**Deep learning – HW2 (CNN)**

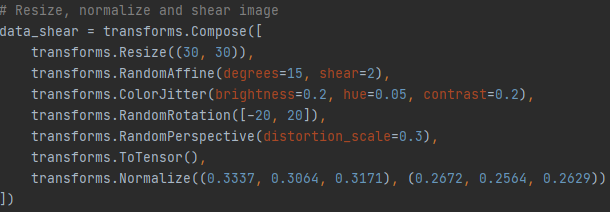
**Data:**

Dataset: German Traffic Sign Recognition Benchmark

Splits: We use the data descripted in the csv files. For validation set, we took 20% from the un-augmented train set.

Data augmentation:

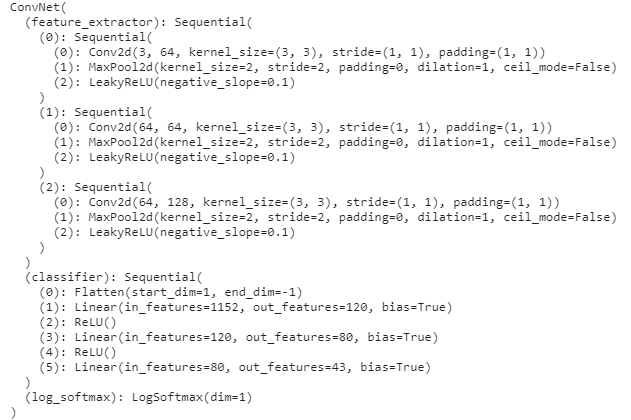
We apply augmentation on the train set and concatenate the original (normalized) training set with the augmented one. This approach helped the model to converge faster and proved to be beneficial.

****

**Model architecture:**

**Feature extractor:**

Without dropout and batch normalization:

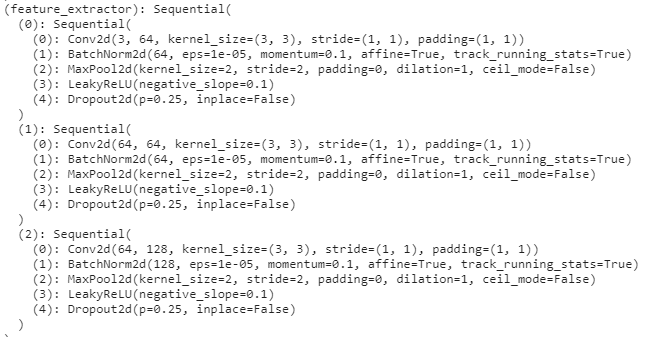


First convolutional layer with output channel = 64, kernel size = 3 and padding = 1 to keep dimensions, followed by max pooling the Leaky-ReLU as activation with negative slop = 0.1.

Second convolutional layer with output channel = 64, kernel size = 3 and padding = 1 to keep dimensions, followed by max pooling the Leaky-ReLU as activation with negative slop = 0.1.

Third convolutional layer with output channel = 128, kernel size = 3 and padding = 1 to keep dimensions, followed by max pooling the Leaky-ReLU as activation with negative slop = 0.1.

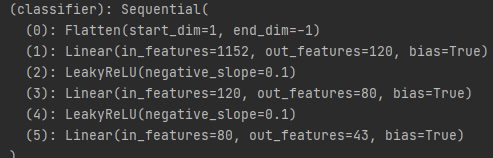
With dropout and batch normalization:

****

Same as before but in this case each of the Conv2d layer will be followed by a batch normalization, and each block will be ended with a dropout with p=0.25 as the probability.

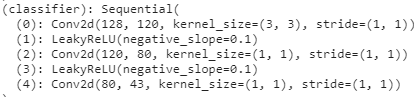
**Classifier:**

Fully connected classifier:



First, we will flatten the input from the feature extractor. After, we will decrease the dimension using fully connected layers to until we reach number of classes to achieve the result for each class.

Fully convolutional classifier:



First, we will lower the height and width to 1 using kernel size = (output height, output width). After, we will apply Leaky-ReLU, and continue by 2 layers of 1x1 convolution to decrease the number of channels to be equal to number of classes (43).

**Training:**

Optimizer: Adam.

Loss function: Cross entropy.

Learning rate: 0.0005.

We use decreasing learning rate with step size = 10 and gamma=0.1, which means that every 10 epochs our learning rate will decrease by a factor of 0.1.

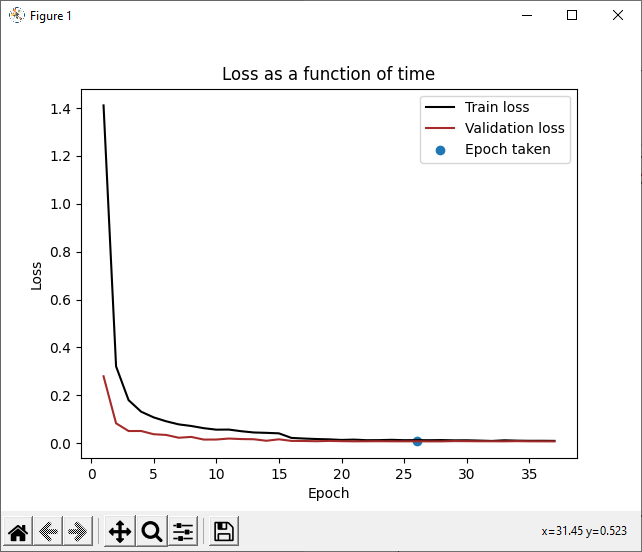
We applied Early stopping with patience = 5 to avoid over-fitting and take the best model.

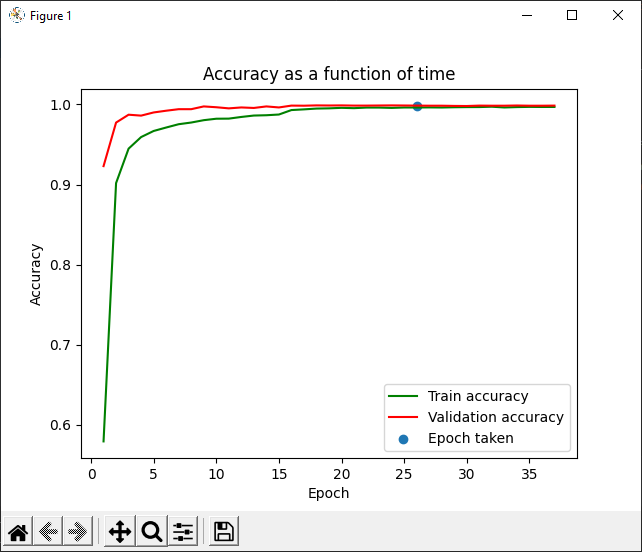
Max epochs = 50.

**Model without dropout and batch normalization (fully connected layers):**

Number of params: 264,099.

**Test set accuracy: 0.9627078175544739**

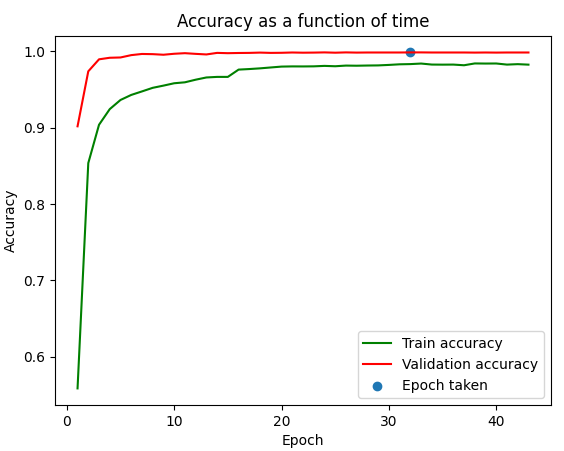


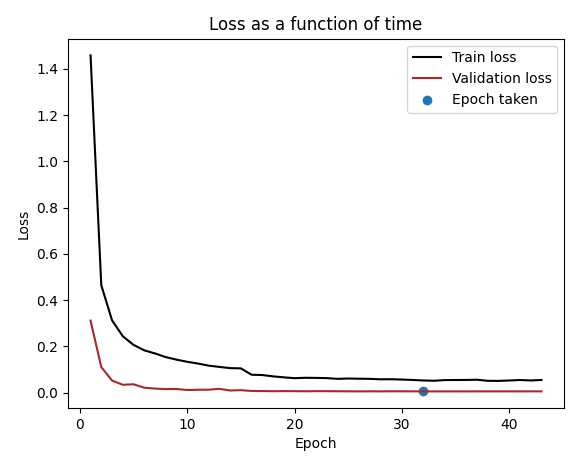


**Model with dropout and batch normalization (fully connected layers):**

Number of params: 264,611.



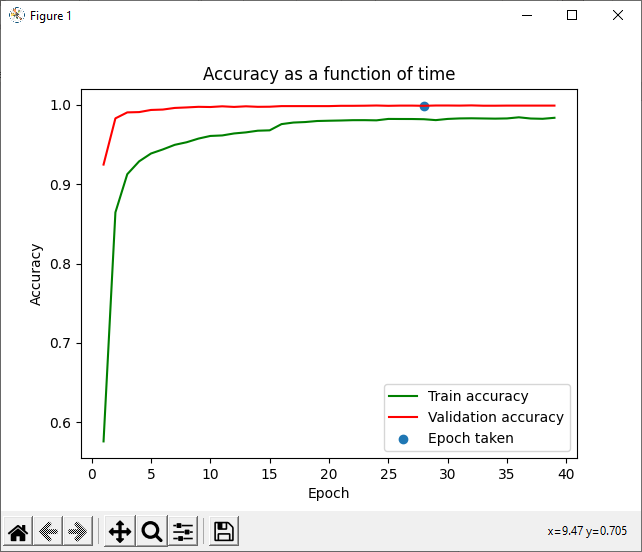


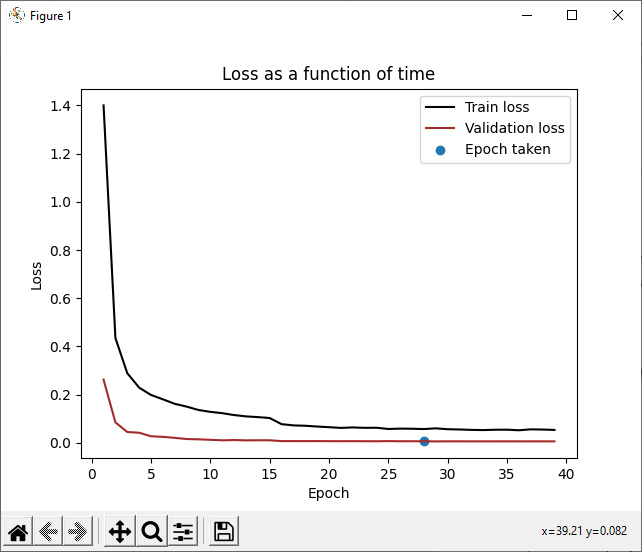


**Model with dropout and batch normalization (fully convolutional):**

Number of params: 264,611.







**Difference between the 3 models:**

* Based on the graphs we can notice that when training a model with drop-out, the train accuracy is lower than the validation accuracy. This happens because when training with drop-out, the model uses only part of the neurons available thus the results can be lower.
* Fully convolutional model has the same number of parameters as the Fully connected model.
* Drop-out and batch normalization models have better test set results thus, better generalization abilities.
* Drop-out and batch normalization add trainable parameters to the network.

**Other attempts:**

* We tried to use a skip-connections with a larger network, which was not beneficial.
* We tried p = 0.5 as a drop-out parameter.
* Many different augmentations.
* Adding layers.
* Add neurons to the fully connected classifier.
* Different weight initialization techniques (e.g., Xavier).
* SGD as optimizer.
* Different learning rates.

**Conclusions:**

Training using a fully connected classifier has similar performance to the fully convolutional network. The number of trainable parameters is identical.

Drop-out and batch normalization can improve test set results.

Using augmentation can improve results significantly when used properly and can lead to better generalization of the model. However, we should choose our transformations wisely with respect to the dataset to keep data integrity.

**GUI print screens of 2 examples:**



