Final Project Report - Human Pose Estimation

Denis Kim, Valeriy Novossyolov, Yernur Nurambek

***Abstract*—The main aim of this project is to reduce the computational complexity of a human pose estimation neural network and fine-tune its hyperparameters without significant reduction to the accuracy of the model. Our team looked at several different pose estimation models and chose the Stacked Hourglass neural network. Our experimental results show that it is possible to achieve relatively the same accuracy by reducing the number of convolution layers. Furthermore, our results show that the best performing seems to be RMSprop because it significantly increases the accuracy of the trained model. Since our team had limited computational resources, a comparison of models was performed after they were trained for five epochs.**

I. Introduction

Human pose estimation is the problem of identifying the human joints and their orientation in space in images or video. Similar technology already found its use in video filters. For example, Facebook or Snapchat filters estimate human face location and dimensions to apply video filters (e.g. adding headwear or makeup) in real-time. Estimation of the entire human pose would allow it to use more advanced video filters or use it in video entertainment systems, such as virtual reality gaming, to enhance the user's experience. Additionally, progress in this field can be applied to robotic vision, human-object interactions, and action recognition. Our project will use deep learning on a dataset of images with annotated body joints. A fully trained classifier should be able to estimate the human pose on a picture containing one subject.

II. Problem Definition

The main challenges of human pose estimation problem are increasing accuracy and efficiency. In our project, we tried to boost efficiency without detriment to accuracy by modifying the Stacked Hourglass model [1]. Our additions include modifying the model's architecture, fine-tuning the hyperparameters, preprocessing the data from the MPII dataset, and the addition of a graphical user interface for a better interpretation of the model's output.

III. Related Work

Human pose estimation is a well-known topic in the scientific community, and several research teams tried to estimate the human pose using deep learning approaches. The dataset MPII [2] that we are going to work with consists of around 25K images containing over 40K people with annotated body joints. Several approaches were used for training a model with this dataset.

The most successful one in terms of both the accuracy and the efficiency was published recently, in February 2020 [3]. The authors propose the gated skip connections with per-channel learnable parameters design and a hybrid network structure that combines the HourGlass and U-Net architectures. The essential idea is to learn how much information from the previous stage is propagated into the next one per channel and decrease the number of identity connections within the network. The results of this approach showed high accuracy, 94.1 percent of correct keypoints.

While [3] applied a hybrid of HourGlass and U-Net architectures, [4] proposes so-called coupled U-Nets (CU-Net), meaning coupling each U-Net pair through the connections of their semantic blocks. Both [3] and [4] aimed to decrease the number of parameters in the model. The results show that for an equal number of parameters, [3] outperforms [4].

On the other hand, [5] introduces the Cascade Feature Aggregation method, which cascades several HourGlass networks, with a multi-stage approach. The authors used the same dataset as [3], and their results are slightly lower, but still sufficiently accurate, 93.9 percent of correct keypoints.

The latest approach proposed by [6] uses high-resolution rich representations to accurately and precisely estimate the human pose. The authors highlight that to achieve this, they connected high-to-low resolution subnetworks in parallel and introduced multi-scale fusion so that each subnetwork repeatedly receives information from other units. Using the MPII Human Pose dataset, this approach achieves satisfactory results, reaching 92.3%. Nevertheless, the ablation study provided in this paper gives essential insights into the problem of human pose estimation and might guide our future research.

Another problem that should be addressed in this field is the computational power requirements of the top-performing approaches. Authors of [7] created a lightweight neural network approach for human pose estimation. The researchers used a single RGB camera located on a mobile device, or to be more specifically a smartphone, and estimated the 3D representation of a human pose to animate a mobile avatar. The main aim of this research was not limited to the use of deep neural networks to estimate the human pose in real-time, but also to reduce power consumption related to the model's prediction and training process.

IV. Proposed Deep Learning Model

For this project, we decided to use the Stacked Hourglass model found at [1]. The model uses a set of convolution blocks followed by a predefined number of Hourglass modules. Hourglass module consists of some convolution and max-pooling layers to reduce the initial image resolution. After achieving the lowest resolution, the model starts to upsample the image features in a top-down sequence. To combine the upsampled information, the model does the nearest neighbor upsampling of the lower resolution followed by an elementwise addition of the two sets of features. Since the architecture of the Hourglass

TABLE I
PROPOSED MODEL ARCHITECTURE OF THE CONVOLUTION LAYERS

Layer	Input Size	Output Size	Kernel Size	Stride	Padding	Input Channel	Output Channel
BatchNorm + ConvLayer + Relu	256	128	7	2	3	3	64
BatchNorm + ConvLayer + Relu	128	128	3	1	1	64	64
BatchNorm + ConvLayer + Relu	128	128	1	1	0	64	128
MaxPool	128	64	2	2	0	128	128
BatchNorm + ConvLayer + Relu	64	64	3	1	1	128	128
BatchNorm + ConvLayer + Relu	64	64	1	1	0	128	256
Total number of Learnable Parameters: 235 328							

is symmetric, the upsampling is followed by the same number of deconvolution layers as the number of convolution layers. In the end, each Hourglass module produces a heatmap for each joint location. The heatmap is a probability matrix that contains the probability of finding a particular joint at particular coordinates.

As for our additions to the model, they include: preprocessing of data from MPII dataset, conversion of joint coordinates into a set of heatmaps which is used as an input to the model, simplification of the initial convolution layers that precede the set of Hourglass modules, the introduction of new optimizers, use of scheduler to change the optimizer's learning rate during the training process, and addition of a graphical user interface for visualization of the model's output.

The data used to train the network was taken from a dataset called MPII Human Pose dataset. This dataset is one of the state-of-the-art benchmarks for evaluating articulated human pose estimations. They provide about 25000 images with annotated body joints. The annotations originally come in a .mat format and have the following information: image name, body annotations for each person in the image, coordinate of the head rectangle, scale of the person, object position, and annotated key joints. In our project, we used a slightly different version of the annotations in .json format that contains only the name of the image, scale of the person with respect to 200 px height, coordinates of the person's center, and annotated key joints. Using this information we processed the data in the following manner. Firstly, we cropped the image by finding the top side and the left side according to these equations:

$$top = center[1] - scale * 105$$

$$left = center[0] - scale * 60$$

And the bottom side and the right side were calculated according the following:

$$bottom = top + scale * 210$$

$$right = left + scale * 120$$

If any of the sides overflowed the dimensions of the original image, it was set to the corresponding edge of the image. Although the original annotations suggest multiplying the scale by 200 in height, our observations showed that this approach is not accurate because many of the images lose the joints of the legs after cropping. Therefore we increased the number of pixels to 210. The cropped image is then transformed into a square and resized to the desired dimensions. In order to avoid any significant distortion of the image, as horizontal stretching, or vertical shrinking, we added zero-padding for squaring the cropped image. The original picture and the result of preprocessing can be seen in Fig. 1.

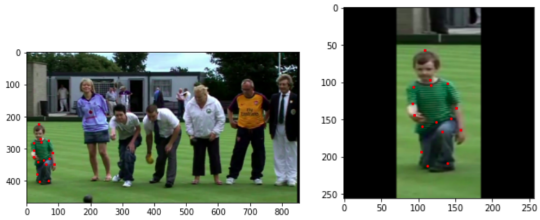

Fig. 1. Example of original image and corresponding processed image

To be more precise, most of the original model's structure remained unchanged. Our changes to the original model architecture affected only the first set of convolution blocks that precede the Hourglass modules. The original model used three Residual blocks that consisted of 3 sets of a batch normalizer, a ReLU activation function, and a convolution layer. Each Residual block was paired with a SkipLayer block, which decided whether to use an additional convolution layer depending on the input and output resolutions of the input. In total, the original model used 12 Conv2D layers before passing the input image into the Hourglass set. In contrast, in our model, we decided to decrease the number of convolution layers to 5 by replacing the Residual block with simple Conv2D layers. This approach led to the reduction of learnable parameters in the model. Originally, there were 350 848 parameters of convolutional layers in the part preceding the Hourglass modules, in contrast, our model has 115 520 fewer parameters. The final architecture of the initial part of the model can be found in Table I

Furthermore, we did a comparative analysis of three different optimizers, namely Adam, RAdam, and RMSprop, to select the best one for our final model. According to [8], RAdam provides a more robust implementation because it is less likely to converge into a local minimum in comparison to more traditional Adam and SGD optimizers. Considering that, our team wanted to see its performance in practice. Another addition related to the model's hyperparameters was an introduction of the scheduler which changed the learning rate of the used optimizer after several training epochs. The results of this analysis can be found in Section V. The final configuration of hyperparameters is presented in Table II

TABLE II
HYPER PARAMETERS OF PROPOSED MODEL

Optimizer:	RMSprop
Batch size:	16
Number of epochs:	10
Learning Rate:	2.5e-4

V. EXPERIMENTAL RESULTS

We tested six different configurations of the model and chose the best performing one as our final network. The main limitation of the current project was that we did not have enough computational power in order to properly train the models. Therefore, we trained each network for five epochs only and compared the behavior of the loss during the training. Based on this we selected the best one. Below is a description the six models:

1) Model 1
   a) Optimizer: RMSprop
   b) Network: Original without changes
   c) Loss after 5 epochs: 0.00113494
2) Model 2
   a) Optimizer: RAdam
   b) Network: Original without changes
   c) Loss after 5 epochs: 0.00129738
3) Model 3
   a) Optimizer: Adam
   b) Network: Original without changes
   c) Loss after 5 epochs: 0.00126786
4) Model 4
   a) Optimizer: Adam with scheduler
   b) Network: Original without changes
   c) Loss after 5 epochs: 0.00132553
5) Model 5
   a) Optimizer: RMSprop
   b) Network: Reduced with less learnable parameters
   c) Loss after 5 epochs: 0.00114283
6) Model 6
   a) Optimizer: Adam
   b) Network: Reduced with less learnable parameters
   c) Loss after 5 epochs: 0.0013586

Fig. 2. represents how each of the optimizers influences the motion of the loss.

We can see from this graph that the losses of models 3 and 4 show quite similar behavior through all five iterations. That means that the influence of the scheduler on loss is insignificant. All of the models make the biggest step after the first iteration and reach approximately the same level. But after that, we can see that model 1 overtakes others and shows the best result after the fifth iteration. Based on these observations it can be concluded that RMSprop performs better on this task than Adam and Radam optimizers.

Fig. 3. and Fig. 4. show how our simplified architecture affects the motion of the loss.

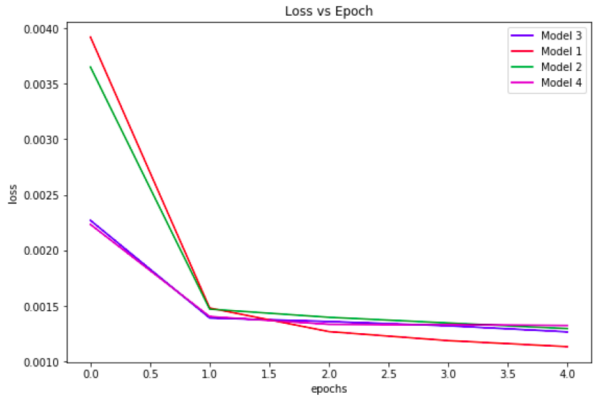

Fig. 2. Loss vs Epoch graph for different optimizers, Model 1 uses RMSprop, Model 2 uses RAdam, Model 3 uses Adam, and Model 4 uses Adam with scheduler.

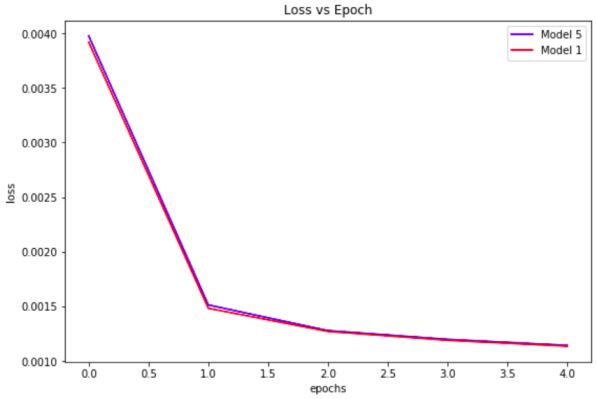

Fig. 3. Loss vs Epoch graph for original model and simplified model with RMSprop optimizer.

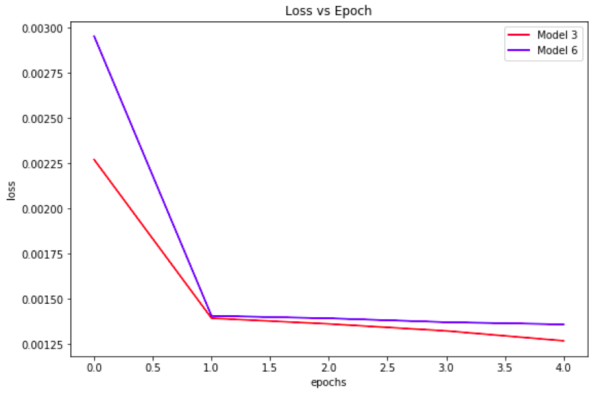

Fig. 4. Loss vs Epoch graph for original model and simplified model with Adam optimizer.

As we can see from these graphs, our reduced model does not negatively affect the performance of the models. Therefore based on the analysis, Model 5 was chosen as our final network. The accuracy of our final model is not high, due to the very few amount of training iterations. But the main purpose was to increase the efficiency of the model without performance decrease. This was done by reducing the number of learnable parameters without a negative effect on loss.

Therefore the aim of the project can be considered to be achieved.

Fig. 5. shows some of the results that we obtained:

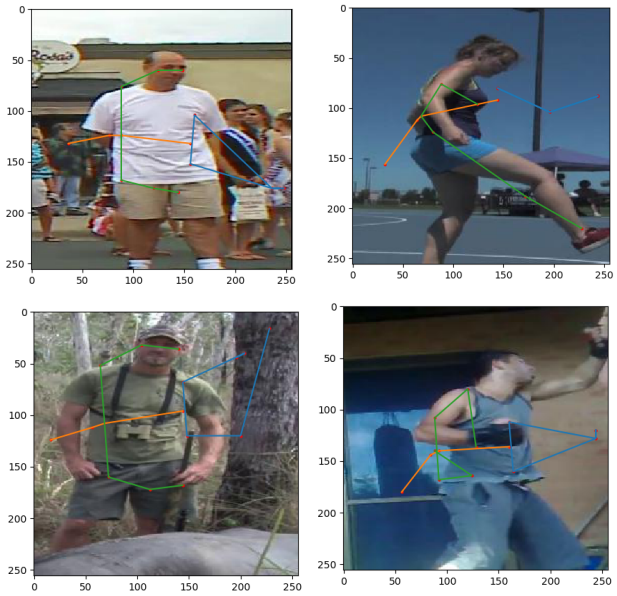

Fig. 5. Some of the results generated

VI. Conclusion

Human pose estimation using a deep learning approach has great potential for the future of video entertainment as well as for the enhancement of social interactions on platforms that support live video functions. Since the deep learning approach requires only video or image data taken from the user, it allows for a cheaper mobile solution, when compared to traditional approaches that use specialized sensors and cameras for motion capture techniques. The main objective of our project was to enhance the efficiency of the human pose estimation model in order to reduce the time spent on inference. This is crucial for the application of technology in both video entertainment and real-time robotic vision. After training five models with different combinations of network architectures and hyperparameters, the best model for human pose estimation was the reduced network with RMSprop optimizer. This result demonstrated that the network with fewer parameters, thus the one that spends less time for inference, is the most accurate. Therefore, the main goal of this project was achieved.

References

[1] N. Jain and S. Shah. Implementation of various human pose estimation models in pytorch. Available at https://github.com/Naman-ntc/Pytorch-Human-Pose-Estimation.

[2] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele, "2D Human Pose Estimation: New Benchmark and State of the Art Analysis," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[3] A. Bulat, J. Kossaifi, G. Tzimiropoulos, and M. Pantic, "Toward fast and accurate human pose estimation via soft-gated skip connections," 2020.

[4] Z. Tang, X. Peng, S. Geng, Y. Zhu, and D. N. Metaxas, "CU-Net: Coupled U-Nets," 2018.

[5] Z. Su, M. Ye, G. Zhang, L. Dai, and J. Sheng, "Cascade Feature Aggregation for Human Pose Estimation," 2019.

[6] K. Sun, B. Xiao, D. Liu, and J. Wang, "Deep High-Resolution Representation Learning for Human Pose Estimation," 2019.

[7] D.-H. Hwang, S. Kim, N. Monet, H. Koike, and S. Bae, "Lightweight 3D Human Pose Estimation Network Training Using Teacher-Student Learning," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, March 2020.

[8] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han, "On the variance of the adaptive learning rate and beyond," *ArXiv*, vol. abs/1908.03265, 2020.