COSC2779 – Deep Learning

Assignment 1 – Introduction to Deep Convolutional Neural Networks

Rutvi Macwan | s3773570

Table of Contents

Table of Figures

Introduction

Purpose of this assignment was to develop a Deep Convolutional Neural Network model that can identify the head pose of a person from the given image. The development process has taken into consideration all the major phases of model-building cycle, including Data loading and exploration, Data pre-processing, Base model creation with Parameter tuning and Evaluation. Two different models were created to predict both the target values ('tilt' and 'pan') individually.

At the initial phase, the data was explored in order to decide various parameters for the model architecture and evaluation techniques. Also, the base models were created to analyse the performance. Later, hyper parameter tuning was performed over a base model to identify the combination of right parameters achieving the best performance. After building the CNN model with tuned parameters, different evaluation metrics were applied to understand the performance. As the final step, predictions over provided file of unseen data was performed.

Data Exploration

The dataset contains the colored images of size 144 x 192 pixels. Figure1 is an example of one image from the data having tilt angle of -15 and pan angle of 30. Figure2 shows the same image resized to 100 x 100 pixels. Resizing was performed in order to achieve swift training of a model, as small-sized images make training process faster.

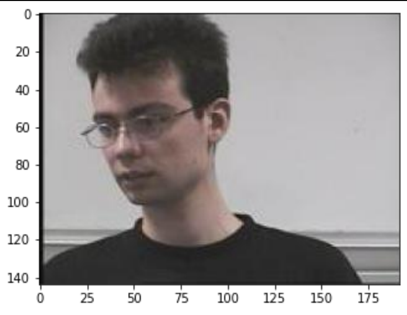
Figure 1: Image size (144 x 192)

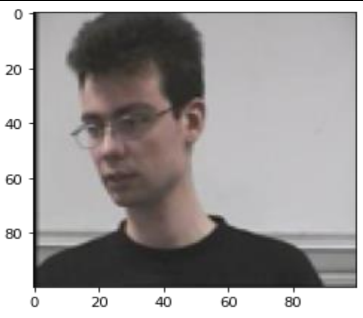
Figure 2: Image size (100 x 100)

Table 1: Example images

To gain better understanding of the data distribution, target class frequency was plotted as shown below for both 'tilt' and 'pan' observations.

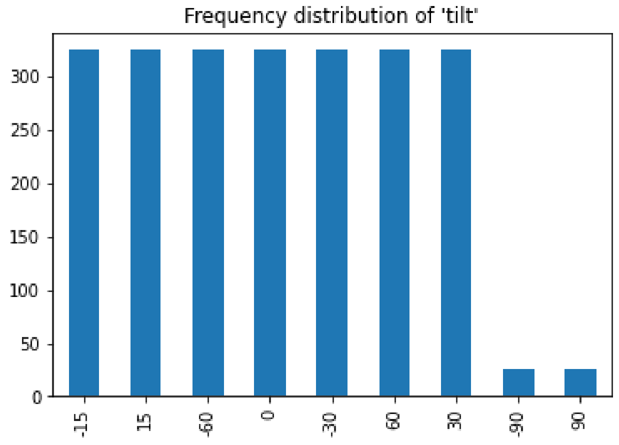

Figure 3: Frequency distribution for 'tilt'

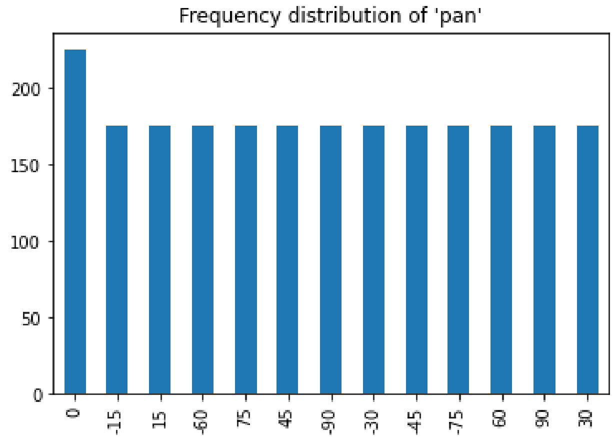

Figure 4: Frequency distribution for 'pan'

Table 2: Target class frequency plots

As observed from the frequency distribution plots, the provided data was imbalanced in nature. To overcome this issue, data augmentation was performed. In addition, at the evaluation phase, apart from just accuracy, other evaluation metrics such as precision and recall were also taken into consideration to provide better judgement of the model performance.

As a part of a data pre-processing, resized images were loaded into the image data generators by converting them into a grayscale format. In addition, a separate batch of augmented data was generated by performing random brightness, padding and random cropping of the images.

Model Building, Tuning and Evaluation

Base Model Creation & Analysis

A single base model was created to predict both 'tilt' and 'pan' individually, only by changing the number of neurons at the output dense layer. The architecture of this model was as shown in figure5. A dense layer with 64 neurons was implemented to convert the 3D feature maps into 1D vectors. The final output layer carried 9 neurons for 'tilt' and 13 for 'pan' prediction.

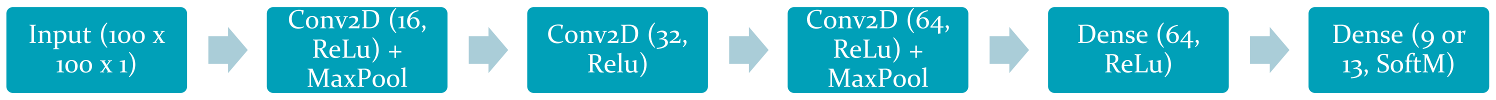

Figure 5: Base Model Architecture

- 'ReLu' activation was applied to convolution layers, as the images were scaled to (0,1) with no negative values and non-saturation gradient. This can help achieve faster convergence of optimizers.
- L2-regularizers and dropout were also introduced to prevent overfitting.
- 'Softmax' activation was applied to output layer, as this belongs to the multi-class classification problem.
- Adam optimizer with learning rate of 0.001 was used to compile the model as it converges faster.
- In addition, to overcome the problem of imbalanced data, the model was shaped to monitor accuracy, precision and recall in order to avoid penalties.

As shown below, for both 'tilt' and 'pan', non-augmented batches showed a huge overfitting while training the model.

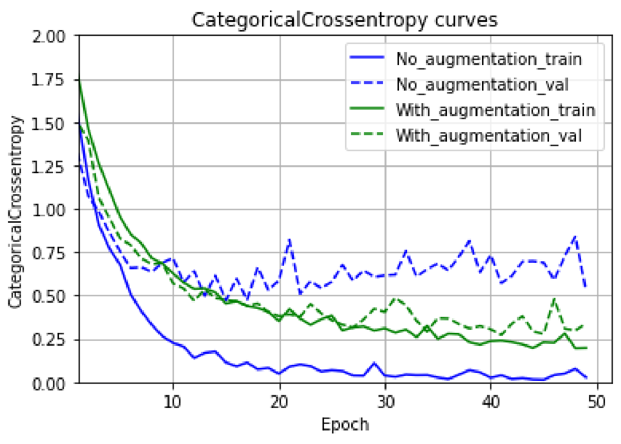

Figure 6: CategoricalCrossentropy for 'tilt'

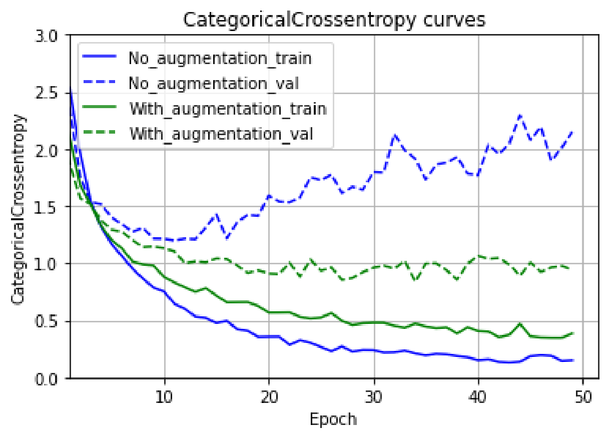

Figure 7: CategoricalCrossentropy for 'pan'

The base models trained with augmented data showed the accuracies of 0.53 and 0.38 respectively, for 'tilt' and 'pan' over the left-out test set. Therefore, it is mandatory to perform hyper-parameter tuning over these models to improve their accuracies as well as to generate more generalized predictions.

Hyper Parameter Tuning

To identify he set of best parameters for both the models separately, RandomizedSearchCV() method has been implemented.

The tilt classifier was achieved by building 3 convolution with max-pooling layers. The pan classifier was achieved by building 4 convolution with max-pooling layers. For both these models dropout of 0.3 and L2-regularizer were introduced at each convolution layer to prevent model from overfitting. The MLP layer was same as in base model carrying one dense layer with 64 neurons and another for output layer.

Training accuracies for both models are as shown below:

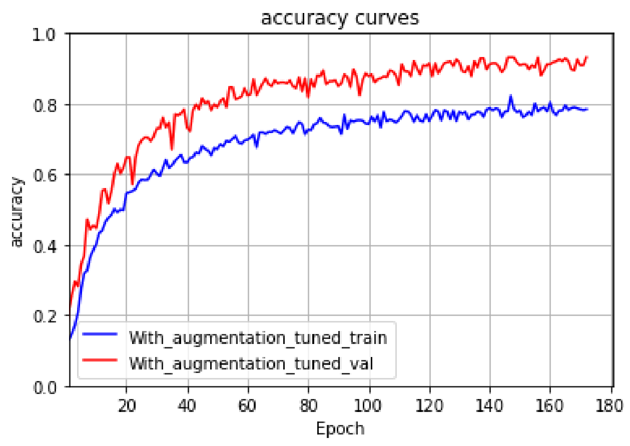

Figure 8: Training accuracy curve for 'tilt'

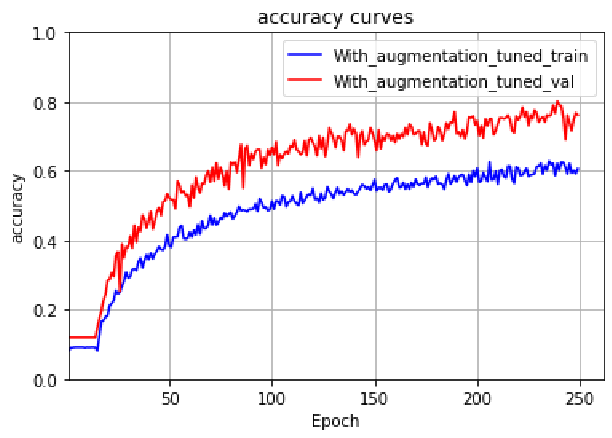

Figure 9: Training accuracy curve for 'pan'

As observed from above accuracy curves, both models represent a good generalization gap indicating the prediction results will be more general in nature.

Model Evaluation

To evaluate the model performance over unseen data, previously holdout test dataset was utilized. The tilt classifier showed an accuracy of 0.63 while the pan classifier received an accuracy of 0.51 over the unseen data. Below table summarizes the evaluation results for both the models.

Classifier	**Accuracy**	**Precision**	**Recall**
Tilt Classifier	0.63	0.63	0.63
Pan Classifier	0.51	0.54	0.51

Table 3: Model evaluation summary

References

1. CV-Tricks.com. 2020. *Tensorflow Tutorial 2: Image Classifier Using Convolutional Neural Network - CV-Tricks.Com*. [online] Available at: <https://cv-tricks.com/tensorflow-tutorial/training-convolutional-neural-network-for-image-classification/> [Accessed 6 September 2020].
2. Stack Abuse. 2020. *Image Recognition In Python With Tensorflow And Keras*. [online] Available at: <https://stackabuse.com/image-recognition-in-python-with-tensorflow-and-keras/> [Accessed 6 September 2020].
3. Brownlee, J., 2020. *How To Grid Search Hyperparameters For Deep Learning Models In Python With Keras*. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/grid-search-hyperparameters-deep-learning-models-python-keras/> [Accessed 6 September 2020].
4. Tennakoon, R., 2020. *Myapps Portal*. [online] Rmit.instructure.com. Available at: <https://rmit.instructure.com/courses/67346/pages/week-4-learning-materials-slash-activities?module_item_id=2521533> [Accessed 6 September 2020].
5. Tennakoon, R., 2020. *Myapps Portal*. [online] Rmit.instructure.com. Available at: <https://rmit.instructure.com/courses/67346/pages/week-5-learning-materials-slash-activities?module_item_id=2521536> [Accessed 6 September 2020].