Machine learning implementation for pose classification

Nguyen Xuan Tung

September 2, 2023

1 Introduction

The task of this assignment is to create an image classification model that can distinguish pose. For the detailed, I've implemented 2 models which based on 2 machine learning architecture: the VGG (Visual Geometry Group) and ResNet (residual network) architectures. After training the models, I deployed them and tested them on new dataset to see how well they can identify different poses.

2 Implementation

I have developed two models which are based from: the VGG (Visual Geometry Group)[1] and ResNet (residual network)[2] architectures. Both of them are a type of CNN, which is the most useful tool for this task. These models were employed to classify the pose of the resultant heatmap. Those heatmaps are stored in numpy array with shape [1, 17, 64, 48]. Each numpy files are belong to different folder belong to the same class, in this case, I use the name of each folder to label the dataset. The implementation was carried out in Python, and due to the dataset's large size, the training phase was executed on the Amazon SageMaker platform within the AWS cloud computing environment.

Model 1: This model is based on VGG architechctures. It consists of 4 blocks, each block is assigned two 2D convolution layers starting with 32 and ending with 256 filters. The activation function chosen is Rectified Linear Unit (ReLU). After the layers, max-pooling and batch normalization were applied for downsampling and stabilizing the training. In the last block, I include the drop-out (rate = 0.25) for regularization. The final layer is to flatten the output of previous layers and add fully connected (dense) layer with 256 units + ReLU activation function. The drop-out is also concluded for further regularization and prevent overfitting.

Model 2: This model is based on ResNet architechctures. It has the initial convolution layer with 16 filters, and a stride of (2, 2) for down-sampling. The

batch normalization is apllied for stability + activation function ReLU. Max-pooling was also performed with a 3x3 kernel and a stride of (2, 2) for further down-sampling. For the residual blocks, each block is applied two consecutive convolutional layers with batch normalization and ReLU activation. Optionally, adjusts the shape of residual using a 1x1 convolution if needed. And finally, the modified residual and the original residual were added to form a skip connection, then applies ReLU activation.

3 Results

In figure 1 we can see the training and validation accuracy increase after each epoch but the validation loss increase after epoch 4, which mean the model does not fit the new data very well. On the other hand, the model 2 (represented in figure 2) has good trend of accuracy and loss for both traning and validation. Moving to figure 3, both models predicts well from the actual heatmap.

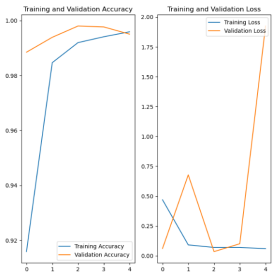


Figure 1: Accuracy and loss values of model 1

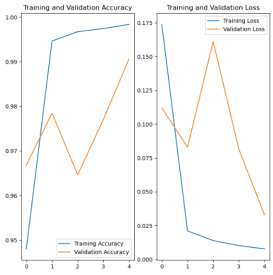
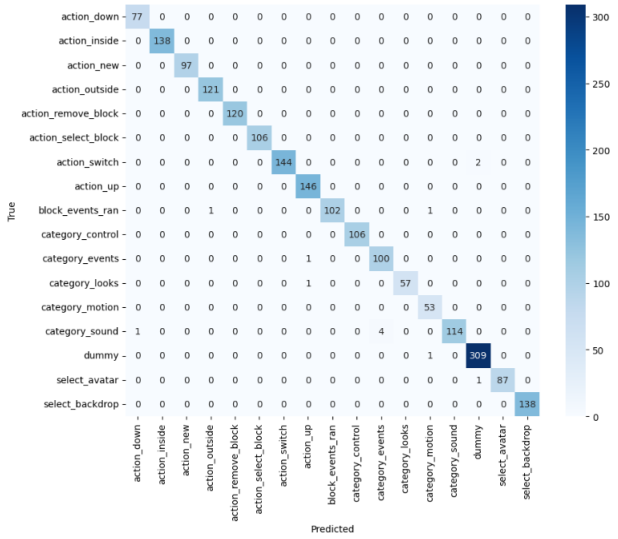
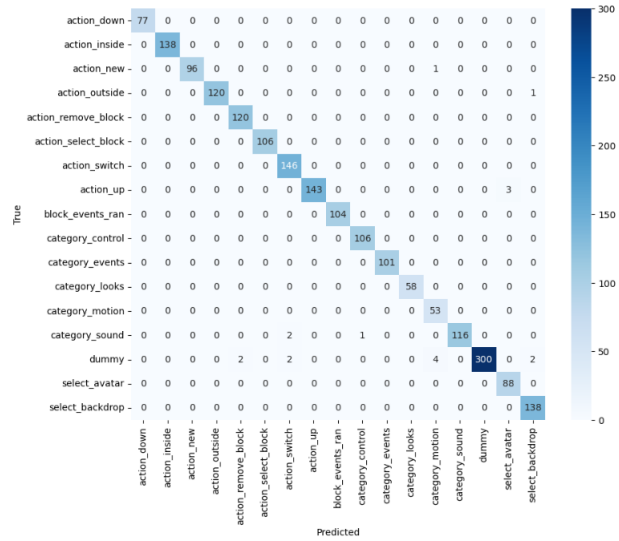


Figure 2: Accuracy and loss values of model 2



A



B

Figure 3: Confusion matrix: **A**: confusion matrix of model 1, **B**: confusion matrix of model 2

4 Deploy model

In order to test the effective of the models, I created a ViTPose algorithm[3] which reads and returns the images into heat maps. These heat maps are

represented as numpy arrays, with each numpy file represents a structure of [1, 17, 64, 48] – signifying the presence of 17 key points of the body. A collection of several images was employed for comprehensive model testing.

A

B

Figure 4: Models classification: **A**: Pose images classification of model 1, **B**: Pose images classification of model 2

5 Limitation and Improvement

Limitations: Both models have slow training times due to large datasets, making local training impractical. Cloud training is dependent on internet connectivity.
Improvement: Fine tune model and we can consider combine these two models for better performance in the future.

6 Accessing Detailed

For more detail and better visualization, access to jupyter notebook `Demo_model1`, `Demo_model2` for training model 1 and 2. `Model_deploy` for deploy model and visualize reuslts for new dataset. `Store_pic_new_data` for ViTPose argrith turning image into heat map with numpy array [1, 17, 64, 48].

7 References

[1] Nagasato, Daisuke , Tabuchi, Hitoshi , Ohsugi, Hideharu , Masumoto, Hiroki , Enno, Hiroki , Ishitobi, Naofumi , Sonobe, Tomoaki , Kameoka, Masahiro , Niki, Masanori , Hayashi, Ken , Mitamura, Yoshinori. (2018). Deep Neural Network-Based Method for Detecting Central Retinal Vein Occlusion Using Ultrawide-Field Fundus Ophthalmoscopy. Journal of Ophthalmology. 2018. 1-6. 10.1155/2018/1875431.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. (2015). Deep Residual Learning for Image Recognition. arxiv. arXiv:1512.03385

[3] Yufei Xu, Jing Zhang, Qiming Zhang, Dacheng Tao. (2022). ViTPose: Simple Vision Transformer Baselines for Human Pose Estimation. arxiv. arXiv:2204.12484v3