Task a

We have a dataset with 6238 trainining entries and 1559 test entries. Every entry has 300 features. We want to classify letter „a“-“z“, so we have 26 classes (our output neurons = „one-hat-encoding“).

First we normalize the data, then we split our training set in 3 parts:

- Training: 70% ~ 4366 entries
- Validation: 15% ~ 935 entries
- Early Stopping: 15% ~935 entries

Early stopping just validates the network on separte data to avoid overfitting.

Task b & c

Here we compare different Algorithms on our network with different activation functions in the hidden layer.

Our network:

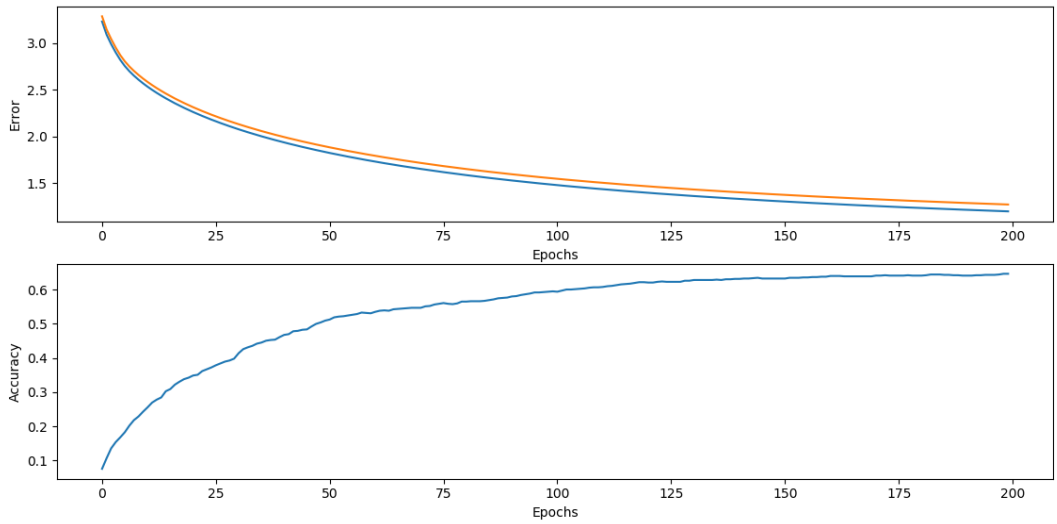
- 300 input neurons
- 20 hidden neurons (different activation functions)
- 26 Output neurons with softmax activation function

We implement early stopping, but for testing we run for total 200 iterations to compare the different algorithms and parameters.
As learning rate we choose: 10e-3
We initialize our weights and bias randomly (normal distributed).
We use mini batches for traing (size = 40)

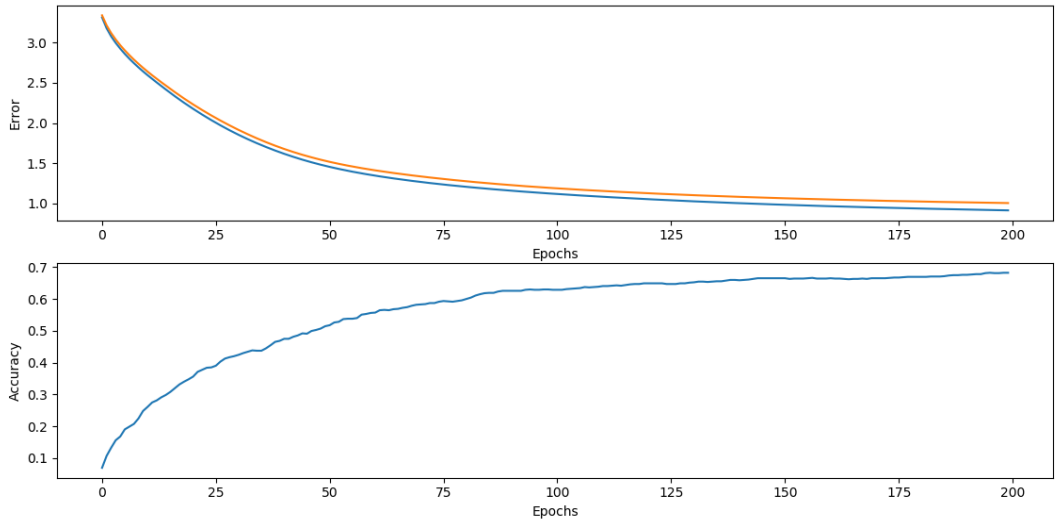
For each combination (algorithm/activation function) we plot the error and accuracy over the iterations.

Algorithm	**Activation function**
Gradient Descent	Logistic sigmoid
RMSprop	Tanh
ADAM	ReLU

Results Gradient Descent



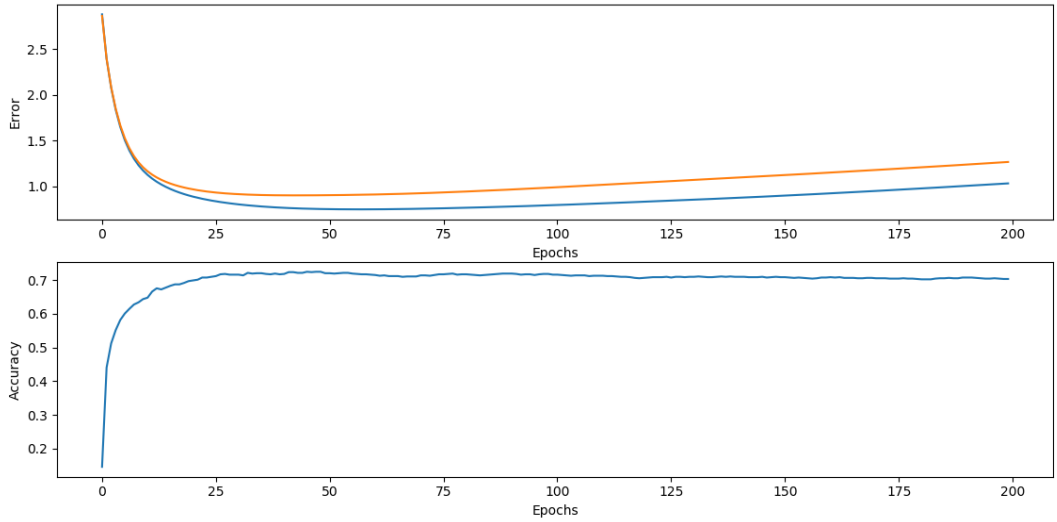
Gradient Descent with Tanh activation function in hidden layer



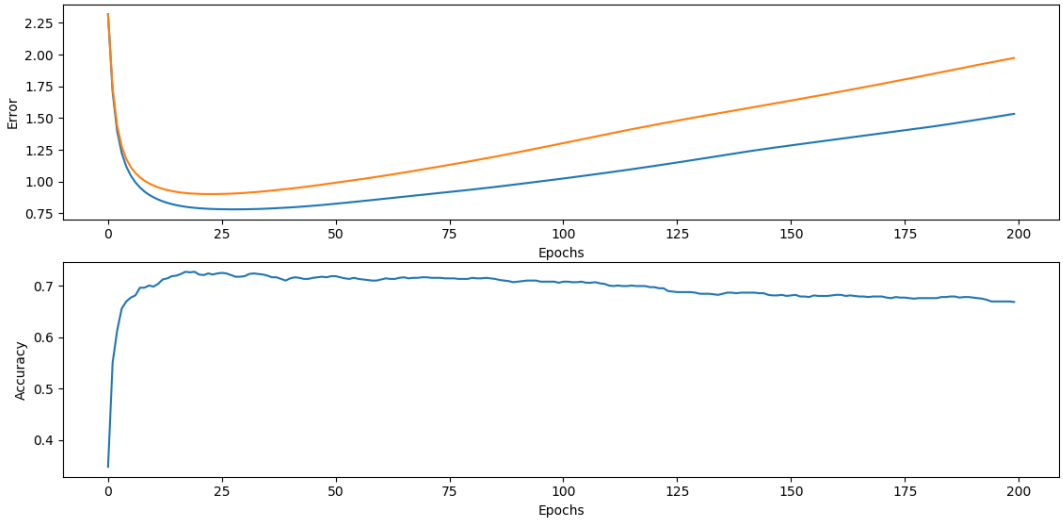
Gradient Descent with ReLU activation function in hidden layer

We can see that for standard gradient descent the ReLU activation functions gets the best result. Still we need a lot of iterations to converge.

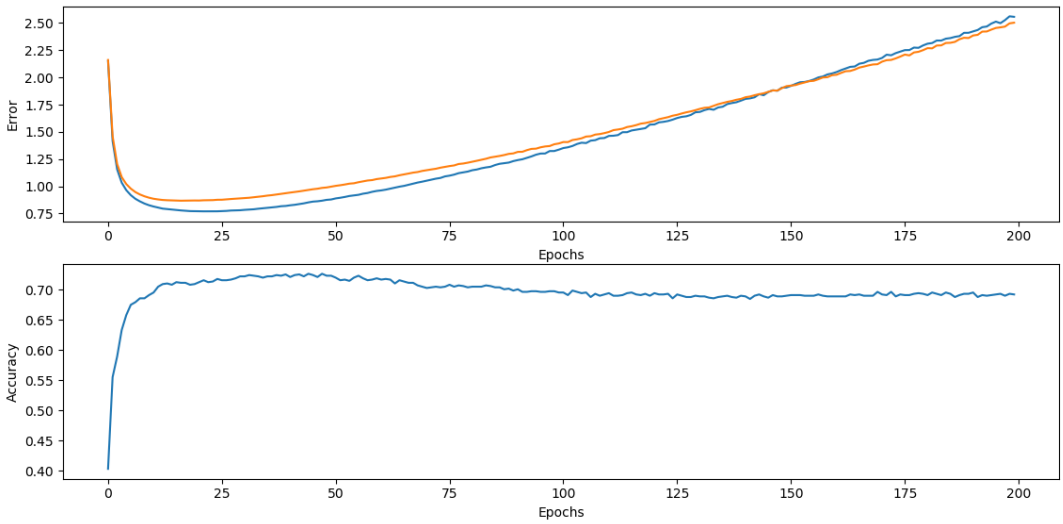
Results RMSprop



RMSprop with Sigmoid activation function in hidden layer



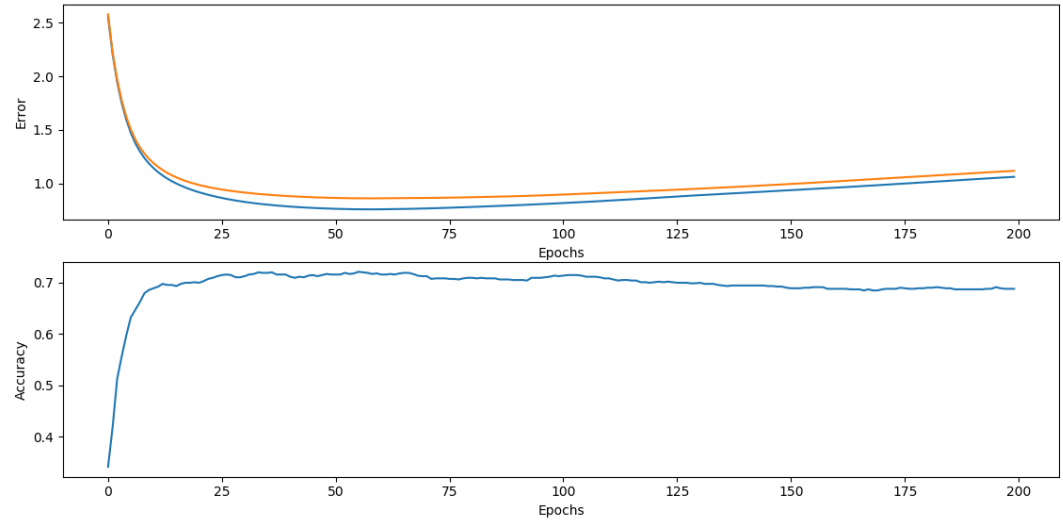
RMSprop with Tanh activation function in hidden layer



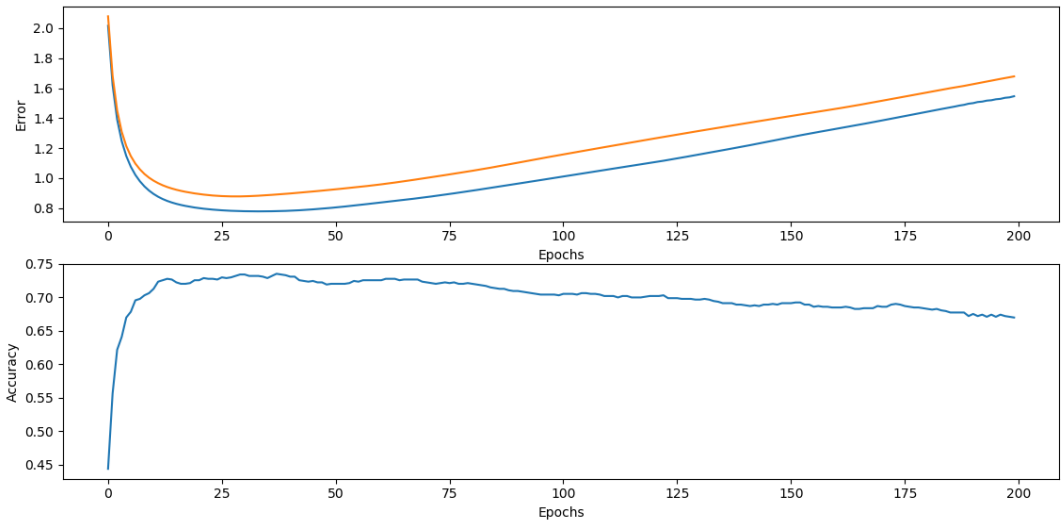
RMSprop with ReLU activation function in hidden layer

This algorithm gets much better results arleady early (15 – 20 iterations). Here we need to use early stopping otherwise we get worse results. I would use the Tanh activation function with early stopping, because it gets good result but is more smooth than the ReLU.

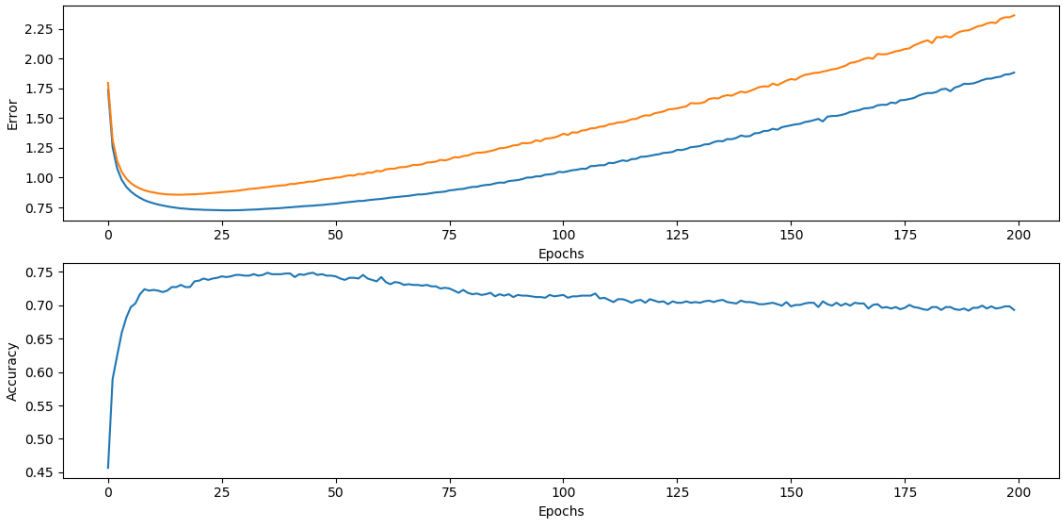
Results ADAM



ADAM with Sigmoid activation function in hidden layer



ADAM with Tanh activation function in hidden layer



ADAM with ReLU activation function in hidden layer

ADAM with ReLU gets the best accuracy very fast, but we need to be aware of overfitting and use early stopping otherwise we again get problems quite fast.

Task d & e

After choosing our algorithm (=ADAM) and our activation function for the hidden layer (= Tanh), we are comparing our result on the real Testset for different number of hidden neurons:

Number of neurons in hidden layer	Accuracy on Test set
20	~ 68%
40	~ 70%
80	~ 71%

Algorithm: ADAM
Activation Function: Tanh
Learning rate: 0.001
Early stopping (usually activated before 20 iterations).

→ As we can see, increasing the number of hidden neurons gives a slightly better result, but still not optimal. We believe better results would be achieved by adding more hidden layers, but we had no time to test this theory.