Homework 1: Supervised Learning

Neural Networks and Deep Learning

Cracco Gianmarco

November 2020

Introduction

The goal for this homework will be to implement several neural networks (NN) to correctly accomplish both regression and classification tasks.
The first one consists of approximating an unknown function f:

$$f : \mathbb{R} \to \mathbb{R}$$
$$x \mapsto y = f(x)$$
$$x \mapsto \hat{y}(x) \approx f(x)$$

The classification task instead consists of correctly classify the data from the well-known MNIST handwritten digit dataset. This last problem will be tackled using both Feed Forward NN and Convolutional NN.

1 Regression Task

1.1 Dataset

The dataset provided is separated in two different `.csv` files and their shapes are the following:

	Rows	Columns
Train	100	2
Test	100	2

As training points, noisy output from the target function are provided: in Fig. 1 a plot of them is reported in order to obtain a first look at the data.

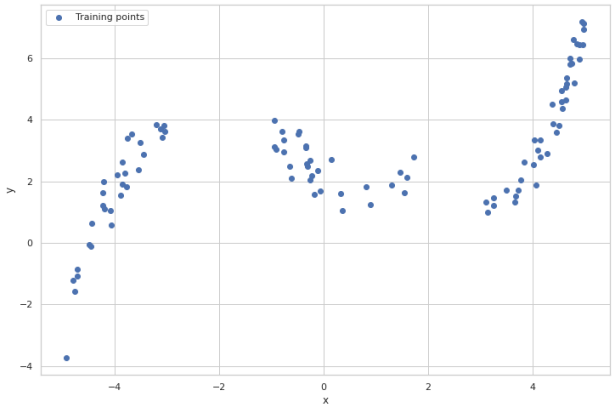


Figure 1: Training points plotted

1.2 Model Initialization & Hyperparameters Search

The NN implemented is composed as following:

INPUT LAYER
⇓
HIDDEN LAYER 1 + ReLU
⇓
DROPOUT LAYER 1
⇓
HIDDEN LAYER 2 + ReLU
⇓
DROPOUT LAYER 2
⇓
HIDDEN LAYER 3 + ReLU
⇓
DROPOUT LAYER 3
⇓
OUTPUT LAYER

where each layer is a fully connected one. The input and output size are both equal to 1 since our target function is $f : \mathbb{R} \to \mathbb{R}$.
As initialization scheme, Xavier distribution has been chosen in order to improve stability during training while, the loss function is the Mean Square Error (MSE).
For what concerns hyperparameters tuning, a grid search has been implemented combined with a k-Fold Cross Validation over 5 folds (the resulting losses are

then averaged across the folds in order to obtain a mean value) while being careful of re-initializing the network at the beginning of each fold; both these steps were feasible given the small size of the dataset.
The grid search has been split into 2 steps to meet the lack of resources: the first one was dedicated to the numbers of neurons, while the second on the remaining hyperparameters:

- Number of units for 1^{st} layer: `list(np.random.randint(50, 250, size=6))`
- Number of units for 2^{nd} layer: `list(np.random.randint(50, 250, size=6))`
- Number of units for 3^{rd} layer: `list(np.random.randint(50, 250, size=6))`
- Optimizer: `[Adam, RMSprop]`
- Learning rate: `[0.001, 0.005]`
- Dropout probability (different for each dropout layer): `[0, 0.2]`
- L2 normalization : `[0, 1e-3, 1e-5]`
- Epochs : `400`

After the end of the search, the best combination of hyperparameters has been chosen to select the one with the lowest loss on the validation set. Here we report the first 5 models:

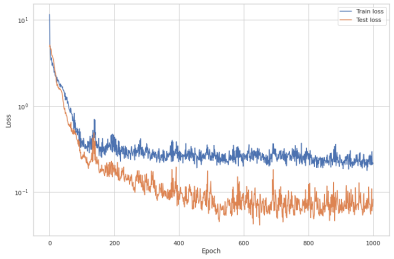
Nh_1	Nh_2	Nh_3	Optimizer	Epochs	Learn. Rate	Prob_1	Prob_2	Prob_3	L2	Loss
113	235	192	Adam	400	0.005	0.0	0.2	0.0	1e-5	0.301
113	235	192	Adam	400	0.001	0.0	0.0	0.0	1e-5	0.312
113	235	192	Adam	400	0.005	0.0	0.2	0.0	1e-3	0.316
113	235	192	Adam	400	0.001	0.0	0.0	0.0	1e-5	0.320
113	235	192	Adam	400	0.005	0.0	0.0	0.2	1e-5	0.320

Table 1: Results of first 5 best models.

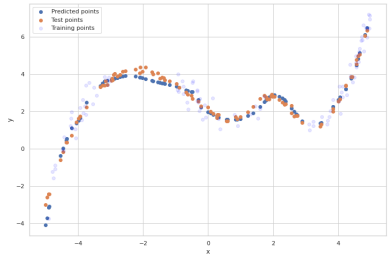
In Appendix A.1 is reported the plot of losses decay for the best model (Fig. 6).

1.3 Final Model & Results

Finally, once the hyperparameters have been selected, a final train has been executed, this time using all the training set and evaluating on the test one, increasing the number of epochs to 1000.
We report the plot of the losses, the final MSE, and the plot of the predicted test data versus the ground truth ones.
In Appendix A.1, is reported a histograms of weights' distributions (Fig. 7) and the activation profile for all the hidden layers using as input a single sample (Fig. 8).



(a) Final train losses.



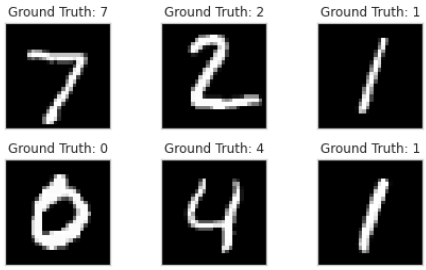
(b) Predicted vs Groud Truth data.

$$\text{MSE} = 0.082$$

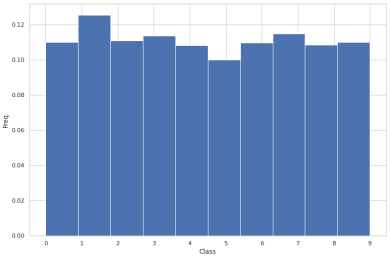
2 Classification Task

2.1 Dataset

For this task, the MNIST dataset has been selected. It consists of 28×28 pixels images of handwritten digits between 0 and 9. The entire dataset is composed of 60000 samples that are then divided into training set (48000 samples) and test set (12000 samples).
Here we report the representation of few samples in addition to the plot of samples distribution:



(a) Training samples.



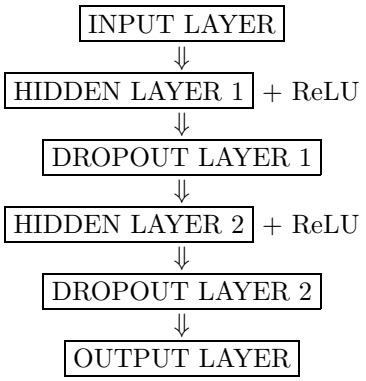
(b) Samples distribution on training set.

Since all samples are evenly distributed across all classes, no further data preprocessing is needed.

2.2 Feed Forward NN

2.2.1 Model Initialization & Hyperparameters Search

The NN implemented is composed as following:



Where the input layer has $28 \cdot 28$ units and the last one 10, just like the number of classes. Again the weights of all layers have been initialized to Xavier distribution while the loss, in this case, is the Cross-Entropy.
Given the size of the dataset and the limited resources, the k-fold cross-validation has been ditched in favor of the less computational demanding simple train-validation split ($80\% - 20\%$) and the grid search has been carried out in the same two phases as in Sec. 1.2.
The sets of hyperparameters are the following:

- Number of units for 1^{st} layer: `list(np.random.randint(100, 300, size=6))`
- Number of units for 2^{nd} layer: `list(np.random.randint(100, 300, size=6))`
- Optimizer: `[Adam, RMSprop]`
- Learning rate: `[0.001, 0.005]`
- Dropout probability (different for each dropout layer): `[0, 0.2]`
- L2 normalization: `[0, 0.1, 0.2]`
- Epochs: `100`

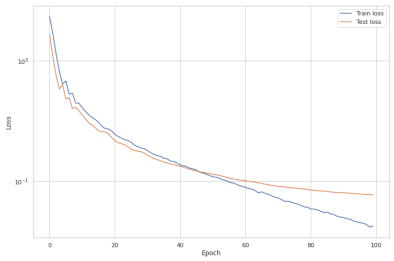
As previously, here we report a table with the first 5 combinations while in Fig 9 can be found a plot containing loss decay during training:

Nh_1	Nh_2	Optimizer	Epochs	Learn. Rate	Prob_1	Prob_2	L2	Loss
277	206	Adam	100	0.005	0.2	0.0	0.0	0.078
277	206	Adam	100	0.005	0.0	0.2	0.0	0.078
277	206	Adam	100	0.005	0.2	0.2	0.0	0.080
277	206	Adam	100	0.005	0.0	0.0	0.0	0.088
277	206	RMSprop	100	0.005	0.0	0.0	0.0	0.124

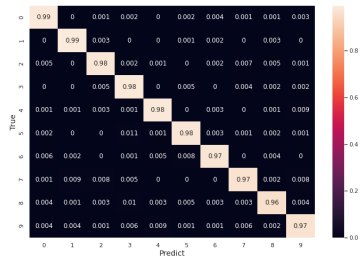
Table 2: Results of first 5 best models.

2.2.2 Final Model & Results

Like preiously, the best combination has been selected and the net retrained using the entire training dataset leading to an accuracy equal to 97.68% on the test set:



(a) Final train losses



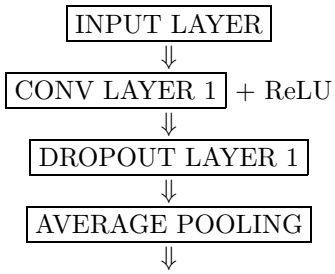
(b) Confusion matrix.

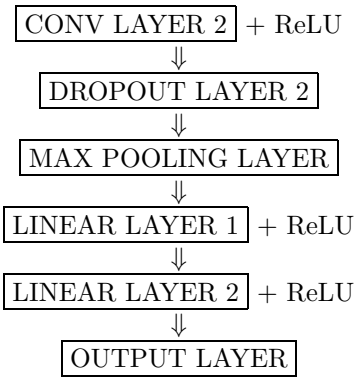
In Appendix A.2, we report histograms of weights distributions (Fig. 10) and the activation profile for the hidden layers for one sample (Fig. 11).

2.3 Convolutional NN

2.3.1 Model initialization

The network is implementes as following:





where the hyperparameters related to the convolutional and pooling layers are the following:

- Conv layer 1: `Conv2d(in channels=1, out channels=6, kernel size=5, stride=1, padding=1)`;
- Conv layer 2: `Conv2d(in channels=6, out channels=12, kernel size=3, stride=1, padding=2)`;
- Pooling layer: `(kernel size=3, stride=2)`.

Proceeding in the same way as in Sec. 2.2.1, the sets of hyperparameters for the grid search are:

- Number of units for 1^{st} linear layer: `list(np.random.randint(50, 300, size=6))`
- Number of units for 2^{nd} linear layer: `list(np.random.randint(50, 300, size=6))`
- Optimizer: `[Adam, RMSprop]`
- Learning rate: `[0.001, 0.005]`
- Dropout probability (different for each dropout layer): `[0, 0.2]`
- L2 normalization: `[0, 1e-3, 1e-5]`
- Epochs: `50`

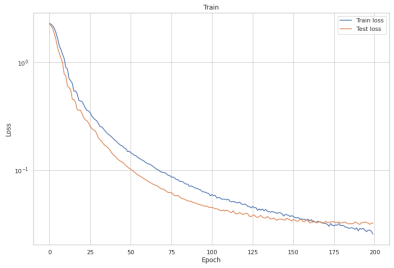
In Table 3 the results are reported while in Fig. 12 can be found the loss decay during the train with the best combination of hyperparameters.

Nh_1	Nh_2	Optimizer	Epochs	Learn. Rate	Prob_1	Prob_2	L2	Loss
227	121	Adam	50	0.005	0.2	0.2	0.0	0.125
227	121	Adam	50	0.005	0.0	0.2	0.0	0.127
227	121	Adam	50	0.005	0.2	0.0	0.0	0.128
227	121	Adam	50	0.005	0.0	0.0	0.0	0.128
227	121	RMSprop	50	0.001	0.2	0.0	0.0	0.241

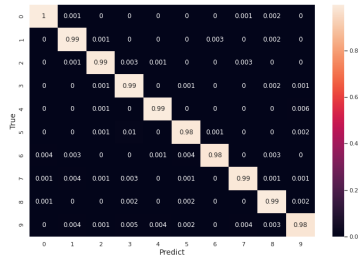
Table 3: Results of first 5 best models.

2.3.2 Final Model & Results

With the best configuration of hyperparameters, training on the entire training dataset and the number of epochsincreased to 200, the accuracy achieved is equal to 99.00% on the test set:



(a) Final train losses



(b) Confusion matrix

3 Conclusion

Both tasks have been accomplished with success. The feed-forward neural network for the classification task performs well although the input data are images is due to the fact that the size of them is still very limited (only 28×28 pixels) and so using fully connected layers is still feasible. As resolution increases the use of CNN becomes mandatory. Further improvement can of course still be made, like the addition of early stopping and data augmentation for the classification task. Instead the implementation of a learning rate scheduler wouldn't be correct since ADAM already changes it according to loss trend.

A Additional figures

Here we report additional figures to better understand the behaviour of our NNs.

A.1 Feed Forward NN for regression task

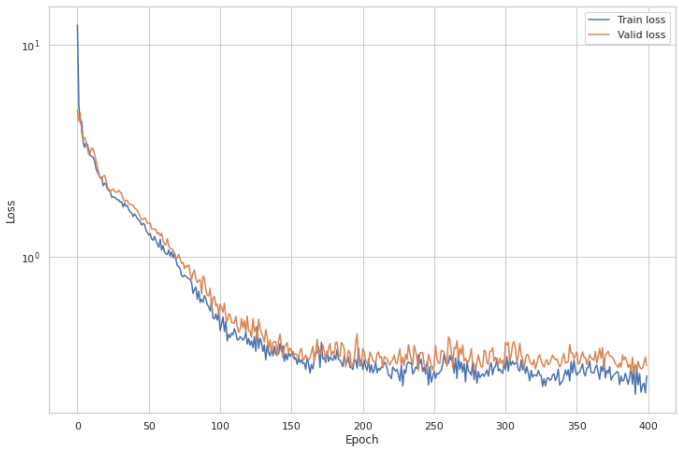


Figure 6: Loss decay during training with k-fold cross validation.

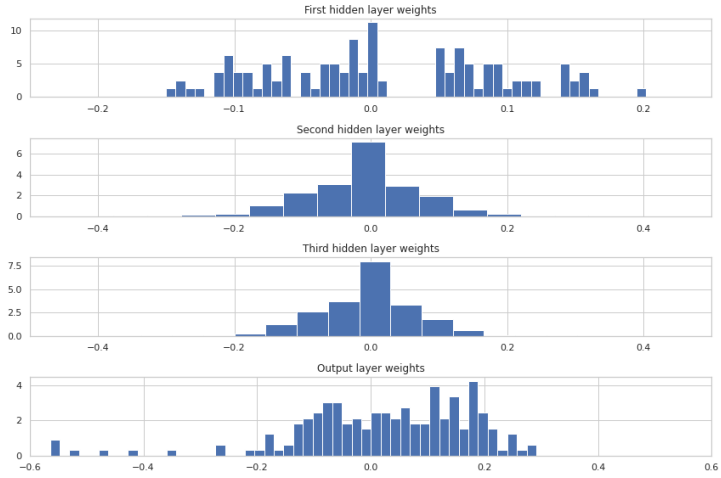


Figure 7: Weight distribution.

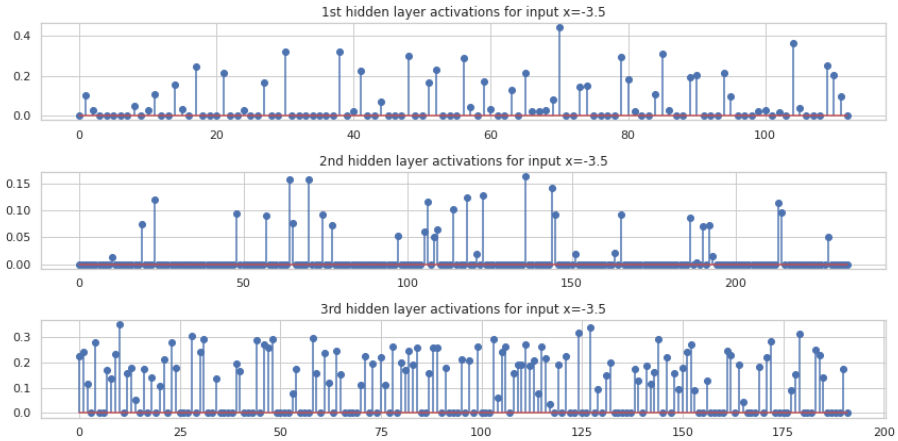


Figure 8: Activation profile of hidden layers.

A.2 Feed Forward NN for classification task

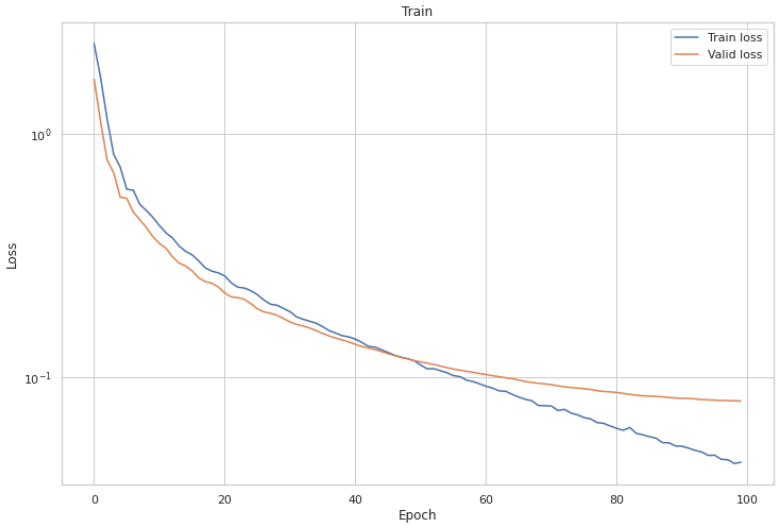


Figure 9: Loss decay during training.

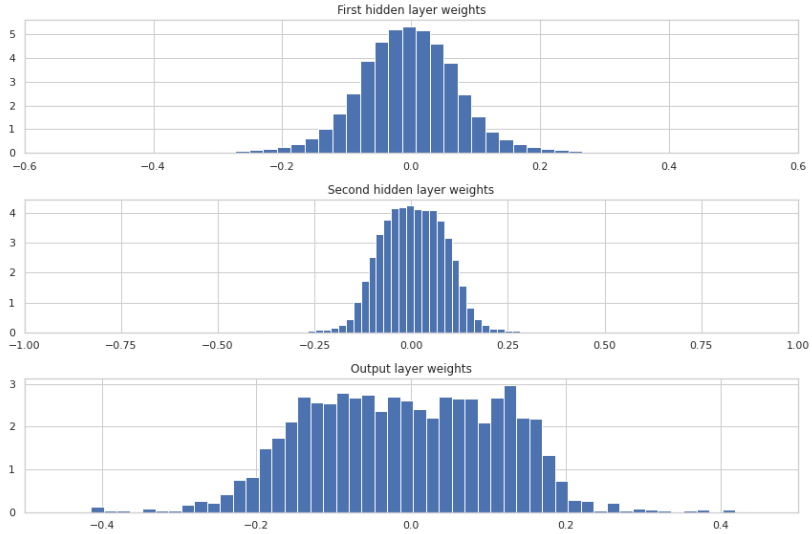


Figure 10: Weight distribution.

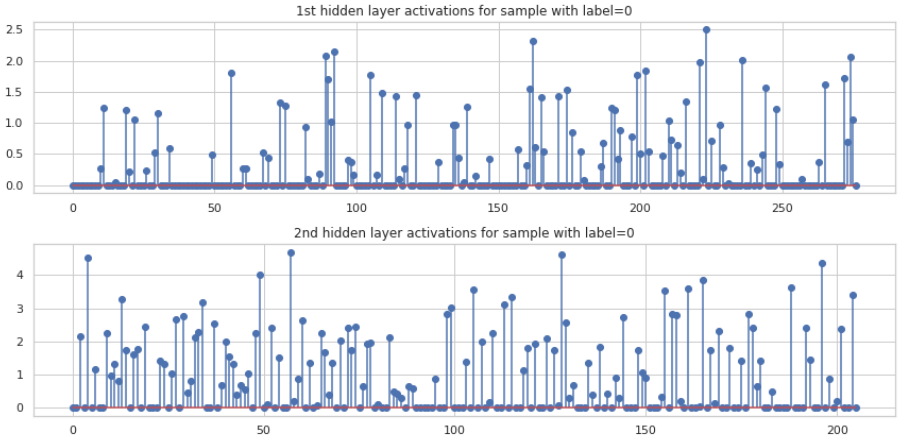


Figure 11: Activation profile of hidden layers.

A.3 Convolutional NN

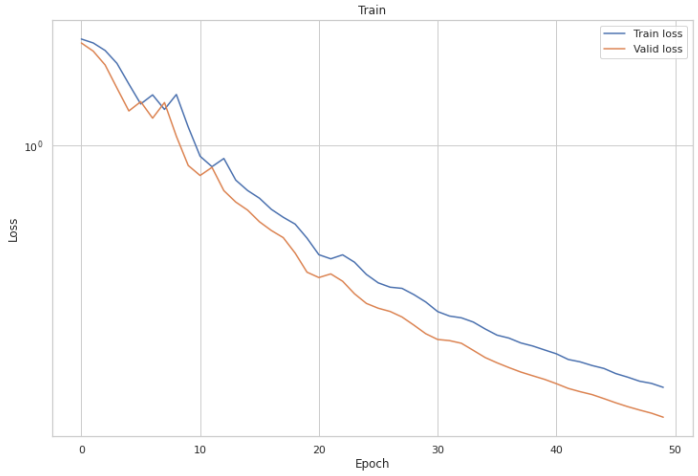


Figure 12: Loss decay during training.

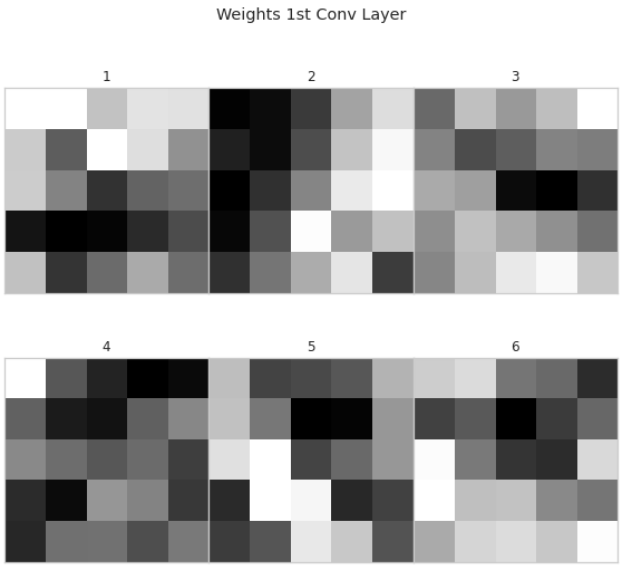


Figure 13: Weight distribution of 1st conv layer.

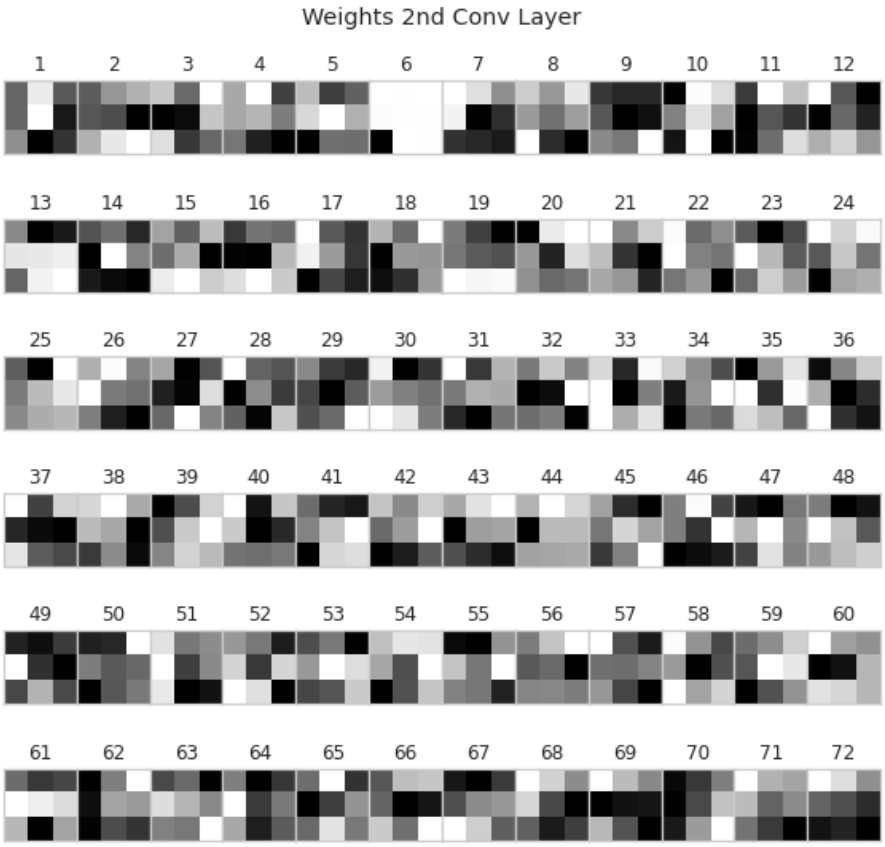


Figure 14: Weight distribution of 2nd conv layer.

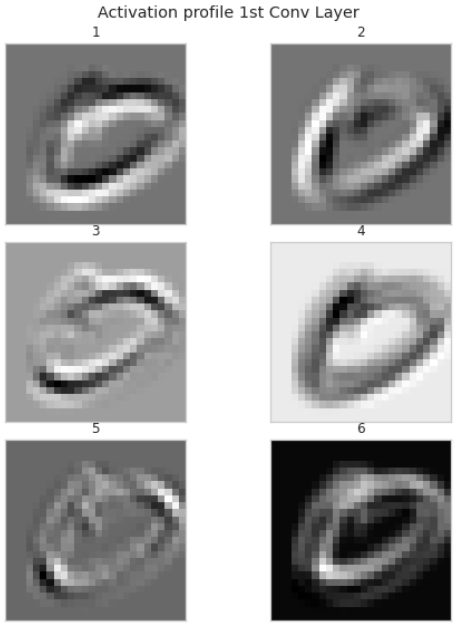


Figure 15: Activation profile of 1st conv layer.

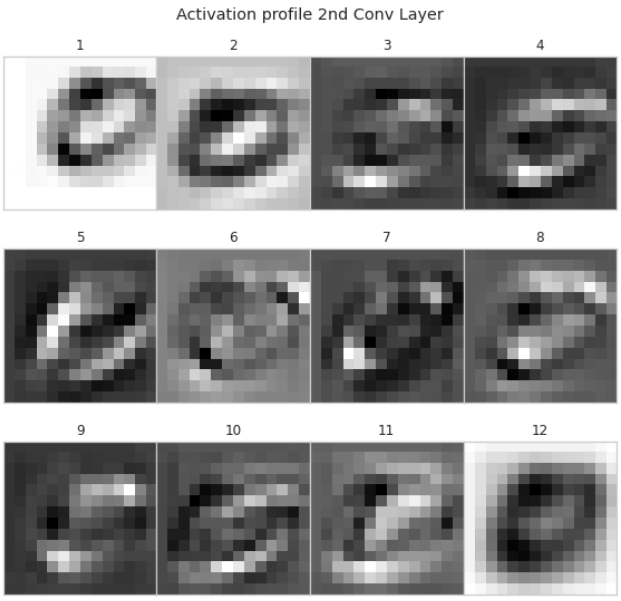


Figure 16: Activation profile of 2nd conv layer.