UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Laura Liis Metsvaht
Navid Bamdad Roshan
Robin Sulg
Heidi Korp

Traffic sign detection and recognition using capsule networks

Report

Intelligent Transportation Systems

Tartu 2020

Report plan (delete this later)

1. **Abstract - in a few sentences describe, what we did**
2. **Introduction - describe the problem**
3. **Related works**
4. **Dataset**
5. **CNN architecture and how we used it**
6. **Capsule Networks and how we used it**
7. **Evaluation**
8. **Results**
9. **Conclusion**

Abstract

Introduction

Together with the rapid advancement in technology the importance of automobiles has increased. Automobiles have a significant role in people's everyday life making the traffic more dense and complex. [6] This in turn results in a bigger number of annual traffic accidents. According to Association for Safe International Road Travel there are approximately 1.35 million people who die in road crashes each year which means that on average 3,700 people lose their lives every day on the roads [5].

According to San Diego Personal Injury Law Offices some of the most common causes of car accidents are distracted driving, speeding and reckless driving [7]. In order to soothe above mentioned trends and increase car and road safety advanced driver-assistance systems (ADAS) could be used. ADAS are electronic systems that aim to assist drivers in both driving and parking tasks. ADAS systems use lidars, cameras, different sensors to detect obstacles or driver errors and respond accordingly while safely communicating through human-machine interface. [8] Some examples of such advanced driver assistance systems are front collision warning, lane departure warning, intelligent speed assistance, pedestrian detection and traffic sign recognition.

Traffic sign recognition is particularly interesting as it depends on the lighting conditions, weather conditions and also presence of other objects [6].

Related works

Traffic sign detection is a crucial part in the development of intelligent vehicles. That is why researchers are constantly searching for better algorithms to overcome common problems such as how easily is the traditional traffic sign detection affected by the environment and poor real-time performance of existing models. For example, Cao et al. [12] have proposed an algorithm that uses HSV color space for spatial threshold segmentation and traffic signs are detected based on their shape. The model uses a classical LeNet-5 convolutional neural network, which consists of two sets of convolutional and average pooling layers, followed by a flattening convolutional layer, then two fully-connected layers and finally a softmax classifier [13]. This architecture is considerably improved by using Gabor kernel as the initial convolutional kernel, adding the batch normalization processing after the pooling layer and selecting Adam method as the optimizer algorithm. The authors have used the same German Traffic Sign Recognition Benchmark datasets as have us in this project.

Zhang et al. [14] have focused on building a very lightweight deep neural network for traffic sign detection. Their approach consists of two lightweight networks that can obtain higher recognition precision while preserving less trainable parameters in the models. Knowledge distillation transfers the knowledge in a trained model, called a teacher network, to a smaller network, called the student network [14]. The student network is a simple end-to-end architecture of five convolutional layers and one fully connected layer, which makes it easy to deploy on mobile devices.

Another solution uses capsule networks in traffic sign detection. According to Koresh et al. [15], the capsule network provides a better resistance for spatial variance and higher reliability in the sensing of the traffic sign compared to the convolutional network. The architecture is built so that the images are received via sensors, the RGB colors are converted to HSV color values, then a HOUGH transform is applied to detect signs and finally, capsule networks are used to classify and recognize the type of the traffic sign. Pari et al. [16] have also used Capsule Networks instead of CNN for traffic sign detection to overcome the challenges imposed by CNN. Capsule Network has the advantage of maintaining spatial relationships among the features which CNN lacks. As a result, CapsNet requires a lot less data for training and the model receives as good results as a CNN with more data. These papers were one of the motivations for using Capsule Networks in our project as well.

Dataset

In our research we are using the data from German Traffic Sign Recognition Benchmark dataset [4]. The dataset tackles multi-class classification problem containing more than 40 different classes. Each class represents a meaning of a traffic sign such as “Stop”, “Speed limit (20km/h)” etc. The dataset is large and lifelike consisting of more than 50000 images in total. It is also important to mention that each real-life traffic sign instance is unique as it only occurs once in the dataset.

Each image only includes one traffic sign and is stored in Portable PixMap format (colourful). The size of the images varies between 15x15 to 250x250 pixels however they are not necessarily squared and the traffic sign might not be in the center of the image.
The annotations of the image include the filename, height and width of the image and four bounding box coordinates (top-left X coordinate, top-left Y coordinate, bottom-right X coordinate, bottom-right Y coordinate).

Figure 1. Examples from the German Traffic Sign Benchmarks dataset.

CNN architecture

A convolutional neural network is a deep learning algorithm which takes an image as an input, differentiates one aspect from another on the image and then assigns importances to each one of them. The preprocessing required in CNN is much lower than for other classification algorithms as convolutional neural networks are able to learn the filters/characteristics by themselves. [9] Convolutional Neural Networks are the most widely used network architecture for processing images and videos [1]. They have proven to be very effective in multiple machine learning tasks such as image recognition and classification [10].

A convolutional network consists of four main operations: convolution, non linearity (here ReLU), pooling or sub sampling and classification (fully connected layer).
The main goal of convolution operation is to extract image features from the input image. In order to learn the features and preserve the spatial relationship between pixels, CNN uses small squares of input data. A filter is slided over the input image and for each position an element wise multiplication is done. Then the multiplication outputs are added to get the final integer which is a single element of the output matrix. [10]

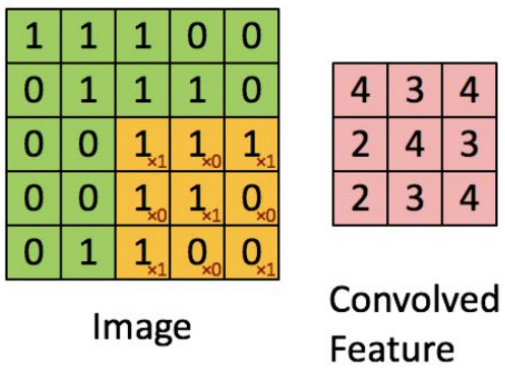

Figure 2. The convolution operation.

After every convolution an additional non-linearity operation (for example ReLU, tanh, sigmoid) is used. It is an element wise operation which in case of ReLU replaces all negative pixel values in the feature map with zero. The previously mentioned convolution step is a linear operation but as we want the CNN to be able to learn non-linear data as well, this is how the non-linearity is introduced. [10]

The spatial pooling operation is applied to reduce the dimensionality of every feature map while retaining the most important information. There are multiple different types of pooling such as max, average and sum. A small window is slided over the input image and in each region dependent on the type of pooling a calculation is applied (for max pooling only the maximum value in the window is taken) reducing the dimensionality of the feature map.
The purpose of pooling is to reduce the spatial size of the input image progressively while also reducing the number of parameters and therefore controlling overfitting. It also makes the network less sensitive to small transformations and distortions in the input image. Pooling makes it possible to detect objects in an image independent of where they are located. [10]

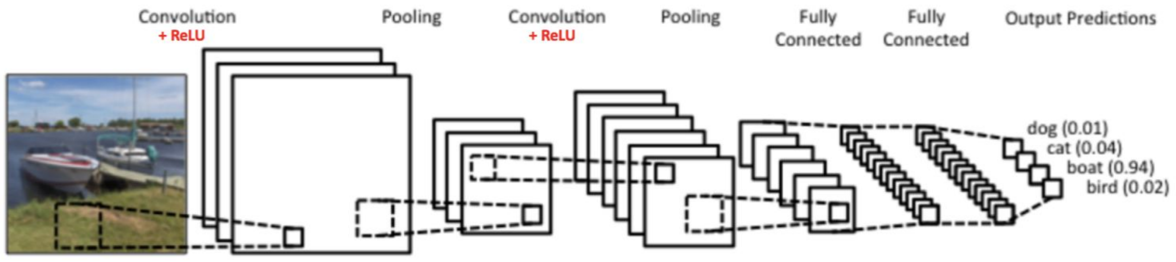

Figure 3. An example of CNN which classifies an input image into four categories: dog, cat, boat or bird.

As a last step of the network there is a fully connected layer, which implies that every neuron from the previous layer is connected to every neuron in the next layer. The purpose of the fully connected layer is to classify the input image into the correct class based on the high level features output from previous layers (convolutional and pooling). [10]

Although CNNs are extremely widely used for image processing they do not encode the orientation and position of the object into their predictions. CNNs completely lose all the information about the pose and orientation of the object, predicting only based on whether some components are present in the image or not. [11]

Capsule Networks

The most widely used architecture for processing images and video data is currently Convolutional Neural Networks [1]. However, it has a few drawbacks which makes them not the best solution for all image recognition tasks. One of such drawbacks is that CNNs in essence, do not take into account relationships between components. For instance, when detecting faces, if a CNN network is able to detect eyes, nose and mouth, then it outputs that a face has been detected, even though the distance or rotation angle between these components might be false. To overcome this problem, CNN uses max pooling or successive convolutional layers that reduce spatial size of the data passing through the network. This method increases the “field of view” of higher layer’s neurons, which allows them to detect higher order features in a larger region of the input image. Unfortunately, by using this method, valuable information is lost.

In order to find an architecture that takes into account the relationships of components when processing images, the Capsule Networks architecture was introduced. Viewpoint changes have complicated effects on pixel intensities, but simple, linear effects on the pose (translational and rotational) matrix that represents the relationship between an object or part of the object and the viewer. The goal of capsules is to make good use of this underlying linearity, both for dealing with viewpoint variations and for improving segmentation decisions. [2]

The algorithm used to train capsule networks is called *dynamic routing between capsules* [3]. Dynamic routing can be viewed as a parallel attention mechanism that allows each capsule at one level to attend to some active capsule at a lower level and ignore the others. This makes it possible to recognize multiple objects in the image even if the objects overlap.

In more detail, a capsule is a group of neurons. Every capsule has an activity vector and its length is used to represent a probability that an entity exists and its orientation to represent the instantiation parameters. Active capsules at one level make predictions using transformation matrices. When multiple predictions agree, a higher level capsule becomes active. Initially, the output of a lower level capsule is forwarded to all of its parents. For each possible parent, the capsule computes a “prediction vector” by multiplying its own output with the weight matrix. If the resulting scalar product is large, there is a top-down feedback that increases the coupling coefficient of the chosen parent and decreases for the other parents. This “routing-by-agreement” algorithm should be more effective than a simple max-pooling when it comes to segmenting highly overlapping objects.

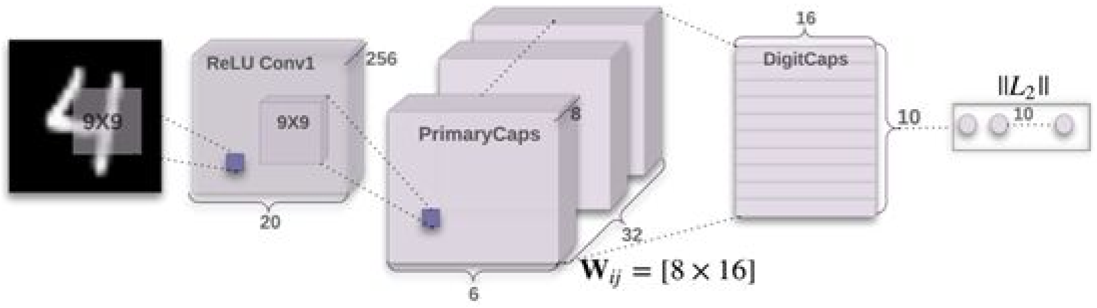

Figure 4. Architecture of Capsule Networks [3]

In Figure 4, there is an illustration of a simple CapsNet architecture. The architecture is shallow and consists of two convolutional layers and one fully connected layer. The first convolutional layer converts pixel intensities to the activities of local feature detectors that are used as inputs to the primary capsules. The second layer (PrimaryCapsules) is a convolutional capsule layer with 32 channels of convolutional 8D capsules (each primary capsule contains 8 convolutional units). Each primary capsule sees the output of all units from the first convolutional layer. The final layer (DigiCaps) has one 16D capsule per digit class and each of these capsules receives input from all the capsules in the layer below. There is routing only between two consecutive capsule layers (PrimaryCapsules and DigiCaps). Since the first convolutional layer is 1D, there is no orientation in its space to agree on.

Evaluation

In our research we have created two models for tackling the traffic sign detection task - one using CNN and another using the capsule network approach. In this section we will go a bit more into detail on how exactly were they implemented.

Both models and their notebooks can be found on the following link:
https://github.com/rosulg/traffic-sign-detection

Convolutional Neural Network

For the CNN version of predicting German traffic signs we used the architecture of Very Deep Convolutional Networks for Large-Scale Image Recognition 7 (VGG 7). The number seven is derived from the number of layers this implementation of VGG has. There are also versions with 16 layers and 19 layers.

As the images in the German Traffic Sign dataset were converted to 32x32x3 the input image for our CNN was a 32x32 RGB (red, green, blue) image. Following by multiple convolutional layers, fully-connected layers and hidden layers.

The convolutional layers use a small kernel/patch size, stride is kept as one to preserve spatial resolution and the number of filters increase by two for each convolution kernel. Each such layer is followed by a ReLU activation function to add non-linearity. In addition, batch normalization and max pooling is applied.

The fully-connected layers have 128 and 43 channels. The last fully-connected layer has 43 channels as we have 43 different traffic signs that we try to predict.

Capsule Network

As the architecture of the Capsule Network is discussed before, the input layer gets an image as input, and then the image will be fed into a single convolutional layer. We resized the traffic sign images to 56x56 pixels to train this network. We chose the image size of 56 pixels over 32 pixels to get a better accuracy. Accordingly, we had to change the convolutional layer so that the output size of the first layer stays the same. So we used stride two rather than original architecture (one). The outputs of the initial convolutional layer go to the PrimaryCaps and then the DigitCaps. The output of the DigitCaps is an array of vectors of size 16. We have 43 different classes in the dataset. So we will have 43 vectors of size 16 as output. The length of each vector describes the confidence of the model about finding the corresponding object (traffic sign) in the image. Therefore, in order to classify the image, the length of each vector can be calculated, the largest vector shows the class that model predicts. Additionally, as part of the capsule network architecture, the output vectors of DigitCaps are also fed into an decoder network in order to reconstruct the input image. The network then calculates the loss by calculating the mean square error between the reconstructed image pixels and the input image pixels. The decoder network consists of four dense layers of sizes 512, 1024, 4096, and 9408, and then reshape to 56,56,3 for reconstructing the initial image. Hence, the network has one input and two outputs, classification and reconstructed image. And the network uses two losses from classification and reconstructed image.

Results

Following sections will present the results of both of the models created.

Convolutional Neural Network

The first result when training our CNN model for 25 epochs we received the accuracy visible on Figure 5 and loss visible on Figure 6.

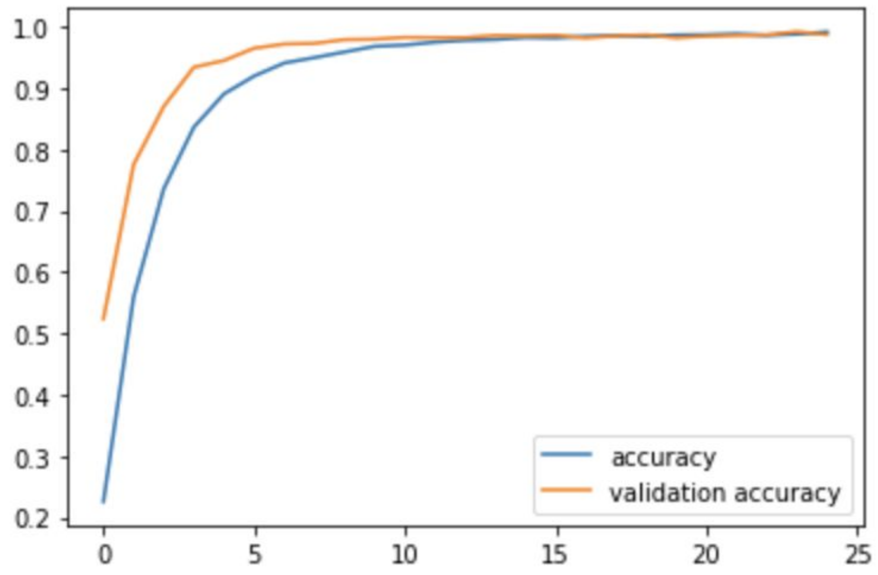

Figure 5. Accuracy vs validation accuracy over 25 epochs during training

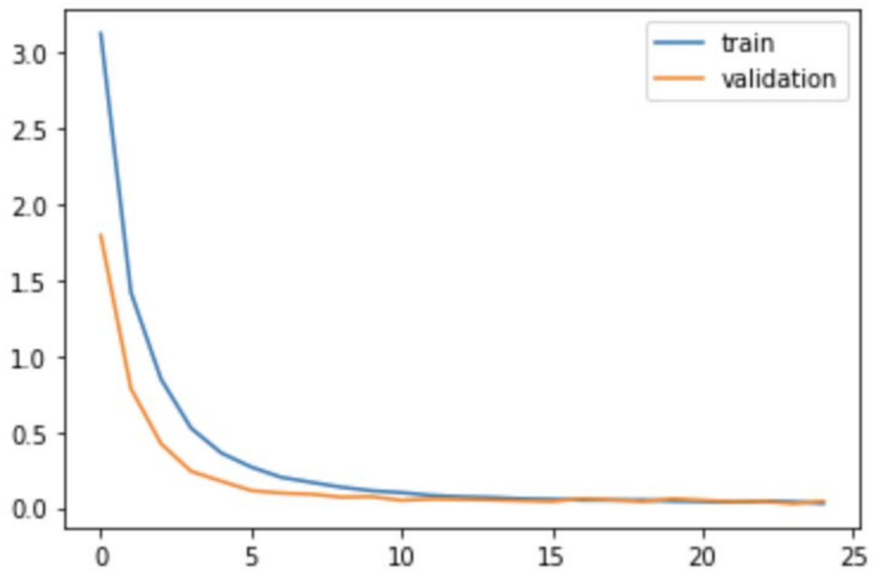

Figure 6. Loss vs validation loss over 25 epochs during training

```
------------------------------Test------------------------------
Test accuracy: 0.9883709068652453
```

In addition, we oversampled the training data and retrained the model on that data for 25 epochs that yielded the results visible on Figure 7 and Figure 8.

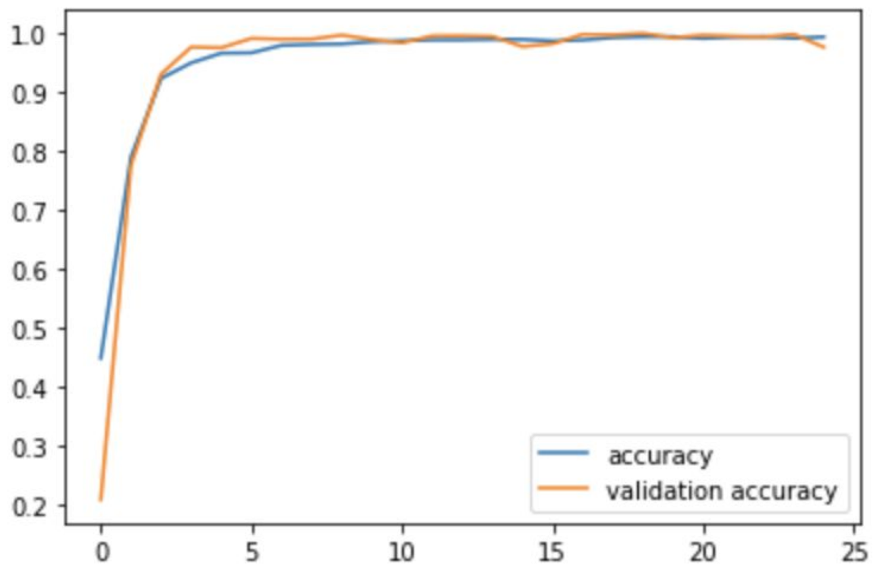

Figure 7. Accuracy vs validation accuracy over 25 epochs during training on oversampled data

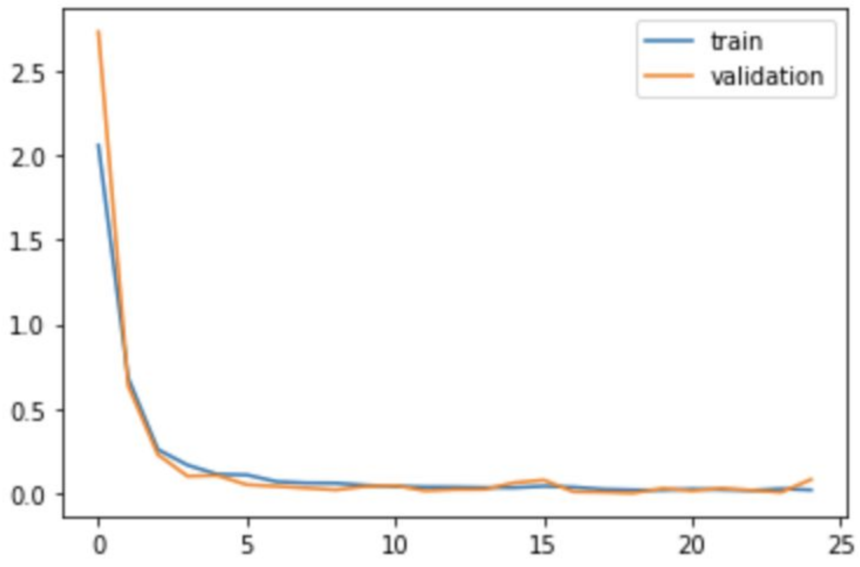

Figure 8. Loss vs validation loss over 25 epochs during training on oversampled data

Test accuracy: 0.9875548301540344

In conclusion CNN is a suitable tool for traffic sign classification as the accuracy was very good. In addition, from the model's learning process graphs, we can see that the deviance from the accuracy and the validation accuracy is small, hence it is unlikely that our model has overfit. From the high test accuracy score it is unlikely that our model has underfitted or overfitted.

Capsule Network

After training the capsule network for 25 epochs, the model had good accuracy on the validation set and test set. The accuracy and the F1 score of the model on train, validation and test set are as follows.

```
------------------------------Train------------------------------
Train accuracy: 1.0
Train f1: 1.0

------------------------------Validation------------------------------
Validation accuracy: 0.9698216735253772
Validation f1: 0.9214275820895599
```

```
------------------------------Test------------------------------
Test accuracy: 0.973950910530483
Test f1: 0.9587816645316041
```

And the training logs of the model are shown in Figure 9. According to the plots, the model was able to fit on the train dataset and predict the validation and test dataset perfectly in less than 5 epochs.

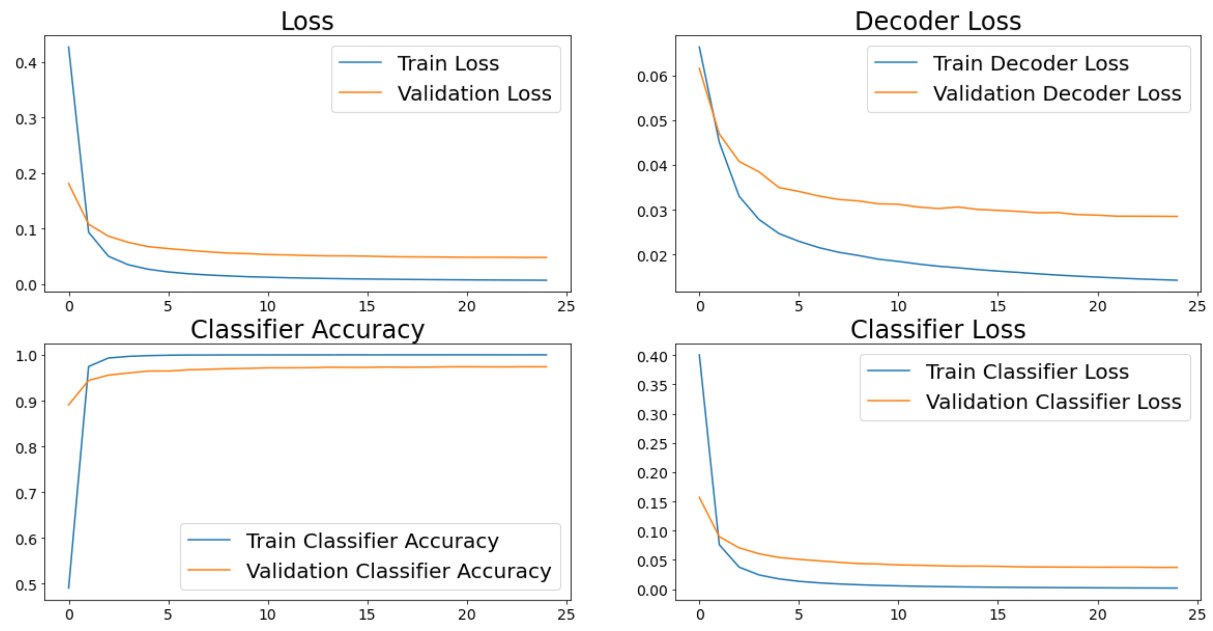

Figure 9. Capsule Network training log

Conclusion

Although the results obtained during the research indicate that CNN resulted in a slightly better accuracy, the difference is so small that it can not be concluded whether one is superior or not. In conclusion both models CNN and Capsule Network are suitable for traffic sign detection.

References

[1] M. Pechyonkin, “Understanding Hinton’s Capsule Networks. Part I: Intuition.,” *Medium*, Dec. 18, 2018. https://medium.com/ai%C2%B3-theory-practice-business/understanding-hintons-capsule-networks-part-i-intuition-b4b559d1159b (accessed Nov. 21, 2020).

[2] G. Hinton, S. Sabour, and N. Frosst, “MATRIX CAPSULES WITH EM ROUTING,” p. 15, 2018.

[3] S. Sabour, N. Frosst, and G. E. Hinton, “Dynamic Routing Between Capsules,” p. 11.

[4] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In Proceedings of the IEEE International Joint Conference on Neural Networks, pages 1453–1460. 2011. http://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset

[5] Association for Safe International Road Travel, Annual Global Road Crash Statistics https://www.asirt.org/safe-travel/road-safety-facts/

[6] P. S. Zaki, M. M. William, “Traffic Signs Detection and Recognition System using Deep Learning”

[7] San Diego Personal Injury Law Offices https://seriousaccidents.com/legal-advice/top-causes-of-car-accidents/

[8] https://en.wikipedia.org/wiki/Advanced_driver-assistance_systems

[9] https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[10] https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

[11] https://sakhawathsumit.github.io/sumit.log/2018/07/21/drawbacks-of-convolutional-neural-networks.html#:~:text=CNN%20do%20not%20encode%20the%20position%20and%20orientation%20of%20the,with%20this%20kind%20of%20information.

[12] J. Cao, C. Song, S. Peng, F. Xiao, and S. Song, “Improved Traffic Sign Detection and Recognition Algorithm for Intelligent Vehicles,” *Sensors*, vol. 19, no. 18, Art. no. 18, Jan. 2019, doi: 10.3390/s19184021.

[13] “LeNet-5 - A Classic CNN Architecture,” *engMRK*, Sep. 30, 2018. https://engmrk.com/lenet-5-a-classic-cnn-architecture/ (accessed Dec. 03, 2020).

[14] J. Zhang, W. Wang, C. Lu, J. Wang, and A. K. Sangaiah, “Lightweight deep network for traffic sign classification,” *Ann. Telecommun.*, vol. 75, no. 7, pp. 369–379, Aug. 2020, doi: 10.1007/s12243-019-00731-9.

[15] J. D. K. H, “COMPUTER VISION BASED TRAFFIC SIGN SENSING FOR SMART TRANSPORT,” *J. Innov. Image Process.*, vol. 1, no. 01, pp. 11–19, Oct. 2019, doi: 10.36548/jiip.2019.1.002.

[16] N. P. S, T. Mohana, and V. Akshaya, “Real-Time Traffic Sign Detection using Capsule Network,” in *2019 11th International Conference on Advanced Computing (ICoAC)*, Dec. 2019, pp. 193–196, doi: 10.1109/ICoAC48765.2019.247140.