Use of transfer learning to improve Convolutional Neural Network performance

Study on a traffic classification problem – Phase III

Eliza Czaplicka 2022110771

1. Description

One of the leading themes concerning day-to-day development is the introduction and expansion of intelligent cities. More digitalisation in today's world leads to reliability on devices and apps. That is why they need to have accurate data to avoid misguiding users. One such app that we use almost every day is different kinds of maps. They need accurate data about roads, traffic and accidents to work most efficiently. This data is also helpful to the emergency services to quickly respond to collisions and reroute the drivers onto other roads. Another use of such information is the traffic control and data for the city's infrastructure changes. Overall, there is a lot that we can do with data and we need a fast and reliable way to obtain it.

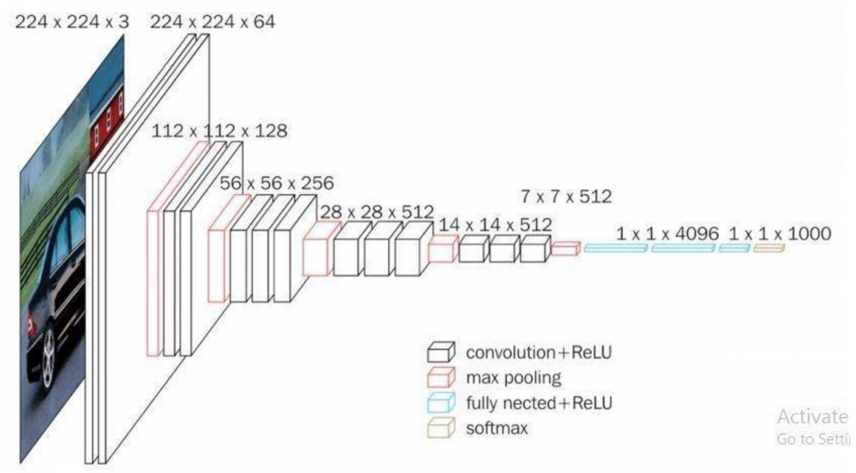
That is where this project comes to light. The goal of the research was to categorize snapshots of different roads into four categories: accidents, fires, dense traffic and sparse traffic. The dataset used was taken from a GitHub repository: https://github.com/OlafenwaMoses/Traffic-Net.

2. Methodologies

The goal of this part of the project was to achieve the best accuracy of classification using transfer learning. Of course, there is no one way to do it. In my trials, I used pre-trained models like VGG16, VGG19 and ResNet50. I will go through my experiments chronologically and further explain the inner workings of these models.

2.1 VGG16

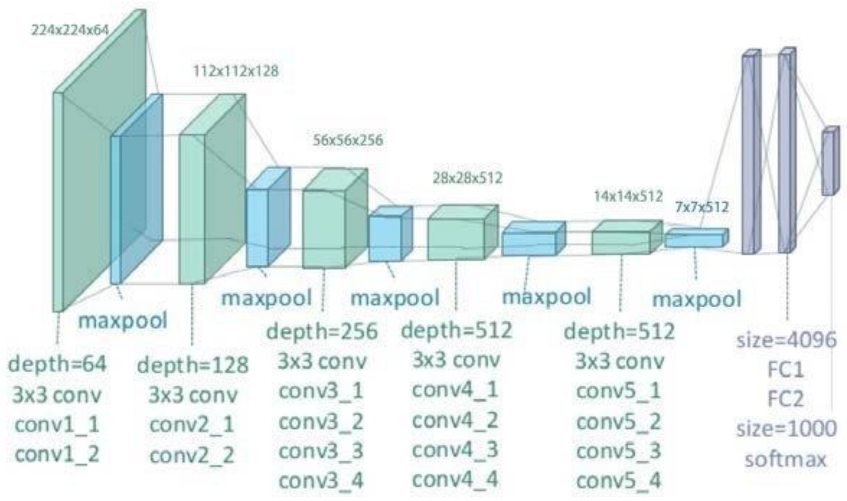
VGG16 was introduced in a paper "Very deep convolutional networks for large-scale image recognition", which was made for the ImageNet Large-Scale Visual Recognition Challenge in 2014 and won 1st place. It is a type of convolutional neural network used for object detection and classification tasks.



It was made as a successor to the AlexNet, created by the Visual Geometry Group at Oxford hence the name VGG. It was trained on the ImageNet dataset of over 14 million images. The architecture is shown below. The 16 in the name comes from 16 layers that have weights, convolutional and dense. The VGG16 is classifies images into 1000 categories.

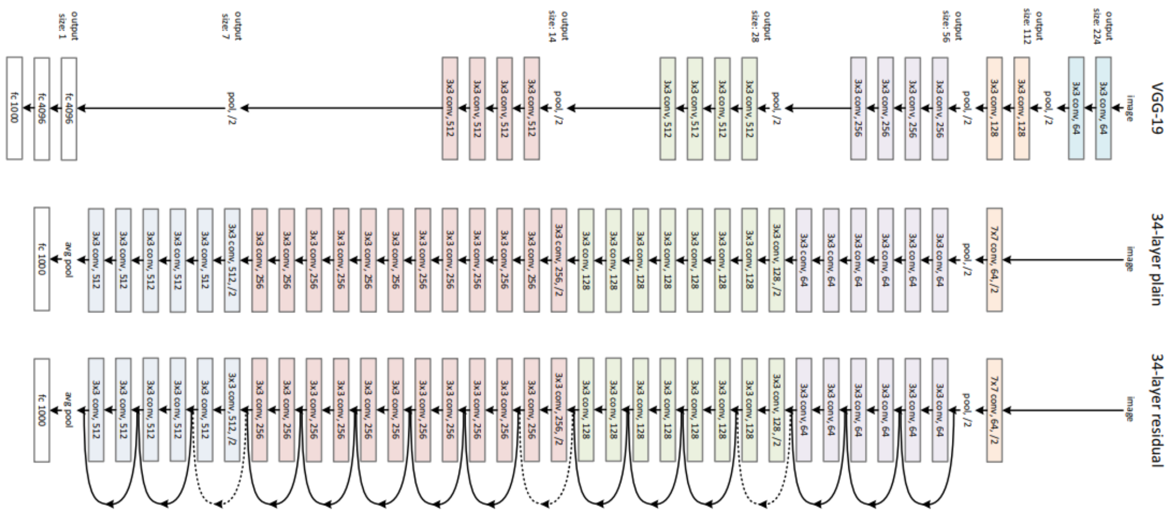
2.2 VGG19

VGG19 as the name suggests has three more convolutional layers. The architecture is very similar to the VGG16.



2.3 ResNet50

ResNet stands for a residual network, which is a type of convolutional neural network. It was introduced in 2015 in a paper titled “Deep Residual Learning for Image Recognition”. The number after the ResNet indicates the number of layers, in this case, it is 48 convolutional layers, one Max Pooling and one average pooling. There are also 18, 34, 101 and 152 versions.



ResNet being a residual network means that it jumps over some layers. It is made of building blocks with a bottleneck design to reduce the number of parameters. It enables faster training of layers. In the graph above, the 34-layer network has blocks of two layers, while ResNet50 has blocks of 3 layers.

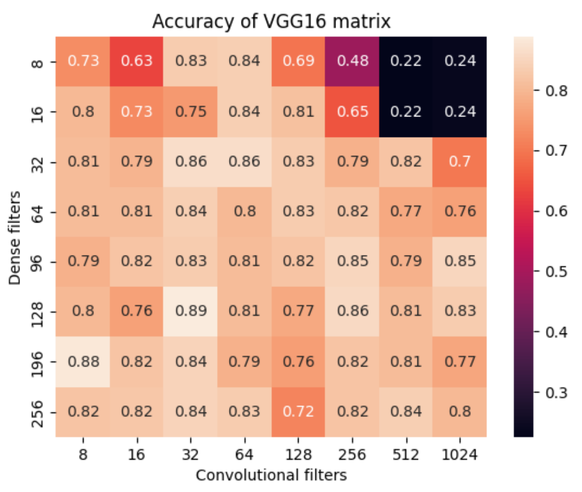
3. Implementation

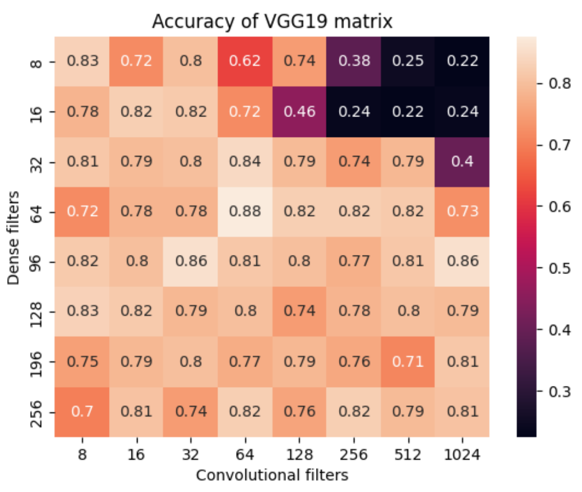
3.1 VGG with a convolutional layer

The first experiment I did I to transfer the VGG 16 or 19 layers to my model, make them not trainable and on top of that put a convolutional layer with a varying number of filters ([8,16,32,64,128, 256, 512, 1024]), a flattening layer, a dense layer with a varying number of neurons ([8, 16, 32, 64, 96, 128, 196, 256]) and finally a softmax layer for the final classification.

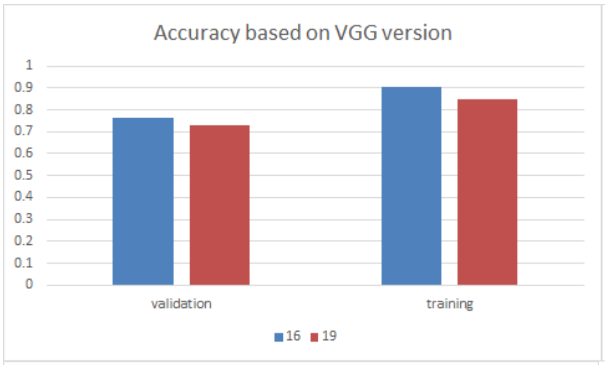
First I used a Whale Swarm algorithm in order to find the best combination of the number of filters, number of neurons and the version of the VGG algorithm. The best result found by that method was VGG16 model with 32 convolutional filters and a dense layer of size 8. The accuracy with that model was just under 0.9 on the validation set.

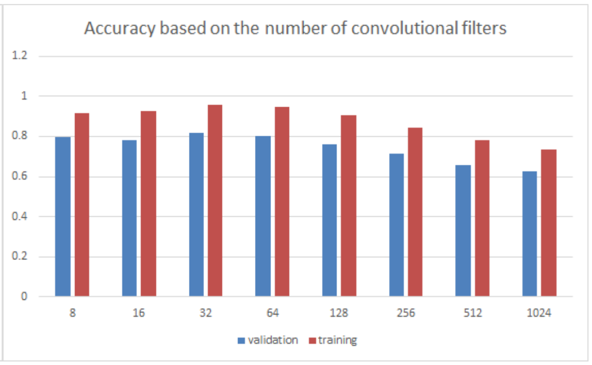
Then I checked all the combinations of the available hyperparameters and the results are shown below.

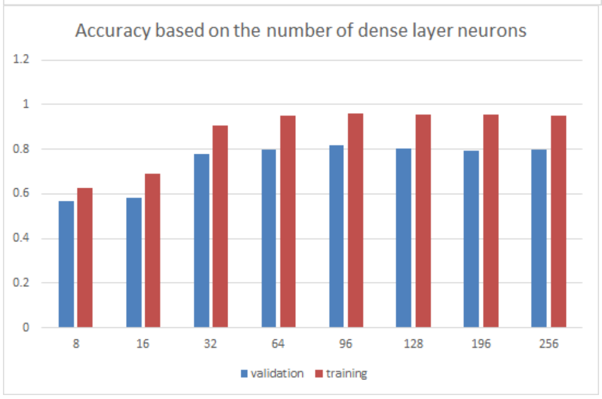




The best result of this method was the 32 convolutional filters but with a dense layer with 128 neurons. We can also see the superiority of VGG 16 for this problem. We can also notice that apart from the worse performance of models with more convolutional filters and less dense neurons, the results do not vary much. Below we can see more in-depth how hyperparameters influence performance.

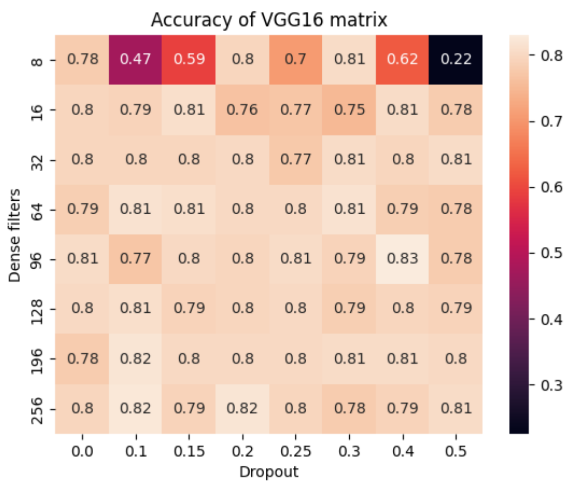


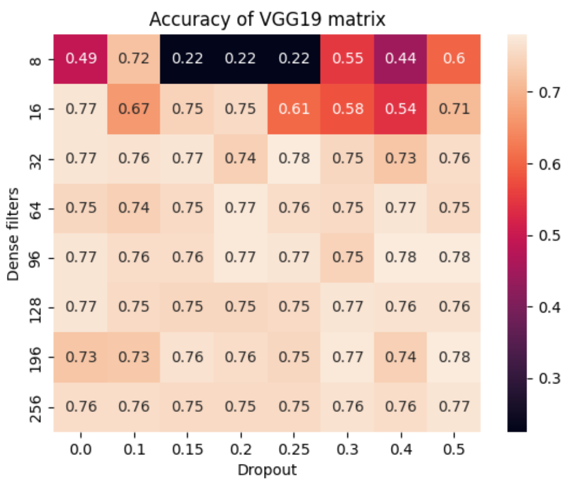




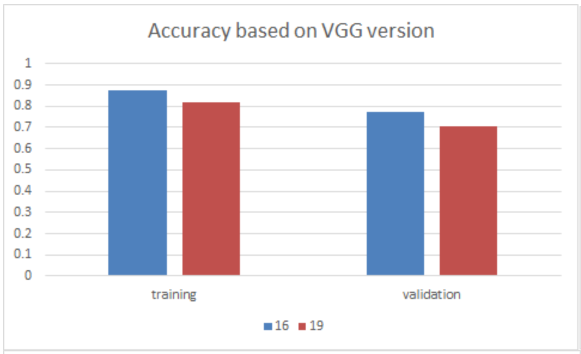
3.2 VGG with only dense and dropout layers

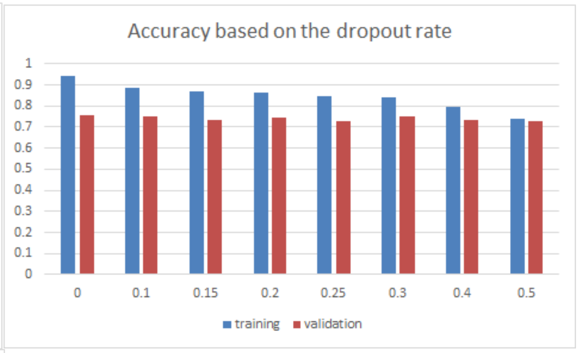
I then took the results from the pure VGG16 and VGG19 models and used them as input to a different model. This one was without the convolutional layer but with a dropout one. The variable here was the dropout rate ([0.0, 0.1, 0.15, 0.2, 0.25, 0.3, 0.4, 0.5]), and the same varying number of dense neurons. Here results were as could be expected by the little change of code.

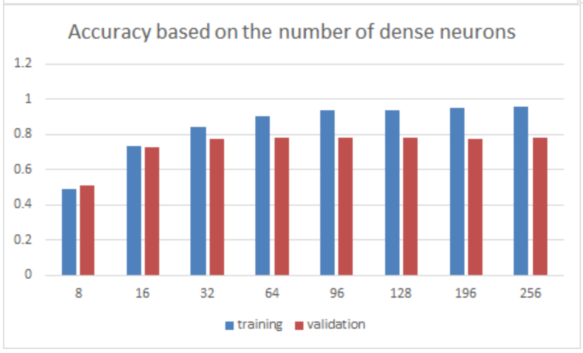




The lower number of neurons in the dense layer is still performing very badly, but the rest of the variables produce very similar results. The VGG19 algorithm is also still performing worse than the VGG16.

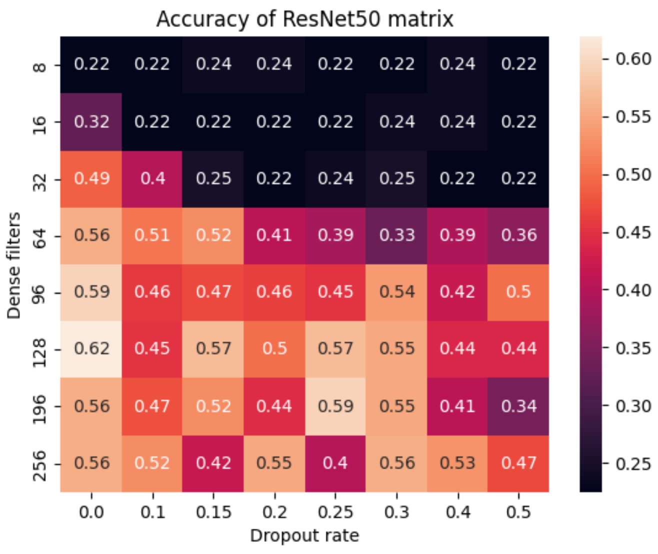






3.3 ResNet50 with dropout and dense layers

Because of the different architecture of the base model, for this experiment, I used two dense layers, the first of the size 512 and the second of varying size, and two dropout layers with the same dropout rate to be found during the hyperparameter search. Unfortunately, the results were a lot worse than with the use of VGG pre-trained models.



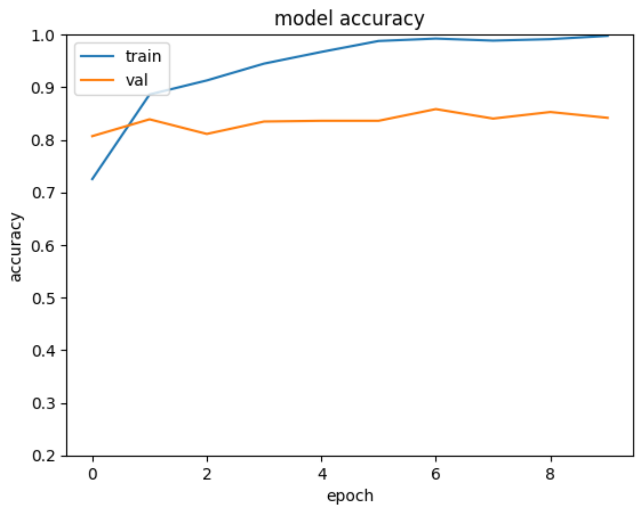
The best result was achieved with a 0 dropout rate and a dense layer of size 128 but even the best result is immensely worse than in the other experiments.

4. Final model architecture

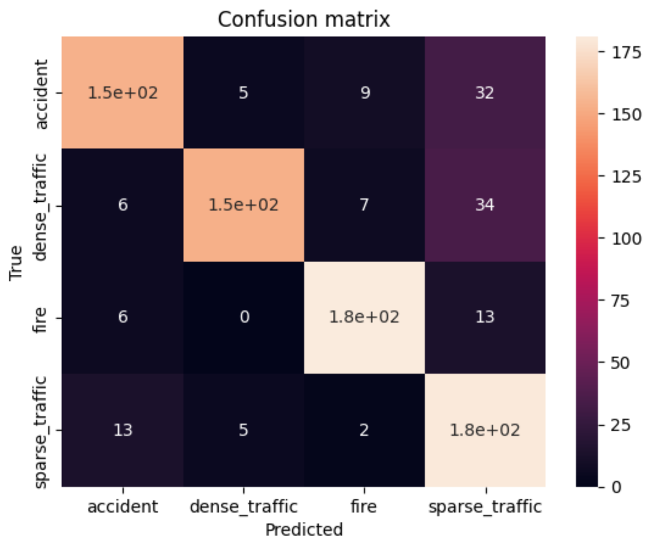
Considering all experiments, the first one produced the best results for the given problem. This means the final model had the initial layers of the VGG16 with pre-trained values, then it had a convolutional layer with 128 filters, a flattening layer, a dense relu layer of size 32 and a dense softmax layer of size 4. With that, the model was trained on the whole training dataset and tested on the test set.

5. Analysis

After training the final model it was evaluated on the testing dataset. The graph below shows the testing and validation accuracy during training.



The final evaluation produced satisfactory results with an accuracy of 0.835. The thing that can be noticed in the confusion matrix of the results is the overfitting of the sparse traffic class. This may be the fault of the appearance of inconclusive images that can confuse the algorithm. Despite this, the created model was thought to achieve suitable results.



True \ Predicted	accident	dense_traffic	fire	sparse_traffic
accident	1.5e+02	5	9	32
dense_traffic	6	1.5e+02	7	34
fire	6	0	1.8e+02	13
sparse_traffic	13	5	2	1.8e+02

6. Bibliography

- arXiv:1512.03385 (Deep Residual Learning for Image Recognition)
- https://medium.com/@mygreatlearning/everything-you-need-to-know-about-vgg16-7315defb5918
- https://www.kaggle.com/code/vincee/intel-image-classification-cnn-keras
- https://blog.techcraft.org/vgg-19-convolutional-neural-network/
- https://github.com/ovh/ai-training-examples/blob/main/notebooks/computer-vision/image-classification/tensorflow/resnet50/notebook-resnet-transfer-learning-image-classification.ipynb
- https://datagen.tech/guides/computer-vision/resnet-50/