Improving performance just by Data

Submitted by: **Dmitry Kremiansky**

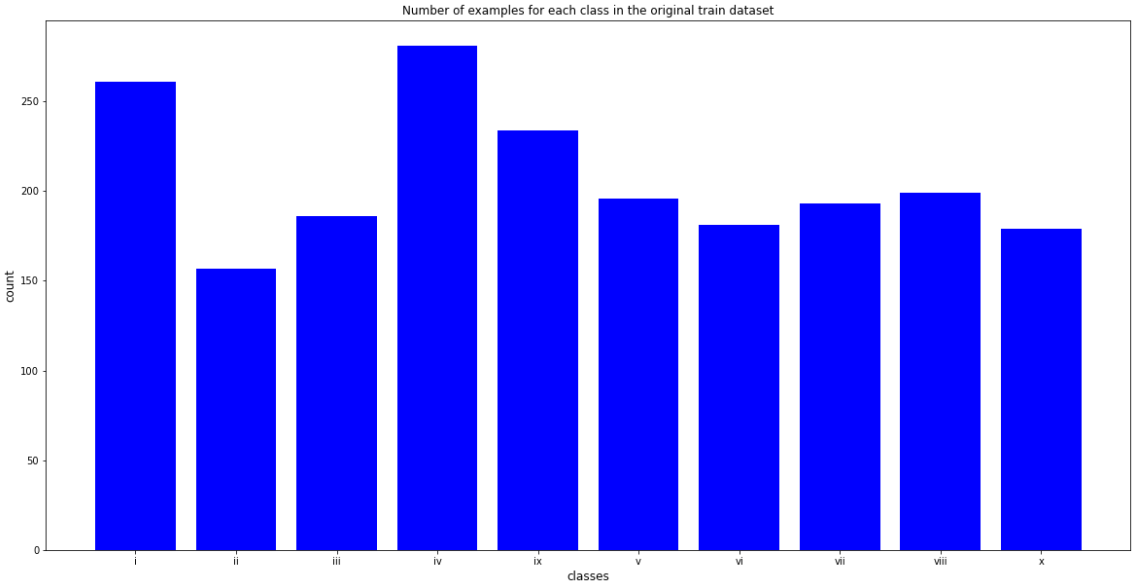
Yossi Gisser

Link to Github repo: [094295_hw2]

Original Data Exploratory

We started our work with looking on the original data that we got. The original train set has 2067 examples, and the original validation set has 10 examples, one for each class.

The counts per class for original train dataset are:$[261, 157, 186, 281, 234, 196, 181, 193, 199, 179]$.

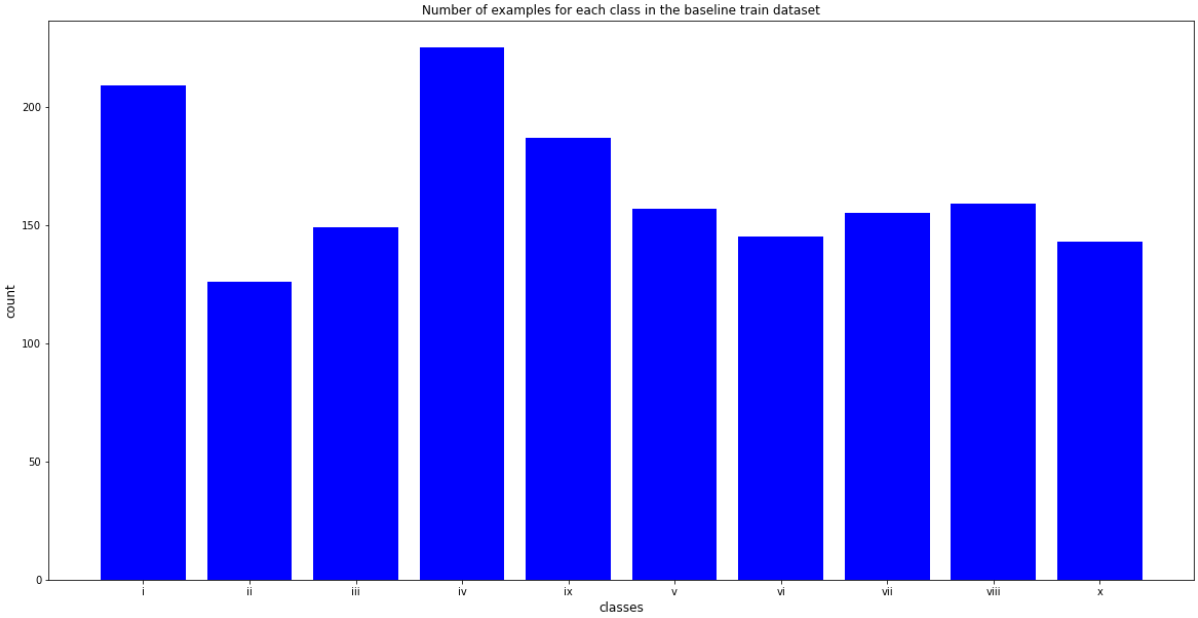


Before we started some preprocessing on the data, we combined the original train and validation datasets and then split them with 80/20 proportion. We split every class by this proportion and in this way, we make sure the balance between train and validation across all classes is the same. We treat to this data as baseline data and the result that obtained using given code as baseline result.

```
The length of the baseline train dataset is: 1655
The length of the baseline validation dataset is: 422
```

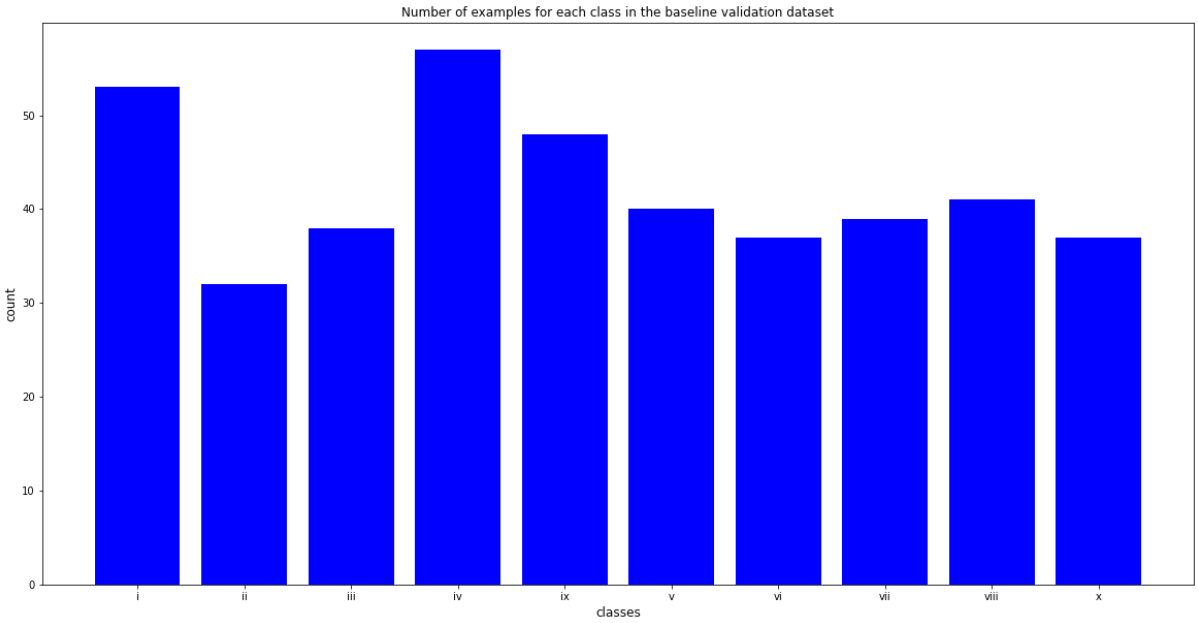
The counts per class for baseline train dataset are:

$$[209, 126, 149, 225, 187, 157, 145, 155, 159, 143]$$



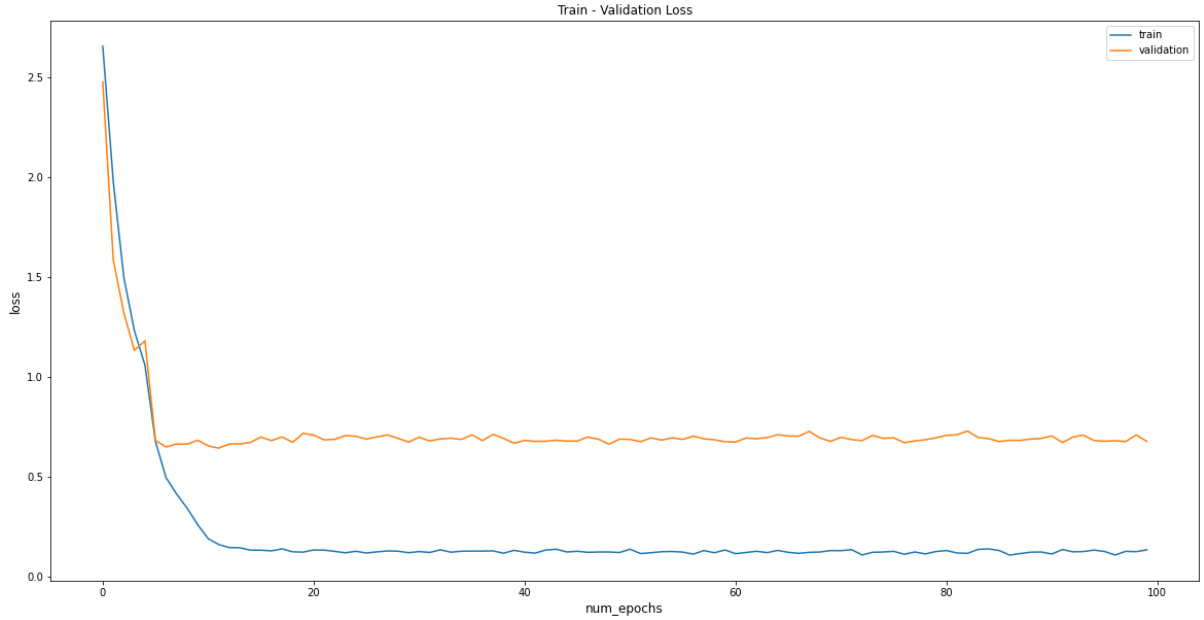
The counts per class for baseline validation dataset are:

$$[53, 32, 38, 57, 48, 40, 37, 39, 41, 37]$$

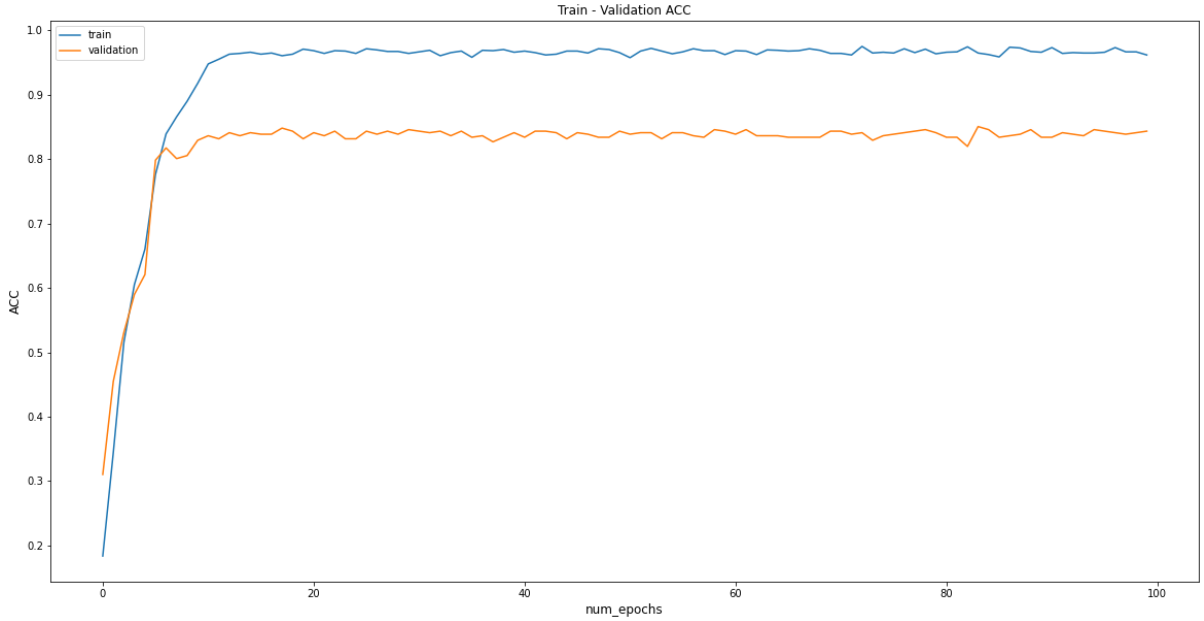


The baseline result is:

```
Training complete in 15m 35s
Best val Acc: 0.850711
```



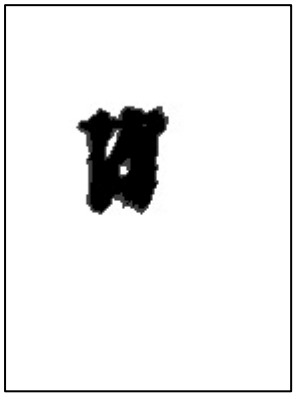
On the train set the loss stabilizes about epoch 15, and on the validation set the loss stabilizes about epoch 10.



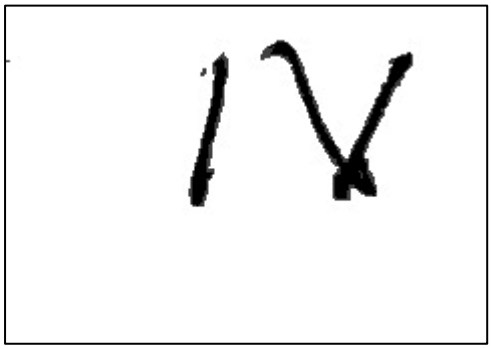
On the train set the accuracy stabilizes about epoch 10, and on the validation set the accuracy stabilizes about epoch 5.

Now we went through over all the data for checking the correction of the labeling. We found out that there are some images that are not even numbers and we deleted them. In addition, we fixed the labels where the image of number had the wrong label and moved it to the right directory. There were some images that are ambiguous about what is the number and we decided to leave it in the original directory.

Examples of not numbers:



Example of ambiguous numbers:



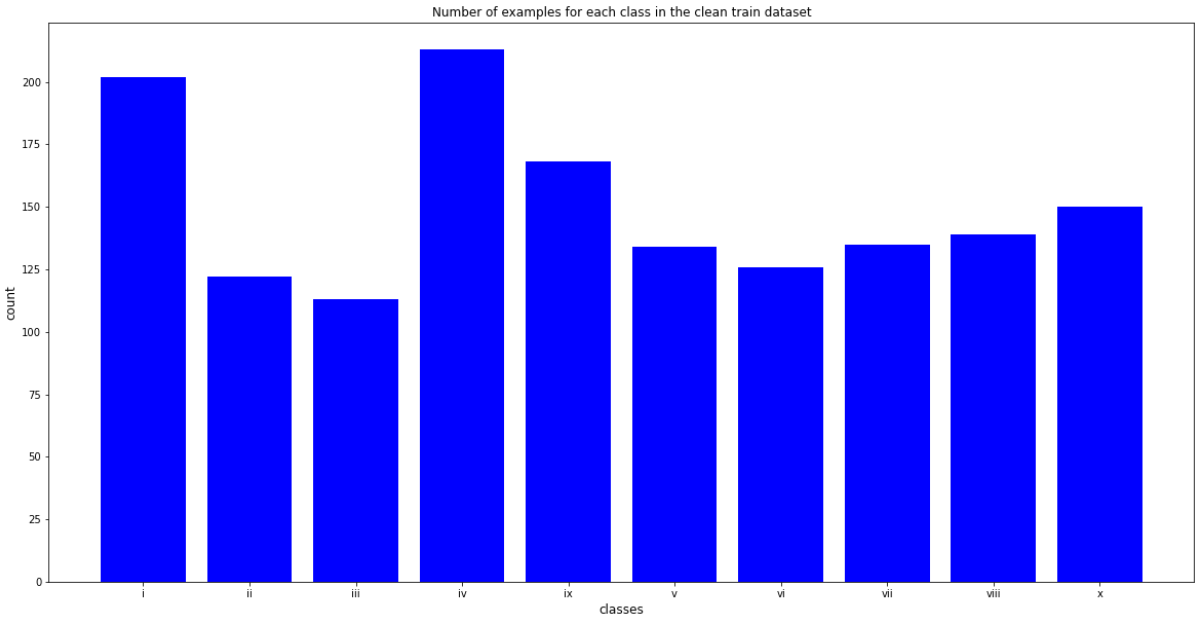
This image in "ix" (9) directory but it also could be "iv" (4).

After we cleaned all data (the combined original train and validation datasets), we split the remain data into train and validation with proportion of 80/20. We split every class by this proportion and in this way, we make sure the balance between train and validation across all classes is the same.

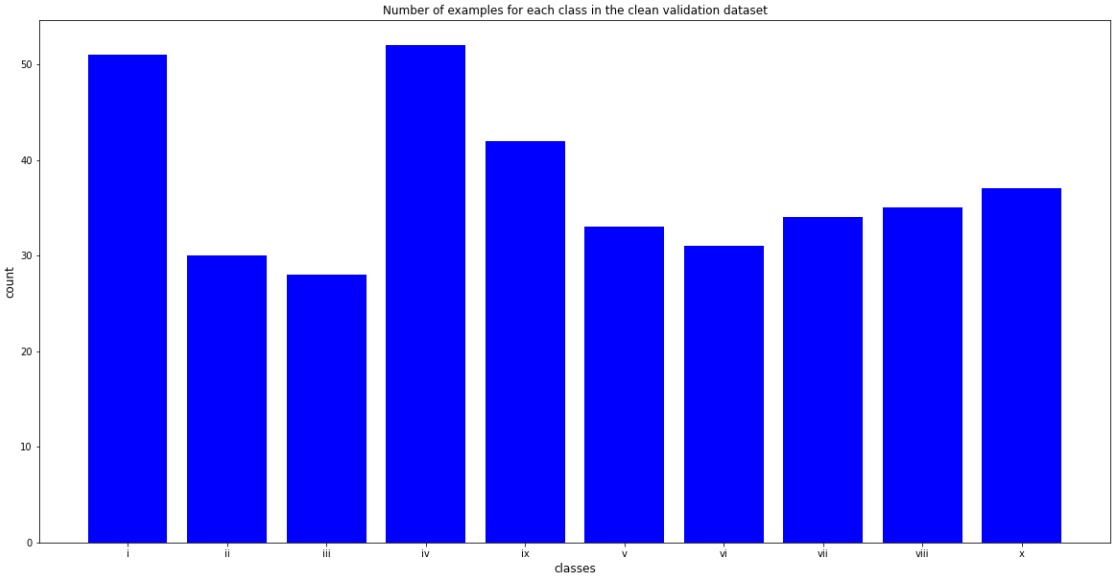
Cleaned Data Exploratory

The clean train set has 1502 examples, and the clean validation set has 373 examples.

The counts per class for clean train dataset are: $[202, 122, 113, 213, 168, 134, 126, 135, 139, 150]$.



The counts per class for clean validation dataset are: $[51, 30, 28, 52, 42, 33, 31, 34, 35, 37]$.

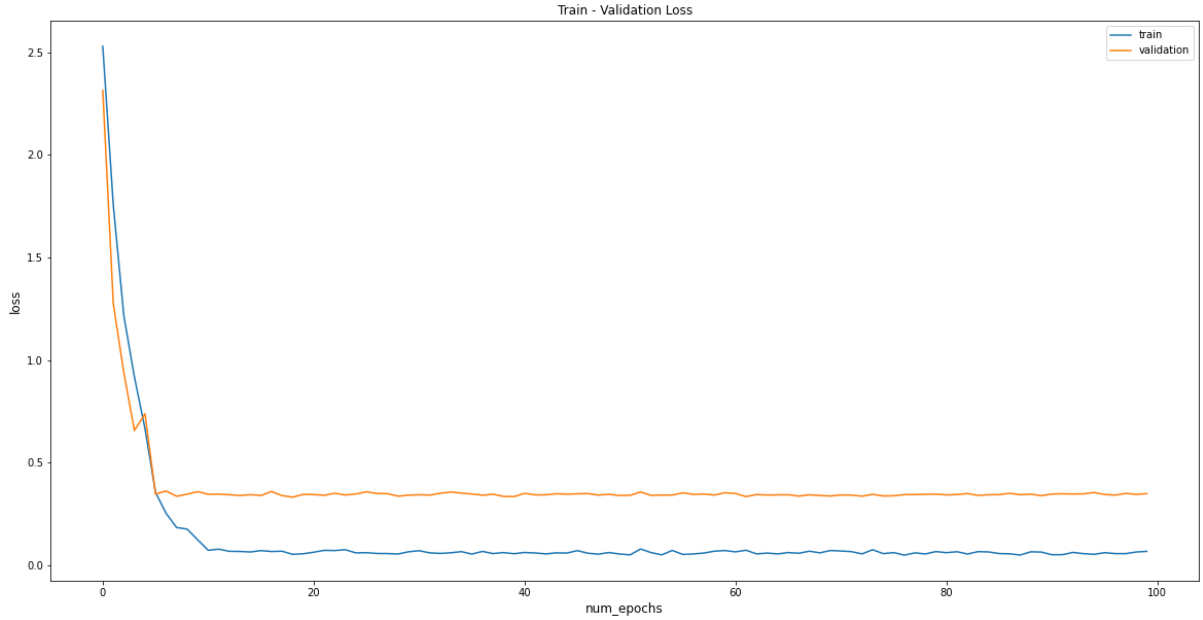


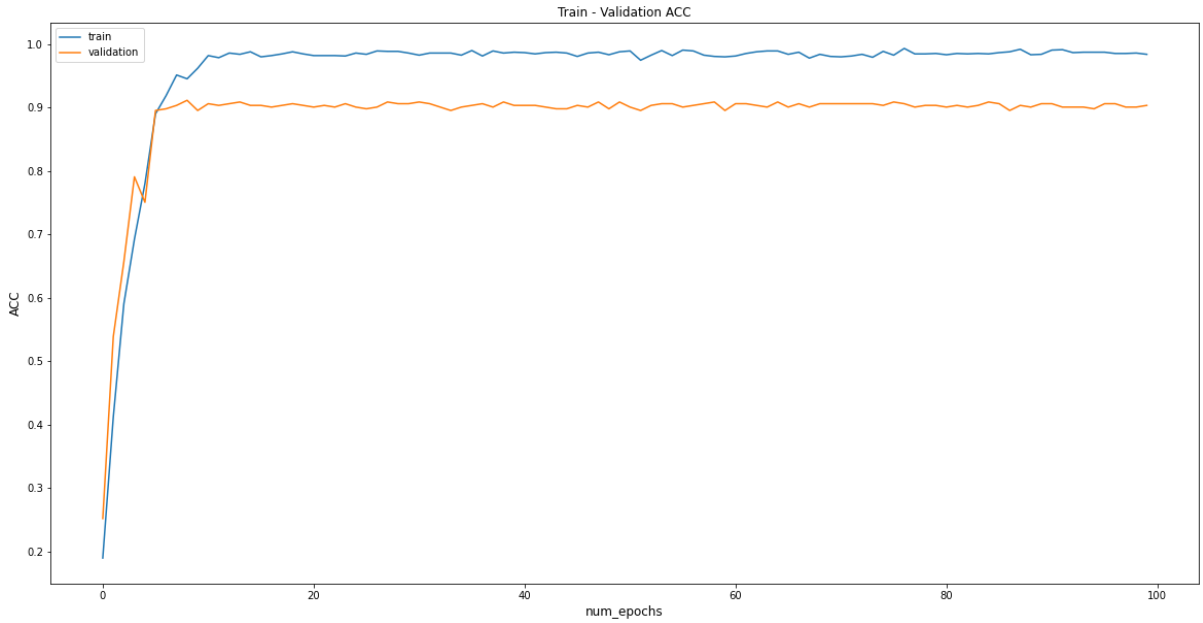
In this point we run the given code for evaluate the model that obtained on the clean data.

The result is:

```
Training complete in 14m 6s
Best val Acc: 0.911528
```

We can see that just removing the images that didn't contain numbers and fixing the labels improve the performance in 6%!



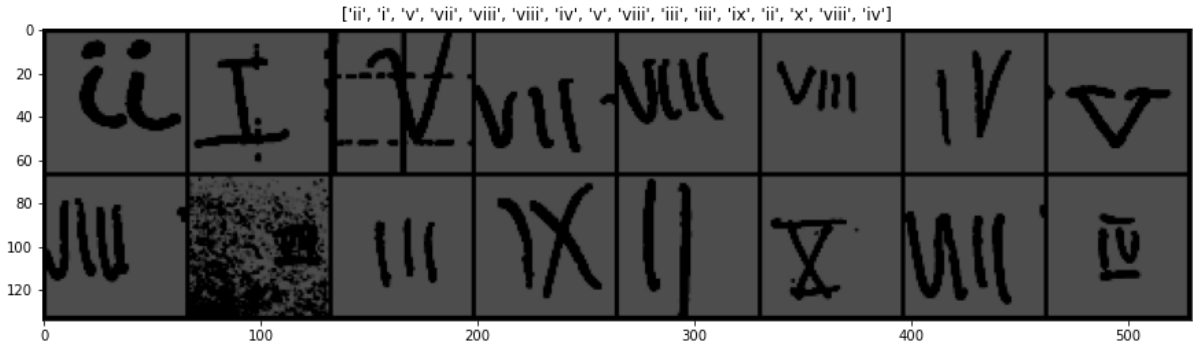


We can see in the graphs that the trend is the same as in the baseline.

Data Normalization

Now we tried to improve our results by normalized the data. We calculated the mean and the std of our train set and normalized the train and the test sets.

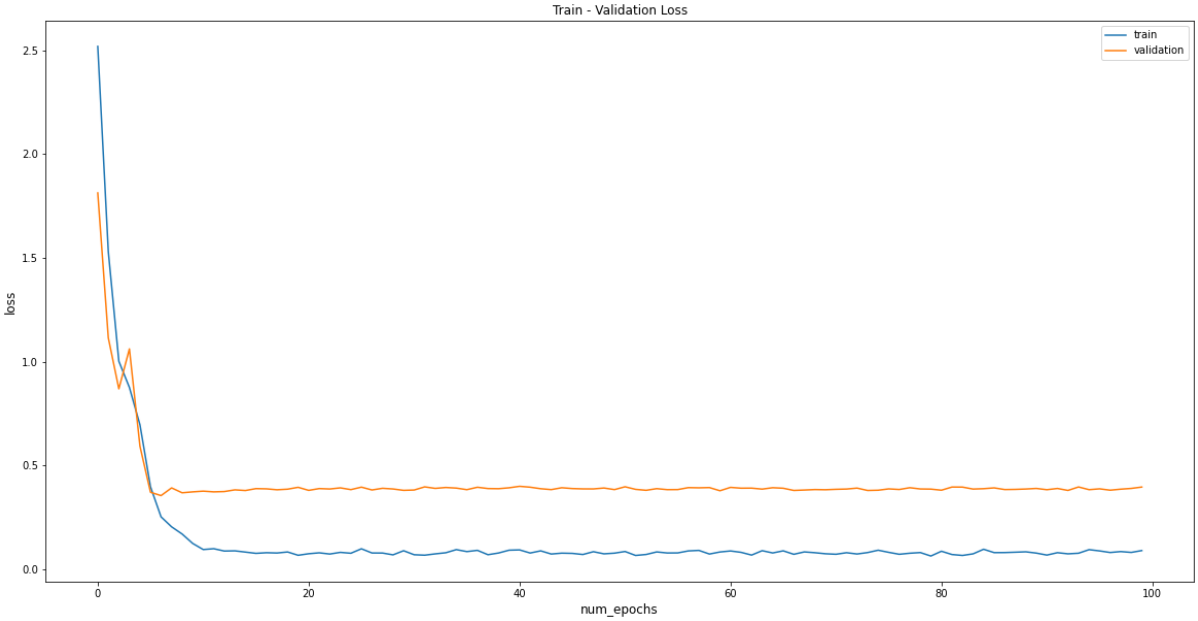
Sample from the data after normalization:

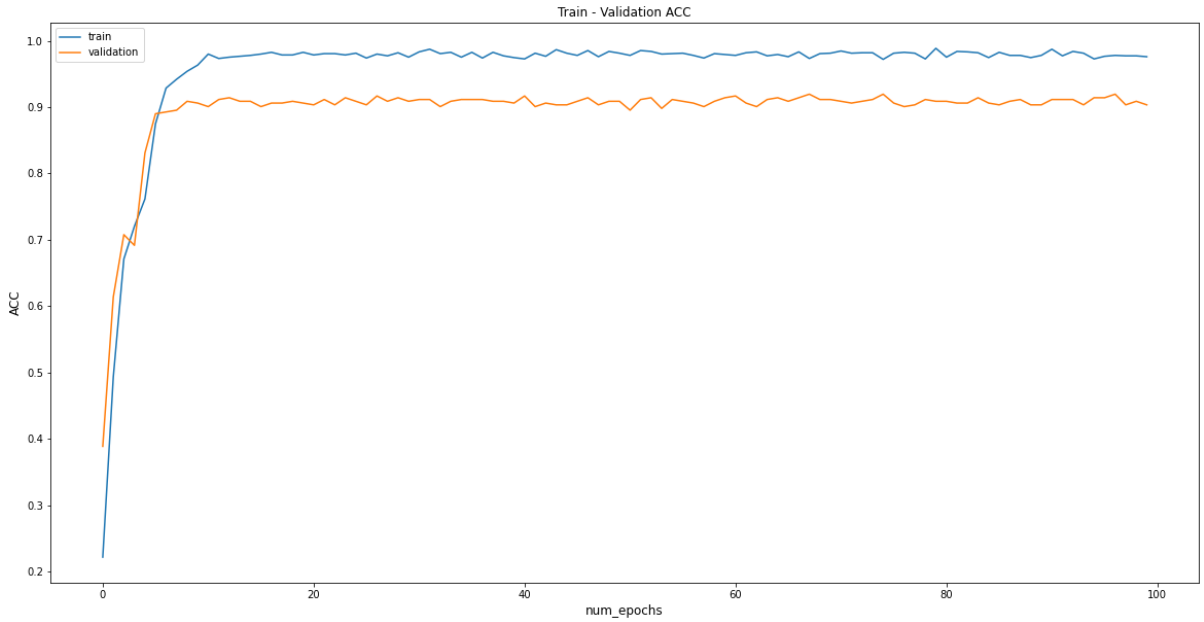


The result is:

```
Training complete in 14m 53s
Best val Acc: 0.919571
```

We can see a little improvement after adding the normalization.





Data augmentation

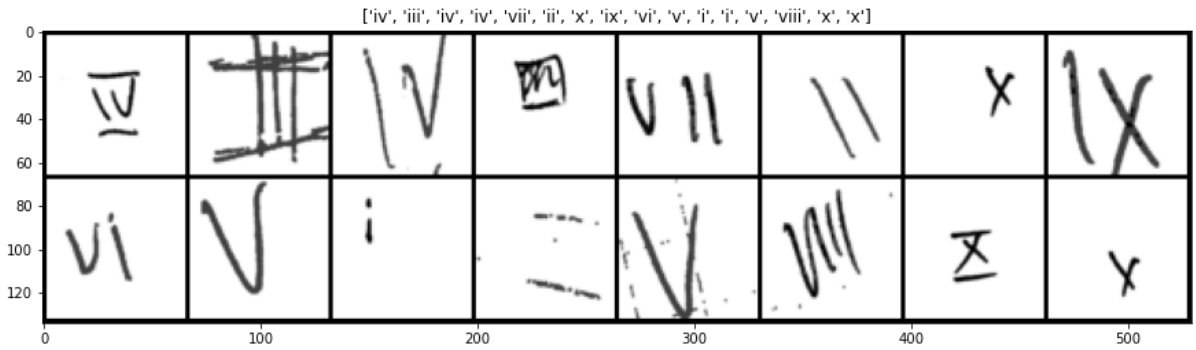
We decided to add a rotation to our train set. The rotation: We randomly choose the sign of the rotation and then we added or subtracted 15 degrees depending on the sign. Because this action doubled now our train set the proportion between train and validation is about 90/10.

The new sizes are:

```
The length of the rotated train dataset is: 3006
The length of the validation dataset is: 372
```

We calculated again the mean and std of the new train set for normalization.

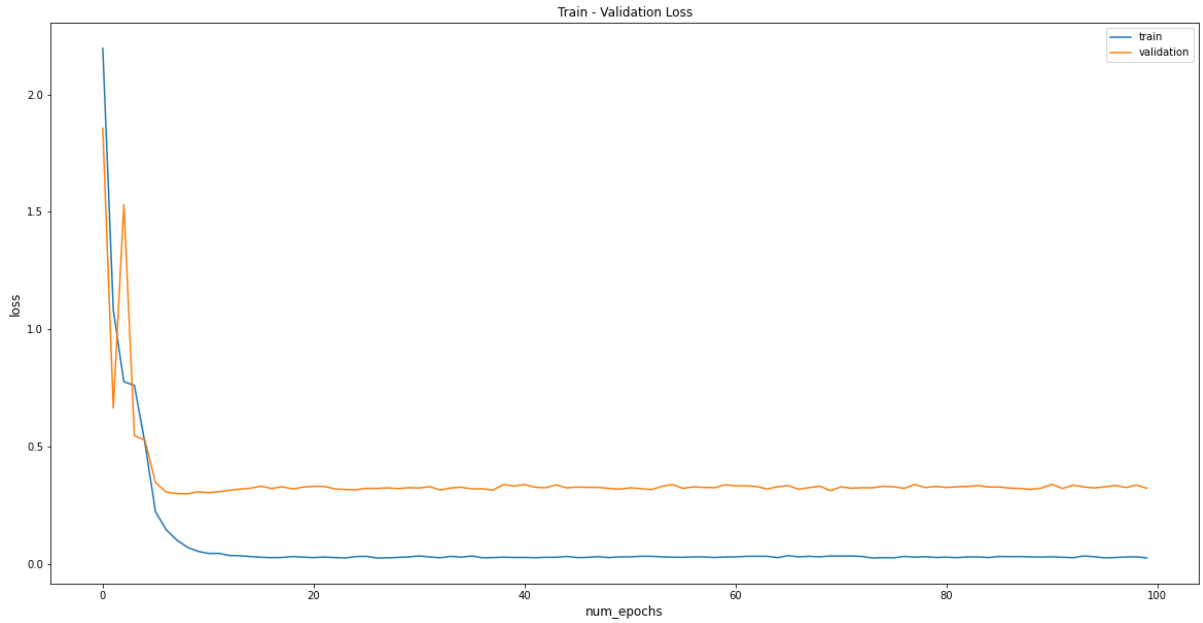
Sample of the data after rotation:



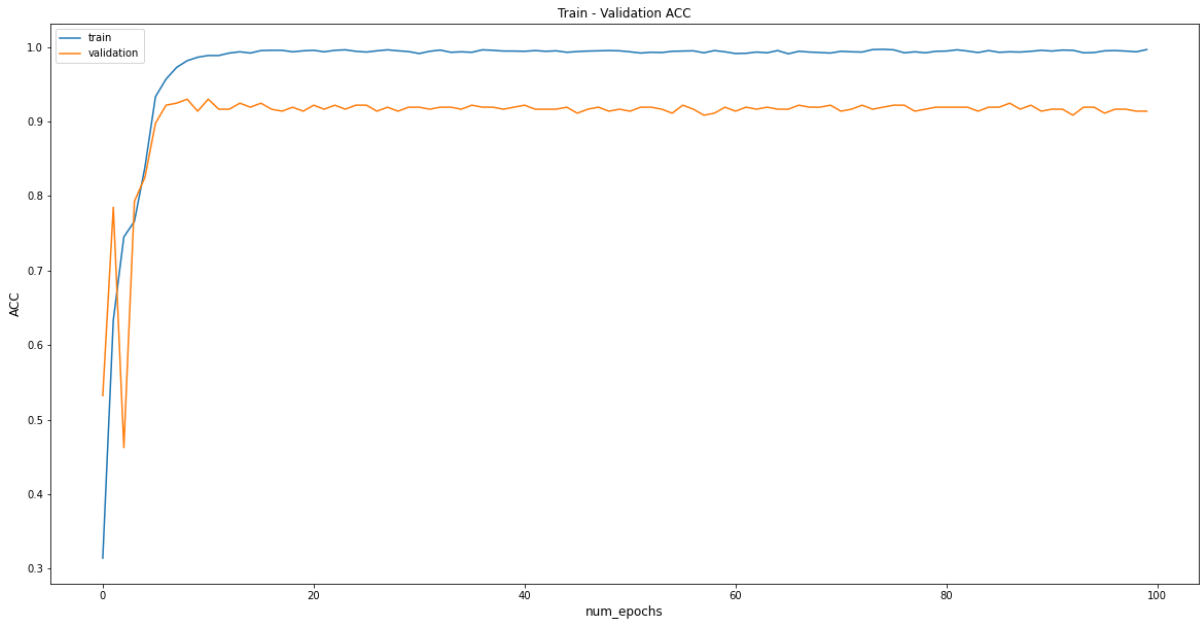
The result is:

```
Training complete in 26m 37s
Best val Acc: 0.930108
```

We see additional improvement.



We can see in the validation loss that we have some jumping around epoch 5.



Here we see the opposite for the loss, that around epoch 5 the validation accuracy has dividing.

The chosen data is the clean data after our splitting and rotation over the train set.