EMERGENCY VEHICLE IDENTIFICATION

Abstract

The pursuit of reducing the disparity between machines and humans has prompted researchers to endow machines with the capability to perceive and comprehend their environment. The emergence of Computer Vision as a field was facilitated by the development of the Convolutional Neural Network algorithm, which served as a fundamental framework for enabling machines to visually perceive their surroundings. This report details the utilisation of Convolutional Neural Networks for the purpose of categorising images into two distinct categories, namely emergency and ordinary vehicles.

1.0. Introduction

According to Saha (2022), a Convolutional Neural Network (CNN) is a type of Deep Learning algorithm that has the capability to receive an input image, allocate significance to different features within the image through the use of learnable weights and biases, and effectively distinguish between them. Convolutional networks share a similar structure to that of the visual cortex in the human brain. Solely within the limited region of the visual field, commonly referred to as the Receptive Field, do singular neurons exhibit a response to stimuli.

2.0. Steps taken prior to building the model

2.1. Establishing image test labels

The initial step was to allocate label values to the test target images. Since the training images were pre-labeled, they did not require any further attention. The implementation of such labels would facilitate the assessment of the performance of the model that has been trained.

2.2. Establishing directories

Two variables, namely train_dir and test_dir, were designated to contain the directories of the images for the training and testing data sets, respectively. Similar procedures were applied to both the train and test dataframes, which comprised the filenames and image labels indicating whether the vehicle was an emergency one or not. The variables train_df and test_df were assigned to the dataframes, respectively. Subsequently, these variables would function as parameters during the creation of image generator functions for training, validation, and testing, which will be elaborated on in subsequent sections.

2.3. Set up of training, test, and validation images generator functions

In order to mitigate the risk of overfitting in the trained models, it was necessary to perform transformations on the images within the training dataset to alter their dimensions. The aforementioned procedure is commonly referred to as image augmentation. In instances where a model is trained on images of identical dimensions, there is a risk of overfitting, resulting in an inability to distinguish the same object in a different setting. According to Lau (2017), the training images were subjected to various transformations such as rotation angles, shear, zoom range, and height shift by the training Image generator in order to alter their shape orientations. Similar

transformations would be applied to the validation and test images to evaluate the model's performance on new images with varying orientations.

3.0. Model building

The development of the ultimate model was an iterative procedure that involves a significant amount of trial and error. Five distinct models were designed utilising varying architectures, and subsequently assessed and ranked based on their respective performances. The optimal model is the one that will be discussed.

3.1. The model architecture

A Sequential model with five convolutional layers, each utilising a 3x3 kernel size, exhibited superior performance. The neural network consisted of multiple layers, with three of them containing 64 filters. The first convolutional layer of the network contained 32 filters, while the final layer contained 128 filters. The utilised model incorporated three Maxpooling layers, each with a pool size of 2x2, to facilitate the downsampling of the feature maps. Subsequent to the application of the convolutional and pooling layers, a fully connected neural network was employed, characterised by a density of 200 and a terminal node situated at the output of the layer. The activation function utilised for all intermediary layers was the leaky rectified linear unit (ReLU). The input layer was implemented with leaky rectified linear unit (Relu) activation function, while the output layer utilised sigmoid activation, since it’s a classification task.

3.2. Compilation

The model compilation was done utilising the binary crossentropy loss function and the Adam optimizer. The modification of the optimizer's learning rate to 3e-4 was implemented due to the observation of suboptimal performance in prior training iterations, characterised by the inability to

attain the global minimum and erratic fluctuations in both training and validation losses following each Epoch.

4.0. The impact of increasing the number of layers on performance and time

The addition of layers results in a consistent improvement in model performance. The reason for this is attributed to the model's ability to gather more comprehensive features of the images through the incorporation of additional layers, thereby leading to enhanced accuracy. As more layers are added, the increase in precision begins to decrease. Beyond a certain threshold, the addition of further layers fails to enhance performance, and it is at this juncture that overfitting happens. The addition of excessive layers within a neural network may result in overfitting, thereby causing a subsequent decline in performance or a reduction in efficacy. The incorporation of additional layers in the model results in an increase in the training time.

5.0. Was there any instance of overfitting observed in the analysis?

During the model training process, overfitting did not occur as evidenced by the gradual decrease in both the training and validation loss after every alternate epoch. The absence of a convergence of losses in the opposite direction indicates the absence of overfitting.

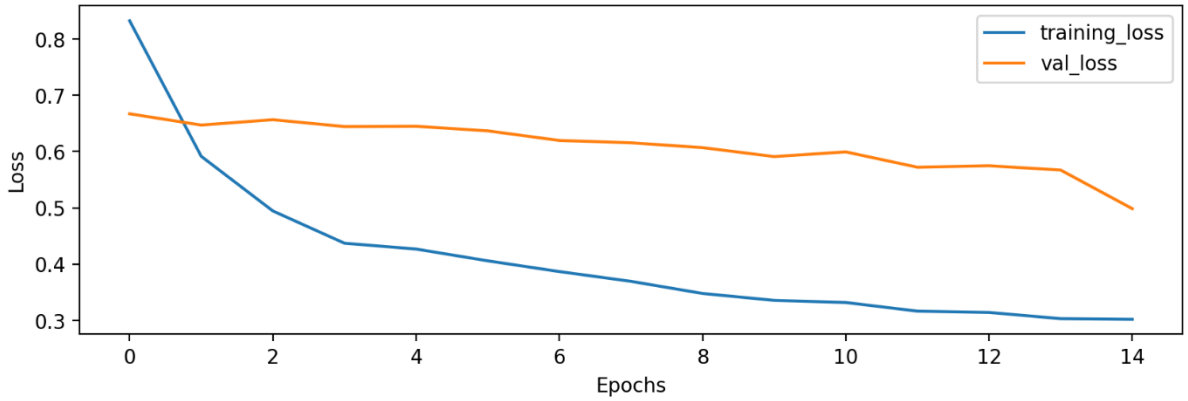

Figure of the losses vs Epochs

6.0. Evaluation Metrics

The evaluation metrics employed for the model comprised of accuracy and loss, for both training and validation. The training loss metric was utilised to evaluate how well the model did on the training data, while the validation loss metric was employed to gauge the performance of the model on the validation set. Additional metrics employed included accuracy, precision, and recall when evaluating the test dataset.

7.0. Results

The optimal model was testing and produced a validation accuracy of 82% and a training accuracy of 86%. The performance of the model was assessed on a set of test images, resulting in a recall rate of 76% and an accuracy rate of 55%.