Tramspotting
- Is that a tram I glimpse in the distance?

Jens Ifver
Jonatan Hellgren
Sulaiman Carneil

May 2021

Abstract

The following is a report written for a project in the course *MSA301 - Spatial statistics and image analysis* at The University of Gothernburg. The report covers a machine learning project focusing on image classification of the trams in Gothenburg, Sweden. The project consists of two parts: (1) a binary classification and (2) a multi-class classification. The purpose of part 1 is to identify if there is a tram or not and part 2 identifies which tram line an image of a tram consists of. The machine learning model used for this project is a convolutional neural network. Data collection was performed by the authors themselves and several data sets where constructed from the collected data. We show a successful architecture for the first model with acceptable performance. For the second model we where less successful, we bring up what we think the reason for this might be.

Contents

1 Introduction

1.1 Background

The city of Gothenburg is to a great extent characterized by its trams that are a part of the public transport system. The use of trams in Gothenburg have a long history and today trams of several different models travel around the city. The trams are numbered 1 through 13[1] where each number represents a route. Each route also has its own unique color connected to it. This machine learning project is focused on classifying the Gothenburg trams. Specifically we construct two models: (1) Tram or no tram, and (2) Which tram. Model 1 will be a binary classification model that predicts whether there is a tram or not on a specific image. Model 2 will be a multi-class model that predicts which of the trams that are shown in a specific image. The collection of data was done by ourselves with mobile phone cameras and image augmentation was performed. Both Model 1 and Model 2 are constructed using Convolutional Neural Nets (CNNs) to perform the classification. The goal of the project is to:

- Produce a data set that has size and quality enough for the image classification for the trams of Gothenburg.
- Investigate the behaviour of CNNs and find "optimal" architectures for the two models.

1.2 Purpose

Our motivation to execute this project originates from our many years of traveling with the Gothenburg trams and also seeing new tram models being introduced where as old models are discontinued. We also find it educational to collect our own data since we have always been given data in our courses that we already know in advance that it is possible to acheive a good result with. With this approach we really have no idea whether we will be able to do it or not and that excited us.

The reasoning behind our choice for the CNN model is that they are often used for this sort of task and have a reputation of performing well. We also wanted to try out neural networks a bit more since we have just covered them briefly in earlier courses.

1.3 Brief theory

A neural network is a model that is modelled similarly to the structure of a biological brain, where different neurons communicate with each other and the useful connections grows stronger and makes a bigger contribution in further decision making while the less useful ones diminishes and is likely to become

[1] Route number 12 is a special one that is only active during the summer when Liseberg is open, it drives between the central station to Sankt Sigfridsplan with a older tram and is called Lisebergslinjen.

irrelevant. A neural network is basically a very large function with many parameters that is fitted to the data using some sort of gradient decent algorithm in an attempt to minimize a loss function. This process can be computationally heavy and the model usually goes though the data multiple times, each time the model cycles through the training data is called an epoch.

A Convolutional neural network is a standard neural network but before the layers with neurons, one or several convolutional layers are added. The function of these convolution layers is similar to image pre-processing, with the exception that it finds the pre-processing steps automatically and the rest of the network is then trained on the transformed information of the image. Another advantage of using CNNs, is that you can use the lattice structure of a regular RGB image as input and hence utilize the spatial dependencies that neighboring pixels possible possess. In our case we use max pooling to reduce image size and "summarize" information between layers in the network. This pooling can make the output invariant to rotations which we find appropriate in our case.

2 Data

The collection of data was performed by ourselves by photographing trams with mobile phone cameras, with a mix between photos containing trams and photos not containing trams.

2.1 Format

The images was taken in PNG format with the size 320x240. We choose this size since it was the smallest format available on our mobile phones and we didn't want to fill up our memory with high quality images of trams. We managed the data with a GitHub repository where folders for each class contained the corresponding pictures. A cleaning procedure was performed on the data to remove blurry images or images taken from a far distance. We divided all version of the data with a 80/20 training/validation split.

2.2 Versions

Since we have chosen to construct two separate models for the two task, we thought it would be necessary to version the data. The first version of data for model 1 is a balanced dataset with 541 images for each of the two classes "Tram" and "Other".

For model 2 we created a data set with 11 classes where each class contained images containing a single tram. We had to make this 11 since we only managed to collect less then ten images of line 13. We made two different versions of data for model 2, one balanced (Set 1) and one unbalanced (Set 2). In Table 1 we can see the counts for each class in the unbalanced data. In the balanced version for the data we included 58 instances for each class.

Class	1	2	3	4	5	6	7	8	9	10	11
Nr of images	58	99	93	71	101	144	93	73	100	82	100

Table 1: The class sizes for Set 1 used in training of Model 2.

2.3 Possible issues

When collecting the data with our method we have to be aware of possible issues that may occur. With the main one being that the data will be influenced by our own personal bias of what a good tram image is, and thus would probably generalize badly if we where to try it on images of trams that other people had collected. A second issue we could think of is that the size of our data set isn't likely large enough, as neural networks are very data intensive and to create a good CNN model a data set containing at least 1000 images per class is often used.

3 Method

To perform the tasks of image recognition we used the Python3 module `Tensorflow`. The architectures of the two models are different and adjusted to their individual task (architectures can be seen below). CNN models consume a lot of memory, hence both models have a memory limit of 8gb each[2]. Both models are sequential with serveral convolutional layers, max pooling layers followed by fully connected layer(s). In an attempt to improve the performance of Model 2, a method of image segmentation was applied.

3.1 Image augmentation

Neural network tend to overfit easily i.e. completely memorizes the training data and as a result the performance on the validation data suffers, essentially causing no generalization. To solve this issue we used image augmentation. With image augmentation we augment all of the training images before each epoch in order to get more variation in our data. The augmentation was performed on the training data with the following settings:

```
rotation_rage = 10
shear_range = 0.1
zoom_range = 0.1.
```

An horizontal flip was also enabled for the augmentation of the training data used for Model 1 (Tram or no tram). Since Model 2 is expected to find tram numbers in the images in order to classify correctly, we found it reasonable not to enable horizontal flip for the corresponding training data. In Figure 1 we can see an example of an augmentation.

[2] The storage limit came from the maximal storage of our GPUs at home, and it was nice to have some sort of limitation for the model.

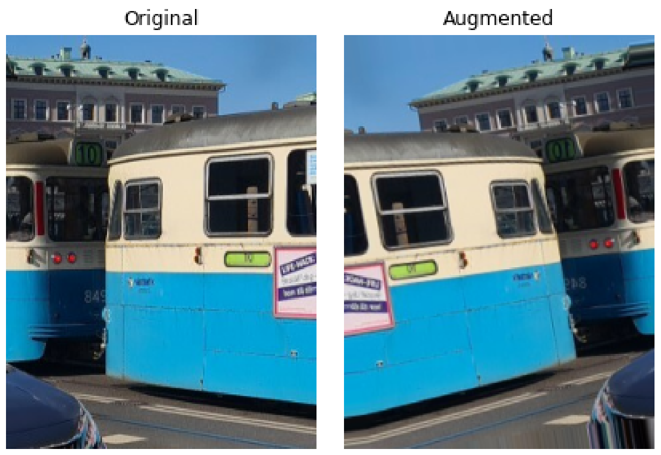


Figure 1: In this figure we can see and example of an augmentation, in the augmented image we can see that the image have been rotated and flipped horizontally.

3.2 Model 1: A binary model

The architecture that we achieved the best performance with can be seen in Architecture 1 below.

Architecture 1 Architecture for Model 1.

1. 2D-convolutional layer: 16 kernels of size (3,3) with strides (1,1) followed by a (2,2) max pooling with strides (2,2).
2. 2D-convolutional layer: 32 kernels of size (3,3) with strides (1,1) followed by a (2,2) max pooling with strides (2,2).
3. 2D-convolutional layer: 64 kernels of size (3,3) with strides (1,1) followed by a (2,2) max pooling with strides (2,2).
4. 2D-convolutional layer: 64 kernels of size (3,3) with strides (1,1) followed by a (2,2) max pooling with strides (2,2).
5. Dense layer: 1280 nodes, l2 regularization = 1e-3,
6. Dense layer: a single node as output, Sigmoid activation function.

Layer 1 through 5 use ReLu as activation function.

We interpret the value in the last node as the probability that the image is containing a tram.

To reduce the overfitting even further we included regularization in the dense layer. With regularization we add a penalty term to the loss function so that parameter values won't become too large. Large parameter values in a neural network is often a sign of overfitting since a large parameter will influence the result, which works if we are classifying the same image but this usually doesn't generalize well. So we are basically letting the network recognize more features and balancing those features. Due to the regularization the learning becomes quite slow and we will thus train the model for 200 epochs with a binary cross entropy loss function and the ADAM optimizer.

3.3 Model 2: A categorical model

Model 2 was trained with two different architectures. The output layers uses a number of nodes equal to the number of classes and softmax as activation function to produce probabilities for each class. As loss function the model uses categorical cross entropy. Since we collected the data ourselves and issues of fitting Model 2 came up, a part of our method includes testing the different data sets to investigate what kind of data is needed to successfully produce a model for solving these kinds of problems.

Architecture 2 First architecture for Model 2.

1. 2D-convolutional layer, 32 kernels of size (3,3) with strides (2,2) followed by a (2,2) max pooling with strides (2,2).
2. 2D-convolutional layer, 64 kernels of size (3,3) with strides (2,2) followed by a (2,2) max pooling with strides (2,2).
3. 2D-convolutional layer, 128 kernels of size (3,3) with strides (2,2) followed by a (2,2) max pooling with strides (2,2).
4. Flattening layer
5. Dense layer, 500 nodes.
6. Dense layer, 100 nodes.
7. Dense layer, 11 nodes, softmax activation.

Layer 1 though 6 uses ReLu activation function.

Architecture 3 Second architecture for Model 2.

1. 2D-convolutional layer: 16 kernels of size (3,3) with strides (2,2) followed by a (2,2) max pooling with strides (2,2).
2. 2D-convolutional layer: 32 kernels of size (3,3) with strides (2,2) followed by a (2,2) max pooling with strides (2,2).
3. 2D-convolutional layer: 64 kernels of size (3,3) with strides (2,2) followed by a (2,2) max pooling with strides (2,2).
4. 2D-convolutional layer: 128 kernels of size (3,3) with strides (2,2) followed by a (2,2) max pooling with strides (2,2).
5. Flattening layer
6. Dense layer: 1280 nodes.
7. Dense layer: 11 nodes, softmax activation.

Layer 1 though 6 uses ReLu activation function.

3.3.1 Image pre-processing

In an attempt to increase the performance of our models we also tried out manual image pre-processing that we learned during the course. We tried segmentation both with and without relative colors, the thought behind this was that the model only need to notice what color the sign of the tram is to be able to correctly classify it, and thus with segmentation it could be possible to "highlight" the tram signs.

With this line of thought an unsupervised segmentation wouldn't suit this model well. What we instead choose to try was to first sample rgb values from the signs of the trams. Then, with these sampled values we could estimate the distribution of that color by estimating the mean vector $\mu_{\mathbf{i}}$ for class $i \in \{1, 2.., 11\}$ and a corresponding covariance matrix $\mathbf{\Sigma_i}$ for that class. With the mean vector and the covariance matrix we can estimate the distribution for the color of the trams sign, we assume that the distribution will be a multivariate gaussian distribution. Then every pixel is classified to belong to the class which pdf achieves the highest value.

$$\text{class}(x) = \text{Max } pdf_i(\mu_{\mathbf{i}}, \mathbf{\Sigma_i}) \text{ for } i = 1, 2..11.$$

This is quite similar to a Gaussian mixture model, the mayor difference is that we estimate the distribution with hand picked values from the signs instead of letting the model find which distributions fits the best, this is done so that we can keep the focus on color of the sign.

3.4 Model evaluation

To begin with we want to state that we will only evaluate the model at the last epoch and the model that achieved the lowest validation lost during the training.

What we will use to evaluate these models are confusion matrices and accuracy. A confusion matrix is a matrix where each column represents the predicted value and the row represents the actual value, leading to a clear overview of how the predictions of the different classes gets distributed. Accuracy is simply the proportion of correctly classified instances. We will also take a look at the ROC-curve for the binary case and compute the area under the ROC-curve called AUC which gives a measure of how stable our model is according to it's threshold where we classify an image as positive (contains a tram). Both optimizer Adam and Stochastic Gradient Decent (SGD) were tested.

4 Result

In this chapter we will present the results for the model architectures described in the previous chapter. We will take a look at the metrics described and also some images that where misclassified by out binary model.

4.1 Binary model

We can see the performance of the model during the training in Figure 2, on the last epoch the model achieved the lowest validation loss of 27.4 and an accuracy of 0.8925. In Table 2 we can see the confusion matrix for Model 1. In Figure 3 we can see the ROC-curve for Model 1. Lastly we can also see some examples of images where the model didn't perform that well in Figure 4.

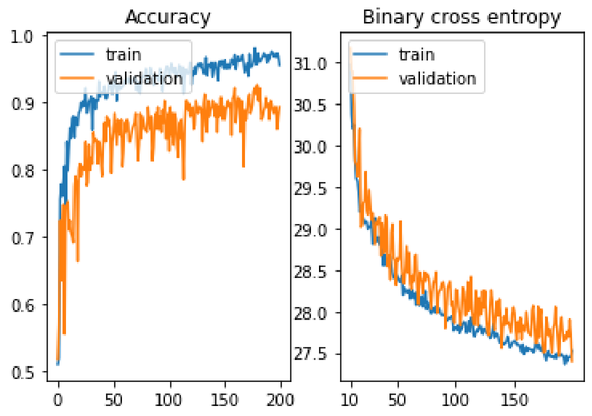


Figure 2: Here we can see how the accuracy and the loss varies during the epochs. Note that the loss functions starts out at epoch 10, this is due to the fact that the loss was simply to large in the beginning and would just make the whole plot very hard to interpret.

	Pred Other	Pred Tram
Other	98	9
Tram	14	93

Table 2: A confusion matrix for model 1, with these value we can compute that the accuracy for the model on the validation data where 0.8925.

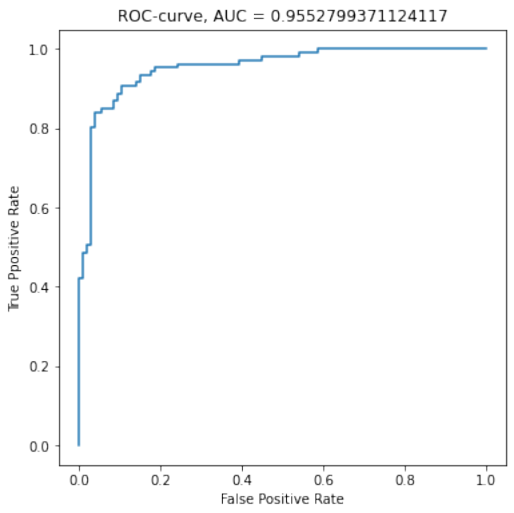


Figure 3: Here we can see the ROC-curve for model 1, we can also the the AUC in the suptitle.

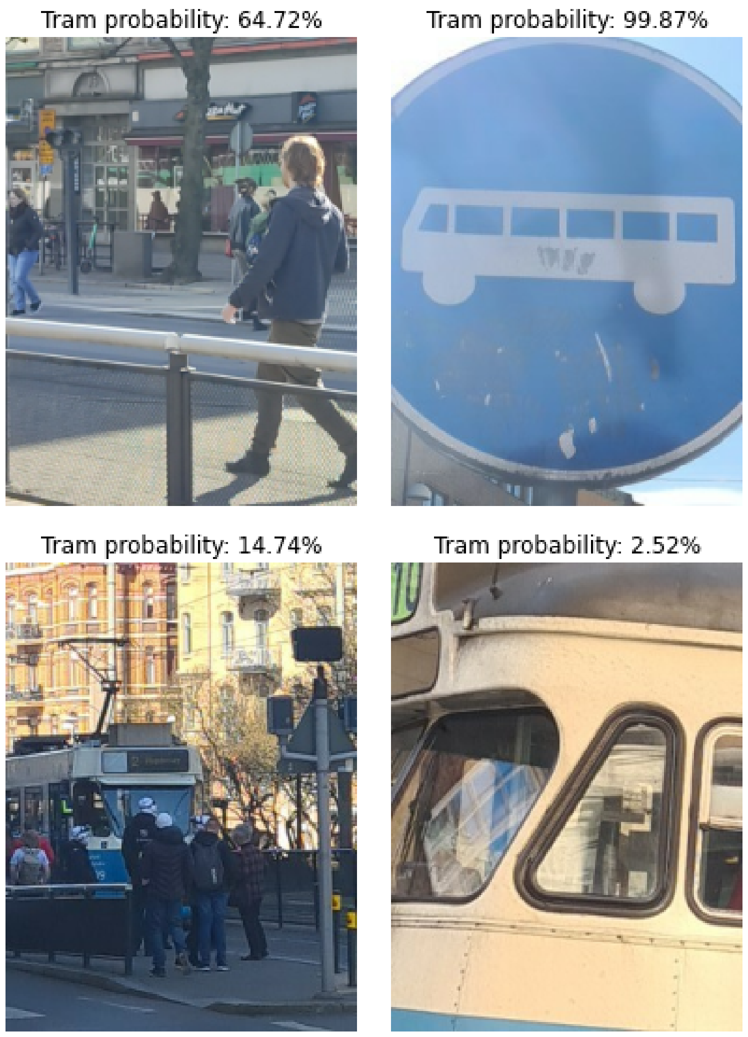
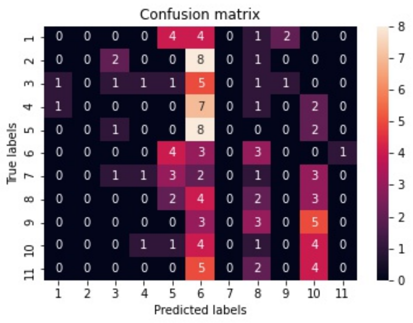


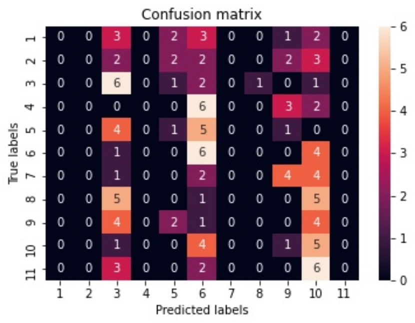
Figure 4: Here we can see some images where the model misclassifies the images.

4.2 Categorical model

The data set that resulted in the best performing version of Model 2 was Set 2 (balanced with 58 images from each class). This model used SGD as optimizer. Both data sets and both optimizers gave rise to overfitted models and the best weights in respect to validation loss were always produced in the early epochs. In Figure 5 we see the confusion matrices for Model 2 with Architecture 2 before regularization. In Figure 6 we see how the confusion matrices looks with Architecture 3 when l1 ($\lambda_1 = 10^{-1}$) and l2 ($\lambda_2 = 10^{-1}$) regularization were introduced.

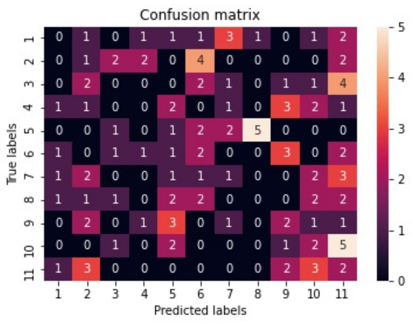


(a) Last epoch (20), accuracy 0.08

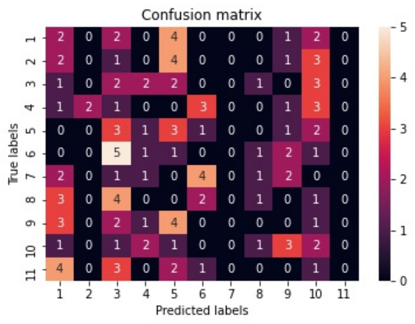


(b) Lowest validation loss, accuracy 0.15

Figure 5: A figure with two confusion matrices for Model 2, Architecture 2. a) is produced by the weights from the last training epoch and b) from the weights that gave the lowest validation loss.



(a) Last epoch (200), accuracy 0.09



(b) Lowest validation loss, accuracy 0.08

Figure 6: A figure with two confusion matrices produced using regularization in Model 2, Architecture 3. a) is produced by the weights from the last training epoch and b) from the weights that gave the lowest validation loss.

4.2.1 The effect of image pre-processing for the categorical model

We implemented the image segmentation on a smaller data set to see how the results would end up. In Figure 7 we see examples of the segmentation for two different images. When training the model on this segmented data set we didn't see any improvements compared to the original data set, and thus we abandoned this idea. [3] The results where similar when using relative colors.

[3] However it did give us some quite artsy looking pictures, which is nice.

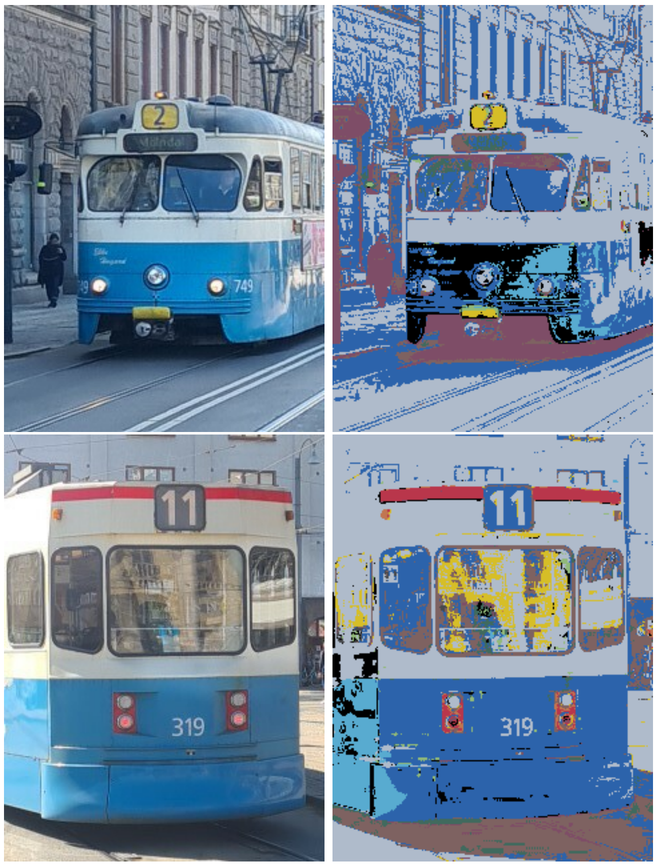


Figure 7: Here we can see an image of tram line 2 and 11 in the left column and a segmented version of it in the right column, using the method for segmentation in Section 3.3.1.

5 Discussion

5.1 Binary classification

The binary classification task of ”Tram or no tram” is handled with acceptable quality by Model 1 with roughly 89% correctly classified objects, considering that data collection and data cleaning was performed by ourselves. We can also see in Figure 3 that the results depending on which threshold we choose is very stable with at AUC of 0.955. This binary task is obviously straight forward since a majority of images corresponding to different classes are fairly dissimilar in appearance. Note that the metrics doesn’t seem to have reached a plateau at 200 epoch, which suggests that better performance could have been reached if we would have run the code for more epochs.

The images of the trams that got misclassified where mainly in the shadow or only showing parts of the tram, this suggests that the model is better at classifying images of trams with good lighting and that contains the entire tram and preferably the front of it. This is likely due to the fact that during data collection, we tried to get images where the sign of the tram where clear and readable for the categorical model. As for the images that got wrongly classified as trams it is hard to find a pattern in, it seems to be quite random, where some being of images of busses and other of shadows or different road signs.

5.2 Categorical classification

The multi-class classification task of "Which tram" is handled poorly by Model 2. We tried several data sets, model architectures, regularization and image segmentation without seeing any significant improvement in model performance. When compared with a majority vote dummy classifier, Model 2 yields equally inadequate performance. The confusion matrices in Figures 5, 6 show no sign of the model finding any useful patterns for classifying the images correctly. In 5 we see a pattern of bias towards class 6 and 10. To understand the reason for this one would have to do an in depth analysis of the data to see what the misclassified images have in common, etc. To draw general conclusions it is preferable to have more data in order to clearly see the incorrect patterns of the classification. We can conclude that regularization increased the distribution of the predictions over more classes. Although this did not increase the accuracy, it is still a sign that the regularization did work in the sense that it's forcing the model to use more subtle patterns that possibly are not in use when the model predicts the validation data to consist of only a few of the classes.

5.2.1 Image pre-processing

We didn't get any notable improvements in performance when applying the image pre-processing. This is likely due to the fact that the convolutional layers performs their own pre-processing steps and adding manual ones is mostly futile. This characteristic is a great reason to why convolutional networks are able to perform so well on different data set.

If we take a look at Figure 7 we can see that the image of line number 2 in the upper row the segmentation might have been successful since it removes redundant information in the background and still preserves the tram line sign. But in the image of tram line 11 in the lower row the segmentation isn't as successful, since the color of the sign is changed to dark blue which is the color for tram line 3. We do however belive that that if there would have been a lesser amount of classes and thus also colors to segment into, it could be possible to highlight the sign of the tram with more success.

5.3 Possible improvements

In order to gain interpretability in the evaluation of the model a method which tell us which part of the images that influence the classification could be implemented. One could do this by inserting a blank square over parts of the image and store how much the output changes with this blank square. Then we could move this square until we have tried all possible locations for it. With the stored changes in the output it would then be possible to construct something similar to a heatmap, which would tell us how much different parts of the image influence the output.

The lacking performance in the categorical case suggests that our model is not suited for this sort of task. The part of the image that signifies which class it belongs to is simply too small for the model to handle. A possible improvement would be to first feed the images through a model that removes redundant information and basically segments out everything but the tram or the number sign of the tram. Then these cropped out images would be possible to classify easily with a CNN since the digits are much clear and regular than the digits in MNIST which we know a CNN can classify with great accuracy.

Another improvement would of course be to collect even more data so that the model would have more instances to train on and thus not overfit as much and therefore generalize better on the validation data.

6 Conclusion

The hype around CNNs and NNs in general promotes the idea that these machine learning methods possess capabilities that makes classification and especially image classification an easy task. To an extent they do deserve their reputation since they often achieve good results, however they are not prebuilt models and treating them like it will often yield bad results. They are also quite hard to interpret, a fact that possibly makes the configuration of the model tougher then in a more transparent model approach.

In our case we see both sides of the coin. In the binary case our model achieves great results and manage the task well with relatively low effort put into configuration of the model construction. In the multi-class case, where we put in considerably more effort into the configuration of the model, the result reveal that the model is incapable of finding any useful features that makes a classification possible. The models performance is thus far below what would be needed for any kind of application. We managed to produce a data set appropriate for one of the two models and although there is always room for improvements the result for the binary model indicates that we found an architecture considerably fit for the task.