

DriveGuard - First Progress

**Submitted in the partial fulfilment of the Degree of Bachelor of
Technology
(Computer Science and Engineering)**

Submitted by

Krishan Gopal, 04115002719,

Harshvardhan Singh, 03515002719,

Pawan Sethi, 04015002719

Under the supervision of

Ms. Jyoti



Department of Computer Science and Engineering

Maharaja Surajmal Institute of Technology Janakpuri, New Delhi.

2019-23

INDEX

Objectives	1.
Progress	2.
Gantt Chart	8.
References	9.

Objectives

- To Detect Driver Drowsiness using Object detection for classification problem.
- To Implement AI Technologies like, OpenCV, Convolution Neural Network (CNN), Tensorflow, Keras etc.
- To Build a feature to identify Vehicle Crash via Motion Sensors
- To create an android app using Kotlin. It will send the detected accident information to given Emergency contact.
- A Feature to send Real-time GPS location of driver and sending in the generated report

Progress Overview

ML model for detecting Drowsiness of driver has been built and trained

1. Visualisation of Data using Matplotlib

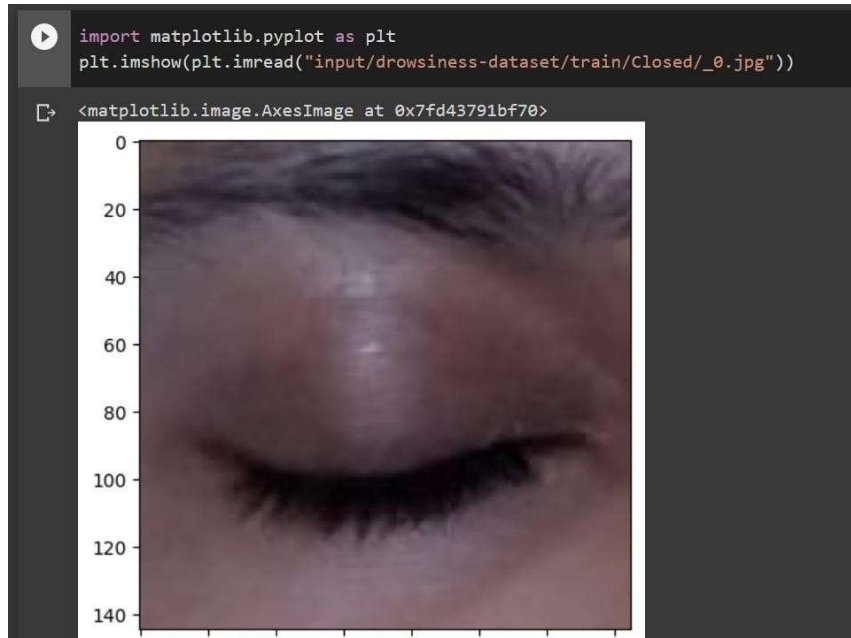


Fig 1 : Data visualization – Closed eye



Fig 2 : Data visualization – Open mouth

2. Pre-processing of data to extract faces from the images, and resize them

```
for yawn and not_yawn. Take only face

[ ] def face_for_yawn(direc="input/drowsiness-dataset/train", face_cas_path="input/prediction-images/haarcascade_frontalface_default.xml"):
    yaw_no = []
    IMG_SIZE = 145
    categories = ["yawn", "no_yawn"]
    for category in categories:
        path_link = os.path.join(direc, category)
        class_num1 = categories.index(category)
        print(class_num1)
        for image in os.listdir(path_link):
            image_array = cv2.imread(os.path.join(path_link, image), cv2.IMREAD_COLOR)
            face_cascade = cv2.CascadeClassifier(face_cas_path)
            faces = face_cascade.detectMultiScale(image_array, 1.3, 5)
            for (x, y, w, h) in faces:
                img = cv2.rectangle(image_array, (x, y), (x+w, y+h), (0, 255, 0), 2)
                roi_color = img[y:y+h, x:x+w]
                resized_array = cv2.resize(roi_color, (IMG_SIZE, IMG_SIZE))
                yaw_no.append([resized_array, class_num1])
    return yaw_no

yawn_no_yawn = face_for_yawn()
```

Fig 3: Pre-processing of data – yawn and no_yawn faces

```
for closed and open eye

[ ] def get_data(dir_path="input/drowsiness-dataset/train/", face_cas="input/prediction-images/haarcascade_frontalface_default.xml", eye_cas=".../input/prv
labels = ['Closed', 'Open']
IMG_SIZE = 145
data = []
for label in labels:
    path = os.path.join(dir_path, label)
    class_num = labels.index(label)
    class_num += 2
    print(class_num)
    for img in os.listdir(path):
        try:
            img_array = cv2.imread(os.path.join(path, img), cv2.IMREAD_COLOR)
            resized_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))
            data.append([resized_array, class_num])
        except Exception as e:
            print(e)
    return data
```

Fig 4 : Pre-processing of data – Closed and Open eyes

3. Giving label encoding to classes, like “yawn” “no_yawn” and splitting of data for testing (30%) and training (70%) purpose

```
train test split

[ ] from sklearn.model_selection import train_test_split
seed = 42
test_size = 0.30
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=seed, test_size=test_size)
```

Fig 5 : Splitting Training and testing data

4. Data Augmentation

- i. Normalize/rescale
- ii. Zoom
- iii. Horizontal flip
- iv. 30% rotation

```
▼ keras version

[ ] import keras
    keras.__version__

'2.12.0'

▼ Data Augmentation

[ ] train_generator = ImageDataGenerator(rescale=1/255, zoom_range=0.2, horizontal_flip=True, rotation_range=30)
    test_generator = ImageDataGenerator(rescale=1/255)

    train_generator = train_generator.flow(np.array(X_train), y_train, shuffle=False)
    test_generator = test_generator.flow(np.array(X_test), y_test, shuffle=False)
```

Fig 6 : Data augmentation

5. Training CNN Model on drowsiness data. Combination of convolutional, maxpooling layer, linear layer and activation functions is used to create a sequential model.

Classes : ["yawn", "no_yawn", "Closed", "Open"]

Loss used - Categorical cross entropy loss

Optimizer – Adam

Accuracy metrics used for calculating accuracy

Model

```
[ ] model = Sequential()

model.add(Conv2D(256, (3, 3), activation="relu", input_shape=X_train.shape[1:]))
model.add(MaxPooling2D(2, 2))

model.add(Conv2D(128, (3, 3), activation="relu"))
model.add(MaxPooling2D(2, 2))

model.add(Conv2D(64, (3, 3), activation="relu"))
model.add(MaxPooling2D(2, 2))

model.add(Conv2D(32, (3, 3), activation="relu"))
model.add(MaxPooling2D(2, 2))

model.add(Flatten())
model.add(Dropout(0.5))

model.add(Dense(64, activation="relu"))
model.add(Dense(4, activation="softmax"))

model.compile(loss="categorical_crossentropy", metrics=["accuracy"], optimizer="adam")
```

Fig 7 : Model Architecture Code

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 143, 143, 256)	7168
max_pooling2d (MaxPooling2D)	(None, 71, 71, 256)	0
conv2d_1 (Conv2D)	(None, 69, 69, 128)	295040
max_pooling2d_1 (MaxPooling2D)	(None, 34, 34, 128)	0
conv2d_2 (Conv2D)	(None, 32, 32, 64)	73792
max_pooling2d_2 (MaxPooling2D)	(None, 16, 16, 64)	0
conv2d_3 (Conv2D)	(None, 14, 14, 32)	18464
max_pooling2d_3 (MaxPooling2D)	(None, 7, 7, 32)	0
flatten (Flatten)	(None, 1568)	0
dropout (Dropout)	(None, 1568)	0

Fig 8 : Model Architecture Output

```

history = model.fit(train_generator, epochs=10, validation_data=test_generator, shuffle=True, validation_steps=len(test_generator))

Epoch 1/10
43/43 [=====] - ETA: 0s - loss: 0.6250 - accuracy: 0.7535 WARNING:tensorflow:AutoGraph could not transform <function Model.make_test_function.<locals>.test_function at 0x7fd3a07f1f70>. Note that function
Cause: Unable to locate the source code of <function Model.make_test_function.<locals>.test_function at 0x7fd3a07f1f70>. Note that function
To silence this warning, decorate the function with @tf.autograph.experimental.do_not_convert
WARNING: AutoGraph could not transform <function Model.make_test_function.<locals>.test_function at 0x7fd3a07f1f70> and will run it as-is.
Cause: Unable to locate the source code of <function Model.make_test_function.<locals>.test_function at 0x7fd3a07f1f70>. Note that function
To silence this warning, decorate the function with @tf.autograph.experimental.do_not_convert
43/43 [=====] - 537s 12s/step - loss: 0.6250 - accuracy: 0.7535 - val_loss: 0.3812 - val_accuracy: 0.8356
Epoch 2/10
43/43 [=====] - 553s 13s/step - loss: 0.4032 - accuracy: 0.8315 - val_loss: 0.3204 - val_accuracy: 0.8512
Epoch 3/10
43/43 [=====] - 525s 12s/step - loss: 0.3344 - accuracy: 0.8530 - val_loss: 0.2521 - val_accuracy: 0.8927
Epoch 4/10
43/43 [=====] - 512s 12s/step - loss: 0.3093 - accuracy: 0.8723 - val_loss: 0.2019 - val_accuracy: 0.9135
Epoch 5/10
43/43 [=====] - 491s 11s/step - loss: 0.3104 - accuracy: 0.8797 - val_loss: 0.2910 - val_accuracy: 0.8841
Epoch 6/10
43/43 [=====] - 508s 12s/step - loss: 0.2642 - accuracy: 0.8849 - val_loss: 0.1990 - val_accuracy: 0.9014
Epoch 7/10
43/43 [=====] - 507s 12s/step - loss: 0.2702 - accuracy: 0.8864 - val_loss: 0.1746 - val_accuracy: 0.9291
Epoch 8/10
43/43 [=====] - 474s 11s/step - loss: 0.2378 - accuracy: 0.8953 - val_loss: 0.1595 - val_accuracy: 0.9360
Epoch 9/10
43/43 [=====] - 529s 12s/step - loss: 0.2454 - accuracy: 0.9013 - val_loss: 0.1654 - val_accuracy: 0.9377
Epoch 10/10

```

Fig 9 : Loss and Accuracy Output

6. Accuracy Testing using Accuracy metrics. Plot accuracy and loss with matplotlib against epoch.

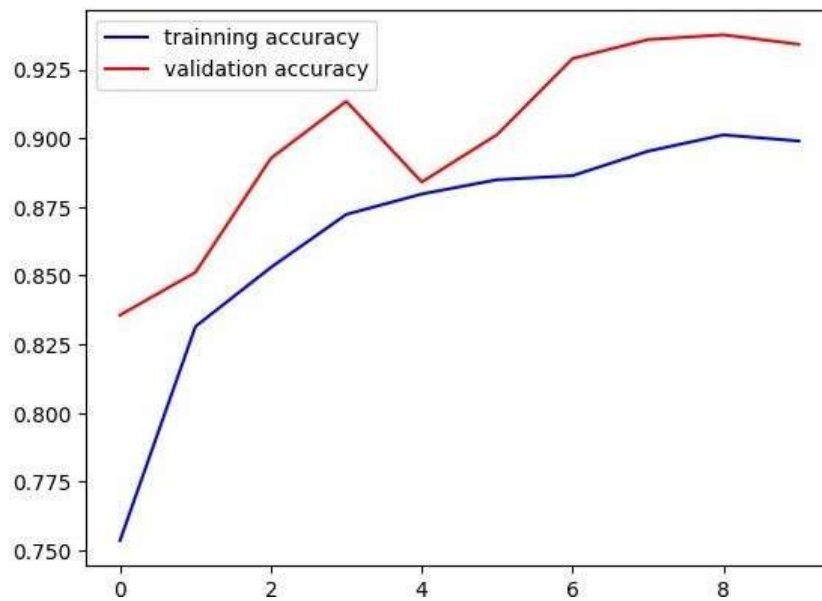


Fig 10 : Epoch Vs Accuracy Graph

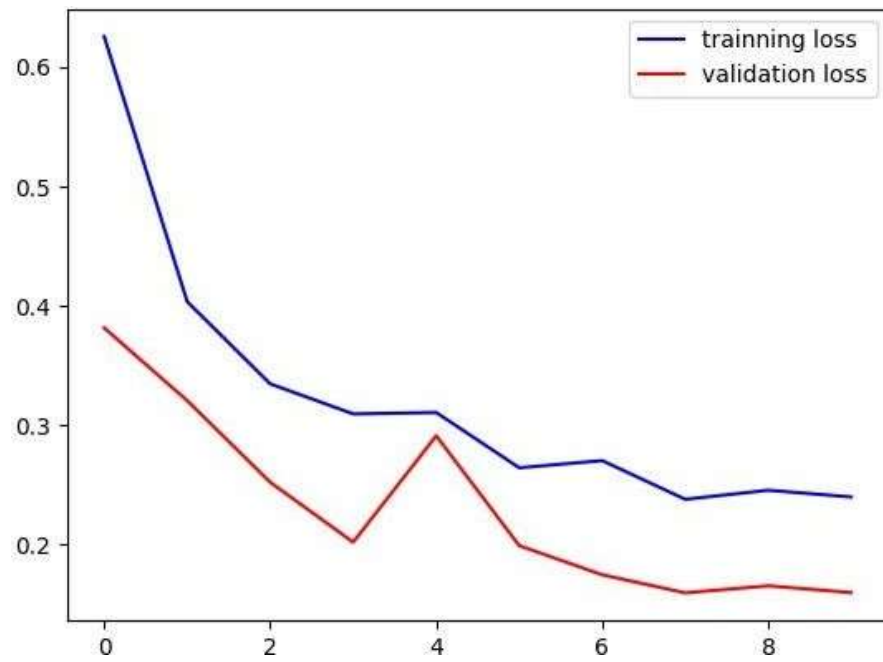
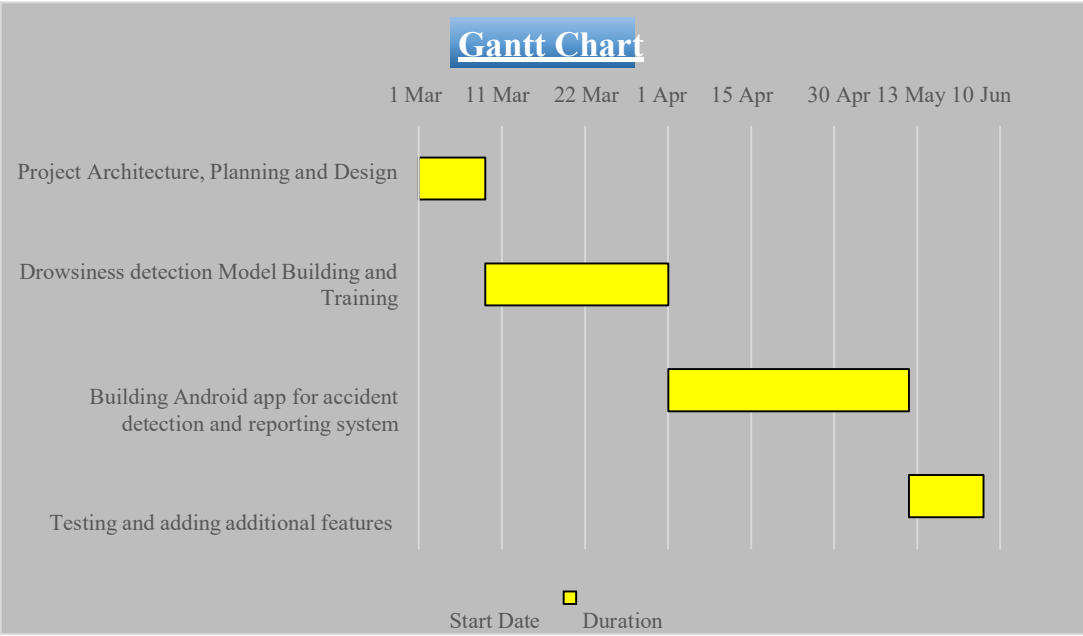


Fig1 : Epoch Vs Loss Graph



References

- [1]. Rajkumarsingh, B., & Totah, D.. (2021). Drowsiness Detection using Android Application and Mobile Vision Face API. *R&D Journal*, 37, 26-34.
- [2]. Prajwal Bs. (2022), "Driver Drowsiness Detection", NMAM Institute of Technology, Research Gate.
- [3]. Mishra, A., & Shukla, A. (2019). A survey on driver drowsiness detection using physiological, vehicular, and behavioral approaches. Proceedings of the International Conference on Advanced Computational and Communication Paradigms (ICACCP), 1-6. doi: 10.1109/ICACCP46912.2019.8943222
- [4]. V B Navya Kiran, Raksha R, Anisoor Rahman, Varsha K N, Dr. Nagamani N P, 2020, Driver Drowsiness Detection, INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) NCAIT – 2020 (Volume 8 – Issue 15)
- [5]. Hashemi, M., Mirrashid, A. & Beheshti Shirazi, A. Driver Safety Development: Real-Time Driver Drowsiness Detection System Based on Convolutional Neural Network. *SN COMPUT. SCI.* 1, 289 (2020).
- [6]. Jabbar, Rateb & Al-Khalifa, Khalifa & Kharbeche, Mohamed & Alhajyaseen, Wael & Jafari, Mohsen & Jiang, Shan. (2018). Real-time Driver Drowsiness Detection for Android Application Using Deep Neural Networks Techniques. *Procedia Computer Science*. 130. 400-407.10.1016/j.procs.2018.04.060.
- [7]. K. Satish, A. Lalitesh, K. Bhargavi, M. S. Prem and T. Anjali., "Driver Drowsiness Detection," 2020 International Conference on Communication and Signal Processing (ICCSPP), 2020, pp. 0380-0384.
- [8]. Jo, Sang-Ho et al. "Heart Rate Change While Drowsy Driving." *Journal of Korean medical science* vol. 34,8 e56. 12 Feb. 2019, doi:10.3346/jkms.2019.34.e56
- [9]. R. Jabbar, M. Shinoy, M. Kharbeche, K. Al-Khalifa, M. Krichen and K. Barkaoui, "Driver Drowsiness Detection Model Using Convolutional Neural Networks Techniques for Android Application," 2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIOT), 2020.
- [10]. Girish, A. Kumar, A. Kumar and A. M, "Driver Fatigue Detection," 2020 IEEE 17th India Council International Conference (INDICON), 2020, pp. 1-6, doi: 10.1109/INDICON49873.2020.9342456.