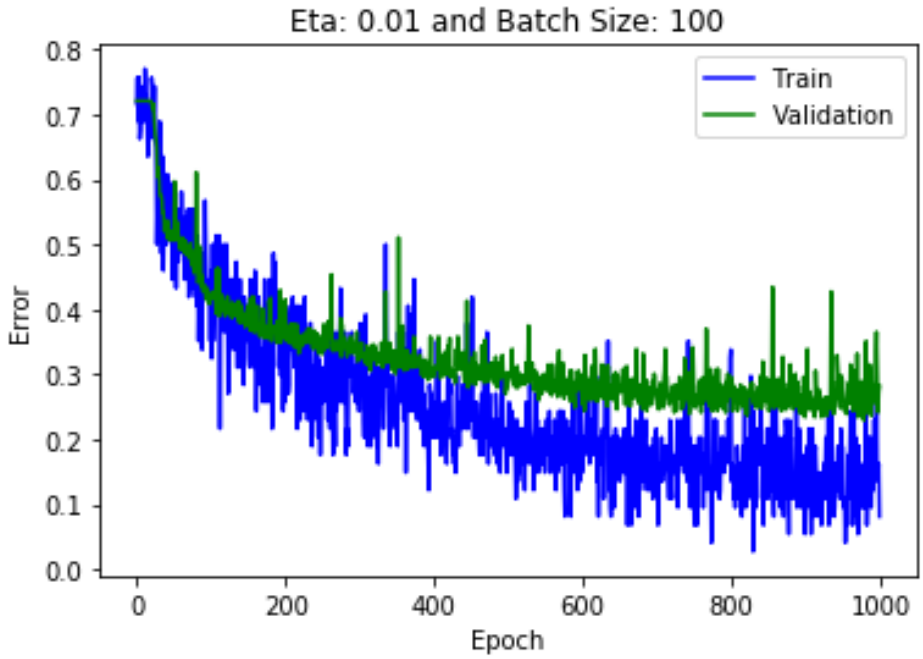
### 2.2) Generalization

Training error: 13.60%

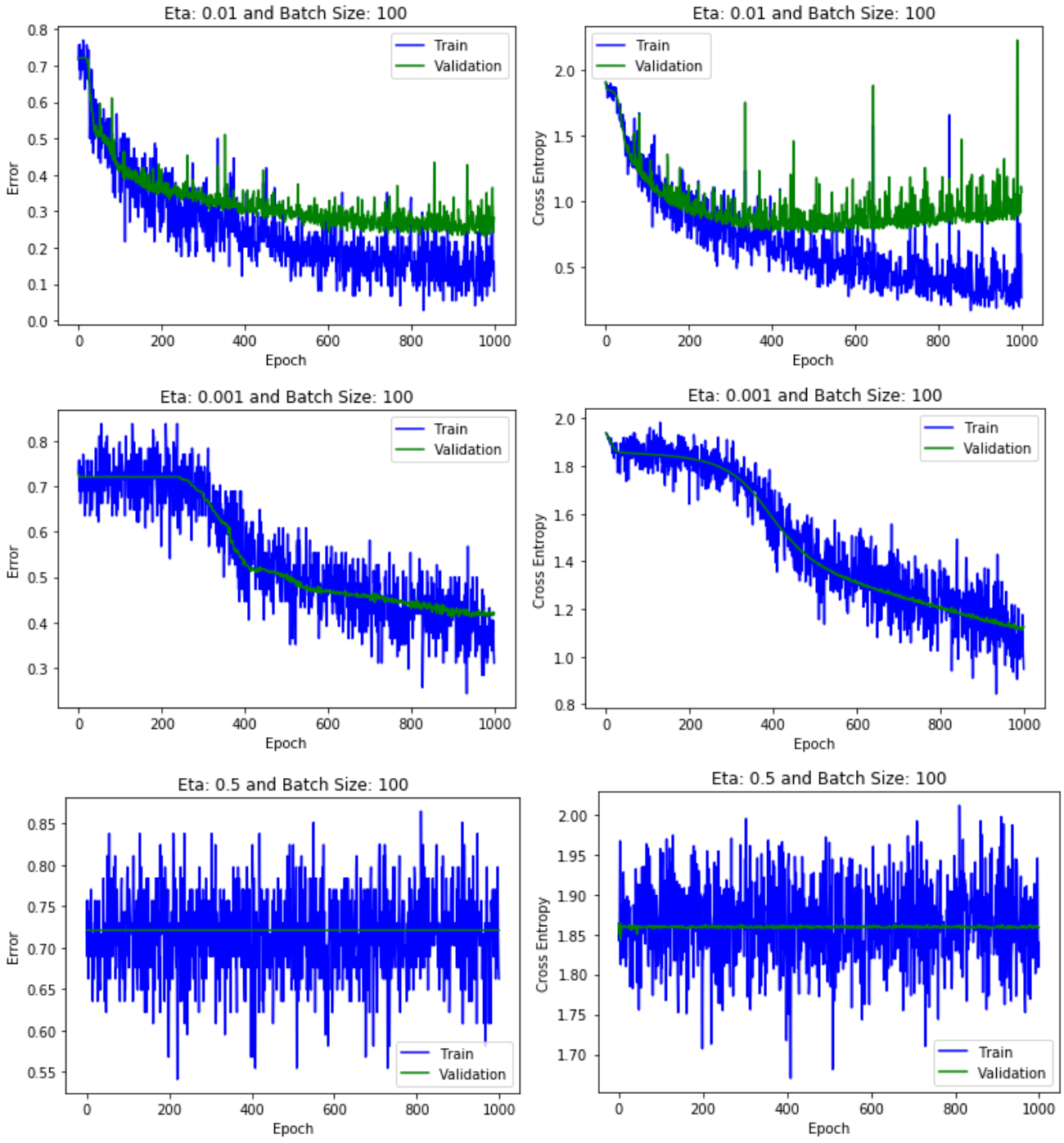
Validation Error: 28.40%

```
CE: Train 0.38604 Validation 1.07142 Test 0.91233
Error: Train 0.13604 Validation 0.28401 Test 0.28312
```



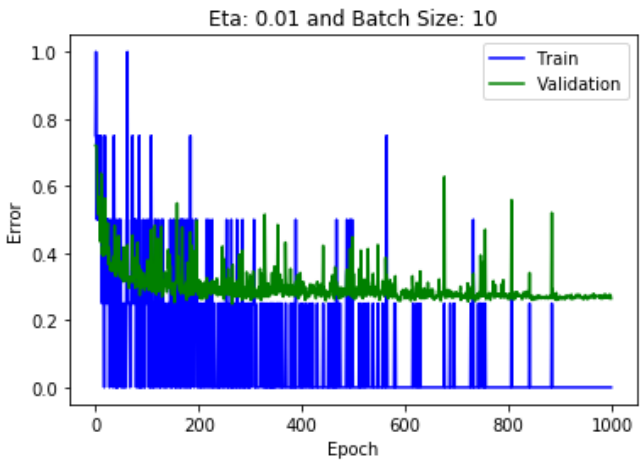

Above is the graph of error (training and validation) across all iterations. You can notice quite the difference between training and validation set. For example, Training set has lower error rate than the validation set for the most part. This tells us that the network performs much better in the training data. This makes sense given that we are fitting our model using training data. Validation is used to check how model performs to unseen data.

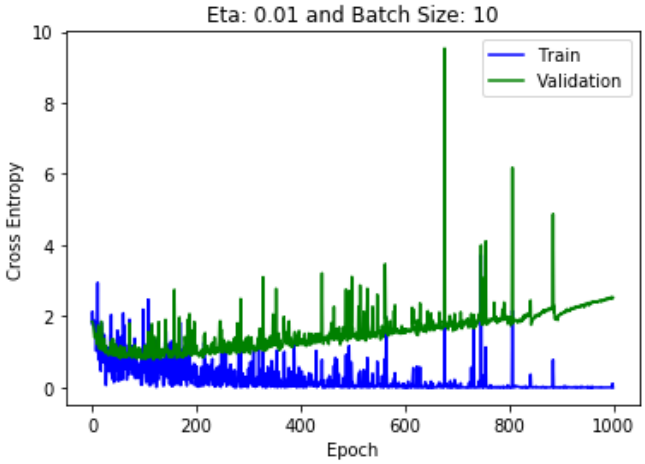
### 2.3) Optimization - Learning Rate

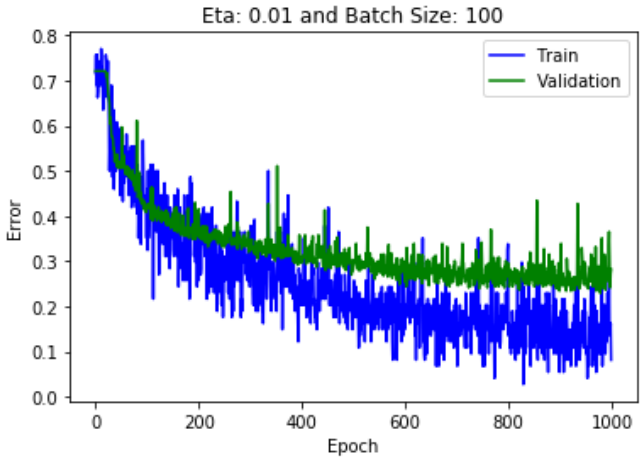


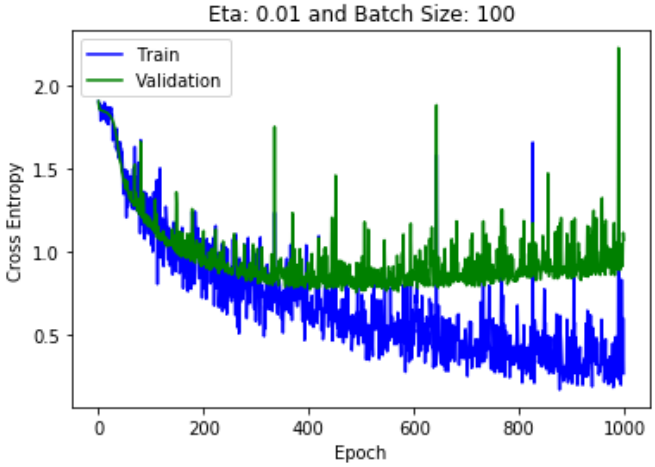

The above models have batch size = 100 kept as constant. The learning rate (Eta) is seen ranging from {0.001, 0.01, 0.5}. Looking at both cross entropy and error, we can see that Eta = 0.01 actually converges (to 0.2 for Error) after around 650 epochs iterations. However, for Eta = 0.001, we can see somewhat of an asymptotic behavior. It steeps down at ~300th iteration and seems to be converging but not so much. For Eta = 0.5, both error and cross entropy sits at 0.72 and 1.86 respectively. It does not converge but stays indifferent across iterations with fluctuations. I would choose Eta = 0.001 to be the best parameter as it converges to a pretty low number and we see the gradual decrease in error rate of the network.

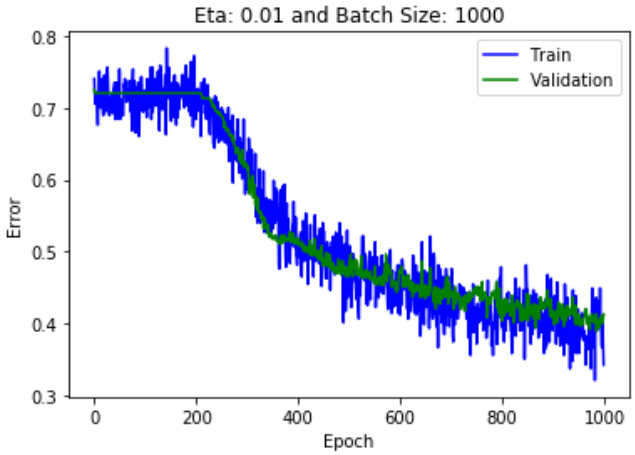
### 2.3) Optimization - Batch Size

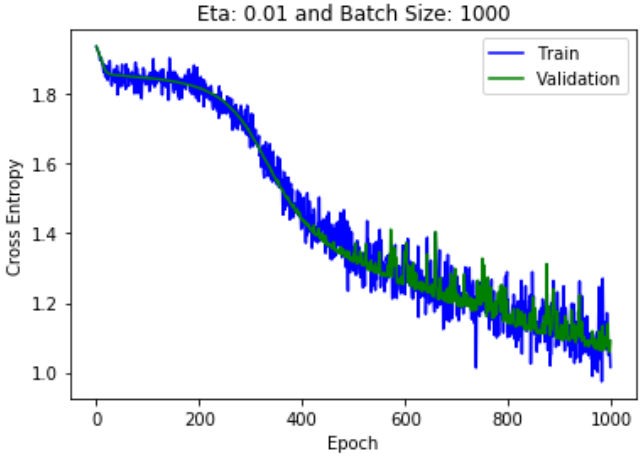


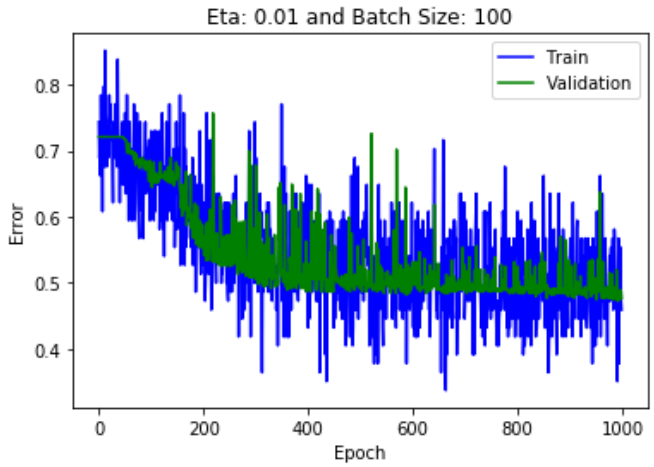



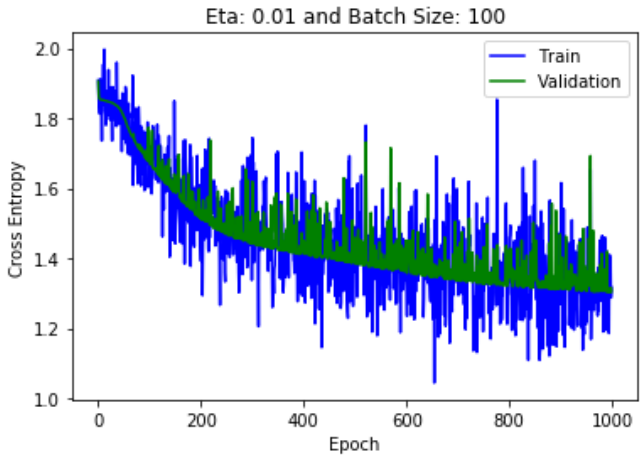

The above models have Eta = 0.01 kept as constant. The batch size is seen ranging from {10, 100, 1000}. If you notice batch size = 10, we actually have 0 error rate in the training set. This means we overfit the training data too much that in validation set we have 0.3 error rate. Even in cross entropy, as number of iterations go up, our loss goes up. This model cannot explain unseen data well. Batch size 100 actually explains the training data and validation data thoroughly. Both sets have converging error rate and cross entropy and generalized the features pretty well. However, for batch size of 1000, we actually generalized features of faces too much. Although it looks like we have converging error rate, this model does not explain that much and cannot classify with good certainty.

### 2.4) Architecture

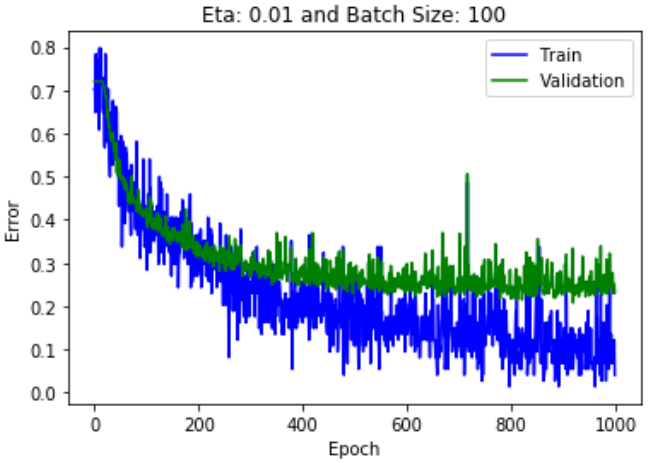
**Hidden Layer**

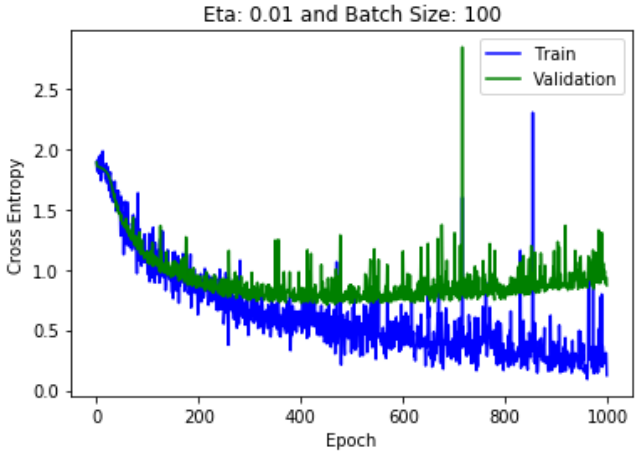
**[2, 32]**



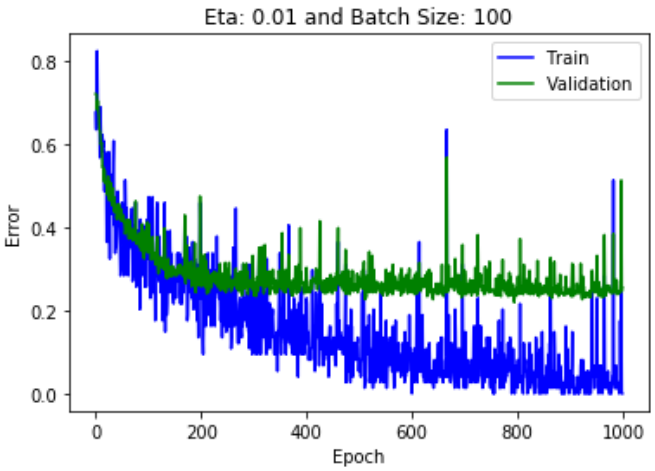



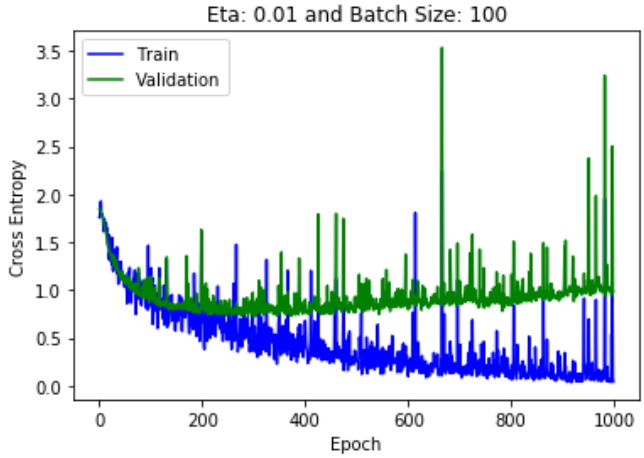

**[20, 32]**




**[80, 32]**



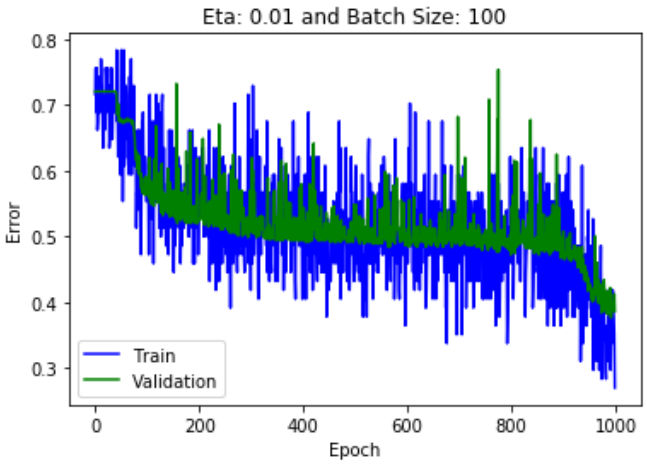



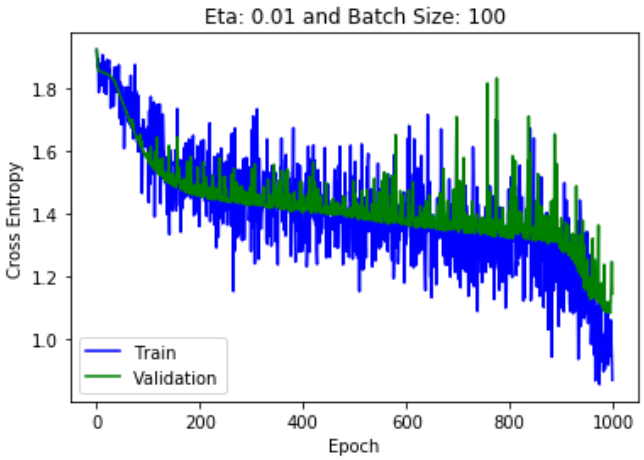

Notice the above graphs of 3 different input layers (details shown on the left). 2 hidden unit does not explain the input data very well. It's essentially extracting 2 features, so the error rate is quite high. Then if you look at 80 hidden units, they are actually looking for way too many features in the image. This usually ends up overfitting and doesn't explain the test data very well. We have to use something in between, say 20 hidden units. This actually explains the data pretty well without overfitting training data hence we achieve generality with [20, 32].

## 2.4) Architecture Continued

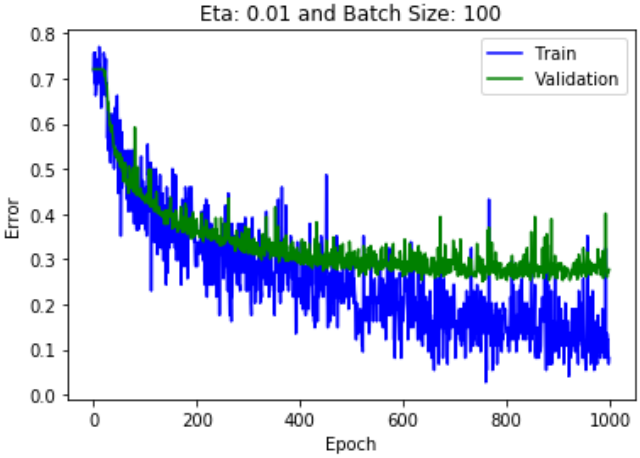
**Hidden Layer**

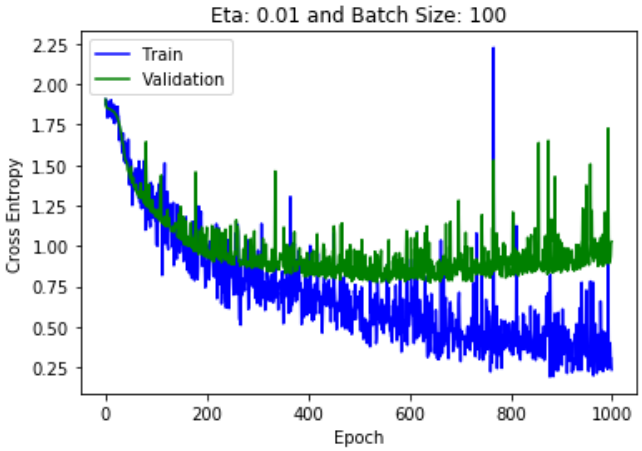
**[16, 2]**



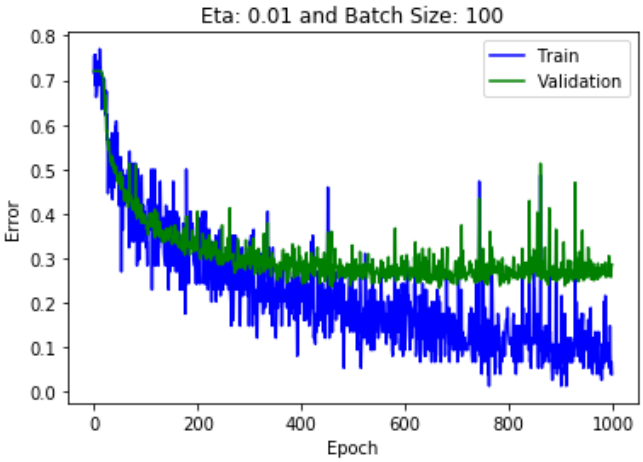



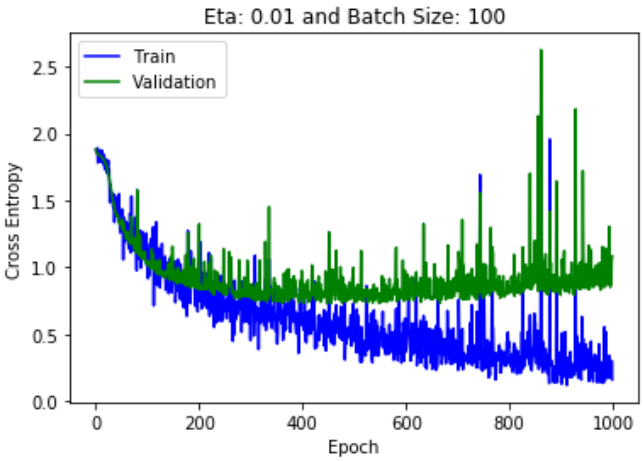

**[16, 20]**



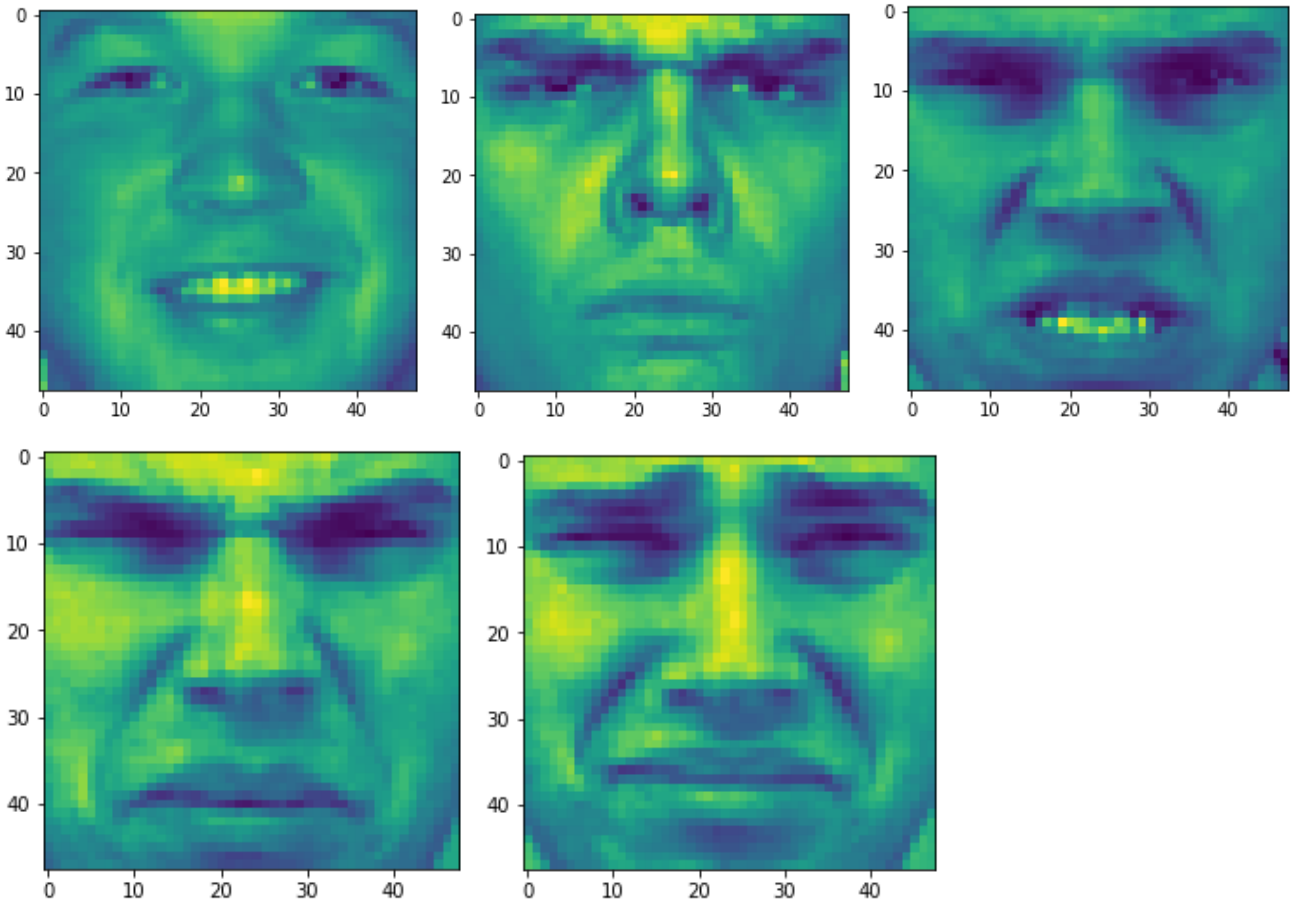

**[16, 80]**




Same argument applies here. We don't want to have too little/too many hidden units in our layers. Notice how the convergence changes as you change the number of hidden units. [2, 32] converges to around 0.5 error rate and [80, 32] converges to almost 0 error rate. We need to be wary of number of hidden units to achieve generality.

**2.5) Network Uncertainty**



```
t[0:5], pred[0:5]
([3, 0, 0, 1, 4], [3, 0, 0, 1, 1])
```

Here are the five example faces where the network's top score was below 60% threshold. If the network has convergence in its error rate, then we can say lower misclassification rate. Classifier will sometimes be correct if it outputs the top scoring class anyways. For my example above (in order left to right), the faces were classified as 3, 0, 0, 1, 1, in order, when in fact they are actually 3, 0, 0, 1, 4. So, last face is misclassified as 4 (sad) instead of 1(anger). I would've guessed that to be sad rather than anger too. Nonetheless, the network is never perfect and has error rates. So, even if we output the top scoring class, it could be misclassified. But if we have perfect prediction on the training set, we would have overfitting and little to no generalization.