Head Pose Estimation Using Deep Convolutional Neural Network

INTRODUCTION

Head pose estimation has become a familiar problem in the field of Machine learning/AI and has many applications such as driver monitoring, attention recognition and multi-view facial analysis. This report is going to explain and justify the approaches taken to deal with the problem of developing a deep convolution neural network to identify the head pose of a person given an image.

DESCRIPTIVE ANALYSIS

The dataset was given in two sets; train.csv and test.csv. The train dataset had image paths along with their class labels unlike the test dataset. Therefore, train dataset was found suitable for the purpose of training and validating the neural networks. There were in total 2325 images with 9 tilt class labels (Vertical angle of the head) and 13 pan class labels (Horizontal angle of the head). On further analysis, image samples with tilt class labels were found unbalanced. Samples for 90 degrees and – 90 degrees were 13 times lesser than the rest of the class labels and this raised the idea of data augmentation for these particular class labels. On the other hand, image samples with pan class labels were found mostly balanced.

DATA PREPARATION

The training dataset was divided into 3 subsets i.e. train, validation, and test. To perform this task, hold-out cross-validation was used to first randomly divide the dataset into 80(Train):20(Test) ratio and the train was further divided into 80(Train):20(Validation) ratio. Images in all the subset were rescaled and normalized using ImageDataGenerator from the TensorFlow library. Generally, in an image, each pixel varies from 0-255 and it's highly recommended for the deep learning models to normalize them in the range of 0-1 to speed up the model training process. The data generators were defined for the train, validation, and test dataset. Also, we converted the images to grayscale so that the model doesn't try to learn non-essential features like different skin tones, eye color, etc, and just try to learn the orientation of the head.

MODELING

Two different Jupyter Notebooks were used to train models on the tilt and pan of the head pose respectively and therefore, these were considered as two different tasks. For training the model, we have used Convolution neural networks (CNN) which are known to perform better for complicated image classification problems and are much more flexible in terms of setting the parameters in the architecture. However, CNN includes lots of different types of hyperparameter tuning. Following section will discuss how the various types of decisions were made using performance analysis and/or literature review.

The task was formulated as a multi-class supervised classification problem where class labels were discrete numerical values. Usually, in the case of angle predictions, regression is considered to be a more suitable approach as we deal with the continuous angle values. However, this particular problem had discrete target values and therefore, it was considered as a classification problem.

Base Model Selection

Head pose estimation is not a novel problem and therefore, there are a few papers already being published online. However, these works are based on different head pose dataset and class labels. Our base model was inspired by a research paper [1] in which the author has experimented and recorded the performance using models like standard LeNet-5 and AlexNet model for head pose estimation. The size of the dataset (Prima head-pose) used by the author is similar to what we had for the given problem. The work seemed promising to use as a motivation for this task. Our base model was inspired by an LeNet-5 architecture having 5 layers (inclusive of fully connected and output layer) with MaxPooling

in between the two convolution layers. Instead of LeNet-5's tanh activation function, we used ReLu because of its simpler and faster computation which highly suitable for deep NNs.

Base Model Specifications

- Activation function: ReLu (Rectified linear unit) was applied between the dense layer over other activation functions like tanh and sigmoid as ReLu is much faster and hence, accelerates the convergence of SGD. In addition to that, ReLu doesn't have vanishing gradient problems like tanh or Sigmoid function. Also, the "SoftMax" function was added as an output layer. The idea behind SoftMax function is that it converts the numeric output of the last linear layer into probabilities which sum up to 1. The class with high probability should be considered as the labeled class
- Pooling Function: There was a need for a function to progressively reduce the spatial size of the representation so that the number of parameters and computation in the network can be reduced. For this, we used the MaxPooling function that chooses the maximum grid value in order to extracts the most important features. This would make the model training faster while consuming less memory.
- Optimizer: Adaptive Moment Estimation Optimizer was selected to make the model computationally efficient. Adam is relatively easy to configure and combines the best properties of RMSProp & AdaGrad algorithms which allows it to handle sparse gradients on noisy problems.

Performance Metrics

Our model was built to estimate the head pose which has various real-world applications, for example, the head pose of a driver can be used to identify if the driver is distracted and the car can automatically alert the driver using an alarm if his/her head is moved to either sides beyond a certain degree. The failure in predicting the correct head movement can cause serious accidents too. So, for this case, it is critical for the model to be accurate as the cost of False Negative is high. For instance, if a 90-degree head pose (Non-attentive - True Negative) is predicted/labelled as zero-degree head pose (Attentive - False Negative), the consequence can be very bad. However, there could be other applications of head pose estimation where the error cost can be different and low. As the given task was not focused in solving a particular problem and the cost of error was unknown, therefore we choose F1-score as it is more suitable metric in the case of unknown cost of error and imbalanced class labels. F1 score conveys the balance between recall and precision.

EXPERIMENTS

The base model with the specification mentioned above produced 99% accuracy in training but 90% accuracy in validation. This contrast in performance helped us to realise the generalization gap and the need for regularization in architecture. The network with regularization obtained much better results by increasing the bias, thus preventing overfitting and leading towards a stable solution. For the convolution layer, we used dropout technique over L2 regularization based upon performance analysis. The dropout technique helped our model to be less sensitive against the specific weights of neurons. This helps the model to become capable of generalizing the images well and less likely to overfit the training data.

As there was room to increase the performance on both training and validation, the next network was built even deeper and wider by increasing another layer with more neurons to see the impact on the accuracy and losses. The idea was to keep the network as similar as possible to isolate the impact of additional layers. We performed a few experiments to test the different optimizers. Based on the performance analysis, it can be concluded that Adam and RMSProp performed better in comparison to the conventional SGD. It was also noticed that the computational time taken by SGD was much more than the other two.

The same approaches were used to train on the "Pan" of head pose where the base model was improved using regularization and RMSProp/Adam optimizer. However, in pan, the model suffered from consistent oscillation of validation accuracy throughout the epochs as shown in Figure 2. The reason behind this might had been related to the low number of samples in validation dataset. Also, overall training and validation accuracy were not that high in comparison to what we got for tilt. One of the

causes of achieving low accuracy for pan could be less training data available for each class. As a fact, Deep learning algorithms works well if you provide huge amount of data. So, we tried Image Data Augmentation to increase the data samples that responded with the improvement in the performance. Nonetheless, still the model was suffering from loss in accuracy. A network inspired by VGG-11 model was also used as an experiment following the research work in a hope to increase the performance using a deeper model. Yet, there was no improvement. Further research was performed to find a solution for the challenge in classifying the horizontal orientation of the head. While finding whether classification or regression is better for head pose estimation, [2] did conclude based on their evaluation that regression performs better for both vertical and horizontal direction, however, classification performs poorly in the horizontal direction. This finding was a realisation about the poor accuracy that we were achieving and thus, it is recommended to use Regression for the horizontal direction. Because of the time constraint, this report wouldn't be covering the analysis with regression.

RESULTS

All the results are recorded in the Table 1 and Table 2 for tilt and pan respectively. The base model for both the task suffered from huge overfitting. Thus, it was necessary to use heavy regularization for this task. Model3 worked really well for tilt classes with 89.9% accuracy on validation and 0.91 F1-score. As shown in Figure 1, the confusion matrix from model3 displayed that the most of mislabelled images were of adjacent head tilt classes. We can say that it was hard for the model to learn the nearby by head angles. However, it was learning and generalizing the far away head angles really well. For example, there are 16 images of - 30-degree head pose mislabelled as -15 degree. If we use this model, let's say to solve a car alert system as discussed in the above section of the report where the classes could be binary (Attentive and Non-attentive), then under such case, this model could have performed much better because there are high chances of nearby head angles to fall under same category and hence, predicting the correct class.

ULTIMATE JUDGEMENT

Task 1- Head Tilt Classification: Based on the overall performance and weighted F1-score on our hold-out test dataset, model3 performed the best for predicting the tilt with the F1-score of 0.91. Apart from F1-score, this judgement also considered the following criteria:

- High accuracy and low losses on validation and training data.
- Less prone to over-fitting or dropping accuracy.
- Good generalization to the unseen dataset (testing dataset) with 0.91 % F1-score which can be even improved if we increase the samples overall.
- Based on the confusion matrix, we can say that the model can perform exceptionally well, in case of binary classification (Attentive or Non-Attentive).

Drawback:

- With more than one million parameters, the number of trainable parameters was high and hence, require more computational time.
- Trained on a small set of head pose dataset. Therefore, it may not be able to generalize well for other head pose dataset available.

Task 2 – Head Pan Classification: If someone has to perform classification task on the head pose estimation, then Model2 should be taken into consideration. In reference to Table 2, Model2 obtained the best results among all the other models including VGG and LeNet-5 with the F1-score of YY. Based on the literature review, it is also recommended to explore the Regression technique for this particular task.

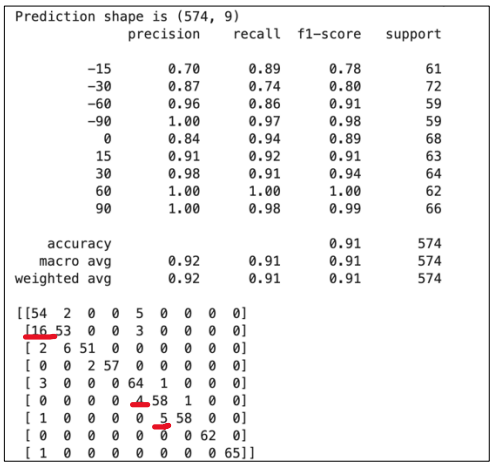
Drawback:

- Moderate level of accuracy on validation and test data.
- Regular oscillation of validation accuracy. This can be fixed by employing more data into the model.

APPENDICES

Model Performance : Tilt										
Model Name	Trainable Parameters	Data Augmentation	Train Accuracy : Loss	Validation Accuracy : Loss	Layers	Regularization	Activation function	Pooling	Optimizer	F1 Score
Model1	682,121	Yes	0.9943 : 0.0142	0.9022 : 0.3493	5	No	Relu	Max	Adam	0.87
Model1v2	682,121	Yes	0.9412 : 0.1591	0.9022 : 0.3100	5	Yes	Relu	Max	Adam	0.9
Model2	686,409	Yes	0.8392: 0.4029	0.8857: 0.3433	7	Yes	Relu	Max	Adam	0.88
Model3	1,146,569	Yes	0.9391 : 0.1973	0.8999 : 0.3713	7	Yes	Relu	Max	RMSProp	0.91

Table1: Top Results for Head Tilt Estimation



```
Prediction shape is (574, 9)
              precision    recall  f1-score   support

         -15       0.70      0.89      0.78        61
         -30       0.87      0.74      0.80        72
         -60       0.96      0.86      0.91        59
         -90       1.00      0.97      0.98        59
           0       0.84      0.94      0.89        68
          15       0.91      0.92      0.91        63
          30       0.98      0.91      0.94        64
          60       1.00      1.00      1.00        62
          90       1.00      0.98      0.99        66

    accuracy                           0.91       574
   macro avg       0.92      0.91      0.91       574
weighted avg       0.92      0.91      0.91       574

[[54  2  0  0  5  0  0  0  0]
 [16 53  0  0  3  0  0  0  0]
 [ 2  6 51  0  0  0  0  0  0]
 [ 0  0  2 57  0  0  0  0  0]
 [ 3  0  0  0 64  1  0  0  0]
 [ 0  0  0  0  4 58  1  0  0]
 [ 1  0  0  0  0  5 58  0  0]
 [ 0  0  0  0  0  0  0 62  0]
 [ 1  0  0  0  0  0  0  0 65]]
```

Figure 1: Confusion Matrix from Titlt- model3

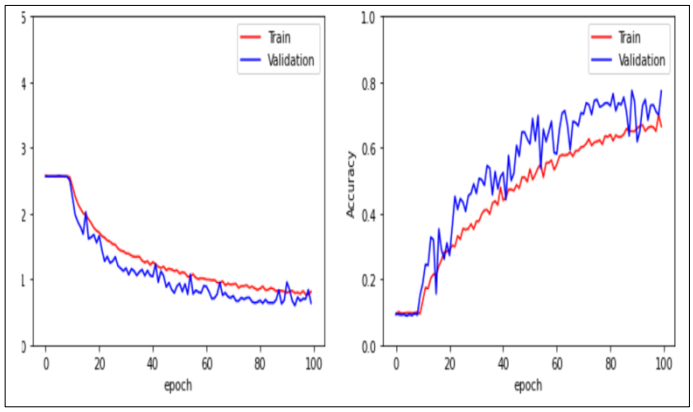


Figure 2: Model's Accuracy Fluctuation

Model Name	Trainable Parameters	Data Augmentation	Train Accuracy : Loss	Validation Accuracy : Loss	Layers	Activation function	Pooling	Optimizer	F1 Score
model1	682,637	Yes	0.5212 : 1.0606	0.5515 : 1.0728	5	Relu	Max	Adam	0.45
model2	686,925	Yes	0.6837 : 0.7984	0.7104 : 0.7137	7	Relu	Max	RMSProp	0.71
model3	3,060,429	Yes	0.6491: 0.8741	0.6895: 0.7185	10	Relu	Max	RMSProp	0.71
model4	660,237	Yes	0.6429 : 0.8855	0.7635 : 0.6543	8	Relu	Max	RMSProp	0.65

Table2: Top Results for Head Pan Estimation

REFERENCES

[1] Patacchiola, M. and Cangelosi, A., 2017. Head pose estimation in the wild using Convolutional Neural Networks and adaptive gradient methods. Pattern Recognition, 71, pp.132-143.

[2] Guo, G., Fu, Y. and Dyer, C., 2008. Head Pose Estimation: Classification or Regression? IEEE, [online] Available at: <https://ieeexplore.ieee.org/document/4761081> [Accessed 6 September 2020].