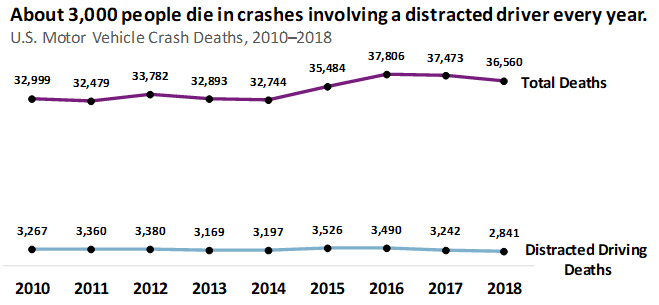
**Distracted driving statistics**

Why is distracted driving a problem?

Every type of distracted driving increases your risk of a car crash, injury, and even death. For example, reaching for an object increases a driver's risk of crashing by 800%.



**Intro/overview**

Due to improvement in cv field new approaches and datasets appeared to overcome this problem by developing more advanced systems , most these systems were trained and tested and applied on closed source data sets , but there’s always an important question , what would happen if the system faced a new class that it wasn’t trained on before , especially that it’s a case that could always happen in production since we can’t predict human behavior to a limited count of actions

That why we used the DAD dataset for this problem along with a model based on contrastive learning approach to detect the normal and the distracted driving

* Mention the paper we worked on (Driver Anomaly Detection: A Dataset and Contrastive Learning Approach , Technical University of Munich)
* Mention the accuracy (0.9673 AUC)

**Related works**

**Detecting Driver Distraction Using Deep-Learning Approach**

* Numerous papers worked with numerous approaches on our problem starting from the AUC and the state farm datasets
* the paper which used a modified CNN model VGG16 to detect distracted drivers and achieved 96.95% accuracy  on the state farm closed dataset

Diagram

Description automatically generated

[This Photo](https://www.freesion.com/article/79931367117/) by Unknown author is licensed under [CC BY-SA](https://creativecommons.org/licenses/by-sa/3.0/).

**Driver Distraction Identification with an Ensemble of Convolutional Neural Networks**

* In this paper, they proposed a deep learning-based solution that achieves a 90% accuracy.
* They also studied the effect of different visual elements in distraction detection by means of face and hand localizations, and skin segmentation.
* Finally, they  could achieve 84.64% classification accuracy and operate in a real-time environment.

**Driver Anomaly Detection: A Dataset and Contrastive Learning Approach**

* The paper that we worked on is one of the recent works using contrastive learning , it was published in 2020 , they achieved 0.9673 auc on a large data set.

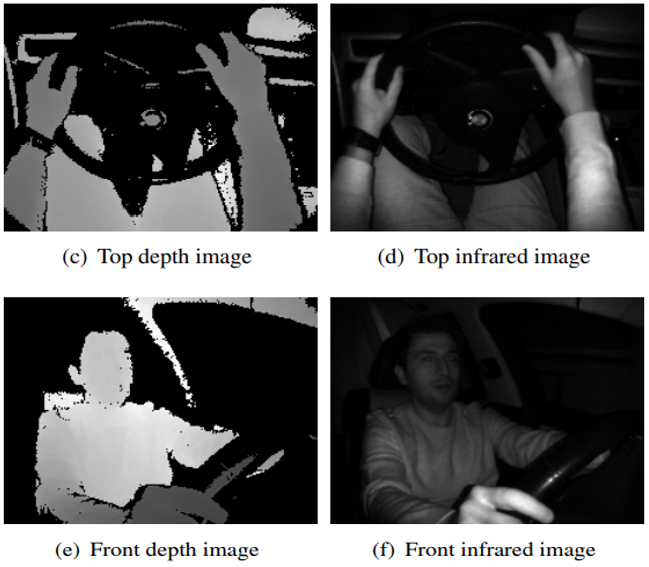
**Data Sets**

**As for datasets we had 2 different datasets**

* **Distracted Driver dataset ( Closed )**
* **Driver Anomaly Detection dataset ( Open )**
* **Distracted Driver Dataset**
* They collected the dataset from scratch as the other datasets weren’t suitable for their purpose like State Farm’s dataset which was used for their Kaggle competition purposes only and for Southeast University
* And (SEU) dataset was that it presents only four distraction postures.



* Distracted Driver dataset contains 10 classes which are 1 class for normal driving and all other classes for distracted driving.
* The data was collected in a video format and, then, cut into individual images, 640 × 480 pixels for each image. The cameras are fixed using an arm strap to the car roof handle on the side view of passenger’s seat.
* **Driver Anomaly Detection dataset Data set**

****

* The DAD dataset is large enough to train Deep Neural Network architectures from scratch.
* The DAD dataset is multi-modal containing depth and infrared and it is multi-view containing front and top views.
* For the dataset recording, 31 subjects were asked to drive in a computer game performing either normal driving or anomalous driving.
* Each subject belongs to either a training or test set. The training set contains recordings of 25 subjects and each subject has 6 normal driving and 8 anomalous driving video recordings.

Text

Description automatically generated

* In the training, we have 8 anomaly classes like Drinking and adjusting radio.
* In the testing, we have the same 8 anomalies in addition to the new 16 anomalies like adjusting the side mirror, adjusting clothes, eating, and reading.
* We faced a problem during experimenting on this data since it's big
* 20 epochs on one view of the data took more than three days to run
* The results weren't good enough of course so we had to switch to another dataset which is the Distracted Driver dataset

**Methodology**

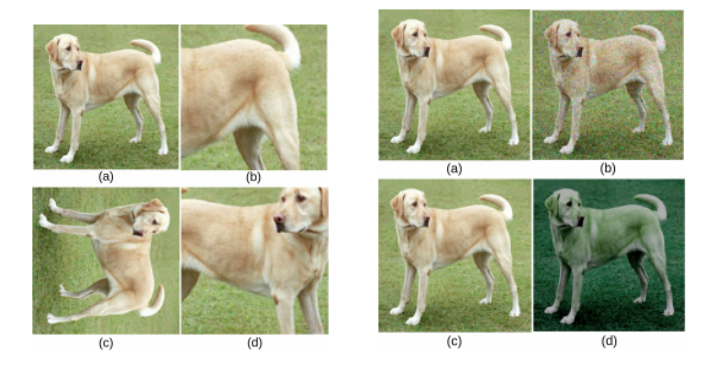
* First Approach - Contrastive Learning :

The architecture of Contrastive Learning basically consists of :

* Data Augmentation
* Base Encoder
* Projection Head
* Contrastive Loss

**data augmentation**

* In contrastive learning, the data augmentation pipeline has a secondary goal which is to generate the anchor, positive and negative examples that will be fed to the encoder and will be used for extracting representations.

****

**Encoder and Projection Head**

* to create *vector representations* for each image most of the approaches use ResNets of various widths and depths. The goal is to train the model to output similar representations for similar images.
* The output of the CNN is then inputted to a set of Dense Layers called the projection head, *z = g(h)*to transform the data into another space. This extra step is empirically shown to improve performance.
* The output of the projection head is normalized

**Custom Image Generator**

* In implementation we used tensorflow ,we used image generator to read the data which has the flow\_from\_dataframe function , it takes a data frame with image’s path along with it’s label as input
* It divides the data into batches in which we decide the size of the batch , this is morepractical for the memory , instead of processing on the whole dataset at once so we could face memory issues .
* The image generator does the data augmentation
* To summarize , it reads the data , does augmentation and feed it to the model

**Why did we use customized data generator ?**

* We wanted to shuffle the data using the generator to randomize our dataset
* But lead to a problem , which is the fact that some batches had zero normal examples
* also for the loss function to work properly we needed to have the same ratio between the normal and abnormal examples in each batch so the model will be stable

**Contrastive Loss**

* Impose that normalized embeddings from the normal driving class are closer together than embeddings from different anomalous action classes.
* maximize the similarity between normal driving samples and minimizing the similarity between normal driving and anomalous driving Samples
* we have K normal driving images and M anomalous driving images in each mini-batch.
* Form positive pairs & negative pairs
* There are in total K(K−1) positive pairs and KM negative pairs in every mini-batch.
* Use cosine similarity

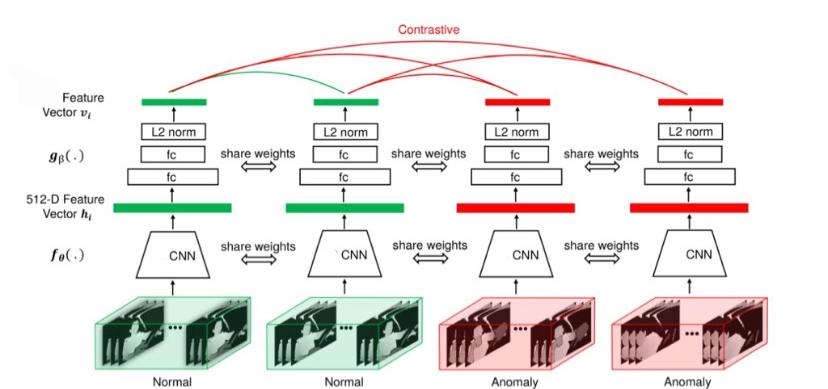
Text, letter

Description automatically generated

**Custom Loss Function**

* Loss functions in Tensorflow
* N-Pair Loss
* Triplet Loss

**Architecture Summary**



**Testing Specifications**

* At testing time we can use different evaluation protocols
  + K Nearest Neighbors (KNN):
    - Store the feature vectors for all training-set images
    - At testing: measure the distance between the feature vector of the tested image and the feature vectors of all training Data.
  + Compact Normal Driving Representation:
    - we remove the projection head and use the output of the encoder
    - We normalize the output of the encoder
    - Then we Generate the compact feature vector of the normal examples
    - During testing time we compare the normal vector with the input image vector and decide whether it’s normal or distracted based on a threshold

**Training Details For Contrastive Training**

* For training phase in contrastive learning we used learning rate 0.01 with learning rate decay 0.9 and batch size 64.
* We did several data augmentation techniques which are :  Rescale ,Rotation , Channel shift range and  Horizontal flip
* We used SGD as an optimizer
* It trained for 120 epochs and achieved 92% auc .
* We tried several models that'll be shown in the results
* **Second approach - Classification**
* We have 2000 images which are distributed among 10 classes, 200 image per class .

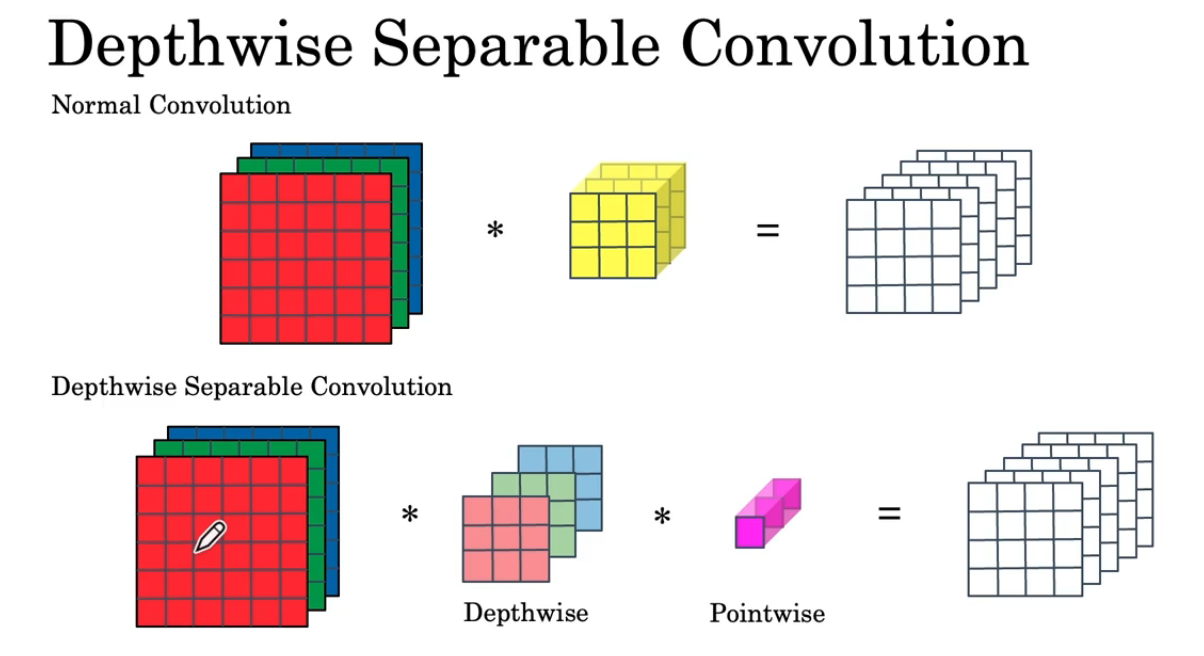
2 approaches for training:

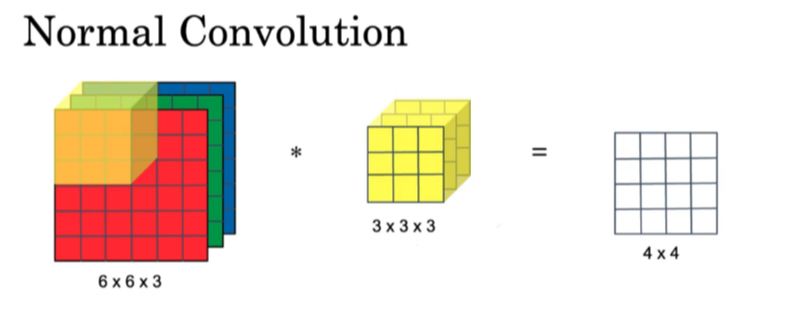
* Custom CNN Model
* Pretrained Model on Imagenet
* Image classification involves the extraction of features from the image to observe some patterns in the dataset.
* In our experiments our best results was 91% using MobileNet

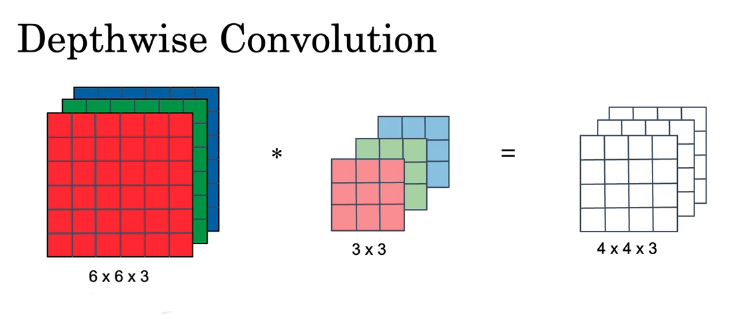
MobileNet

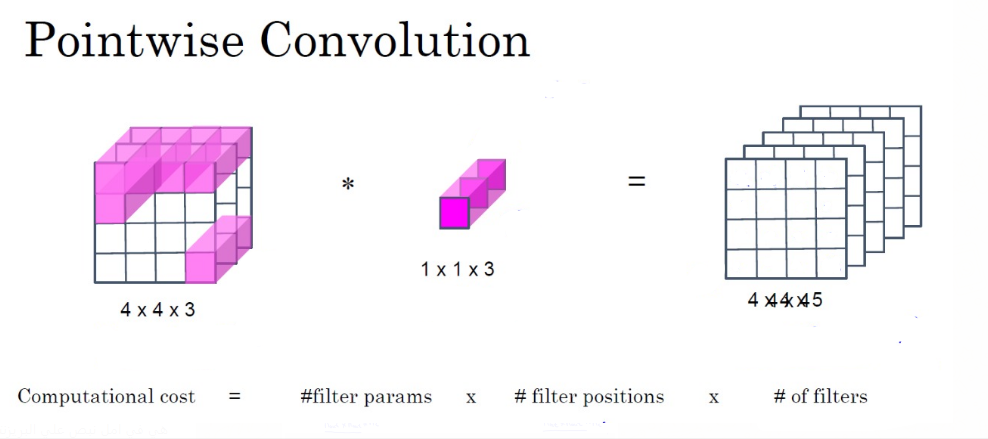
* The network was designed to maximize accuracy with restricted resources for an on-device or embedded application. Mobile Nets are small, low-power models parameterized to meet the resources constraints.
* MobileNet uses depthwise separable convolutions which are based on two operations :

1. Depthwise convolution.
2. Pointwise convolution.









**Cost Comparison**

* Computation cost = #filter parameters x #filter positions x #filters
* Cost of normal convolution = 3x3x3 \* 4x4 \* 5 = 2160
* Cost of Depthwise Separable Convolution = Depthwise cost + Pointwise

= 3x3 \* 4x4 \* 5 + 1x1x3 \* 4x4 \* 5 = 672

* Ratio of Cost Reduction = 1 - ( 672 / 2160 ) = 0.69

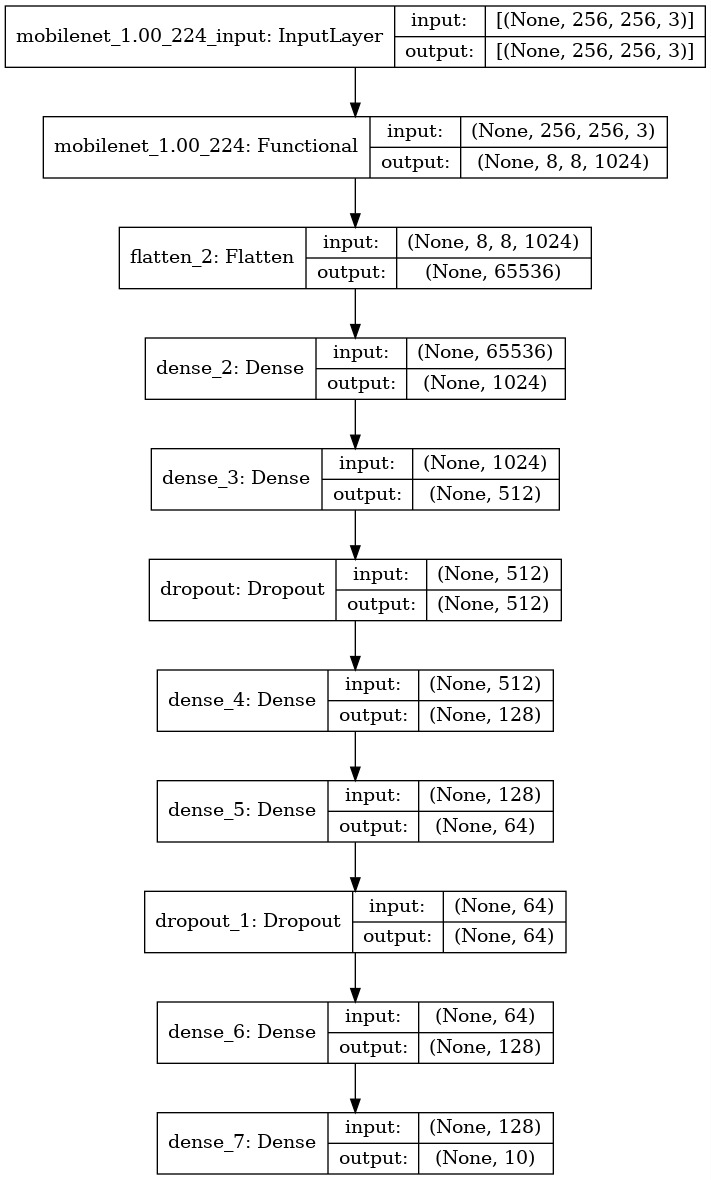
**Loss Function**

* Categorical cross entropy :

Also called Softmax Loss. It is a Softmax activation plus a Cross-Entropy loss. If we use this loss, we will train a CNN to output a probability over the CC classes for each image. It is used for multi-class classification.

Diagram

Description automatically generated



**Training Details**

**Classification training**

* For the Model training in classification we used transfer learning using ImageNet , with multiple models like MobileNet , ResNet50 and InceptionV3 .
* Hence, our dataset has a few images so the model did not generalize well on unseen data due to overfitting. The highest accuracy on test data was 0.91 after implementing different techniques such as Data augmentation,L1 /L2 Regularization and Dropout.
* The Data augmentation techniques were : Rotation Range ,Width Shift Range ,Height Shift Range ,Shear Range ,Zoom Range

**Evaluation**

**ROC/AUC**

* We majorly used Area Under Curve to evaluate our model
* For each possible threshold, the ROC curve plots the False positive rate versus the true positive rate.
* False Positive Rate: Fraction of negative instances that are incorrectly classified as positive.
* True Positive Rate: Fraction of positive instances that are correctly predicted as positive.

Chart, pie chart

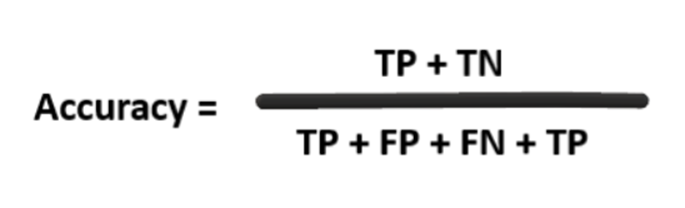
Description automatically generated

**Recall (Sensitivity or True positive rate):**



**Accuracy**

* In our case the accuracy wasn't the best option to evaluate our models since there's class imbalance since we have 9 classes abnormal and 1 normal , so even if the model classified all the images as abnormal it will still get 90% accuracy .



**Results**

**Contrastive results**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Epochs** | **Input shape** | **From scratch or**  **Image Net** | **Out dimensions** | **Batch size** | **Dense layers** | **Conv and**  **Avg pooling** | **Learning Rate** | **threshold** | **AUC** | **accuracy** | **F1** |
| **50** | **256** | **Image net** | **256** | **64** | **512** | **512 (1x1)**  **Avg** | **0.01**  **With decay** | **0.5** | **0.92** | **0.94** | **0.91** |
| **50** | **128** | **Image net** | **128** | **128** | **512** | **512(1x1)**  **Max** | **0.1**  **With decay** | **0.7** | **0.75** | **0.80** | **0.82** |
| **50** | **128** | **Image net** | **128** | **32** | **256** | **512(3x3)**  **Avg** | **0.01**  **With decay** | **0.65** | **0.84** | **0.85** | **0.88** |
| **50** | **256** | **Image net** | **128** | **16** | **256** | **256(3x3)**  **Max** | **0.01**  **With decay** | **0.6** | **0.80** | **0.94** | **0.93** |

**Final Contrastive Model**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Network**  **type** | **Epochs** | **Input shape** | **From scratch or**  **Image Net** | **Out dimensions** | **Batch size** | **Dense layers** | **Conv and**  **Avg pooling** | **Learning Rate** | **threshold** | **AUC** | **accuracy** |
| **MobileNet** | **120** | **256** | **Image net** | **256** | **64** | **512** | **512 (1x1)**  **Avg** | **0.01**  **With decay** | **0.5** | **0.92** | **0.93** |

**A picture containing graphical user interface

Description automatically generated**

**Classification results**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Network**  **type** | **Epochs** | **Input shape** | **From scratch or image net** | **Batch size** | **optimizer** | **Activation function** | **Accuracy**  **On train** | **Accuracy**  **On test** |
| **MobileNet** | **20** | **256** | **Image net** | **32** | **SGD** | **RELU** | **0.97** | **0.91** |
| **MobileNet** | **30** | **256** | **Image net** | **32** | **Adam** | **LRELU** | **0.95** | **0.85** |
| **MobileNet** | **20** | **224** | **Image net** | **16** | **Adam** | **RELU** | **0.94** | **0.86** |
| **ResNet50** | **20** | **224** | **Image net** | **32** | **Adam** | **RELU** | **0.97** | **0.82** |
| **ResNet50** | **20** | **224** | **Image net** | **32** | **SGD** | **LRELU** | **0.98** | **0.84** |
| **InceptionV3** | **20** | **256** | **Image net** | **16** | **SGD** | **RELU** | **0.98** | **0.79** |

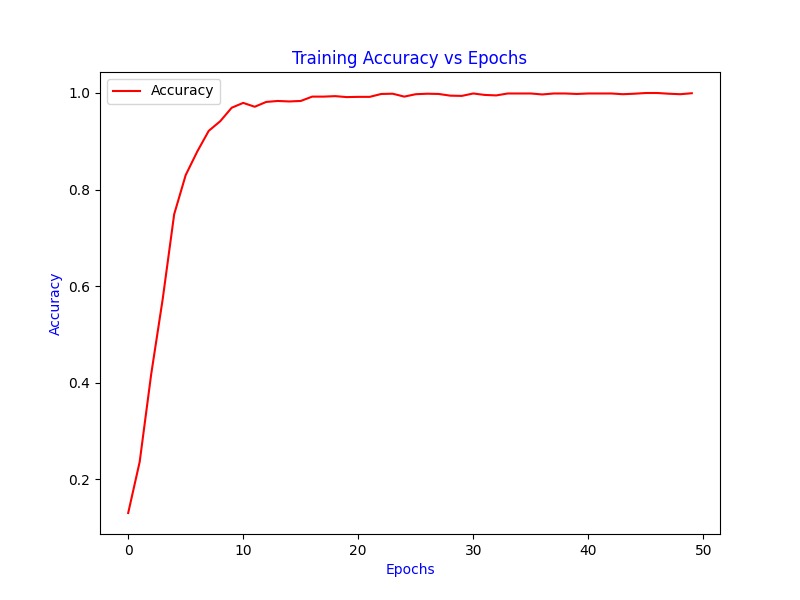
**Final Classification Model**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Network**  **type** | **Epochs** | **Input shape** | **From scratch/ image net** | **Batch size** | **optimizer** | **Activation function** | **Accuracy**  **On train** | **Accuracy**  **On test** |
| **MobileNet** | **50** | **256** | **Image net** | **32** | **SGD** | **RELU** | **0.97** | **0.91** |

**Timeline

Description automatically generated**

**Chart, line chart

Description automatically generated**

**Comparison**

|  |  |  |
| --- | --- | --- |
|  | **Classification** | **Contrastive learning** |
| **Epoch** | **50** | **120** |
| **Accuracy** | **0.91** | **0.92** |
| **Network-Type** | **MobileNet** | **MobileNet** |
| **Input Shape** | **256** | **256** |
| **Batch Size** | **32** | **64** |
| **ImageNet of from scratch** | **ImageNet** | **ImageNet** |
| **Learning Rate** | **0.01** | **0.01 with decay** |
| **Conv and pooling layers** | **(1024),(512) Max** | **512 (1x1) Avg** |

**Deployment**

* We made an API that takes a list of images and it returns predictions either Normal(0) or anomaly(1) and the output will be as JSON.
* Any application after that can use our API, either the application is a camera in a car or in Traffic violations.

**Conclusion & Future work**

* Driver distraction is a serious problem leading to a lot of road crashes worldwide. Hence detection of distracted driver becomes an essential system component is self-driving cars. Here, we present a Contrastive learning approach to detect anomalies behaviors  and also identify the cause of distraction, and Classification approach. We modify the MobileNet architecture for the two particular tasks and apply several regularization techniques to prevent overfitting to the training data.
* The future work for contrastive approach will be as follows if a car has a camera taking pics for the driver so based on that, the system will alert the driver if he did any anomaly action regardless of how was the distraction behavior happened
* The future work for the classification approach is if we managed to collect a dataset for drivers with their plate numbers, we can build a traffic model that assigns violations automatically based on drivers' behavior.