Executive summary

Created model (CNN with residual blocks) is classifying letters (A-Z) and numbers (0-9). On validation set (15% of images from original dataset) it achieved accuracy: 0.941 and f1 score: 0.878.

The given dataset is unbalanced. I have decided to increase the number of examples in some classes, using augmentation methods.

Method description

Data preprocessing: Provided dataset is unbalanced. There is one class that has only 1 picture, 17 in range (1, 425> and 18 in range <548, 3291>. The dataset was divided into two disjoint sets training and valid. To increase the number of training examples in classes, that have less pictures than 450 but more than 1 I have used, described below, augmentation methods. After this process our dataset consists of 43447 pictures.

Classifier is a Convolution Neural Network with Residual Blocks. It uses ReLU as activation function and additionally batch normalization is present in residual blocks. It has following structure:

```
Sequential(
  (0): Conv2d(1, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (1): ReLU()
  (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (3): ResidualBlock(
    (conv2d1): Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (conv2d2): Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (bn1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (bn2): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (activation_fn): ReLU()
  )
  (4): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (5): ReLU()
  (6): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (7): ResidualBlock(
    (conv2d1): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (conv2d2): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (bn1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (bn2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (activation_fn): ReLU()
  )
  (8): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (9): ReLU()
  (10): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (11): Flatten()
  (12): Linear(in_features=3136, out_features=500, bias=True)
  (13): ReLU()
  (14): Linear(in_features=500, out_features=36, bias=True)
)
```

Data augmentation:

To increase the number of images in some classes I have used rotation by 25°, -15° of images and shearing by radius 0.3 and -0.3. In my opinion the transformations are safe (transformed picture is not going to resemble image from a different class)

Used libraries:

The model is written in Pytorch. Images transformations are implemented using skimage library.

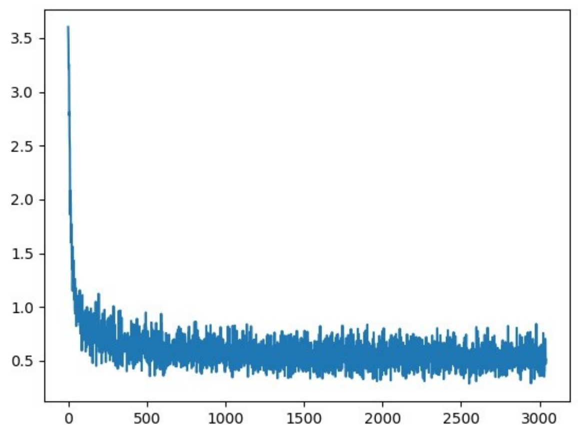
Implementation details:

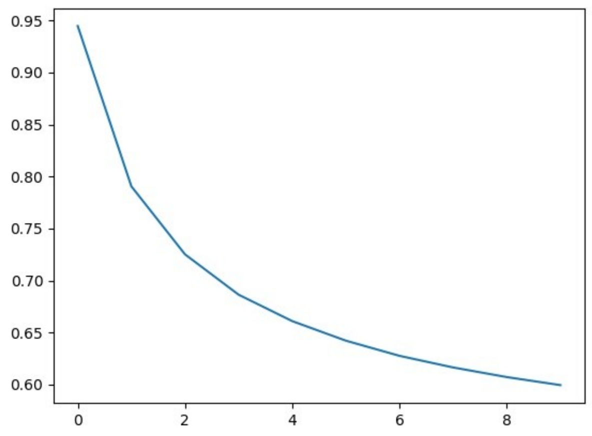
As a **loss function** I have used cross entropy loss.

Training details

I have splitted given dataset into two disjoint sets (training, valid). The examples were randomly selected into one or the other dataset so that the validation set consists of 15% of images from the original dataset.

On the following image learning curve (left) and graph presenting loss average after each epoch (right) is presented:





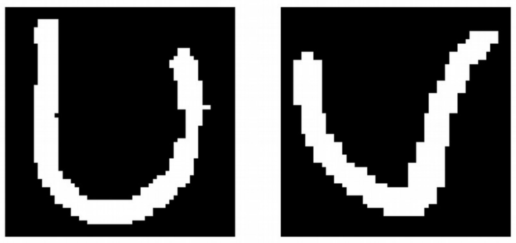
I have not used automated hyperparameters search.

Error analysis

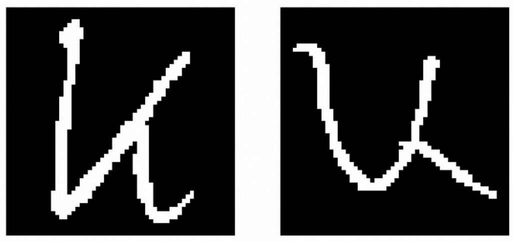
To validate my model I have used both accuracy and f1 score (computed as a average of f1_scores for each class). The best model achieved the accuracy: 0.941 and f1 score: 0.878.

The values of accuracy and f1 score indicate that the model is not focusing only on 1 most numerous class, but is capable of distinguishing classes and making correct predictions.

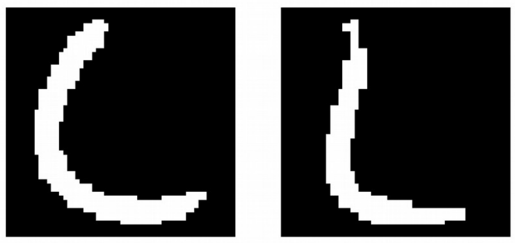
The examples of wrong predictions on validation set: (Image on the left presents a letter/number that has been incorrectly classified. Image on the right presents an example from class that should have been given.)



Letter V has been classified as letter U.



Letter K has been classified as letter U.



Letter L has been classified as letter C.

In my opinion it's difficult to unambiguously classify presented misclassified images.

Conclusions and recommendations:

The overall performance of the model is satisfactory. But in order to increase its performance I would try to (first 3 are the most important):

1. increase the number of images, either by finding another dataset consisting of images of letters and numbers or by using more augmentation methods.

2. run automated hyperparameters search, like gridsearch or use some bayesian methods

3. get more examples from class 30

4. train 3 models with different parameters and use them as an ensemble model