DLE Homework 4
Initialization and Regularization

Vojtěch Michal, michavo3

***Abstract*—The assigned[1] homework focuses on various strategies of neural network weight initialization and regularization. Several strategies for the initialization of weights of linear layers are implemented and compared. The homework further discusses the importance of the normalization of data on a shallow network and concludes with a demonstration of Dropout on the MNIST dataset.**

I. Initialization of Deep Networks

A fully connected neural network with 50 layers of 512 neurons each was implemented using the Torch library. In each tested case, model weights (all biases were forced to zero) were initialized using a selected strategy and the appropriate activation function was applied after each linear layer. Subsequently, a large batch of 10000 random vectors with zero mean and unit variance was fed through the network. Recorded means and standard deviations of individual layers' activations are visualized and compared for various initialization strategies in Fig. 1 and 2. Similarly Fig. 3 and 4 show means and standard deviations of gradients calculated during the backward step when using the cross-entropy loss on the neural network output.

Since the vertical axis uses logarithmic units, the magnitude of mean rather than the (possibly negative) mean itself is recorded. Standard deviations for some combinations of initialization strategy and activation function are not shown in Fig. 2 and 4. These combinations lead to total vanishing of either activations or gradients, resulting in zeros that can't be visualized on the logarithmic scale.

Intuitively the "best" initialization method ensures that the variance of layer activations and gradients stays as high as possible for as long as possible to ensure "richness" and prevent vanishing of terms to zero. Looking at Fig. 2, by far the highest standard deviation is held by Xavier uniform initialization with Tanh activation functions, a popular choice

[1]The homework assignment is available on https://cw.fel.cvut.cz/wiki/courses/bev033dle/labs/lab3_fromscratch/start

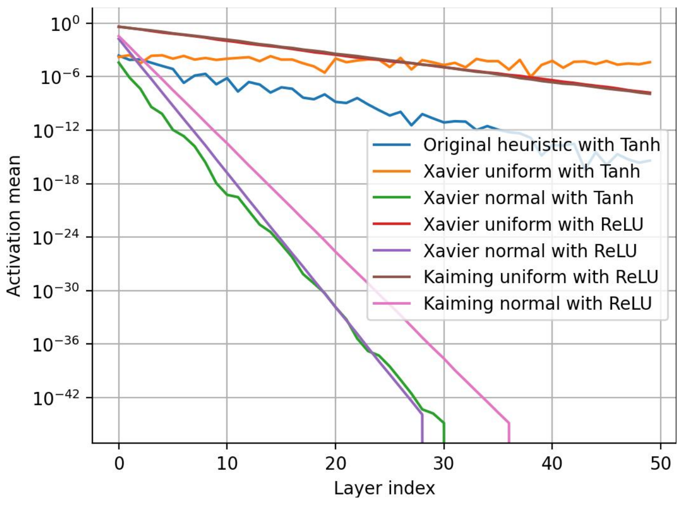

Fig. 1: Activation mean μ per layer for various initialization strategies

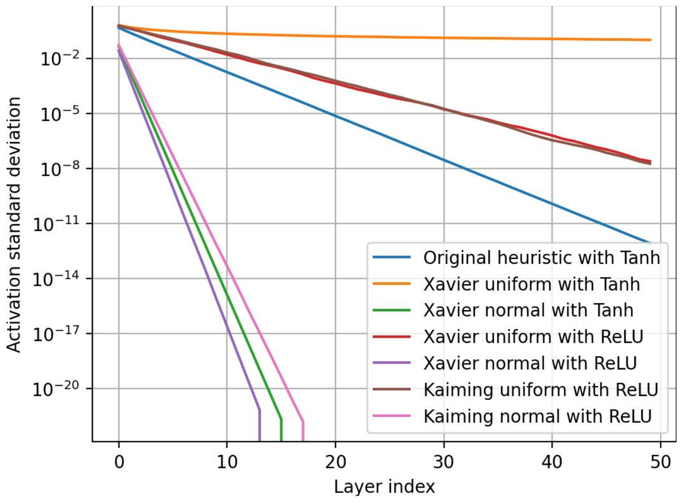

Fig. 2: Activation standard deviation σ per layer for various initialization strategies

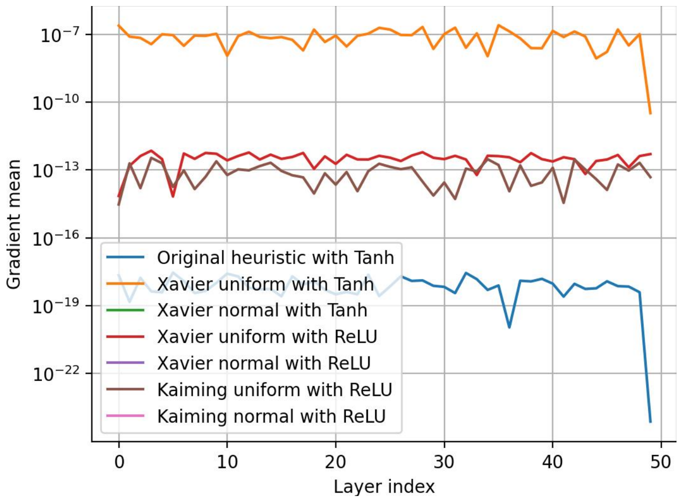

Fig. 3: Gradient mean μ per layer for various initialization strategies

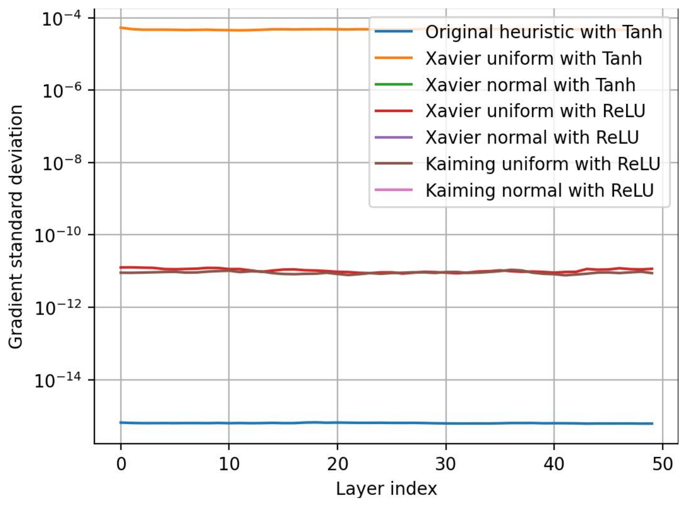

Fig. 4: Gradient standard deviation σ per layer for various initialization strategies

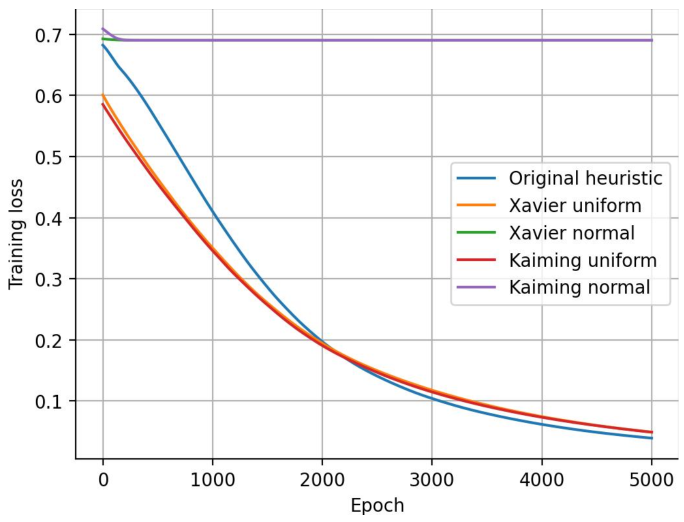

Fig. 5: Evolution of training loss for normalized data

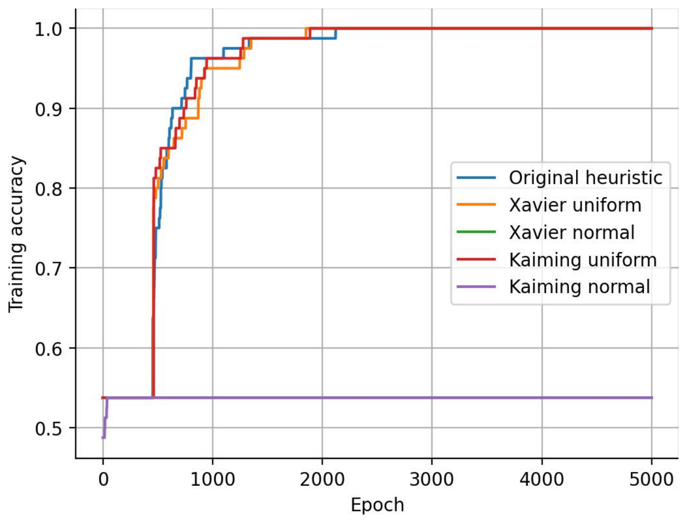

Fig. 6: Evolution of training accuracy for normalized data

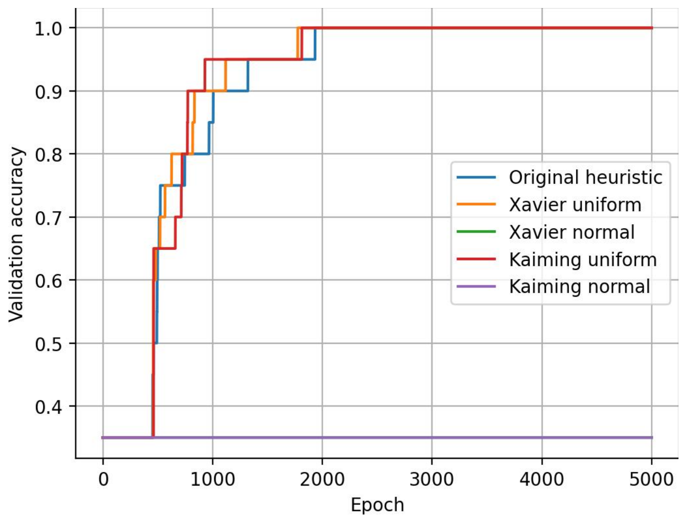

Fig. 7: Evolution of validation accuracy for normalized data

and a ”rule of thumb”. The second and third place belong to ReLU activation function with either the Xavier uniform or Kaiming uniform initialization. These combinations also achieve the best performance w.r.t. to the variance of gradients from the backwards pass, as shown in Fig. 4 and hence are preferable to other tested alternatives.

II. Initialization of a shallow network

A shallow neural network with 2 inputs, 2 hidden layers of size 6 and 3 and ReLU activation function was trained to classify two-dimensional data distributed along two circles with the same center at the point (10, 10). Three separate cases were tested:

1) Without data normalization, i.e. data stay centred around the point (10, 10),
2) With normalized data (zero mean and unit variance),
3) With zero mean data scaled down to achieve 0.01 standard deviation.

The training was carried out by the AdamW optimization algorithm with a learning rate of 0.0003 and batch size equal to the size of the whole training dataset, i.e. corresponding to classical Gradient Descent.

The gradual improvement of classification performance is illustrated in Fig. 5 through 7. Especially note that the training losses (normalized by the sample size) for most initialization strategies start at a value slightly below 0.7. This perfectly corresponds with $-\log 1/n \approx 0.69$, where n is the number of classes (here 2), that should be approximately equal to the cross entropy loss calculated on a randomly initialized model without learning.

An interesting observation is how badly the Kaiming normal initialization performed compared to all other algorithms. At first, this was assumed to be an error in the implementation, but even using the Kaiming normal initialization built into Torch yielded the same bad results, where the optimization settles in a local minimum and can’t improve the model further.

Successfully learned decision boundaries for classifiers with the highest success rate for individual cases are shown in Fig.

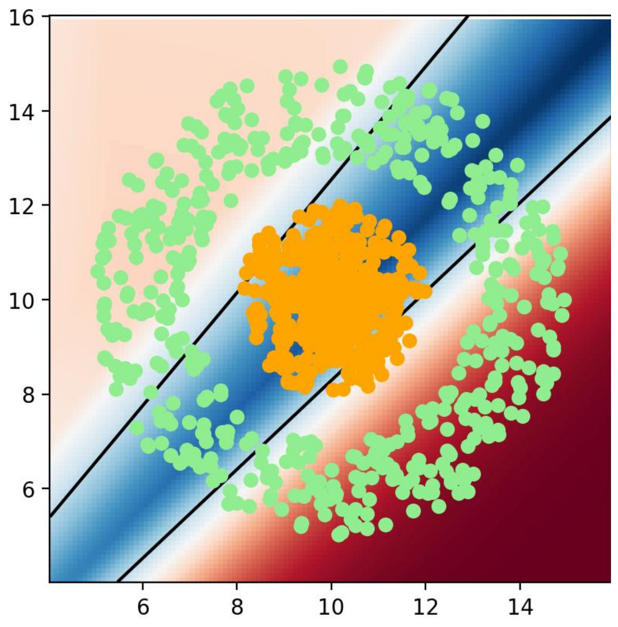

Fig. 8: Decision boundary for unnormalized data trained after Xavier normal initialization

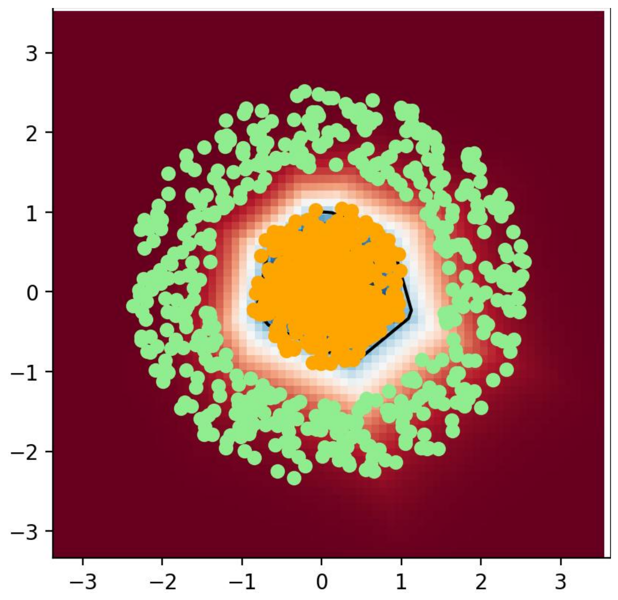

Fig. 9: Decision boundary for normalized data trained after Xavier uniform initialization

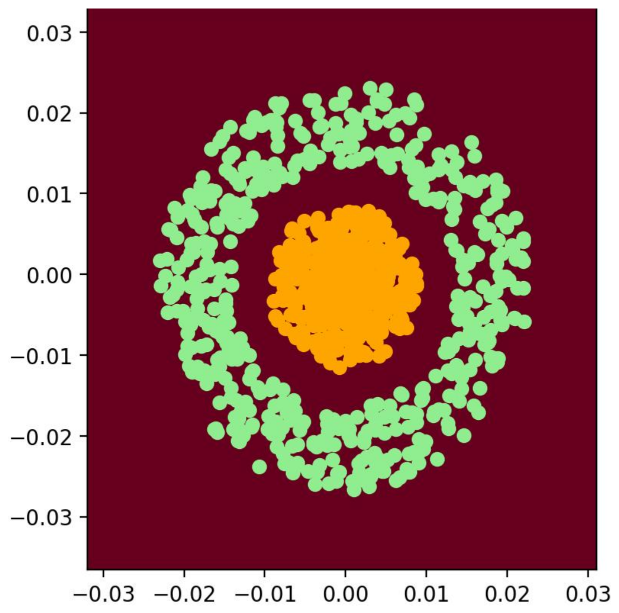

Fig. 10: Decision boundary for normalized downscaled data trained after Xavier normal initialization

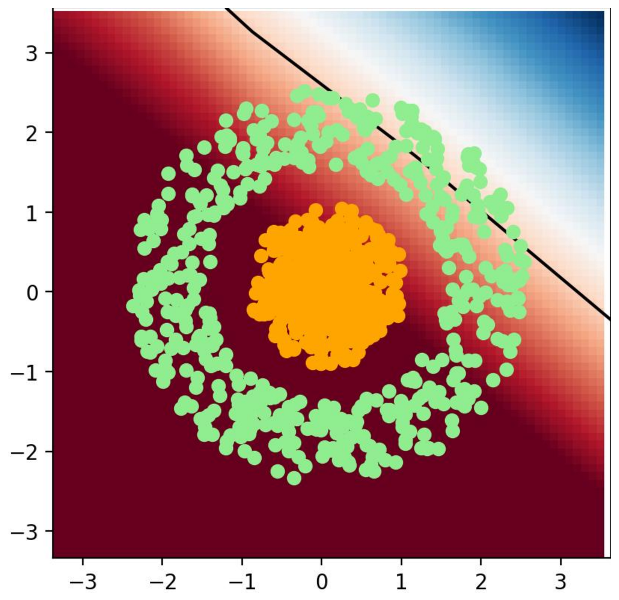

Fig. 11: Failed decision boundary for normalized data trained after Kaiming normal initialization

8 and 9. When data is not normalized (Fig 8), the model achieved at most 84.2 % test accuracy. It is clear that the blue narrow band is not a perfect generalization of training data visualized by colourful dots. On the other hand, using a simple normalization to centre the data around zero and scale them to unity variance, training quickly converged to great test accuracy of 99.4 % for ”Original heuristic” initialization and 99.3 % for ”Xavier uniform”, as shown in Fig 9. Scaling the data down further by a factor of 100 reduced the accuracy again to only 59.8 % achieved by the Xavier normal initialization, as shown in Fig 10. For an example of completely unsuccessful initialization, one could look at the classification boundary learned by the neural network initialized using the Kaiming normal rule in Fig. 11.

III. Dropout

In the final section of this homework, a neural network for the classification of handwritten digits from the MNIST dataset was implemented. It is fed by flattened 28x28 pixel images (i.e. input size 784) and contains two hidden linear layers of width 800 units each. The output layer has 10 units for the prediction of digits. Parameters are initialized using the normal distribution with zero mean and variance 0.1. A dropout module was inserted in front of each layer of the

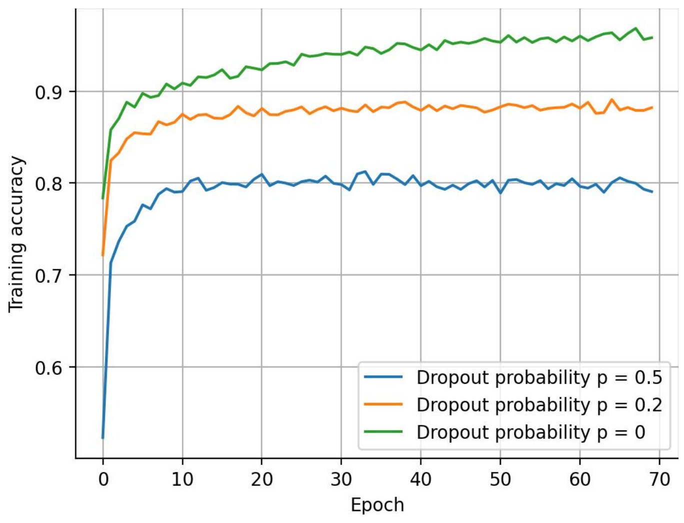

Fig. 12: Evolution of training accuracy for various dropout probabilities.

network to zero out elements of tensors during the forward pass randomly. Three values of the dropout probability p were tested: 0 % (essentially no dropout), 20 % and 50 %.

Using the same training setup as in Section II, i.e. AdamW optimizer and a safe learning rate 0.0003, results shown in Fig. 12 through 14 were obtained. It is possible to conclude that through the introduction of dropout modules into the model structure, training loss and accuracy became worse whereas validation performance indicators (loss and accuracy) improved.

The original Dropout paper[2] provides a very intuitive explanation – randomly zeroing some elements of tensors during the forward pass (equivalent to eliminating some connections, i.e. giving them weight zero) forces each individual unit to become more independent as it can't rely with certainty on the presence of other units. Through this independence, the model generalizes better, improving the validation accuracy by several %.

A. Ensemble of models using Dropout

In the last experiment, the network was used in training mode (Dropout active even during evaluation); every evaluation (forward pass) then experiences some (unique) combination of dropouts, essentially constructing a new, slightly different neural network. Performing many evaluations with the same data and averaging the class scores can lead to improved performance. Table I summarizes results obtained for various ensemble sizes (number of evaluations per each test sample) and dropout probabilities p. Evaluations were done using the most successful model saved during the training process for each combination of hyperparameters. An ensemble size "None" corresponds to the baseline when the dropout is only active during evaluation rather than training and there is only one forward pass per piece of data. There is some improvement (roughly 2.5 %) for $p = 0.5$ but otherwise both approaches to Dropout seem to yield results of similar quality.

[2]Accessible online at https://arxiv.org/pdf/1207.0580.pdf

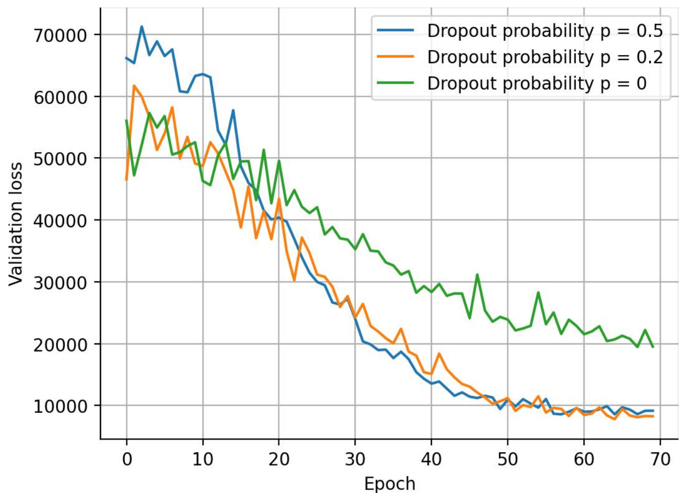

Fig. 13: Evolution of validation loss for various dropout probabilities.

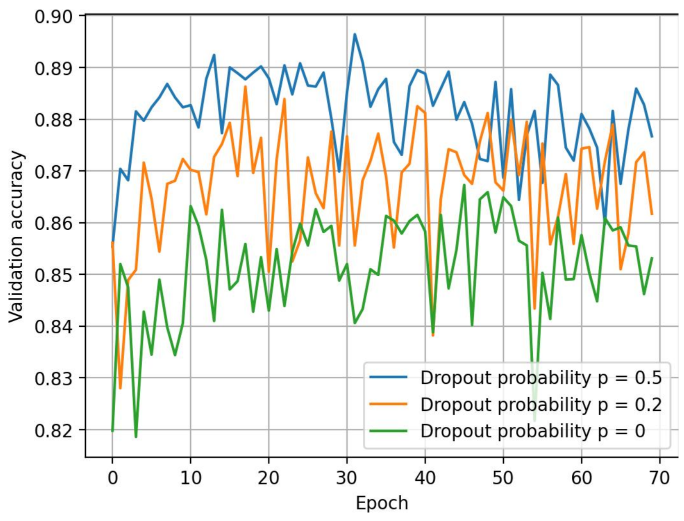

Fig. 14: Evolution of validation accuracy for various dropout probabilities.

Dropout probability p	ensemble size	Test accuracy
0.0	None	86.3 %
0.0	5	87.62 %
0.0	50	87.62 %
0.0	500	87.62 %
0.0	1000	87.62 %
0.2	None	86.75 %
0.2	5	89.33 %
0.2	50	89.31 %
0.2	500	89.4 %
0.2	1000	89.36 %
0.5	None	88.54 %
0.5	5	87.03 %
0.5	50	86.58 %
0.5	500	87.1 %
0.5	1000	86.81 %

TABLE I: Achieved test accuracy for various Dropout hyperparameters.