Adam Gross, Josh Palmer

February 27, 2017

Assignment 4: Neural Networks Write-up

Link to dataset:

https://github.com/adamMgross/ai-art/tree/master/scraper/examples

Our newest art dataset includes 675 images from artsy.net, a site dedicated to providing information and famous content from all styles of visual art. We are specifically interested in Surrealist and Impressionist paintings, so our images are about half surrealist and half impressionist. They all come from the most famous artists from those movements. We download these with a simple web-scraper that scrapes the top images from each artist's page that we specify to the scraper. Thus we can add more URLs for more artists' pages to download additional instances easily.

For each trial, we varied one hyperparameter/variable while fixing the others at their "standard values". The hyperparameters and the values they took on (with **standard** values in bold) are as follows: the number of hidden layers [0, **1**, 2], the number of neurons in each hidden layer [10, **30**, 50], the mini batch size [1, **10**, 100], and the learning rate [0.01, **3.0**, 30]. Then we trained the network three times with these parameters (each time training on the training set) – once using the training data as the test data, once using the validation data as test data, and once using the test data as test data.

On the following pages are the results of our trials. Each row represents a trial, presenting the values of each hyperparameter as well as a plot/interpretation of the testing accuracies across the three datasets. The first is the MNIST dataset (labeled 28 x 28 handwritten digits), the second is our newest art dataset (surrealist or impressionist painting uniformly resized to 30 x 30), and the third is the old art dataset used in a previous decision tree learning algorithm (10 discrete attributes and three labels: surrealist, impressionist, or neither). Splitting into training/validation/testing datasets for MNIST was already done for us from existing code. We randomly split up the newer and older art datasets as follows: 8/15 for training, 4/15 for validation, 3/15 for testing.

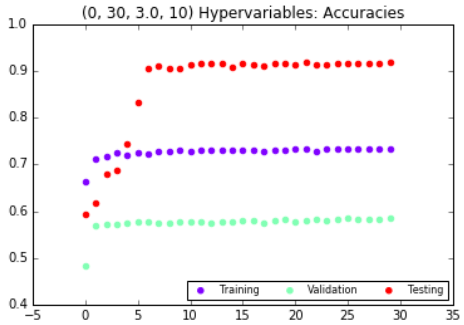
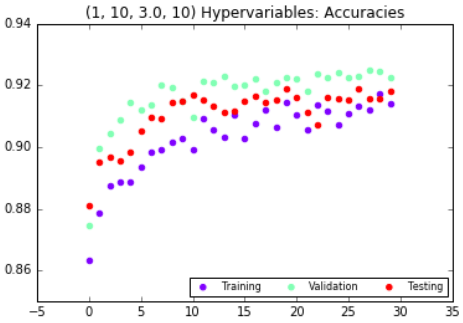
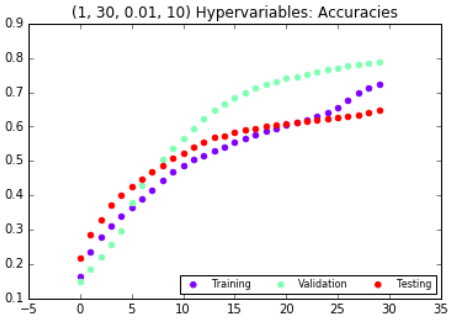
- **Note:** we also tried resizing our paintings to 150 x 150 – we figured that the network would perform better with better resolution. Surprisingly, classification suffered; it appears that the resizing process was a layer in it of itself. Since surrealist vs. impressionist is less about object recognition and perhaps more about overall hues, this makes sense.
- **Note:** The hardware was consistent across all trials: our machine had 4 CPUs, 16 GB of RAM, an Intel Xeon CPU E5-1620 v2 processor, and a clock speed of up to 3.7 GHz.

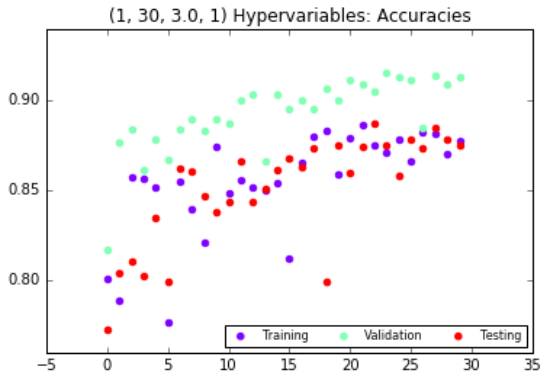
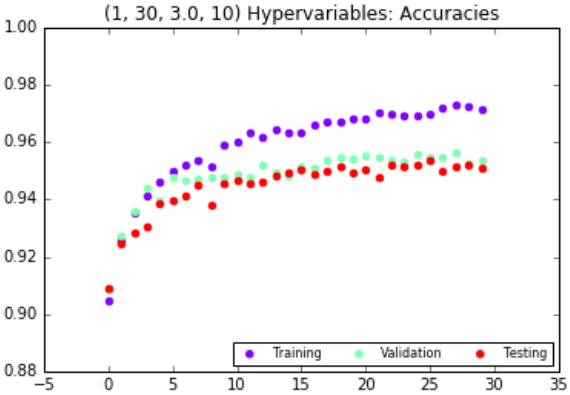
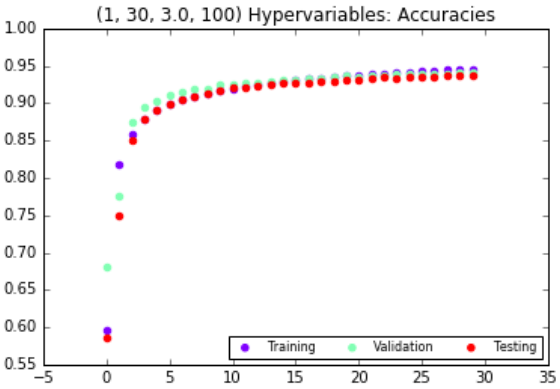
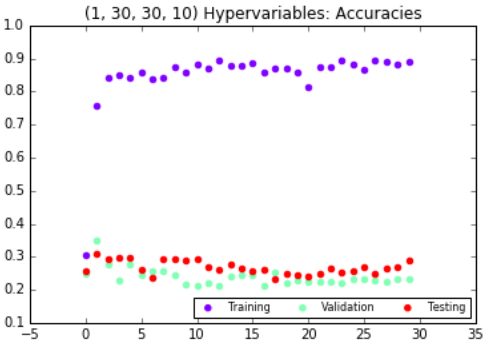
MNIST Dataset

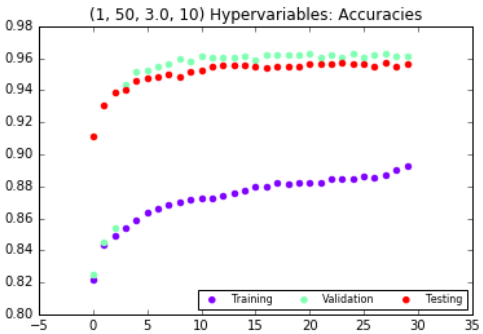
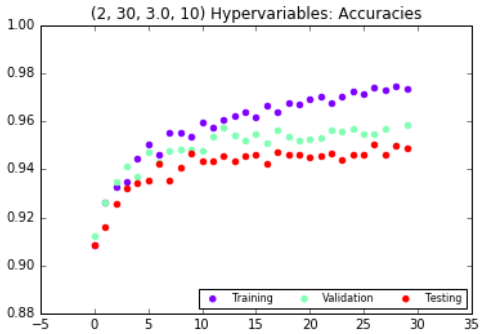
Size: 10,000 instances

Average training duration: 259 seconds

Epochs: 30 (x axis on plots)

Hidden Layers	Hidden Layer Size	Mini Batch Size	Learning Rate	Plot	Interpretation
0	N/A	10	3.0		It plateaued faster on the training data because the algorithm was already suited for it. Higher accuracy for testing data must be due to implicit bias in hyperparameters.
1	10	10	3.0		Hidden layer allowed network to train in a more robust, noise independent manner.
1	30	10	0.1		Decreased learning rate allowed descent to proceed more smoothly.

1	30	1	3.0		Mini batch size too small. Samples provided poor representation of entire space, so behavior is inconsistent.
1	30	10	3.0		Good hyperparameter settings. Training accuracy should be higher, accuracy should go up over epochs.
1	30	100	3.0		Larger batch size allows for better representation of the space with samples chosen. Similarity in performance across datasets is indicative of a good network.
1	30	10	30		Too large learning rate. Clearly biased the algorithm in favor of the training data used.

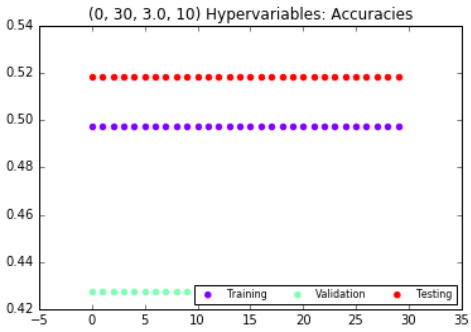
1	50	10	3.0		Unclear results here. Hyperparameter settings must have implicitly biased the algorithm to be more agreeable with the testing and validation sets.
2	30	10	30		Increase in hidden layers allowed for overall improvement.

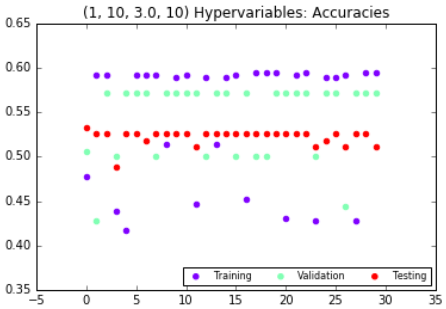
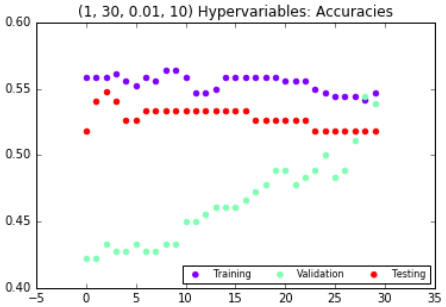
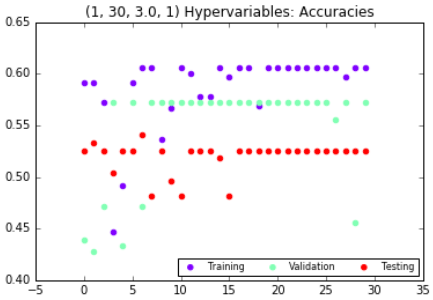
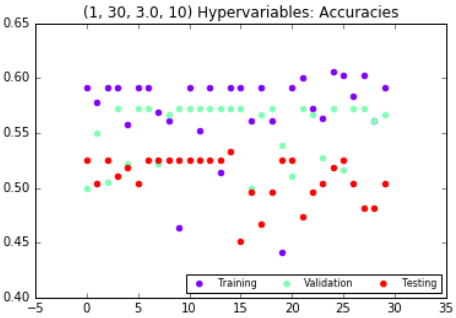
Our dataset

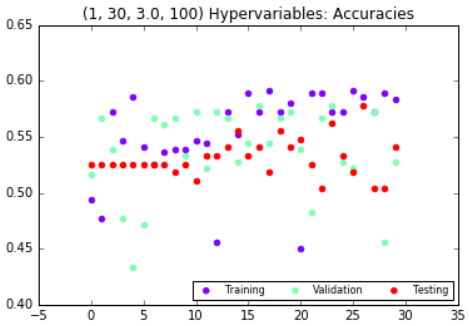
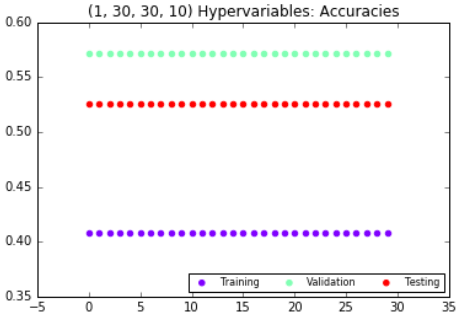
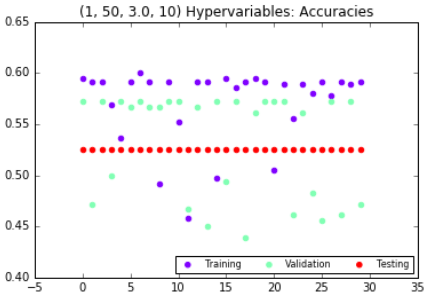
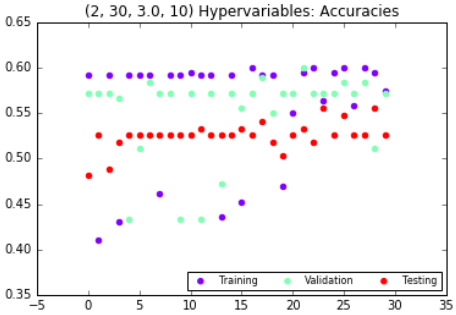
Size: 675 instances

Average training duration: 2.19 seconds

Epochs: 30 (x axis on plots)

Hidden Layers	Hidden Layer Size	Mini Batch Size	Learning Rate	Plot	Interpretation
0	N/A	10	3.0		No hidden layers combined with small dataset restricted algorithm to staticness.

1	10	10	3.0		Good that it was more accurate in general with the training set. Small dataset accounts for variance.
1	30	10	0.1		Larger hidden layer helped consistency. Decreased learning rate improved smoothness of accuracy over time.
1	30	1	3.0		Increased learning rate, combined with batch size of 1 caused more fluctuation over epochs.
1	30	10	3.0		Should be better than previous plot with larger batch size, but ~0.60 limit suggests that overfitting as well as significant variance between datasets is occurring.

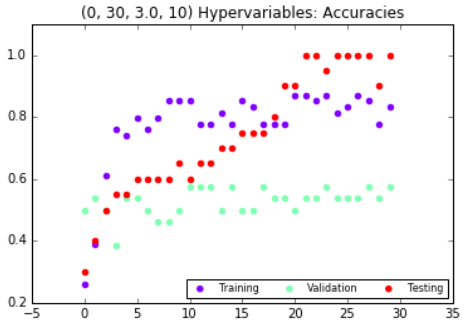
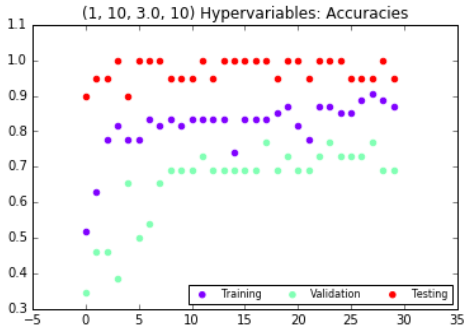
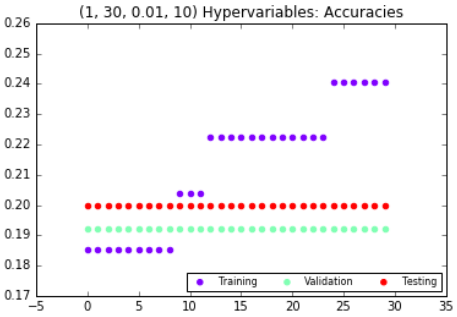
1	30	100	3.0		Significant jumping around most likely due to mini batch size of 100 being very large relative to the dataset size. Thus algorithm prone to overfitting different noise from iteration to iteration.
1	30	10	30		Learning rate too large. Gradient descent hits minimum fast and probably overshoots other better minima.
1	50	10	3.0		Interesting that testing performance was consistent and training/validation was not. Perhaps the topology was well suited for the testing dataset but not the other two.
2	30	10	30		Learning rate too high – inconsistent jumps. Hidden layers allowed for some general increase in accuracy over time, however.

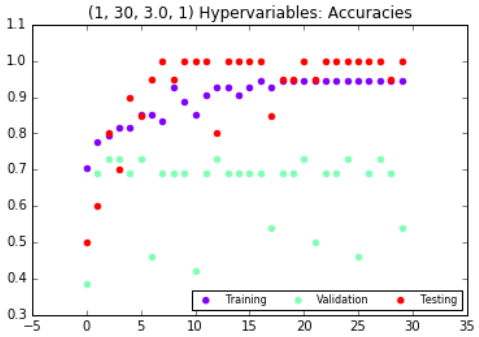
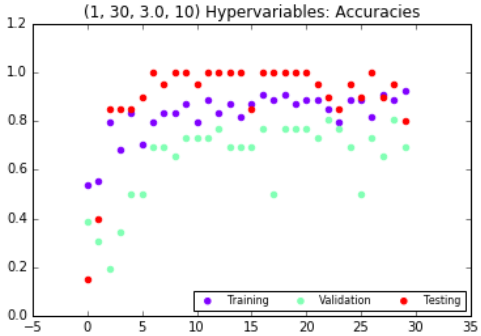
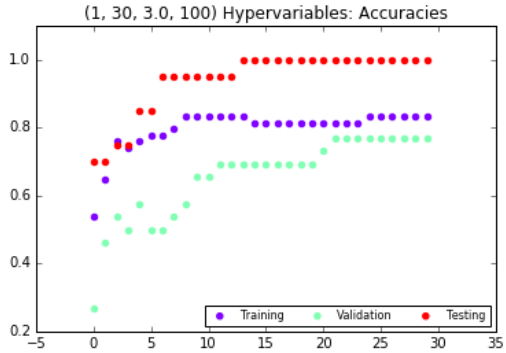
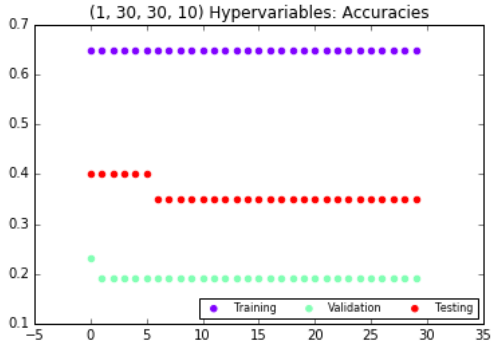
Assignment 3 dataset

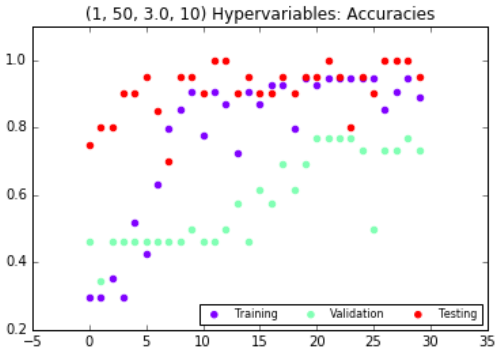
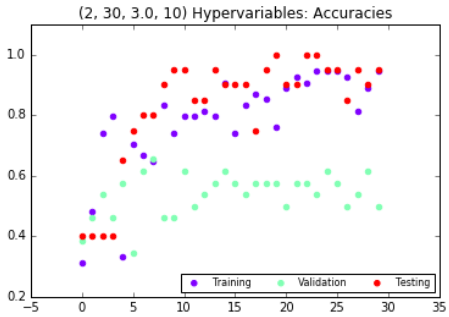
Instances: 100

Average training duration: 0.138 seconds

Epochs: 30 (x axis on plots)

Hidden Layers	Hidden Layer Size	Mini Batch Size	Learning Rate	Plot	Interpretation
0	N/A	10	3.0		As before, training plateaued faster because the network's weights were designed to converge towards its labels. And validation accuracy stopped getting better as training improved
1	10	10	3.0		Much better accuracy with addition of a hidden layer; resulted in faster convergence
1	30	10	0.1		Low learning rate means that convergence occurred slower. Larger mini batch size means less volatile swings in accuracy across epochs.

1	30	1	3.0		Erratic performance of validation due to low mini batch size; interestingly, there was slower convergence for training set --- a low mini-batch size → less accurate gradient → less effective minimizing of cost function → less training set accuracy
1	30	10	3.0		Less erratic validation performance because of "standard" mini-batch size
1	30	100	3.0		Even less erratic performance all around because of even larger mini-batch size; network topology seems ill-suited to training set still
1	30	10	30		Learning rate way too high; can't find minimum of loss function so no increase in accuracy across epochs; validation set lower accuracy than randomly picking (1/3 with three classes)?

1	50	10	3.0		Interesting…slower convergence with more neurons per layer…perhaps requires more iterations to learn additional weights; no meaningful difference in ultimate accuracy after 30 epochs
2	30	10	30		Same as previous, more neurons (additional layer) yielded slower convergence; lower accuracy on validation set can be chalked up to small data size