Neural Networks from Scratch Final Project - #3 Mystery Dataset

Edmund Tsou

Johns Hopkins University

B.S. Biomedical Engineering & Computer Science

Goal: Create the best MLP to classify mystery dataset

Dataset Exploration

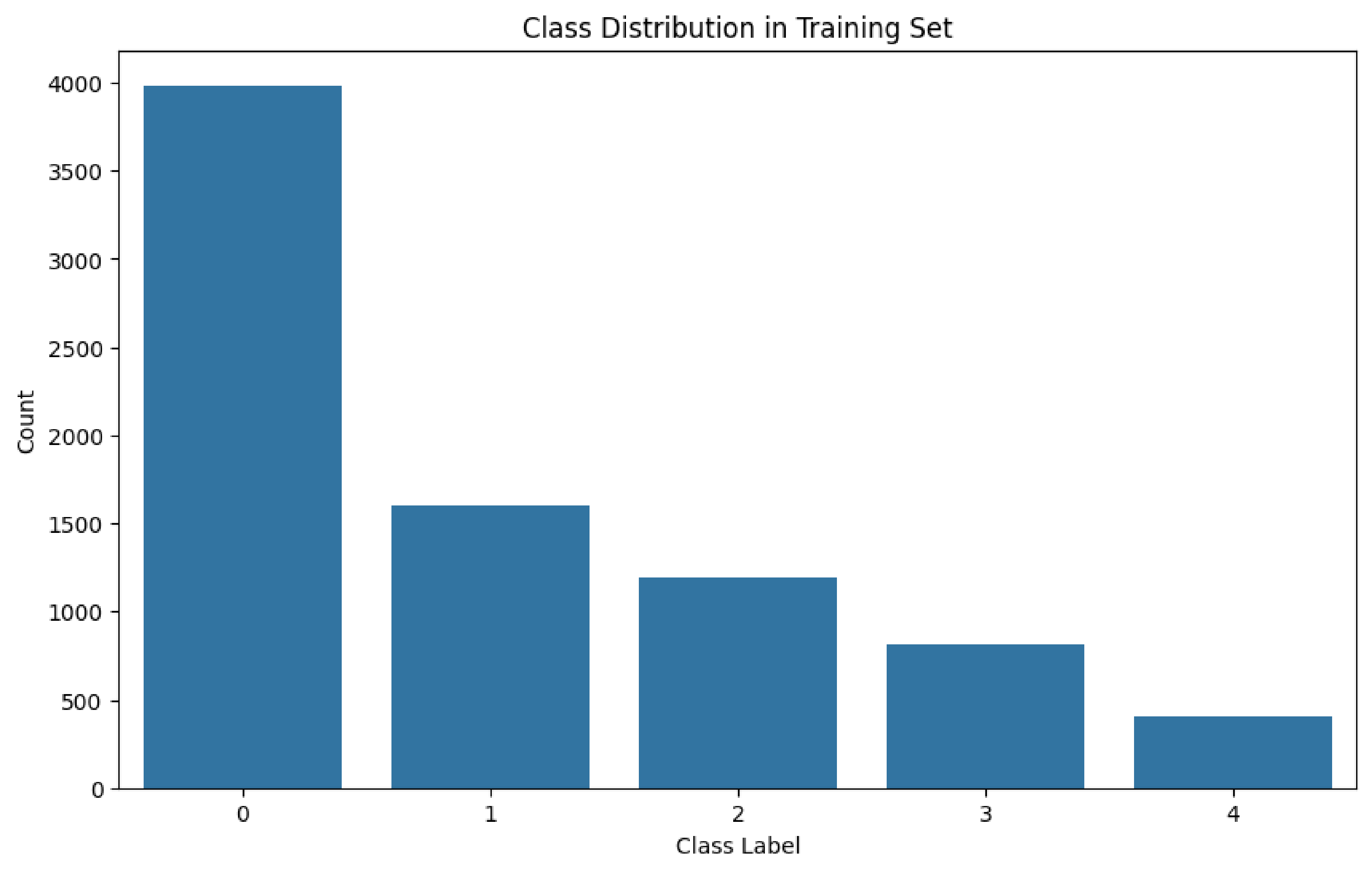
Dataset details:

CSV tabular data

- train: 8000 examples x 205 features
  - Each example had a class label (0, 1, 2, 3, 4) with 5 possible classes
- test: 2000 examples x 205 features with no labels

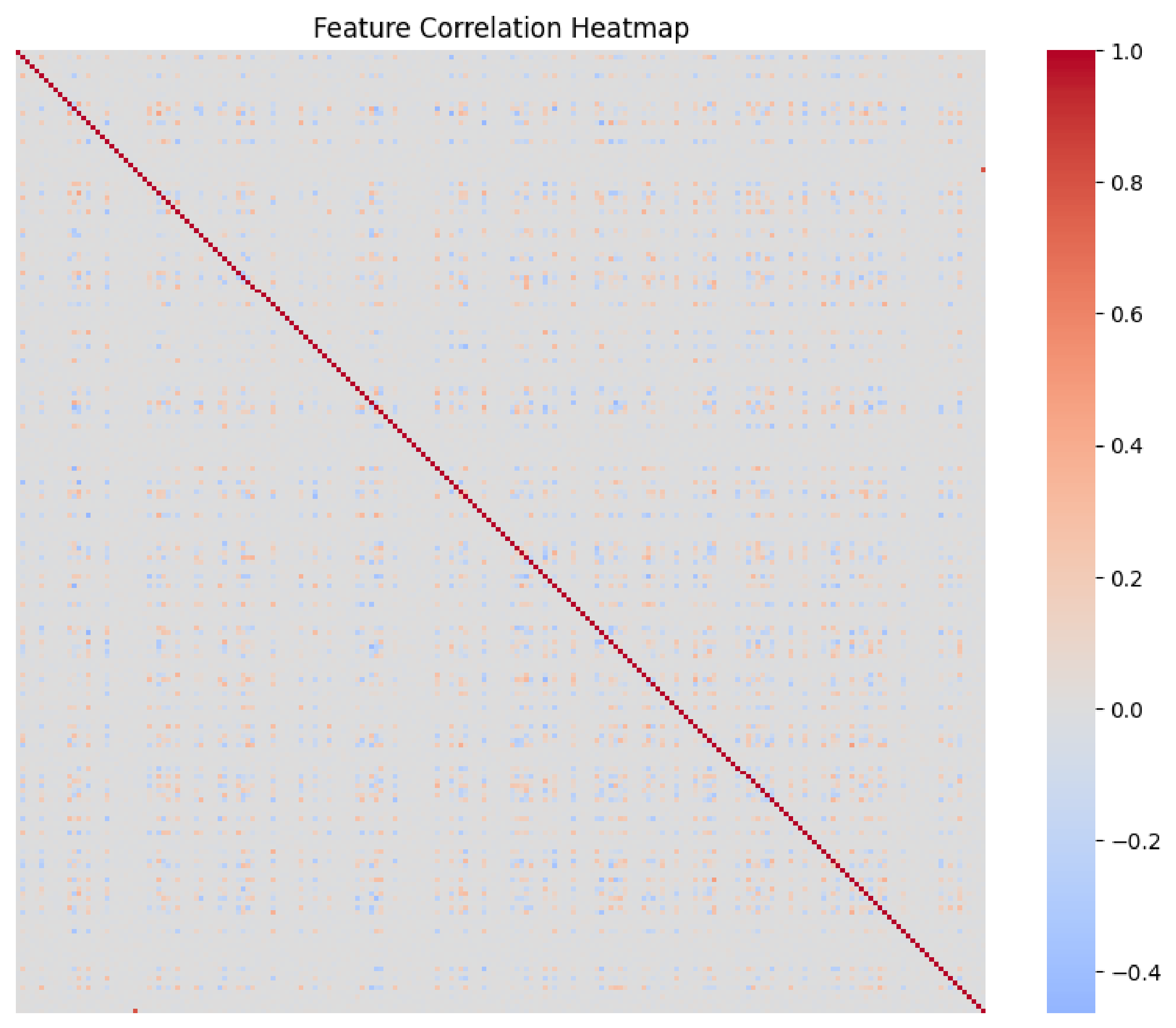
Dataset Exploration: class imbalance

Observation 1: clear class imbalances -> will be theme later



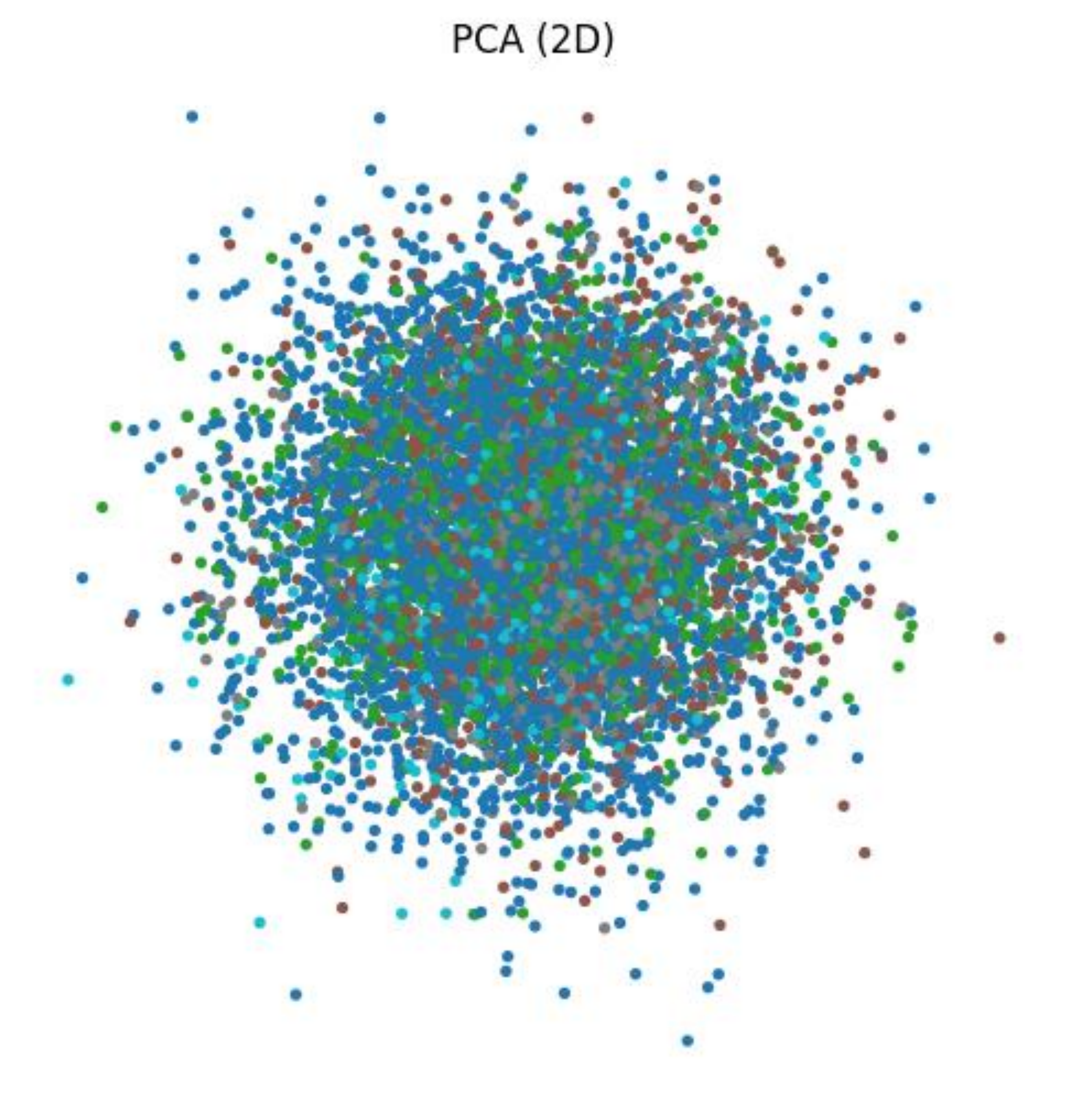
Dataset Exploration: feature correlations

Observation 2: features are roughly uncorrelated



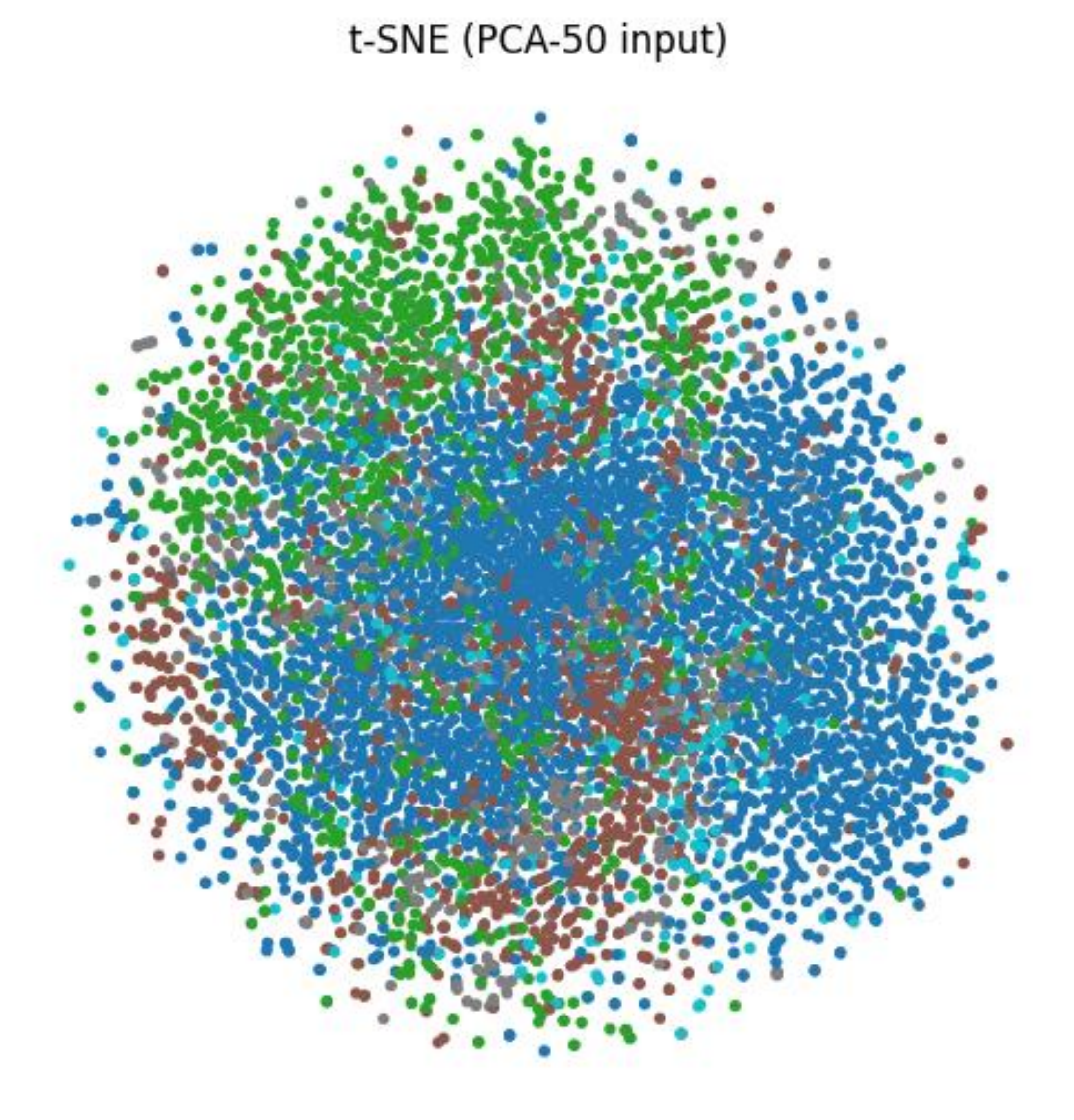
- Uses Pearsons correlation
- No multicollinearity to worry about

Dataset Exploration: PCA



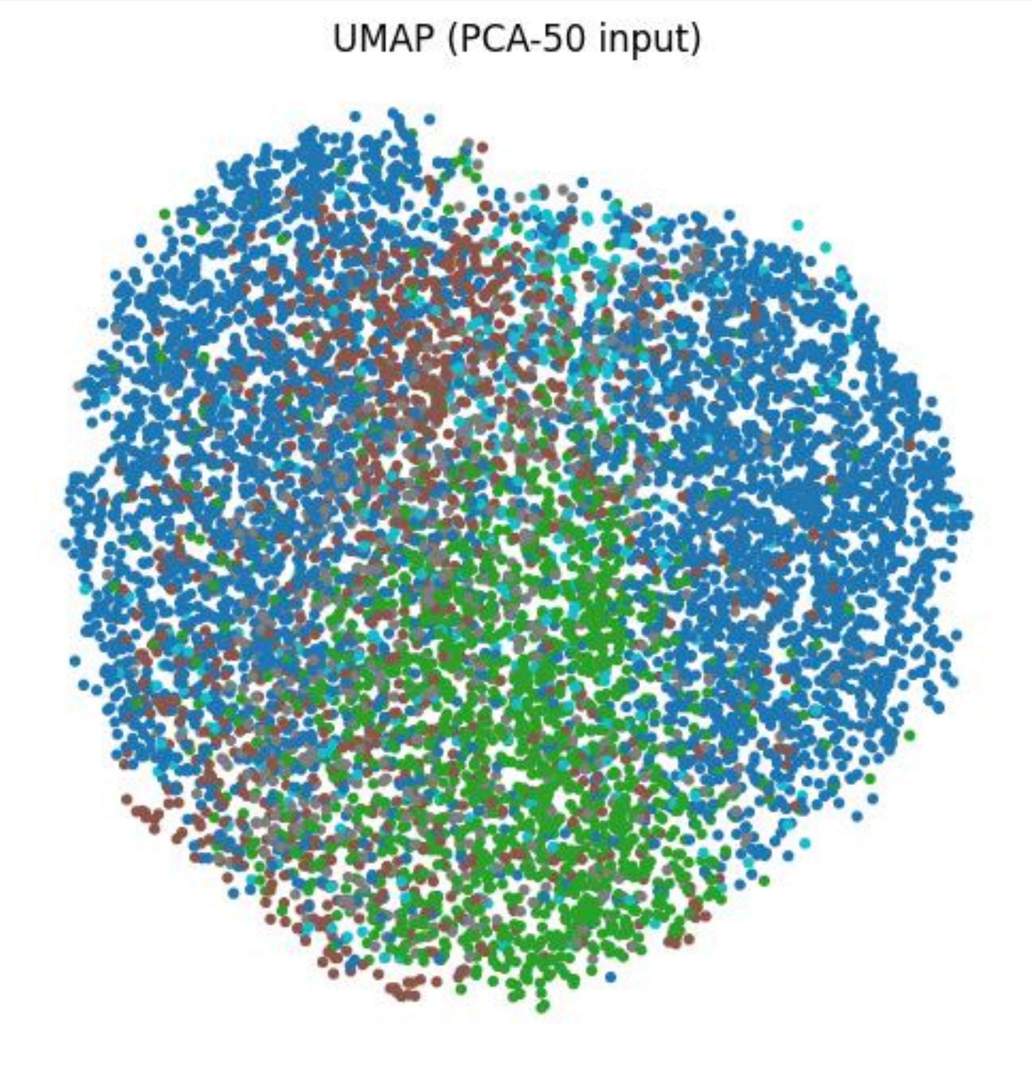
- First 50 components
- Explained variance (50 PCs): 47.6%
    - Makes sense considering features are mostly uncorrelated

Dataset Exploration: T-SNE



- First 50 components
- Preserves local neighborhoods

Dataset Exploration: UMAP

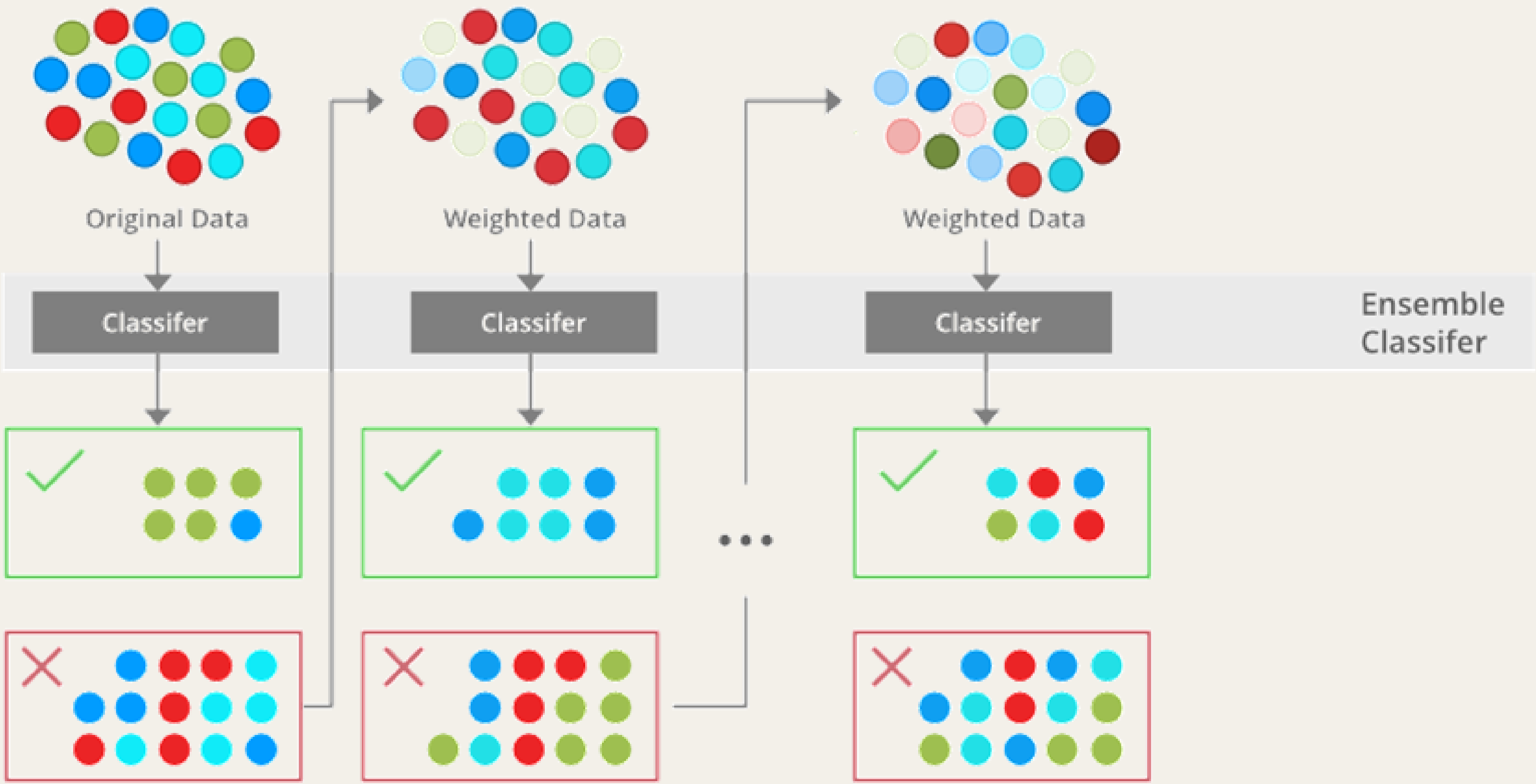


- First 50 components
- Preserves local neighborhoods and some global structure

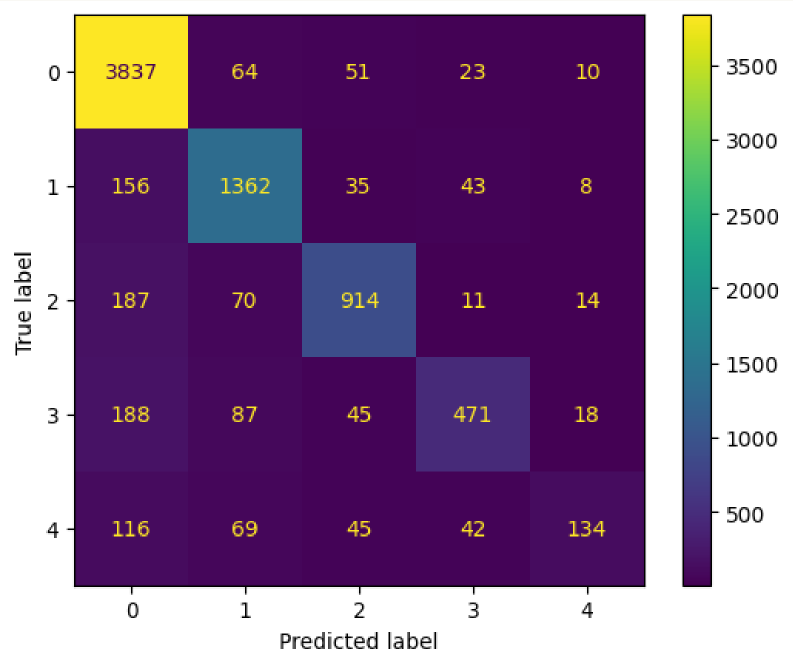
- Takeaway: class differences are subtly and may rely on non-obvious patterns

Baseline Model: XGBoost

- XGBoost is often best performing model on tabular data
- Use XGBoost as baseline and see if we can beat it with MLP



Baseline Model: XGBoost



=== Cross-validated performance ===

	precision	recall	f1-score
0	0.856	0.963	0.906
1	0.824	0.849	0.837
2	0.839	0.764	0.800
3	0.798	0.582	0.673
4	0.728	0.330	0.454
accuracy			0.840
macro avg	0.809	0.698	0.734
weighted avg	0.835	0.840	0.830

Methods:

- Use 5-fold cross validation with shuffling
- For each training fold, calculate inverse weights based on the frequency of the class
- Train XGboost model with 660 estimators (found from hyperparameter grid search)

Results:

- Accuracy: 0.84
- Performed much worse on smaller classes

MLP Model 1: standard MLP

Methods:

1) Standardize features**
2) Architecture and hyperparameters
   - 2 hidden layers (512, 256)
   - Dropout (0.35)
   - Weight Decay (1e-4): basically L2 regularization
   - AdamW optimizer (lr=1e-3)
   - Batch Normalization
   - Cross-Entropy Loss
3) Inverse class weights
4) Trained for 40 epochs
5) 5-fold cross validation

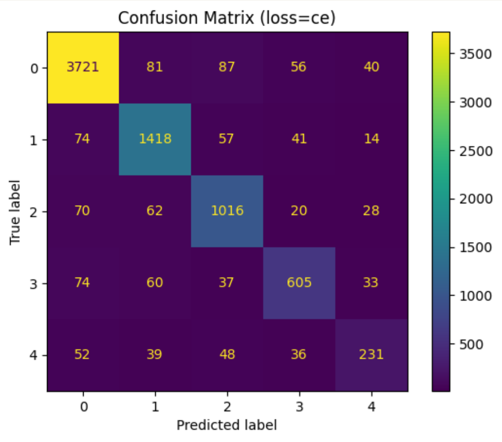
Regularization to prevent overfitting

** forgot this at first but makes a big different, accuracy shot up 3-4% after standardizing

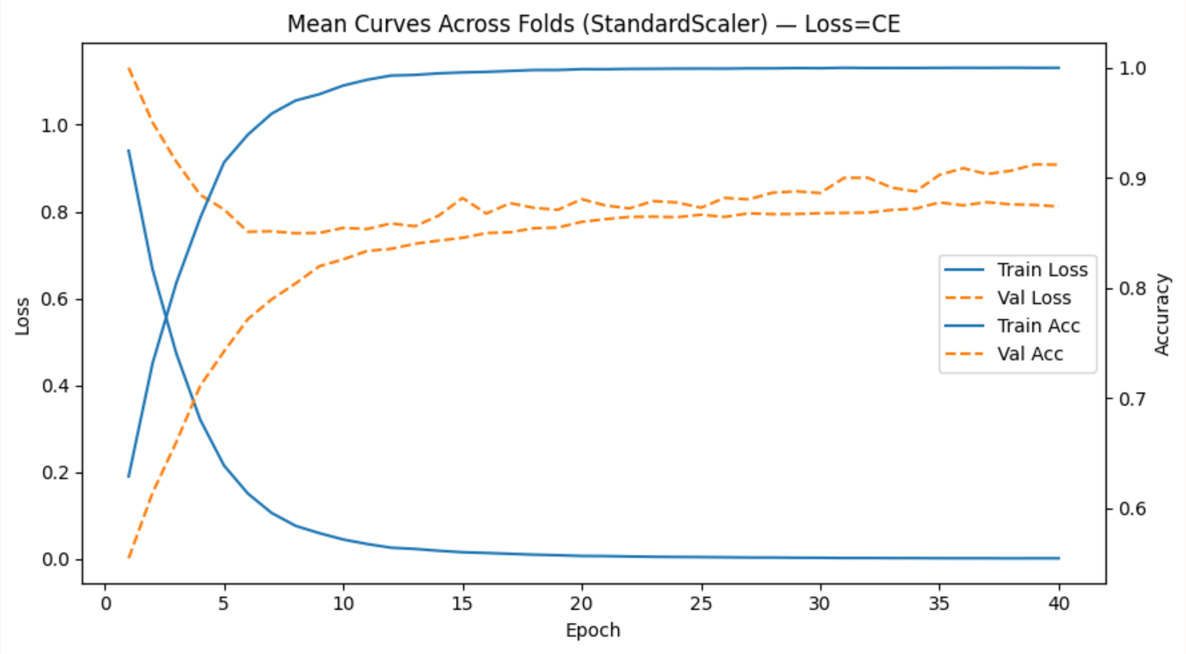
MLP Model 1: standard MLP

=== Cross-validated performance (StandardScaler, loss=ce) ===

	precision	recall	f1-score	support
0	0.932	0.934	0.933	3985
1	0.854	0.884	0.869	1604
2	0.816	0.849	0.832	1196
3	0.798	0.748	0.772	809
4	0.668	0.569	0.614	406
accuracy			0.874	8000
macro avg	0.814	0.797	0.804	8000
weighted avg	0.872	0.874	0.873	8000



- Overall accuracy: 0.874
  - Better than XGboost!
- Problem 1: does better on smallest dataset but still not great

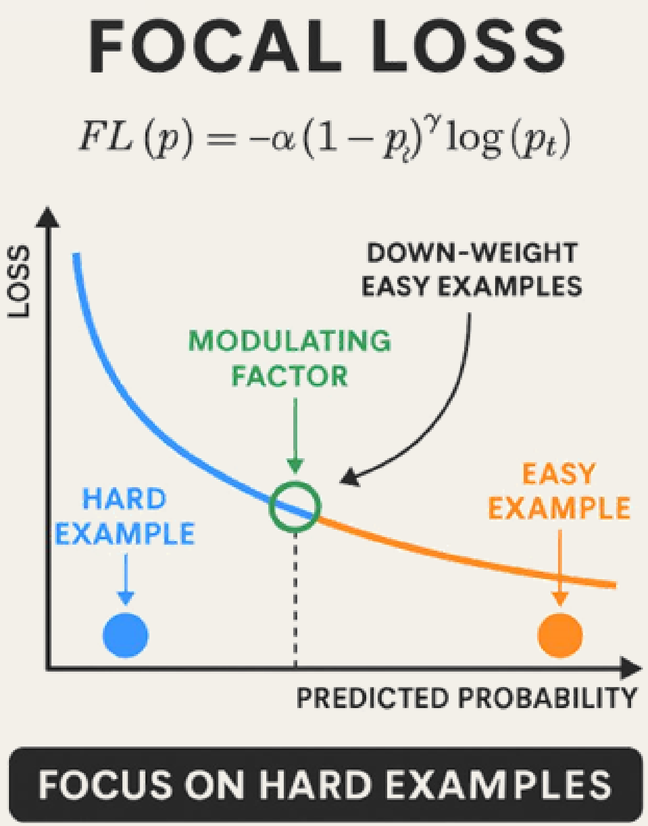


- Problem 2: Clear signs of overfitting as accuracy holds constant but loss increases over training epochs

MLP Model 2: standard MLP with focal loss

Methods:

- Same exact model as model 1 but using focal loss

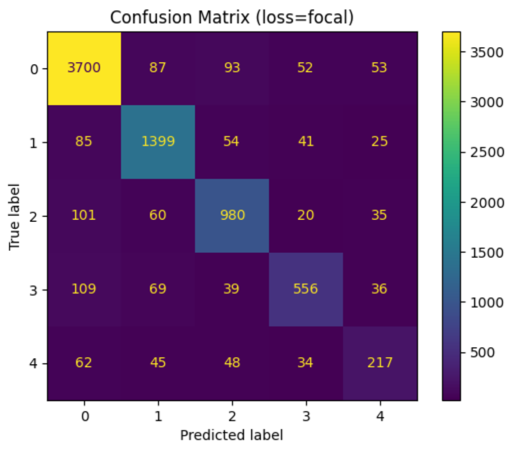


- Focuses on "hard" example (i.e smaller classes) and less on "easy" examples
- Modified cross-entropy loss
- Gamma is focusing parameter
  - When gamma = 0, same as CE
  - When gamma higher, higher probability or confidence outputs are weighted less to loss function

MLP Model 2: standard MLP with focal loss

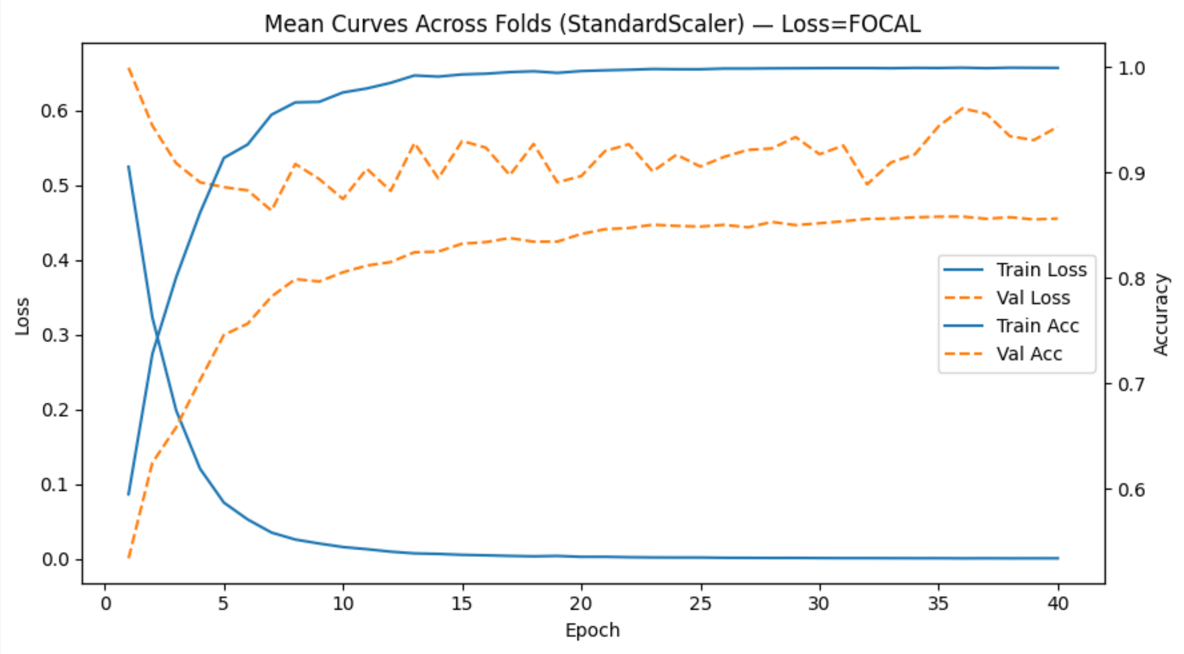
=== Cross-validated performance (StandardScaler, loss=focal) ===

	precision	recall	f1-score	support
0	0.912	0.928	0.920	3985
1	0.843	0.872	0.857	1604
2	0.807	0.819	0.813	1196
3	0.791	0.687	0.735	809
4	0.593	0.534	0.562	406
accuracy			0.857	8000
macro avg	0.789	0.768	0.778	8000
weighted avg	0.854	0.857	0.855	8000



- Overall accuracy: 0.857
- Performance got worse

Hypothesis: focal loss is notoriously sensitive to outliers



Problem: still overfitting

MLP Model 4: wide and deep NN

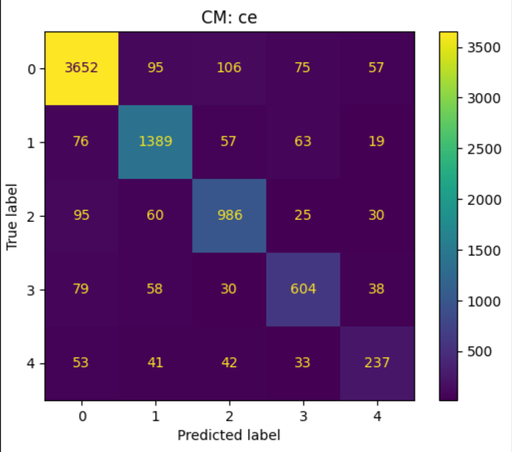
Methods:

1) Standardize features
2) Architecture and hyperparameters
   - Deep: 3 hidden layers funneled (256, 128, 64) (generalize)
   - Wide: input features fed directly to final logits (memorize)
   - Dropout (0.35) + Weight Decay (1e-4)
   - AdamW optimizer (lr=1e-3)
   - Switch to LeakyReLU (alpha=0.1)
   - Batch normalization
   - Cross-Entropy Loss
3) Inverse class weights
4) Trained for 40 epochs
5) 5-fold cross validation

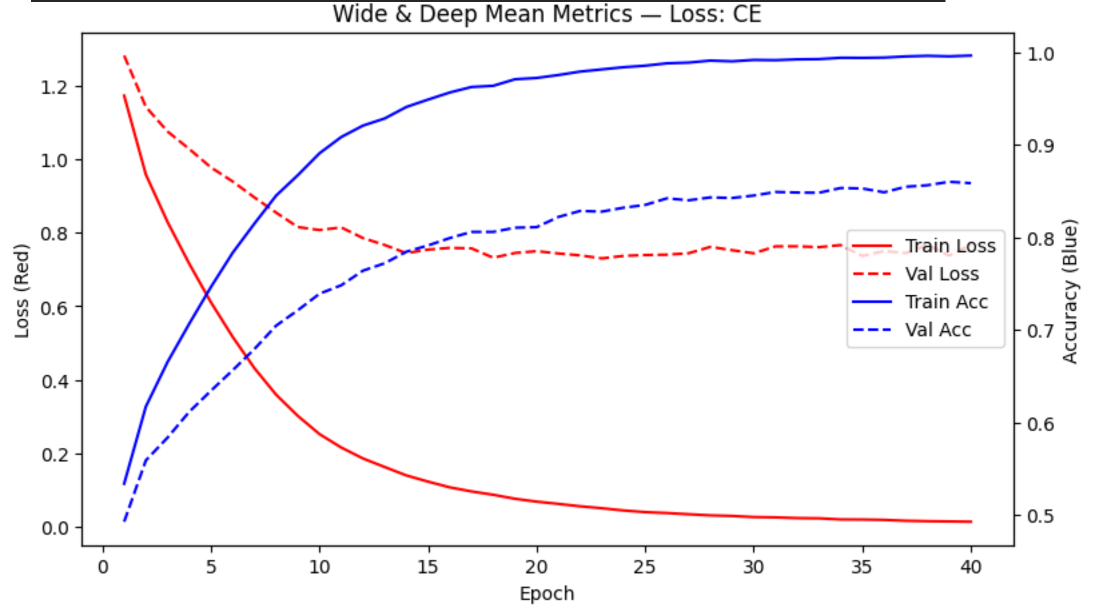
MLP Model 4: wide and deep NN

Report (ce):

	precision	recall	f1-score	support
0	0.923	0.916	0.920	3985
1	0.845	0.866	0.856	1604
2	0.808	0.824	0.816	1196
3	0.755	0.747	0.751	809
4	0.622	0.584	0.602	406
accuracy			0.859	8000
macro avg	0.791	0.787	0.789	8000
weighted avg	0.858	0.859	0.858	8000



- Overall accuracy: 0.859
- Performance got worse Hypothesis: focal loss is notoriously sensitive to outliers



- Seemed to help overfitting, more stable training loss and accuracy curves

MLP Model 5: two-stage classifer

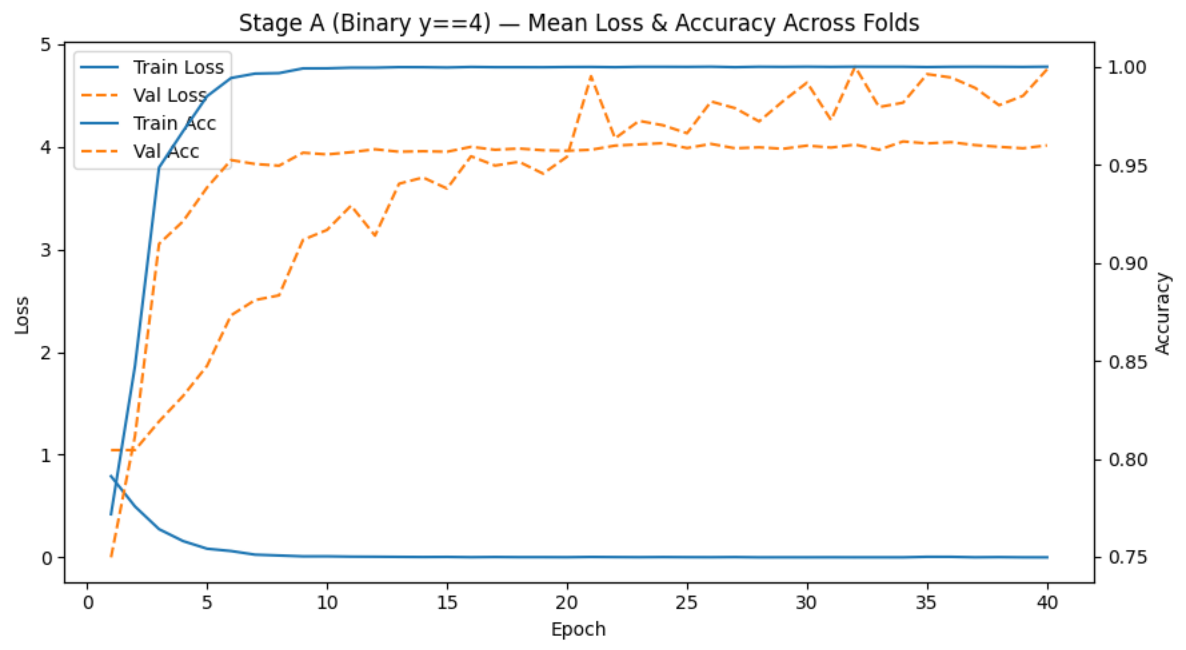
Methods:

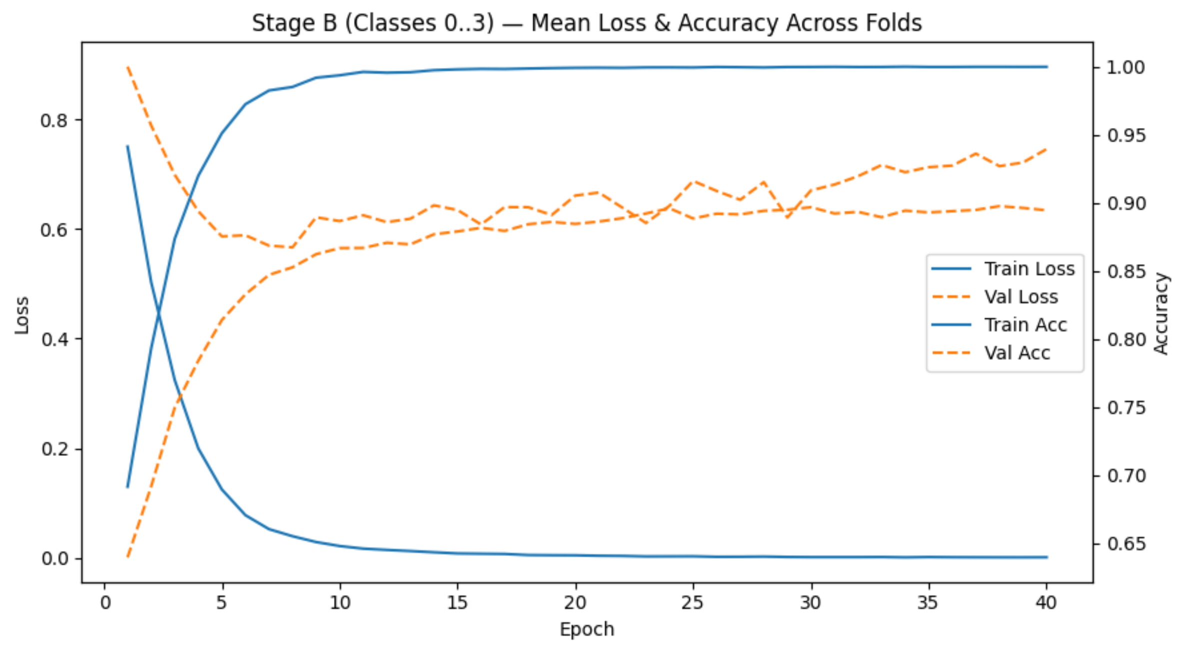
1) Standardize features
2) Binary detector: just distinguish class 4 from rest of classes to help class imbalance
3) Multiclass detector: distinguish classes 0-3
4) Same architecture as model 1

4) Train both for 40 epochs

5) 5-fold cross-validation

MLP Model 5: two-stage classifer

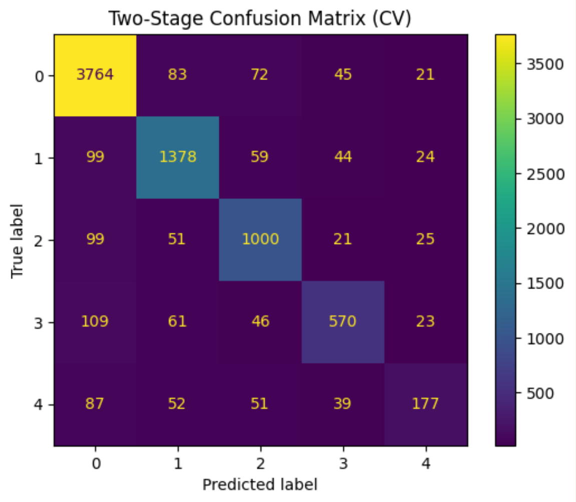





```
=== Cross-validated performance (Two-Stage, StandardScaler) ===
              precision    recall  f1-score   support

           0      0.905     0.945     0.924      3985
           1      0.848     0.859     0.854      1604
           2      0.814     0.836     0.825      1196
           3      0.793     0.705     0.746       809
           4      0.656     0.436     0.524       406

    accuracy                          0.861      8000
   macro avg      0.803     0.756     0.775      8000
weighted avg      0.856     0.861     0.857      8000
```

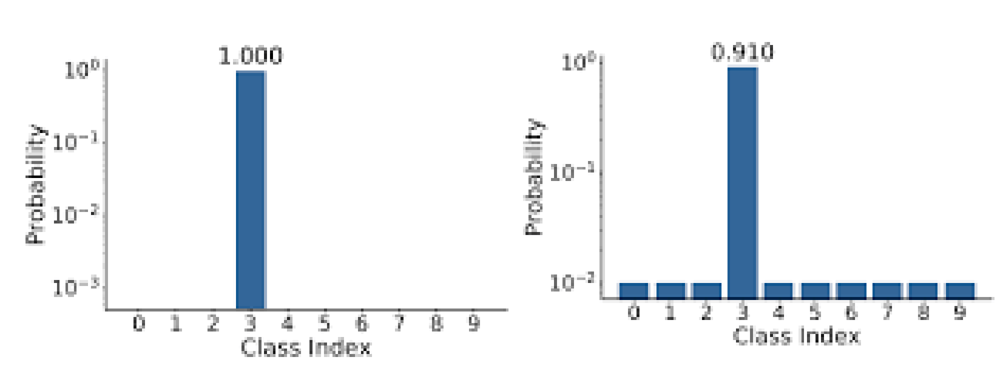


TLDR: accuracy 0.861, didn't help much

Hypothesis: errors from first classifier propagate; loss of shared feature representation and interclass contrast

MLP Model 6: standard MLP + more regularization

Model 1 + early stopping and label smoothing + more weight decay

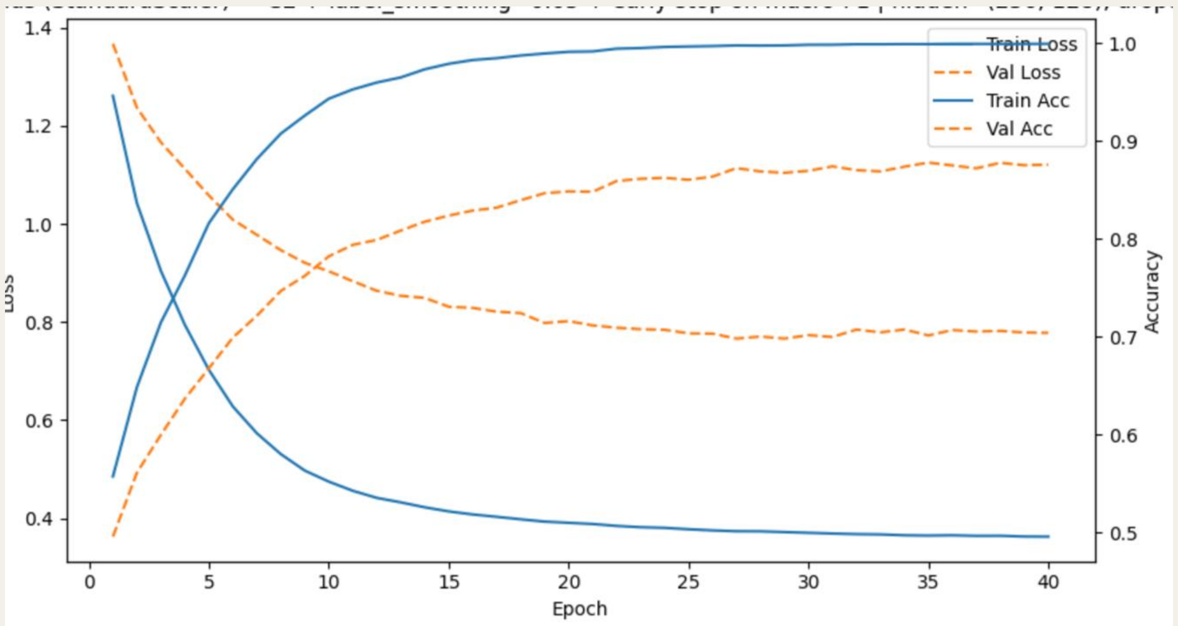


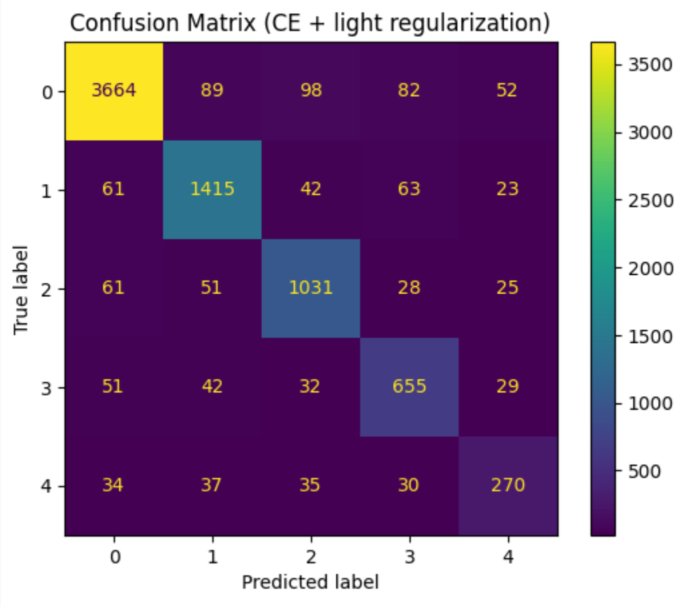
(a) Hard Label (b) LS

```
=== Cross-validated performance (StandardScaler, CE + light regularization) ===
              precision    recall  f1-score   support

           0      0.947     0.919     0.933      3985
           1      0.866     0.882     0.874      1604
           2      0.833     0.862     0.847      1196
           3      0.763     0.810     0.786       809
           4      0.677     0.665     0.671       406

    accuracy                          0.879      8000
   macro avg      0.817     0.828     0.822      8000
weighted avg      0.881     0.879     0.880      8000
```

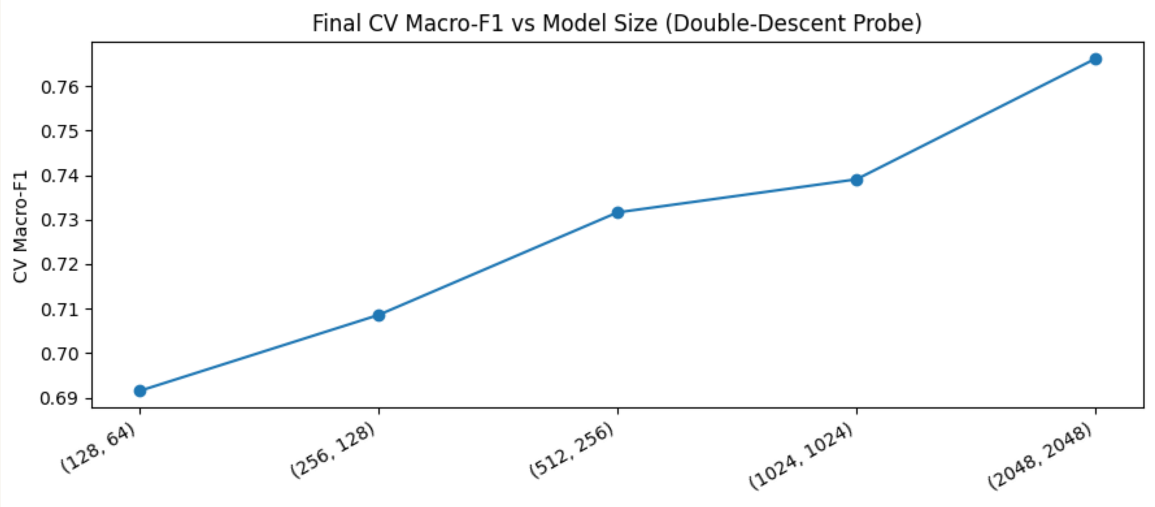


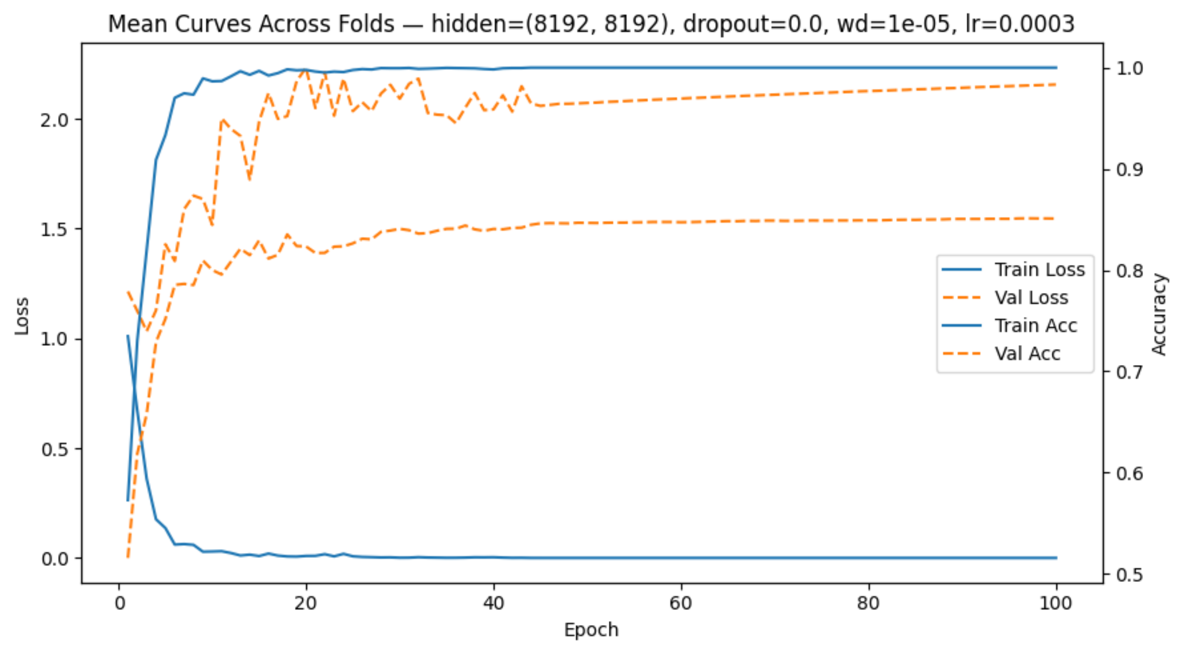


TLDR: accuracy = 0.879
Helped smooth loss and accuracy curves showing best performance

MLP Model 7: double descent

Tried heavily overparameterizing the model to get double descent, trained up to (8192, 8192), yielded continuously increasing accuracies up to 0.86 but couldn't train anything larger with laptop compute





Feature importance: from model 1 MLP

=== Top 20 features (CV-averaged permutation importance) ===

rank	feature_idx	feature_name	importance_mean	importance_std
1	133	f132	0.026020	0.003864
2	188	f187	0.025005	0.005350
3	158	f157	0.023084	0.004589
4	156	f155	0.022115	0.005197
5	127	f126	0.021987	0.003795
6	28	f027	0.021693	0.005374
7	63	f062	0.019266	0.004356
8	72	f071	0.019200	0.003858
9	66	f065	0.018938	0.004160
10	38	f037	0.018607	0.004299
11	184	f183	0.017422	0.002916
12	180	f179	0.017050	0.003507
13	137	f136	0.015701	0.005140
14	109	f108	0.015544	0.003615
15	39	f038	0.015169	0.003834
16	129	f128	0.014918	0.003921
17	43	f042	0.014689	0.004692
18	105	f104	0.014678	0.003371
19	12	f011	0.014569	0.004115
20	11	f010	0.014280	0.004333

If we randomly shuffle a features values, how much does accuracy drop?

Takeaways

- Experimented with a lot of MLP architectures and parameters, but couldn't pass the high 80s for accuracy
- Likely class imbalance holding the model back and this is near the limit of what MLPs can do
- More advanced techniques can often overcomplicate things and lead to worse results
- Hard to beat simple MLP model with regularization

Final Scores

- Trained best light regularization Model 6 on full train.csv and predicted test.csv for 40 epochs

```
id,label
0,4
1,0
2,1
3,0
4,1
5,1
6,0
7,3
8,4
9,3
10,4
```