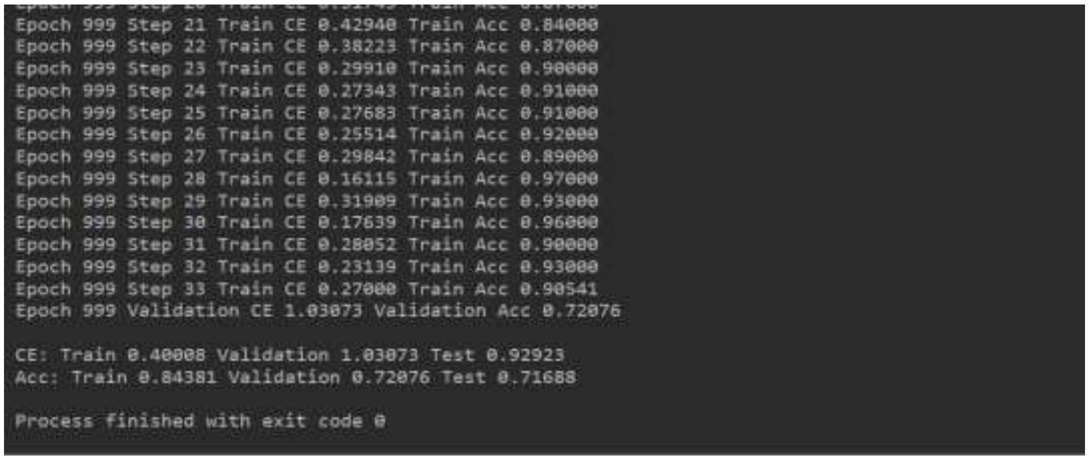
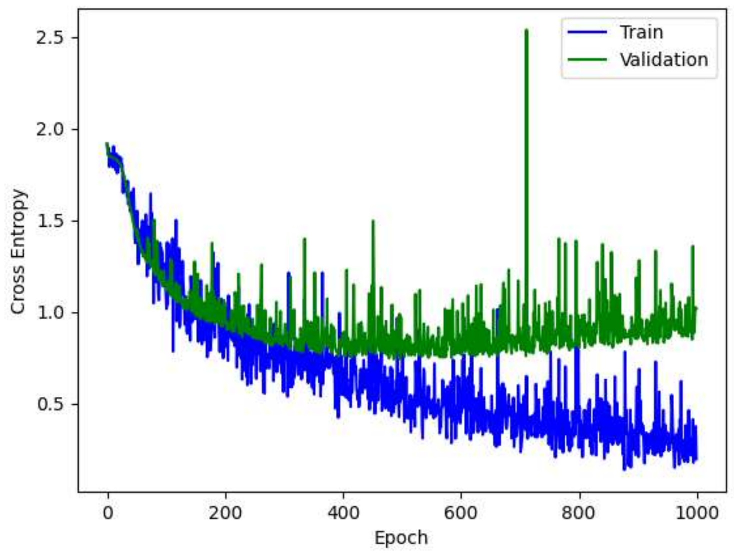
2.2 Generalization [10 points]. Train the neural network with the default set of hyperparameters. Report training, and validation errors and a plot of error curves (training and validation). Examine the statistics and plots of training error and validation error (generalization). How does the network's performance dffer on the training set vs. the validation set during learning?

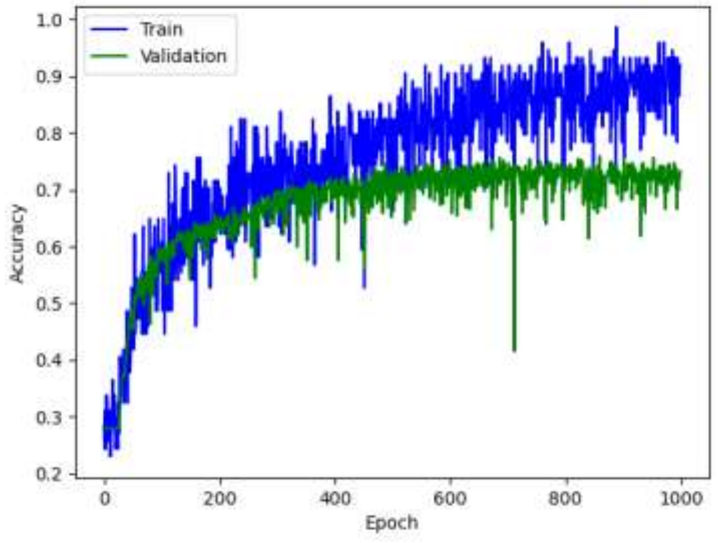


```
Epoch 999 Step 21 Train CE 0.42940 Train Acc 0.84000
Epoch 999 Step 22 Train CE 0.38223 Train Acc 0.87000
Epoch 999 Step 23 Train CE 0.29910 Train Acc 0.90000
Epoch 999 Step 24 Train CE 0.27343 Train Acc 0.91000
Epoch 999 Step 25 Train CE 0.27683 Train Acc 0.91000
Epoch 999 Step 26 Train CE 0.25514 Train Acc 0.92000
Epoch 999 Step 27 Train CE 0.29842 Train Acc 0.89000
Epoch 999 Step 28 Train CE 0.16115 Train Acc 0.97000
Epoch 999 Step 29 Train CE 0.31909 Train Acc 0.93000
Epoch 999 Step 30 Train CE 0.17639 Train Acc 0.96000
Epoch 999 Step 31 Train CE 0.28052 Train Acc 0.90000
Epoch 999 Step 32 Train CE 0.23139 Train Acc 0.93000
Epoch 999 Step 33 Train CE 0.27000 Train Acc 0.90541
Epoch 999 Validation CE 1.03073 Validation Acc 0.72076

CE: Train 0.40008 Validation 1.03073 Test 0.92923
Acc: Train 0.84381 Validation 0.72076 Test 0.71688

Process finished with exit code 0
```



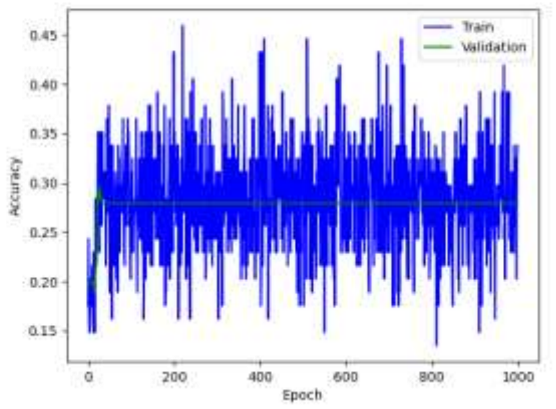


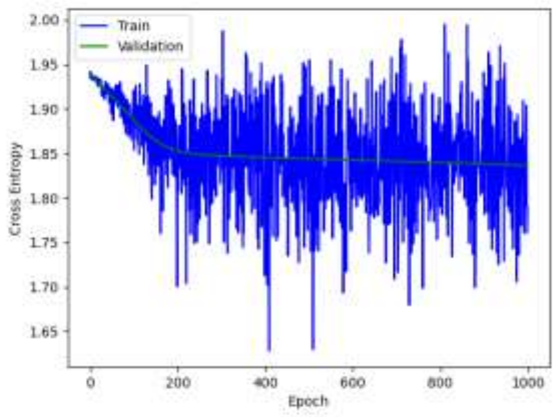
As expected, the training accuracy was higher than the validation set accuracy and the training cross entropy lower than the validation set cross entropy. The validation set exists to estimate the approximate test error for an unseen dataset, trained on data that is similar, (training data) but not quite the same. A validation error that is substantially higher than the training error would imply that our model is overfitting, and thus not a model suitable for good generalization. In this case, the validation error is only slightly higher than our training model, so we can assume that there is generalization.

2.3. Optimization [10 points]. Try dfferent values of the learning rate (step size) _ (\eta")
ranging from _ 2 f0:001; 0:01; 0:5g. What happens to the convergence properties of the algorithm (looking at both cross-entropy and percent-correct)? Try 3 dfferent mini-batch sizes ranging from

f10; 100; 1000g. How does mini-batch size affect convergence? How would you choose the best value of these parameters? In each of these hold the other parameters constant while you vary the one you are studying.

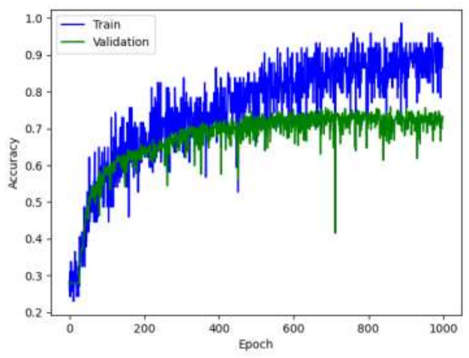
Eta = 0.0001

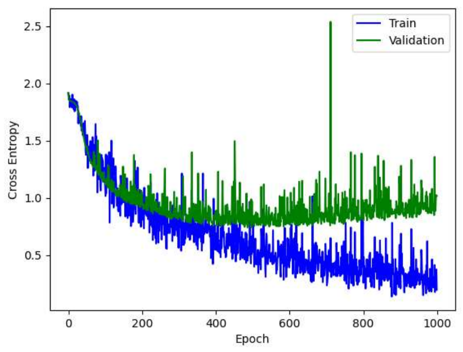




Model does not converge,

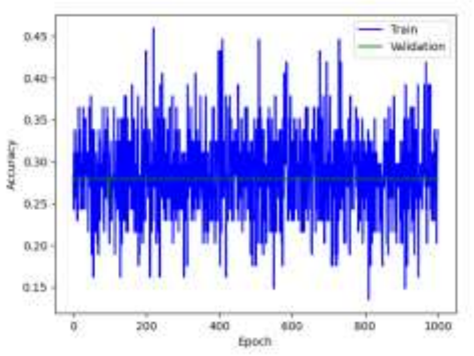
Eta = 0.01

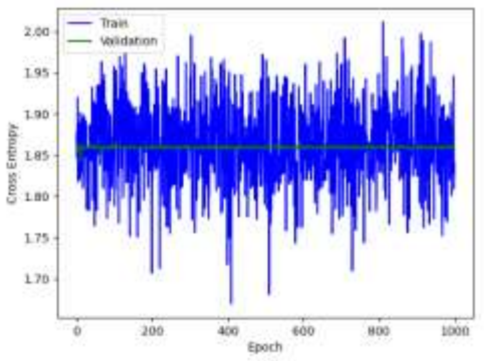




Model converges

Eta = 0.5

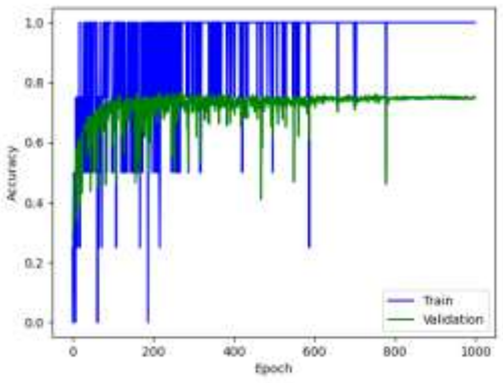


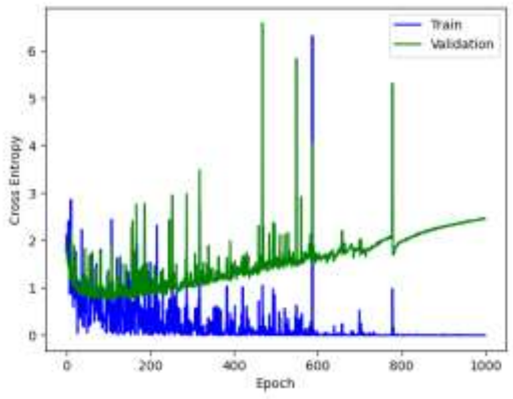


Model does not converge

If eta is made to large, then the model will diverge, if it is made to small, than the model will also diverge, in this case, it is necessary to strike a balance and choose an eta that is still small, but not too large, otherwise the model will diverge

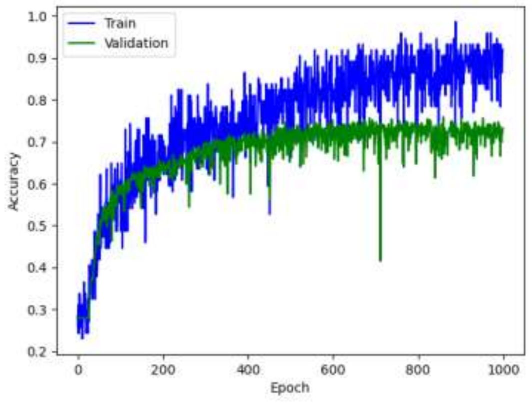
Mini-batch = 10

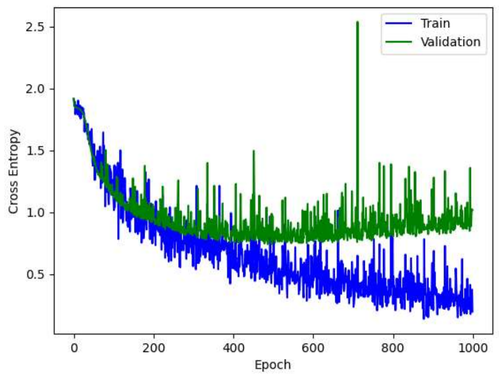




Model converges in a step function like fashion for accuracy, has the potential vary wildly in some epochs however.

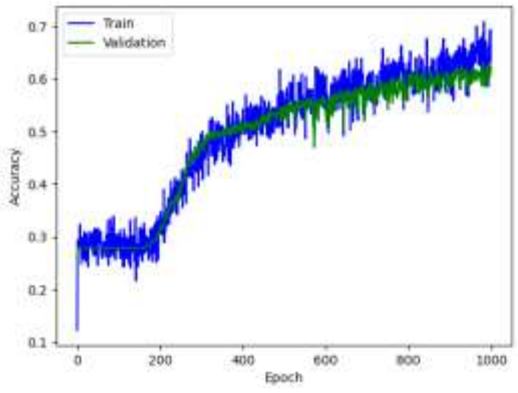
Mini-batch = 100

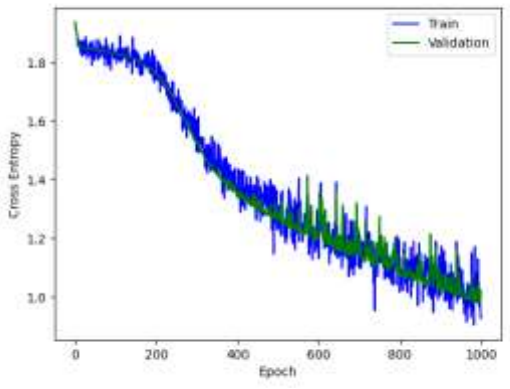




Model converges smoothly and without many “large deviations” as seen in the smaller mini-batch

Mini-batch = 1000

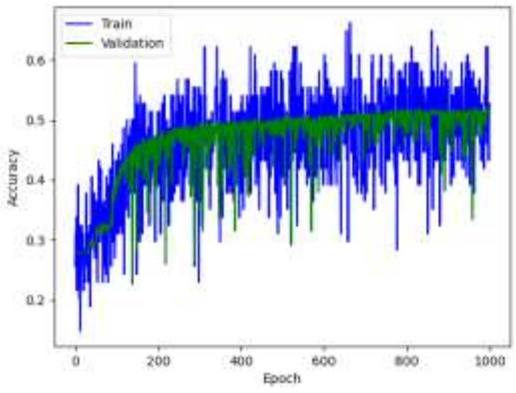


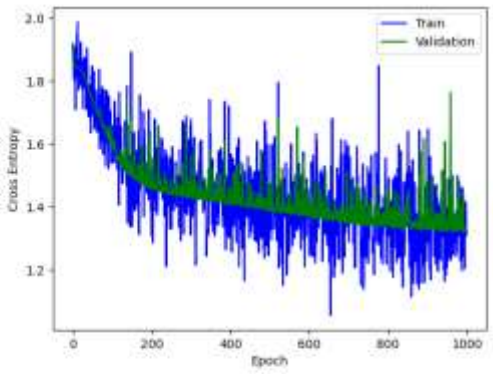


Model converges very smoothly, hardly any deviations from the curve.

We notice then that as the batch size increases, there is more stability in our models, however, this seems to increase computation time.

2.4 Model architecture [10 points]. Try 3 dfferent values of the number of hidden units for each layer of the fully connected network (range from f2; 20; 80g). You might need to adjust the learning rate and the number of epochs (iterations). Comment on the effect of this modification on the convergence properties, and the generalization of the network.



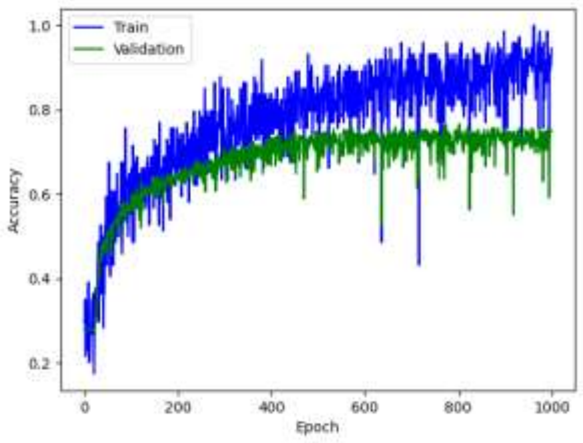


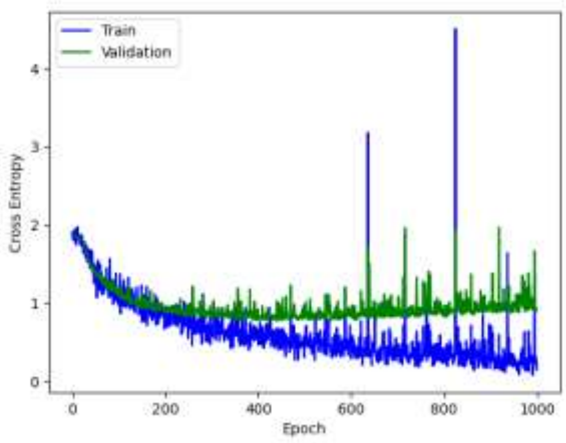
[2,32]

CE: Train 1.29294 Validation 1.32240 Test 1.35710

Acc: Train 0.52282 Validation 0.51313 Test 0.52208

Model converges nicely to 50% accuracy for all data sets. While the model generalizes well as the accuracies over all data sets are very similar, it is misclassifying data about half the time.





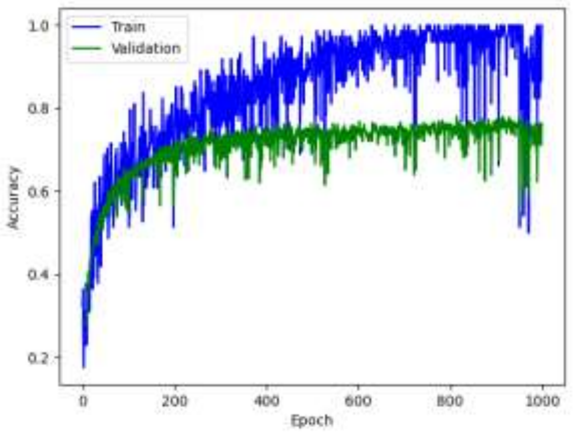
[20,32]

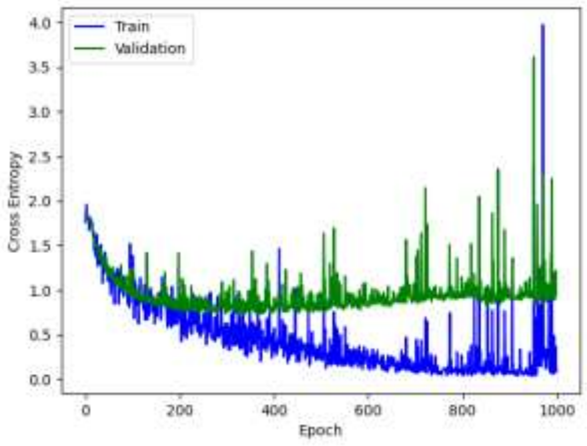
CE: Train 0.20991 Validation 0.92742 Test 0.79656

Acc: Train 0.93065 Validation 0.74940 Test 0.75584

Model has higher training accuracy, slightly higher validation accuracy, and slightly higher test accuracy as opposed to 16,32. All accuracies have gone up as we have increased the number of units in the model, the model should still generalize, as the test accuracy does not deviate significantly from the validation set. Model still converges, but varies slightly more

[80,32]



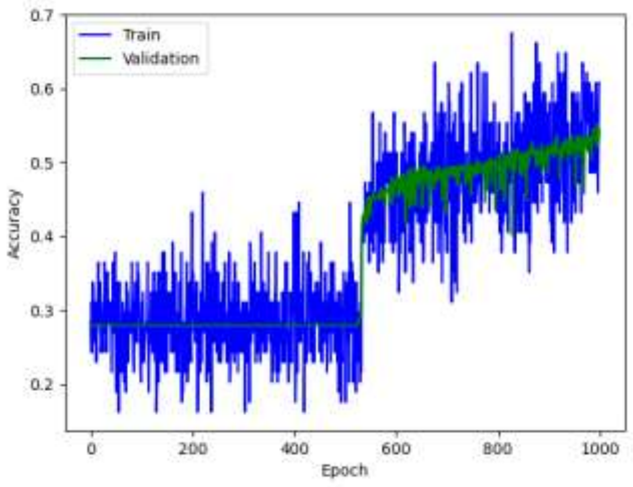


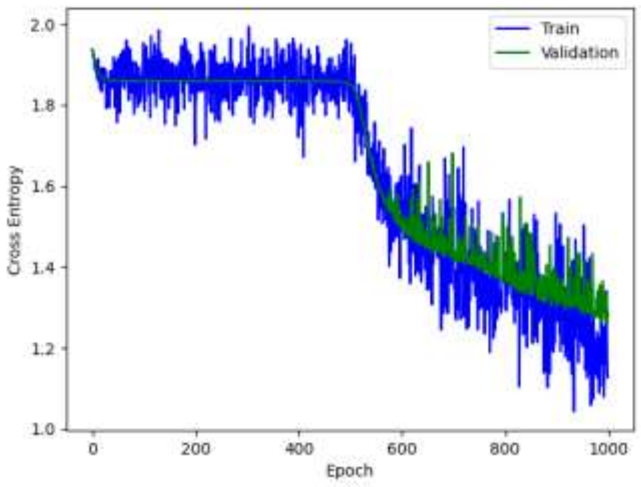
CE: Train 0.09762 Validation 0.94889 Test 0.80966

Acc: Train 0.97807 Validation 0.76372 Test 0.74286

Training accuracy almost perfect, validation accuracy slightly higher, and slightly higher test accuracy, model hardly converges, and has numerous points where the accuracy and cross entropy spike and deviate significantly from the general trend in the error curve.

[16,2]



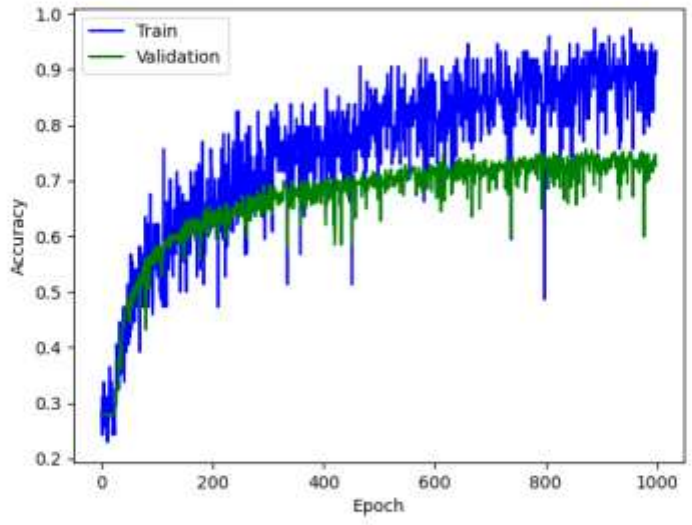


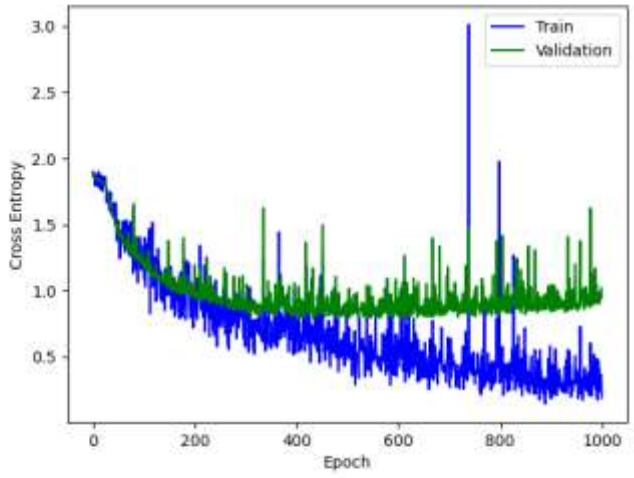
CE: Train 1.19892 Validation 1.28107 Test 1.36060

Acc: Train 0.55424 Validation 0.54415 Test 0.52208

Like [2,32], all the accuracies for the training sets are very similar, ranging from 55% to 52%. Cross entropy is also very similar across the training, validation, and test set. It is clear then that using a sparse number of hidden units will limit the model. Model converges strongly

[16, 20]



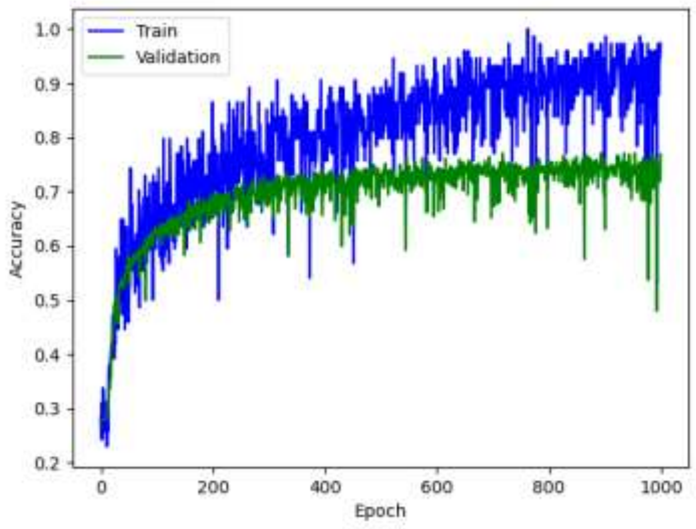


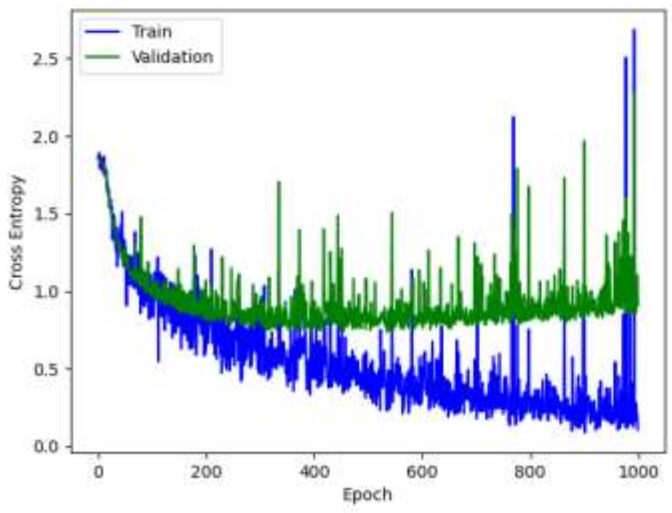
CE: Train 0.26777 Validation 0.95433 Test 0.76329

Acc: Train 0.90901 Validation 0.72792 Test 0.74805

Model still converges, and has slightly better accuracies across all data sets, (except for validation which remains the same) than the original parameters.

[16,80]



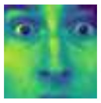


CE: Train 0.18324 Validation 0.90720 Test 0.82007

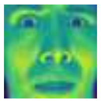
Acc: Train 0.94072 Validation 0.76850 Test 0.74805

Model converges still but has numerous points where the accuracy and cross validation suddenly spike down or upwards, in comparison to (32,80), these spikes are slightly smaller, but significant, nonetheless. Very high training accuracy, higher validation accuracy, higher test accuracy compared to the original parameters.

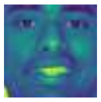
2.5. Network Uncertainty [10 points]. Plot _ve examples where the neural network is not confident of the classification output (the top score is below some threshold), and comment on them. Will the classfier be correct if it outputs the top scoring class anyways? see attached image for list of “uncertain” choices, (predictions where the max probability of a particular class was not significantly different than the probability of another class)



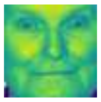
Face 1: Model chose neutral with probability 0.2112, the face appears to be surprised, but the mouth is cut off in the image, likely leading to a faulty image



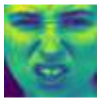
Face 2: The model chose neutral with probability 0.238, this face looks like it is conveying disgust or fear. In this case the model seems to have chosen incorrectly.



Face 3: Model chose neutral with probability 0.2113, this face does not appear to be any particular mood, so it seems to have correctly classified neutral, although it is ambiguous



Face 4: Model chose neutral with probability 0.2265,0020 this is clearly the face of an elderly person, while they may be slightly smiling, it is possible for this face to represent a neutral mood.



Face 5: Model chose neutral with probability 0.177, this face is clearly portraying disgust. The model classified incorrectly.

In all instances where the probability was below .30 for the highest class, the classifier seems to have chosen neutral. Perhaps this is an error in how the values in prediction were displayed, or a mixture of features that would indicate certain classes as opposed to others, when combined can lead the model to choose neutral due to ambiguity. Regardless, it has not done well with these uncertain picks, and clearly made mistakes with face 2 and 5.