Classifying Fashion Items From Images Using Logistic and Softmax Regression

Mitchell Zhang
miz134@ucsd.edu

Jun (Moses) Oh
j6oh@ucsd.edu

Abstract

In our project, we use logistic and softmax regression on a fashion item dataset describing clothing images to classify them into correct categories. Each item (image) comes with one label of 10 categories. By using regression on predicted versus actual label, our program learned to predict and classify these images at up to over 99.8% accuracy on a test set. However, the speed and accuracy of the program differs heavily depending on number of labels, number of principle components (dimension reducing), learning rate, batch size, loss threshold (before early stopping), and other features. Some of our key findings include how these factors, which we call hyperparameters, affect the speed and accuracy, and what our chosen optimal rates were. We also compared the average loss per item for both the testing and validation set. One interesting finding was that there was a larger discrepancy between these two on some datasets. For logistic regression, our final test accuracy was about 99.85% on the first dataset and 83.5% on the second dataset. For softmax regression, our final test accuracy was about 83.8%. These values were tested after discussion about speed vs. accuracy tradeoffs on hyperparameter decisions.

1 Introduction

We used a dataset composing of fashion item images with one of ten labels. We filtered the data and one-hot encoded the labels as needed. We shuffled the dataset, and split the dataset into ten folds. For each fold, we used one as a holdout validation set and the other nine as training sets. We initialized weights (including a bias weight) to 0, and then used either the logistic regression or softmax regression with stochastic gradient descent to train the model on the training set. We separated the data into batches when calculating the gradient/activation function.

For logistic regression, we used these functions:

$$\hat{y} = P(C_1|x) = \frac{1}{1+e^{-w^T x}} \tag{1}$$

$$P(C_0|x) = 1 - \hat{y} \tag{2}$$

$$E(w) = -\sum_{n=1}^{N} \{y^n ln(\hat{y}^n) + (1-y^n) ln(1-\hat{y}^n\} \tag{3}$$

$$-\frac{\partial E(w)}{\partial w_j} = \sum_{n=1}^{N} \{(y^n - \hat{y}^n) x_j^n\} \tag{4}$$

For softmax regression, we used these functions:

$$\hat{y}_k^n = \frac{e^{a_n^k}}{\sum_k e^{a_n^k}} \tag{5}$$

$$a_k^n = w_k^T x^n \tag{6}$$

$$E = -\sum_n \sum_{k=1}^{c} y_k^n ln(\hat{y}_k^n) \tag{7}$$

$$-\frac{\partial E^n(w)}{\partial w_{jk}} = (y_k^n - \hat{y}_k^n)x_j^n \tag{8}$$

E is the loss function.

$\frac{\partial E}{\partial w}$ is the activation function

$\hat{y}$ is the prediction function (probability that x is in class k, given w)

x is the input

y is the actual target

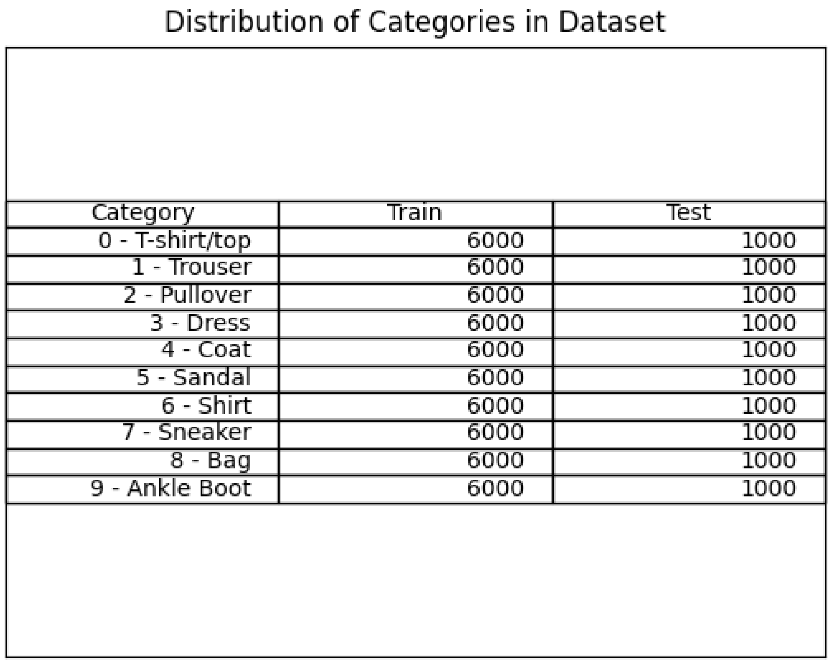
w is the weights

Our hyperparameters consisted of learning rate, threshold, batch size, number of principle components (PCs), and number of epochs. Learning rate had to be fine tuned particularly for each dataset and regression function, as too high or low of a learning rate would stop the training and learning too early. Threshold was the level we set the difference in loss to be too low to continue learning, and so we would early stop the training. Lowering the threshold would increase run time, but slightly increase our accuracy and make our number of epoch and accuracy over the training and validation sets more consistent. Since we implemented stochastic gradient descent, we used learning in sets of batches. Decreasing batch size would drastically increase run time while only slightly increasing accuracy. Number of PCs is the number of principle components we used during principle component analysis to reduce the number of dimensions. Decreasing number of PCs would greatly increase run time but lower accuracy, while increasing number of PCs would decrease run time but increase accuracy. Number of epochs had little effect since we implemented early stopping, unless we changed hyperparameters specifically to target longer epochs.

For logistic regression, our final test accuracy was about 99.85% on the first dataset and 83.5% on the second dataset. For softmax regression, our final test accuracy was about 83.8%. This was tested on a test set that was not used to train on. This is the total percentage correct of the whole test set, used with the weight that gave the highest accuracy on the validation set when training.

2 Dataset

The dataset consists of 60,000 training set examples and 10,000 testing set examples. These examples describe a 28x28 image and come with a label classifying it one of ten labels: T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle Boot. We primarily focus on classifying between T-shirt/top or Ankle Boot and Pullover or Coat for logistic regression and classifying between all 10 categories for softmax regression. We converted the labels to one-hot vectors for both logistic and softmax regression. For logistic regression, we filtered out all examples that did not follow the label of the two categories that we were classifying.


Distribution of Categories in Dataset

Category	Train	Test
0 - T-shirt/top	6000	1000
1 - Trouser	6000	1000
2 - Pullover	6000	1000
3 - Dress	6000	1000
4 - Coat	6000	1000
5 - Sandal	6000	1000
6 - Shirt	6000	1000
7 - Sneaker	6000	1000
8 - Bag	6000	1000
9 - Ankle Boot	6000	1000

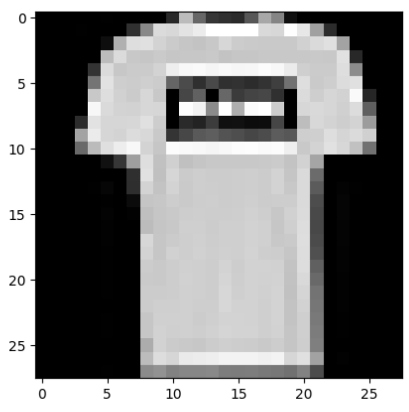


Figure 1: Category 0 - T-shirt/Top

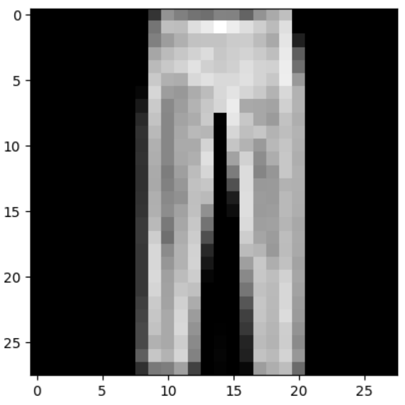


Figure 2: Category 1 - Trouser

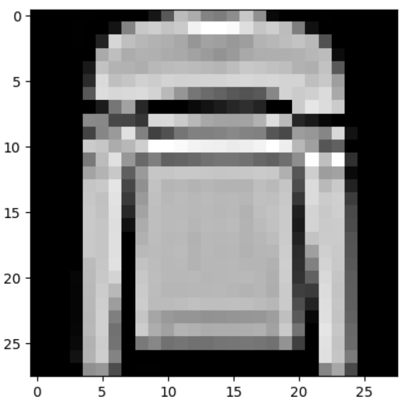


Figure 3: Category 2 - Pullover

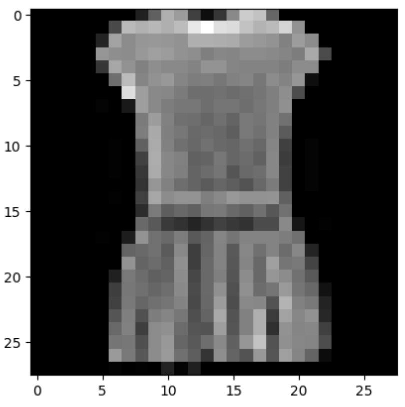


Figure 4: Category 3 - Dress

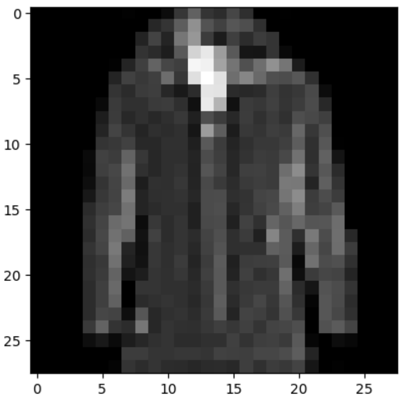


Figure 5: Category 4 - Coat

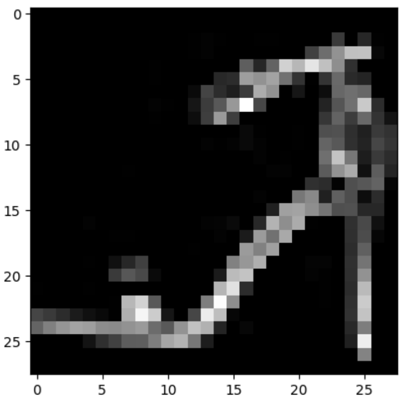


Figure 6: Category 5 - Sandal

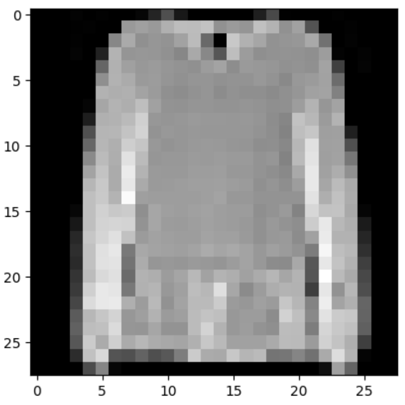


Figure 7: Category 6 - Shirt

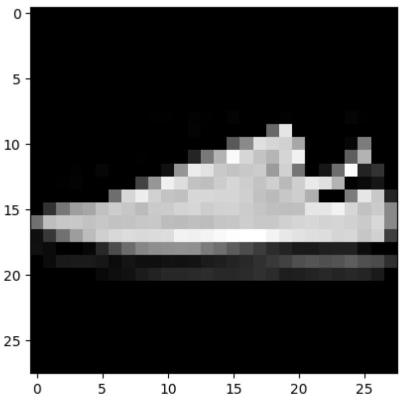


Figure 8: Category 7 - Sneaker

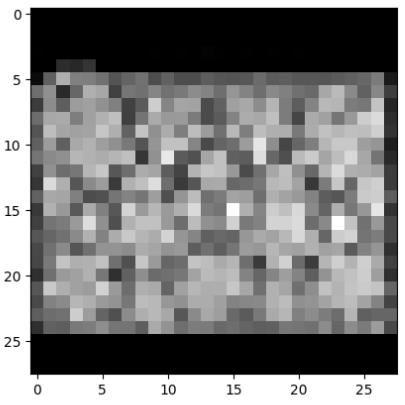


Figure 9: Category 8 - Bag

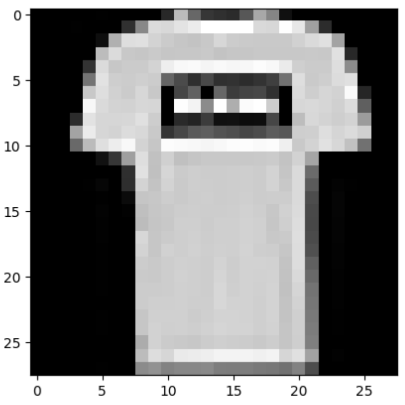


Figure 10: Category 9 - Ankle Boot

3 Principle Components Analysis

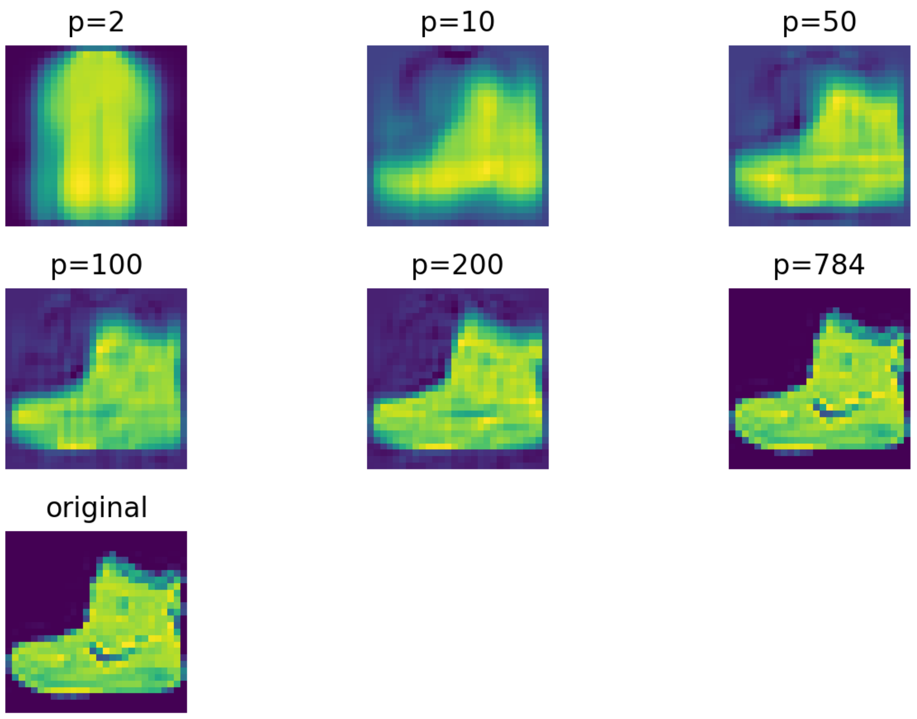


Figure 11: PCA Reconstruction

Loss of information is apparent for every p less than 784. This makes sense because PCA reduces the dimensions by capturing the larger trends of the data and leaving out those with little correlation.

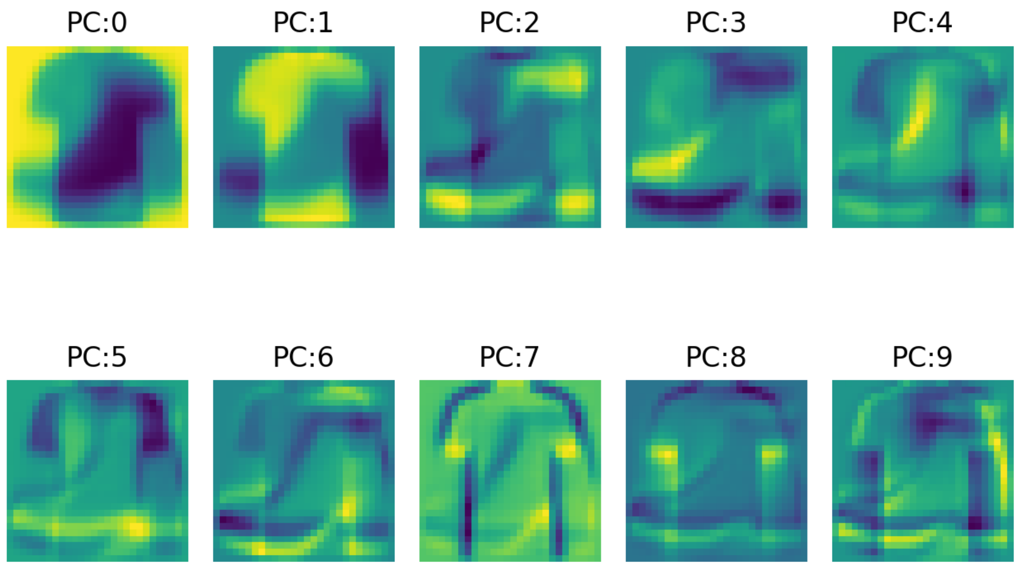
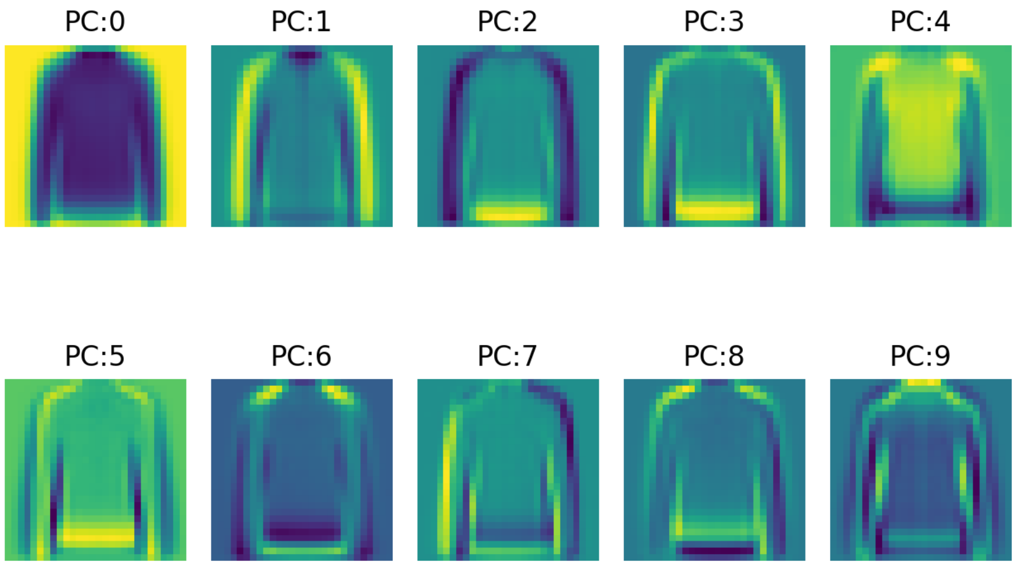


Figure 12: Set 1 Principal Components

In figure 12, we see the most important pixels of each category - T-shirt and Ankle boot. We see the shape or outline of the two labels (T-shirt/Top and Ankle Boot) that we are looking to classify. Since we have filtered out all images with other labels, all images in this dataset will be either of these two labels. In PCA, we analyze the most important components/pixels, which would be the shirt/boot. As a result, it makes sense to see the general shape or outline of the shirt/boot as part of the PC image. We can also point out that the distinctions of each shape becomes more and more apparent as the PC decreases. This makes sense the PCs are sorted from most correlation to least correlation.



0

Figure 13: Set 2 Principal Components

In figure 13, similar to figure 12, we see the shape of the two categories in Set 2 - Pullover and Coat.

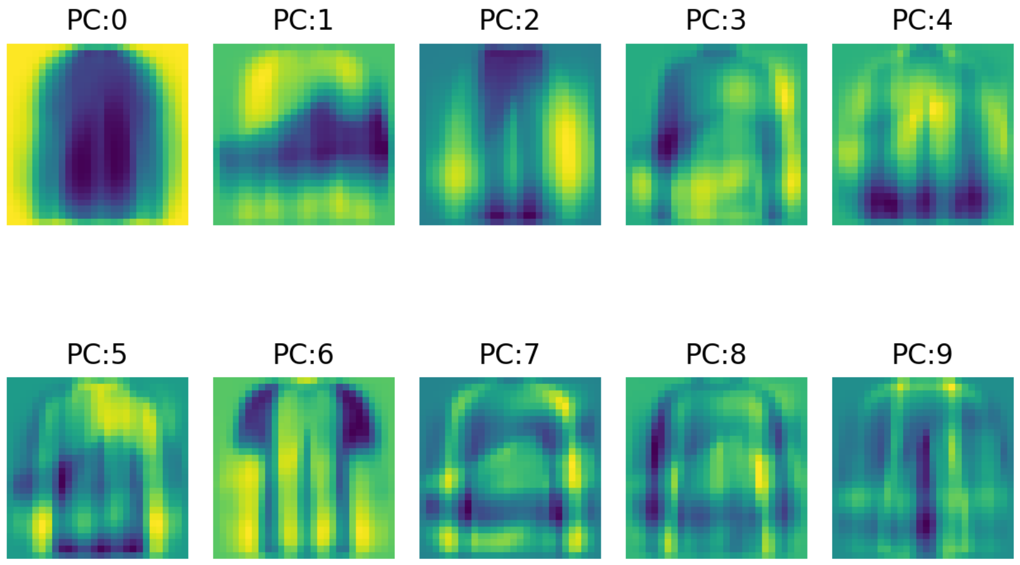


Figure 14: Set 3 Principal Components

In figure 14, we see the general outlines of every category in the data set since we are not filtering the training set at all. PC:0 can be interpreted as the most ”generic” image of all the categories.

4 Logistic Regression

For the first dataset, we filtered out all images with labels other than 0 (T-shirt/Top) and 9 (Ankle Boot). Prior to using the training set, we made sure to shuffle the dataset and split the data into ten folds. We then trained on one fold at a time, and took the best accuracy on the validation set from all folds as the set of best weights to test the final testing set on. The resulting final accuracy on the test set was around 99.7-99.85%. Too high of a learning rate (0.001 and 0.0005) resulted in the epochs ending too soon (since we implemented early stopping), a rigid average loss curve. We decided to test with a learning rate of 0.0001, which ended at epoch 11 for a threshold of 0.0001. This threshold was decided after testing multiple thresholds (0.001, 0.00001 -.00001, 0), but 0.001 gave us a consistent accuracy at a relatively short run time. Batch size did not seem to have a large impact, so we maintained it at 512. The number of PC did not seem to have too large of an impact. A relatively low PC number of 10 resulted in a test accuracy of 99.7%. To slightly increase this accuracy, we decided on using PC of 50, which resulted in an accuracy of 99.85% but still ran at a reliable pace. A higher PC did not have marked improvement in accuracy, but did increase run time. While small changes to certain single hyperparameters did not affect run time too much, changing many of them (such as PC, threshold, learning rate) to too low/high would drastically increase run time to upwards of 10 minutes. These hyperparameters chosen resulted in a run time of less than 1 minute.

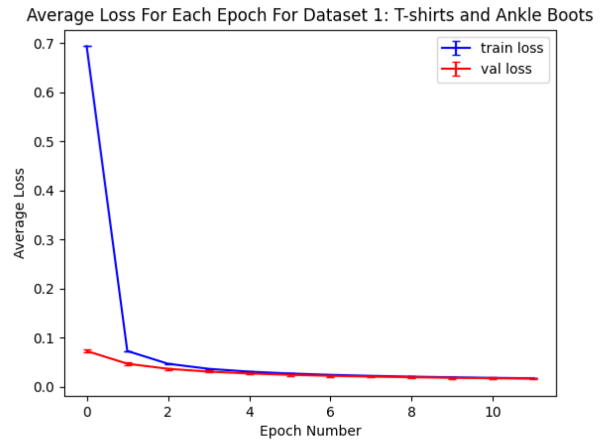


Figure 15: PC = 100, Epoch = 100, Batch = 512, Learning Rate = 0.0001, Threshold = 0.0001, Final Accuracy = 99.85%

For the second dataset, we filtered out all images with labels other than 2 (Pullover) and 4 (Coat). The resulting final accuracy on the test set was around 82-85%. We maintained the relatively low learning rate and batch size, as these provided the best speed vs. accuracy tradeoffs, much as in the previous dataset. However, we chose to increase number of PCs (200) and lower threshold (-0.00001) because this dataset had lower accuracy, and increasing/decreasing these hyperparameters respectively resulted in the highest accuracy increases. The PC was chosen because these images had similar images, so it would have been easier with more components to analyze to classify between the two. Since there was also more error, we decreased the threshold to -0.00001 to allow our model to train and learn for longer epochs (about 11-20) to distinguish between the two. Our final accuracy for this model with these settings was 83.5%. The run time was about 1-2 minutes long.

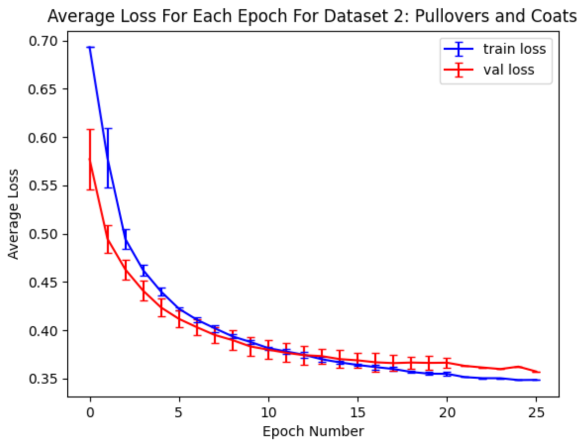


Figure 16: PC = 200, Epoch = 100, Batch = 512, Learning Rate = 0.0001, Threshold = -0.00001, Final Accuracy = 83.5%

There is a marked gap between the first dataset and the second. While the first dataset has an accuracy typically above 99% for a variety of hyperparameters, the second dataset varies much more in regards to the hyperparameters. Moreover, the accuracy is only about 80-85%. This difference could be attributed to the categories that we're testing. For example, the first dataset consists of images of shirts and boots - two relatively unlike images, as compared to pullovers and coats. Since it is easier to distinguish between a shirt and boot, it is expected that the accuracy of the first data set is higher.

One interesting finding is that the train loss is usually higher than validation loss initially. This could be attributed to our creation of the weights and calculation of the loss. We initialized the weights to all be 0. While we used this to test train loss, we only tested validation loss at the end, once all the weights were completely updated for that epoch. This could possibly explain this discrepancy, as the validation loss at testing would slightly be more ”accurate” than the train loss, which is computed before and during the learning and training process. We trained the second dataset for longer (lower threshold), which would explain the higher number of epochs.

Another interesting finding is that the average loss for dataset 1 is much lower than the average loss of dataset 2, especially initially for the train loss. This may also be attributed to the different categories, as pullover and coats are much more similar and therefore harder to classify. As a result, there is higher loss as the model seeks to train to classify between the two, but computes more errors while doing so.

The standard deviation is much lower for the train loss in both sets. This could be attributed purely to the sample size. Since we are splitting the data set into 10 folds, and using 9 of them for the train set but only 1 for the validation set, the train set has 9x more examples than the validation set. A larger sample size results in lower variance.

5 Softmax Regression

Our process for softmax regression was similar to logistic regression, focusing on changing only the needed calculation differences. With softmax regression, we did not filter out any of the images and trained and tested using the whole dataset, as opposed to logistic regression. This is because we are looking to classify an image between all 10 categories, rather than just distinguishing between two categories. To encourage learning, we tuned the hyperparameters to learn more iteratively and for longer epochs by setting learning rate to 0.0006 and setting a low threshold of -0.01. The learning rate was critical here, as even small amounts of change (0.0004, 0.0001, 0.001) would change the final accuracy down to 75%. We lowered the threshold to encourage more epochs to learn from, which made the testing/validation accuracy much more consistent across the different folds. Decreasing batch size drastically increased run time and number of epochs at minimal increase in accuracy. For example, decreasing batch size only increased accuracy by 0.05% - a tradeoff we deemed too low of an improvement for to warrant the approximately 30 minute run time. We kept PC at 200 after testing 50, 100, and 784. Decreasing the PC lowered final accuracy but increasing the PC to 784 increased final accuracy by 0.3% made run time longer as well. The resulting accuracy was 83.8% on the test set with about a 10-15 minute run time.

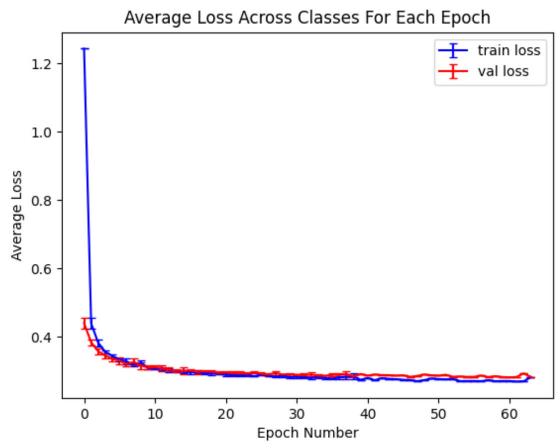


Figure 17: PC = 200, Epoch = 100, Batch = 512, Learning Rate = 0.0006, Threshold = -0.01, Final Accuracy = 83.8%

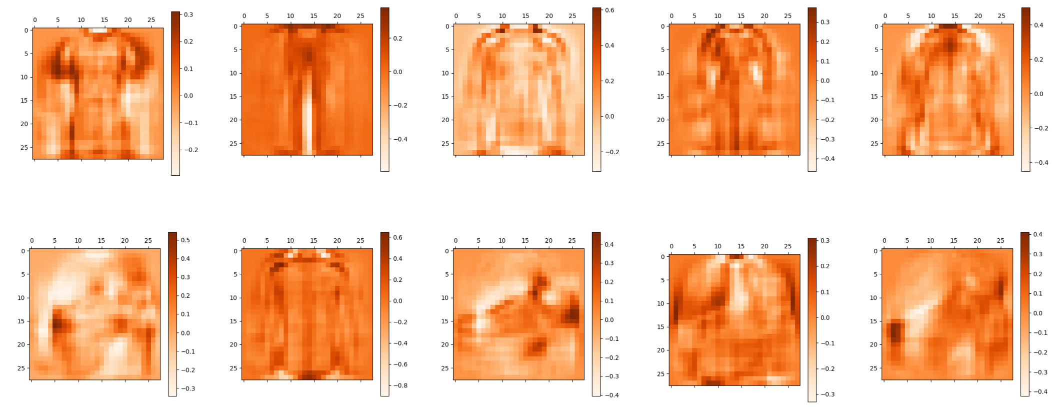


Figure 18: Collage of the best weights for each category

The softmax regression average loss across classes looks very similar to the logistic regression loss function. The overall standard deviations are much lower, likely because much of larger sample size. The loss curve also seemed to waver up and down more than the logistic regression loss curve. This may be because it is hard to be consistent and accurate with many more categories, hence the fluctuation in the loss. The epoch number also increased dramatically. This is because we tuned the hyperparameters to allow for longer training. The best weights visualized for each category look much like images itself. It it easy to see the outlines and shapes of the overall figure. However, there are a few categories that are harder to distinguish and are a little more ”blurry”. This may because there is another category that has a similar shape to the image, and therefore is harder to distinguish. One thing to note is that some weights are much darker overall and have a more consistent color throughout, rather than sharp contrasts. This may be because they have less sharp edges and share more edges with other labels as well.

6 Team Contributions

Mitchell: Worked on both the logistic and softmax regression coding. Much of this was done in conjunction with Moses through pair programming through VS Code Live Share. Met together many times to work on the coding aspect together. Wrote parts of the NIPS report, such as the abstract/introduction/logistic regression portions.

Moses (Jun Hwee): Worked on data pre-processing such as min-max normalize and one-hot encoding. Implemented PCA. Worked on logistic and softmax regression with Mitchell through pair programming. Wrote part of the NIPs report such as the Principal Components Analysis portion.

7 Related work

[1] Mehrab Tanjim (2020) 151B Discussion pp. 11-21. UCSD

[2] Gary Cottrell (2020) Logistic Regression Multinomial Regression Lecture 2 pp.46-47 UCSD

[3] Gary Cottrell (2019) Maximum Likelihood: The theoretical basis for our objective functions pp.34 UCSD