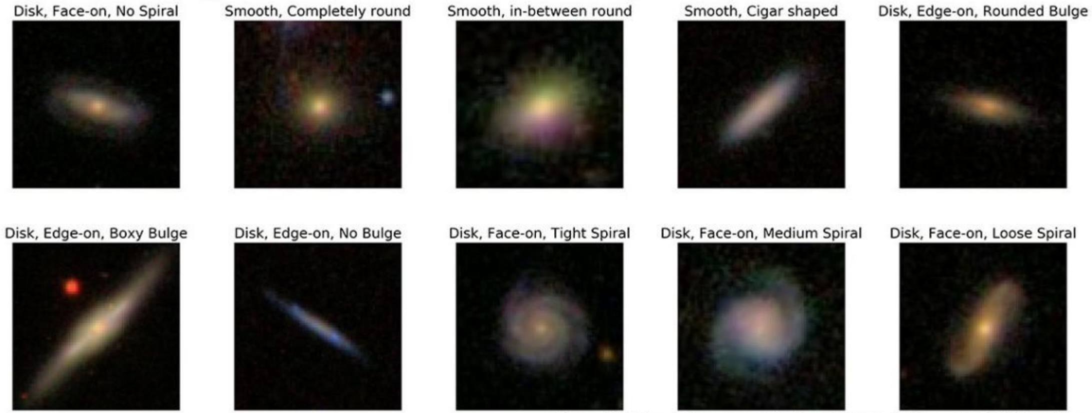
Galaxy Classification Challenge

Group #1

Timo Schellhaas
Jaroslav Storek
Leonid Lunin
Lisa Lou Krümmel
Mathis Nolte

Dataset Exploration and the Class 5 Problem

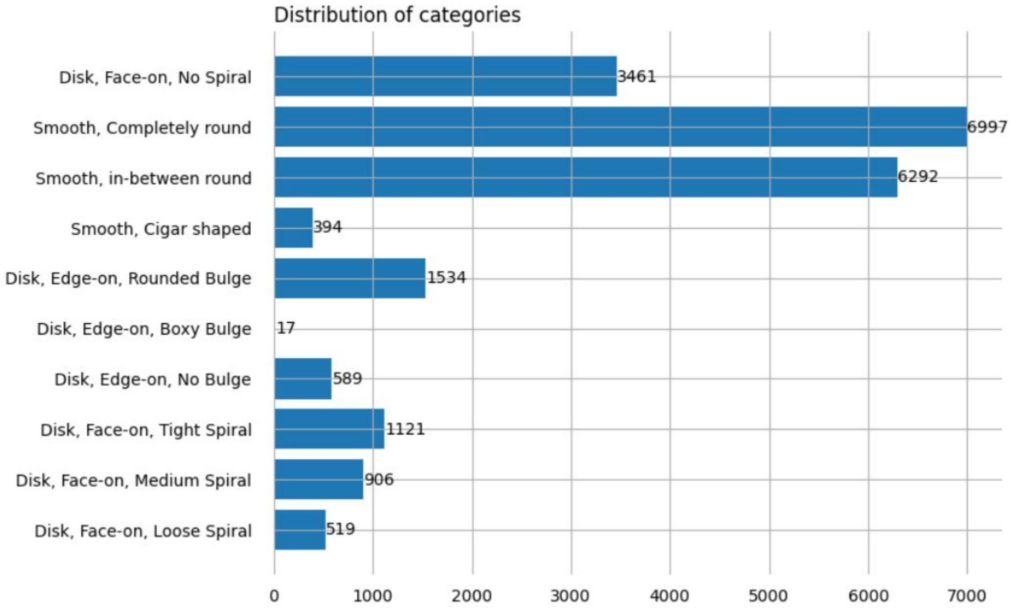
Example images of each class from Galaxy10 dataset


Disk, Face-on, No Spiral
Smooth, Completely round
Smooth, in-between round
Smooth, Cigar shaped
Disk, Edge-on, Rounded Bulge
Disk, Edge-on, Boxy Bulge
Disk, Edge-on, No Bulge
Disk, Face-on, Tight Spiral
Disk, Face-on, Medium Spiral
Disk, Face-on, Loose Spiral

Galaxy10 Dataset: Henry Leung/Jo Bovy 2018, Data Source: SDSS/Galaxy Zoo

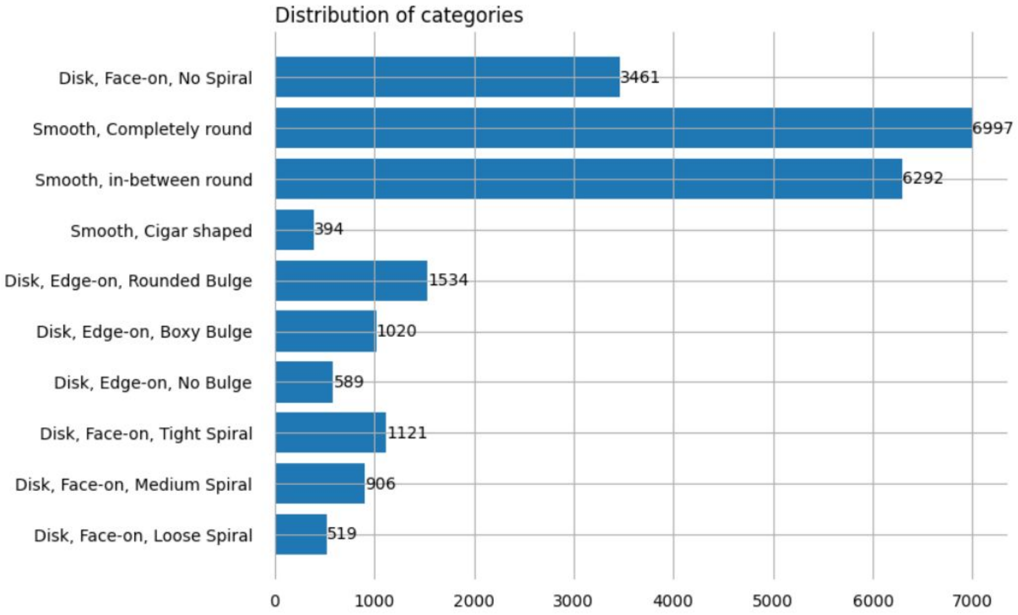
Class 5 Problem

- Class 5 is dramatically underrepresented



Our solutions

- Replicating class 5 images to increase its impact
- Data augmentation will prevent overfitting (random rotation and flip of images but no stretching)
- Increase batch size to have a higher possibility to actually see a Class 5



Neural Network Architecture

Which network have we used and how do we measure the performance?

Average accuracy among the classes -to rate the overall performance

Confusion matrix - to rate the performance of the individual classes

What we have tried:

- LeNet
- AlexNet
- Branching

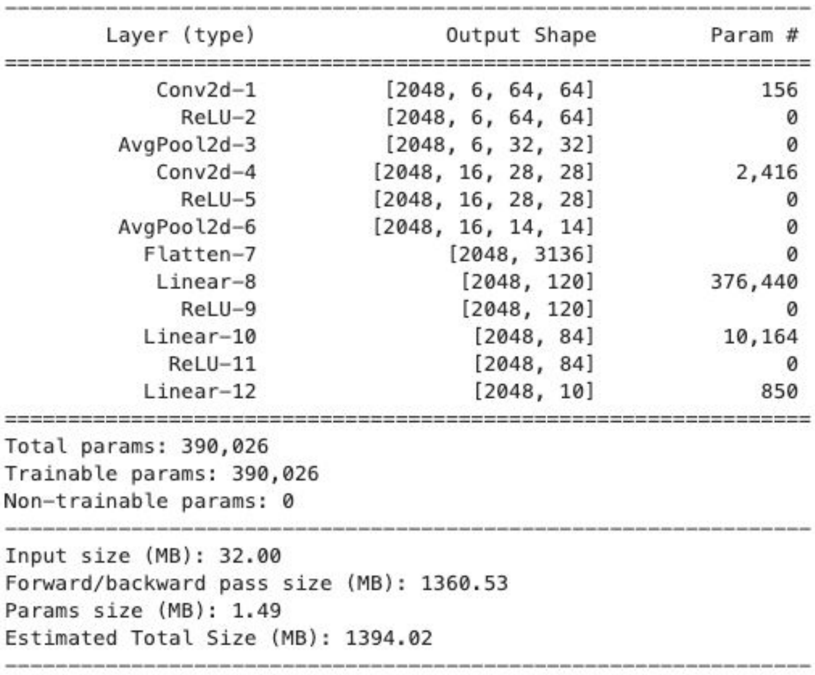
Other possible approaches:

- ResNet [1]
- Assembles

[1] Ba Alawi, A. E., & Al-Roainy, A. A. (2021). Deep Residual Networks Model for Star-Galaxy Classification. 2021 https://doi.org/10.1109/icoten52080.2021.9493433

LeNet and AlexNet inspired architecture

- Dropout layers to reduce overfitting
- Optimizer: Adam
- Using L2 weight_decay option
- Kernel sizes matched to the feature sizes in the images
- Convolution kernel sizes: 5 and 7

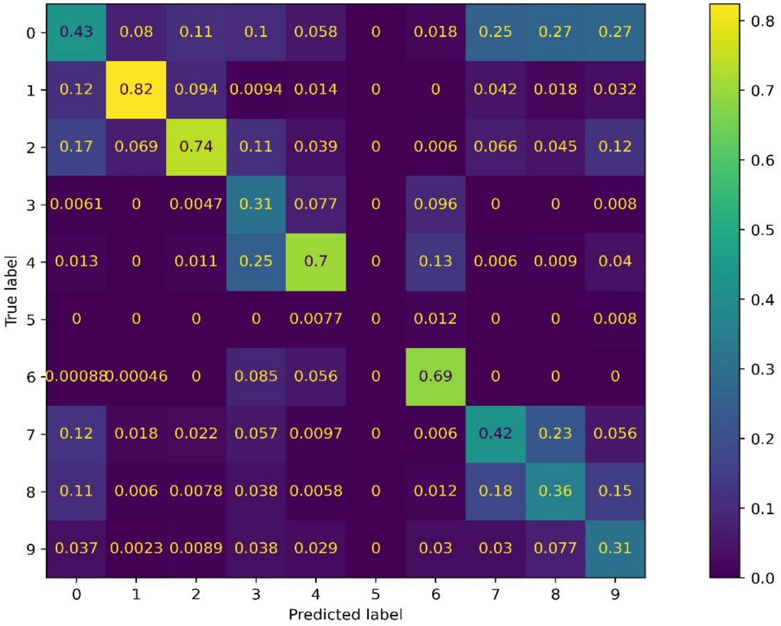


```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [2048, 6, 64, 64]             156
              ReLU-2           [2048, 6, 64, 64]               0
         AvgPool2d-3           [2048, 6, 32, 32]               0
            Conv2d-4          [2048, 16, 28, 28]           2,416
              ReLU-5          [2048, 16, 28, 28]               0
         AvgPool2d-6          [2048, 16, 14, 14]               0
           Flatten-7                [2048, 3136]               0
            Linear-8                 [2048, 120]         376,440
              ReLU-9                 [2048, 120]               0
           Linear-10                  [2048, 84]          10,164
             ReLU-11                  [2048, 84]               0
           Linear-12                  [2048, 10]             850
================================================================
Total params: 390,026
Trainable params: 390,026
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 32.00
Forward/backward pass size (MB): 1360.53
Params size (MB): 1.49
Estimated Total Size (MB): 1394.02
----------------------------------------------------------------
```

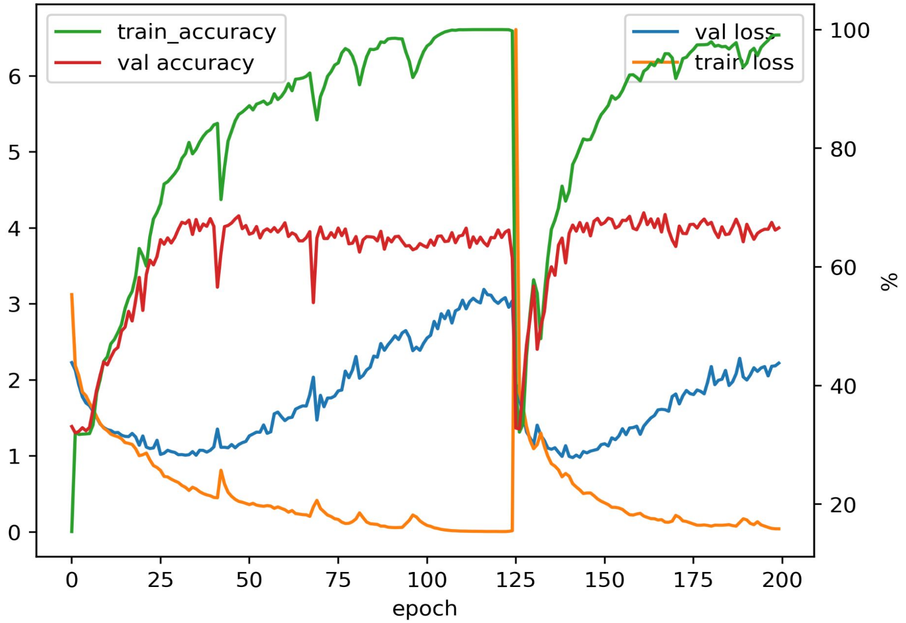
Network Training and Evaluation

No data augmentation (200 epochs, 2048 batch size)

- Overfitting!

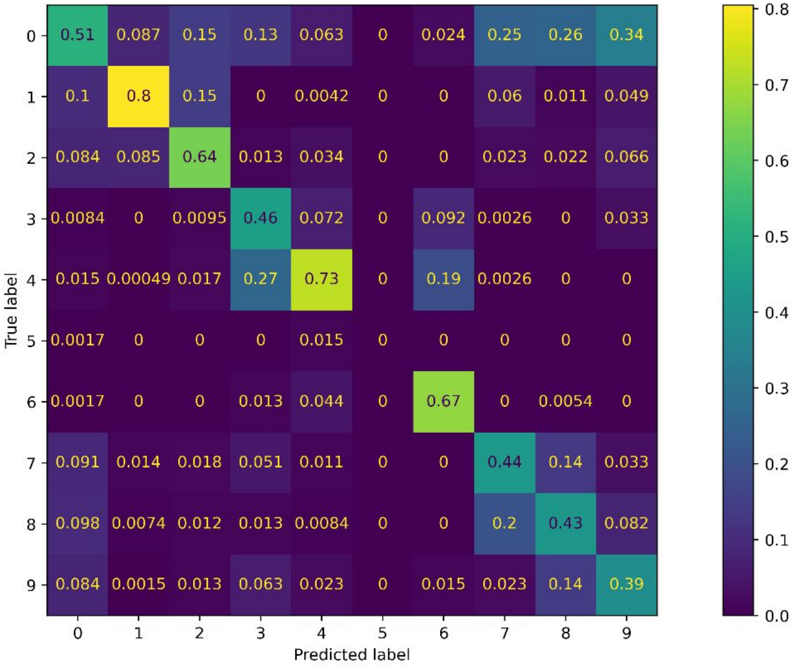


	0	1	2	3	4	5	6	7	8	9
0	0.43	0.08	0.11	0.1	0.058	0	0.018	0.25	0.27	0.27
1	0.12	0.82	0.094	0.0094	0.014	0	0	0.042	0.018	0.032
2	0.17	0.069	0.74	0.11	0.039	0	0.006	0.066	0.045	0.12
3	0.0061	0	0.0047	0.31	0.077	0	0.096	0	0	0.008
4	0.013	0	0.011	0.25	0.7	0	0.13	0.006	0.009	0.04
5	0	0	0	0	0.0077	0	0.012	0	0	0.008
6	0.00088	0.00046	0	0.085	0.056	0	0.69	0	0	0
7	0.12	0.018	0.022	0.057	0.0097	0	0.006	0.42	0.23	0.056
8	0.11	0.006	0.0078	0.038	0.0058	0	0.012	0.18	0.36	0.15
9	0.037	0.0023	0.0089	0.038	0.029	0	0.03	0.03	0.077	0.31

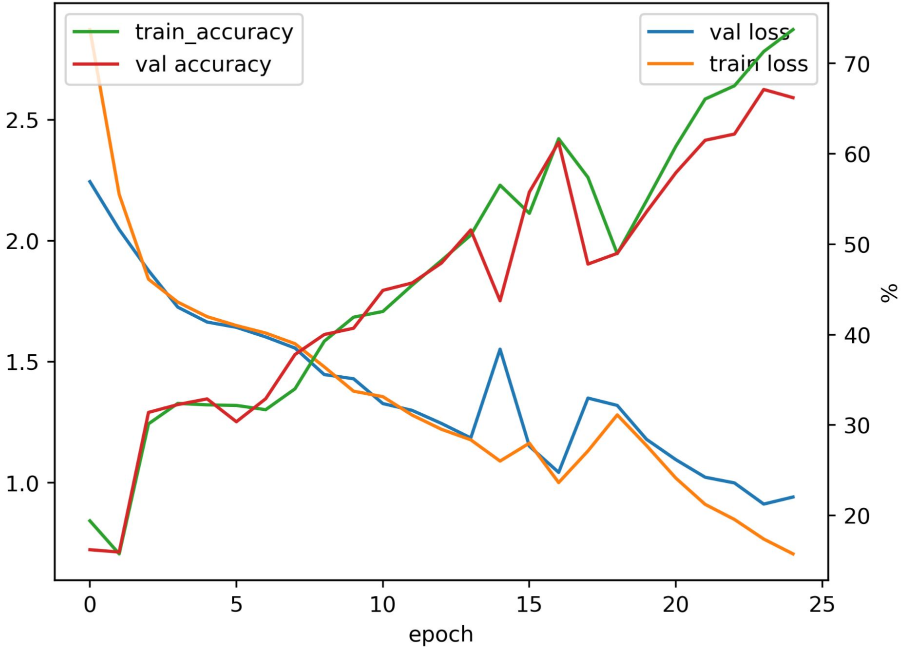


No data augmentation (25 epochs, 2048 batch size)

- No overfitting. 66% accuracy on the test dataset with 75% accuracy on train dataset

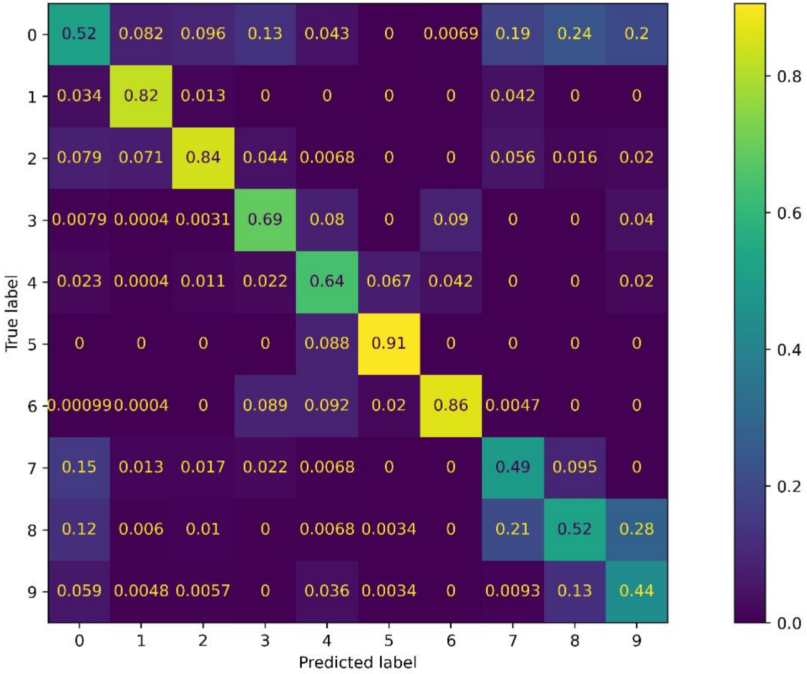


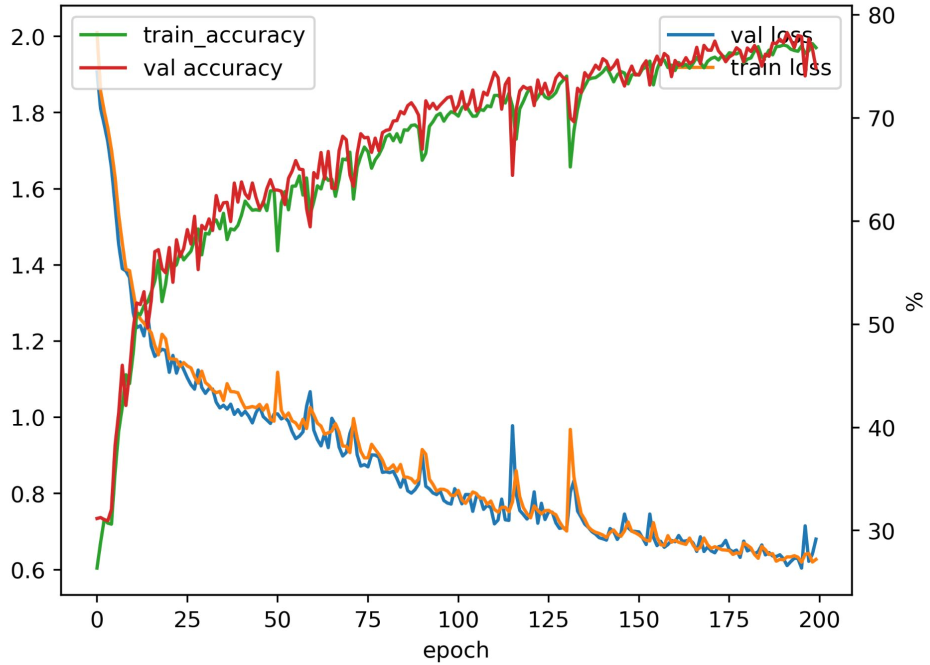
	0	1	2	3	4	5	6	7	8	9
0	0.51	0.087	0.15	0.13	0.063	0	0.024	0.25	0.26	0.34
1	0.1	0.8	0.15	0	0.0042	0	0	0.06	0.011	0.049
2	0.084	0.085	0.64	0.013	0.034	0	0	0.023	0.022	0.066
3	0.0084	0	0.0095	0.46	0.072	0	0.092	0.0026	0	0.033
4	0.015	0.00049	0.017	0.27	0.73	0	0.19	0.0026	0	0
5	0.0017	0	0	0	0.015	0	0	0	0	0
6	0.0017	0	0	0.013	0.044	0	0.67	0	0.0054	0
7	0.091	0.014	0.018	0.051	0.011	0	0	0.44	0.14	0.033
8	0.098	0.0074	0.012	0.013	0.0084	0	0	0.2	0.43	0.082
9	0.084	0.0015	0.013	0.063	0.023	0	0.015	0.023	0.14	0.39



With data augmentation (200 epochs, 2048 batch size)

- Best result: 78% accuracy on the test dataset with 78% accuracy on train dataset



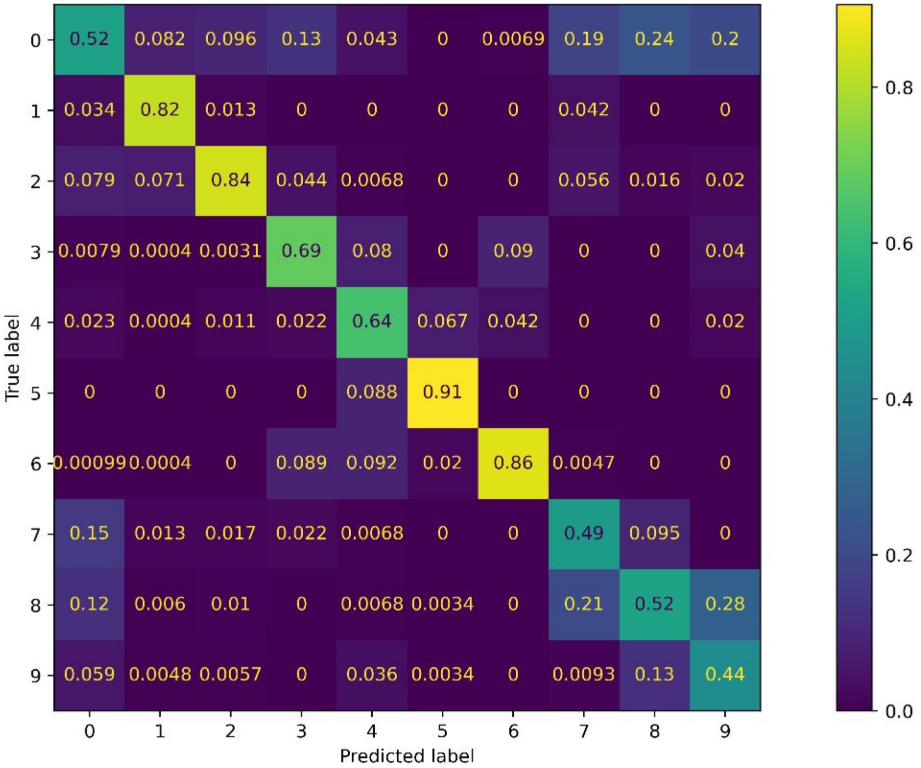


Performance on the individual classes

- Excellent performance on classes with high statistics (1 & 2)
- Outstanding performance on class 5
- Worst performance on poorly represented classes with high similarity

Potential for improvement:

- Replicating the last three classes



	0	1	2	3	4	5	6	7	8	9
0	0.52	0.082	0.096	0.13	0.043	0	0.0069	0.19	0.24	0.2
1	0.034	0.82	0.013	0	0	0	0	0.042	0	0
2	0.079	0.071	0.84	0.044	0.0068	0	0	0.056	0.016	0.02
3	0.0079	0.0004	0.0031	0.69	0.08	0	0.09	0	0	0.04
4	0.023	0.0004	0.011	0.022	0.64	0.067	0.042	0	0	0.02
5	0	0	0	0	0.088	0.91	0	0	0	0
6	0.00099	0.0004	0	0.089	0.092	0.02	0.86	0.0047	0	0
7	0.15	0.013	0.017	0.022	0.0068	0	0	0.49	0.095	0
8	0.12	0.006	0.01	0	0.0068	0.0034	0	0.21	0.52	0.28
9	0.059	0.0048	0.0057	0	0.036	0.0034	0	0.0093	0.13	0.44

Thank you for your attention!

Jupyter notebook available at:

github.com/lrlunin/erum-datahub-challange-2023/tree/main