Deep Vision
Yoga-82

Robin Ruland (R.Ruland@stud.uni-heidelberg.de); Zhao Sun (pz237)

August 2020

1 Introduction

In this project, we aim to apply what we have learnt in the Deep Vision course to implement an application that could help people self-improve pose alignment in yoga practices. Given a picture of a person doing a yoga pose, the system would use a deep learning classifier to predict the actual pose name. Subsequently, based on this predicted pose class, it would generate a picture of that same person doing the standard pose with 'corrected' alignment. By comparing the original pose with the corrected pose, this person could learn to improve his/her yoga pose alignment without a teacher's supervision.

Our project is based on the Yoga-82 dataset, a diverse dataset of 82 complex yoga poses [1] (https://arxiv.org/pdf/2004.10362.pdf). This dataset contains a varying number of images in each class from 64 (minimum) to 1133 (maximum) with an average of 347 images per class. These images contain yoga pose with clean background as well as in the wild (i.e. with random backgrounds in the forest, beach, indoor, etc.). In addition, some images only contain silhouette, sketch, and drawing version of yoga poses. The dataset provides web links of images along with train and test splits. The dataset is released by a group of researchers from Osaka University, Japan, and IIT Gandhinagar, India, in April 2020.

2 Classification (Robin Ruland)

Abstract

Recognising human actions or poses is an important problem in computer vision. This is mostly done by using human pose estimation, which focuses on predicting the localisation of human joints (keypoints).
In the paper [**B-CNN**] from 2017 a method called hierarchical classification was introduced in which a model outputs multiple predictions from coarse to fine. It was shown that the coarse labels can help the model to learn and improve

the accuracy on the fine labels.
This method was used in a resent paper [**Yoga-82**] from 2020 in which a human pose recognition problem was solved by only using hierarchical classification. Therefore they constructed a dataset called Yoga-82. The model introduced in this paper outperformed different state-of-the-art convolutional neural network architectures on this dataset.
Since the new methods works without any keypoint prediction it leads to the question if a combination of both methods can yield a better result than the methods on there own. The goal of this part of the project is to combine the pose estimation approach and the hierarchical classification approach and try to exploited the benefits of both methods in one model.

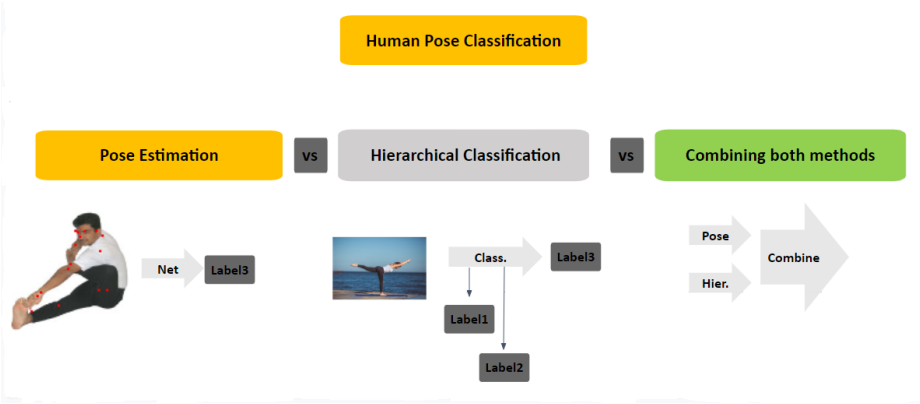

Figure 1: goal of this project

2.1 Related work

This project mainly focuses on [**Yoga-82**] in which a version of [**DenseNet**] was extended as in [**B-CNN**] to enable hierarchical classification on the Yoga-82 dataset.
In [**DenseNet**] they use so called dense blocks in which each layer takes all preceding feature-maps as input. This helps alleviating the vanishing-gradient problem, strengthen feature propagation, encourage feature reuse and substantially reduce the number of parameters. This leads to nets that are substantially deeper, more accurate, and more efficient to train.
In [**B-CNN**] they use a normal CNN but extend it in a way, that at some points of the net the output of the model at that point is used and fed into an own net called branch which predicts an additional label. By backpropagation from all outputs of the model the model learns to not only predict the final label but also learns to get meaningful enough representations at the chosen points in the net to let the branches classify the branch labels.

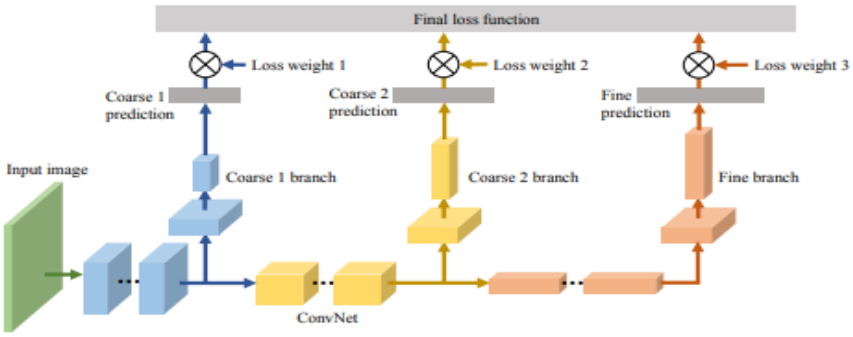

Figure 2: Architecture of Branch Convolutional Neural Network. The network at the bottom can be an arbitrary CNN. There can be multiple branch networks and each of them outputs a coarse prediction. The final loss function is a weighted summation of all coarse losses.
Source: https://arxiv.org/pdf/1709.09890.pdf

In [**Yoga-82**] the Dense-201 net is used and 2 branches are added to predict the 2 coarse labels. They have 3 variants with different positions of the branches and in variant 3 the first branch is a dense block itself followed by a fully connected layer. All other branches are just simple 1 layer fully connected nets. Other papers which inspired this project are [**Human pose**], [**3D pose**].

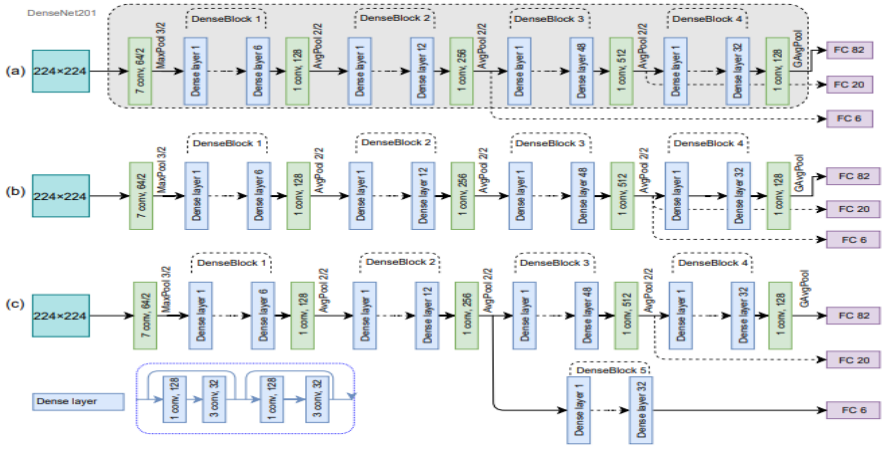

Figure 3: DenseNet-201 modified hierarchical architectures.
Source: https://arxiv.org/pdf/2004.10362.pdf

2.2 The Yoga-82 Dataset

For the classification task the dataset uses a three-level hierarchical structure with 6, 20, and 82 classes in the first-, second-, and third level, respectively. The dataset is split by [**Yoga-82**] into a training set of 21,009 images and a test set of 7,469 images. Furthermore to evaluate the models the training set is split into 20% evaluation and 80% training for this project.

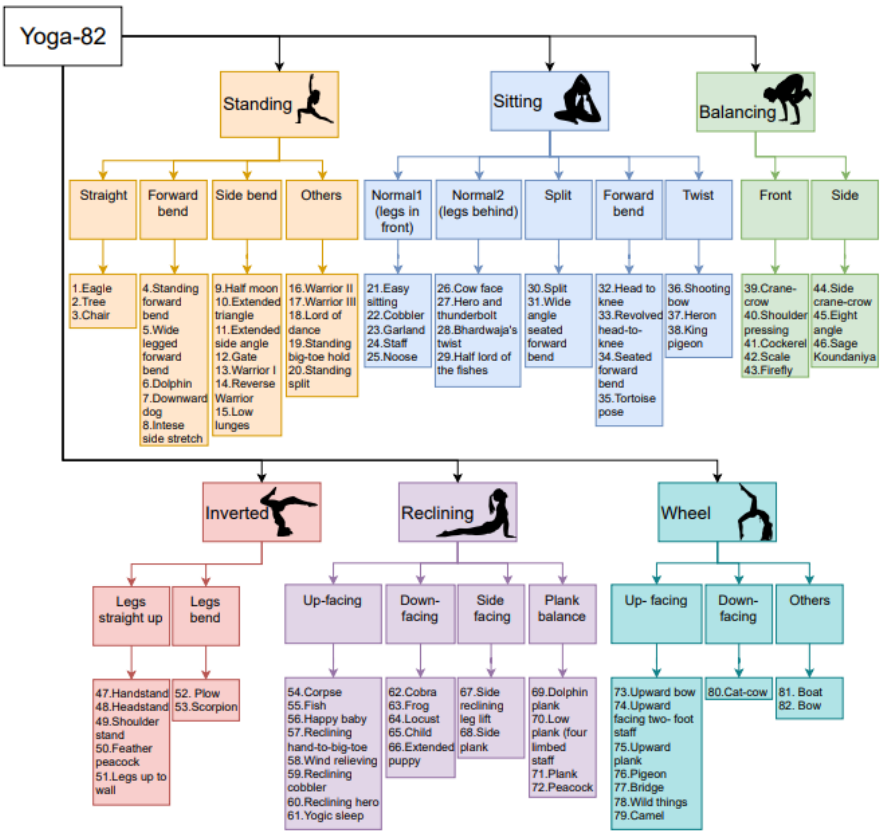

Figure 4: Yoga-82 dataset label structure. Hierarchical class names at level 1, 2, and 3

2.3 Conception

The main goal of this project is to investigate if and to what extend a combination of the 2 previously explained methods can improve the overall accuracy or boost the speed in which the model is learning.
Besides an own model a model like in [**Yoga-82**] is used in this project and further extended to utilise both keypoints and hierarchical classification.

For the pose estimation part of the models the "Keypoint R-CNN" from "torchvision.models" is used to generate the keypoints which are used as features for a net. Since the keypoints can be viewed as high level features the net is a very simple net out of 4 fully connected layers.
To combine pose estimation with hierarchical classification two methods are introduced in this project.

2.3.1 Method 1

Here the model learns features from the keypoints and from the images separately and than combine the features later on to generate the final prediction.

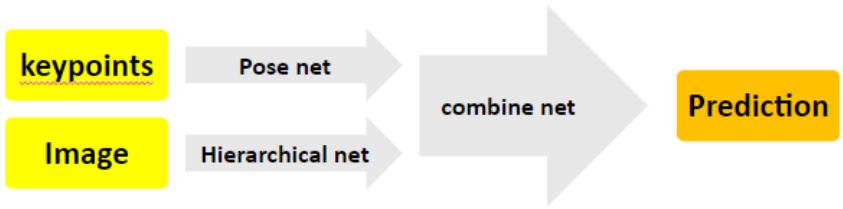

Figure 5: Sketch of the first combination method

2.3.2 Method 2

For this method the keypoints are reconstructed in a image like fashion. This is done by using a zero matrix with the same size as the images and adding 1 to each coordinate where the pose estimation model predicts a keypoint. The constructed keypoint images are then added as 4th channel to the RGB images and the model is trained on this 4 channel datapoints.

Figure 6: Sketch of the second combination method

2.4 Models

In this project an own model was constructed as well as the model from [**Yoga-82**] reconstructed and both extended to combine keypoints and hierarchical classification.
In 7, 8, 9, 10, 11 the models are shown graphically. Max pooling (Max P in the graphics) is always size 2x2 with stride 2. BN means batch normalisation, drop means dropout and all convolutional layers have stride 1 and padding 0.

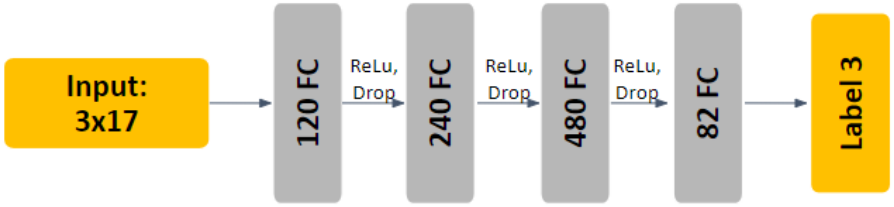

Figure 7: Sketch of the pose model

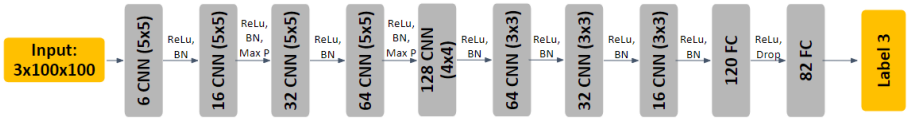

Figure 8: Sketch of the CNN model

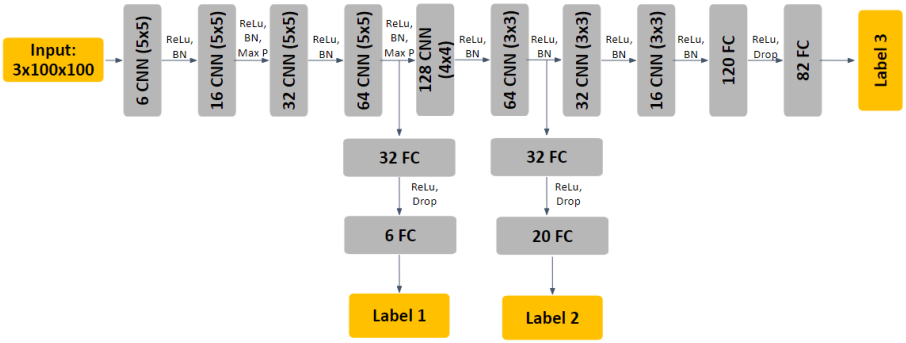

Figure 9: Sketch of the hierarchical model

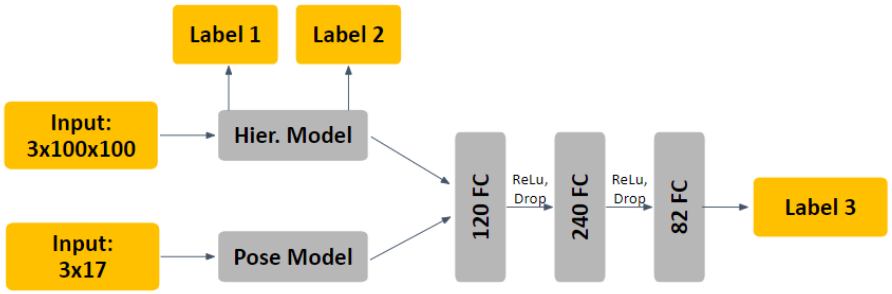

Figure 10: Sketch of the method 1 model

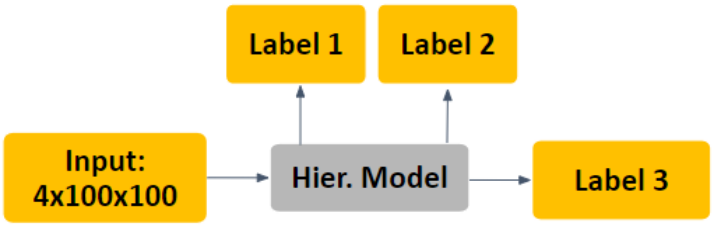

Figure 11: Sketch of the method 2 model

The hierarchical DenseNet model is like version a in 3 but to make it feasible on the used setup the growth rate is reduced to 6 (instead of 32 as in the paper) and the dense blocks are all 5 layers deep (instead of 6,12,48,32 in the first,

second, third and fourth dense block as in the paper). In comparison a typical small version of DenseNet has only 3 dense blocks which are all 5 layers deep and a growth rate of 12. The DenseNet models for combining keypoint with hierarchical classification are implemented in the same way as in the own model.

2.4.1 Hyperparameters

Like in [**Yoga-82**] a learning rate of 0.003 is used, the labels are all equally weighted, Cross-entropy loss is used as the loss function with stochastic gradient descent and momentum of 0.9. Since the batch size is not stated in the paper and a higher batch size is not feasible for the used setup the batch size is set to 16 and dropout is set to 0.2. Furthermore is the learning rate multiplied by 0.5 every time the loss plateaus.
All experiments were conducted on a system with AMD Ryzen^{TM} 5 2600 CPU (6x 3,40 GHz), 16 GB RAM, and an NVIDIA GeForce GTX 1660 Ti GPU with 6 GB memory.

2.5 Results

All models were trained for either 40 or 100 Epochs. In 12 the label 3 (the most difficult) accuracy of all models are shown.

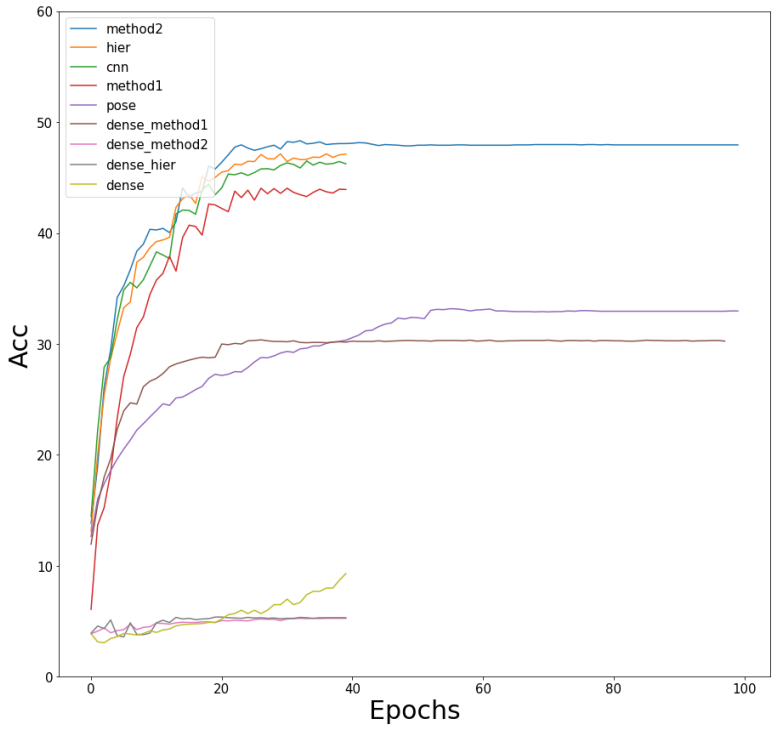

Figure 12: Accuracy of the models on the evaluation dataset

2.5.1 Results on the test set in %

Model	top evaluation accuracy	test accuracy
Pose model	33.1789	31.5121
CNN	46.4968	40.0073
Hierarchical CNN	47.1337	41.2779
Method 1	44.0359	38.6459
Method 2	48.3207	41.5683
DenseNet	9.3225	2.7773
Hierarchical DenseNet	5.3850	3.1585
Method 1 DenseNet	32.1366	29.3883
Method 2 DenseNet	5.2692	3.3400

2.6 Critical Review

As we can see the DenseNet models that are feasible for my setup are properly to shallow to learn a complex dataset like yoga-82. The "DenseNet" model that works like the model in the Yoga-82 paper only achieves 3.1585% accuracy but the model in yoga-82 achieves 79.35% by having a higher growth rate and deeper dense blocks and it properly was also trained over more epochs (this is not stated in the paper).
The "Method 1 DenseNet" probably only uses the pose estimation part to classify which is why its accuracy is close to the accuracy of the "pose" model. Because it also has to backpropagate through the DenseNet model it learns a bit slower.
Also both hierarchical models do perform better than without hierarchical classification but only by about 1.27% and 0.4% on the test set.
The best result of 41.5683% is achieved by using the keypoints as 4th channel in the RGB images, but it is only about 0.3% better on the test set and about 1.2% better on the evaluation set than the hierarchical model.
It might be that the hierarchical classification model learns features which are very close to keypoints and therefore combining both methods dose not yield much new information accessible to the net compared to only using pose estimation or hierarchical classification on its won.
The hierarchical classification is more complex than only classifying one label so maybe to get the full advantage of this method the training needed to be longer. This would also explain why in the yoga-82 paper they raised the accuracy from 74.91% with DenseNet-201 to 79.35% in there best hierarchical DenseNet version.

2.7 Conclusion

Both using hierarchical classification compared to a normal model and combining pose estimation with hierarchical classification compared to only using hierarchical classification yields a quite small advantage. Admittedly to really see the advantage of combining both methods further research needs to be done. A better setup should be used to train deeper nets for more epochs. Also in this project a pretrained pose estimation model is used and since the dataset shows humans in "unnatural" positions the predictions of the keypoints are sometimes very weak. A pose estimation model trained on yoga-82 could boost the result a lot if the features learned by the hierarchical part are not very similar to the keypoints. Also a lot of Hyperparameter optimisation and fine cut of the models could be done which were not doable within the scope of this project.

3 Image Generation (Zhao Sun)

3.1 Modification from the Original Proposal

In the original project proposal, we had stated that we wanted to synthesize images based on a Generative Adversarial Network (GAN). However, as we deepened our research on the related works [8,9,10], we realized that the GAN-based proposal is not suitable for our project due to limitations in our chosen dataset. The Yoga82 dataset contains labelled images of yoga poses in the wild; it lacks person re-identification, i.e. it does not have the same person doing different sets of yoga poses. Hence, a discrimination of real (ground truth images) vs fake (synthesized images) is impractical. Subsequently, as we progressed with the Deep Vision course, we were inspired by the last exercise on conditional Variational Autoencoders (VAE) [2] as well as Patrick Esser's paper on VUNet [1]; we have thus decided to adopt a conditional VAE architecture instead of GAN for the image synthesis part of the project.

3.2 Objectives and Related Works

The aim of the generative model is to synthesize an image of a person doing a yoga pose based on an input image of the same person and a label for a target pose. Figure 13 illustrates the overall schema.

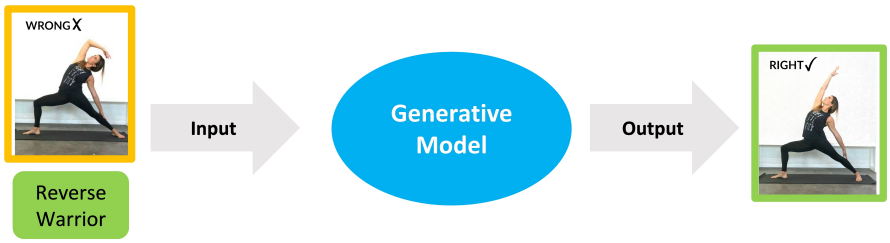

Figure 13: The overall schema for yoga pose generation conditioned on a pose label.

In order to achieve this, we can break the task down into several sub-parts: Human Poses Estimation [3,4], Pose Transfer [1,8,10], Text-to-Image Generation [5], etc. While each of the sub-topics is interesting and worth being researched separately, we have, in the interest of time, opted to adapt existing models rather than developing our own end-to-end ones from scratch. For example, the standard off-the-shelf models from the Pytorch library are used for the subtask key points detection and the conventional method of one-hot encoding is applied to convert the text-based pose labels.

The key challenge in this project is two-fold: firstly, our model needs to disentangle appearance from pose for an input image; secondly, it needs to learn the association of yoga poses with the respective labels so that it can generate

a desired pose conditioned on a given label. We have made a few attempts at different model architectures, as detailed below.

3.3 Model Architecture I

This first attempt is based on a combination of VUNet [1] with modifications inspired by the conditional VAE in the Deep Vision Exercise Sheet 10.

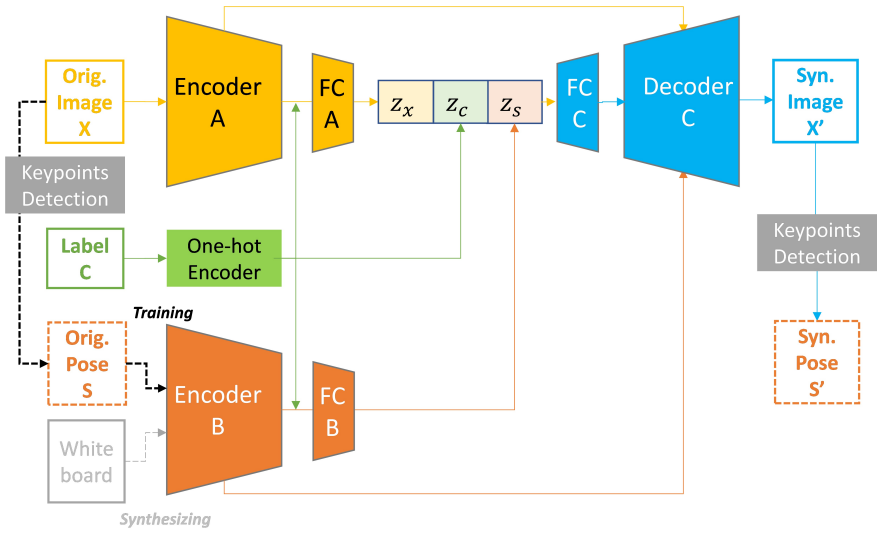

Figure 14: Illustration of the schematic diagram on Model Architecture I.

Overall the model has a joint-UNet structure with two Encoders and one Decoder. Each encoder/decoder unit consists of three residual blocks [6], two downsampling/upsampling layers, and four fully connected layers, respectively. Encoder A takes the original image as input; Encoder B takes the pose estimation as input, which is derived from feeding the original image into a pre-trained key points detection model based on Mask R-CNN ResNet-50 FPN [2]. The text-based pose label is converted via one-hot encoding [5] before feeding into the fully connected layers. Instead of using 82 binary bits to directly encode the corresponding 82 yoga poses in the database, we retain the hierarchical classification structure by using the first 6 bits to represent the Level 1 classification, the next 6 bits to represent the Level 2 classification and the last 8 bits to represent the Level 3 classification. Finally, skip-connections have also been separately established between each encoder and decoder.

The intention behind the model design is to represent the yogi's appearance X by latent features z_x and the pose shape S by latent features z_s, while simultaneously conditioned on pose label C during model training. When synthesizing

images, we only need the raw image X and pose label C as inputs, as we replace the target pose by a 'whiteboard' and sampling the latent feature z_s at random.

Loss Function

The loss function associated with this model is the sum of the following items:

- Negative conditional log-likelihood [2], $\breve{}logP(X|C)\breve{}logP(S|C)$ of image X and pose S given a conditioning C, the text-based label in this case. Similar to the approach shown in the lecture, we obtain the evidence lower bound (ELBO):

$$\log p_\theta(\mathbf{y}|\mathbf{x}) \geq -\boldsymbol{KL}\left(q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})\|p_\theta(\mathbf{z}|\mathbf{x})\right) + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})}\big[\log p_\theta(\mathbf{y}|\mathbf{x},\mathbf{z})\big]$$

  by assuming the latent variables z_x and z_s are statistically independent of input variables X and S, i.e. $p(z|x) = p(z)$. By further assuming the priors to be a diagonal unit Gaussian, we can simplify the loss function to be the sum of the negative Kullback–Leibler divergences between the variational posterior and the prior on the latent variable z_s and z_x.

- Perceptual loss between the original input image X and the reconstructed image X', which is obtained by adding the content loss and the style loss derived from a VGG19 model [7]

- Pixelwise MSE Loss between the pose estimation S derived from the original image X and the pose estimation S' derived from the reconstructed image X'

Model inputs

- Input image size: 100x100x3
- Number of model parameters: 2.6M
- Batch size: 10 [constraint of GPU]
- Learning rate: 1e-3
- ADAM optimizer

Results Discussion

After training for 5 epochs, we inference from the model to generate the following images:

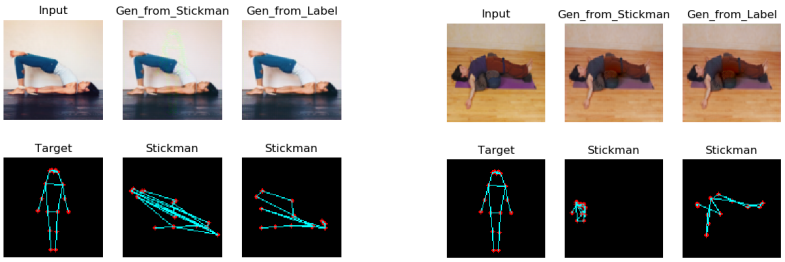

Figure 15: Output from Model I

The model fails to generate the desired output, i.e. a synthesized image of the input person doing the target pose. Instead, it results in an image of the same person doing the original pose. There are several potential causes for the model failure:

- The model fails to separate poses from appearances in the feature space, probably due to the design of the network which might require more layers of depth.
- The conditioning influence from the pose label is too weak, probably due to the limitation of the dataset, i.e. for each pose, the number of images per class varies from 64 to 1133. For such small sample sizes (and without any data augmentation), the model cannot effectively learn to associate pose shapes with labels.
- The loss function is probably biased towards enforcing the synthesized image to be as similar as possible to the original image, resulting in the skip-connections being over-weighted to channel the original image to the output.

In addition, the training is extremely slow due to the loss function requiring the reconstructed image to be fed into an external VGG-19 model for perceptual loss calculation as well as the key points detection model for comparing the pixel-wise stickman figures. We have subsequently decided to limit our target poses to only 10 classes, instead of the entire set of 82 classes in the database. We have chosen our yoga classes carefully such that the hierarchical classification structure is retained and well represented.

3.4 Model Architecture II

In the second attempt, we have simplified the model structure to include only one encoder and one decoder. In order to speed up the training time, the encoder now takes a pre-processed image which is a combination of a segmentation layer (blue channel) based on the original image, a key-point layer (red channel)

and a stickman layer (green channel) connecting the key joints. This alteration on the input is intended to focus the model on the shape of the human pose in the foreground rather than all the pixels in the background. Like before, our aim is to condition the model on learning various poses associated with the hierarchical class labels.

The encoder and decoder have the same basic components as the previous model, with each unit consisting of 3 residual blocks, 2 downsampling/upsampling layers, and 4 fully connected layers, respectively. We attempt to abstractly represent only the yoga pose by superimposing a mask layer on top of the stickman figure, in contrast to considering both appearance and shape in the previous model. The one-hot encoder for the hierarchical class label is kept the same as before, i.e. the conditioning label is fed into the fully connected layers as well as concatenated with the latent features z_x before the decoder. We have experimented with two different versions of design: with skip-connections and without skip-connections between encoder and decoder.

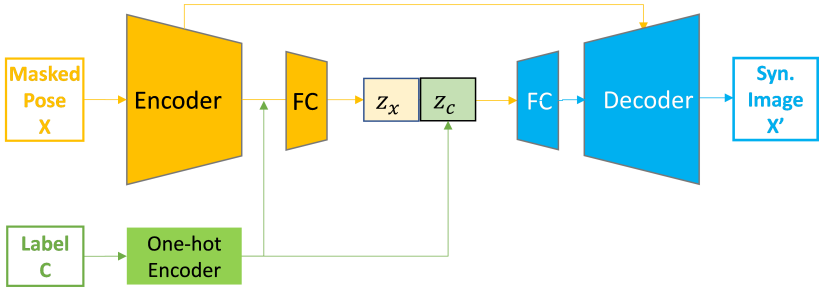

Figure 16: Illustration of the schematic diagram on Model Architecture II.

Loss Function

We also further simplified the loss function in order to speed up training. The loss function now only includes two items:

- Negative Kullback–Leibler divergence between the posterior and the prior on the latent variable z_x, assuming the prior to be a diagonal unit Gaussian
- Pixelwise MSE Loss between the synthesized image and the input

Model inputs

- Input image size: 100x100x3
- Number of model parameters: 1.9M
- Batch size: 32 [constraint of GPU]
- Learning rate: 1e-3

- ADAM optimizer

Results Discussion

After training for 20 epochs, we inference from the model to generate the following images:

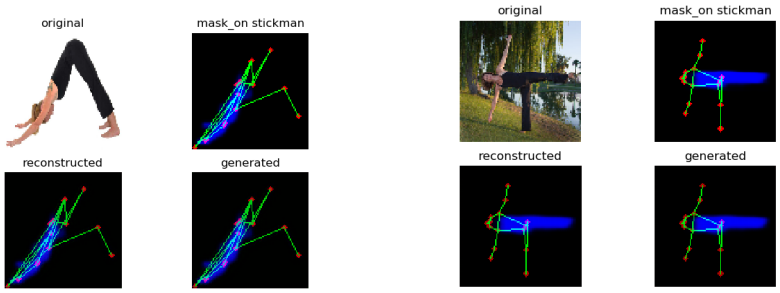

Figure 17: the top row shows the original image and the pre-processed input image consisting of a mask layer on the key points and stickman figure; the bottom row shows the reconstructed image and the generated image.

This simplified model again does not generate the desired output, i.e. we failed to condition the model on the pose label. Like before, the output is exactly like the input image. This is probably due to the presence of the skip-connections that the model just learned to transfer the input image directly into the output image. Furthermore, in order to test our hypothesis on the skip-connections, we have tested the model with the skip-connections removed. We have also experimented with replacing MSE loss with L1 loss for the pixel by pixel comparison.

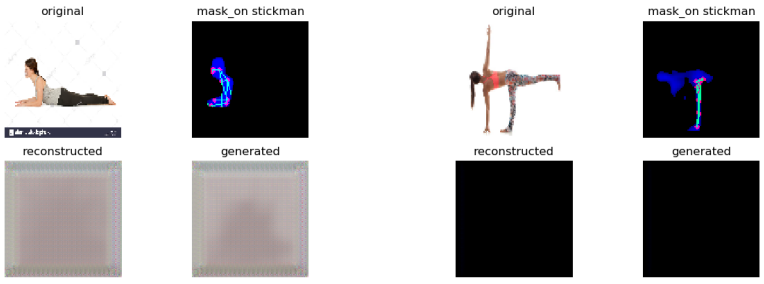

Figure 18: Results from removing skip-connections from the model; the 4 images on the left are the experimental results under MSE loss; the 4 images on the right are associated with L1 loss.

Notice the inaccuracies associated with the mask and key points detection generated from the original image by the pre-trained models. These are probably due to the inherent limitation of the yoga82 dataset, which might also contribute to the failure of the model to separate the poses conditioned on the label.

3.5 Reflections and Possible Future Work

We originally aimed to generate standard yoga poses with correct alignment from an input image with misaligned posture. However, we faced the following limitations:

1. The accuracy of the Pytorch pre-trained key points detection model for human pose estimation is low for the Yoga82 dataset.
2. We cannot train our own pose estimation model due to the lack of keypoint ground truth labelling in the Yoga82 dataset.
3. The Pytorch segmentation model for the mask layer is similarly dysfunctional when applied to the Yoga82 dataset.
4. The above limitations in input quality render loss function optimization during training futile (garbage-in garbage-out).

In the future, we would greatly benefit from a better suited dataset, that should include multiple sets of yoga poses performed by the same set of persons, preferably with labelled body parts. We could further enlarge the training dataset via data augmentation. We need a better model to generate sensible human stick figures and masks. Moreover, an advanced future target would be to incorporate video streams of people doing yoga. This would provide greater scope to train the model as each video frame would not only provide incremental example poses, but also the graph of pose transformations necessary to more accurately generate pose images or video.

References

[1] Manisha Verma, et al. *Yoga-82: A New Dataset for Fine-grained Classification of Human Poses (2020).*

[2] Xinqi Zhu, Michael Bain. *B-CNN: Branch Convolutional Neural Network for Hierarchical Classification (2017).*

[3] Gao Huang, et al. *Densely Connected Convolutional Networks (2018).*

[4] Amy Bearman, Catherine Dong *Human Pose Estimation and Activity Classification Using Convolutional Neural Networks.*

[5] Diogo C. Luvizon, et al. *2D/3D Pose Estimation and Action Recognition using Multitask Deep Learning (2018).*

References

Image Generation

1. Esser, et al. A Variational U-Net for Conditional Appearance and Shape Generation. (2018)
2. Sohn, et al. Learning Structured Output Representation using Deep Conditional Generative Models (2015)
3. He, et al. Mask R-CNN (2018)
4. Cao, et al. Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. (2017)
5. Reed, et al. Generative Adversarial Text to Image Synthesis. (2016)
6. He, et al. Identity Mappings in Deep Residual Networks. (2016)
7. Johnson, et al. Perceptual Losses for Real-Time Style Transfer and Super-Resolution (2016)
8. Siarohin, et al. Appearance and Pose-Conditioned Human Image Generation using Deformable GANs (2019)
9. Zhu, et al. Progressive Pose Attention Transfer for Person Image Generation (2019)
10. Ma, et al. Pose Guided Person Image Generation (2017)