Tamás Csilling [UP1105444]
Aritifcial Intelligence (CEID1060)
April 28, 2023

Neural Network Assignment

https://github.com/OnionKiller/AI_Homework_HAR-PUC-Rio

A1. Data Preprocessing AND Preparation

Question 1. a) Data prepocessation

To properly asses the data, I used the *ydata-profiling* python library. It showed, that the data mainly consist two separate part: the parameters about the users which is very highly correlated, and the measured values. There was a value error in the Z4 data, which was improperly exported from the csv file. The errorous live was dropped. Also the non US compatible floating point format resulted in custom data exporting functions. Because all of the user data is highly self correlated, the repetative data is not nescesserily related, so the label type user data were all dropped. The neural network would be able to easily transform between these datas, with for example a simple LUT like structure, they are not cruicial for the training. The dropped data columns are: *age*, *user*, *gender*. Otherwise these data is not highliy correlated to any other measured values, so it is not a clear issue to keep them in the trainig dataset.

Because the data will be used in a neural network, to make the trainig easier, the values are transformed to a $[0, 1]$ interval. It could be done with simple min-max scaling, but it is errorprone for outlayers, which is the case for some of the truncated data noise. The data otherwise shows a clear gaussian distribution, with ocassional multi gauss distributions. Only centerig the data is not really nescessery, most of the data is relatively centered. In the end I choose to Standardize the data, because of the general properties of the Standardization, and the general Gaussian structure of the data.

Question 2. b) cross-validation

To make my life easy, I used *Scikit Learns KFold* module, which allows to make the 5 fold CV automaticly. It samples the training set based on random uniform sampling, which I considered balanced in this particular case. It was later verified by the results, that it was indeed balanced, as the variance of the results was relatively low.

All of the users are collected into the training dataset, which is probably not efficient for testing the generalisation capabilities of the dataset. To test the generalisation power of the modell, I also created a dataset, where one of the user was completely removed, and only used in the verification of the modell. This is not easy to compare to the earlier results, so it is only a sidenote at the end of the assignment. (It is not comparable, as with this modell it is not really practical to make 5 fold CV.)

A2. Choice of architecture

Question 3. a) Which one is preferable for training (loss)?

Cross -Entropy measures the difference between the predicted probabilities and the true probabilities, by measuring how well the predicted probability distribution of the model matches the actual probability distribution of the target classes. A lower Cross-Entropy value indicates a better model performance.

Mean Square Error, measures the average squared difference between the predicted and actual values. This shows the distance between the result and the actual values, but for a classification problem it is not really a good measure.

Accuracy shows the percentage of correct predictions from the network. It is a good measurement to show how well a network performs. It is not really a good measure for loss, because the output layer gives a probabailistic dictribution over the possible classes, and accuracy would drop this information.

Question 4. b) How many neurons will you need at the output level?

For the multiclass type of classification because there are 5 classes, it needs 5 neutrons to output. The transformation to this five class output is created with the *to_categorical* from library Keras, which creates binary vector outputs form the categorical int inputs.

Question 5. c) Choose an appropriate activation function for the hidden nodes?

For best adaptation for the hidden layer, and to have a good fit I used sigmoid activation, and fully connected layers.

Question 6. d) Which activation function will you use for the output level?

For classification modells it is very common to use softmax activation. It is pereferable because it normalizes the output of the last layer to obtain a probability distribution over the classes. The output values of the Softmax function represent the probabilities of each class, which always sum up to 1.

Question 7. Experiment with 3 different values for the number of neurons in the hidden layer

Number of neurons in the hidden layer	CE loss	MSE	Acc
THE1 = O	0.079 (+/- 0.001)	0.009 (+/- 0.000)	0.976 (+/- 0.000)
THE1 = (O+I)/2	0.116 (+/- 0.002)	0.014 (+/- 0.000)	0.966 (+/- 0.001)
THE1 = O+I	**0.063** (+/- 0.005)	**0.008** (+/- 0.001)	**0.981** (+/- 0.002)

The problem converges very quickly, and it has semengly no overfitting, so early stopping is altough possible, it is not a nescessity. The average of the validation accuracy is consistently getting higher, and validation loss is monotonic, altough it has a fairly high standard deviation. On Figure 1 it is cleary visible, that there is no overfitting in this configuration. One single fold's result is showing that these validation values are somewhat noisy, but if patience is used at the early stopping, they consistently converge to a lower value. Figure 2 shows this.

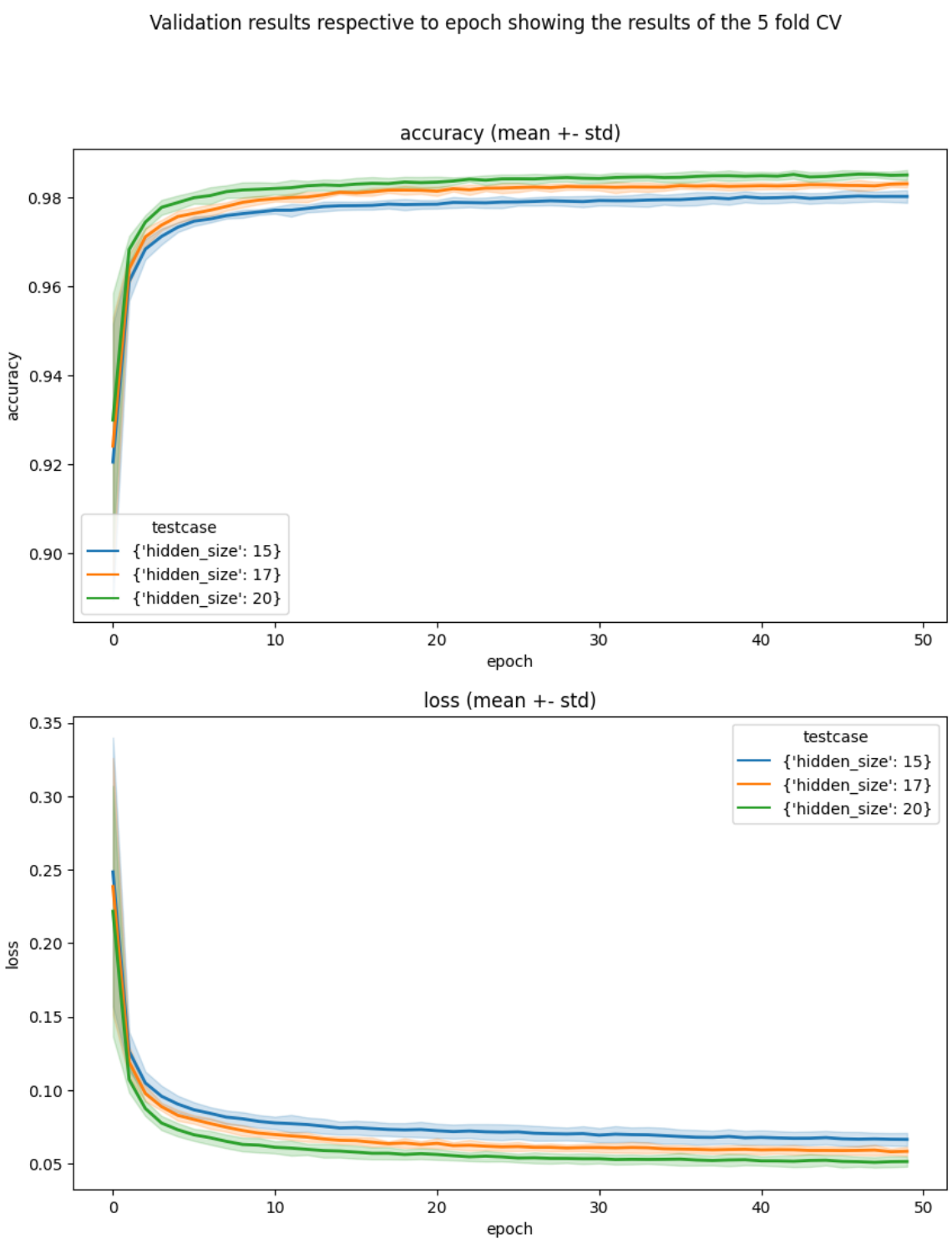

FIGURE 1. Accuracy and loss for the validation dataset for different number of hidden layers

Question 8. State your conclusions about (i) the number of hidden nodes, (ii) the choice of the cost function, and (iii) the convergence speed with respect to the training epochs.

The hidden nodes seems to have an impact on convergence speed, but in general they don't make a huge difference. The loss function is predecided to be CE and it seems to be effective. The best results were from when the number of hidden nodes were the highest, altogh computationally it is the most hardest task. The effects of the different hidden layer node counts can be seen on Figure 1. The convergence speed of these models were relatively the same.

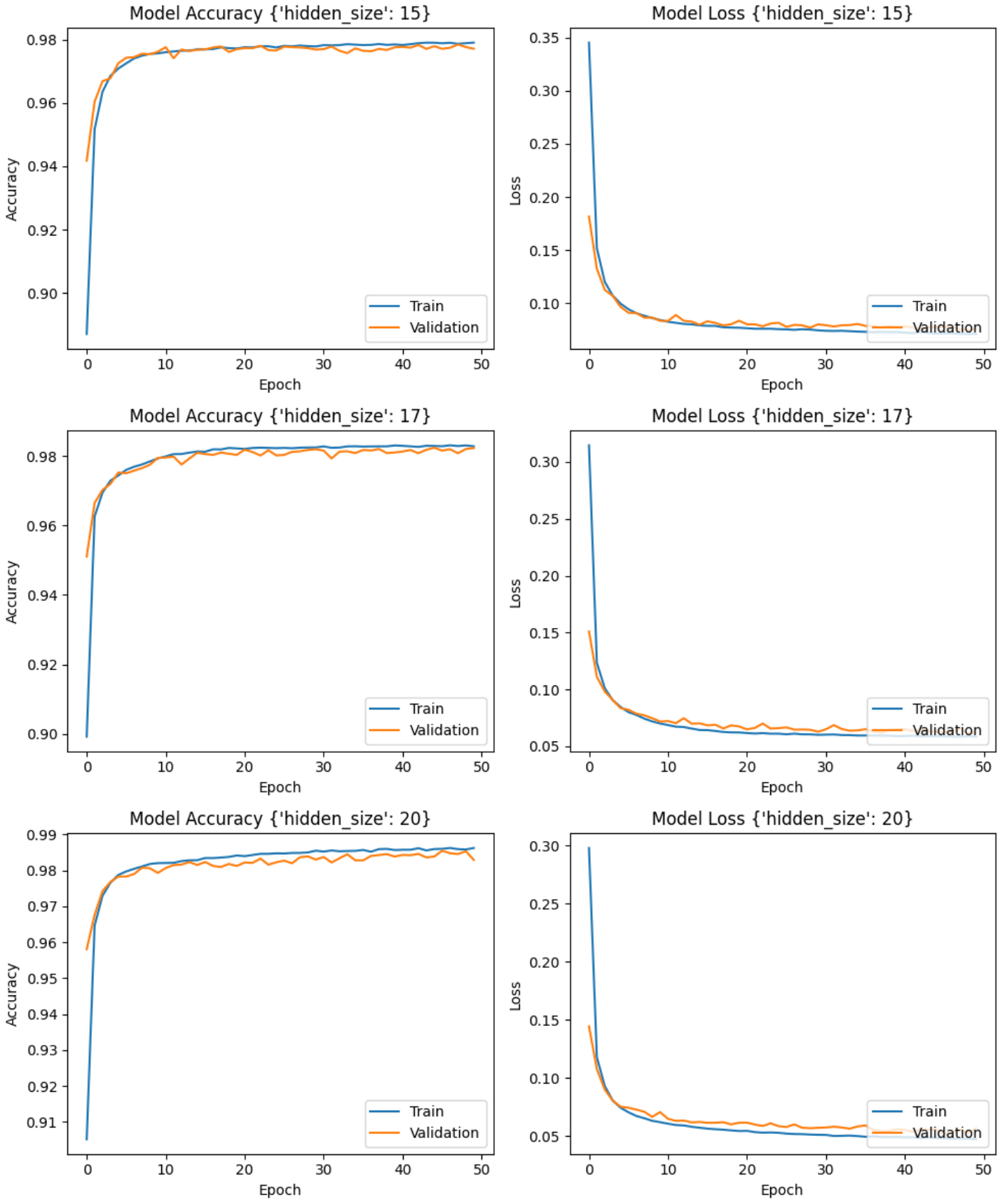

FIGURE 2. Loss and Accuracy for different number of hidden layers

A3. Changes in training rate AND momentum constant

Question 9. Optimize learning rate and momentum.

It is clearly visible on Figure 3 that the lower learning rate results in a very slow learning rate, while the

η	m	CE loss	MSE	Acc
0.001	0.2	0.794 (+/- 0.001)	0.105 (+/- 0.000)	0.737 (+/- 0.001)
0.001	0.6	0.654 (+/- 0.001)	0.088 (+/- 0.001)	0.781 (+/- 0.001)
0.05	0.6	0.118 (+/- 0.002)	0.014 (+/- 0.000)	0.966 (+/- 0.001)
0.1	0.6	**0.095**(+/- 0.002)	**0.011** (+/- 0.000)	**0.972** (+/- 0.000)

higher ones have a way better result. On the second row it is visible, that a high momentum results in a high initial correlation, which later slows down because of the low learning rate. It is also noticeably that the low learning rate reuslts in a very inconsistent initial iterations, altough later all of the data becomes very consistent across folds.

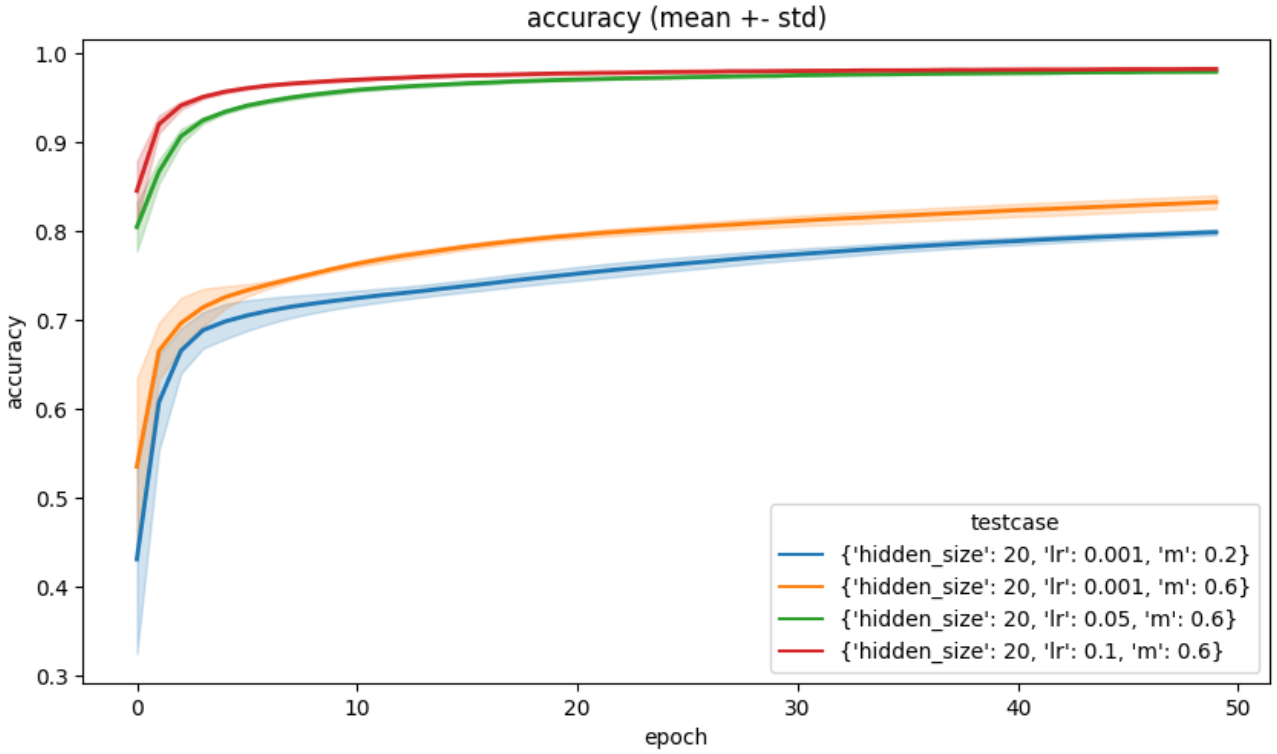

Figure 3. Validation accuracy and loss for different learning rates, and moments

Question 10. Document theoretically why m < 1

Using m > 1 would result in a divergent solution.

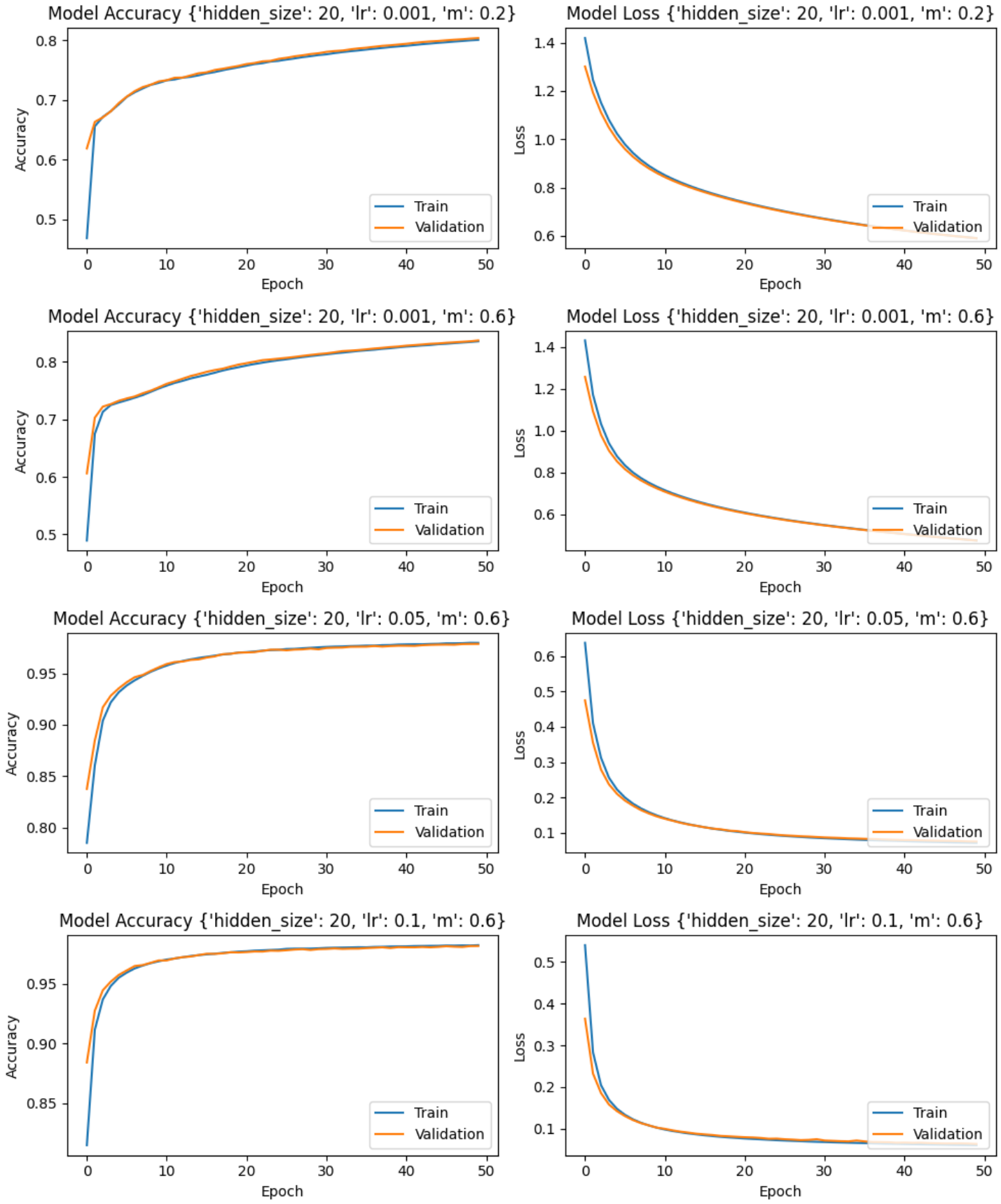

FIGURE 4. Loss and Accuracy for different learning rates, and moments

A4. Normalisation

Question 11. Explain which regularization method (L 1 or L 2) is preferable for this problem.

The preferred method is L2 because it distributes the weight values more evenly across all the neurons, resulting in a smoother decision boundary and better generalization performance. This is easy to implement with *Keras*, but the results were significant enough to see the difference between the configurations., as it is visible on the Accuraccy comparsion on Figure 5

coef	CE loss	MSE	Acc
0.1	0.0615 (+/- 0.0038)	0.0056 (+/- 0.0004)	0.9822 (+/- 0.0014)
0.5	**0.0606** (+/- 0.0020)	**0.0055** (+/- 0.0002)	**0.9827** (+/- 0.0008)
0.9	0.0617 (+/- 0.0003)	0.0057 (+/- 0.0001)	0.9817 (+/- 0.0005)

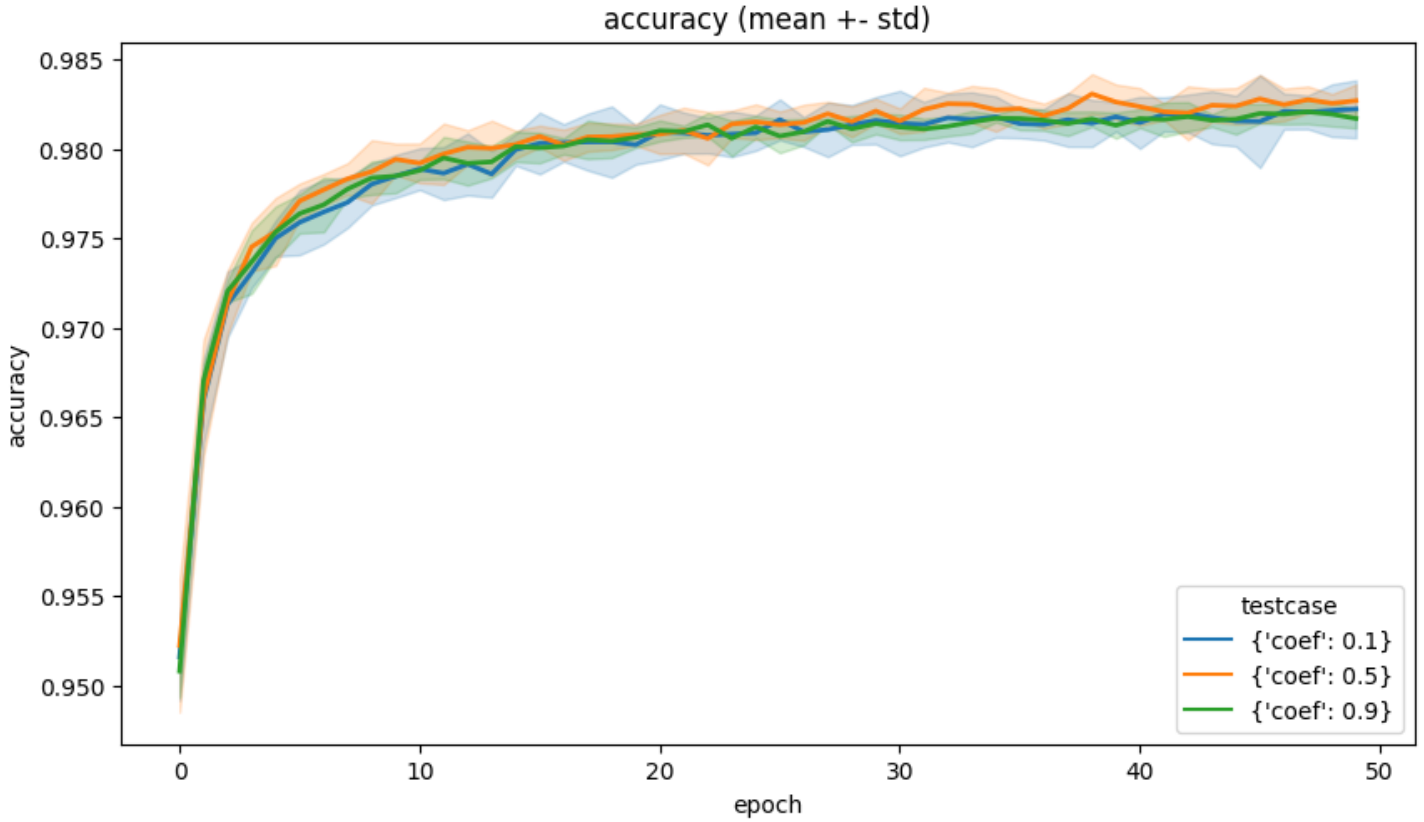

Figure 5. Normalisation parameter results compared, in terms of Accuracy.

Generalisation test

To test the normalisation on a more challenging enviroment, I removed the smalles count user from the dataset, and trained the network on the remaining 3(!) users. This was to test the generalisation power for the paramer range based on phisical parameters like mass, height, aand other factors. The removed dataset was used as the validation set.

To no surpsise, this turned out to be a real tough challenge, where most of the networks overfitted, and had a really poor performance. Compared to the 5 fold cross validation in this case the lower count hidden layers worked better, which also suggests that there may be a trivial connection bertween some of the parameters. Probably a DNN will perform better for this task, but it would require additional testing.

The results showed, that most of the settings overfitted really badly, but some of them performed surprisingly well. This seems to be totally inconsistent, altough in general it seems likely that the L2 normalisation has a better performance in the long run. Figure 7 ans Figure 8 shows different configurations and their reuslt in therm of epochs and accuracy and loss. For the coefficient it turned sout, that the 0.9 coefficient is way to high ,and the modells performed best with the 0.5 coefficient. Figure 6 shows how the different coefficients affected an L2 normed training, with low hidden layer size.

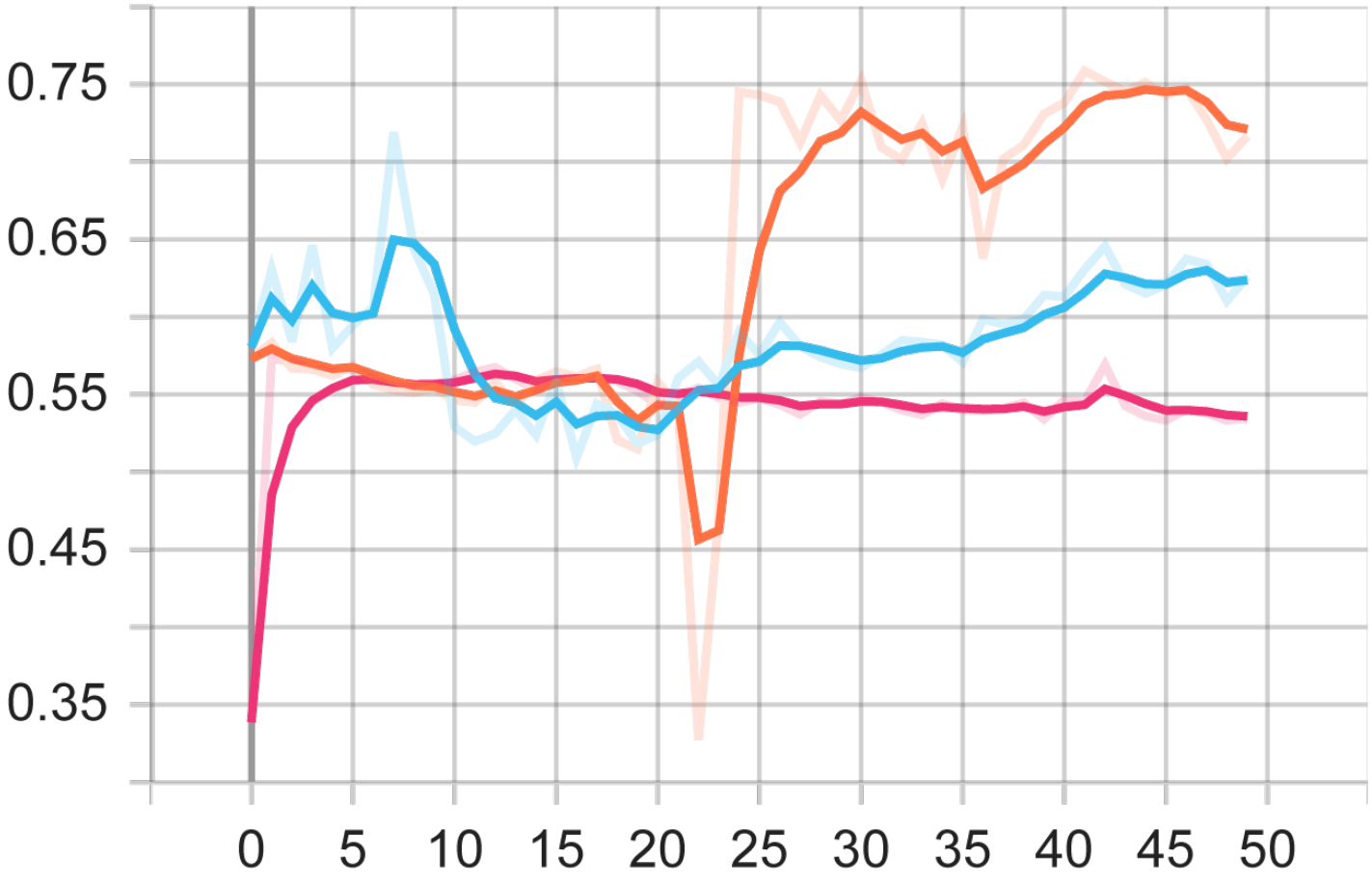

Figure 6. Accuracy of different coefficients. All of the trainig were done with 5 hiddel layers, and Adams optimiser with $\eta = 0.001$ and used L2 normalisation.
The coefficients: *blue* $= 0.1$ *orange* $= 0.5$ *red* $= 0.9$

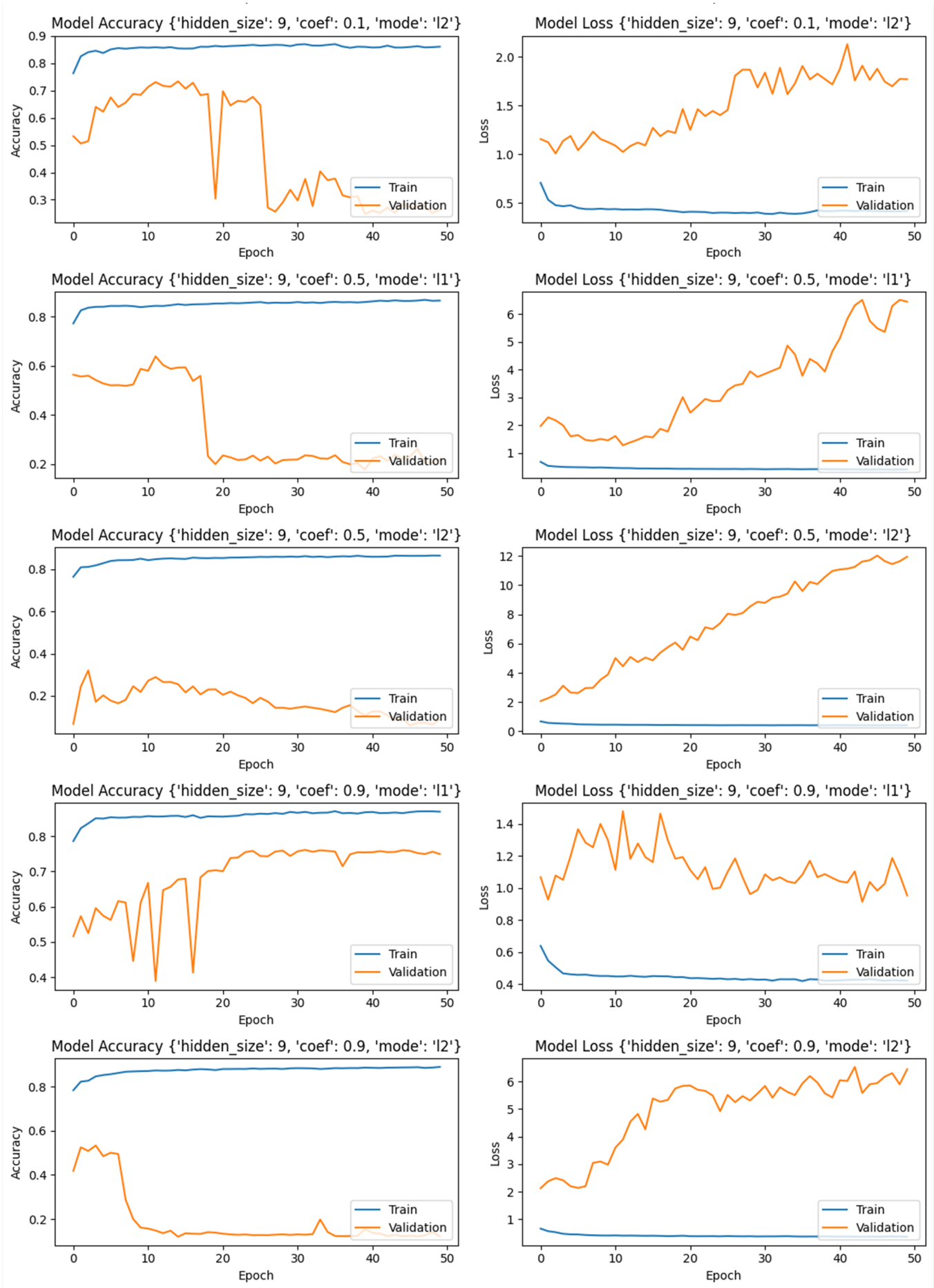

FIGURE 7. Loss and Accuracy for different configurations using completely independent user data

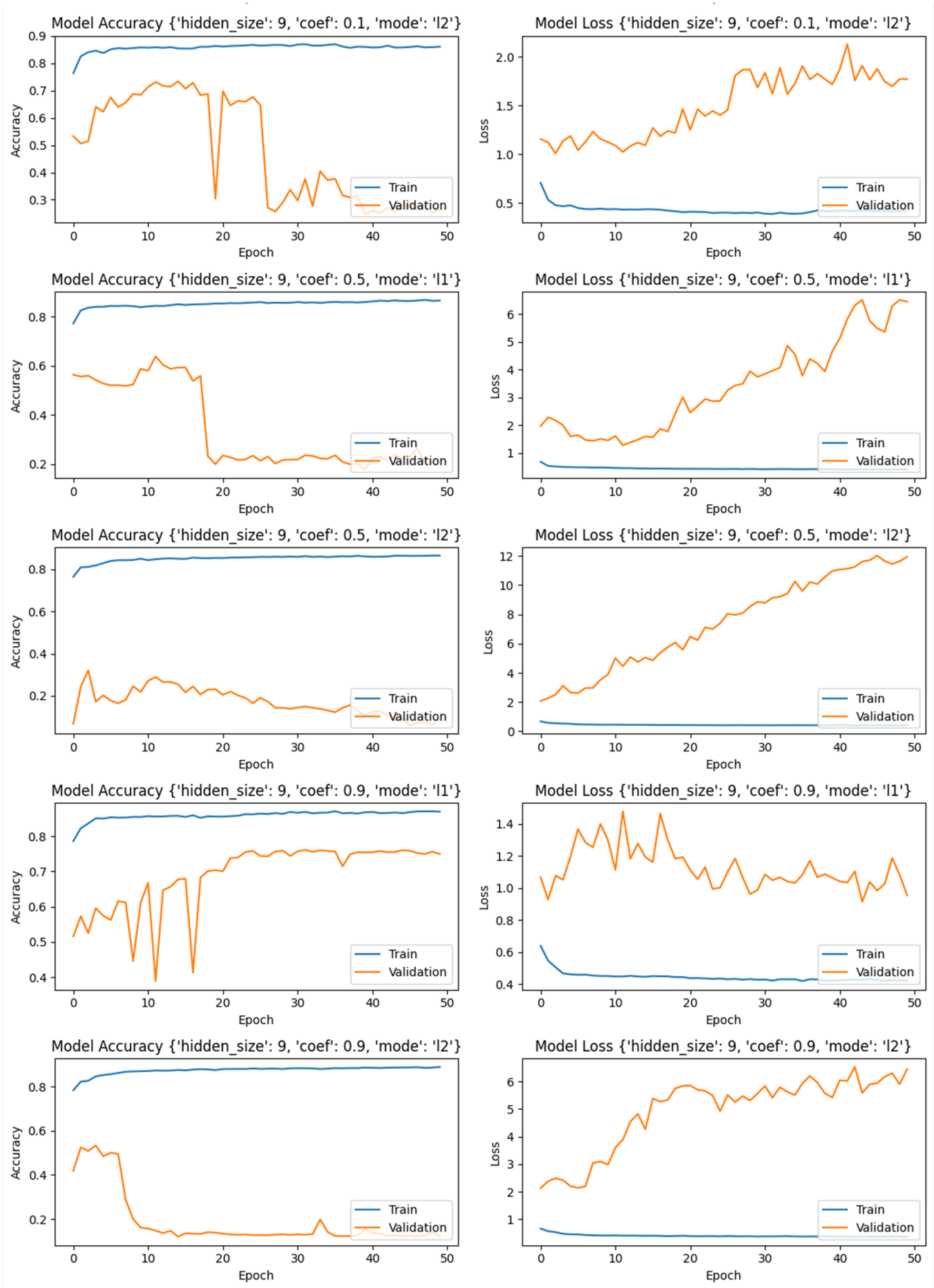

FIGURE 8. Loss and Accuracy for different configurations using completely independent user data