Faculty of Computer Science and Engineering
Ho Chi Minh City University of Technology

– Intelligent System – Semester 201 –

Assignment
Distracted Driver Detection

GROUP 6

No.	Name	Student ID
1	Nguyen Anh Hoang Phuc	1752041
2	Truong Minh Duy	1652113
3	Tran Trung Quan	1752044

Instructor: Assoc. Prof. Quan Thanh Tho

Contents

1 Introduction

1.1 Overview

Every country has its law to enforce the driver to focus on driving. This is mainly because driving a car is a complex task, and it requires complete attention. However, it is very common to see drivers doing something else during their trips: drinking, texting, talking on the phone, reaching behind, etc. In general, many research has classified three main types of distraction: visual distractions (driver's eyes off the road), manual distractions (driver's hands off the wheel), and cognitive distractions (driver's mind off the driving task). No matter which type of the distractions, this behavior easily lead to serious incidents.

According to the CDC motor vehicle safety division, one in five car accidents in the US is caused by a distracted driver. Sadly, this translates to 425,000 people injured and 3,000 people killed by distracted driving every year [1]. In Vietnam, Ford Viet-Nam reported 87% young adults do not concentrate when driving [5]. These alarming statistics emphasize the necessary of detect distracted driver action.

Figure 1: The distracted driver [5]

Recently, it is undoubted that machine learning and its subset, deep learning, achieve incredible results in the computer vision field. Moreover, currently, most major cities in the world install surveillance camera systems because police cannot be on all roads at any one time to monitor drivers. Therefore, machine learning and deep learning techniques seem promising to sidestep this problem.

In 2016, State Farm held an online Kaggle competition to encourage Kaggler to build a robust computer vision system [4] . Although this competition is ended, using their provided data, our group try to make an autonomous image-based distracted driver detecting system. This task is very meaningful for improving the drivers' safety and further secure their customers.

1.2 Goal

The goals of this project include:

- Survey and discover some machine/learning techniques, especially in the computer vision field.
- Apply machine learning techniques to build an application.
- Build an application that can predict the distracting action by a/an input image/video

1.3 Application features

- The application can receive either image or video as its input.
- The application can predict the distracting action (belong to 10 labels) with the accepted accuracy.
- The time for prediction should be reasonable to apply to real-life.

2 Theoretical basis and Technologies used

2.1 Machine learning

2.1.1 Support vector machine

2.1.1.1 Classical SVM

Support vector machine (SVMs) [2] are among the best (and many believe are indeed the best) "off-the-shelf" supervised learning algorithms. To illustrate this algorithm idea, consider the following figure, in which x's represent positive training examples, o's denote negative training examples, a decision boundary (and is also called the separating hyperplane) is also shown, and three points have also been labeled A, B and C.

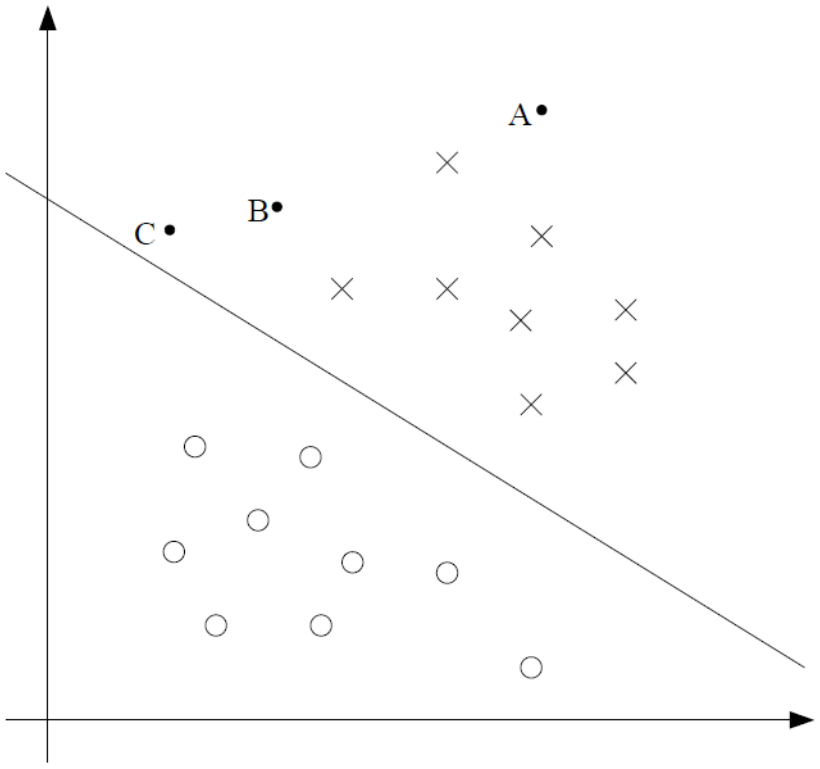

Figure 2: Support Vector Machine illustration

Notice that the point A is very far from the decision boundary. If we are asked to make a prediction for the value of y at A, it seems we should be quite confident that $y = 1$ there. Conversely, the point C is very close to the decision boundary, and while it's on the side of the decision boundary on which we would predict $y = 1$, it seems

likely that just a small change to the decision boundary could easily have caused out prediction to be $y = 0$. Hence, we're much more confident about our prediction at A than at C.

The point B lies in-between these two cases, and more broadly, we see that if a point is far from the separating hyperplane, then we may be significantly more confident in our predictions. Again, informally we think it would be nice if, given a training set, we manage to find a decision boundary that allows us to make all correct and confident (meaning far from the decision boundary) predictions on the training examples.

Our group does not present the mathematical background in this report as it was quite difficult. In short, we solve it using the dual optimization problem in lieu of solving the original (primal) problem. However, the parameter α only differs from 0 as "support-vector" points (or points which are nearest with decision boundary). This property makes SVMs require less time compared to the artificial neural network.

$$\begin{aligned} \max_\alpha \quad & W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle. \\ \text{s.t.} \quad & \alpha_i \geq 0, \quad i = 1, \ldots, n \\ & \sum_{i=1}^{n} \alpha_i y^{(i)} = 0, \end{aligned}$$

Figure 3: Dual optimization problem

2.1.1.2 Kernel trick

In classical form, SVMs always represent the linear function. If the data is not linearly separable, classical SVMs cannot work well. However, we can use kernel trick to map the data from original space to the new space. If the data in the new space is linearly separable, we can use the SVMs on it.

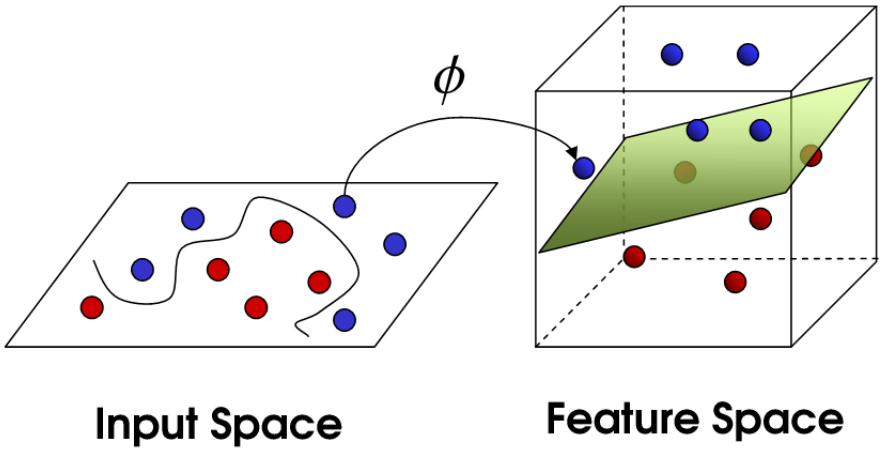

Figure 4: Kernel trick

2.1.2 Softmax function

The softmax function, is a generalization of the logistic function to multiple dimensions. It is used in multinomial logistic regression and is often used as the last activation function of a neural network to normalize the output of a network to a probability distribution over predicted output classes.

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \text{ for } i = 1, \ldots, K \text{ and } \mathbf{z} = (z_1, \ldots, z_K) \in \mathbb{R}^K$$

Figure 5: Softmax function

2.1.3 Artificial Neural Networks (ANN)

Artificial Neural Networks or ANNs [3] is an information retrieval model that is inspired by the way the biological nervous system, such as brain information, processes information. It consists of a large number of highly interconnected computing components (neurons) working together to solve a particular problem.

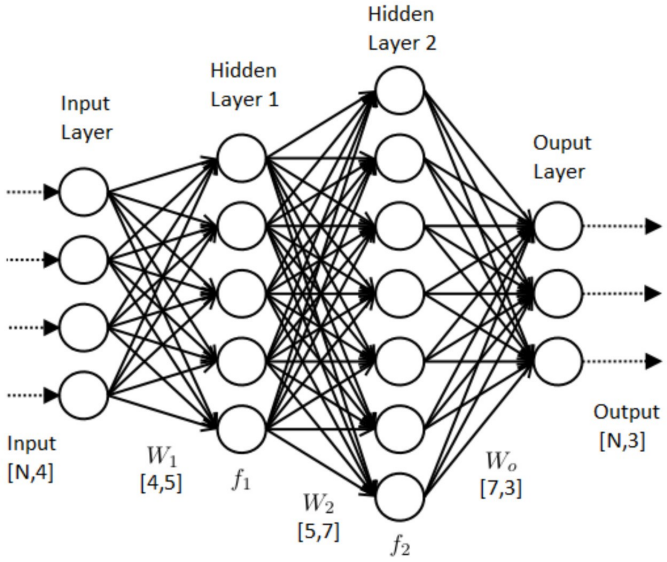

Figure 6: Artificial Neural Networks

The neural network is made up of layers. The layers consist of an interconnected network of units (or nodes) called artificial neurons. An artificial neuron may receive a signal, process it, and send other signal neurons attached to it. A neuron can integrate feedback with a series of weights such that it can assign the value of each input to learn which input is more useful when it comes to the problem it is attempting to solve. Since incorporating it, the weights of the inputs, then the weighted inputs, are added up and transferred through the activation function. This step is to decide if the information it contains can be further analyzed and forwarded to the next layer. If the signal passes, we assume the neuron has been triggered.

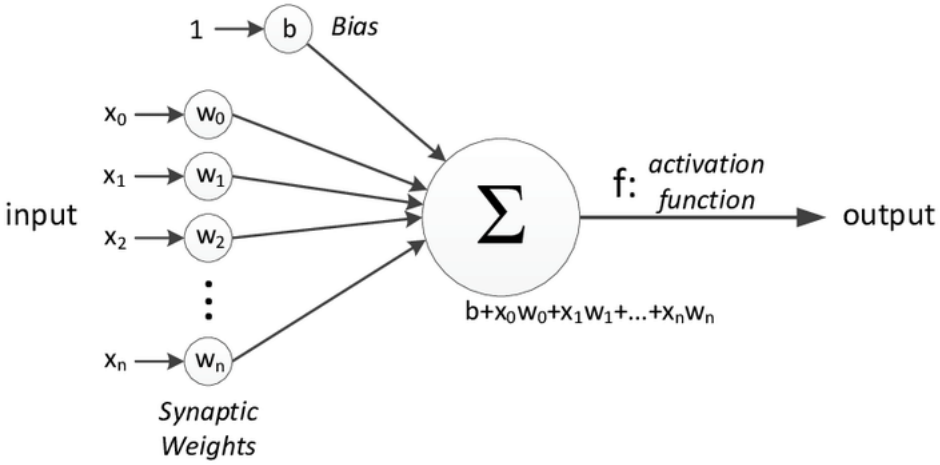

Figure 7: Inside a neuron

Structure of a neuron node

Inputs: inputs are usually displayed as a vector with dimension of N.

Summation function: to calculate the sum of products of the inputs and their weights.

$$net_i = \sum_{j=1}^{N}(w_{ij}x_i + \theta_i)$$

$x_i(i = 1, 2, ..., N)$: *Set of inputs*
w_{ij} : *Connection weights*
θ_i : *Bias term*
net_i : *Network input signal*

Activation function: the output of this function will be the output of the neuron.

$$y = f(net_i)$$

2.1.4 Gradient descent and stochastic gradient descent (SGD)

Gradient descent

Gradient descent is a first-order iterative optimization algorithm for finding a local minimum of a differentiable function. The idea is to take repeated steps in the opposite direction of the gradient (or approximate gradient) of the function at the current point, because this is the direction of steepest descent. In other words, it is an iterative algorithm, that starts from a random point on a function and travels down its slope in steps until it reaches the lowest point of that function.

Stochastic gradient descent

Stochastic gradient descent is similar to gradient descent, except it randomly picks one data point from the whole data set at each iteration to reduce the computations enormously.

2.2 Deep learning

2.2.1 Convolutional Neural Network

Convolutional Neural Networks (CNN or ConvNets) [3] are ordinary neural networks that assume that the input is an image. They are used to evaluate and identify images, cluster images by similarities, and perform object recognition in a frame. For example, convolutional neural networks are used to classify faces, people, street signs, tumors, platypuses, and many other facets of visual data.

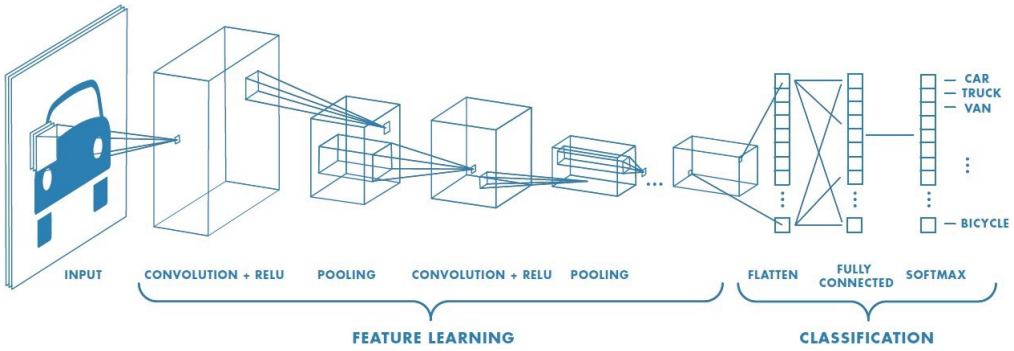

Figure 8: Neural network with many convolutional layers

There are four main layers in CNN:

- Convolutional Layer
- Rectified Linear Units Layer (ReLU)
- Pooling Layer
- Fully connected Layer

These operations are repeated over hundred layers, so that different features of each layer are identified.

Convolutional Layer

Convolution operation operates with 2 signals in 1D and 2 images in 2D. The main purpose of a convolutional layer is to detect patterns or visual features in images such as borders, shapes, color drops, etc.

CNN uses filters (also known as kernels, feature detectors) to detect features such as edges that are present in the image. A filter is just a matrix of values, called weights, which are learned to detect unique features. The filter passes over each part of the image to check whether the function it is supposed to detect is present. In order to have a value that indicates how sure it is that a certain function is present, the filter performs a convolution operation, which is an element-wise product and a calculation of two matrices.

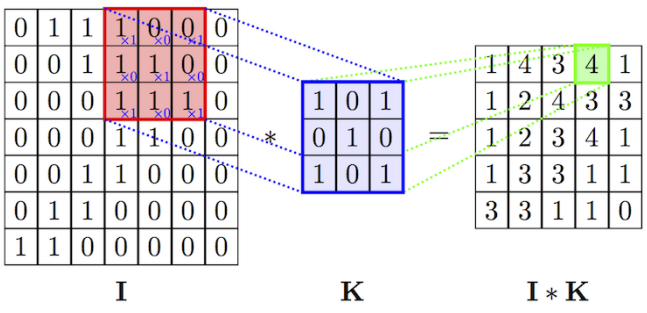

Figure 9: Convolutional Layer

Rectified Linear Units Layer

For ReLU, negative values must be omitted and assigned to 0 by the function index. A matrix of the same size as the entry is the output of this level. ReLU is superior because it trains neural networks quicker without degrading their performance relative to other features.

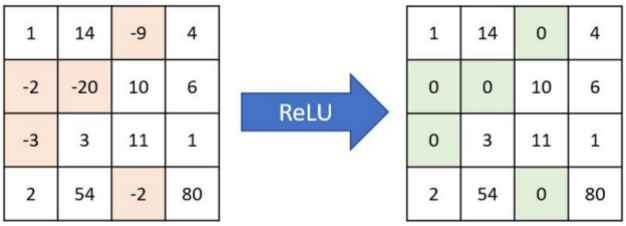

Figure 10: ReLU Layer

Pooling Layer

After ReLU comes to a pooling step, in which the CNN downsamples the convolved feature (to save the time needed for processing) and reduces the image size. This helps to minimize the over-fitting that would arise if CNN were given too much information, particularly if that information was not important to the image classification.

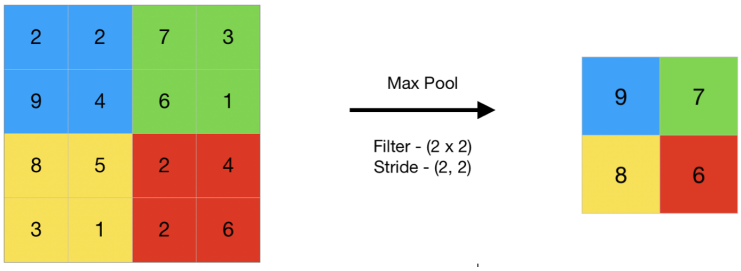

Figure 11: Pooling Layer

Fully Connected Layer (Dense Layer)

This layer is used for recognition and classification. The number of nodes in this layer refers to the number of classes to be classified. This layer would fully connect the nodes in the previous layer to the next layer. The fully connected layers are commonly used at the end of a CNN because the data size is no longer so big after compilation by other layers that it can be passed directly to that layer.

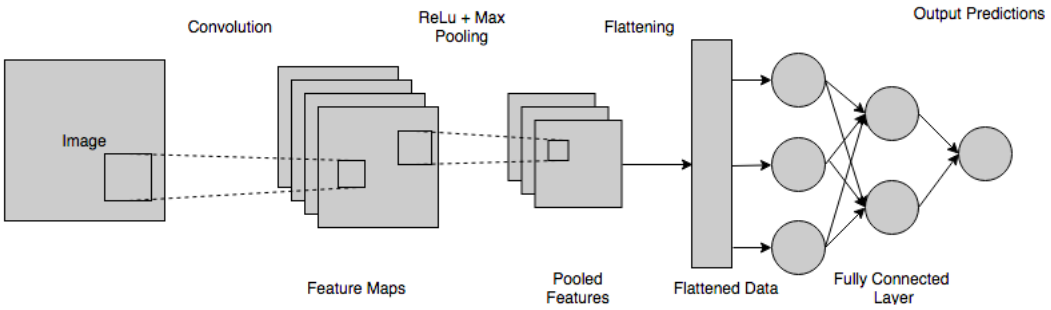

Figure 12: Fully Connected Layer

2.2.2 Some other techniques/layers

2.2.2.1 Dropout

Dropout is a technique used to prevent a model from overfitting. Dropout works by randomly setting the outgoing edges of hidden units (neurons that make up hidden layers) to 0 at each update of the training phase. More technically, at each training stage, individual nodes are either dropped out of the net with probability 1- p or kept with probability p, so that a reduced network is left; incoming and outgoing edges to a dropped-out node are also removed.

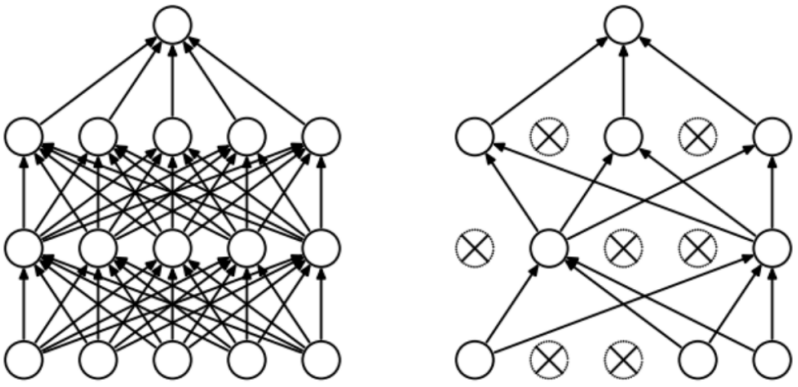

Figure 13: Dropout technique

2.2.2.2 Batch normalization

Batch normalization is a technique for training very deep neural networks that standardizes the inputs to a layer for each mini-batch. This has the effect of stabilizing the learning process and dramatically reducing the number of training epochs required to train deep networks.

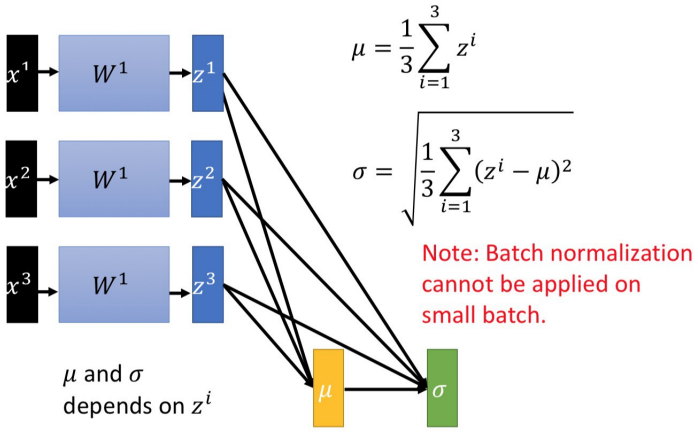

Figure 14: Batch normalization

2.2.2.3 Softmax layer

As mentioned above, softmax can be used at the final output of the network, as it can produce the categorical distribution.

2.2.3 VGG16

VGG16 model [6] has 16-layers. The basic model as in the figure below. However, in our assignment, we apply some extra techniques as we mentioned above.

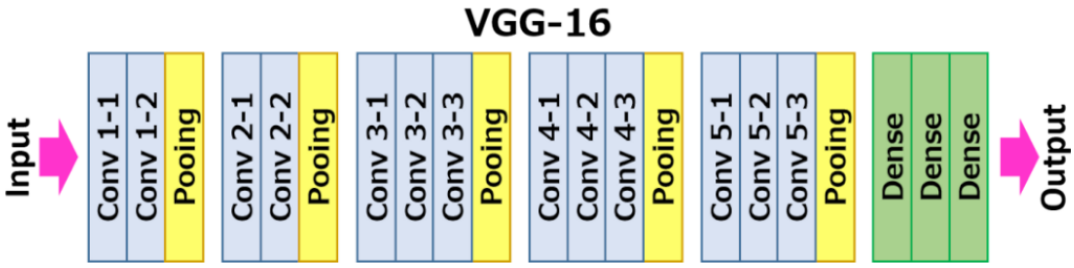

Figure 15: VGG16 Architecture

2.3 Technologies used

2.3.1 Programming Language

2.3.1.1 QML

QML is a JavaScript-based language used to construct an application's user interface. It helps you to use both the standard components (buttons, tables, etc.) and the graphical elements to which logic is applied. The Ubuntu Touch user interface is an example of the first scenario. There are several examples on the QT Site for the second case. QML performs the visualization, but not the logic of the program.

This logic can be written in a variety of languages, depending on the needs of the application. If the program is a game and requires computational resources, the preferred language will be C/C++. In the other hand, if the output of the software isn't important, you should use JavaScript or Python. In this project, we choose Python for the logic of the program.

2.3.1.2 Python

Python is an interpreted, high-level, and general-purpose programming language created in the late 1980s, and first released in 1991, by Guido van Rossum. Python was developed in an open-source project, managed by the nonprofit Python Software Foundation.

Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is widely used in the area of Machine Learning with several powerful support libraries such as OpenCV for image processing or the fast deployment of Machine Learning and Deep Learning frameworks such as Tensorflow.

2.3.2 Framework and Library

2.3.2.1 Qt Quick

Qt Quick is a free software development framework developed and managed by the Qt Community under the framework of Qt. It provides a collection of technologies that are designed to help developers create the kind of intuitive, modern, and fluid user interfaces that are increasingly used on mobile phones, media players, set-top boxes, and other portable devices.

For our project, the driver distracted detection program may install on a car or in the main operator. Therefore, the Qt Quick framework is suitable for the development of this application.

2.3.2.2 PyQt5

PyQt5 is a set of Python bindings for Qt5 application framework. Qt library is one of the most powerful GUI libraries. PyQt5 is implemented as a set of Python modules. It has over 620 classes and 6000 functions and methods. It is a multiplatform toolkit which runs on all major operating systems, including Unix, Windows, and Mac OS. PyQt5 is dual licensed.

In this project, we use PyQt5 as a bridge between User Interface by Qt Quick and the logic of the program written in Python.

2.3.2.3 OpenCV

OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning applications library. OpenCV was developed to provide a shared infrastructure for computer vision applications and to accelerate the use of machine perception in consumer products. OpenCV was originally developed by Intel back in 1999 and is now being maintained by Willow Garage and ItSeez. OpenCV (Open Source Computer Vision) is a library of real-time computer vision programming functions that uses a BSD license and is also free for both academic and commercial use. It has C++, C, Python and Java (Android) interfaces and supports Windows, Linux, Android, iOS and Mac OS. It has over 2,500 optimized algorithms. OpenCV has more than 9 million downloads worldwide and is rising by almost 200K/month. Usage ranges from digital painting, mining inspection, web-based stitching of maps by advanced robotics.

We use OpenCV-Python in our project. OpenCV-Python is a library of Python bindings designed to solve computer vision problems. OpenCV-Python uses Numpy, a highly structured library for numerical operations with MATLAB-style syntax. Both OpenCV array structures are converted from and to Numpy arrays. This also makes it easy to combine with other Numpy libraries, such as SciPy and Matplotlib.

2.3.2.4 Tensorflow

TensorFlow is a software library and open source machine learning for numerical computation. It uses nodes to represent mathematical operations and graph edges to represent multidimensional data arrays or tensors interacted between them in the data graph. Its architecture is flexible and helps users to expand computing to one or more CPUs or GPUs on a desktop, server or mobile device with a single API. TensorFlow is written in C++ and currently has Python, C++, Java and Go APIs available. It will construct a graph for computation and perform symbolic distinction.

In our project, we use the Tensorflow library in Python to load our pre-trained model and applied it to predict the distraction action on the image.

3 Initial Experiment

3.1 Data

Our data is collected from Kaggle competition [4], which is already divided into two part : train data and test data. However, the label for test data is unknown.

Figure 16: Sample image in our data

```
0 - Normal driving
1 - Texting - right
2 - Talking on the phone - right
3 - Texting - left
4 - Talking on the phone - left
5 - Operating the radio
6 - Drinking
7 - Reaching behind
8 - Hair and makeup
9 - Talking to passenger
```

Figure 17: Ten labels for our data

3.2 Initial Experiment

In the machine learning context, support vector machine and softmax regression are usually considered as two powerful classifiers. The advantage of two techniques is that it can provide good results in some cases without requiring a large training time as neural networks. In our initial experiment, we use support vector machine (SVM) with **polynomial kernel** to solve our problem.

Our initial pipeline as in the figure 18 below :

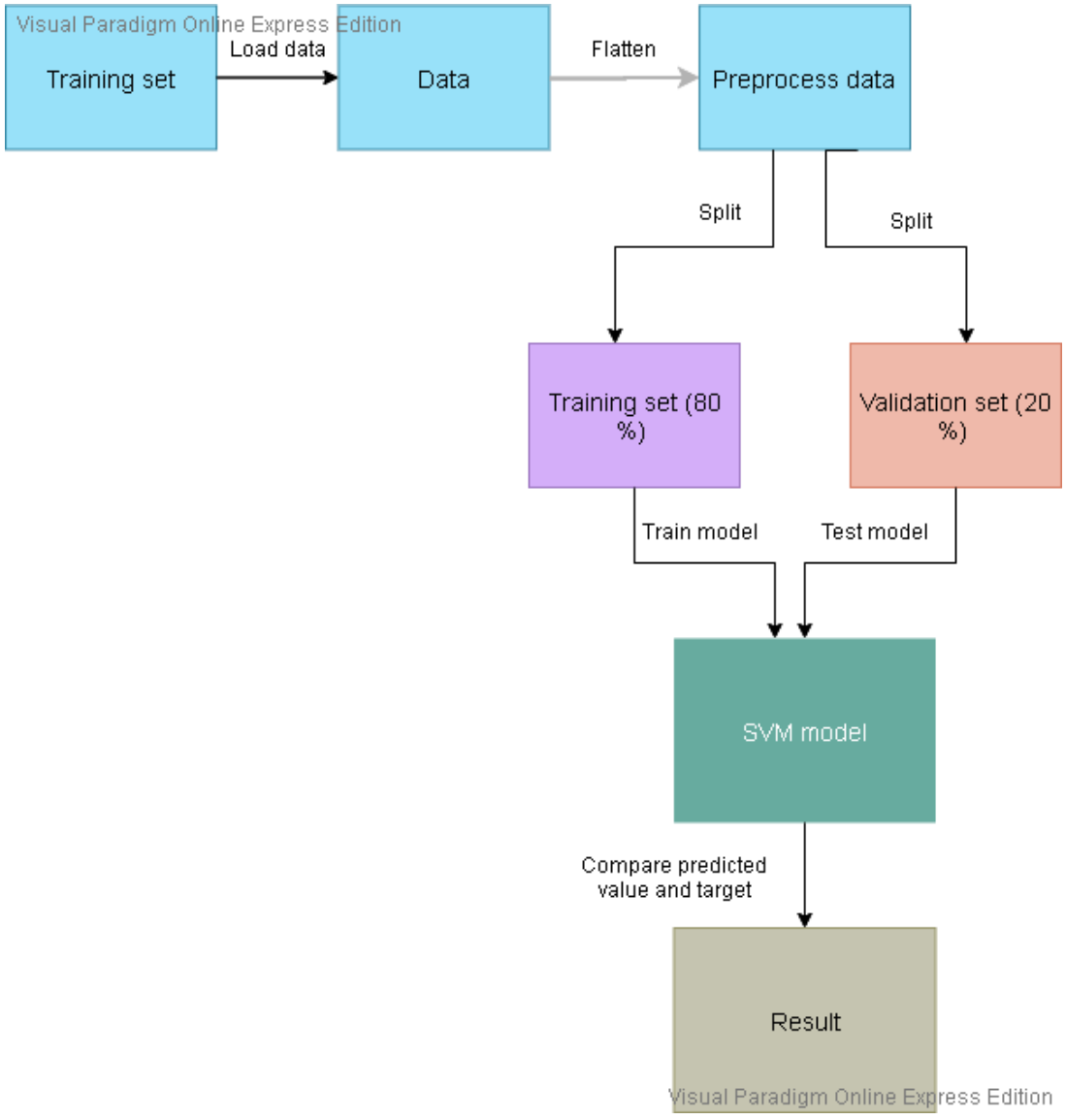

Figure 18: The initial pipeline

3.3 Initial results

3.3.1 Validation set

In short, SVM model provides the very good results in validation set.

Classification report for classifier SVC(kernel='poly', max_iter=100):

	precision	recall	f1-score	support
0	0.93	0.98	0.95	463
1	0.97	0.98	0.98	500
2	0.97	0.97	0.97	480
3	0.97	0.93	0.95	491
4	0.92	0.95	0.94	466
5	1.00	1.00	1.00	466
6	0.98	0.96	0.97	457
7	0.99	0.99	0.99	377
8	0.95	0.91	0.93	390
9	0.96	0.95	0.96	395
accuracy			0.96	4485
macro avg	0.96	0.96	0.96	4485
weighted avg	0.96	0.96	0.96	4485

Figure 19: The classification report of SVM model on validation set

Confusion matrix:

```
[[452   3   0   4   1   0   0   0   2   1]
 [  5 491   3   1   0   0   0   0   0   0]
 [  2   6 468   0   0   0   2   0   1   1]
 [  3   0   1 456  31   0   0   0   0   0]
 [ 10   2   0   7 445   0   1   0   1   0]
 [  0   0   0   0   1 464   0   0   1   0]
 [  3   3   6   0   2   0 441   0   1   1]
 [  2   0   0   0   1   0   0 374   0   0]
 [  6   2   1   1   4   0   5   4 355  12]
 [  3   0   2   0   0   0   0   0  13 377]]
```

Figure 20: The confusion matrix of SVM model on validation set

3.3.2 Test set

Our group does not have the label of test set. Therefore, our group must label by themselves. The figure 21 and 22 show our model results on test set (contain 1000 pictures). Although the very good results in validation set, our model cannot provide the acceptable result on testing set.

	precision	recall	f1-score	support
0	0.38	0.39	0.38	131
1	0.34	0.47	0.40	113
2	0.50	0.31	0.38	91
3	0.43	0.39	0.41	106
4	0.46	0.29	0.35	107
5	0.60	0.70	0.65	113
6	0.49	0.40	0.44	107
7	0.58	0.58	0.58	104
8	0.33	0.51	0.40	80
9	0.20	0.19	0.19	48
accuracy			0.44	1000
macro avg	0.43	0.42	0.42	1000
weighted avg	0.45	0.44	0.43	1000

Figure 21: The classification report of SVM model on testing set

```
Confusion matrix:
[[51 18  2 17  6  8  2  4 13 10]
 [ 9 53  8  6  1  3  4 10 12  7]
 [ 6 14 28  4  2  8  8  8 13  0]
 [16 11  4 41 13  8  0  2  7  4]
 [10 11  2 13 31  6  4  7 15  8]
 [ 6  8  0  1  6 79  5  0  7  1]
 [11 19  5  2  2  7 43  6  8  4]
 [ 5  7  6  2  3  4 11 60  5  1]
 [ 7  9  1  4  1  4  8  4 41  1]
 [15  4  0  5  3  4  2  2  4  9]]
```

Figure 22: The confusion matrix of SVM model on testing set

3.3.3 Our problems

After review our data and pipeline, our group recognize some problems:

- Digital images store information in pixel format. In general, the neighbor pixels may have similar value compare to farther pixels. The picture stores information not only in pixel alone but also in the group of the pixel. When we flatten the data, we drop that information as we consider all pixels equally.
- When reviewing our data, we recognize that some pictures belong to the same person. Our group recognizes that although each driver only appears on the training set and test set, each driver can appear multiple times. To illustrate this, please sea the Figure 23
- The test set size is large more than four times as training test size. This property makes our model easy to overfit.

Figure 23: Person appear two times in train data

Therefore, we redesign our model to solve those problems.

4 System design

4.1 AI design

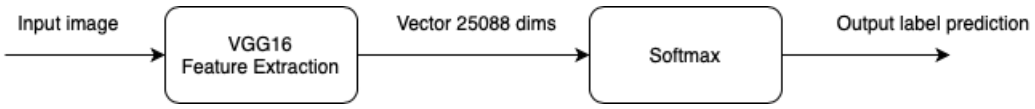

Figure 24: Pipeline of VGG16 with Softmax

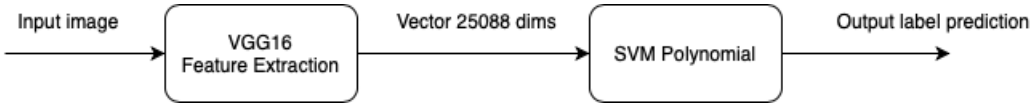

Figure 25: Pipeline of VGG16 with SVM Polynomial

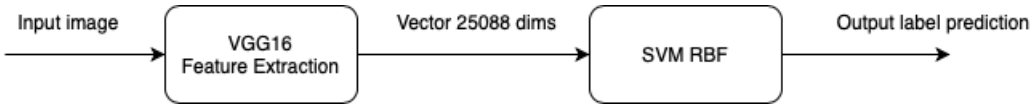

Figure 26: Pipeline of VGG16 with SVM RBF

4.2 System design

4.2.1 Software architecture

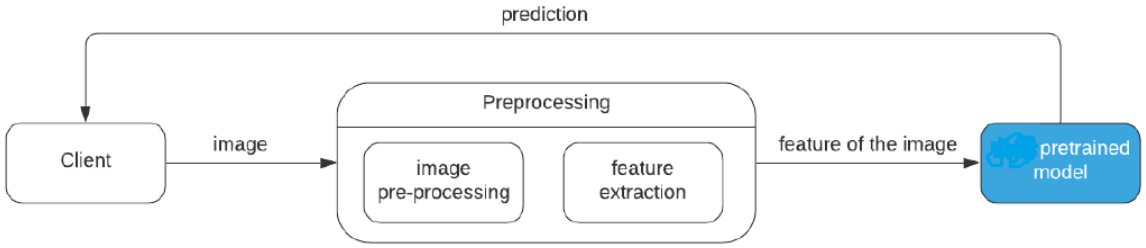

Figure 27: Distracted Driver Detection System Architecture

4.2.2 Activity diagram

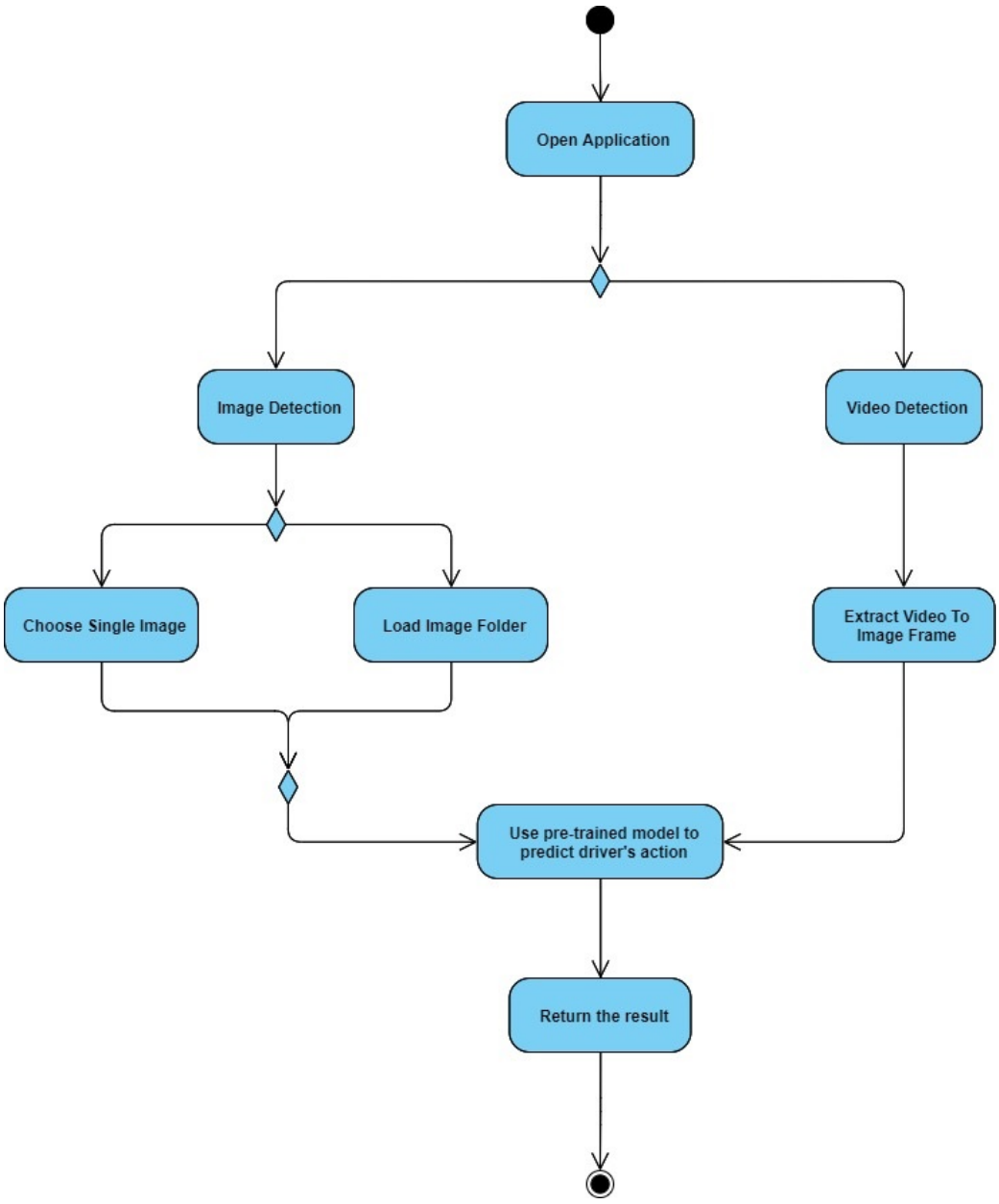

Figure 28: Distracted Driver Detection Activity Diagram

5 Evaluation

Our group uses some basic metrics: accuracy, precision, recall, F1-score to evaluate the model. We test the model on both the validation and test set. As we mentioned above, the label of the test set is private. Therefore, our group tries to label the test set by ourselves.

5.1 Training Result

Model	Training accuracy	Training loss	Validate accuracy	Validate loss
VGG16+Softmax	0.9046	0.2982	0.8806	0.4684

Table 1: Training VGG16+Softmax model with learning rate=0.001

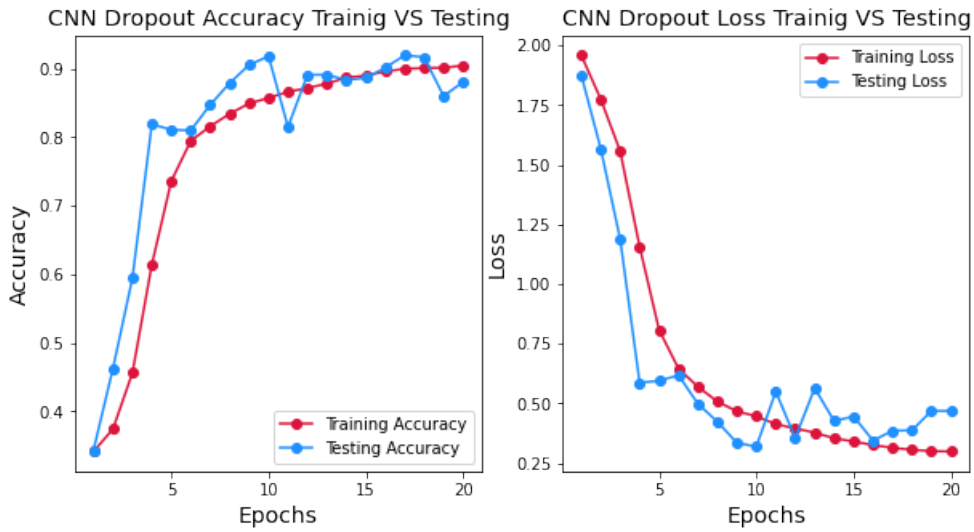

Figure 29: Validation result of VGG16+Softmax model with learning rate=0.001

5.2 Test set

Model	Metric			
	Accuracy	Precision	Recall	F1-score
VGG with SVM polynomial	0.54	0.77	0.54	0.57
VGG with SVM RBF	0.91	0.78	0.76	0.75
VGG with softmax	0.89	0.73	0.73	0.72

Table 2: The classification report in test set

Model	Accuracy class									
	0	1	2	3	4	5	6	7	8	9
VGG with SVM Polynomial	0.41	0.88	0.89	0.96	0.73	0.71	0.54	0.49	0.04	0
VGG with SVM RBF	0.95	0.98	0.91	0.96	0.76	0.95	0.81	0.95	0.24	0
VGG with softmax	0.92	0.99	0.96	0.99	0.70	0.67	0.87	0.82	0.28	0

Table 3: The accuracy for each class in test set

```
Classification report for classifier SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='poly',
    max_iter=100, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=1):
              precision    recall  f1-score   support

           0       0.96      0.26      0.41      1334
           1       1.00      0.79      0.88       613
           2       1.00      0.80      0.89       470
           3       1.00      0.92      0.96       357
           4       0.98      0.58      0.73       301
           5       0.96      0.57      0.71       204
           6       0.91      0.38      0.54       168
           7       0.86      0.34      0.49       106
           8       0.02      0.74      0.04        46
           9       0.00      0.00      0.00        17

    accuracy                           0.54      3616
   macro avg       0.77      0.54      0.57      3616
weighted avg       0.96      0.54      0.65      3616
```

Confusion matrix

```
[27] print("Confusion matrix:\n%s" % metrics.confusion_matrix(y_testsvmpoly, y_predsvmpoly))

Confusion matrix:
[[347   0   0   0   0   0   0   0 987   0]
 [  6 486   0   0   0   2   0   0 119   0]
 [  0   0 374   0   3   0   0   0  93   0]
 [  0   0   0 330   0   0   0   0  27   0]
 [  0   0   0   0 176   0   0   0 125   0]
 [  0   0   0   0   0 116   0   2  86   0]
 [  0   0   0   0   0   0  64   0 104   0]
 [  0   0   0   0   0   0   0  36  70   0]
 [  2   0   0   0   0   0   6   4  34   0]
 [  6   1   0   0   0   3   0   0   7   0]]
```

Score

```
[28] PolySVM_model.score(polyfeatures,y_testsvmpoly)

0.5428650442477876
```

Figure 30: VGG16+SVM Polynomial with Test set

```
Classification report for classifier SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=100, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=1):
              precision    recall  f1-score   support

           0       0.90      1.00      0.95      1334
           1       1.00      0.96      0.98       613
           2       1.00      0.84      0.91       470
           3       0.95      0.98      0.96       357
           4       0.98      0.62      0.76       301
           5       0.97      0.93      0.95       204
           6       0.92      0.73      0.81       168
           7       0.91      0.99      0.95       106
           8       0.16      0.52      0.24        46
           9       0.00      0.00      0.00        17

    accuracy                           0.91      3616
   macro avg       0.78      0.76      0.75      3616
weighted avg       0.93      0.91      0.92      3616
```

Confusion matrix

```
[36] print("Confusion matrix:\n%s" % metrics.confusion_matrix(y_testsvmRBF, y_predsvmRBF))

Confusion matrix:
[[1333    0    0    0    0    1    0    0    0    0]
 [  20  591    0    0    0    0    2    0    0    0]
 [   0    0  394    0    3    0    0    0   73    0]
 [   6    0    0  350    1    0    0    0    0    0]
 [  80    0    0    0  186    1    0    0   34    0]
 [  14    0    0    0    0  189    0    0    1    0]
 [   6    0    0   19    0    0  122    0   21    0]
 [   0    0    0    0    0    0    0  105    1    0]
 [   3    0    0    0    0    0    8   11   24    0]
 [  13    1    0    0    0    3    0    0    0    0]]
```

Score

```
[37] RBFSVM_model.score(rbffeatures,y_testsvmRBF)

0.9109513274336283
```

Figure 31: VGG16+SVM RBF with Test set

```
Classification report for classifier <tensorflow.python.keras.engine.functional.Functional object
              precision    recall  f1-score   support

           0       0.95      0.88      0.92      1355
           1       1.00      0.98      0.99       611
           2       0.98      0.94      0.96       473
           3       0.98      0.99      0.99       405
           4       0.88      0.59      0.70       249
           5       0.50      1.00      0.67       226
           6       0.92      0.83      0.87       147
           7       0.83      0.81      0.82        83
           8       0.28      0.27      0.28        56
           9       0.00      0.00      0.00        12

    accuracy                           0.89      3617
   macro avg       0.73      0.73      0.72      3617
weighted avg       0.91      0.89      0.89      3617
```

```
[ ] print("Confusion matrix:\n%s" % metrics.confusion_matrix(y_testsm, y_predsm))

Confusion matrix:
[[1197    0    0    0   16  142    0    0    0    0]
 [   0  597    0    0    0    0    3    0    0   11]
 [   0    0  444    0    3    0    1    0   25    0]
 [   3    0    0  401    1    0    0    0    0    0]
 [  46    0    0    0  146   57    0    0    0    0]
 [   0    0    0    0    0  226    0    0    0    0]
 [   0    0    0    8    0    4  122    0   13    0]
 [   0    0    0    0    0   16    0   67    0    0]
 [   4    2    8    0    0    6    7   14   15    0]
 [  10    0    0    0    0    2    0    0    0    0]]
```

Figure 32: VGG16+Softmax with Test set

This evaluation presents the comparison between Softmax, SVM polynomial, and SVM RBF after feature extraction by VGG16. SVM RBF gives the best evaluation, while Softmax is also a good one.

6 Demonstration Application

6.1 Demonstration

Option Menu

- Select "Image" to go to Image Detection Mode.
- Select "Video" to go to Video Detection Mode.

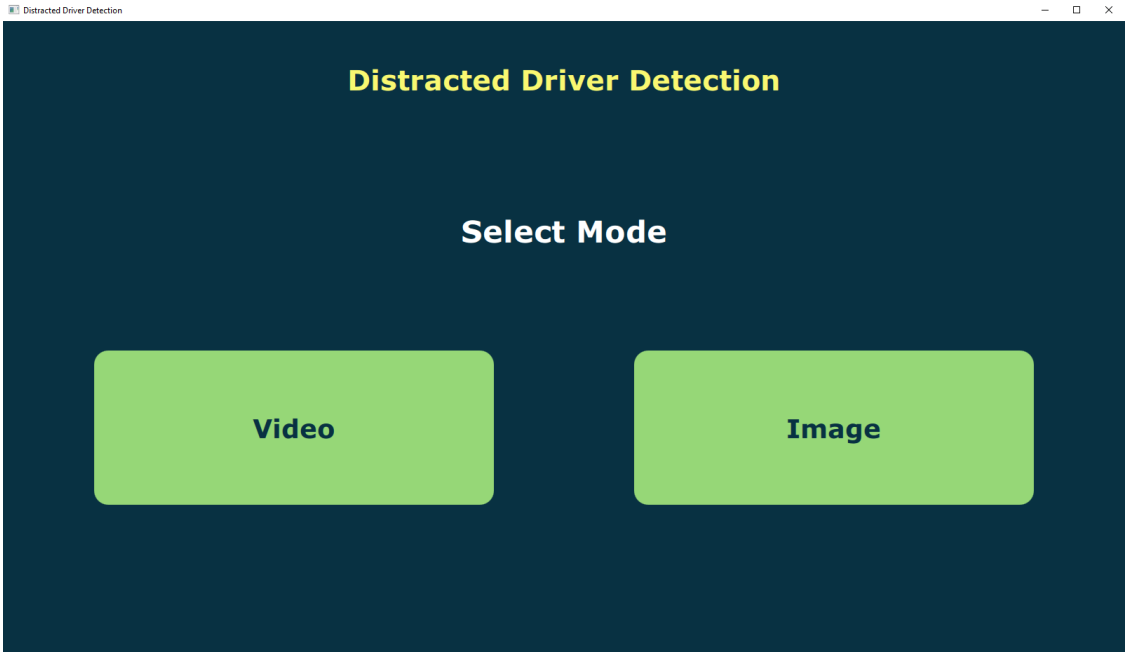

Figure 33: Distracted Driver Detection Option Menu

Image Detection

- Press "Choose Image" and select the image you want to detect.
- Put all images you want to detect in "program_directory/data/testImage" folder and press the "Start" button for automatic detect all the images.

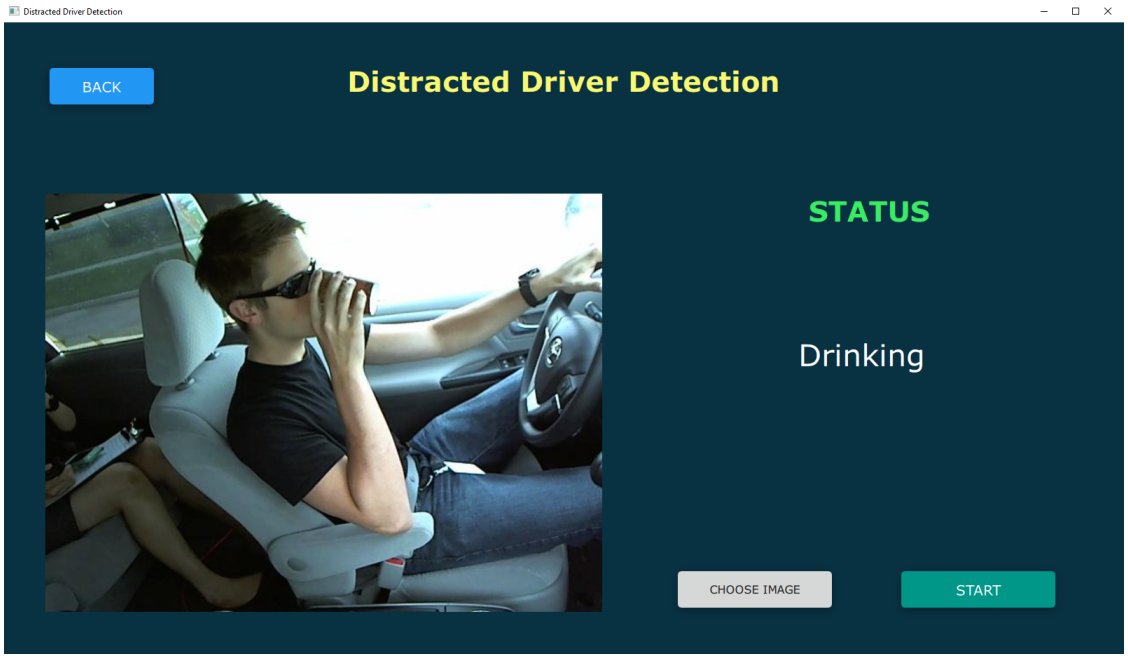

Figure 34: Distracted Driver Detection Image Detection

Video Detection

- Press "Choose Video" and select the video you want to detect.
- The application will extract the video to many image frames and put them in "program_directory/data/testVideo/name_of_video" folder.

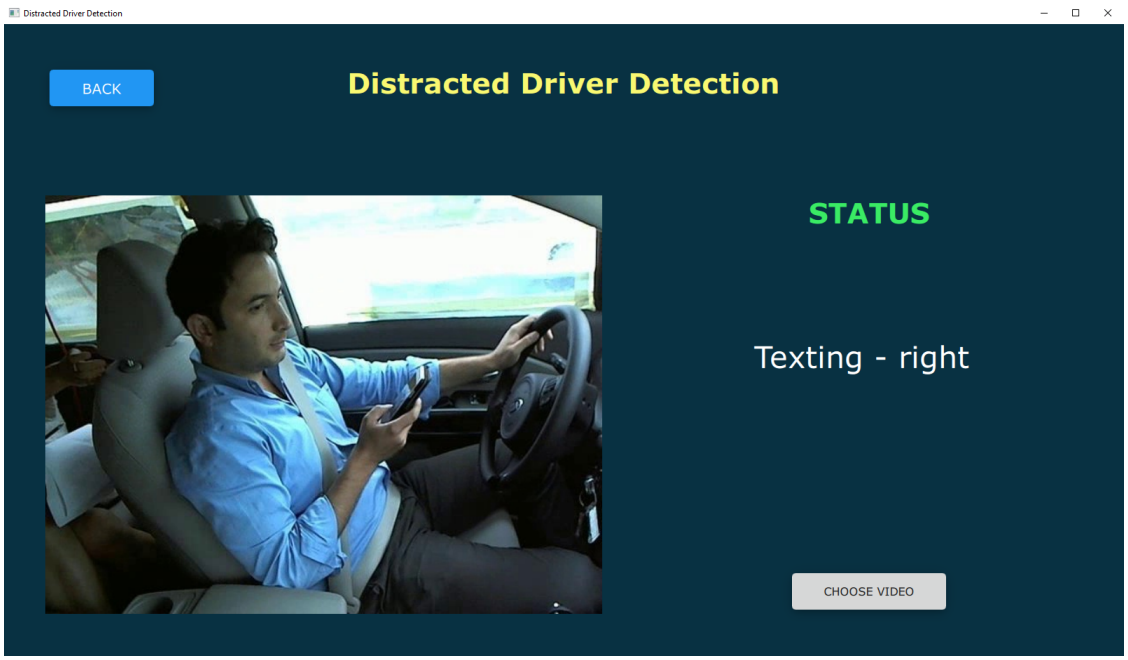

Figure 35: Distracted Driver Detection Video Detection

7 Conclusion

7.1 Conclusion

After redesigning our pipeline, the new approach provides a better result on the validation set. This assignment provides our group some lessons :

- Not only the model but also the data is very important. The good analysis of data helps to choose the better model.
- All features should not be considered equally. Some features are very important, some others can be removed. Therefore, pre-processing techniques play an important role in the procedure.
- When applying AI into real applications, we should consider both time and accuracy.

7.2 Future work

Our work can make better using some interesting idea below :

- Our group can try other state-of-the-art classification method such as : **ResNet** or **MobileNet**
- Another potential idea to improve accuracy is using ensemble learning. In other words, the final result is the major vote of many classification models. However, this method increases the prediction time.
- Some Kaggler suggest using kNN for output weight. This method can work well in the case of video prediction, at some consecutive frames have similar images. Therefore, we can use pixel-loss to update their weight based on kNN

Bibliographies

[1] CDC. *Distracted Driving*. 2016. URL: http://www.cdc.gov/motorvehiclesafety/distracted_driving/ (visited on 09/30/2020).

[2] Theodoros Evgeniou and Massimiliano Pontil. "Support Vector Machines: Theory and Applications". In: vol. 2049. Jan. 2001, pp. 249–257. DOI: 10.1007/3-540-44673-7_12.

[3] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[4] State Farm Insurance. *State Farm Distracted Driver Detection*. 2016. URL: https://www.kaggle.com/c/state-farm-distracted-driver-detection.

[5] Jessica Nguyen. *Distractions a danger on the road*. 2019. URL: https://vneconomictimes.com/article/society/distractions-a-danger-on-the-road (visited on 11/20/2020).

[6] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *International Conference on Learning Representations*. 2015.