Leo Kim and George Mcgurkin
CS135: Machine Learning
November 18th, 2023

Project B - Classifying Images

Problem 1

1A: Dataset Exploration

Table 1A

Class	Train Set Count	Valid Set Count
Dress	400	100
Pullover	100	100
Top	1	100
Trouser	1	100
Sandal	800	100
Sneaker	800	100

Caption: The major challenge we foresee in building classifiers for the provided training data is that there needs to be more top and trouser silhouettes in the training set. With only one example of each, our model will not be able to identify tops or trousers in the held-out data effectively.

1B: Model Development

We did not perform any preprocessing on our training and validation data because we determined that the provided feature vector representation of grayscale values ranging from 0 to 255 for each pixel of the corresponding image was sufficient to train our MLP. When performing our hyperparameter search, we optimized balanced accuracy as our scoring metric, as it is the metric used to evaluate our model on the leaderboard. We performed a grid search to select our model's hyperparameters, searching over alpha values ranging from 1e-5 to 1e5, hidden layer sizes from 2^3 to 2^8, four different random states, and max iterations of 10, 100, 1,000, and 10,000. Our final selected hyperparameters are an alpha value of 0.1, hidden layer size of 128, random state of 2, and a max iteration of 1,000. We choose LBFGS over SGD or ADAM for our solver function because our dataset is not large enough to necessitate the stochasticity of SDG or ADAM. Because our dataset is small enough to fit in memory, LBFGS can use the entire dataset when choosing the next step, allowing for a smooth and faster optimization curve. In addition, omitting hyperparameters like learning rate and batch size greatly sped up our grid search process. We decided to search over many different random states because the loss function of MLPs are not convex, so the starting state has a significant effect on the ultimate weight optimization. We decided to search over different max iterations, because early stopping can be a good control against overfitting the training data.

Table 1B

Alpha Value	Train Set Balanced Accuracy	Validation Set Balanced Accuracy
0.00001	1.000	0.643
0.01	1.000	0.647
0.1	1.000	0.782
100	1.000	0.658
100,000	0.540	0.550

Caption: Each of the rows above represents 5 different models from our grid search, each using ReLU as the activation function, 128 nodes in the hidden layer, a random state of 2, and a max iteration of 1,000. The only thing that varies between these models is the alpha value. As the table shows, we found that an alpha value of 0.1 yielded the best results on the validation data, with a balanced accuracy score of 0.782. Models with lower alpha values, which correspond to less weight penalization, were able to perfectly predict the training data but performed worse on the validation set, which could be seen as evidence of overfitting. Models with higher alpha values, indicating more weight penalization, were not able to perfectly predict the training data and performed worse on the validation set, which could indicate underfitting.

1C: Model Analysis

Table 1C

	Predicted Class						
True Class	Class	Dress	Pullover	Top	Trouser	Sandal	Sneaker
	Dress	99	1	0	0	0	0
	Pullover	10	90	0	0	0	0
	Top	83	14	0	0	2	1
	Trouser	83	5	0	12	0	0
	Sandal	0	1	0	0	94	5
	Sneaker	0	0	0	0	4	96

Caption: While our model was effective at classifying dresses, pullovers, sandals, and sneakers, it was very poor at classifying tops and trousers. It only correctly labeled 12% of the trousers and

0% of the tops. This is likely because both trousers and tops were only represented by one image each in the training set, thus the model does not have sufficient data to effectively label images in either of the two classes.

1D: Training Set Modification

We plan on duplicating the training set images for the classes that initially have lower counts. We want to duplicate images in each set such that the new training set will have an equal count of images for every class. For the dress and pullover images, we will duplicate each image once to have 800 total images for each. For the top and the trouser images, we will have to duplicate the one image for each 799 times. We chose to duplicate the data points because our training dataset had an unequal distribution of data points for each class. Because of this, our optimization process (i.e. forward prop and back prop) would unevenly optimize the weights for classes with more data points. Having a more equal distribution, even though some data points are duplicated, would allow for each class to be equally weighted in our training process.

1E: Duplicated-Data Model Development

Table 1E

Alpha Value	Train Set Balanced Accuracy	Validation Set Balanced Accuracy
1	1.000	0.670
10	0.999	0.725
100	0.990	0.763
1,000	0.978	0.825
10,000	0.775	0.608

Caption: Each of the rows above represents 5 different models from our duplicate dataset grid search, each using ReLU as the activation function, 64 nodes in the hidden layer, a random state of 0, and a max iteration of 100. The only thing that varies between these five models is the alpha value. As the table shows, we found that an alpha value of 1,000 yielded the best results on the validation data, with a balanced accuracy score of 0.825. Overall we observe similar trends to our table from 1B, where lower alpha values are able to predict the training set well but perform poorly on the validation set, and high alphas perform poorly on both. One difference to note between these results and our findings from 1B is that our models of lower alphas (i.e. 100 or less) are not able to perfectly predict the training set. This is likely because we use a max iteration value of 100 as opposed to 1,000 from 1B, which allows us to stop early which can help us avoid overfitting.

1F: Modified Model Analysis

Table 1F

	Predicted Class						
True Class	Class	Dress	Pullover	Top	Trouser	Sandal	Sneaker
	Dress	96	2	0	2	0	0
	Pullover	9	89	0	0	2	0
	Top	39	0	44	14	3	0
	Trouser	14	8	0	78	0	0
	Sandal	0	0	0	0	95	5
	Sneaker	0	0	0	0	7	93

Caption: We can see that these results are significantly better than our model from 1C for top and trouser classification while being very similar for the other classes. These results support our modification procedure because they show that duplicating the training set images for the underrepresented classes greatly increased our model's ability to predict those classes.

1G: Submit to Leaderboard and Record Test-Set Performance

Our ultimate test set performance for our final model was a balanced accuracy score of 0.815. This performance was quite similar to our balanced accuracy score of 0.825 on our validation set in testing. The difference between the two balanced accuracy scores can likely be attributed to natural variability between the datasets.

Problem 2

<u>2A : Method Description</u>

Attempt That Did Not Make the Final Model

For our model in Problem 2, we attempted to employ the pre-trained MobileNet (V3) neural network that was trained on images from the “imagenet” dataset. This meant that we needed to perform significant preprocessing on our images in order to feed them to MobileNet in the correct format. First, we performed the same duplication from Problem 1 on our training set in order to ensure equal representation of all of the clothing classes. Then, we had to reformat our images from 28x28 grayscale images to 224x224 RGB images. In order to do this, we had to take the single dimension luminance values from our grayscale images and propagate them across the three R, G, and B dimensions. Then, we converted these 28x28 RGB image arrays into PIL images and used the built-in resize function to convert them to 224x224 RGB images. We then passed them into the MobileNet model.

Passing each image into the MobileNet model yielded an array of length 1,000, with each entry representing the likelihood that the image belongs to the corresponding class. After passing all our preprocessed images through the MobileNet model, we then used the 1,000 length arrays that the MobileNet returned for each image as the feature representation for our model. We decided to try and use the MobileNet results because we know that MobileNet is a pretrained model on large amounts of data, so we figured that it would be much better at reading images than our model just trained on our limited dataset. We knew that MobileNet didn’t contain the same classifications that we were searching for in our overall model, but our idea was that different clothing images would fit into different MobileNet classes and then by passing in the MobileNet values, our MLP would be able to draw meaning from the different class scores.

Final Model

After trying to develop a model that worked using the MobileNet using many different strategies and approaches we finally concluded that the MobileNet class values were not an effective feature representation for our classification model. This is likely because the built-in MobileNet classes were not varied enough to distinguish between our different clothing items. When looking into the specific classifications given by MobileNet, we found that clothing items belonging to different classes were being labeled under similar MobileNet classes such as “wool” and “toaster”. Thus, in our final model, we performed the same duplication from Problem 1 on our training set in order to ensure equal representation of all of the clothing classes and we included our validation data in our training data for the final version of our model. We used the same feature representation from Problem 1, which was a 1-D array representing a 28x28 grayscale image.

For our classification method we used an MLP with more than one hidden layer. We decided to use more than one hidden layer because we hoped that more hidden layers would give the model more predictive power by making the predictions less linear and able to fit more complex representations. We performed grid searches in order to find the optimal size and number of layers, as well as searching for other optimal hyperparameters.

2B: Model Development

When selecting our hyperparameters for our MLP classifier with more than one hidden layer, we decided once again to perform grid search, reusing our splitter from part 1, where our models were trained on the duplicated dataset, and our validation set was used to validate model performance on held out data. In addition to our hidden layer configurations from part 1, we also add two layer and three layer variations for each size, as the extra layers could give our model the ability to represent more complex (i.e. less linear) relationships between the features and the predicted y values. We initially searched over 11 alphas ranging from 1e-5 to 1e5, and then performed a second search over 10 alphas ranging from 1e3 to 1e5 once we identified the general area where our model performed the best. Once again we searched over 4 different random states, and 4 different max iteration values, those being 10, 100, 1,000, and 10,000, as more random states allowed us to try out multiple initial weight configurations, as the optimization curve for MLPs are not convex, and multiple max iterations allows us to stop early if our models become overfit after too many iterations. We used balanced accuracy as our scoring metric, as that's what our model is evaluated on in the leaderboard. Our final model was the best resulting model from our grid search, with the following hyper parameters:

- Alpha - 21544.35
- Hidden layer sizes - (16, 16)
- Max iterations - 100
- Random state - 102

We used both validation and training data to optimize our model for our final test data prediction.

Table 2B

Alpha Value	Train Set Balanced Accuracy	Validation Set Balanced Accuracy
1,668	1.000	0.757
12,915	0.958	0.788
21,544	0.934	0.876
35,938	0.723	0.836
100,000	0.167	0.167

Caption: Each of the rows above represents 5 different models from our grid search, each using ReLU as the activation function, 2 hidden layers with 16 nodes in each layer, a random state of 102, and a max iteration of 100. The only thing that varies between these five models is the alpha value. We can clearly see that the magnitude of the alpha value had a clear effect on the quality of our model's predictions on heldout data. We found that an alpha value of 21,544 yielded the best results on the validation data, with a balanced accuracy score of 0.876. We see that when the alpha value is too low, the model perfectly predicts the training set but has a worse prediction on the heldout validation set which shows overfitting of the training set. We also see that when the alpha value is too high, the model reduces to random predictions on both the train set and the validation set, showing underfitting of the data.

2C: Model Analysis

Table 2C

	Predicted Class						
True Class	Class	Dress	Pullover	Top	Trouser	Sandal	Sneaker
	Dress	98	4	20	3	0	0
	Pullover	4	96	0	0	0	0
	Top	20	19	58	2	0	0
	Trouser	3	5	0	92	0	0
	Sandal	1	0	2	0	89	8
	Sneaker	0	0	0	0	7	93

Caption: We found that these results were significantly better than our models from part 1 and our model was very effective at classifying clothing items within all classes besides Tops. These results do support our hypothesis from 1A because by increasing the number of hidden layers, our model was better able to understand the more subtle differences between similarly appearing classes.

2D: Submit to Leaderboard and Record Test-Set Performance

Our ultimate test set performance for our final model was a balanced accuracy score of 0.906. This improvement from Problem 1 in our balanced accuracy score was likely due to fact that our MLP was multi-layered as well as the fact that we fit our final model on a combination of our training and validation data. While a significant chunk of our project time was spent trying to get the MobileNet model to yield accurate classifications, our model improvements occurred after we finally decided to move on from the MobileNet model. If we could start this project again from the beginning, we would have allocated more time to improving our model from other angles and likely been able to achieve a higher balanced accuracy score by spending less time trying to work with MobileNet.