Traffic sign recognition - transfer learning using VGG19 and ResNet50

Moa Hoffström moahof@kth.se
Olivia Höft hoft@kth.se
Vasigaran Senthilkumar vsen@kth.se
Aksel Uhr auhr@kth.se

Abstract

In this project, CNN Architecture (VGG19 and ResNet50) and Transfer Learning are used to determine the best classifier for traffic sign recognition. The study involves two classification tasks: a multiclass-classification task and a binary classification task for detecting speed limit signs. The dataset used is GTSRB (German Traffic Sign Recognition Benchmark). The outcomes of this project indicate that batch size, learning rate, and picture resolution all have a significant influence on the performance. On the contrary, utilizing a deeper network (ResNet50), or the SGD optimizer did not improve the performance. only accept psuedo labels with certain threshold accuracy in their probability

1. Introduction

Traffic sign recognition (TSR) is a crucial task when developing secure Advanced Driver Assistance Systems (ADAS) [1] and autonomous vehicles [2]. Previous studies indicate that Convolutional Neural Network (CNN) architectures yield promising results for TSR classification problems [3]. CNN is a common architecture for solving image classification problems and consists of several layers: convolutional layers, non-linear layers, pooling layers and fully connected layers [4]. An interesting property of a CNN is that it can identify an object regardless of its position in an image, as opposed to traditional ANNs which do not perform as favorably [4]. Typically, general features (e.g. shapes, corners) of the images are identified by the network in the initial layer(s) and more detailed features of the image are identified by subsequent layers.

1. **Input layer**: In the context of an image classification problem, the input layer is an image consisting of pixel values [5].

2. **Convolutional layer**: This layer could be considered a filter. Essentially, the input vector is projected to a smaller dimension by converting all the pixels from the input to a single value [5].

3. **Pooling layer**: Pooling is commonly used for downsampling. It takes an input vector and produces an output vector with a reduced number of parameters within an activation. For instance, PoolMax takes the max value of a pixel within a filter (e.g. 3x3) and appends it to the output vector [5].

4. **Fully connected layer**: The fully connected layer is the final layer where class probabilities are produced from the activation functions [5].

There are various pre-trained CNN models available for re-usage. A common approach to transfer learning is downloading, reusing the

weights and fine tuning a pre-trained model for a specific purpose. This approach can help to save both computational power and data collection resources [6]. Typically, the data scientist starts by conducting feature extraction which is the process of freezing all the layers (i.e. preventing the pre-trained model to re-train its weights) and secondly adding a fully connected layer in line with her classification problem. In case the results are poor, the data scientist can carry on with fine tuning the final layers. That is, unfreezing some pre-trained layers in combination with adding and/or removing layers. In order to further improve the results, additional experiments such as hyperparameter tuning, data augmentation and tuning Batch Normalization layers (if they exist in the pre-trained model) can be evaluated in order to create the best performing model for the certain classification problem.

As CNNs are widely used and important for TSR classification problems, the purpose of this project is to present and compare various CNN model experiments using transfer learning (VGG19, ResNet50) on the GTSRB dataset which is described in section 3. Thus, the aim is to provide insights into how different settings, parameters and approaches of transfer learning yield various results applied to the GTSRB dataset.

2. Related Work

The GTSRB dataset was realeased in conjunction with a competition where winners Cireşan et al. [8] constructed a memory friendly CNN that achieved the best results (98.89% pattern recognition) and outperformed other machine learning models. Three years later, Jin et al. [9] ensembled 20 CNN models which outperformed the winners model in terms of pattern recognition to the cost of additional complexity. A recent study used transfer learning with several pre-trained CNN models including Xception, ResNet50 and VGG-19. The study suggests that among all the studied models, Xception's performance was highly successful in terms of accuracy and rate of training, whereas deeper networks achieved good accuracy to the cost of slower training rate [10].

3. GTSRB

The German Traffic Sign Recognition Benchmark (GTSRB) contains over 50000 real images of German traffic signs with 43 different labels [7]. The dataset includes a Train and Test folder containing the images as well as corresponding csv files. Since the images are captured under different circumstances (e.g. weather and daytime) it mimics various driving scenarios.

4. Method

In this section, the workflow, different experiments and their differences will be highlighted. The experiments were conducted using NVIDIA TESLA P100 GPU.

The model used for the majority of the experiments was VGG19 which is pre-trained on the ImageNet dataset. It consists of five blocks, each containing four convolutional layers and a MaxPool layer, as well as three fully connected layers at the end. It was trained using an input shape of (224,224,3) [11].

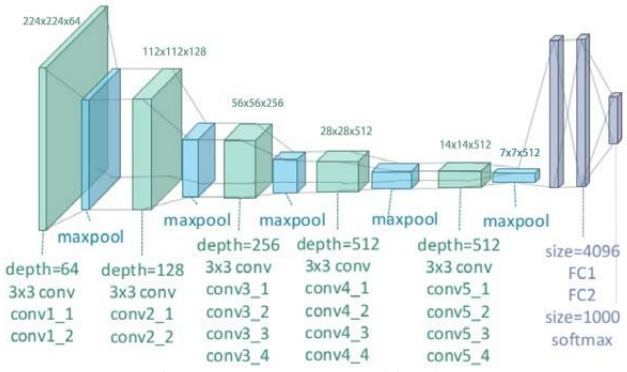

Figure 1: VGG19 Architechture

4.1. Preprocessing

Preprocessing is the task of preparing the data for the model. It is an important aspect of machine learning projects which aims to prepare the dataset before being fed into a model in order to

generate the best possible generalisable results. The steps carried out to apply preprocessing in this project will be described below. Firstly, each image was divided by 255 which projected all pixel values to fall inside the range of 0 to 1. Calculation complexity was decreased in this manner. Furthermore, the training data was randomly shuffled in order to prevent bias before the train / validation split. The resolution of the images varied throughout the project but an input shape of (32, 32, 3) was mainly used for most of the experiments. Ultimately, data augmentation experiments were carried out in order to discover how it may or may not be beneficial for the test results. The train data was split into a training set and a validation with a ratio of 70% belonging to the training data and 30% belonging to the validation data. The number of instances in the training set was measured to 27446, and 11763 number of instances in the validation set.

4.2. Binary classification

The first experiment was a binary classification task to identify whether a sign is a speed-limit (1) sign or not (0). Since the GTSRB dataset comes with 43 classes, we created an additional dataset used for this purpose. The steps carried out to adapt the dataset for the binary classification task was to (1) copy the original dataset, (2) reconstruct the folder hierarchy to contain only two folders labelled as 0 and 1 and putting each image into the folder of its belonging label and lastly (3) the ClassId was remapped to either belonging to 1 or 0. The experiment was conducted without data augmentation and using VGG19 having all layers frozen. Binary cross entropy was used as a loss function and a final fully connected with 2 neurons was added with the Sigmoid activation function. For this experiment, hyperparameter tuning, fine tuning or additional data preprocessing was not carried out.

4.3. Multiclass classification

Various experiments will be carried out for the multiclass classification. At first, a baseline model will be trained and used as a starting point. For this, feature extraction will be used, namely using the pre-trained VGG19 with all layers frozen and a fully connected layer of 43 neurons with the softmax activation function. For this experiment, hyperparameter tuning, fine tuning or additional data preprocessing was not done.

Having the results from the baseline model, the experiments continued in order to achieve better test results. This included: Incrementally fine tuning the model, namely unfreezing, adding and removing various layers of VGG19, grid search and optimal image resolution, data augmentation techniques, comparing the results with a deeper network (ResNet50) and trying different loss functions.

4.3.1 Fine Tuning

The process of fine tuning started off by unfreezing two layers of VGG19. More specifically, at this stage, the final MaxPool layer and the final convolutional layer were trainable. Subsequently, an entire block was unfrozen, namely 4 convolutional layers and the final MaxPool layer. The experiment was carried on by unfreezing two blocks of VGG19.

In between unfreezing the layers, further layers were added which made the network even deeper. Those layers were various dense layers with different numbers of neurons (256, 512, 1024) for more parameters to be trained, as well as Dropout and Batch Normalization layers for overfitting prevention/reduction. Additionally, convolutional layers and pooling layers were added to the network with the aim to train the network to detect highly specific features on the input images. This was more of an experimental phase of the project to see how the test result was affected by the various added layers.

As the final layers of the VGG19 are designed

to identify specific features of the images it is trained upon (imagenet), another experiment was to remove those specific layers entirely.

4.3.2 Grid search, optimizer and optimal image resolution

Initially, ADAM Optimizer was used by default after a quick comparison with SGD optimizer. Afterwards, a grid search for finding an optimal learning rate and batch size was conducted. The learning rate values ranged between 0.00005 and 0.001 with various steps in between. The number of batches ranged from 32 to 512 with a 2x increase for each experiment ([32, 64, ..., 512]). Furhtermore, the image resolution was maximized relative to the amount of memory provided by Kaggle and reached a dimension of (60, 60, 3).

4.3.3 Data Augmentation

This VGG model is trained without data augmentation. This model achieved a testing accuracy of 93.6. The same is then implemented after performing data augmentation with learning rate at 0.0003 and epochs as 15. The data augmentation techniques used include rotation, zoom, width shift, height shift and shear range of 0.15. The testing accuracy achieved is 95.81

4.3.4 Comparison with Resnet50

Two experiments with Resnet50 were carried out, however not using the optimal hyperparameters described in section 4.3.2. In the first experiment feature extraction was performed and for the second experiment fine tuning was applied to see whether the results improved.

4.3.5 Different loss functions

Three different loss functions were used to find the one that gave the highest accuracy for the multiclass classification problem. The first experiment was with the loss function poisson, the second one was categorical crossentropy and the last one was categorical hinge.

4.3.6 Semi-supervised Learning

We used Naive semi-supervised deep learning using psuedo labelling[12] on the dataset. The idea here is to have a small labelled dataset and a large unlabelled set to help improve increasing the accuracy of the model in scenario were we do not have access to large labelled set. We also set aside some data for evaluation datatest. We are making sure they are equal number of samples for each number exist in each dataset. We train the model with labeled data and then predict the Predict labels (pseudo-label) for unlabeled data using the model. Then, we Pre-train the deep learning model with pseudo-labeled data from the previous step and Fine-tune the model with labeled data. The repetition of pseudo-labeling, pre-training, and fine-tuning is the naive semi-supervised deep learning. First, we train the Model on 860 labelled images with 20 images for each label .Then we Train the model with unlabelled dataset and fine tune it by training the model for one epoch on labelled dataset. And during this step, we accepted psuedo labels with threshold accuracy of 0.90 in their probability.

5. Results

In this section the results of the experiments described in section 4 are presented and discussed.

5.1. Binary classification

The first task was to classify signs as being speed-limit signs or not. Prior to performing fine-tuning a test accuracy of .87 was achieved. After performing fine-tuning the best achieved test accuracy was measured to .9973. After achieving this result we decided to not proceed with data augmentation since it was not necessary in order to achieve a high test accuracy.

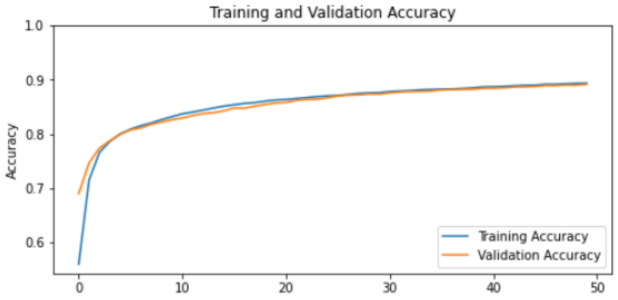

Figure 2: Training and validation accuracy during feature extraction for binary classification

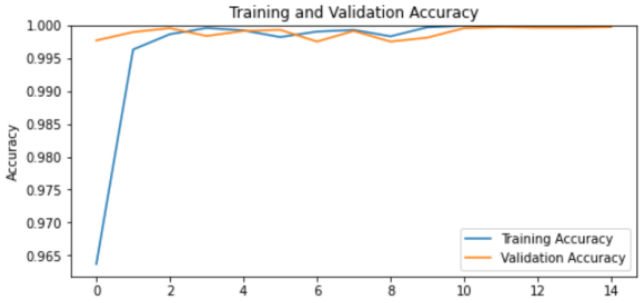

Figure 3: Training and validation accuracy after fine-tuning 10 layers for binary classification

5.2. Multiclass classification: Feature Extraction and Fine Tuning

The initial experiment was carried out using a VGG19 baseline model (trained with feature extraction) generated promising results and was not overfitting at this stage. However, unfreezing the two final layers resulted in an overfitted model. Allowing the model to re-train more parameters (unfreezing more layers, i.e. 10) reduced the overfitting problem with approximately .17 discrepancy between the validation data and test data (.97 vs .80).

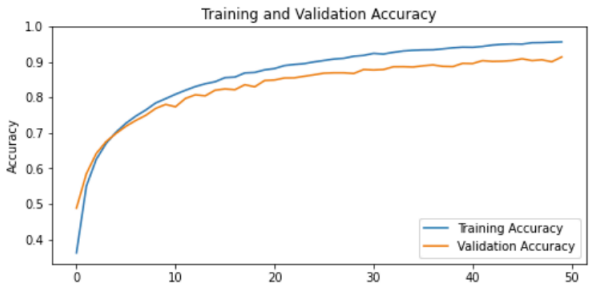

Figure 4: Training and validation accuracy after fine-tuning 2 layersfor multiclass classification

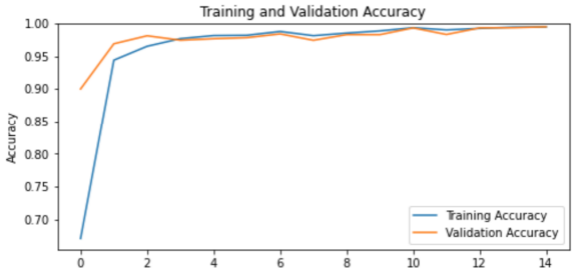

Figure 5: Training and validation accuracy after fine-tuning 10 layers for multiclass classification

At this point, adding and removing different layers to the model did not have a significant effect on the model; the test accuracy varied with approximately +- 2 percent, whereas adding more layers was the reason for the increased accuracy. Obviously, by adding layers (primarily Dense and Convolutional layers), the training and inference time increased and hence there was a trade-off between a slight increase in accuracy for increased computational power. The small accuracy increase could not be motivated and thus the added layers were dismissed.

Regarding the two ResNet50 experiments, the results were poor where the feature extraction experiment resulted in half as promising accuracy compared to VGG19. By unfreezing some layers of the ResNet50, the model overfitted. The reason for those results are plausibly (1) ResNet50 is pre-trained to find exceedingly detailed features based on imagenet dataset which are not generizable using the GTSRB dataset (2) Tweaking the Batch Normalization parameters of ResNet50 and keeping them frozen although their succeeding and predecessing layers are trainable. Hence, it would be interesting to implement further and proper fine tuning in order to discover better results.

5.2.1 Further experiments

Interestingly, the SGD optimizer yields poor results and the reason for this would be interesting to review further. After finding an optimal learning rate (0.0003) the accuracy increased by .02 on the test set. Furthermore, the batch size

of 128 worked best for this classification task. The smaller batch size (32 and 64), training was slower and the accuracy was slightly lower. The larger batch size (256 and 512), training was faster but the accuracy decreased significantly (roughly around .05). Hence, with a batch size of 128, the model was able to generalize well.

The experiment that was the most impactful on the test results was the resolution of the images. By increasing the image resolution ((32,32,3)) to (60,60,3)), the test accuracy increased by over .1. Doing so, the model is exposed to more image data and thus learns patterns that are more generalisable.

The loss function that accomplished the highest accuracy and was the most promising for the dataset used was categorical crossentropy loss. Hence, the categorical crossentropy loss was the one used in the other experiments done in this project.

During the semi supervised learning training with 860 labelled images with 32* 32 dimension achieved a maximum testing accuracy of .542 and evaluation accuracy of .70 for 100 epochs. There was a minor improvement in accuracy compared to training model with labelled date and training the model using psuedolabelled semi supervised learning. The evaluation accuracy after this semi supervised learning process for 10 epochs of fine tuning was .73 and the testing accuracy was .5573 . When the number of epochs during finetuning the labelled dataset was increased to 30 epochs, there was also a slight decrease in the testing accuracy observed.

6. Conclusion

In this project, multiple experiments were conducted in order to solve the German traffic sign classification problems using transfer learning. For the binary classification, the results were promising without extensive experimentation, fine-tuning the two top blocks improved the performance significantly. This approach was shown to generate the best results for multiclass classificaiton as well. Furthermore, the results indicate that the resolution of the images have a large impact on the test performance, as well as the batch size and learning rate. Applying data augmentation was shown to yield a minimal improvement. On the contrary, the SGD optimizer was shown to preform worse than ADAM optimizer, and utilizing a deeper network (ResNet50) was shown to not be suitable for this classification task. We also tried implementation of the semisupervised on the dataset which proved there is not a significant increase observed after implementing the naive pseudo labelling. Some of the labels didn't have enough samples and also Data augmentation could have improved the accuracy more.

7. References

[1] Stallkamp, Johannes, et al. "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition." Neural networks 32 (2012): 323-332.

[2] Farag, Wael. "Recognition of traffic signs by convolutional neural nets for self-driving vehicles." International Journal of Knowledge-based and Intelligent Engineering Systems 22.3 (2018): 205-214.

[3] Mao, Xuehong, et al. "Hierarchical CNN for traffic sign recognition." 2016 IEEE Intelligent Vehicles Symposium (IV). IEEE, 2016.

[4] Albawi, Saad, Tareq Abed Mohammed, and Saad Al-Zawi. "Understanding of a convolutional neural network." 2017 international conference on engineering and technology (ICET). Ieee, 2017.

[5] O'Shea, Keiron, and Ryan Nash. "An introduction to convolutional neural networks." arXiv preprint arXiv:1511.08458 (2015).

[6] Weiss, Karl, Taghi M. Khoshgoftaar, and DingDing Wang. "A survey of transfer learning." Journal of Big data 3.1 (2016): 1-40.

[7] https://benchmark.ini.rub.de/

[8] Cireşan, Dan, et al. "A committee of neural networks for traffic sign classification." The 2011

international joint conference on neural networks. IEEE, 2011.

[9] Jin, Junqi, Kun Fu, and Changshui Zhang. ”Traffic sign recognition with hinge loss trained convolutional neural networks.” IEEE transactions on intelligent transportation systems 15.5 (2014): 1991-2000.

[10] Sowjanya Palavanchu. (2021). Transfer Learning Models for Traffic Sign Recognition System. Annals of the Romanian Society for Cell Biology, 25(2), 3477–3489. Retrieved from https://www.annalsofrscb.ro/index.php/journal/article/view/1333

[11] Understanding the VGG19 Architecture. (2020). https://iq.opengenus.org/vgg19-architecture/.

[12] Zhun Li, ByungSoo Ko Ho-Jin Choi. Naive semi-supervised deep learning using pseudo-label

8. Appendix

8.1. Feature Extraction and Fine Tuning

Model	Classification Problem	Trainable Layers	Train Accuracy	Validation Accuracy	Train Loss	Validation Loss	Test Accuracy
VGG19	Binary	0	.89	.88	.2	.2	.87
VGG19	Binary	10	1.0	1.0	.0	.0	.9973
VGG19	Multclassificaiton	0	.52	.51	1.8	1.9	.42
VGG19	Multiclassification	2	.94	.89	.3	.4	.59
VGG19	Multiclassification	5	.98	.96	.15	.2	.71
VGG19	Multiclassification	10	.99	.97	.15	.2	.80
ResNet50	Multiclassificaiton	0	.24	.23	2.9	2.9	.24
ResNet50	Multiclassificaiton	100	.97	.57	.2	2.3	.45

8.2. Coarse Grid Search and Fine Grid Search

Learning Rate	Batch Size	Image Shape	Augmentation	Optimizer	Epochs	Test Accuracy	Loss function
.0001	256	(32,32,3)	None	SGD	15	.10	Categorial CE
.0001	256	(32,32,3)	None	ADAM	15	.80	Categorial CE
.001	256	(32,32,3)	None	ADAM	15	.78	Categorial CE
.01	256	(32,32,3)	None	ADAM	15	.06	Categorial CE
.1	256	(32,32,3)	None	ADAM	15	.06	Categorial CE
.00005	256	(32,32,3)	None	ADAM	15	.77	Categorial CE
.0003	256	(32,32,3)	None	ADAM	15	.82	Categorial CE
.0005	256	(32,32,3)	None	ADAM	15	.81	Categorial CE
.0007	256	(32,32,3)	None	ADAM	15	.80	Categorial CE

8.3. Image resolution and augmentation

Learning Rate	Batch Size	Image Shape	Augmentation	Optimizer	Epochs	Test Accuracy	Loss function
.0003	128	(50,50,3)	None	ADAM	15	.91	Categorial CE
.0003	128	(60,60,3)	None	ADAM	15	.93	Categorial CE
.0003	128	(60,60,3)	Yes	ADAM	15	.958	Categorial CE

8.4. Loss functions

Learning Rate	Batch Size	Image Shape	Augmentation	Optimizer	Epochs	Test Accuracy	Loss function
.0003	256	(50,50,3)	None	ADAM	15	.92	Categorical CE
.0003	256	(50,50,3)	None	ADAM	15	.91	Poisson
.0003	256	(50,50,3)	None	ADAM	15	.05	Categorical hinge