Emergency Vehicle Identification Report.

Introduction

The goal of this task is to quickly and accurately detect emergency vehicles (e.g., ambulances, police cars, fire trucks) to allow for immediate clearance on roads and prioritized response. The data set used to develop the CNN (Convolutional Neural Network) model was gotten from kaggle.

Problem Statement

Developing and training a CNN involves several challenges, which can arise from data, computational constraints, or model specific issues.

Some major challenges faced are:

1. Insufficient or Imbalanced Data.

CNN requires large amounts of labelled data to learn effectively. Imbalanced datasets can cause bias prediction.

1. Data Quality.

Poor data quality e.g., noisy, blurry, can degrade model performance.

1. Computational Challenges.

CNNs can take long time to train on large datasets, making iterative experimentation slow.

1. Model challenges.

Finding the right values for learning rate, batch size, number of filters and layers in hyperparameter tuning is complex.

Review Work Summary.

Dulari Bhatt et al 2021(CNN variants for computer vision: History, architecture, application, challenges and future scope), discussed the choice of hyper-parameters as a challenge in CNN development, as it has a significant impact on the CNN performance. A slight change can influence the overall performance of a CNN in hyper- parameter values. As a result, selecting hyper-parameters with care is a critical design issue that must be addressed using an appropriate optimization technique.

Architecture of CNN model used.

The CNN model was built by using the sequential model from keras library. Three layers were added for development, with 2D convolutional layer (Conv2D), filters of 32, kernel size of 3x3, the input shape (128, 128, 3) and activation function ‘relu’. Max-Pooling 2D was added at each layer with a pool size of (2, 2). After the convolutional and pooling layers have been added, the flatten layer was added. Flatten layer which is used to transform the multidimensional data output from the convolutional and pooling layers into a 1D vector.

A dense layer of 128 unit was added next, with an activation function of ‘relu’. The dense layer is simple layer of neurons in which each neuron receives input from all the neurons from previous layer. It used to classify image based on output from convolution layers (Govinda Dumane, Published in Towards Data Science Mar 2, 2020). 128 unit was chosen because it provides a good balance between model complexity and computational efficiency, it is not too small to miss patterns and not too large to overfit or make the model unnecessarily slow.

A dropout rate of 50% (0.5) was set to the model. Dropout is a regularization technique to prevent overfitting. Overfitting occurs when a model performs well on training data but fails to generalize to new, unseen data (Dropout in Convolution neural Networks (CNN) a detailed explanation, Vishnuam Oct 7, 2024). Finally, the output dense layer with a single neuron and the activation function ‘sigmoid’ for binary classification. The binary classification was used because there are only two possible outcomes (emergency or nonemergency). The sigmoid activation function outputs a value between 0 and 1, representing the probability of one class. if the output is closer to 1 it predicts emergency and closer to 0 it predicts non-emergency.

The SDG (Stochastic Gradient Descent) was imported from tensorflow.keras to compile the CNN model as optimizer with a learning rate of 0.01, momentum of 0.9, the loss function was binary cross-entropy since it is a binary classification, and the metrics was accuracy. The model was trained for 20 epochs, using batches of randomly transformed images generated from the training data, with a batch size of 32.

Regularisation method used.

Batch normalization is the regularisation method adopted, which involves normalising the outputs from a layer such that they have a mean of zero and a standard deviation of 1. Using Max Pooling with 5x5 kernal size without regularisation precision was 0.77 precision for non-emergency vehicles from the classification report, the model recalled 0.94 with af1 score of 0.85. This means the model performed well in recognising non-emergency vehicle. While for emergency vehicle the model had a 0.88 precision, but poor in recalling with a 0.63, and had a f1 score of 0.73.

The confusion metrics suggests that the model performed well in recognising non-emergency vehicle with true negatives of 178, and false positives of 12.

And for emergency vehicles it has a true negative of 88 and false positive of 52.

A screenshot of a computer

Description automatically generated

*Classification report for CNN, with max-pooling 5x5 kernal size, without regularization*.

*Below is the confusion metrics.*

A diagram of a diagram

Description automatically generated with medium confidence

After introducing batch normalisation as regularisation method, the model performed slightly better having a precision of 0.9 0for non-emergency vehicle, but could only recall an average of 0.72, with a f1 score of 0.80. While for the emergency vehicle, the model had a precision of 0.70 and recall significantly better with a 0.89 average and f1 score of 0.79. The confusion metrics suggests that the model had correct predictions for non-emergency vehicles of 137 and wrongly selected 53 as non-emergency. While it improved in recognising emergency vehicles with an average of 125 and wrongly selected 15 vehicles as emergency.

This shows the addition of regularisation improved the model, although it could do better as it fluctuates during validation accuracy, while training accuracy steadily improves. This fluctuation in the model shows it slightly overfits but still performs reasonably well.

A number of numbers in a row

Description automatically generated with medium confidence

*Above is the classification report after regularisation.*

A diagram of a diagram

Description automatically generated with medium confidence

*Above is the improvement in the model after regularisation has been added.*

Other Hyperparameter Tuning used.

Other hyperparameter tuning used to optimise the model was the batch size. The batch size which is simply the number of images used to train a single forward and backward pass (Ibrahem Kandel, Mauro Castelli, The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset, [https://www.sciencedirect.com/science/article/pii/S2405959519303455#:~:t ext=Many%20hyperparameters%20have%20to%20be,single%20forward%20 and%20backward%20pass)](https://www.sciencedirect.com/science/article/pii/S2405959519303455#:~:text=Many%20hyperparameters%20have%20to%20be,single%20forward%20and%20backward%20pass).

Training the CNN model with a Conv2D, filter 32 and a kernal size of 3x3, with a max-pooling size of 2, 2. The batch size used was 32 and trained for 20 epoch. The Classification report had a precision for non-emergency vehicles, an average of 0.78, and could recall 0.91, with a f1 score of 0.84, while emergency vehicles had a precision of 0.83, recall of 0.65 and a f1 score of 0.73.

The confusion metrics suggests that the model could identify non-emergency vehicles a score of 172 and had an incorrect prediction of 18 data. While for emergency vehicles it had a wrong prediction of 49 data and had 91 correct.

*Below is the graph showing the training and validation loss of the model.*

A graph of a graph with blue and orange lines

Description automatically generated

*The graph showing the training and validation accuracy of the CNN model*

A graph of a line graph

Description automatically generated with medium confidence

The model trained well; a steady increase shows the model is learning patterns from the training data.

The classification report for the model showing how it learned and was able to classify the data given.

A number of numbers in a row

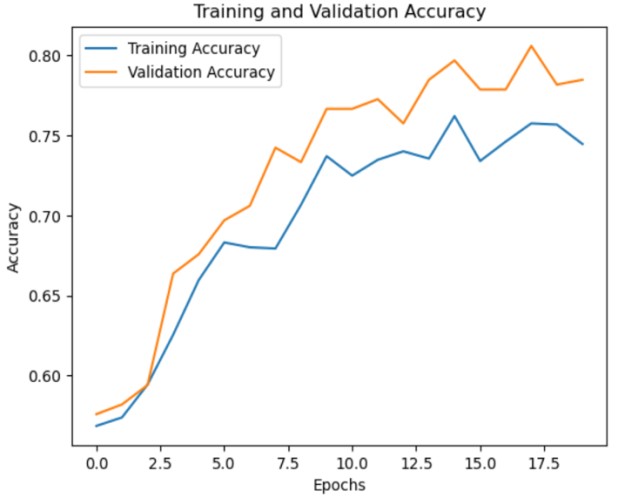
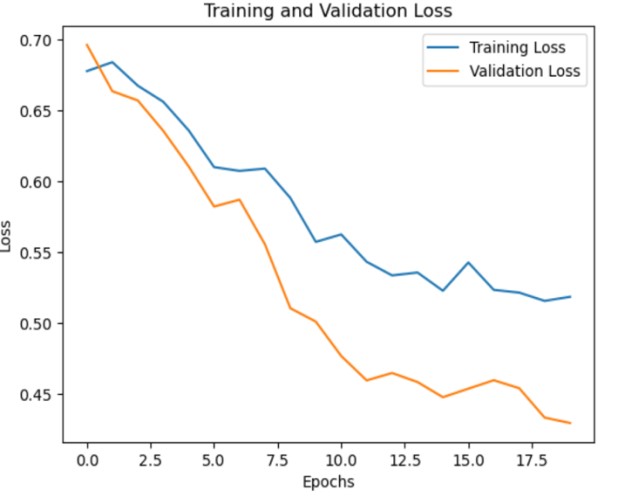
Description automatically generated with medium confidence

*The confusion metrics showing how batch size 32 was able to classify the data it got*

A diagram of a confusion matrix

Description automatically generated

Hyperparameter tuning with a batch size of 128 gave a significantly improved model after training.



The plots above show how the CNN model performed during training process, no strong evidence of over-fitting. And the model validation accuracy follows closely to the training accuracy.

A number of numbers in a row

Description automatically generated with medium confidence

Above is the classification report. It shows using batch size 128 tuning, the model improved in recalling the emergency vehicles and has a f1 score of 0.74. Also, the precision in non-emergency vehicles improved with an average of 0.81.

The confusion metrics showing the model improved significantly with a batch size of 128, given that it could detect 156 non- emergency vehicles with wrong choice of 34 and improved in identifying emergency vehicles with 103 positives and negatives of 37.

The confusion metric graph below shows how hyperparameter tunning with a batch size of 128 optimized the model and having a strong effect in its performance.

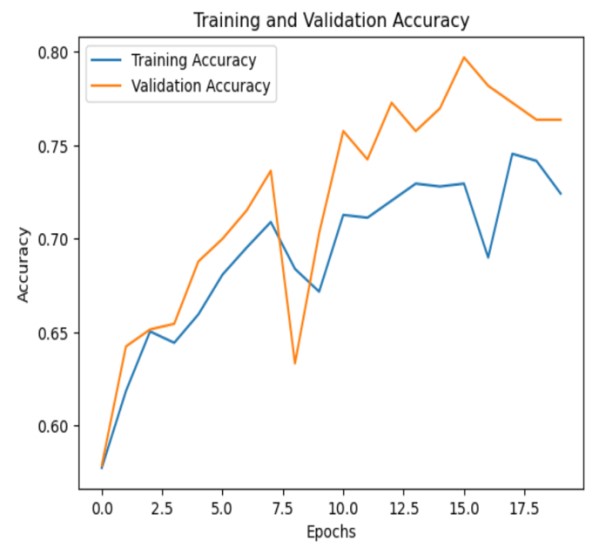
A diagram of a number of emergency

Description automatically generated with medium confidence

Evidence of overfitting in the model.

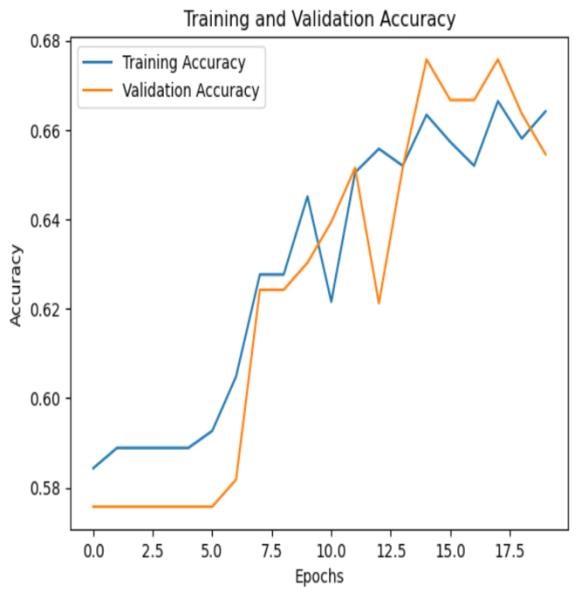
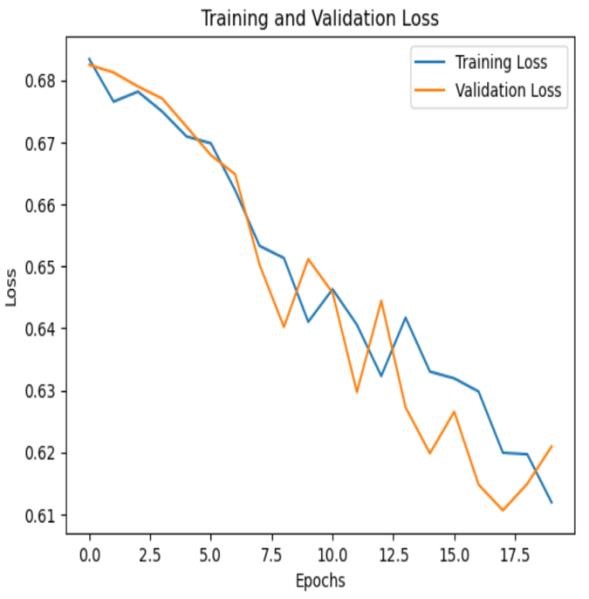
There was evidence of overfitting in the CNN training. Using the average pooling method in each pooling layer, there was an obvious overfitting recognised from the Training plots.

Using the average pooling with a kernel size of 3x3, activation as ‘relu’, with a batch size of 32 to train the model, after training it shows there was overfitting.

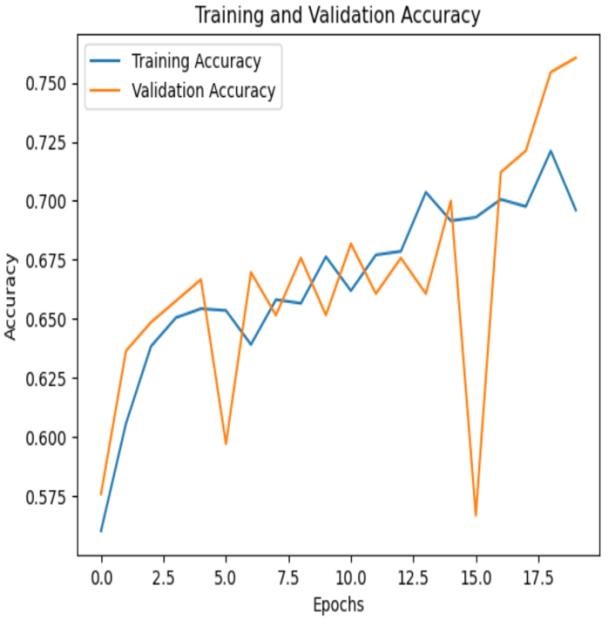
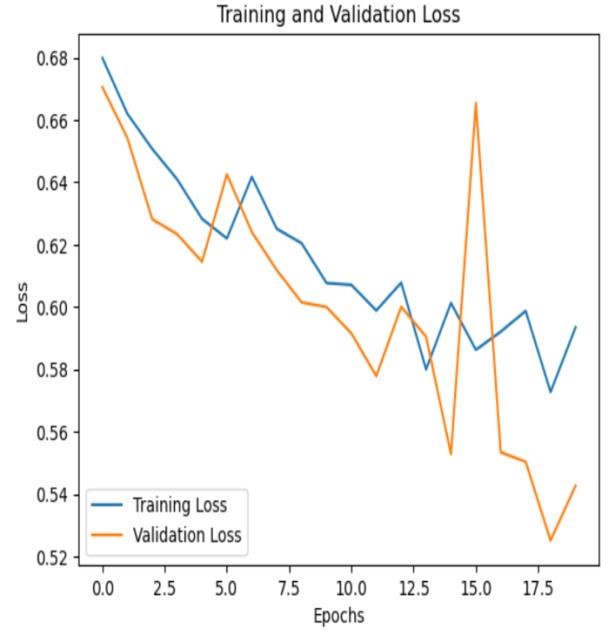


The plots above show overfitting in the training process. This overfitting occurred when the average pooling was added to each layer of the model.

Additionally, adding stride and padding to the model from the just concluded training with average pooling, there was overfitting in the model using the same batch size of 32.



Training the model with a 5x5 kernal size with average pooling in each layer, and a batch size of 128, overfitting was also spotted in the training plots respectively.



These graphs above show over fitting in using average pooling to train the model with a batch size of 128.

In conclusion, developing a CNN model can pose a lot of challenges, regarding what hyperparameter tuning to adopt. The project carried out shows that hyperparameter tuning can affect the performance of your model when training.

Reference.

Dulari Bhatt et al 2021(CNN variants for computer vision: History, architecture, application, challenges and future scope.

Link: https://www.mdpi.com/2079-9292/10/20/2470

Govinda Dumane, Published in Towards Data Science Mar 2, 2020,

Link: https://towardsdatascience.com/introduction-to-convolutional-neuralnetwork-cnnde73f69c5b83#:~:text=Dense%20Layer%20is%20simple%20layer,multiple% 20number%20of%20such%20neurons.

Dropout in Convolution neural Networks (CNN) a detailed explanation, Vishnuam Oct 7, 2024.

Link: https://medium.com/@vishnuam/dropout-in-convolutional-neuralnetworks-cnn-422a4a17da41

Ibrahem Kandel, Mauro Castelli, The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset,

Link:[https://www.sciencedirect.com/science/article/pii/S2405959519303455](https://www.sciencedirect.com/science/article/pii/S2405959519303455#:~:text=Many%20hyperparameters%20have%20to%20be,single%20forward%20and%20backward%20pass)

[#:~:text=Many%20hyperparameters%20have%20to%20be,single%20forward %20and%20backward%20pass](https://www.sciencedirect.com/science/article/pii/S2405959519303455#:~:text=Many%20hyperparameters%20have%20to%20be,single%20forward%20and%20backward%20pass)