Assignment 1: Deep Learning COSC 2779

Report : **Head Pose Estimation using Deep Neural Network**

Abstract

Head Pose Estimation has many applications in driver monitoring, attention recognition and multi-view facial analysis systems. It has a great importance in real-world applications such as video surveillance, human machine interaction and security systems. (https://iopscience.iop.org/article/10.1088/1755-1315/170/3/032110/pdf, n.d.) As compared to other traditional machine learning approaches, deep learning approaches shows better performance in this area in case of Accuracy and speed of processing in image classification. In this assignment we developed a deep convolutional neural network (CNN) to identify the head pose of the given image of a person. This head pose is quantified by two values : Tilt and Pan angles. Tilt represents vertical angle of a head and Pan represents horizontal angle of a head. So, the aim of this assignment is to predict this tilt and pan for the provided image of a person. The dataset we have for this experiment is sourced from "Head Pose Image Database published with N. Gourier, D. Hall, J. L. Crowley, "Estimating Face Orientation from Robust Detection of Salient Facial Features", Proceedings of ICPR International Workshop on Visual Observation of Deictic Gestures 2004." (Assignment1 COSC2779 Deep Learning)

Introduction

The main goal of this assignment is to build a deep convolutional neural network to predict tilt and pan angles of unseen image of a person. The data we have 3 main files. 'train_data.csv' file contains training data on which we can train our network and 'test_data.csv' is the dataset of unseen images for which we have to predict the tilt and pan with the help of our built network. All the images for both these files are stored in 'modified_data.zip' folder. If we check the values of the tilt and pan features in 'train-data.csv' file, we can see that, tilt values vary in {-90, -60, -30, -15, 0, 15, 30, 60, 90} and pan values vary in {-90, -75, -60, -45, -30, -15, 0, 15, 30, 45, 60, 75, 90}. That means tilt have 9 unique values while pan have 13 unique values. Positive values of tilt represent upward direction and Negative tilt values represent downward direction. Similarly, positive pan values represent right direction and Negative values represent left direction. This problem can be solved using both machine learning approaches classification and regression. The approach chosen in this assignment is classification machine learning approach where we predict the tilt and pan classes for unseen images. As we know the image data comes under topological structured data, so typical CNN model is considered for this problem.

Basic Considerations and Assumptions

At present, there are many typical architectures present for CNN models which includes LeNet5, AlexNet, ZF Net, VGGNet etc. The typical LeNet5 architecture with modifications is considered as a base model for our problem. The goal of this assignment is to predict the tilt and pan class for unseen image. This problem can be solved using two different image classification methods. One is by building a single multi-task CNN model to classify both pan and tilt. And the other is to predict tilt and pan separately and then combining their respective results (https://www.researchgate.net/publication/322303457_Face_Recognition_Based_on_Convolutional_Neural_Network, 2017). The assignment below is using the second approach to classify tilt and pan separately and then combine their results. The classic approach used for this assignment is to build a convolutional neural network same as typical architecture of LeNet5 and then do some modifications in the architecture as to improve the performance of our model for this specific problem.

As we can see that tilt class have 9 unique class labels while pan have 13 unique class labels. So, for this multiclass classification problem we are considering the 'CategoricalCrossEntropy()' as an error metric to check the errors while training the model. We are considering 'Accuracy()' as our performance metric to check the accuracy in predicting correct labels of images. If we see the class distribution of tilt and pan, we can see that for tilt class, the upper top value (+90) and lowest down value (-90) have very few examples so if the model is performing poorly for tilt we have to consider some different performance metric to solve this class imbalance problem. For pan all the values are almost equally distributed.

To generalize the performance of the trained model, we have to split the data in file 'train_data.csv' into training and validation dataset. As we know there are 2325 total records in this file. We can't split it into a half portion for training and half for validation as the number of records are comparatively low. And Neural network require considerable data to train the model. The more the data, better the model is trained to predict unseen image. Hence, we are splitting this data into 80:20 split in which 80% of total records are kept in training dataset while remaining 20% data is kept as a validation data. The basic 'train_test_split()' function from 'sklearn' library is used for this purpose. So, we are training our model on 1860 records and check if it is generalizing on remaining 465 records. We import the image data with the help of ImageDataGenerator for both training and validation datasets. For the first run we didn't use any augmentation techniques while importing images and we fixed the batch size as 32.

Methodology and Model Architecture

By taking inspiration from standard classic architecture of LeNet5 model, which is use for predicting handwritten symbols, we first build our model with some modifications. As we have two classification problems for tilt and pan, respectively. We are trying to build a model which shares the common feature extraction power of a neural network for both tilt and pan prediction. That means, we are using common convolutional layer structure for extracting features which are used for predicting tilt and pan both.
We tried with the standard LeNet5 architecture and don't get enough performance hence after some modifications we came out with the modified CNN architecture to solve this problem.
So, the proposed final model consists of 3 Convolutional Layers and single MLP layer for tilt prediction and two MLP layers for pan prediction. Convolutional layers are followed with the down sampling layers and MLP layers are followed with dropout layers to prevent the model from overfitting. We still faced some overfitting in the model training, so we add some kernel regularizer to simplify the model complexity by penalizing each layer parameters. We are applying class L2 regularizers with values 0.0001 for first conv layer and 0.001 for others. we came up with this value by tuning the model with all different values like (0.01, 0.001, 0.0001, 0.005) and checking the performance of model each time.

Tilt model architecture:

- Conv layer block 1 : The first convolutional layer took input as tensors [32,32,32,1]. This layer has kernel size 63 and stride size of (3,3) and L2 Kernel_regularizer of value 0.0001. This layer is followed by activation layer of 'Relu' which is non-linear activation function and a down sampling layer of stride (2,2).
- Conv layer block 2 : This block consists of a convolutional layer with 128 kernel filters and stride of same size (3,3) with L2 regularizer of 0.001. This layer is followed by the activation layer with same activation function 'Relu'.
- Conv layer block 3 : This block consists of a convolutional layer with 256 filter size and stride size of (3,3) with L2 regularizer of 0.001. This layer is followed by Activation layer and down sampling layer same as that in block 1.

These convolutional blocks are connected to MLP layer block that consists of dense layers as well as the output layer.

- MLP layer 1 : this block consists of a flatten layer to map 3d features to 1d vector and then pass it to the Dense layer of size 128 which is fully connected layer which is followed by a dropout layer of rate 0.5 which helps to prevent model from overfitting. And finally, we have an output layer for predicting 9 class labels of tilt which are predicted using a 'softmax' activation function

Pan model architecture:

- This model also shares same convolution layer architecture and just differs with the MLP layers. It consists of 2 Dense layers with 'Relu' activations. First dense layer is of size 128 filters fully connected layer while second dense layer is of size 64 . Both the layers are followed by the dropout layers of rate 0.5. The final output layer is of size 13 to predict pan class labels using 'softmax' activation function.

While compiling the model, Optimizer Adam(), CategoricalCrossEntropy() loss function and Accuracy () performance metric is used for both the models.

Hyper Parameter setting and Tuning

The next step is to check the performance of the model using learning curves produced during training and validation of the data. These curves can be observed with respect to the error metric which is CategoricalCrossEntropy as well as performance metric which is Accuracy().
At first, we tried to check the performance of a model which is replica of traditional LeNet5 model (with Relu and softmax activations). We get to know that the models are highly overfitting that mean they are not generalizing the data well. As a solution we do have 3 options to reduce the overfitting of the models :

1. Increase the training data : we tried increasing the size of training data by changing validation split ratio from 80:20 to 85:15 and 90:10 but the result is not improved marginally.
2. Use Image Augmentation : image Augmentation helps to distort image which may create better results while extracting the features from the images. ImageDataGenerator comes with default options for image augmentations. These options involved rescaling, flipping, rotating, shifting height and width, shearing , zooming the images etc. But as our class labels are associated with direction or position of head, flipping the image, or rotating or changing orientations may result into wrong predictions as labels associated with those images get out of sync with the actual directions of image. Hence, we are unable to do much augmentation techniques. We tried changing contrast and brightness as well as shifting height and width of images but as a result, I got to know that after certain epoch, models are not learning anything new from the images which increases the CategoricalCrossEntropy values after certain epoch resulting in overfitting again.
3. Tuning values for dropout and regularizations: The next method is to add dropout layers and regularizations into the model which may help in decreasing the overfitting. I tried different rate values of dropout rates (0.3, 0.4, 0.5) and for L2 regularizers (0.01, 0.001, 0.0001) manually and finally came up with dropout rate of 0.5 and L2 regularization rate of (0.0001 for Conv layer 1 and 0.001 for layers 2 and 3). In that way, the overfitting problem of those models reduced by a marginal amount.

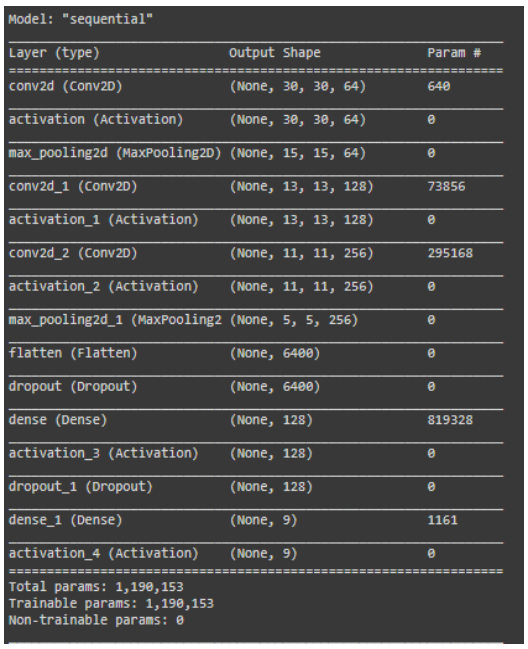
The final models after hyper parameter tuning and reducing the overfitting gave the accuracy score of around 87% on tilt validation data and around 70% on pan validation data which I think is better for these models.

Limitations and Improvements for the real-world implementations.

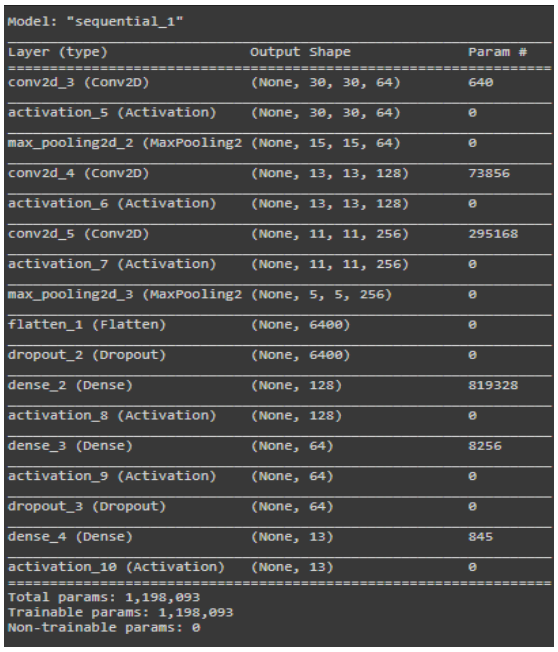
The main limitation for this model is we have comparatively less data to check and improve the performance of any model and try out any further tuning. We can still consider more complex architectures like ResNet or VGGNet as our base model for this problem which may solve this problem with better performance. Changing few parameters like optimizer functions from Adam to may be 'RMSProp or SGD with momentum' may change the performance of these models. Using different techniques like K-fold cross validation or more image augmentation techniques may help to improve the results but that too require more training data. This problem can be solved as a regression problem with MAE or MSE as a performance metric. We can solve this also using different architectures as a baseline model.

Model performance and Summary

1. Tilt and Pan model summary

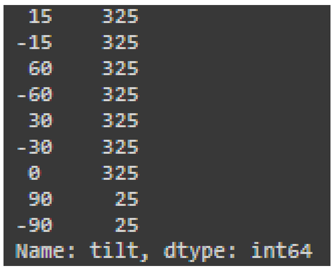


```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 30, 30, 64)        640
_________________________________________________________________
activation (Activation)      (None, 30, 30, 64)        0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 15, 15, 64)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 13, 13, 128)       73856
_________________________________________________________________
activation_1 (Activation)    (None, 13, 13, 128)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 11, 11, 256)       295168
_________________________________________________________________
activation_2 (Activation)    (None, 11, 11, 256)       0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 5, 5, 256)         0
_________________________________________________________________
flatten (Flatten)            (None, 6400)              0
_________________________________________________________________
dropout (Dropout)            (None, 6400)              0
_________________________________________________________________
dense (Dense)                (None, 128)               819328
_________________________________________________________________
activation_3 (Activation)    (None, 128)               0
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 9)                 1161
_________________________________________________________________
activation_4 (Activation)    (None, 9)                 0
=================================================================
Total params: 1,190,153
Trainable params: 1,190,153
Non-trainable params: 0
_________________________________________________________________
```

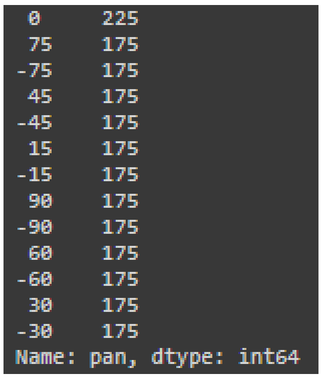


```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 30, 30, 64)        640
_________________________________________________________________
activation_5 (Activation)    (None, 30, 30, 64)        0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 15, 15, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 13, 13, 128)       73856
_________________________________________________________________
activation_6 (Activation)    (None, 13, 13, 128)       0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 11, 11, 256)       295168
_________________________________________________________________
activation_7 (Activation)    (None, 11, 11, 256)       0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 5, 5, 256)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 6400)              0
_________________________________________________________________
dropout_2 (Dropout)          (None, 6400)              0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               819328
_________________________________________________________________
activation_8 (Activation)    (None, 128)               0
_________________________________________________________________
dense_3 (Dense)              (None, 64)                8256
_________________________________________________________________
activation_9 (Activation)    (None, 64)                0
_________________________________________________________________
dropout_3 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_4 (Dense)              (None, 13)                845
_________________________________________________________________
activation_10 (Activation)   (None, 13)                0
=================================================================
Total params: 1,198,093
Trainable params: 1,198,093
Non-trainable params: 0
```

2. Class label distribution for tilt and pan classes.

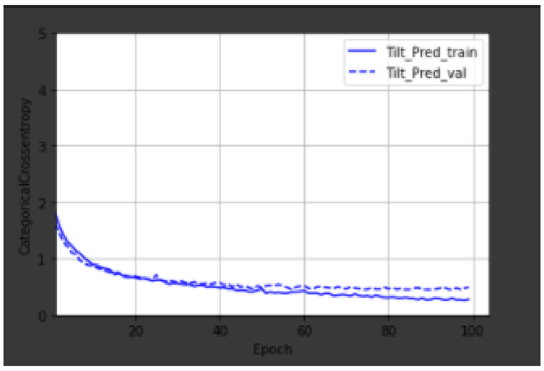


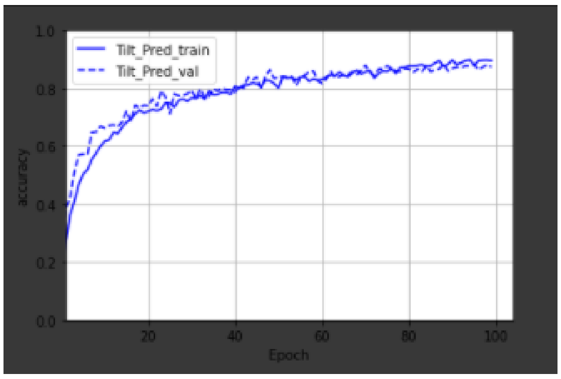
```
 15     325
-15     325
 60     325
-60     325
 30     325
-30     325
 0      325
 90      25
-90      25
Name: tilt, dtype: int64
```



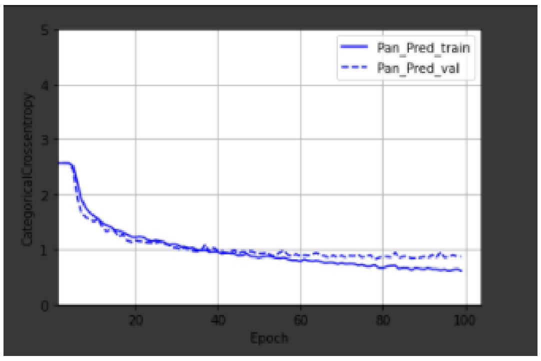
```
 0      225
 75     175
-75     175
 45     175
-45     175
 15     175
-15     175
 90     175
-90     175
 60     175
-60     175
 30     175
-30     175
Name: pan, dtype: int64
```

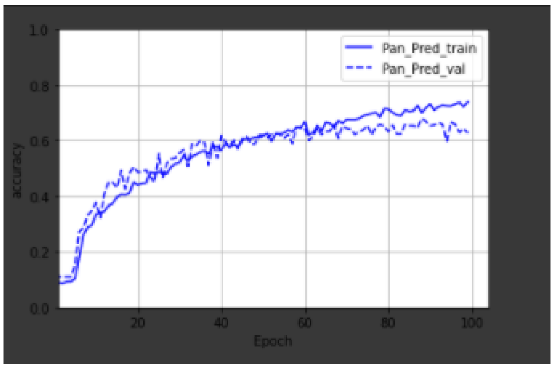
3. Learning Curves for Tilt prediction.





4. Learning Curves for Pan prediction.





References

- Assignment1 COSC2779 Deep Learning. (n.d.).
- https://iopscience.iop.org/article/10.1088/1755-1315/170/3/032110/pdf. (n.d.). Retrieved from https://iopscience.iop.org/article/10.1088/1755-1315/170/3/032110/pdf.
- https://www.researchgate.net/publication/322303457_Face_Recognition_Based_on_Convolutional_Neural_Network. (2017, 11). Retrieved from https://www.researchgate.net/publication/322303457_Face_Recognition_Based_on_Convolutional_Neural_Network.