Base model

I used Pytorch framework for building the model. Pictures on SVHN and MNIST datasets differ in color and size. The following transformations:

- for MNIST: size increased to 32x32, transferred to 3 channels and normalized
- for SVHN: converted to shades of gray and normalized

The network consists of six layers (not including input and output): convolution (x4), union (x2). Model parameters: learning rate 10^{-3} , the batch size is the same for training and testing. The full structure of the model is presented below.

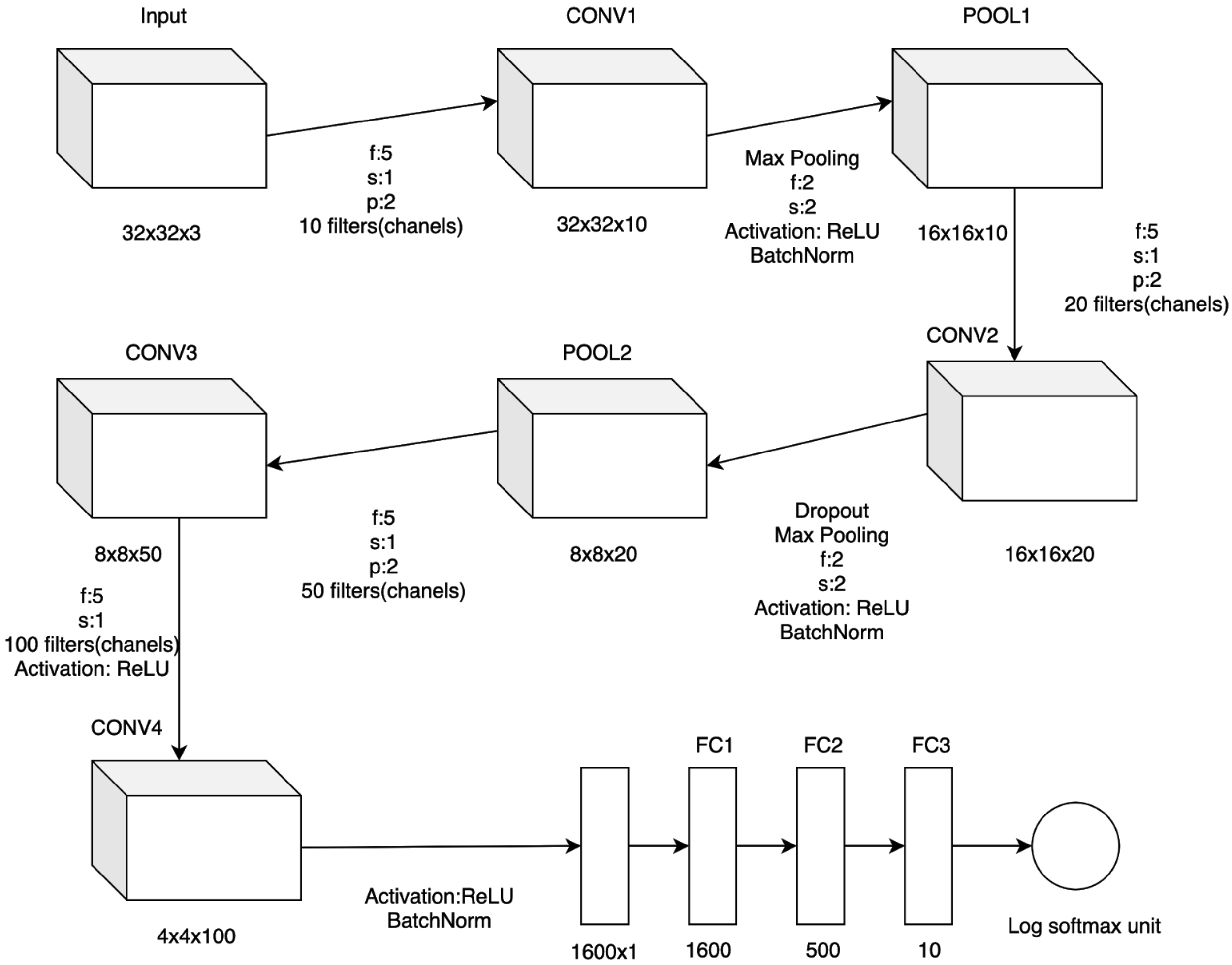

Figure 1. Full Model structure

To train the model, I used svhn-train and calculated the accuracy of the prediction on each of the following datasets: svhn-train, svhn-test, mnist-test. I used AdamW as an optimizer. The accuracy data are summarized in table #1.

Additionally, the following types of activations on the intermediate layer were tested (on previous version of the model): relu, relu6, elu, selu, celu, leaky_relu, rrelu, softplus, gelu, logsigmoid, hardshrink, tanhshrink, softsign, softmin, softmax, softshrink, gumbel_softmax log_softmax, tanh, sigmoid. Accuracy data for each type of activation are summarized in table #2. Several other types of optimizers were also tested: Adadelta, Adagrad, Adam, Adamax, RMSprop, Rprop. Accuracy data for each optimizer are summarized in table #3. The best activations (and most popular) for this tas are leaky_relu, relu. The best optimizers are Adam, AdamW, RMSprop.

Batch size can imply model in next manner: if a batch is too small your model will too much rely on certain examples and variance will be big. Otherwise, big number of elements in batch make the model unable to see hidden patterns due to big normalization and increase the bias as well. This is why batch size equal 64, for example, is a good choice.

Dropout is a good way of normalization and making the model not to rely on special features of certain examples and bathes but finding the needed level (probability of applying) may be hard. For example, if we chose the 50% probability (default value) accuracy will be 62.5 and both other options – increasing (75%) and decreasing (25%) – will show worse results: 53.03 and 60.5 respectively.

Table 1. Accuracy on each of the datasets

Dataset	Accuracy (%)
svhn-train	97.39
svhn-test	91.99
mnist-test	70.48

Table 2. Accuracy for each type of activation (on the mnist-test dataset)

activation	hardtanh	relu	relu6
accuracy	45.8	62.7	59.0
activation	rrelu	softplus	gelu
accuracy	55.45	59.4	53.2
activation	softmin	softmax	softshrink
accuracy	46.75	45.32	11.3
activation	celu	leaky_relu	tanhshrink
accuracy	59.65	59-62	55.32
activation	sigmoid	elu	selu
accuracy	46.43	59.71	57
activation	logsigmoid	hardshrink	gumbel_softmax
accuracy	43	49.87	33.18
activation	log_softmax	softsign	tanh
accuracy	39.43	48.57	45.83

Table 3. Accuracy for each optimizer (on the mnist-test dataset)

Optimizator	Accuracy (%)
Adadelta	60.5
Adagrad	61.0
Adam	62.3
Adamax	62.26
RMSProp	63.73
Rprop	19.05
SGD	61.23