UFCFSN-15-3 – Artificial Intelligence Documentation

# Vehicle classifier

Patryk Ostrowski 17015998

## 1. A brief description of an idea and aims

The main idea for this classification model was to be able to recognize four common types of vehicles that can be seen on roads. This includes HGVs (Heavy goods vehicle) commonly known as lorries, SUVs, Vans and standard passage cars. The aim was to collect specific images showing these vehicles from a perspective that a camera placed above motorway could have and make a model which could predict a type of vehicle shown to it.

## 2. Research into other related work

The idea of a system being able to recognize different types of vehicles is not something new. With constantly growing number of vehicles on roads every year, such a system could help provide important information for road monitoring, traffic planning as well as improving the safety of transportation. Such systems are already being applied on some roads and motorways with a potential to be used for much more applications in the future.

(Suhao et al., 2018) Article is focused on the detection algorithm that could detect and classify vehicles based on their features. It also focuses on difficulties that a classifier encounters such as different lighting conditions, view angles and specific characteristics depending on vehicle type. In this article Faster RCNN model is applied along with other techniques such as self-building data. End results show better average detection accuracy and rate than traditional machine learning methods. The classifier is able to recognize three type of cars, minibus and SUV with a good result.

Similar to (Suhao et al., 2018), (Vijayaraghavan and Laavanya, 2019) also focus on detecting and classifying vehicles such as cars, buses and motorbikes. Their approach was to create a CNN based on fast regions and treat an entire image as an input. Then create boxes around objects with probability estimates of the feature output.

An interesting example of how vehicle detection might be used is shown in this article (Wong, Goh, Yap and Ng, 2020). It shows a problem with automated toll collection systems and transponders that are used to collect toll. They can be easily tricked by placing inappropriate transponder in vehicles, for example a lorry with a transponder from a passenger car. Therefore, a lorry driver in this example would be charge less than he should be. They implemented video-based detection and classification system that can check what type of a vehicle is passing by and make sure that it has a correct transponder on board, so the correct amount of money can be taken. This way there is no need for workers to check vehicles manually.

## 3. Methods that were used

A first step of making this classifier was to collect images representing chosen types of vehicles. To do that quickly and efficiently a Google Chrome extension called **Image Downloader –**

**Imageye**, was used. With an easy to navigate interface it helped with choosing and downloading suitable images. A total number of about 900 images was collected and split into three folders: training, validation and testing. Each of them having four sub-folders for each vehicle type.

An **Image Data Augmentation** is a process of applying changes to an existing data set to make it more diversify. These changes are for example: rotating an image, zooming in or flipping horizontally. In the result, we can prevent model from overfitting which is a case where model knows learning data too well, therefore it performs worse on unseen images.

A **Convolutional Neural Network (CNN)** is a type of artificial neural network that was used in this mode. It could contain convolutional layers as well as other types of layers such as dense or max pooling layers.

A **Transfer Learning** method is a way of training a model using already existing, pre-trained models. These are made by experts and trained on large data sets. It helps to speed up learning process and achieve better performance. To make it work last layer of pre-trained model has to be changed to match a number of classes in a model that needs to be trained.

## 4. Technical description of implementation

**(Figure 1)** A final model has a few characteristics that helped to achieve better accuracy results.

```
model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(16, (3,3), padding='same', activation='relu',
                           input_shape=(IMAGE_WIDTH, IMAGE_HEIGHT, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),

    tf.keras.layers.Conv2D(32, (3,3), padding='same', activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),

    tf.keras.layers.Conv2D(64, (3,3), padding='same', activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),

    tf.keras.layers.Conv2D(128, (3,3), padding='same', activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),

    tf.keras.layers.Dropout(0.5),

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(256, activation='relu'),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(NUMBER_OF_CLASSES,  activation='softmax')
])
```

*Figure 1 Model code overview*

One of them is use of several convolutional layers called Conv2D. These layers apply a kernel, which size is defined as a second parameter that is 3 by 3 pixels, to the area of an input image. Then values of the kernel and selected area are multiplied and sum together to create one piece of a new convoluted image. This process is repeated until the kernel is applied to all pixels in the original image. In case that a part of the kernel ends up outside of an image, for example if we apply it to first pixel in top-right corner, we use padding. It adds pixels with value of 0 around the image, so we can still perform our multiplication.

MaxPooling2D is another layer that was used for this model several times. It reduces an image by summarizing regions of this image. Therefore, in this case a 2 by 2 pixels grid is moved onto the image. That creates a “window” with four pixels in it at a time. Then a pixel with the highest value is selected and moved to a new image. The grid moves then by stride of 2 pixels to cover another part of the original image and process continues. After the grid covers the entire image, we get a new down-sampled image.

As can be seen many layers use **ReLU** activation function. It is a simple function that can be described as **f(x) = max(0,x)**. It returns 0 if it receives input which is less than 0 or returns the value x itself if it is greater than 0. Because it can account for nonlinear problems, it gave this network much better performance.

```
image_gen_training = ImageDataGenerator(
    rescale=1./255,
    rotation_range=0.1,
    zoom_range=0.1,
    horizontal_flip=True
)
```

*Figure 2 Data augmentation*

**(Figure 2)** Shows a data augmentation function. By applying some random changes to training data set, an overfitting problem was reduced. This gave the model much better accuracy and prevent it from learning images which causes worse performance on data that it has never seen before.

## 5. Results with testing dataset and discussion of findings

The results of this model changed during the development as the model kept being testing using different methods.

Therefore, in the first iteration Loss and Accuracy were respectively *1.93* and *0.44*. In this stage a model contained just one dense layer with 128 nodes and it also used validation data set. **(Figure 3)** Shows an accuracy graph for this stage of progress.

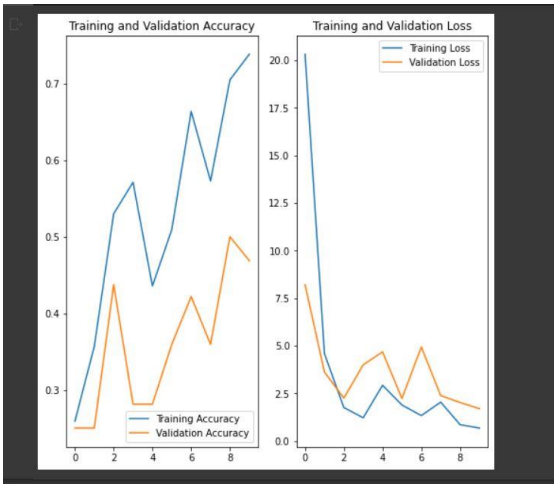


*Figure 3 First stage model accuracy*

In the second iteration a Convolutional Neural Network was applied to the model. That improved an accuracy to *0.64*, which is significantly better than before and increased loss to *2.26*. If we look at an accuracy graph **(Figure 4)** for training and validation data sets, we can see that overfitting takes place which means model learnt training data and performed great on it, whereas did worse on images from validation data set.

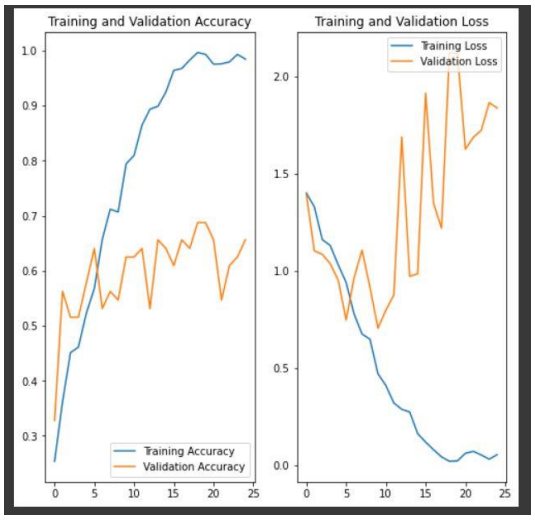


*Figure 4 Second stage model accuracy, overfitting*

To prevent model from overfitting, image augmentation method was used, in order to diversify images by applying random changes to them. For example: horizontal and vertical flip, rotation, zoom as well as shifting images in width and height. Unfortunately, images were modified to much, making it too hard for a model to make a good progress. It caused worse performance at accuracy level of *0.5* and loss equal to *1.15*. It also affected training process making it substantially slower and with worse accuracy for training and validation sets than previously.

After a few changes to data augmentation function, it was reduced to only apply three techniques. These were: rotation, zoom and horizontal flip.

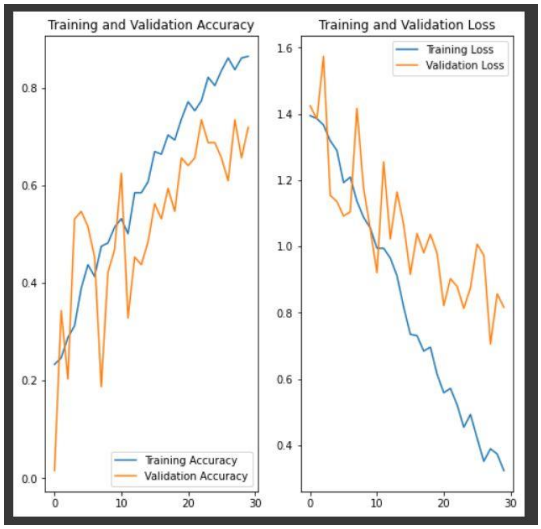


*Figure 5 Final model accuracy and loss comparison*

Final model loss and accuracy on testing data set were *1.00* and *0.65*. Furthermore, looking at **(Figure 5)** graph, much better performance can be seen on training and validation sets. That means correct data augmentation helped reducing overfitting problem. Additionally, adding new different CNN layers for instance dropout and max pooling, made the model more accurate.

## 6. Demonstrated application to a creative task

This project could be applied to many different tasks for instance discussed earlier problem with automated toll collection system. Another use case could be monitoring specific roads which struggle with congestion problem. In such a case, the system would monitor the road and collect data. It could give a key information about what types of vehicles use the road the most and in what time during the day. This could help to improve existing infrastructure by for instance re-organizing road layout or make some vehicles use detour route to split traffic. Furthermore, it could help planning new road developments based on collected data and needs in certain areas.

## 7. Conclusion and suggestion for further work

In conclusion this vehicle classifier made a really good progress. In its final stage it uses a few methods, such as CNN, Data Augmentation, and Transfer Learning. There are many things that could be improved for instance larger data set, but overall performance is quite good.

For future work, it would be great to take this trained model and create a web-based or mobile application. It could use a web-camera or a camera in a smartphone which would allow to classify vehicles in real time.

## 8. Bibliography

Suhao, L., Jinzhao, L., Guoquan, L., Tong, B., Huiqian, W. and Yu, P., 2018. Vehicle type detection based on deep learning in traffic scene. *Procedia Computer Science*, [online] 131, pp.564-572. Available at: <https://www.sciencedirect.com/science/article/pii/S1877050918306616> [Accessed 8 March 2021].

Vijayaraghavan, V. and Laavanya, M., 2019. Vehicle Classification and Detection using Deep Learning. *International Journal of Engineering and Advanced Technology (IJEAT)*, [online] 9(1S5). Available at: <https://www.ijeat.org/wp-content/uploads/papers/v9i1s5/A10061291S52019.pdf> [Accessed 14 March 2021].

Wong, Z., Goh, V., Yap, T. and Ng, H., 2020. Vehicle Classification using Convolutional Neural Network for Electronic Toll Collection. *Lecture Notes in Electrical Engineering*, [online] 603, pp.169-177.

Available at: <https://link.springer.com/chapter/10.1007/978-981-15-0058-9_17> [Accessed 17 March 2021].

B, D., 2018. *Rectified Linear Units (ReLU) in Deep Learning*. [online] Kaggle.com. Available at: <https://www.kaggle.com/dansbecker/rectified-linear-units-relu-in-deep-learning> [Accessed 16 March 2021].

Balla, N., 2020. *Five Popular Data Augmentation Techniques In Deep Learning*. [online] Dataaspirant. Available at: <https://dataaspirant.com/data-augmentation-techniques-deep-learning/> [Accessed 14 March 2021].

n/a, n., n.d. *What is a Convolutional Neural Network (CNN)? - Definition from Techopedia*. [online] Techopedia.com. Available at: <https://www.techopedia.com/definition/32731/convolutional-neural-network-cnn> [Accessed 20 March 2021].