elaborated in papers such as [1, 7, 14, 20]. The important hacks or tricks from these papers and other sources are compiled in [15] and are wildly adopted in the field. Since the majority of the work in the field is on images, some of these tricks do not help and sometimes even hurt the training. These ticks along with the pre-processing steps mentioned in **??** are critical for proper training of the network. The tricks are as follows.

Normalizing Inputs It is common practice to normalize the inputs before training a neural network. When it comes to GANs, the output of the generator is the input of the discriminator. Hence it is suggested to normalize the images between [-1,1] by using a Tanh activation function at the output layer of the generator. This is another motivation behind predicting the poses in the range [-1,1] and then scaling the make the upper half of the pose to unit length.

Modified Loss function The theory the generator component of the GAN loss **??**, $log(1 - D(G(z)))$ is minimized. As explained in **??**, it is replaced with maximizing $log(D(G(z)))$ as the former leads to vanishing gradients early on during the training. This is referred to as the $-logD$ trick. In practice, the labels for fakes are flipped as reals while training the generator, since the goal is to make the generator's output real according to the discriminator. And as the Binary Cross Entropy (BCE) loss 2.8 which has a negative magnitude in its formulation is used, maximization is achieved while minimizing the loss during training. Note that the generator variable $G(z)$ in our training procedure is actually $P(Q(\mathbf{x}))$.

Spherical Latent Space Sample z from a spherical distribution instead of uniform distribution. Doing interpolation along the great circle 3.3.1 rather than a straight line from sample A to sample B. This spherical linear interpolation prevents the divergence of samples from the model's prior distribution and produces output with better features.

BatchNorm Training the GAN using separate batches of real and fake data without mixing them. And perform instance normalization when batch normalization is not

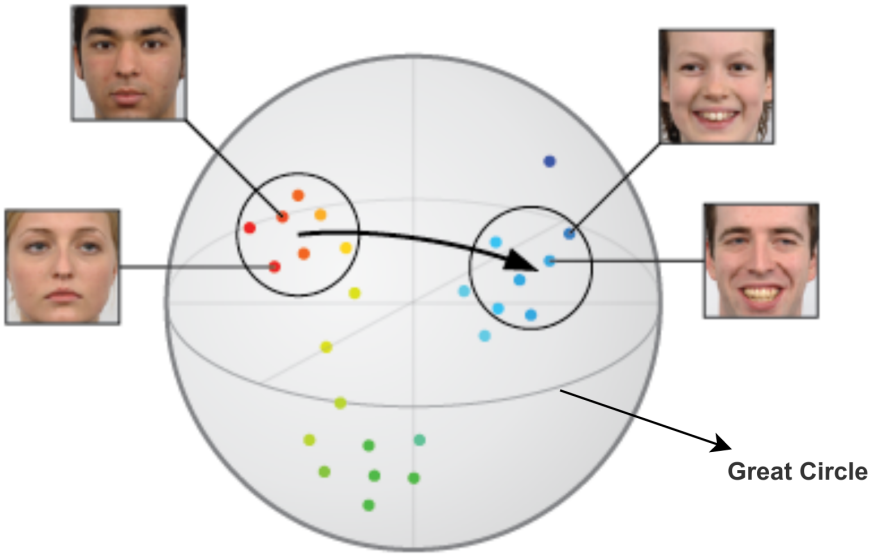

Figure 3.3.1: Illustration of the sampling from the great circle. Image source [21]

possible.

Avoid Sparse Gradients Sparse gradients affect the training stability of the GAN. Hard functions like hard- Rectified Linear Unit (ReLU), Max Pooling that have sparse gradients should be avoided. Instead, Leaky-ReLU work well for both the generator and the discriminator. And average pooling or convolution layers with stride are better for downsampling or upsampling.

As mentioned previously Tanh and Sigmoid activations are used for the output layers of the generator and the discriminator respectively. For all the remaining layers Leaky-ReLU was tried and good results were observed. Experimenting with other combinations of activation functions, using Mish activation function [13] for the VAE and Leaky-ReLU for the discriminator is found to give small but noticeable improvement to the performance.

Based on the choice of activation, Kaiming He initialization with Leaky-ReLU non-linearity is used for all the layers.

Label Smooting The label smoothing technique mentioned in [15] is by replacing the binary labels for real and fake i.e 1, 0 with random numbers, between 0.7

and 1.2 and 0.0 to 0.3 respectively. However, there are different ways of doing smoothing. These include random/non-random, one-sided/two-sided, only for discriminator/both networks. Only the real labels are smoothed for both the discriminator and the generator in the experiments.

Label Flipping The other way of making the labels noisy for the discriminator so as to prevent it from being too strong is by flipping labels during the discriminator training. Real labels are changed to fake and vice-versa occasionally to confuse the discriminator.

This had some effect during the initial phase of the training but is not observed to directly affect the quantitative results and hence and kept inactive for simplicity.

GAN Model Choice As training GANs is notoriously difficult, the most explored and studied variant DCGAN (Deep Convolutional GAN) is suggested to be used. The next best approach is to use hybrid models that use Kullback–Leibler Divergence (KLD) or a combination of VAE and GAN. Since it is not desired to use convolutional layers in lifting networks, the proposed method is already following the best alternative.

Optimizer choice The best configuration of optimizers is Adam for the generator and SGD for the discriminator. However, this combination made the models diverge and has severely affected the training of the proposed method. Hence Adam is chosen for both the networks.

Noisy Inputs Adding noise to inputs and decaying over time helps the training. The 2D poses were added random noise in different scales but improvements were not noticeable in the basic experiments.

Training Discriminator More To consistently get good feedback from the discriminator, it is ideal to keep always keep the discriminator at good performance. To achieve this, the discriminator is iterated n times before every iteration of the VAE. However, n is set to 1 as the benefit could not be seen in the initial observations.

Dropout Adding additional noise to the generator using dropout with $p = 0.5$ during training and test time [8]. The discriminator uses dropout with $p = 0.5$. But it is

observed that adding dropout layers have a negative effect on VAEs and $p = 0.2$ was found to have a good trade-off.

Chapter 4

Results

The results presented here are after training the networks for ∼400 epochs (5.5 hours) on approximately 300,000 2D poses with a batch size of 2560 on an Nvidia Titan X. The input poses are flipped with a probability of 0.5. The model takes 16 joints as the output where the root is added at the origin for validation. The proposed architecture consists 1024 hidden units per linear layer and 51 latent dimensions. Both the Variational Auto-Encoder (VAE) and the discriminator are trained using Adam optimizer with default hyperparameters and with a learning rate of 2e-4. The gradient norms of the discriminator is clipped to 1 when training the discriminator. While training the generator the gradient norms are clipped to 2 for all the models while the gradient values are clipped to 1000.

One of the challenging parts is finding the optimal weights for each of the terms in the triplet loss. The loss coefficients λ_{recon}, λ_{KLD}, $\lambda_{disc.}$ are set to 1, 0.001, 0.001 respectively. The higher weight is motivated by 2 reasons. λ_{recon} refers to the constrained optimization and irrespective of how realistic it is, projection loss is desired to be consistently low to get better Mean Per Joint Position Estimate (MPJPE). That leads to the other reason that the quantitative results are given higher importance.

The values of λKLD and $\lambda disc.$ can be tuned according to the task at hand based on how well the poses are to be clustered or how important it is to reject poses that are not realistic. The β value for the VAE is cycled from 0 to λ_{KLD} every 40 epochs. While keeping it constant at λ_{KLD} for 10 epochs with a 10 epoch warmup at the beginning of the training.

4.1 Quantitative Results

The results obtained by the networks with the above configuration in addition to the choices mentioned in 3 are summarized in Table 4.1.1. The summaries of the models are provided in appendix A to help reproduction.

Supervision	Algorithm	Error (mm)
Full	Martinez *et al.* [12]	37.1
	Chen *et al.* [10] (SH, MH)	42.6
Weak	3D Interpreter *et al.* [22]	88.6
	AIGN *et al.* [17]	79.0
	Wandt *et al.* [18]	38.2
	Drover *et al.* [4]	38.2
	Chen *et al.* [11] (SH)	48.7
	Chen *et al.* [11] (BH)	31.6
Unsupervised	Ching *et al.* [2]	58
	Ching *et al.* [2] (DA)	55
	Ching *et al.* [2] (DA) (TD)	51
	Ours (ZV)	52.74
	Ours (BH)	50.37

Table 4.1.1: Results on Human3.6M in MPJPE under Protocol #2 using ground truth 2D pose as input.

SH refers to the results using 2D Stacked Hourglass detections as input. These detections are noisy and directly affects the predictions of the model. And **DA** refers to using Domain Adaptation network to include additional datasets and **TD** denotes the use of temporal data. It is important to note that the proposed unsupervised method and the one presented in [2] are the only methods that do not predict the scale of the 3D pose due to the processing technique.

R_Hip	R_Knee	R_Ankle	L_Hip	L_Knee	L_Ankle	Torso	Neck
58.01	61.42	81.97	52.45	60.13	92.82	44.78	25.43

Nose	Head	L_Shoulder	L_Elbow	L_Wrist	R_Shoulder	R_Elbow	R_Wrist
33.39	46.29	30.48	55.72	80.84	33.53	59.23	80.05

Table 4.1.2: Average per joint position error (in mm) for each joint under Protocol #2 using 2D ground truth.

The best hypothesis, **BH** has improved the results by 2.7 mm which is considerable in comparison to the equivalent gain in [2] using domain adaptation network with more data and temporal information. Since there is no technique to pick the best hypothesis

without having access to the ground truth, the results referred to, are the ones obtained using **ZV** unless specified otherwise. The average MPJPE using ZV, 52.74mm, is equivalent to the settings of [2] without additional data or temporal information, 58 mm. This is a significant improvement considering that the network from [2] only predicts the depth, cannot handle missing points, gives a single hypothesis, and uses a self-symmetry technique that takes twice as many training iterations for equivalent network complexity. Fig 4.1.1 illustrates the distribution of the MPJPE errors for each action.

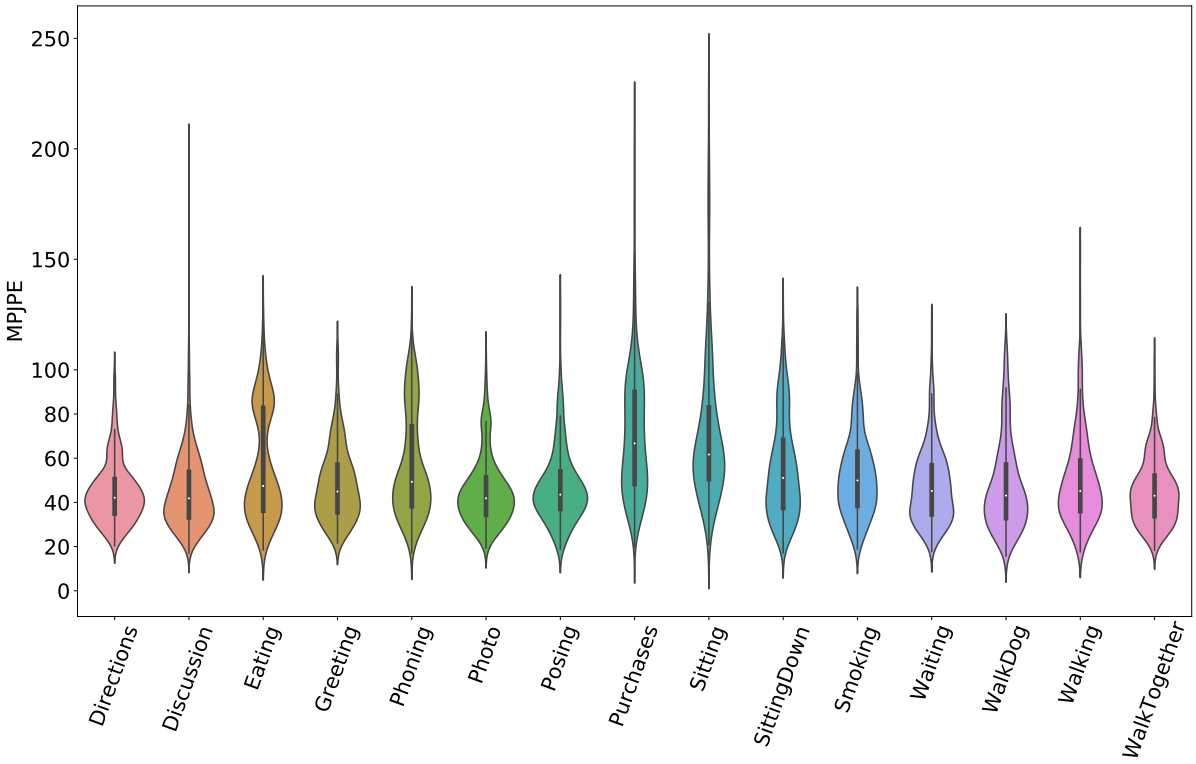

Figure 4.1.1: Visualization of the MPJPE error distribution under Protocol #2 for each action.

The MPJPE error per joint increases as we move from inner to outer joints. This can be observed in Table 4.1.2 which shows the average per joint position error. The error in the limb joints is due to the larger variation in their location through out the data compared to joints like the neck or the nose. Moreover, these joints are often occluded by the person's body especially when doing actions such as sitting. The 2D projections a pose from sitting posture is clustered to a smaller region from certain Point of Views (POVs). This makes it challenging to estimate the scale and depth of each joint and leads to outliers as visualized in the violin plots.

Since the data consists an equal mix of data from different POVs, the predictions from

the POV, where the limbs are in the Field of View (FOV) are more informative than the other POVs. When the limbs are not in the FOV, the model learns to guess their location based on other features such as articulation or posture of the body. The effect of POV is the possible cause of the bimodal error distribution in actions such as purchases, eating, phoning. Since the actions are hands specific, they are restricted to a particular region making the pose more predictable from one POV and less from another. These challenges are inherent to the task of pose lifting and in many cases is challenging even for a human eye.

4.2 Qualitative Results

Some of such outliers are presented in fig 4.2.2, the predictions in (a) are the ones the model is unable to learn. While (b) is the evidence of the shortcoming of the current processing technique. Rectifying that would improve the evaluation metric of the model quite significantly.

4.3 Latent Space

The visualization of 2D pose embedding in latent space after dimensionality reduction using Uniform Manifold Approximation and Projection (UMAP) is shown in fig. **??**. Each action is given a unique color. Though we small clusters of blues, browns, pinks the overall space looks very mixed up. This is expected as many of the instances in different actions overlap. For example, the action standing up and sitting down have instances while both or standing or sitting, etc.

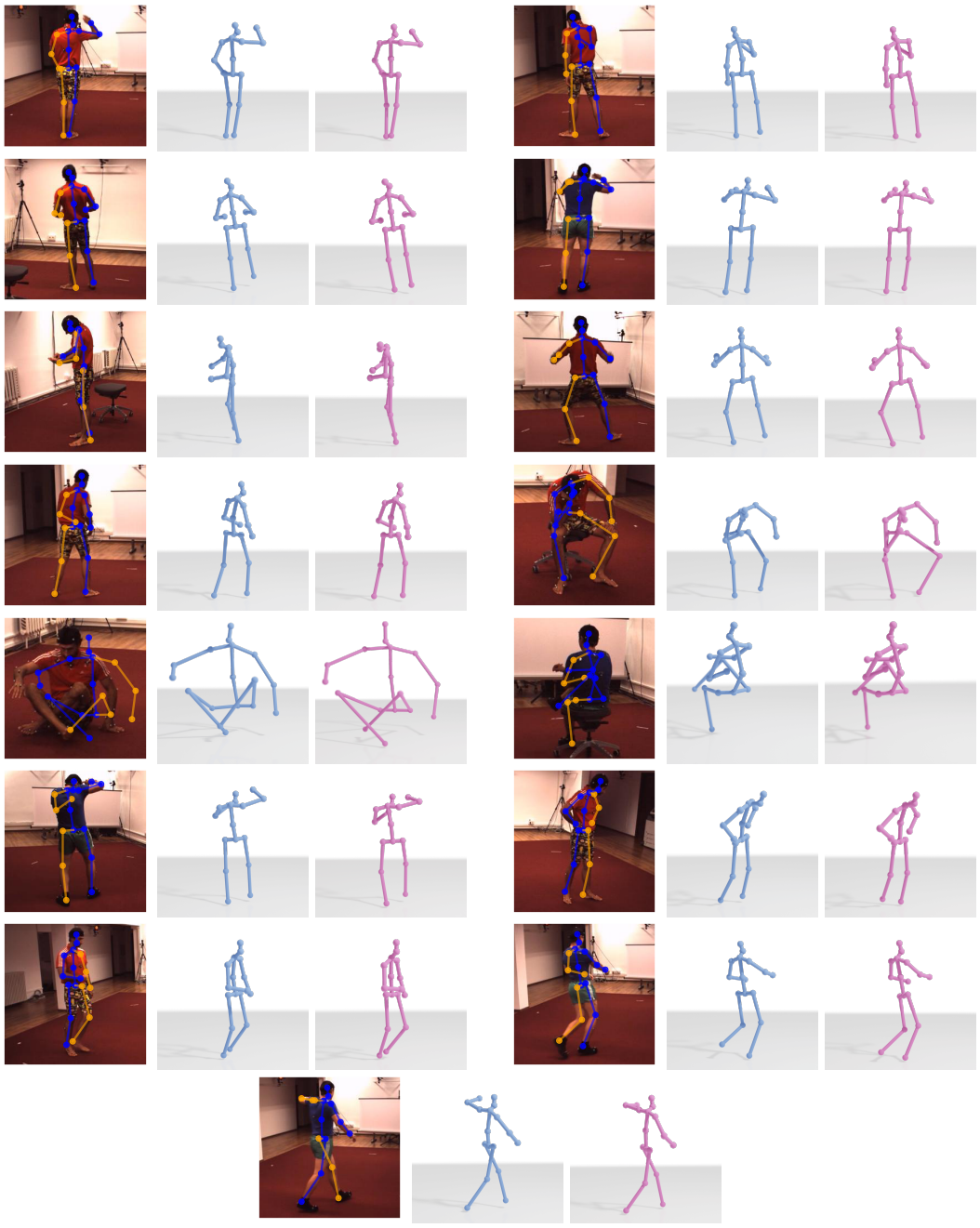

Figure 4.2.1: Visualization of (left to right) the input 2D poses with RGB background for reference, the predicted 3D pose after Procrustes alignment (in blue) and the ground truth 3D pose (in pink). The aligned predictions are the ones closest to the mean MPJPE error of each action.

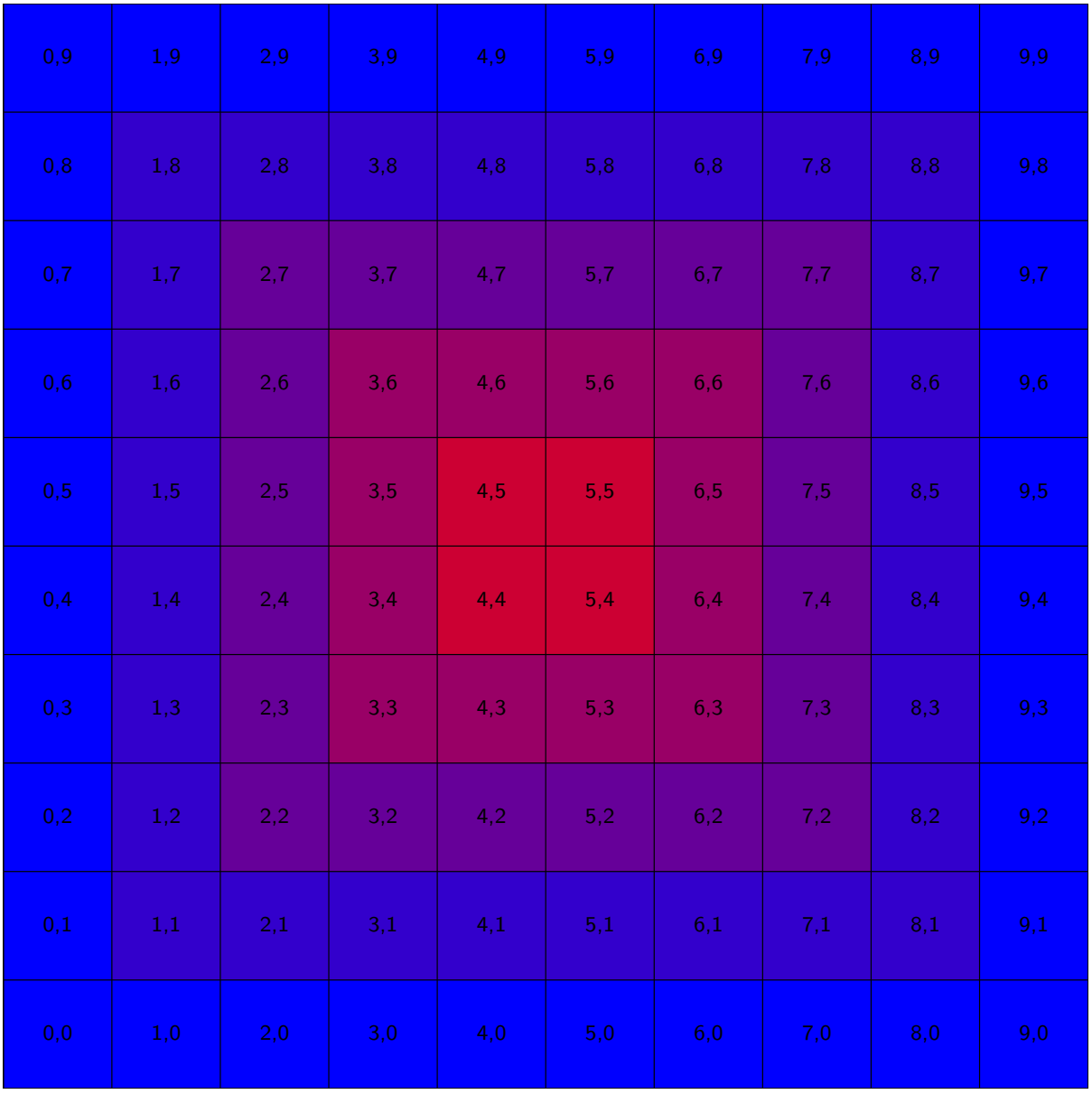

Figure 4.2.2: (a) Prediction on hard poses with hih ambiguity. (b) Poses that can be improved with changes to data processing.

Chapter 5

Conclusions

YET TO BE WRITTEN

Describe the conclusions (reflect on the whole introduction given in Chapter 1).

Discuss the positive effects and the drawbacks.

Describe the evaluation of the results of the degree project.

Describe valid future work.

The sections below are optional but could be added here.

5.1 Discussion

5.1.1 Future Work

5.1.2 Final Words

Bibliography

[1] Arjovsky, Martin, Chintala, Soumith, and Bottou, Léon. *Wasserstein GAN*. 2017. arXiv: 1701.07875 [stat.ML].

[2] Chen, Ching-Hang, Tyagi, Ambrish, Agrawal, Amit, Drover, Dylan, Rohith, M. V., Stojanov, Stefan, and Rehg, James M. "Unsupervised 3D Pose Estimation with Geometric Self-Supervision". In: *CoRR* abs/1904.04812 (2019). arXiv: 1904.04812. URL: http://arxiv.org/abs/1904.04812.

[3] Dario Amodei, Danny Hernandez. *AI and Compute*. https://openai.com/blog/ai-and-compute/. (Accessed on 05/17/2020).

[4] Drover, Dylan, MV, Rohith, Chen, Ching-Hang, Agrawal, Amit, Tyagi, Ambrish, and Huynh, Cong Phuoc. *Can 3D Pose be Learned from 2D Projections Alone?* 2018. arXiv: 1808.07182 [cs.CV].

[5] Fu, Hao. "Cyclical Annealing Schedule: A Simple Approach to Mitigating KL Vanishing". In: *NAACL*. 2019.

[6] *GAN Training Generative Adversarial Networks Google Developers*. URL: https://developers.google.com/machine-learning/gan/training.

[7] Goodfellow, Ian J., Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron, and Bengio, Yoshua. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML].

[8] Isola, Phillip, Zhu, Jun-Yan, Zhou, Tinghui, and Efros, Alexei A. *Image-to-Image Translation with Conditional Adversarial Networks*. 2018. arXiv: 1611.07004 [cs.CV].

[9] Larsen, Anders Boesen Lindbo, Sønderby, Søren Kaae, Larochelle, Hugo, and Winther, Ole. *Autoencoding beyond pixels using a learned similarity metric*. 2015. arXiv: 1512.09300 [cs.LG].

[10] Li, Chen and Lee, Gim Hee. *Generating Multiple Hypotheses for 3D Human Pose Estimation with Mixture Density Network*. 2019. arXiv: 1904.05547 [cs.CV].

[11] Li, Chen and Lee, Gim Hee. *Weakly Supervised Generative Network for Multiple 3D Human Pose Hypotheses*. 2020. arXiv: 2008.05770 [cs.CV].

[12] Martinez, Julieta, Hossain, Rayat, Romero, Javier, and Little, James J. "A Simple Yet Effective Baseline for 3d Human Pose Estimation". In: *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*. IEEE Computer Society, 2017, pp. 2659–2668. DOI: 10.1109/ICCV.2017.288. URL: https://doi.org/10.1109/ICCV.2017.288.

[13] Misra, Diganta. *Mish: A Self Regularized Non-Monotonic Activation Function*. 2020. arXiv: 1908.08681 [cs.LG].

[14] Salimans, Tim, Goodfellow, Ian, Zaremba, Wojciech, Cheung, Vicki, Radford, Alec, and Chen, Xi. *Improved Techniques for Training GANs*. 2016. arXiv: 1606.03498 [cs.LG].

[15] Soumith. *How to Train a GAN? Tips and tricks to make GANs work*. URL: https://github.com/soumith/ganhacks.

[16] Spurr, Adrian, Song, Jie, Park, Seonwook, and Hilliges, Otmar. "Cross-Modal Deep Variational Hand Pose Estimation". In: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. IEEE Computer Society, 2018, pp. 89–98. DOI: 10.1109/CVPR.2018.00017. URL: http://openaccess.thecvf.com/content%5C_cvpr%5C_2018/html/Spurr%5C_Cross-Modal%5C_Deep%5C_Variational%5C_CVPR%5C_2018%5C_paper.html.

[17] Tung, Hsiao-Yu Fish, Harley, Adam W., Seto, William, and Fragkiadaki, Katerina. *Adversarial Inverse Graphics Networks: Learning 2D-to-3D Lifting and Image-to-Image Translation from Unpaired Supervision*. 2017. arXiv: 1705.11166 [cs.CV].

[18] Wandt, Bastian and Rosenhahn, Bodo. "RepNet: Weakly Supervised Training of an Adversarial Reprojection Network for 3D Human Pose Estimation". In: *CoRR* abs/1902.09868 (2019). arXiv: 1902.09868. URL: http://arxiv.org/abs/1902.09868.

[19] Weng, Lilian. *From GAN to WGAN*. Aug. 2017. URL: https://lilianweng.github.io/lil-log/2017/08/20/from-GAN-to-WGAN.html.

[20] Weng, Lilian. *From GAN to WGAN*. 2019. arXiv: 1904.08994 [cs.LG].

[21] White, Tom. *Sampling Generative Networks*. 2016. arXiv: 1609.04468 [cs.NE].

[22] Wu, Jiajun, Xue, Tianfan, Lim, Joseph J., Tian, Yuandong, Tenenbaum, Joshua B., Torralba, Antonio, and Freeman, William T. "Single Image 3D Interpreter Network". In: *Lecture Notes in Computer Science* (2016), pp. 365–382. ISSN: 1611-3349. DOI: 10.1007/978-3-319-46466-4_22. URL: http://dx.doi.org/10.1007/978-3-319-46466-4_22.

Appendix - Contents

Appendix A

Model Summaries

A.1 Encoder

	Layers	Parameters
Upsampling Block		
	Linear(in features=32, out features=1024, bias=True)	33792
	BatchNorm1d(1024, eps=1e-05, momentum=0.1)	2048
	Mish()	0
	Dropout(p=0.2, inplace=False)	0
Residual Block		
	Linear(in features=1024, out features=1024, bias=False)	1048576
	BatchNorm1d(1024, eps=1e-05, momentum=0.1)	2048
	Mish()	0
	Dropout(p=0.2, inplace=False)	0
	Linear(in features=1024, out features=1024, bias=False)	1048576
	BatchNorm1d(1024, eps=1e-05, momentum=0.1)	2048
	Mish()	0
	Dropout(p=0.2, inplace=False)	0
Downsampling Block		
	Linear(in features=1024, out features=51, bias=True)	52275
	Linear(in features=1024, out features=51, bias=True)	52275

A.2 Decoder

	Layers	Parameters
Upsampling Block		
	Linear(in features=51, out features=1024, bias=True)	53248
	BatchNorm1d(1024, eps=1e-05, momentum=0.1, affine=True, track running stats=True)	2048
	Mish()	0
	Dropout(p=0.2, inplace=False)	0
Residual Block		
	Linear(in features=1024, out features=1024, bias=False)	1048576
	BatchNorm1d(1024, eps=1e-05, momentum=0.1, affine=True, track running stats=True)	2048
	Mish()	0
	Dropout(p=0.2, inplace=False)	0
	Linear(in features=1024, out features=1024, bias=False)	1048576
	BatchNorm1d(1024, eps=1e-05, momentum=0.1, affine=True, track running stats=True)	2048
	Mish()	0
	Dropout(p=0.2, inplace=False)	0
	Linear(in features=1024, out features=48, bias=True)	49200
Downsampling Block		
	Linear(in features=1024, out features=48, bias=True)	49200
	Tanh()	0

A.3 Discriminator

	Layers	Parameters
Upsampling Block		
	Linear(in features=32, out features=1024, bias=True)	33792
	LeakyReLU(negative slope=0.01)	0
Residual Block		
	Linear(in features=1024, out features=1024, bias=True)	1049600
	LeakyReLU(negative slope=0.01)	0
	Dropout(p=0.5, inplace=False)	0
	Linear(in features=1024, out features=1024, bias=True)	1049600
	LeakyReLU(negative slope=0.01)	0
	Dropout(p=0.5, inplace=False)	0
Residual Block		
	Linear(in features=1024, out features=1024, bias=True)	1049600
	LeakyReLU(negative slope=0.01)	0
	Dropout(p=0.5, inplace=False)	0
	Linear(in features=1024, out features=1024, bias=True)	1049600
	LeakyReLU(negative slope=0.01)	0
	Dropout(p=0.5, inplace=False)	0
Upsampling Block		
	Linear(in features=1024, out features=1, bias=True)	1025
	Sigmoid()	0