Capstone Project

Machine learning Engineer Nanodegree

Mohammad Akram Hossain

December 20, 2016

I. Definition

Introduction

Image classification or recognition can be generalized as the process of assigning label to an image from a fixed set of label categories. Image classification is a data-driven approach and the workflow pipeline of this approach can be described as following:

- Input: For image classification task, input set consists a set of N images where each image represented by array of pixels and labeled with one of K different class. Input data then randomly split in to train, test and validation dataset. The classifier trained using train data, validated with validation data, and performance evaluated by test data.
- Learning Algorithm: The classifier is based on learning algorithm. The learning algorithm is generally selected based on image data size and complexity of the features. A good learning algorithm must be invariant to the Viewpoint variation, scale variation, deformation, background conditions, class variation, and illuminating conditions.
- Evaluation: the trained classifier label prediction performances are evaluated by applying it on the new set of images i.e. test set and evaluation set. The performance of is measured with appropriate accuracy metrics.

Recently, image classification has drawing a lot interest due to the development of deep convolution neural network architecture based algorithm, and advancement of computational and processing capabilities of the computational devices. A Convolutional Neural Network (CNN) consists of one or more convolutional layers with or without a subsampling step and then optionally followed by one or more fully connected dense layers similar multilayer perception based neural networks. The main advantage of applying CNN for image classification is that, CNN can take the advantage of 2D array structure of the input image through subsample pooling to derive invariant features. Furthermore, CNN architecture have fewer parameters compared to fully connected neural networks with same number of hidden units, therefore, CNN based classifier is much easier to train.

Project Overview

Distracted drivers are can be classified as drivers who are doing distracted driving activities include things like using a cell phone, texting, eating, and using in-vehicle technologies (such as navigation systems). Distraction can be classified into three main categories:

- Visual: taking your eyes off the road;
- Manual: taking your hands off the wheel; and
- Cognitive: taking your mind off of driving

According to the Centers for Disease Control and Prevention motor vehicle safety division, one in five car accidents is caused by a distracted driver. In number, 425,000 people injured and 3,000 people killed by distracted driving every year. In this project, we utilize CNN based image classification algorithm to classify distracted and safe driver from dashboard camera images. This will help insurance company to identify drivers' behavior and provide better insurance coverage based on their driving behavior.

Problem Statement

The goal of this project to develop a CNN based classification algorithm that can automatically classify drivers who are engaged in distracted driving behaviors. The workflow/steps for the development of the classification algorithm are listed below:

- Download the data from Kaggle.com and extract the data
- Preprocess the data set:
    - Convert RGB to grayscale image
    - Crop the images
    - Merge all image datasets
- Create train, test and validation dataset
- Develop simple CNN algorithm with one convolution layer and one dense layer.
- Improve the algorithm's performance adding more convolution, dropout and dense layer.

Dataset

The image dataset is provided by State farm and dataset is available in kaggle.com website [4]. In each image, a driver is engaged in activities. There are total 10 class of activities:

- c0: safe driving
- c1: texting - right
- c2: talking on the phone - right
- c3: texting - left
- c4: talking on the phone - left
- c5: operating the radio
- c6: drinking
- c7: reaching behind
- c8: hair and makeup
- c9: talking to passenger

Test data set is totally unlabeled. Therefore, it will very hard to measure accuracy with the provided test data. Hence, we will develop CNN based classification algorithm based on training data set for this project.

Accuracy Metrics

Since this is a multilabel classification problem, we utilized following equation to determine the accuracy of the classification algorithm.

$$Accuracy(y, \hat{y}_i) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (\hat{y}_i = y_i)$$

where $n_{samples}$ is the number of labels (which is 10 for this project), y_i represents true vector value (assuming one-hot encoding, it is a vector of zeros except one position which is the true category), $\hat{y}_i$ represents the estimation or prediction output at the current step. We have used true-positive and false-positive values for each label to see if any the algorithm struggles to classify any particular labels.

II. Analysis

Data Exploration

We have count the number of occurrence of each labels in the training set (12424 images). Figure 3 illustrates the distribution of these labels. It is obvious from the figure that the distribution is not uniform. Label 'c0' has most images (2489 images) while 'c8' has the least number of images (1911 images). Since the distribution is not uniform, we used stratified sampling to generate train, test and validation data set rather than using simple train-test-validation splitting. Table1 shows example figures of different labels.
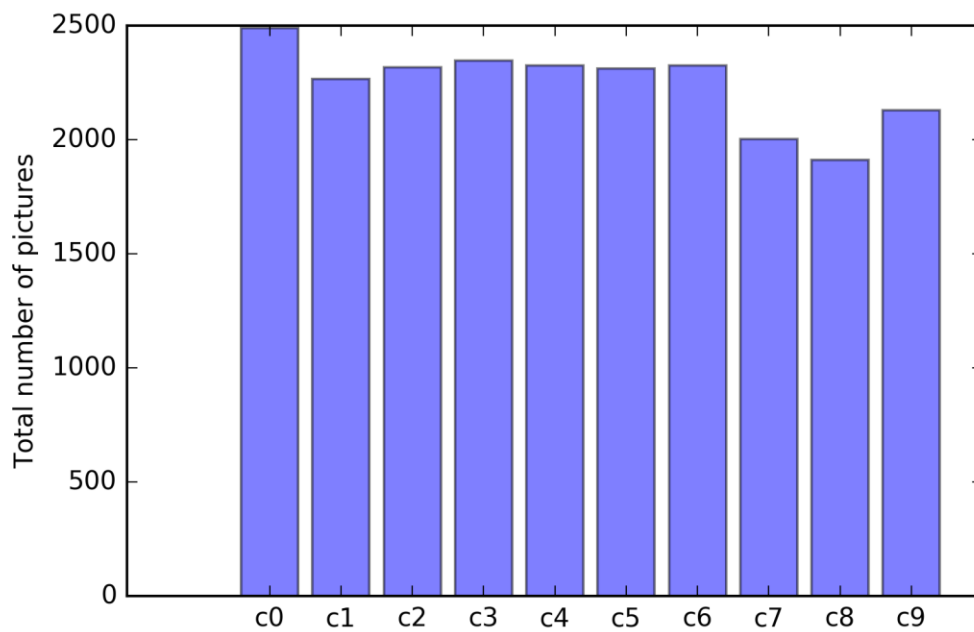
Figure 3: Total number of picture of different labels.

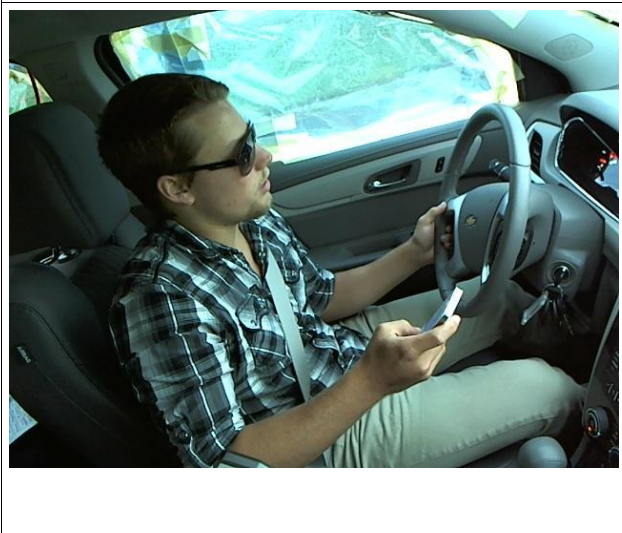Table1: Example images of different labels

Image	Label
	c0
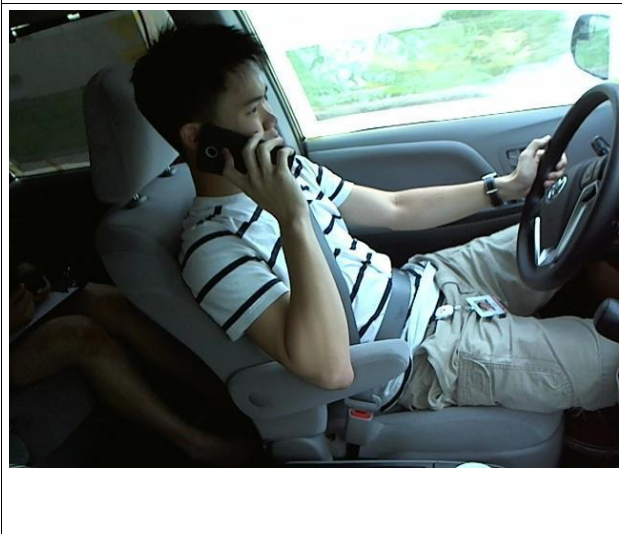
Image	Label
	c1
	c2
	c3

Image	Label
	c4
	c5
	c6

Image	Label
	c7
	c8
	c9

CNN Architecture

CNN is most commonly used for image classification type problems because of its superior accuracy compared to the other algorithms. For example: CNN outperforms many image recognition algorithms in ImageNet dataset [5], and MNIST dataset. CNN are very similar to feed forward neural networks where initial layers of neurons are replaced by convolution layers. A simple CNN architecture is shown in figure 1. In this section, we will discuss briefly about main operations of CNN: convolution layer, activation operation, pooling layer, fully connected dense layer, and classification or output layer.

Convolution Layer: In a CNN, convolution layer generally acts as an input data filtering layer. The filters are pre-defined and consists of learnable filter parameters. The learnable filter parameters have a small receptive field which extends through the full depth of the input array. Each filter is convolved across the length and width of the input array, and compute the dot product between the filter entries and input values to produce 2D activation map or features of the filter. The activation map is also known as feature map.

Activation Operation: Activation operation is typically a nonlinear operation (also known as ReLU). Activation operation is an applied to each elements of convolution output to replace all negative values of the feature map to zero. Activation operation introduces nonlinearity to the decision function (convolution is linear operation) without affecting the shape and size of convolution layer [7].

Pooling Layer: Pooling or subsampling is an important feature of CNN. Pooling develop a set non-overlapping rectangular partitions from the input array of the pooling layer, and outputs the maximum value for each partitions. Pooling layer provides a form translation invariance in CNN architecture and reduce the spatial size of the representation i.e. reduce the total number of parameters. As a result, computation time is reduced significantly. 2x2 or 3x3 are the most common pooling window size.

Fully Connected Dense Layer: Fully connected dense layer has connection to the all activations from the previous layer and it reduces the size of input data array to the size of the total number of labels by combining output of convolution layer with different weights.

Classification Layer or Output Layer: Classification layer is the final layer of the CNN algorithm. It converts the output of fully connected dense layer to probability of each image being in a label. For this project, softmax regression (multinomial logistic regression) algorithm is used in this layer [6].
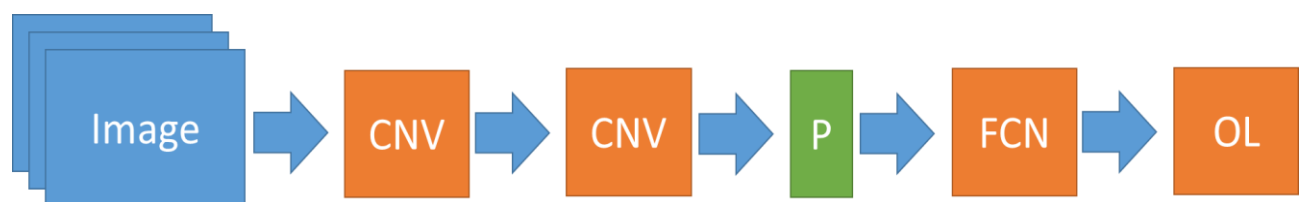

Figure 1: Simple CNN architecture where CNV, P, FCN and OL stand for convolution layer, pooling layer, fully connected dense layer, and output layer respectively.

Benchmark

The goal of this project to develop an efficient classification algorithm that can identify distracted driving activities from dash board camera images. However, no benchmark on the test dataset is available to compare the performance of the developed CNN algorithm. The CNN algorithm developed with manually hand-labeled dataset. Considering the accuracy of manual hand-labeling is fairly high (about 99%), the benchmark should be at least 99%. However, developing such an accurate classification algorithm requires extensive state of art high performance computing facility preferably graphical processing unit (GPU). Due to my limited resource capability, I decided to target an accuracy of 98%.

III. Methodology

Data preprocessing

Data preprocessing code is locate in 'data_preprocessing.py' file. For data preprocessing following steps are performed:

- Extract and read data from all 10 folders.

- Convert the images to grayscale from rgb(color).

- Resize the images to 50x50 size.

- Split the data set into train, test and validation using stratified sampling.

- Reshape test, train and validation data set into proper format.

- Convert the output label to a matrix with a boolean for each class using one hot encoding.

- Final train set consists 12019 images, validation set consists 3005 images, and test set contains 7400 images.

Implementation

In order to implement CNN algorithm following steps are implemented:

- Construct a simple CNN following the structure shown in figure 3. The CNN structure composed of following layers:

  ◦ Convolution layer 1: kernel size 3x3, output shape 48x48x32

- Convolution layer2: kernel size 3x3, output shape 46x46X32

- Pooling layer1: pool size 2x2, output shape 23x23x32

- Fully connected layer1

- Output layer

- We also added a ReLU layers after each layer to add nonlinearity into algorithm.

- CNN is trained with using batch size of 32

- Accuracy/ Loss function for the CNN algorithm is 'categorical_crossentropy'

- Optimizer for the CNN algorithm is stochastic gradient descent (sgd)

Accuracy

The accuracy of the above described CNN algorithm for the validation set is 98.50%, and for test set is 98.24% after 20 epochs. The accuracy is certainly above the benchmark value; which also indicates how effectively CNN works for image classification.

Improvement

Although, simple CNN algorithm performs above benchmark conditions, we still made some changes to improve its performances. The changes are:

- Explore larger kernel size convolution networks and larger pooling size.

- Explore rgb images instead of grayscale images.

- Add dropout layers.

Final model topology looks like this:

- Convolution layer 1: kernel size 5x5, output shape 46x46x32

- ◦ Dropout layer1: dropout 0.2

- ◦ Convolution layer2: kernel size 5x5, output shape 42x42X32

- ◦ Pooling layer1: pool size 2x2, output shape 14x14x32

- ◦ Dropout layer2: dropout 0.2

- ◦ Fully connected layer1

- ◦ Dropout layer3: dropout 0.5

- ◦ Output layer

- ◦ We also added a ReLU layers after each layer to add nonlinearity into algorithm.

Final accuracy of the model on test set is 98.797%.

IV. Results

The accuracy of the final model is 98.797% on test set which above the benchmark set for this project. The usage of larger kernel size, larger pooling size, and dropout layers improved the performance of the algorithm significantly. Furthermore, usage of dropout layers also made the model robust and generalized enough to perform well on most of the data.

Figure 2 shows a plot of accuracy on the training and validation datasets over training epochs and figure 3 shows a plot of loss on the training and validation over training epochs. From figure 5, accuracy do not change very much after epoch 45 for both train and validation dataset. Similarly, loss does not decrease that much after epoch 45. This indicates the model is not over-trained.
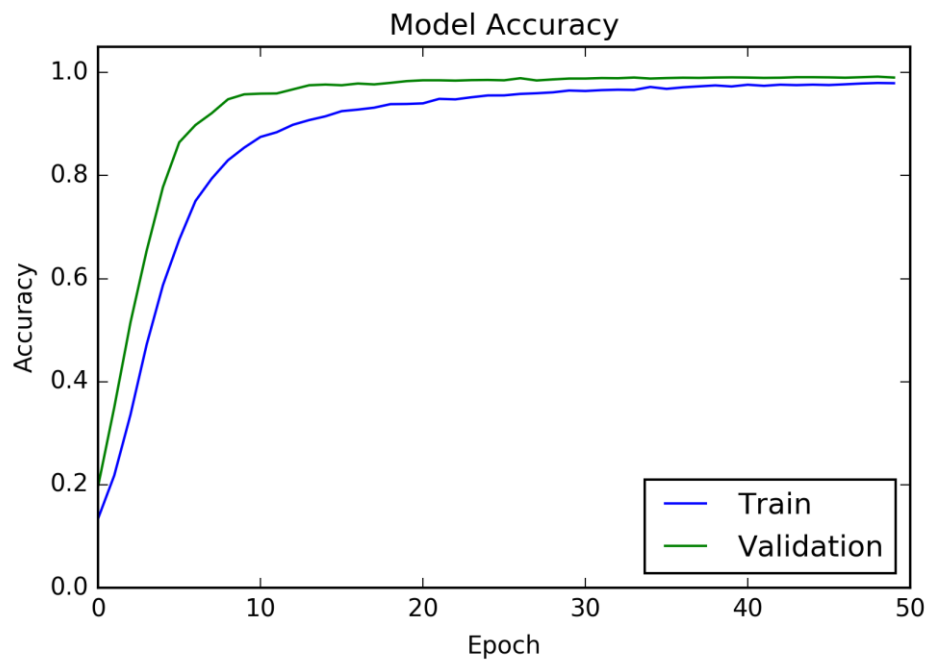
Figure 2: Accuracy curves on the training and validation datasets over training epochs.
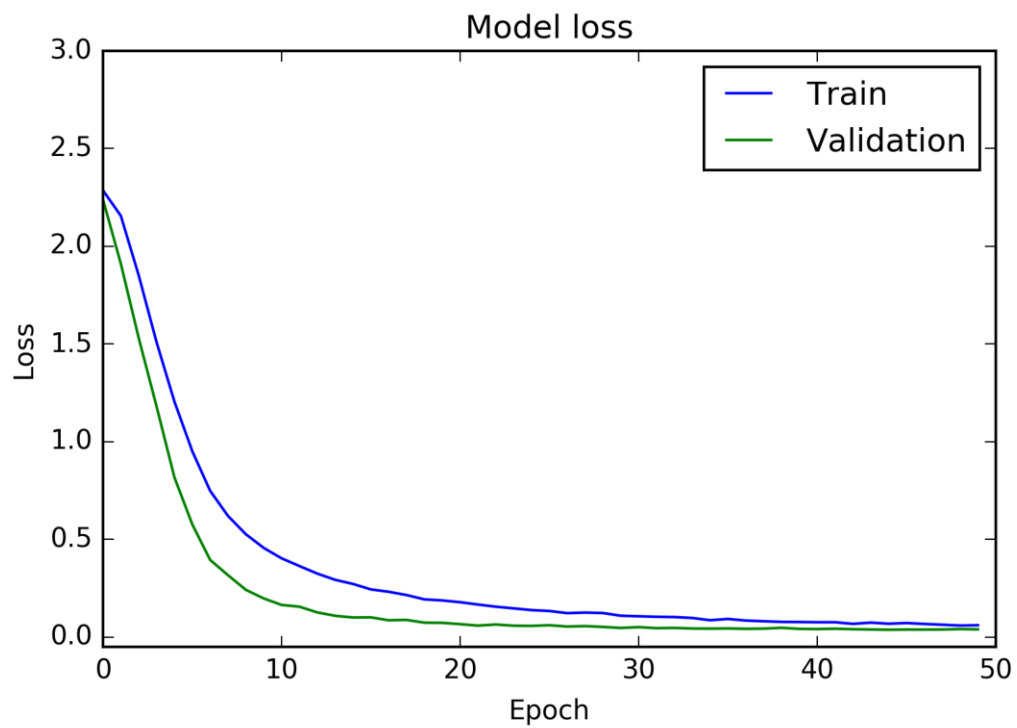

Figure 3: Loss curves on the training and validation datasets over training epochs.

The label wise classification report of the model is listed in table 1. Average precision, recall and f1-score for all labels are 0.98. From precision, recall and f1-scores from individual labels also indicate how well the model is performing.

Table 2: Classification report for the model

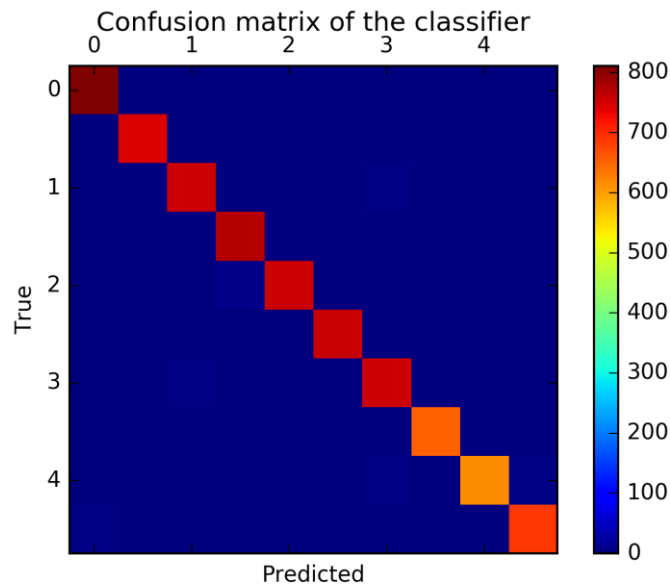| Label	Precision	Recall	f1-score	support
c0	0.98	0.99	.98	821
c1	0.99	1	.99	748
c2	0.98	.99	.99	765
c3	0.99	.99	.99	774
c4	1	.98	.99	768
c5	1	1	1	763
c6	0.98	.99	.99	767
c7	0.99	.99	.99	661
c8	0.98	.98	.98	631
c9	0.99	.98	.98	702

Figure 4: Confusion matrix of the test dataset.

Figure 4 shows the confusion matrix of the model on test dataset where we can see that model slightly under-performs for c8 labels.

Justification

The model accuracy is well above the set benchmark of 98%. Due to the time constraints and hardware constraints, the model is trained with very small epoch for a relative short duration. However, it is expected that accuracy will increase further if we can run it for more epochs. Training time for each epoch is approximately 185 seconds. Training time can be also significantly reduced if we run the model on GPU or high performance clusters.

The aim of this project is to develop a CNN algorithm based classification model that can classify safe drivers and different types of distracted driving acts. Based on the accuracy of the model, we can fairly say that the model is quite effective to classify safe drivers and distracted driving activities.

V. Conclusion

Free-form visualization

Table 2 shows some example output of the model. Model predicts correct classification for all images except c8 label. c8 label stands for hair and makeup and the picture does not show any makeup or hair

activity, that's why it is very difficult to identify the proper label for that picture. Moreover, the model might fail if

- Image is blurry

- Unwanted noise in image making very difficult detect features

Table 2: Example model output

Image	True label	Predicted label
	5	5
	9	9
	3	3

Image	True label	Predicted label
	c4	c4
	c8	c9
	c0	c0
	c1	c1

Image	True label	Predicted label
	c6	c6
	c2	c2
	c7	c7

Reflection

The project is divided into two major parts:

- data preprocessing
- CNN model

In data preprocessing, we run following the steps:

- Extract and read image data.

- Convert the images to grayscale from rgb, and resize the images to 50x50 size.

- Split the data into train, test and validation using stratified sampling.

- Reshape test, train and validation data set into proper format.

- Convert the output label to a matrix with a Boolean for each class using one hot encoding.

For CNN model, following steps are performed:

- Define the CNN architecture with proper accuracy, learning rate and activation layers.

- Run the CNN model and check validation accuracy

- Tweak the model by changing kernel size, pooling size, and inserting dropout layers

- Save the final model parameters, and report the accuracy on test dataset.

The most important and critical aspects of the CNN model development is choosing the right architecture for the CNN since it is really hard to understand why some particular architecture performs well for some particular type of datasets.

Improvement

In this project, the model struggles to predict c8 levels properly. Furthermore, I believe model will face difficulties when the image is blurry or there is a lot of noise in the image. So for future, we can explore how model performs with blurry and noisy images.

References:

1. http://ufldl.stanford.edu/tutorial/supervised/ConvolutionalNeuralNetwork/
2. http://cs231n.github.io/classification/
3. https://www.cdc.gov/motorvehiclesafety/distracted_driving/
4. https://www.kaggle.com/c/state-farm-distracted-driver-detection/data
5. http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neuralnetworks.pdf
6. http://ufldl.stanford.edu/tutorial/supervised/SoftmaxRegression/
7. https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/
8. Udacity and kaggle forums
9. http://machinelearningmastery.com/display-deep-learning-model-training-history-in-keras/