INF264 Project 2

Predicting traffic

Sophie Blum and Benjamin Friedl — 08.10.2020

About the document: The names behind every title point out, who initially wrote the code for this functionality. The documentation is mostly written by Sophie.

1 Convolutional Neural Network(Sophie)

One of the models we used is a CNN. Convolutional neural networks are multi-layer perceptrons, that have convolutional layers. In these layers, the images get filtered and the output of the neurons is connected to a specific are in the input image. Convolution itself is a filtering operation on images, that can be used in many different ways. It has the same function in the neural network.

1.1 Preprocessing

The input of the CNN can be the images themselves, so that there is no need to do feature-extracting preprocessing steps. To feed the data into the CNN, we still need to do some minor preprocessing. First the dimensionality of the images needs to be changed. The original shape of each image is (28,28) and gets changed to (28,28,1). Each image is now a 28x28 matrix of greyscale values. It is also possible to have colored images. In this case the shape of the input would be (28,28,3).

```
            X_train = X_train.reshape(X_train.shape[0], 28,
    28, 1)
            X_test = X_test.reshape(X_test.shape[0], 28, 28,
    1)
            X_val = X_val.reshape(X_val.shape[0], 28, 28, 1)

```

In addition, the labels need to be converted to a one-hot-encoding format. Each targetvector is now a vector with 10 rows, indicating each number as the label (1) or not the label (0).

```
            y_data_ohe = []
            for label in y_data:
                ohe_list = [0 for _ in range(10)]
                ohe_list[label[0]] = 1
                y_data_ohe.append(ohe_list)
            y_data_ohe = np.array(y_data_ohe)

```

1.2 Hyperparameters

A CNN has many hyperparameters that can be tuned. We decided on working with different learning-rates and the number of epochs. We decided on these hyperparameters, as we expect them to have a higher impact on the resulting accuracy as well as the time-performance of the model.

1.2.1 Learningrate

The learningrate has a direct impact on how fast the model can be trained and how accurate the trained weights are in the end. A smaller learningrate is good to get more accurate weights, but with the slower rate it also takes more time to compute these. A higher learningrate is good to compute the weights fast, but the weights may not be optimal or the training itself is not stable. These assumptions are later confirmed by the training and validation results.

We are also using the Adam-optimizer, which is a standard optimizer, that adapts the learningrate during the training. To get the best possible result, we train the model with different starting-learningrates.

1.2.2 Epochs

The number of epochs is an important factor when it comes to overfitting, but also plays a role in the time needed to train the cnn and the resulting accuracy in combination with the learningrate.

If a very small learningrate is chosen, there are more epochs needed to get the desired result, whereas less epochs are needed with a higher learningrate. After a certain amount of epochs, the cnn also starts to overfit to the trainingdata, which can be seen when looking at validation loss and accuracy compared to training accuracy. To get the best possible model, we train for 20 epochs and look at the corresponding accuracies and losses after each epoch in the end.

1.3 Model selection

1.4 Overfitting