TECHNISCHE UNIVERSITÄT
CHEMNITZ

Driver head orientation detection using Transfer Learning

Research Project

Submitted in Fulfilment of the
Requirements for the Academic Degree
M.Sc.

Dept. of Computer Science
Chair of Computer Graphics and Visualization

Submitted by: Adarsh Mallandur
Student ID: 457084
Date: 19.09.2018

Supervising tutor: Stephen Helmert

Abstract

Estimating the head pose of the driver is a crucial as well as one of the difficult problems in the task of driver attention monitoring systems, given the problems of occlusions, illumination changes and the different poses of the driver. It is useful especially in highly autonomous vehicles where the handoff time to human driver needs to be determined. Traditionally head pose is estimated by detecting facial landmarks and solving the problem of 2D to 3D correspondence. This is a fragile method since it depends entirely on the estimation of landmarks which is difficult especially when there are occlusions and variations in illuminations. In this research project, a more robust way of estimating the head pose is done by training a convolutional neural network through the method of transfer learning. In our case, a network pre-trained on ImageNet dataset is used as feature extractor and further trained on DrivFace, a dataset containing images sequences of subjects while driving in real scenarios. This network classifies the RGB images into three classes: looking left, looking front and looking right. Results show that the method of transfer learning overcomes a DeepaGaze, a head pose estimation framework trained on in-the-wild datasets.

Keywords: Convolutional neural networks, transfer learning, head pose estimation, deep learning

Contents

List of Figures

List of Tables

List of Abbreviations

ADAS Advanced Driver Assistance Systems

CNN Convolutional Neural Networks

IVIS In-Vehicle Information Systems

AFLW Annotated Facial Landmarks in the Wild

ReLU Rectified Linear Unit

ANN Artificial Neural Network

SGD Stochastic Gradient Descent

LSTM Long Short Term Memory

1 Introduction

Driver attention monitoring in real time is a crucial task of advanced driver assistance systems. The driver drowsiness and distraction from the road increases the risk of accident significantly [**?**] [**?**]. The use of in-vehicle information systems (IVISs) such as cell phones, GPS navigation systems, and satellite radios further adds to the risk. [**?**] [**?**]Hence it is necessary to design driver attention montoring systems. One way this could be helpful is to capture the driver distractions and alert the driver in case of dangerous situations. There has been lot of work done in developing driver attention monitoring in the context of Advanced Driver Assistance Systems(ADAS) [**?**] However, there is no system which generalises the driver head pose estimation across various drivers, cars and perspectives. Convolutional Neural Networks (CNNs) have been successful in various image classification and object detection tasks [**?**] In this project, the feasibility of CNNs in driver monitoring systems is analysed.

1.1 Problem statement

The aim of this project is to use CNNs to classify the driver gaze zones. However training CNNs from scratch is often time consuming and difficult. It requires learning of millions of parameters and a large number of annotated images for training. Training CNNs for smaller datasets often leads to overfitting. Transfer learning solves this problem by using pre trained weights learnt by CNNs on large scale datasets. [**?**]n this project, two different networks trained on the ImageNet [**?**]ataset is fine tuned to classify driver gaze zones in the DrivFace dataset [**?**] The DrivFace database [**?**]ontains images sequences of subjects while driving in real scenarios. It is composed of 606 samples of 640*480 pixels each, acquired over different days from 4 drivers (2 women and 2 men) with several facial features like glasses and beard. The performance of the transfer learning network on DrivFace dataset is compared against Deepgaze [**?**, **?**] a CNN trained on AFLW dataset [**?**]

1.2 Structure of the project

The main focus of this project work can be summarised as

1. To study the application of CNNs in head pose estimation of driver inside a car and to verify the robustness of the algorithm against occlusions, illumination changes and different poses of the driver.

2. To train and validate the CNN using the approach of transfer learning on the DrivFace [**?**]ataset with real time dashboard images of drivers inside a car.

3. To compare the performance of our CNN against Deepgaze [**?**, **?**] a CNN trained on AFLW dataset [**?**]

2 Related work

Research study	Objective	Classifier
Driver Gaze Zone Estimation using Convolutional Neural Networks: A General Framework and Ablative Analysis []	Gaze zone estimation using head pose dynamics	Random Forest
On Driver Gaze Estimation: Explorations and Fusion of Geometric and Data Driven Approaches []	Gaze zone estimation using fusion of geometric and learning based method	SVM
Driver Gaze Region Estimation Without Using Eye Movement []	Gaze zone estimation using spatial configurations of facial landmarks	Random Forest
Head pose estimation in the wild using Convolutional Neural Networks and adaptive gradient methods []	Head pose estimation as yaw, pitch and roll angles	CNN

Table 2.1: Related research work on driver head pose estimation

A summary of recent studies on head pose estimation is shown in table 2.1.

3 Background

In this chapter we discuss the theoretical concepts of neural networks, computer vision and deep learning required for understanding the implementation of the project. First, the details of artificial intelligence, machine learning and deep learning are discussed. Finally, the application of deep learning to computer vision is examined in detail.

3.1 Machine learning

Machine learning is used to solve a wide variety of problems in computer vision, speech processing, statistics etc,. It is a subset of artificial intelligence where the computer is shown data to learn patterns and solve the given problem without being explicitly programmed to do so [**?**] Computers have been successful in several tasks that were considered impossible by humans. However they fail to do simple tasks such as recognising objects or speech. Hence, machine learning algorithms were designed to infer the result from features of the input. In most of the cases these features needed to be hand-crafted by an expert. The algorithms work better when given better features. As the availability of data and the computational power has increased in the recent years, machine learning has become more and more practical and successful in various application domains.

3.1.1 Types

Machine learning algorithms can be broadly classified into supervised learning, unsupervised learning and reinforcement learning. In *supervised learning* the algorithm is trained using labelled training data and then shown with test data to predict the results. For example, in object detection problems a training data set with annotated images is shown to the model through which the model learns the representations and the mapping from input to output. After training the model is able to predict the labels of unseen images. Supervised learning can be further classified into classification and regression. In *classification*, the algorithm tries to predict the discrete output class the input belongs to, whereas in *regression* the output is a continuous value. *Unsupervised learning* is the process of building the model without the help of labelled data. It can also be considered as a problem of clustering a given dataset [**?**] The machine learning algorithm needs to separate the set of data points into groups in the best way possible. *Reinforcement learning* uses a system of rewards and punishments. The algorithm learns by interacting with its environment. The

algorithm receives rewards for performing correctly and penalties for performing incorrectly. The algorithm learns without the intervention of human by maximising the rewards and minimising the penalties.

3.1.2 Features

Features are the attributes in the data which are most relevant to the machine learning problem. Selection of features which enhance the performance of the model is a crucial step in machine learning. Feature selection methods are used to remove redundant and irrelevant attributes from data that either do not contribute to the accuracy of the model or may even decrease the accuracy of the model. There are different feature selection techniques such as

3.2 The perceptron

An artificial neuron or perceptron is modelled similar to the neurons in the human brain. It takes several inputs and performs a weighted summation to produce an output. The weight of the perceptron is determined during the training stage. The training is performed using a method called *gradient descent* which will be explained in later sections.

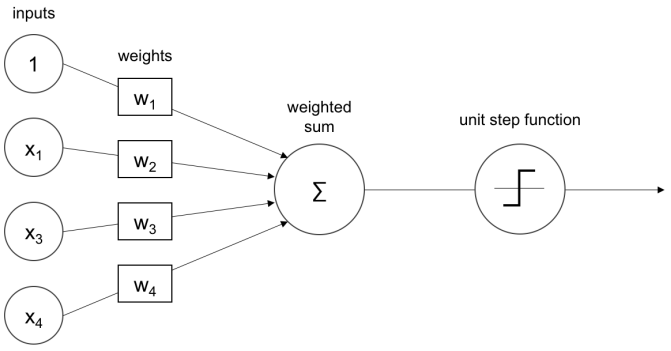
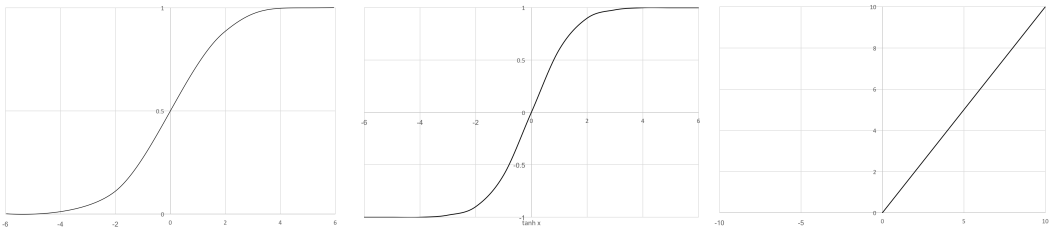


Figure 3.1: The perceptron

The weighted sum is passed through a non-linear function called *activation function* to capture the non-linearities in the input. An *activation function* decides whether a neuron should fire or not. There are various activation functions as shown in 3.2. Sigmoid is useful for converting any value to probabilities and can be used for binary classification. The tanh maps input to a value in the range of -1 to 1 and are more stable than sigmoid. The ReLU maps input x to max (0, x) and works well for a large range of problems.



(a) Sigmoid function (b) Tanh function (c) RELU function

Figure 3.2: Some common activation functions

3.3 Artificial Neural Networks

Artificial neural network is a collection of perceptrons and activation functions. The perceptrons are connected to each other to form the hidden layers as shown in 5.1(b). The training process determines the values of these weights and biases. The training is started by initialising the weights and biases to random values. The error is computed by taking the difference of the actual output to the ground truth. Based on the loss, the weights and biases are tuned in steps. The training is stopped when the error can be minimised anymore and during this point the model is said to learn the features in the input.

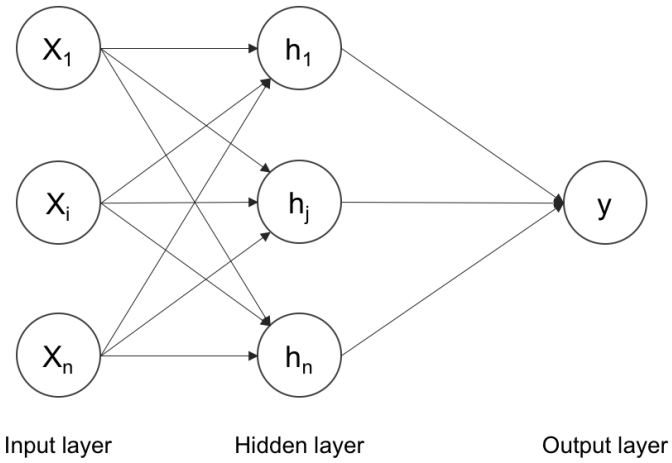


Figure 3.3: Artificial Neural Network

3.4 Training the neural network

3.4.1 Backpropagation

One of the commonly used algorithm for training the neural networks is the back-propagation algorithm. The weights are updated from the back based on the error

that is calculated. The weights are updated using the gradient descent algorithm which is explained in the following section.

3.4.2 Gradient Descent

Often in machine learning, finding the best model for a certain situation means to minimise the error of the model or maximise the likelihood of the data. Hence it can be seen as finding the solution for an optimisation problem. Gradient descent is one of the optimisation problems which is widely used in the algorithms of machine learning and deep learning.

Idea

Suppose we have some function $\mathbf{f(x)}$ that takes as input a vector of real numbers and outputs a single real number. One such simple function is given in equation 3.1.

$$f(x) = x^2 \in R \tag{3.1}$$

The *gradient* of $\mathbf{f(x)}$ gives the input direction in which the function $\mathbf{f(x)}$ most quickly increases. One approach to maximising a function is to pick a random starting point, compute the gradient, take a small step in the direction of the gradient, and repeat with the new starting point. Similarly, we can try to minimise a function by taking small steps in the opposite direction, as shown in 3.4.

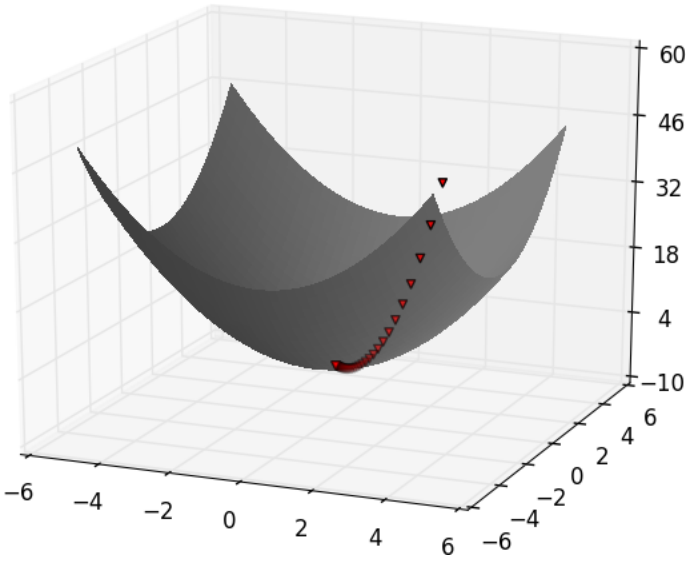


Figure 3.4: Finding a minimum using gradient descent

This procedure will work very well if the function has a unique global minimum. If a function has multiple (local) minima, this procedure might find the wrong one of them, in which case we might re-run the procedure from a variety of starting

points. If a function has no minimum, then it is possible the procedure might go on forever.

Estimating the gradient

If f is a function of one variable, its derivative at a point x measures how f(x) changes when we make a very small change to x. The derivative is the slope of the tangent line at (x, f(x)). The gradient can be estimated as given in 3.2 by considering h to be very small.

$$Gradient = \frac{f(x+h) - f(x)}{h} \tag{3.2}$$

If f is a function of multiple variables then it has multiple partial derivatives each indicating the change in f when we make small changes to one of the input variables by holding the other variables constant. The learning rate η determines the size of the steps we take to reach a (local) minimum. There are three variants of gradient descent as explained in the following sections.

3.5 Types of gradient descents

3.5.1 Batch gradient descent

Batch gradient descent or vanilla gradient descent calculates the gradient of the cost function w.r.t the parameters θ for the entire training dataset.

$$\theta = \theta - \eta \cdot \Delta_\theta J(\theta) \tag{3.3}$$

Batch gradient descent is slow since we need to calculate the gradients for the whole dataset to make just one update of the parameters. Batch gradient descent is guaranteed to converge to the global minimum for convex error surfaces and to a local minimum for non-convex surfaces.

3.5.2 Stochastic gradient descent

Stochastic gradient descent (SGD) performs parameter update for each training sample. Batch gradient descent performs redundant computations for large gradients since it recomputes the gradients for similar examples before each update. SGD does not have this problem since it performs one update for each training data and is much faster. These frequent updates cause the objective function to fluctuate which may lead to overshooting. However, by setting the optimal learning rate, SGD can show same behaviour as the batch gradient descent.

$$\theta = \theta - \eta \cdot \Delta_\theta J(\theta; x^{(i)}; y^{(i)}) \tag{3.4}$$

3.5.3 Mini-batch gradient descent

Mini-batch gradient descent takes the best of SGD and batch gradient descent. It updates the parameters for every mini-batch of n training samples. It reduces the variance of the parameter updates and hence more stable convergence. It also makes computations more efficient.

$$\theta = \theta - \eta \cdot \Delta_\theta J(\theta; x^{(i:i+n)}; y^{(i:i+n)}) \tag{3.5}$$

Challenges of mini-batch gradient descent

- Finding the optimum learning rate is difficult. A learning rate which is too small will take lots of time to converge and a learning rate too fast might overshoot and oscillate around the minimum.
- Neural networks usually have highly non-convex error functions with saddle points. These are the points where one dimension slopes up and another slopes down. The SGD finds these points very hard to escape since the gradient is close to zero in all dimensions.

3.6 Convolutional Neural Networks

Convolutional neural networks are very similar to the perceptron. They are made up of neurons with weights and biases. Each neuron performs a product of the input with the weights and adds a non-linearity at the end. They have loss functions and a fully connected layers at the end. However, in convolutional neural networks the inputs are exclusively images which makes the forward propagation easier to implement and also reduces the number of parameters in the network.

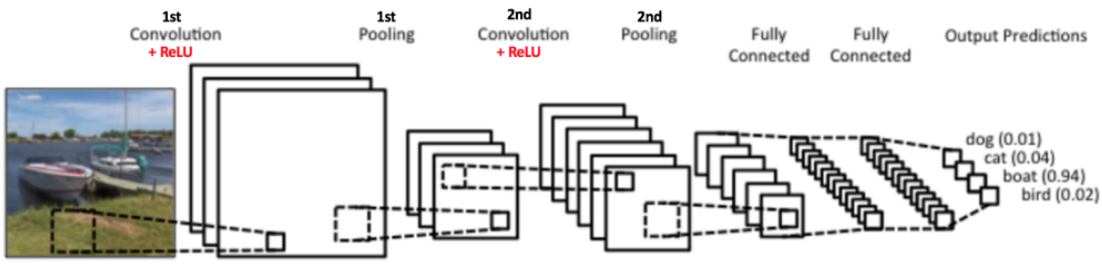


Figure 3.5: Convolutional layers

The regular neural networks do not scale to full images since they have fully connected layers at the input. For example, an image of 240*480*3 would lead to neurons that have 240*480*3 weights. This would lead to a huge number of parameters and hence to overfitting. In a convolutional neural network, the neurons in a layer will only be connected to a small region of the layer before it, instead of a

fully connected network. A convNet is made up of layers and each layer transforms an input 3D volume to an output 3D volume with some differentiable function. The convolution and pooling layers act as feature extractors from the input image and the fully connected layer acts as a classifier. As shown in 5.2 the target probability is 1 for Boat class and 0 for other classes.

4 Transfer learning

Deep learning networks are *data dependent*. The scale of the model and the size of the data is linearly dependent. Many domains face the problem of insufficient training data. Often the model faces situations where it has not seen the data before and deos not know how to deal with it. In all of such situations the current state-of-the-art models either show degraded performance or even break down. Transfer learning can help us deal with such scenarios.

Tranfer learning allows us to use the already existing labeled data of some related task or domain. The name transfer learning is derived from the fact that we try to transfer as much knowledge as we can from the source to our target task or domain.

4.1 Definition

Transfer learning involves a domain and a task. A domain $\mathbf{D}$ consists of a feature space $\mathbf{X}$ and a marginal probability distribution $\mathbf{P(X)}$ over the feature space. Given a domain, $\mathbf{D = (X, P(X))}$, a task $\mathbf{T}$ consists of a label space $\mathbf{Y}$ and a conditional probability distribution $\mathbf{P(Y|X)}$ that is learnt from the training data. Given a source domain $\mathbf{Ds}$ and a corresponding source task $\mathbf{Ts}$ as well as a target domain $\mathbf{Dt}$ and a target task $\mathbf{Tt}$, the objective of transfer learning is to enable to learn the target probability distribution $\mathbf{P(Yt|Xt)}$ in $\mathbf{Dt}$ with the information gained from $\mathbf{Ds}$ and $\mathbf{Ts}$ where $\mathbf{Ds} \neq \mathbf{Ts}$ or $\mathbf{Ts} \neq \mathbf{Tt}$.

Given source and target domains $\mathbf{Ds}$ and $\mathbf{Dt}$ where $\mathbf{D = (X, P(X))}$ and source and target tasks $\mathbf{ts}$ and $\mathbf{Tt}$ where $\mathbf{T = (Y, P(Y|X))}$ there can be four transfer learning scenarios.

1. $\mathbf{Xs} \neq \mathbf{Xt}$. The feature space of source and target domain are different.
2. $\mathbf{P(Xs)} \neq \mathbf{P(Xt)}$. Marginal probability distributions of source and target domain are different.
3. $\mathbf{Ys} \neq \mathbf{Yt}$. The label spaces between the two tasks are different.
4. $\mathbf{P(Ys|Xs)} \neq \mathbf{P(Yt|Xt)}$. The conditional probability distributions of the source and target tasks are different.

4.2 Transfer learning methods

Transfer learning has been investigated in the past []. With the approach of deep learning there are new methods of transfer learning available.

4.2.1 Using pre-trained CNN features

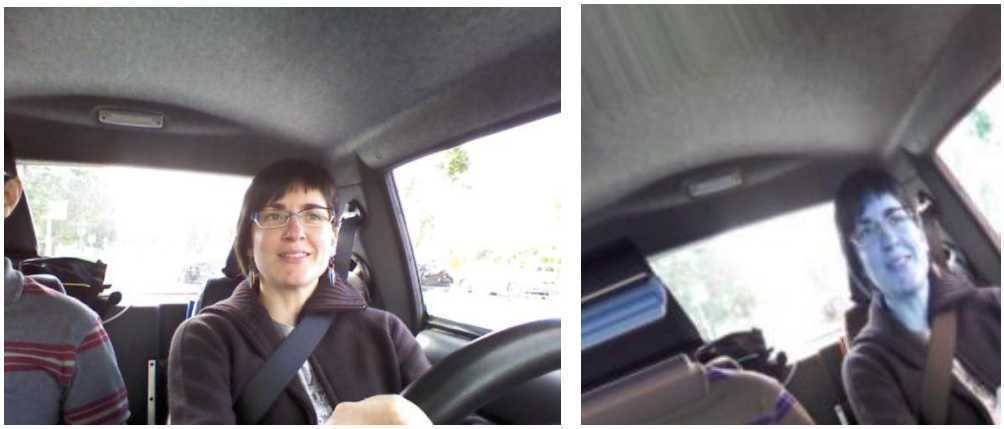
This is the most common way of applying transfer learning for a given problem. It is also the method which has been used in this project. The lower layers of the convolutional neural networks capture the low-level image features such as edges, corners etc., while the higher convolutional layers capture more and more complex details such as body parts, faces etc. The final fully connected layers are assumed to capture information that is relevant to solving the respective task for example the classes in case of classification task. Thus we can simply use the features of a state-of-the-art CNN pre-trained on ImageNet [] and train a new model on these extracted features. In practice, we keep the earlier layers of the CNN fixed or tune them with very low learning rate so as to not unlearn these features. This simple approach has been shown to achieve great results on various computer vision tasks []. Therefore, the ImageNet task seems to be a good base for general computer vision problems.

4.2.2 By developing a model

In this method, a related predictive modelling problem with an abundance of data is selected and a model is developed based on this task. Then the model is either tuned or reused as is to a different task of interest.

5 Method

The aim of the project is to classify the images into three categories based on the head orientation. For this task, the DrivFace dataset [] is used which contains images of drivers while driving in real scenarios. We will first consider the data preprocessing, the model, the training and then the testing of the model.



(a) Original image (b) Augumented image

Figure 5.1: Image augumentation

5.1 DrivFace Dataset

The DrivFace database /cite contains images sequences of subjects while driving in real scenarios. It is composed of 606 samples of 640480 pixels each, acquired over different days from 4 drivers (2 women and 2 men) with several facial features like glasses and beard. The ground truth contains the annotation of the face bounding box and the facial key points (eyes, nose and mouth). A set of labels assigning each image into 3 possible gaze direction classes are given.

1. The first class is the looking-right class and contains the head angles between -45°and -30°.
2. The second one is the frontal class and contains the head angles between -15°and 15°.
3. The last one is the looking-left class and contains the head angles between 30°and 45°.

5.2 Preprocess the data

The data is first augumented using the keras class /textitImageDataGenerator. The parameters of data augumentation are passed while creating an object of the class. Here we set the horizontal flip to false since we need to classify the images as looking left and looking right. We give some rotation ranges, width shift range, height shift range, shear range and zoom range. This generates different augumentated images of our dataset. The images before and after the data augumentation can be seen as below 5.1(b). Since the data set already contains the coordinates of the face annotations we use this to extract face from the images. We then split the dataset into train and test with a test data of 20

5.3 Model

We have tested the dataset on four models

1. DeepGaze CNN model : The model is derived from the DeepGaze model []. There are 9 layers and 4,566,147 parameters as shown in 5.2

2. DeepGaze CNN model with data augumentation: The model is same as the first one but the data is augumented before training.

3. Transfer learning with VGGNet: The model uses the first layers as the standard VGGNet model []. The last layer is removed and replaced with a dense network with 3 neurons each for the 3 outputs.

4. Transfer learning with ResNet: The model uses the first layers as the standard ResNet50 model []. The last layer is removed and replaced with a dense network same as the third model.

5.4 Results

5.4.1 DeepGaze Model

The training and validation accuracy results of the DeepGaze model against the DrivFace dataset is as shown in the figure 5.3(a). We can see clearly that the model is overfitting the data since the dataset is too small. Hence the training accuracy soon reaches 100 percent while the validation accuracy keeps dropping. This can be resolved by increasing the size of the dataset using data augumentation. We have used the ImageDataGenerator of Keras [] to augument the images before passing them to training. This is done by adding some rotation, translation, shearing, zoom to the images. The horizontal flip is set to false since this modifies the head orientation information in the image. Also, to avoid overfitting dropout regularisation []

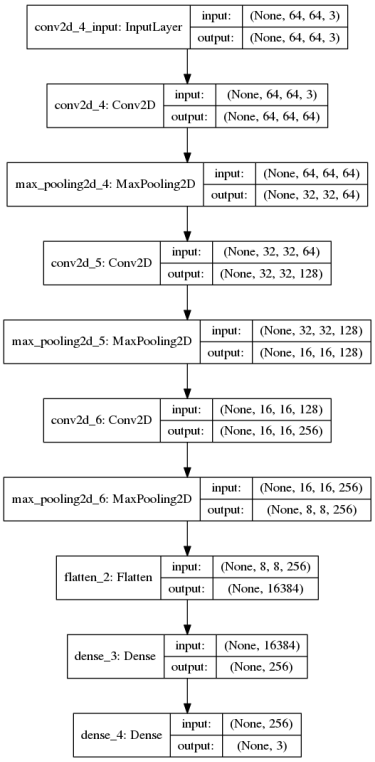
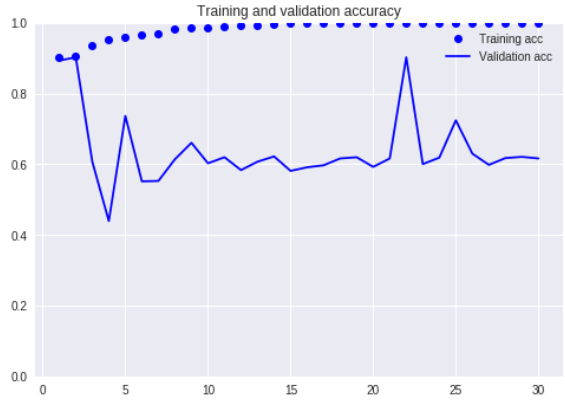


Figure 5.2: DeepGaze CNN Model

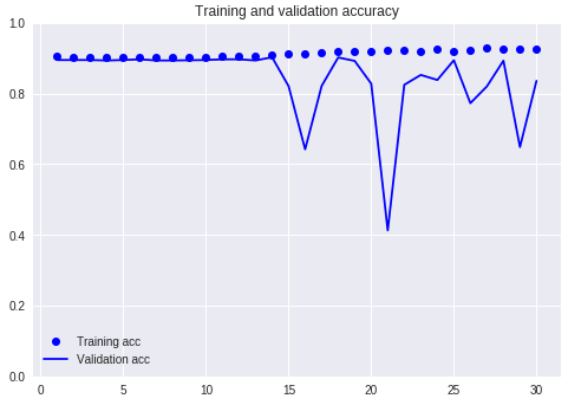
technique is added. The results of the training and validation are as shown in the figure 5.3(b). The validation accuracy is 84% with data augumentation which was 62% without augumentation.

5.4.2 VGGNet and ResNet Model - Transfer learning

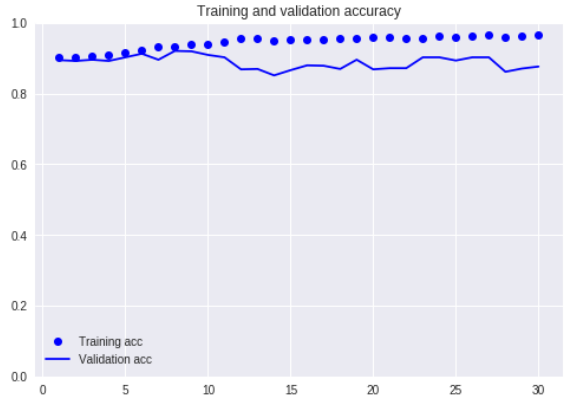
The training and validation accuracy resultsof the VGGNet and ResNet models are as shown in the figure 5.3(c) and figure 5.3(d). Here, we can see that the problem of overfitting is solved. With the method of transfer learning using VGGNet model, the training and validation curves closely follow each other. But the performance is better with the Resnet model as seen in the figure 5.3(d). Also the validation accuracy is 87% with VGGNet and 89% with ResNet model. Thus the accuracy problem of the model with less training data is solved by using transfer learning.



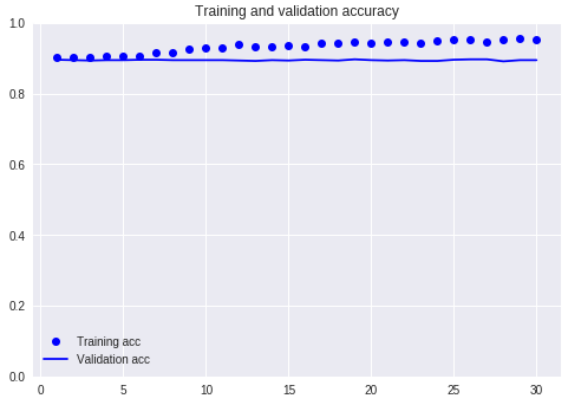
(a) DeepGaze Model without data augumentation



(b) DeepGaze Model with data augumentation



(c) VGG16 Model



(d) ResNet50

Figure 5.3: Training vs validation accuracy of the four models

6 Concluding Remarks

Head pose estimation is a crucial task in autonomous vehicles since alerting the driver at correct time can prevent several road accidents. It will also help autonomous vehicles to determine the handoff time to the human driver. In this project, various works of head pose estimation were studied. There has been a lot of progress made in head pose estimation systems but not towards systems which can generalize to different drivers, cars, perspective and scale. This model can be further improved by adding several driver gaze zones rather than just three. It was shown that using small datasets causes overfitting and also poor performance. This can be solved by using the approach of transfer learning. We demonstrated that using the state of the art models such as ResNet50, we can solve the overfitting problem and can also achieve significant accuracy. Future work in this direction can use utilize Long Short Term Memory(LSTM) **??** which can use information in successive frames.

7 Referencing