Yoga Pose Classification and Fake Pose Generation

Tejas Prashanth
tejuprash@gmail.com
01FB16ECS419

Sumanth V Rao
sumanthvrao@gmail.com
01FB16ECS402

Abstract

The goal of the project is to build a yoga pose classification system and a fake pose generation system. The long term goal of the project is to build a self-assisted yoga pose correction and training system using deep learning. Our primary motivation behind this project is the lack of availability of a good quality dataset for the development of such a system. Consequently, a yoga pose generation system can be used to build a large, good quality dataset of fake as well as real poses. Hence, the report describes approaches of building a yoga pose classification system and a fake pose generation system.

1 Introduction

The goal of this project is to build a deep learning model for fake pose generation of yoga poses. We explore this task by first building a model for yoga pose classification, which is followed by a model for fake pose generation. Firstly, a dataset consisting of images of four different yoga poses is built and various body key points are extracted. Thereafter, key points detected are used as features for the models. The yoga pose classification system consists of Long Short Term Memory(LSTM) cells. The model is capable of classifying four different yoga poses. In addition, the fake pose generation system consists of a Variational AutoEncoder. By training a variational autoencoder and interpolating the latent space, fake poses are obtained.

2 Dataset

One of the core challenges that was encountered during the course of the project is the lack of availability of a pre-existing dataset. Consequently, a dataset of various yoga poses is built. To elaborate, images corresponding to four yoga postures, specifically, tree pose(Vrikshasana), mountain pose(Tadasana), warrior pose(Virabhadrasana) and plank pose(Kumbhakasana), were scraped using publicly available search APIs of Google and Bing Search. A dataset of 2964 images of differing sizes was collected. The dataset includes subcategories of each pose and poses captured from a variety of angles. The dataset can be found using this link: https://drive.google.com/open?id=1JpVGyQMtgsPbaFv7370jxSyF6jVPD7KC

3 Feature extraction

A generic pose detection model is used in order to extract key points corresponding to the human body. OpenPose[1] is a state-of-the-art multi-person real time pose detection tool. OpenPose is trained on the COCO[2] and a foot key-point dataset. The trained model detects 25 body and foot key points, as shown in the figure1. The key points correspond to 19 core body key points and 8 foot keypoints.

The output produced by OpenPose represents a set of 25 X and Y normalized co-ordinates that lie between (0,0) and (1,1). The top left corner of the image represents (0,0) and the bottom right corner represents (1,1)

,

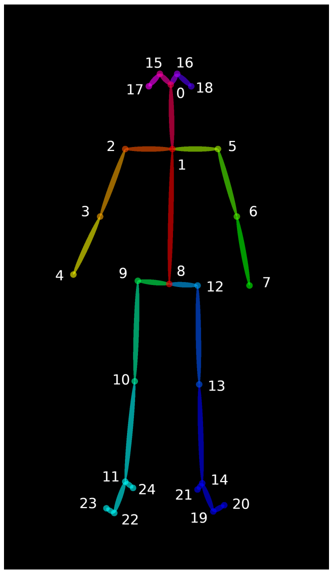


Figure 1: Body and foot keypoints

4 Proposed Model

The output produced by OpenPose is fed as input features to the pose classification and fake pose generation models

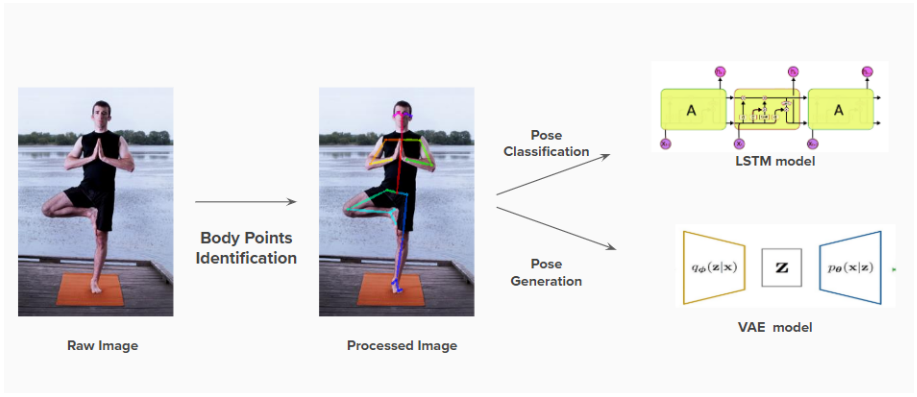


Figure 2: High level block diagram

4.1 Yoga Pose classification

This subsection describes the model used for yoga pose classification. Recurrent models have proven to work for applications such as sentiment analysis, image captioning due to the presence of temporal data. Due to the inherent ordered nature of the input features, a recurrent model is used. The features fed as input represent an ordered set of key points, with the order described in Figure 1. A single 10 neuron LSTM layer is used with the activation function between time steps being sigmoid and tanh as the activation function between the LSTM layer and the output layer. The output layer consists of 4 neurons with softmax as the activation function. Moreover, Adam is used as the optimization algorithm and cross entropy as the loss function.The architecture for the model is described in Figure 3

The model is capable of classifying 4 yoga poses-mountain pose, warrior pose, tree pose and plank pose

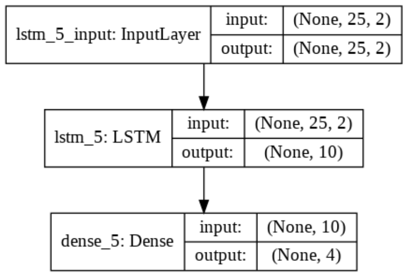


Figure 3: Pose Classification model

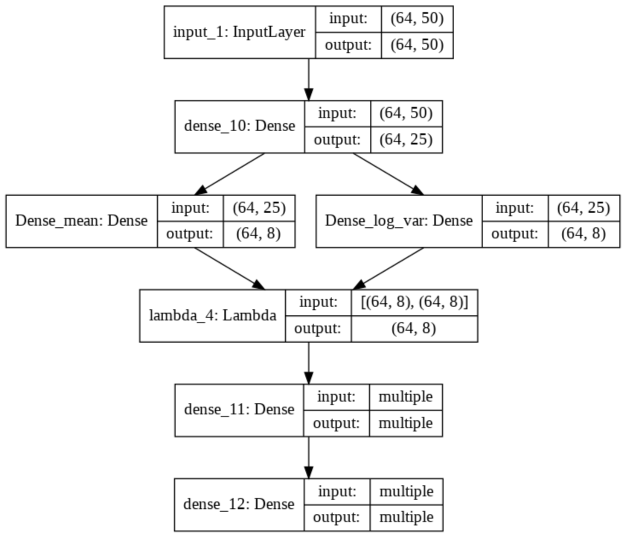
4.2 Fake Pose Generation

The task of fake pose generation is divided into two sub tasks. Firstly, a Variational Autoencoder is trained in order to regenerate correct poses that are fed as input to the model. The aim of this sub-task is for the model to learn an appropriate latent space. Latent space is a compressed form of the input, mathematically expressed in lower dimensions compared to the input. Secondly, latent space interpolation is used in order to sample new points from the latent space and manipulate them so that they can be fed into a trained decoder in order to produce fake poses.

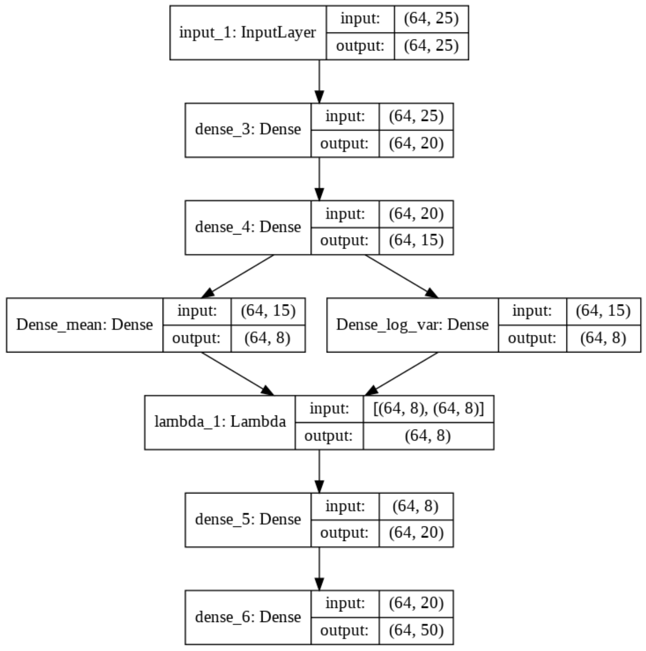
4.2.1 Model architecture

The following sub section describes various models employed.

Firstly, a Variational AutoEncoder with latent space being 8 dimensions in length is trained. Figure 4a describes the model architecture(abbreviated as **2 class Flattened model**). The input features and output are 50 dimensions in length, which represent flattened X and Y co-ordinates(which is in the form of X1,Y1,X2,Y2 and so on).



(a) 2 class Flattened model



(b) 4 class Norm Model

Figure 4: Model architectures

The above mentioned model is trained for only 2 classes of input-Mountain + Plank poses and tree + warrior poses. In this way, 2 separate autoencoder models are obtained for the 2 pairs of classes.

Moreover, the model is also trained with all four classes of input(four different yoga poses). The latent space consists of 8 dimensions and the L2-norm of X,Y co-ordinates of every keypoint fed as input. This model is abbreviated as **4 class Norm Model**. The model architecture as described in Figure 4b

The loss function used in the model is a combination of reconstruction loss and KL divergence. Binary crossentropy is used as the reconstruction loss.

$$Loss = -c * \sum_{i}^{m} (y_i \ln a_i + (1 - y_i) \ln(1 - a_i)) - 0.5 * ((\ln \sigma^2) + 1 - \nu^2 - \sigma^2) \tag{1}$$

where c represents a hyper-parameter tuned during training

y_i represents the corresponding output values in the training set

a_i represents the corresponding predicted values in the training set

σ represents the standard deviation of the approximated distribution predicted by the encoder component

ν represents the mean of the approximated distribution predicted by the encoder component

m is the batch size

The re-paramerization trick is used in the model, using which values are sampled from the Standard Normal Distribution and converted to the appropriate distribution, which is produced by the encoder, by multiplying by the standard deviation σ and adding it to ν. In addition, the activation function used in the last layer is the sigmoid activation function since the output of the network should be between 0 and 1 and Adam is used as the optimization algorithm with the model trained using a batch size of 64.

4.2.2 Latent space interpolation

After training the variational autoencoder, latent space interpolation is performed in order to generate new fake poses. The following algorithm is used for latent space interpolation[3]:

Algorithm 1: Latent space Interpolation

```
Encoder(Input features);
Obtain centroid for 2 different classes of input,say $Z_a$ and $Z_b$. This will correspond to two
  different yoga poses;
Find the difference vector between the centroids,say $Z_{diff}$;
Generate new samples in latent space using $Z_{new}$=$Z_b$+ $\alpha$*$Z_{diff}$ ($0 < \alpha < 1$);
Decoder($Z_{new}$);
```

5 Results and Discussion

5.1 Yoga Pose classification

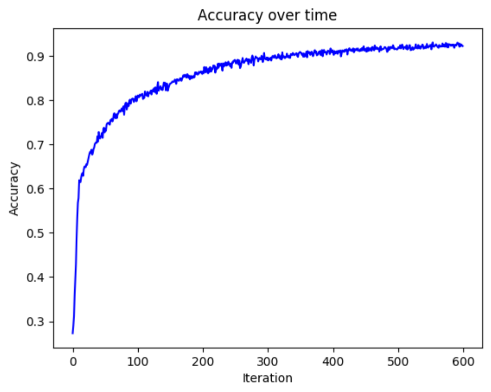
The Pose Classification model is capable of classifying four different yoga poses, specifically Mountain, Plank, Warrior and Tree poses.

Model	Accuracy	Validation Accuracy
LSTM model	92.24%	92.90%

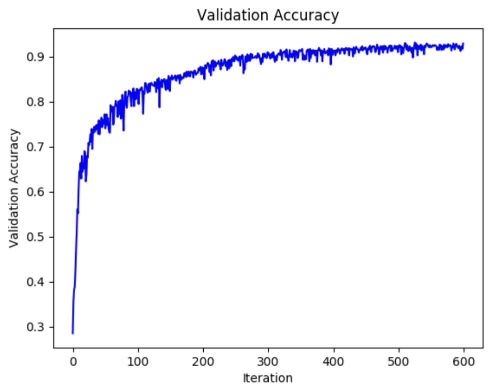
Figure 5a and 5b depict the accuracy and validation accuracy of the model. The classifier is able to achieve 92.90% validation accuracy.

5.2 Fake pose generation

For the variational autoencoder,c represents an important hyper-parameter that is tuned during training.



(a) Accuracy



(b) Validation accuracy

Figure 5: Model performance for pose classification

5.2.1 2 class Flattened Model

The hyperparameter c is obtained to be 30. So, the loss function becomes:

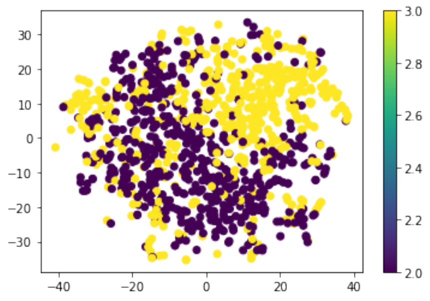
$$Loss = -30 * \sum_{i}^{m} (y_i \ln a_i + (1 - y_i) \ln(1 - a_i)) - 0.5 * ((\ln \sigma^2) + 1 - \nu^2 - \sigma^2) \qquad (2)$$

Two separate variational autoencoders are trained-one for generation of mountain and plank poses and the other for warrior and tree poses. Each model is trained for 2000 epochs and table 1 summarizes their performance

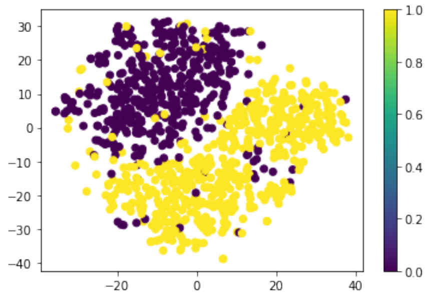
Data	Loss	Validation Loss	KL Loss
Mountain + Plank poses	18.50	18.58	0.395
Warrior + Tree poses	18.81	18.79	0.33

Table 1: Performance of 2 class flattened model

Since the quality of the latent space is an important quality that needs to be assessed, a TSNE plot of the latent space is performed and used as a metric. Figures 6a and 6b depict the TSNE plot of the latent space



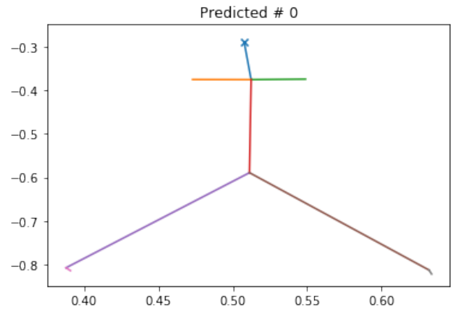
(a) TSNE plot for latent space of Mountain + plank pose



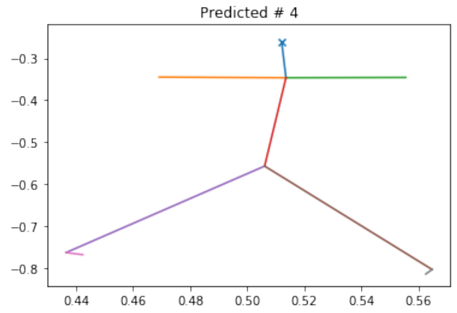
(b) TSNE plot for latent space of warrior + tree pose

Figure 6: TSNE plot for 2 class Flattened model

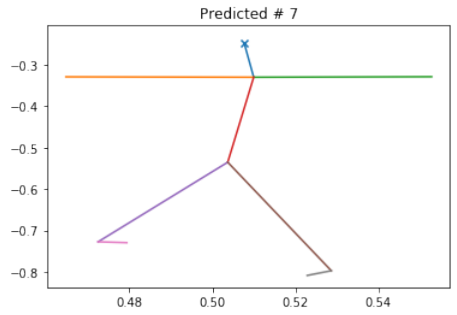
Figure 7 depicts transitions from the Warrior to the Tree pose. The figure shows that the model's latent space has learnt distinguishing features between warrior and tree poses such as the width



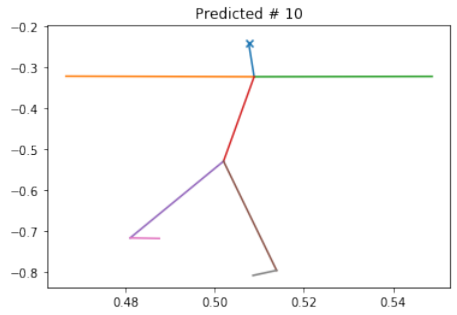
(a) Step 1



(b) Step 2



(c) Step 3



(d) Step 4

Figure 7: Transition from Warrior to Tree Pose

between legs, direction of feet and so on. Intermediate steps 2 and 3, although not of great quality, depict fake poses generated from the learnt latent space

5.2.2 4 class Norm Model

The hyperparameter c is obtained to be 30 and the L2-norm of X,Y co-ordinate of each key point is provided as input. So, the loss function becomes:

$$Loss = -25 * \sum_{i}^{m} (y_i \ln a_i + (1 - y_i) \ln(1 - a_i)) - 0.5 * ((\ln \sigma^2) + 1 - \nu^2 - \sigma^2)$$

The performance of the 4 class norm model is summarized in Table 2

Data	Loss	Validation Loss	KL Loss
Mountain + Plank + Warrior + Tree poses	14.98	15.05	0.605

Table 2: Performance of 4 class model

Although the model's loss is comparable with the performance of the 2-class flattened model, the quality of the latent space is poor, as depicted if Figure 8

5.2.3 Importance of KL Loss

The KL divergence loss is an important component whose value was closely observed during training. KL Loss is used as a regularization term. The variational autoencoder loss function is generally written as

$$Loss = \sum_{i}^{m} (y_i \ln a_i + (1 - y_i) \ln(1 - a_i)) - 0.5 * ((\ln \sigma^2) + 1 - \nu^2 - \sigma^2)$$

The above loss function gives equal weightage to the KL loss term as well as the reconstruction loss term. When attempted with the loss function, the 2 class flattened model caused the KL loss

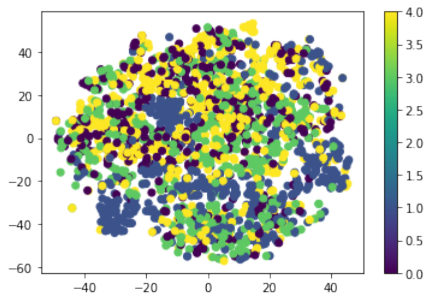
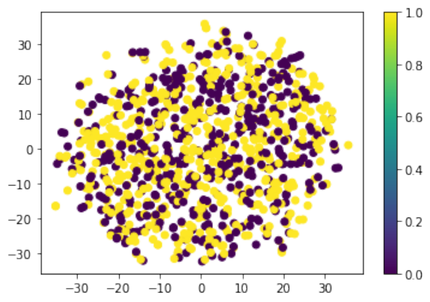
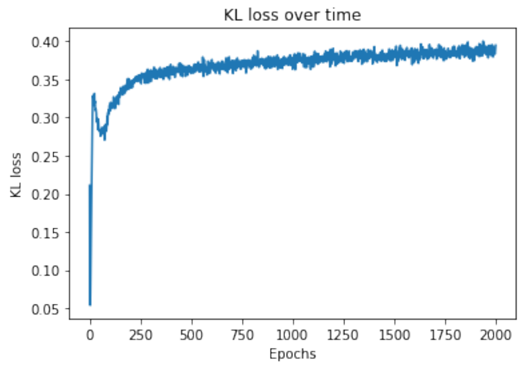


Figure 8: TSNE plot of 4 class norm model

to become zero after a few epochs. This resulted in the *KL vanishing problem*, a common problem encountered when training variational auto encoders.Consequently, the quality of the latent space learnt was poor, as shown in Figure 9a


(a) TSNE plot when KL vanishing problem is encountered



(b) KL Loss that produces a good quality latent space

Figure 9: Analysing KL Loss

On the the hand, when equation (2) was used as the loss function, the quality of the latent space is higher as shown in Figure 6b. The KL loss to produce such a latent space is described by Figure 9b.

6 Conclusion and Future Work

In this project, we explored approaches to build a fake yoga pose generation system using Variational AutoEncoders. The LSTM model is able to classify four different yoga poses with an accuracy of 92.90%. In addition, the Variational AutoEncoder is capable of generating yoga poses and latent space interpolation enables the model to generate fake poses. However, the quality of the poses can be improved using a larger training dataset. Due to the sequential nature of the input features for the Variational AutoEncoder model(ordered set of key points corresponding to each pose), a sequence-to-sequence model can be used as the encoder and decoder components with the output of the encoder representing a multivariate Gaussian distribution, as conventionally used in a variational autoencoder.

References

[1] Cao, Zhe, et al. "OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields." arXiv preprint arXiv:1812.08008 (2018).

[2] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft COCO: common objects in context,” in ECCV, 2014

[3] Krasser, Martin. “Deep Feature Consistent Variational Auto-Encoder.” Deep Feature Consistent Variational Auto-Encoder , 27 July 2018, krasserm.github.io/2018/07/27/dfc-vae/.