This is for the competition in kaggle:
https://www.kaggle.com/t/067f885fce1c49019c3a92f6ccea148c

Start Here

Upload the data files and split

Noah Choate Lab 8

```
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical
import numpy as np

from google.colab import drive
drive.mount('/content/drive')

X = np.load("/content/drive/My Drive/ColabNotebooks/DSC4310/Data/x_train.npy")
y = np.load("/content/drive/My Drive/ColabNotebooks/DSC4310/Data/y_train.npy")
x_test = np.load("/content/drive/My Drive/ColabNotebooks/DSC4310/Data/x_test.npy")

# Loaded in for Jupyter Notebook (Not a fan of Google Colab)
#X = np.load(r"C:\Users\noahc\DSC4310-MachineLearning\Data\x_train.npy")
#y = np.load(r"C:\Users\noahc\DSC4310-MachineLearning\Data\y_train.npy")
#x_test = np.load(r"C:\Users\noahc\DSC4310-MachineLearning\Data\x_test.npy")

x_train, x_val, y_train, y_val = train_test_split(X, y,
                                                  test_size=0.2,
                                                  random_state=42)
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mour
```

```
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_val = x_val.astype('float32')
x_train = x_train.reshape((len(x_train), np.prod(x_train.shape[1:])))
x_test = x_test.reshape((len(x_test), np.prod(x_test.shape[1:])))
x_val = x_val.reshape((len(x_val), np.prod(x_val.shape[1:])))
y_train = to_categorical(y_train, len(set(y_train)))
y_val = to_categorical(y_val, len(set(y_val)))
print('x_train shape:', x_train.shape)
print('x_test shape:', x_test.shape)
print('x_val shape:', x_val.shape)
```

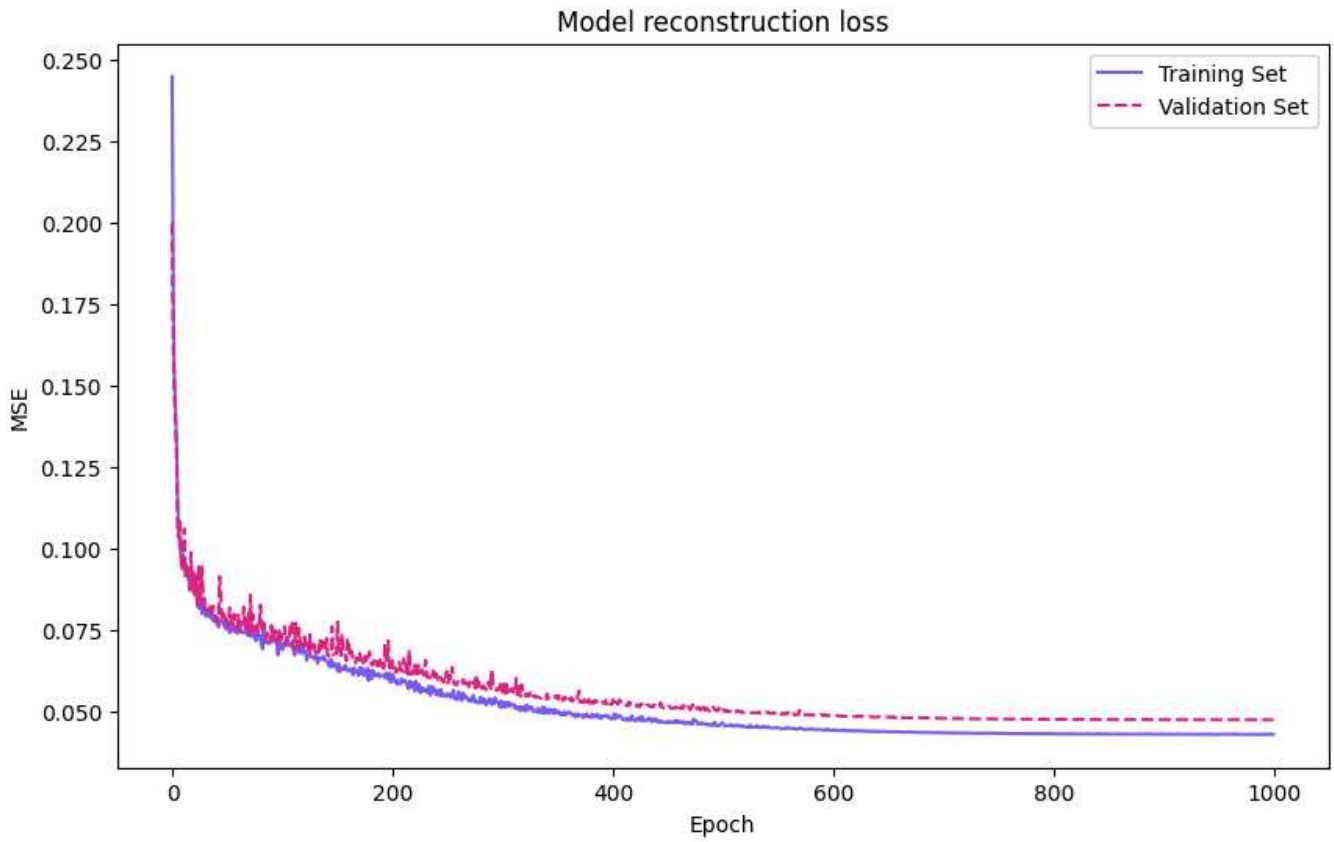
```
x_train shape: (980, 3009)
x_test shape: (1226, 3009)
x_val shape: (245, 3009)
```

```
from tensorflow.keras.layers import Input, Dense, Dropout
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import RMSprop

# dimensionality of input and latent encoded representations
inpt_dim = 59*51
ltnt_dim = 100

# --- define an autoencoder model here
inpt_vec = Input(shape=(inpt_dim,))
           # <-- add more layers here if you want
l0 = Dense(512, activation='relu')(inpt_vec)
l1 = Dropout(0.2)(inpt_vec)
encoder = Dense(ltnt_dim, activation='sigmoid') (l1)   # <-- do not change this

# model that takes input and encodes it into the latent space
latent_ncdr = Model(inpt_vec, encoder)   # <-- do not change this, you will need
                                    #     to access the layers of the encoder
           # <-- add more layers here if you want
l2 = Dense(512, activation='relu')(encoder)
l3 = Dropout(0.2)(encoder)
decoder = Dense(inpt_dim, activation='sigmoid') (l2)

# model that takes input, encodes it, and decodes it
autoencoder = Model(inpt_vec, decoder)

opt = RMSprop(learning_rate=0.01)

autoencoder.compile(loss='mean_absolute_error', optimizer=opt)
# --- end of model definitions
```

```
from tensorflow.keras.callbacks import ReduceLROnPlateau, EarlyStopping
```

```
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.9, patience=10,
                              min_delta=1e-4, mode='min')

stop_alg = EarlyStopping(monitor='val_loss', patience=100, restore_best_weights=True)

hist = autoencoder.fit(x_train, x_train, batch_size=100, epochs=1000,
                   verbose=0, callbacks=[stop_alg, reduce_lr], shuffle=True,
                   validation_data=(x_val, x_val))

autoencoder.save_weights("autoencoder.weights.h5")
```

```
 import matplotlib.pyplot as plt

fig = plt.figure(figsize=(10,6))
plt.plot(hist.history['loss'], color='#785ef0')
plt.plot(hist.history['val_loss'], '--', color='#dc267f')
# plt.yscale('log')
plt.title('Model reconstruction loss')
plt.ylabel('MSE')
plt.xlabel('Epoch')
plt.legend(['Training Set', 'Validation Set'], loc='upper right')
plt.show()
```

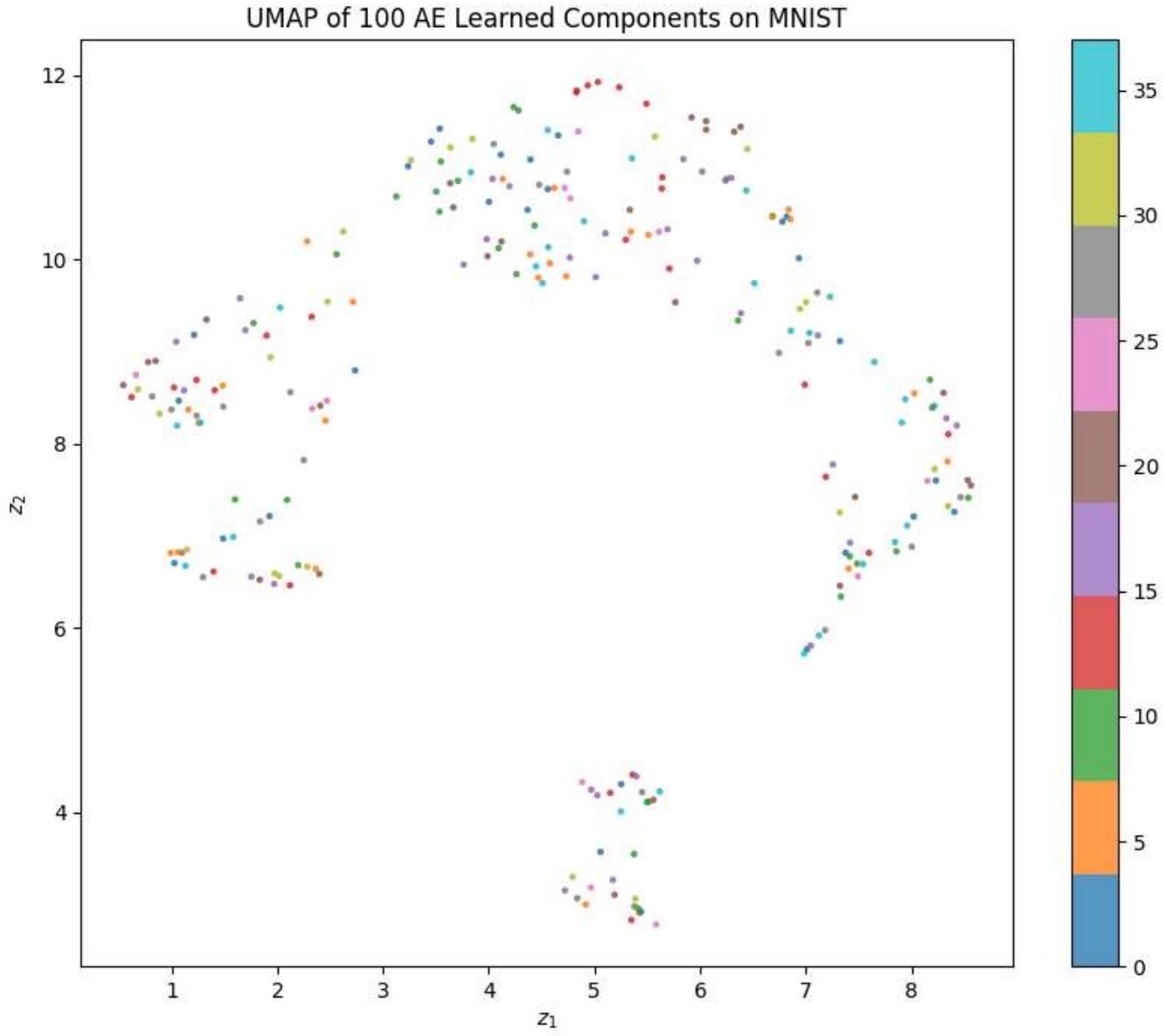



```
plt.figure(figsize=(10, 9.5))
for i, comp in enumerate(latent_ncdr.get_weights()[0].T):
    plt.subplot(10, 10, i + 1)
    plt.imshow(comp.reshape((59, 51)), cmap=plt.cm.gray_r,
               interpolation='nearest')
    plt.xticks(())
    plt.yticks(())
plt.subplots_adjust(0.08, 0.02, 0.92, 0.85, 0.08, 0.23)
```

```
import matplotlib.pyplot as plt
import umap

y_ = list(map(int, np.argmax(y_val, axis=1, out=None)))
X_ = latent_ncdr.predict(x_val)

print(X_.shape)
```

```
X_ = umap.UMAP().fit_transform(X_)
print(X_.shape)

plt.figure(figsize=(10,8))
plt.title('UMAP of 100 AE Learned Components on MNIST')
plt.scatter(X_[:,0], X_[:,1], s=5.0, c=y_, alpha=0.75, cmap='tab10')
plt.xlabel('$z_1$')
plt.ylabel('$z_2$')
plt.colorbar()
```

```
8/8 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
(245, 100)
/usr/local/lib/python3.11/dist-packages/sklearn/utils/deprecation.py:151: FutureWarning:
  warnings.warn(
(245, 2)
<matplotlib.colorbar.Colorbar at 0x7c9f6209da10>
```




```
plt.figure(figsize=(10, 2))
for i in range(10):
  plt.subplot(2, 10, i+1)
  plt.imshow(x_val[i].reshape((59, 51)), cmap='gray')
  plt.xticks(())
  plt.yticks(())
```

```
X_ = autoencoder.predict(x_val[:10])
for i in range(10):
  plt.subplot(2, 10, i+11)
  plt.imshow(X_[i].reshape((59, 51)), cmap='gray')
  plt.xticks(())
  plt.yticks(())

plt.subplots_adjust(0.08, 0.02, 0.92, 0.85, 0.08, 0.23)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 400ms/step
```

Fine-tune autoencoder for classification

```
from tensorflow.keras.layers import Input, Dense, Dropout
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.optimizers import RMSprop

# dimensionality of input and latent encoded representations
inpt_dim = 59*51
ltnt_dim = 100
n_classes = 38

# --- define a dense model here (deep or wide); do not change anything here ...
model = Sequential()                          # except for the two things below
for layer in latent_ncdr.layers:
    print(layer.name)
    model.add(layer)
model.add(Dropout(0.5))  # < -- the dropout rate is the only thing you are ...
model.add(Dense(n_classes, activation='sigmoid'))       # allowed to change here

model.summary()    # v -- you are also allowed to change the optimizer
model.compile(loss='binary_crossentropy', optimizer='rmsprop')
# --- end of model definition
```

```
input_layer_1
dropout_3
dense_6
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
dropout_3 (Dropout)	(None, 3009)	0
dense_6 (Dense)	(None, 100)	301,000
dropout_5 (Dropout)	(None, 100)	0
dense_9 (Dense)	(None, 38)	3,838

Total params: 304,838 (1.16 MB)
Trainable params: 304,838 (1.16 MB)
Non-trainable params: 0 (0.00 B)

```
from tensorflow.keras.callbacks import ReduceLROnPlateau, EarlyStopping

reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.9, patience=10,
                              min_delta=1e-4, mode='min')

stop_alg = EarlyStopping(monitor='val_loss', patience=100, restore_best_weights=True)

hist = model.fit(x_train, y_train, batch_size=100, epochs=1000,
                  verbose=0, callbacks=[stop_alg, reduce_lr], shuffle=True,
                  validation_data=(x_val, y_val))

model.save_weights("tunedautoencoder.weights.h5")


#=== this will prepare a submission using the fine-tuned autoencoder
y_test = model.predict(x_test)
with open('aesubmission.csv', 'w', newline='') as file:
  file.write('Id,Category\n')
  for k in range(y_test.shape[0]):
    line = str(k) + ',' + str(np.argmax(y_test[k]))
    print(line)
    file.write(line)
    file.write('\n')
#=== submit your file and indicate this was with an autoencoder
from google.colab import files
files.download(file_name)
```

```
39/39 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
0,37
1,12
2,12
3,12
4,25
5,4
6,18
7,7
8,7
9,16
10,3
11,36
12,16
13,2
14,12
15,7
16,37
17,28
18,13
19,37
20,13
21,26
22,34
23,1
24,17
25,11
26,9
27,16
28,31
29,21
30,1
31,17
32,35
33,0
34,20
35,22
36,36
37,37
38,24
39,27
40,6
41,6
42,25
43,29
44,33
45,19
46,10
47,25
48,7
49,12
50,6
51,34
52,0
53,35
54,18
```

```
55,32
56,37
57,12
58,3
59,20
60,36
61,9
62,10
63,12
64,30
65,7
66,19
67,37
68,16
69,21
70,18
71,33
72,28
73,9
74,9
75,28
76,33
77,26
78,13
79,2
80,23
81,12
82,28
83,13
84,9
85,30
86,14
87,11
88,37
89,14
90,24
91,1
92,7
93,25
94,29
95,20
96,35
97,2
98,24
99,16
100,3
101,11
102,12
103,13
104,18
105,17
106,5
107,15
108,27
109,26
110,22
111,4
```

```
111,4
112,36
113,24
114,21
115,19
116,33
117,17
118,31
119,19
120,19
121,36
122,5
123,33
124,22
125,9
126,33
127,18
128,28
129,22
130,36
131,12
132,2
133,22
134,7
135,26
136,23
137,24
138,12
139,35
140,6
141,27
142,6
143,32
144,12
145,14
146,19
147,18
148,6
149,24
150,3
151,29
152,22
153,13
154,20
155,1
156,9
157,24
158,34
159,33
160,28
161,27
162,28
163,36
164,19
165,15
166,7
167,8
```

```
168,1
169,7
170,1
171,9
172,2
173,27
174,29
175,23
176,3
177,32
178,6
179,2
180,23
181,28
182,20
183,27
184,27
185,13
186,7
187,7
188,8
189,19
190,14
191,35
192,10
193,19
194,21
195,14
196,24
197,30
198,6
199,6
200,29
201,25
202,35
203,5
204,24
205,22
206,21
207,20
208,25
209,3
210,24
211,3
212,31
213,24
214,14
215,22
216,15
217,12
218,34
219,5
220,15
221,1
222,17
223,15
224,37
```

```
224,37
225,22
226,13
227,37
228,15
229,16
230,19
231,21
232,3
233,4
234,37
235,36
236,0
237,9
238,6
239,31
240,16
241,26
242,18
243,24
244,19
245,20
246,6
247,26
248,32
249,28
250,4
251,0
252,9
253,24
254,30
255,3
256,3
257,25
258,32
259,10
260,34
261,22
262,5
263,12
264,1
265,10
266,34
267,34
268,31
269,23
270,37
271,23
272,16
273,18
274,27
275,8
276,6
277,8
278,16
279,4
280,25
```

```
281,32
282,28
283,17
284,30
285,14
286,35
287,19
288,34
289,25
290,28
291,25
292,37
293,29
294,10
295,29
296,2
297,17
298,7
299,6
300,5
301,27
302,15
303,34
304,8
305,34
306,2
307,2
308,21
309,34
310,18
311,35
312,6
313,23
314,0
315,1
316,13
317,20
318,0
319,14
320,35
321,21
322,21
323,1
324,31
325,32
326,29
327,11
328,10
329,28
330,36
331,16
332,23
333,9
334,19
335,23
336,36
337,32
```

```
338,30
339,25
340,33
341,26
342,3
343,36
344,1
345,19
346,8
347,1
348,30
349,34
350,23
351,28
352,7
353,26
354,6
355,15
356,15
357,2
358,10
359,30
360,33
361,31
362,8
363,17
364,34
365,22
366,0
367,16
368,16
369,16
370,20
371,23
372,28
373,28
374,20
375,28
376,4
377,37
378,14
379,9
380,24
381,28
382,21
383,14
384,36
385,9
386,34
387,12
388,5
389,31
390,18
391,15
392,4
393,33
```

```
394,16
395,8
396,8
397,11
398,32
399,3
400,32
401,34
402,7
403,12
404,1
405,21
406,30
407,19
408,1
409,8
410,16
411,18
412,4
413,27
414,33
415,22
416,3
417,6
418,27
419,8
420,11
421,7
422,36
423,28
424,3
425,34
426,14
427,12
428,22
429,37
430,16
431,33
432,30
433,28
434,36
435,24
436,1
437,22
438,24
439,35
440,17
441,20
442,22
443,1
444,16
445,31
446,32
447,14
448,17
449,2
450,8
```

```
451,21
452,19
453,12
454,27
455,28
456,1
457,26
458,11
459,17
460,4
461,25
462,2
463,7
464,9
465,0
466,9
467,15
468,18
469,17
470,13
471,18
472,34
473,5
474,3
475,34
476,2
477,0
478,37
479,16
480,4
481,28
482,7
483,9
484,15
485,22
486,30
487,2
488,4
489,2
490,15
491,8
492,28
493,32
494,18
495,16
496,1
497,5
498,7
499,17
500,23
501,19
502,3
503,6
504,16
505,4
506,5
```

```
507,8
508,5
509,18
510,15
511,35
512,32
513,28
514,6
515,37
516,28
517,35
518,22
519,2
520,24
521,33
522,24
523,23
524,13
525,22
526,8
527,29
528,34
529,22
530,34
531,20
532,24
533,30
534,34
535,1
536,22
537,4
538,27
539,9
540,27
541,16
542,25
543,30
544,21
545,16
546,10
547,18
548,3
549,15
550,1
551,4
552,33
553,22
554,4
555,21
556,36
557,30
558,12
559,23
560,18
561,4
562,33
563,9
```

```
564,33
565,6
566,0
567,30
568,23
569,17
570,1
571,24
572,15
573,26
574,10
575,3
576,24
577,35
578,27
579,29
580,34
581,16
582,31
583,22
584,37
585,22
586,18
587,28
588,34
589,6
590,14
591,11
592,18
593,8
594,18
595,17
596,33
597,2
598,26
599,15
600,20
601,12
602,26
603,7
604,25
605,16
606,26
607,18
608,32
609,35
610,11
611,16
612,37
613,15
614,0
615,2
616,29
617,13
618,36
619,34
```

```
620,29
621,31
622,19
623,4
624,30
625,8
626,16
627,24
628,4
629,27
630,9
631,21
632,20
633,28
634,35
635,16
636,27
637,14
638,17
639,16
640,29
641,0
642,20
643,7
644,6
645,37
646,4
647,27
648,36
649,11
650,16
651,12
652,5
653,14
654,34
655,29
656,11
657,26
658,4
659,28
660,10
661,1
662,37
663,5
664,22
665,0
666,14
667,35
668,4
669,6
670,1
671,28
672,1
673,29
674,12
675,31
676.26
```

```
677,18
678,1
679,30
680,22
681,35
682,36
683,24
684,33
685,19
686,24
687,16
688,9
689,21
690,13
691,12
692,34
693,12
694,8
695,1
696,10
697,24
698,36
699,23
700,17
701,22
702,37
703,8
704,25
705,33
706,7
707,15
708,34
709,22
710,4
711,1
712,0
713,26
714,11
715,35
716,20
717,20
718,7
719,10
720,6
721,22
722,27
723,20
724,4
725,32
726,17
727,8
728,5
729,8
730,22
731,23
732,11
```

```
733,3
734,8
735,10
736,27
737,9
738,23
739,4
740,31
741,32
742,28
743,0
744,15
745,11
746,0
747,36
748,31
749,12
750,2
751,19
752,8
753,12
754,3
755,7
756,36
757,32
758,36
759,26
760,0
761,0
762,20
763,15
764,16
765,5
766,18
767,19
768,12
769,12
770,1
771,12
772,27
773,19
774,28
775,4
776,16
777,16
778,30
779,33
780,23
781,12
782,14
783,6
784,17
785,33
786,33
787,7
788,1
789.29
```

```
790,0
791,16
792,14
793,34
794,0
795,3
796,20
797,17
798,11
799,19
800,10
801,25
802,35
803,4
804,3
805,14
806,30
807,37
808,11
809,16
810,13
811,31
812,31
813,34
814,20
815,28
816,7
817,29
818,4
819,27
820,3
821,16
822,8
823,6
824,26
825,29
826,9
827,29
828,34
829,37
830,36
831,10
832,22
833,28
834,22
835,28
836,25
837,33
838,23
839,16
840,27
841,0
842,33
843,29
844,4
845,10
```

```
846,9
847,30
848,4
849,36
850,24
851,32
852,27
853,16
854,20
855,27
856,6
857,6
858,8
859,0
860,0
861,31
862,13
863,11
864,25
865,30
866,25
867,22
868,32
869,33
870,17
871,23
872,28
873,5
874,8
875,20
876,28
877,10
878,33
879,23
880,31
881,5
882,1
883,36
884,36
885,18
886,21
887,4
888,34
889,22
890,29
891,25
892,17
893,13
894,8
895,14
896,34
897,31
898,5
899,29
900,27
901,4
902.1
```

```
903,28
904,22
905,35
906,22
907,4
908,30
909,7
910,9
911,2
912,13
913,16
914,21
915,0
916,1
917,30
918,6
919,18
920,20
921,0
922,7
923,36
924,29
925,5
926,33
927,24
928,20
929,35
930,17
931,9
932,8
933,36
934,4
935,36
936,32
937,29
938,8
939,23
940,25
941,13
942,14
943,12
944,16
945,2
946,24
947,25
948,21
949,34
950,13
951,22
952,36
953,7
954,30
955,8
956,1
957,31
958,33
```

```
959,22
960,18
961,19
962,12
963,27
964,0
965,11
966,21
967,32
968,20
969,23
970,7
971,11
972,21
973,29
974,23
975,25
976,0
977,31
978,5
979,16
980,35
981,30
982,4
983,20
984,37
985,35
986,9
987,8
988,0
989,35
990,16
991,5
992,33
993,34
994,30
995,19
996,15
997,25
998,4
999,1
1000,17
1001,16
1002,36
1003,17
1004,16
1005,17
1006,17
1007,28
1008,22
1009,14
1010,30
1011,13
1012,18
1013,3
1014,9
1015,15
```

```
1016,34
1017,1
1018,10
1019,6
1020,8
1021,4
1022,17
1023,27
1024,22
1025,22
1026,0
1027,8
1028,26
1029,5
1030,16
1031,14
1032,4
1033,5
1034,4
1035,10
1036,23
1037,3
1038,22
1039,7
1040,17
1041,21
1042,18
1043,24
1044,17
1045,3
1046,25
1047,16
1048,4
1049,13
1050,36
1051,0
1052,11
1053,33
1054,15
1055,31
1056,29
1057,5
1058,29
1059,2
1060,15
1061,20
1062,26
1063,26
1064,2
1065,9
1066,25
1067,9
1068,28
1069,15
1070,30
1071,30
```

```
1072,19
1073,9
1074,35
1075,27
1076,10
1077,24
1078,2
1079,9
1080,32
1081,21
1082,5
1083,17
1084,26
1085,10
1086,16
1087,29
1088,35
1089,34
1090,7
1091,19
1092,37
1093,0
1094,16
1095,29
1096,21
1097,31
1098,32
1099,37
1100,31
1101,12
1102,16
1103,4
1104,25
1105,13
1106,1
1107,23
1108,1
1109,37
1110,35
1111,28
1112,19
1113,20
1114,25
1115,20
1116,25
1117,11
1118,22
1119,26
1120,36
1121,7
1122,24
1123,32
1124,29
1125,35
1126,24
1127,22
1128,20
```

```
1129,23
1130,27
1131,28
1132,11
1133,12
1134,35
1135,27
1136,21
1137,2
1138,18
1139,29
1140,10
1141,12
1142,30
1143,21
1144,32
1145,10
1146,22
1147,14
1148,4
1149,30
1150,21
1151,26
1152,23
1153,2
1154,11
1155,33
1156,0
1157,21
1158,22
1159,15
1160,16
1161,29
1162,18
1163,36
1164,0
1165,34
1166,25
1167,20
1168,18
1169,9
1170,17
1171,29
1172,20
1173,19
1174,23
1175,7
1176,3
1177,22
1178,8
1179,24
1180,17
1181,11
1182,33
1183,33
1184,0
```

```
1185,9
1186,18
1187,24
1188,9
1189,33
1190,24
1191,35
1192,14
1193,32
1194,36
1195,22
1196,20
1197,27
1198,28
1199,10
1200,15
1201,11
1202,26
1203,20
1204,7
1205,1
1206,33
1207,23
1208,34
1209,26
1210,2
1211,11
1212,23
1213,0
1214,14
1215,31
1216,30
1217,14
1218,7
1219,2
1220,2
1221,3
1222,7
1223,36
1224,17
1225,25
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-29-e99db9452246> in <cell line: 0>()
     10 #=== submit your file and indicate this was with an autoencoder
     11 from google.colab import files
---> 12 files.download(file_name)

NameError: name 'file_name' is not defined
```

```
#=== this will prepare a submission using the fine-tuned autoencoder
y_test = model.predict(x_test)
file_name = 'aesubmission.csv'  # Define file_name here
with open(file_name, 'w', newline='') as file:
  file.write('Id,Category\n')
  for k in range(y_test.shape[0]):
    line = str(k) + ',' + str(np.argmax(y_test[k]))
    print(line)
    file.write(line)
    file.write('\n')
#=== submit your file and indicate this was with an autoencoder
from google.colab import files
files.download(file_name)
```

```
39/39 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
0,37
1,12
2,12
3,12
4,25
5,4
6,18
7,7
8,7
9,16
10,3
11,36
12,16
13,2
14,12
15,7
16,37
17,28
18,13
19,37
20,13
21,26
22,34
23,1
24,17
25,11
26,9
27,16
28,31
29,21
30,1
31,17
32,35
33,0
34,20
35,22
36,36
37,37
38,24
39,27
40,6
41,6
42,25
43,29
44,33
45,19
46,10
47,25
48,7
49,12
50,6
51,34
52,0
53,35
54,18
```

```
55,32
56,37
57,12
58,3
59,20
60,36
61,9
62,10
63,12
64,30
65,7
66,19
67,37
68,16
69,21
70,18
71,33
72,28
73,9
74,9
75,28
76,33
77,26
78,13
79,2
80,23
81,12
82,28
83,13
84,9
85,30
86,14
87,11
88,37
89,14
90,24
91,1
92,7
93,25
94,29
95,20
96,35
97,2
98,24
99,16
100,3
101,11
102,12
103,13
104,18
105,17
106,5
107,15
108,27
109,26
110,22
111,4
```

```
111,4
112,36
113,24
114,21
115,19
116,33
117,17
118,31
119,19
120,19
121,36
122,5
123,33
124,22
125,9
126,33
127,18
128,28
129,22
130,36
131,12
132,2
133,22
134,7
135,26
136,23
137,24
138,12
139,35
140,6
141,27
142,6
143,32
144,12
145,14
146,19
147,18
148,6
149,24
150,3
151,29
152,22
153,13
154,20
155,1
156,9
157,24
158,34
159,33
160,28
161,27
162,28
163,36
164,19
165,15
166,7
167,8
```

```
168,1
169,7
170,1
171,9
172,2
173,27
174,29
175,23
176,3
177,32
178,6
179,2
180,23
181,28
182,20
183,27
184,27
185,13
186,7
187,7
188,8
189,19
190,14
191,35
192,10
193,19
194,21
195,14
196,24
197,30
198,6
199,6
200,29
201,25
202,35
203,5
204,24
205,22
206,21
207,20
208,25
209,3
210,24
211,3
212,31
213,24
214,14
215,22
216,15
217,12
218,34
219,5
220,15
221,1
222,17
223,15
224,37
```

```
224,37
225,22
226,13
227,37
228,15
229,16
230,19
231,21
232,3
233,4
234,37
235,36
236,0
237,9
238,6
239,31
240,16
241,26
242,18
243,24
244,19
245,20
246,6
247,26
248,32
249,28
250,4
251,0
252,9
253,24
254,30
255,3
256,3
257,25
258,32
259,10
260,34
261,22
262,5
263,12
264,1
265,10
266,34
267,34
268,31
269,23
270,37
271,23
272,16
273,18
274,27
275,8
276,6
277,8
278,16
279,4
280,25
```

```
281,32
282,28
283,17
284,30
285,14
286,35
287,19
288,34
289,25
290,28
291,25
292,37
293,29
294,10
295,29
296,2
297,17
298,7
299,6
300,5
301,27
302,15
303,34
304,8
305,34
306,2
307,2
308,21
309,34
310,18
311,35
312,6
313,23
314,0
315,1
316,13
317,20
318,0
319,14
320,35
321,21
322,21
323,1
324,31
325,32
326,29
327,11
328,10
329,28
330,36
331,16
332,23
333,9
334,19
335,23
336,36
337,32
```

```
338,30
339,25
340,33
341,26
342,3
343,36
344,1
345,19
346,8
347,1
348,30
349,34
350,23
351,28
352,7
353,26
354,6
355,15
356,15
357,2
358,10
359,30
360,33
361,31
362,8
363,17
364,34
365,22
366,0
367,16
368,16
369,16
370,20
371,23
372,28
373,28
374,20
375,28
376,4
377,37
378,14
379,9
380,24
381,28
382,21
383,14
384,36
385,9
386,34
387,12
388,5
389,31
390,18
391,15
392,4
393,33
```

```
394,16
395,8
396,8
397,11
398,32
399,3
400,32
401,34
402,7
403,12
404,1
405,21
406,30
407,19
408,1
409,8
410,16
411,18
412,4
413,27
414,33
415,22
416,3
417,6
418,27
419,8
420,11
421,7
422,36
423,28
424,3
425,34
426,14
427,12
428,22
429,37
430,16
431,33
432,30
433,28
434,36
435,24
436,1
437,22
438,24
439,35
440,17
441,20
442,22
443,1
444,16
445,31
446,32
447,14
448,17
449,2
450,8
```

```
451,21
452,19
453,12
454,27
455,28
456,1
457,26
458,11
459,17
460,4
461,25
462,2
463,7
464,9
465,0
466,9
467,15
468,18
469,17
470,13
471,18
472,34
473,5
474,3
475,34
476,2
477,0
478,37
479,16
480,4
481,28
482,7
483,9
484,15
485,22
486,30
487,2
488,4
489,2
490,15
491,8
492,28
493,32
494,18
495,16
496,1
497,5
498,7
499,17
500,23
501,19
502,3
503,6
504,16
505,4
506,5
```

```
507,8
508,5
509,18
510,15
511,35
512,32
513,28
514,6
515,37
516,28
517,35
518,22
519,2
520,24
521,33
522,24
523,23
524,13
525,22
526,8
527,29
528,34
529,22
530,34
531,20
532,24
533,30
534,34
535,1
536,22
537,4
538,27
539,9
540,27
541,16
542,25
543,30
544,21
545,16
546,10
547,18
548,3
549,15
550,1
551,4
552,33
553,22
554,4
555,21
556,36
557,30
558,12
559,23
560,18
561,4
562,33
563,9
```

```
564,33
565,6
566,0
567,30
568,23
569,17
570,1
571,24
572,15
573,26
574,10
575,3
576,24
577,35
578,27
579,29
580,34
581,16
582,31
583,22
584,37
585,22
586,18
587,28
588,34
589,6
590,14
591,11
592,18
593,8
594,18
595,17
596,33
597,2
598,26
599,15
600,20
601,12
602,26
603,7
604,25
605,16
606,26
607,18
608,32
609,35
610,11
611,16
612,37
613,15
614,0
615,2
616,29
617,13
618,36
619,34
```

```
620,29
621,31
622,19
623,4
624,30
625,8
626,16
627,24
628,4
629,27
630,9
631,21
632,20
633,28
634,35
635,16
636,27
637,14
638,17
639,16
640,29
641,0
642,20
643,7
644,6
645,37
646,4
647,27
648,36
649,11
650,16
651,12
652,5
653,14
654,34
655,29
656,11
657,26
658,4
659,28
660,10
661,1
662,37
663,5
664,22
665,0
666,14
667,35
668,4
669,6
670,1
671,28
672,1
673,29
674,12
675,31
676.26
```

```
677,18
678,1
679,30
680,22
681,35
682,36
683,24
684,33
685,19
686,24
687,16
688,9
689,21
690,13
691,12
692,34
693,12
694,8
695,1
696,10
697,24
698,36
699,23
700,17
701,22
702,37
703,8
704,25
705,33
706,7
707,15
708,34
709,22
710,4
711,1
712,0
713,26
714,11
715,35
716,20
717,20
718,7
719,10
720,6
721,22
722,27
723,20
724,4
725,32
726,17
727,8
728,5
729,8
730,22
731,23
732,11
```

```
733,3
734,8
735,10
736,27
737,9
738,23
739,4
740,31
741,32
742,28
743,0
744,15
745,11
746,0
747,36
748,31
749,12
750,2
751,19
752,8
753,12
754,3
755,7
756,36
757,32
758,36
759,26
760,0
761,0
762,20
763,15
764,16
765,5
766,18
767,19
768,12
769,12
770,1
771,12
772,27
773,19
774,28
775,4
776,16
777,16
778,30
779,33
780,23
781,12
782,14
783,6
784,17
785,33
786,33
787,7
788,1
789.29
```

```
790,0
791,16
792,14
793,34
794,0
795,3
796,20
797,17
798,11
799,19
800,10
801,25
802,35
803,4
804,3
805,14
806,30
807,37
808,11
809,16
810,13
811,31
812,31
813,34
814,20
815,28
816,7
817,29
818,4
819,27
820,3
821,16
822,8
823,6
824,26
825,29
826,9
827,29
828,34
829,37
830,36
831,10
832,22
833,28
834,22
835,28
836,25
837,33
838,23
839,16
840,27
841,0
842,33
843,29
844,4
845,10
```

```
846,9
847,30
848,4
849,36
850,24
851,32
852,27
853,16
854,20
855,27
856,6
857,6
858,8
859,0
860,0
861,31
862,13
863,11
864,25
865,30
866,25
867,22
868,32
869,33
870,17
871,23
872,28
873,5
874,8
875,20
876,28
877,10
878,33
879,23
880,31
881,5
882,1
883,36
884,36
885,18
886,21
887,4
888,34
889,22
890,29
891,25
892,17
893,13
894,8
895,14
896,34
897,31
898,5
899,29
900,27
901,4
902.1
```

```
903,28
904,22
905,35
906,22
907,4
908,30
909,7
910,9
911,2
912,13
913,16
914,21
915,0
916,1
917,30
918,6
919,18
920,20
921,0
922,7
923,36
924,29
925,5
926,33
927,24
928,20
929,35
930,17
931,9
932,8
933,36
934,4
935,36
936,32
937,29
938,8
939,23
940,25
941,13
942,14
943,12
944,16
945,2
946,24
947,25
948,21
949,34
950,13
951,22
952,36
953,7
954,30
955,8
956,1
957,31
958,33
```

```
959,22
960,18
961,19
962,12
963,27
964,0
965,11
966,21
967,32
968,20
969,23
970,7
971,11
972,21
973,29
974,23
975,25
976,0
977,31
978,5
979,16
980,35
981,30
982,4
983,20
984,37
985,35
986,9
987,8
988,0
989,35
990,16
991,5
992,33
993,34
994,30
995,19
996,15
997,25
998,4
999,1
1000,17
1001,16
1002,36
1003,17
1004,16
1005,17
1006,17
1007,28
1008,22
1009,14
1010,30
1011,13
1012,18
1013,3
1014,9
1015,15
```

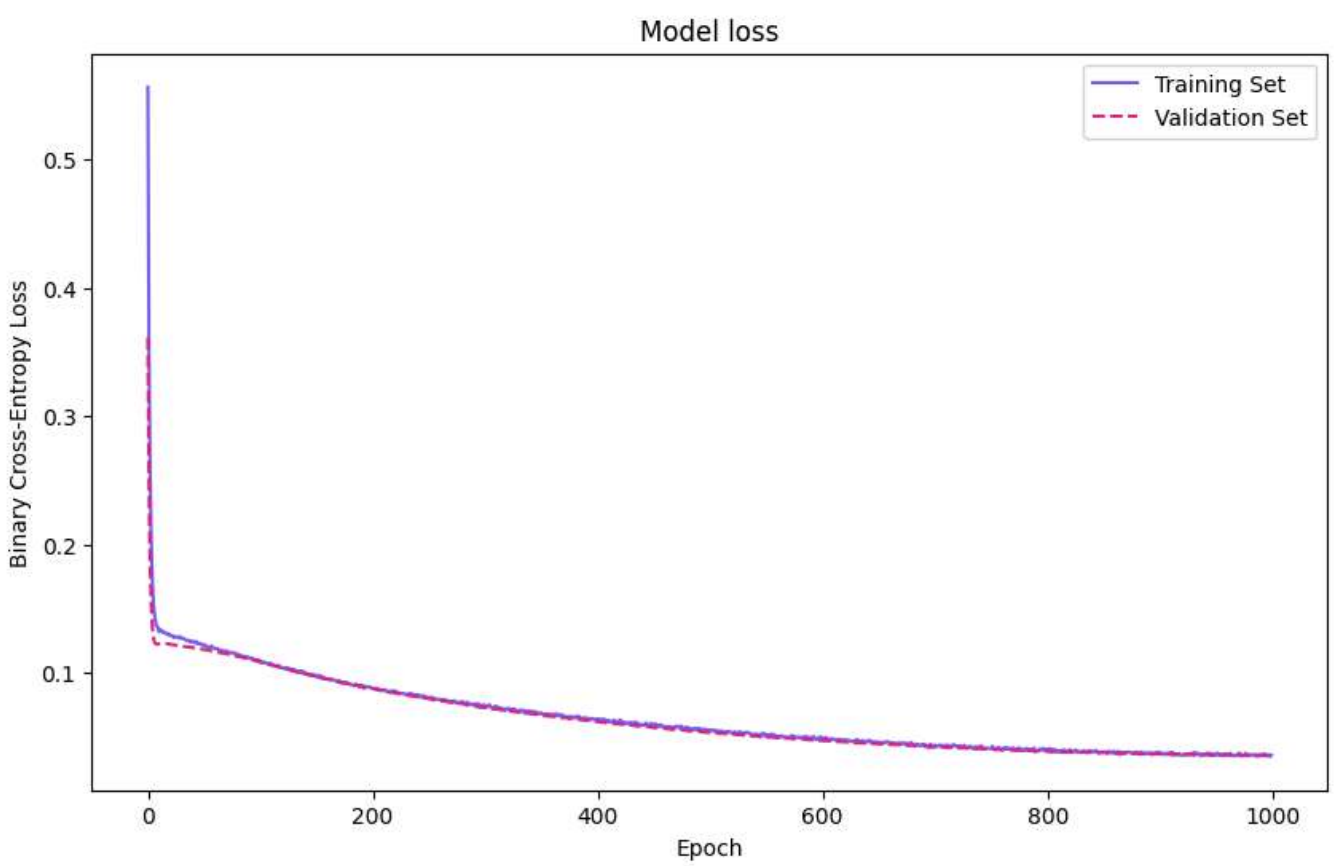
```
1016,34
1017,1
1018,10
1019,6
1020,8
1021,4
1022,17
1023,27
1024,22
1025,22
1026,0
1027,8
1028,26
1029,5
1030,16
1031,14
1032,4
1033,5
1034,4
1035,10
1036,23
1037,3
1038,22
1039,7
1040,17
1041,21
1042,18
1043,24
1044,17
1045,3
1046,25
1047,16
1048,4
1049,13
1050,36
1051,0
1052,11
1053,33
1054,15
1055,31
1056,29
1057,5
1058,29
1059,2
1060,15
1061,20
1062,26
1063,26
1064,2
1065,9
1066,25
1067,9
1068,28
1069,15
1070,30
1071,30
```

```
1072,19
1073,9
1074,35
1075,27
1076,10
1077,24
1078,2
1079,9
1080,32
1081,21
1082,5
1083,17
1084,26
1085,10
1086,16
1087,29
1088,35
1089,34
1090,7
1091,19
1092,37
1093,0
1094,16
1095,29
1096,21
1097,31
1098,32
1099,37
1100,31
1101,12
1102,16
1103,4
1104,25
1105,13
1106,1
1107,23
1108,1
1109,37
1110,35
1111,28
1112,19
1113,20
1114,25
1115,20
1116,25
1117,11
1118,22
1119,26
1120,36
1121,7
1122,24
1123,32
1124,29
1125,35
1126,24
1127,22
1128,20
```

```
1129,23
1130,27
1131,28
1132,11
1133,12
1134,35
1135,27
1136,21
1137,2
1138,18
1139,29
1140,10
1141,12
1142,30
1143,21
1144,32
1145,10
1146,22
1147,14
1148,4
1149,30
1150,21
1151,26
1152,23
1153,2
1154,11
1155,33
1156,0
1157,21
1158,22
1159,15
1160,16
1161,29
1162,18
1163,36
1164,0
1165,34
1166,25
1167,20
1168,18
1169,9
1170,17
1171,29
1172,20
1173,19
1174,23
1175,7
1176,3
1177,22
1178,8
1179,24
1180,17
1181,11
1182,33
1183,33
1184,0
```

```
1185,9
1186,18
1187,24
1188,9
1189,33
1190,24
1191,35
1192,14
1193,32
1194,36
1195,22
1196,20
1197,27
1198,28
1199,10
1200,15
1201,11
1202,26
1203,20
1204,7
1205,1
1206,33
1207,23
1208,34
1209,26
1210,2
1211,11
1212,23
1213,0
1214,14
1215,31
1216,30
1217,14
1218,7
1219,2
1220,2
1221,3
1222,7
1223,36
1224,17
1225,25
```

```
import matplotlib.pyplot as plt

fig = plt.figure(figsize=(10,6))
plt.plot(hist.history['loss'], color='#785ef0')
plt.plot(hist.history['val_loss'], '--', color='#dc267f')
# plt.yscale('log')
plt.title('Model loss')
plt.ylabel('Binary Cross-Entropy Loss')
plt.xlabel('Epoch')
plt.legend(['Training Set', 'Validation Set'], loc='upper right')
plt.show()
```



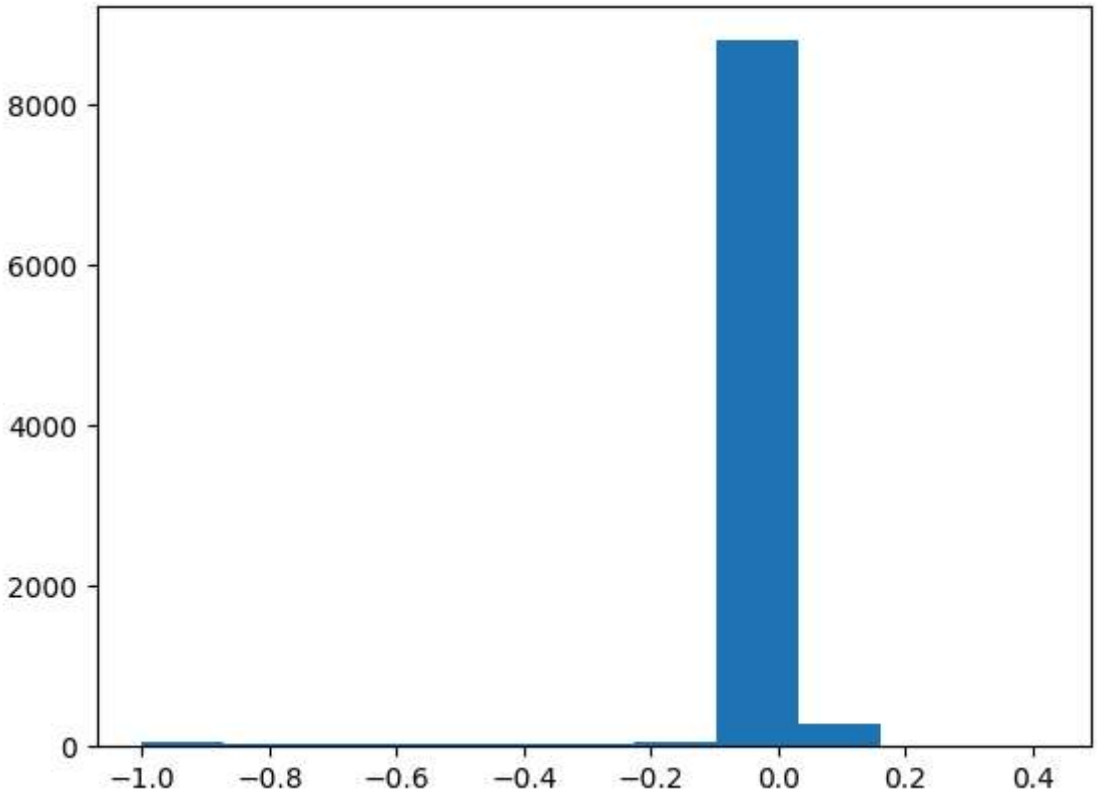

```
plt.figure(figsize=(10, 9.5))
for i, comp in enumerate(model.get_weights()[0].T):
    plt.subplot(10, 10, i + 1)
    plt.imshow(comp.reshape((59, 51)), cmap=plt.cm.gray_r,
               interpolation='nearest')
    plt.xticks(())
    plt.yticks(())
```

```
plt.subplots_adjust(0.08, 0.02, 0.92, 0.85, 0.08, 0.23)
```

```
import random
import numpy as np
import matplotlib.pyplot as plt
```

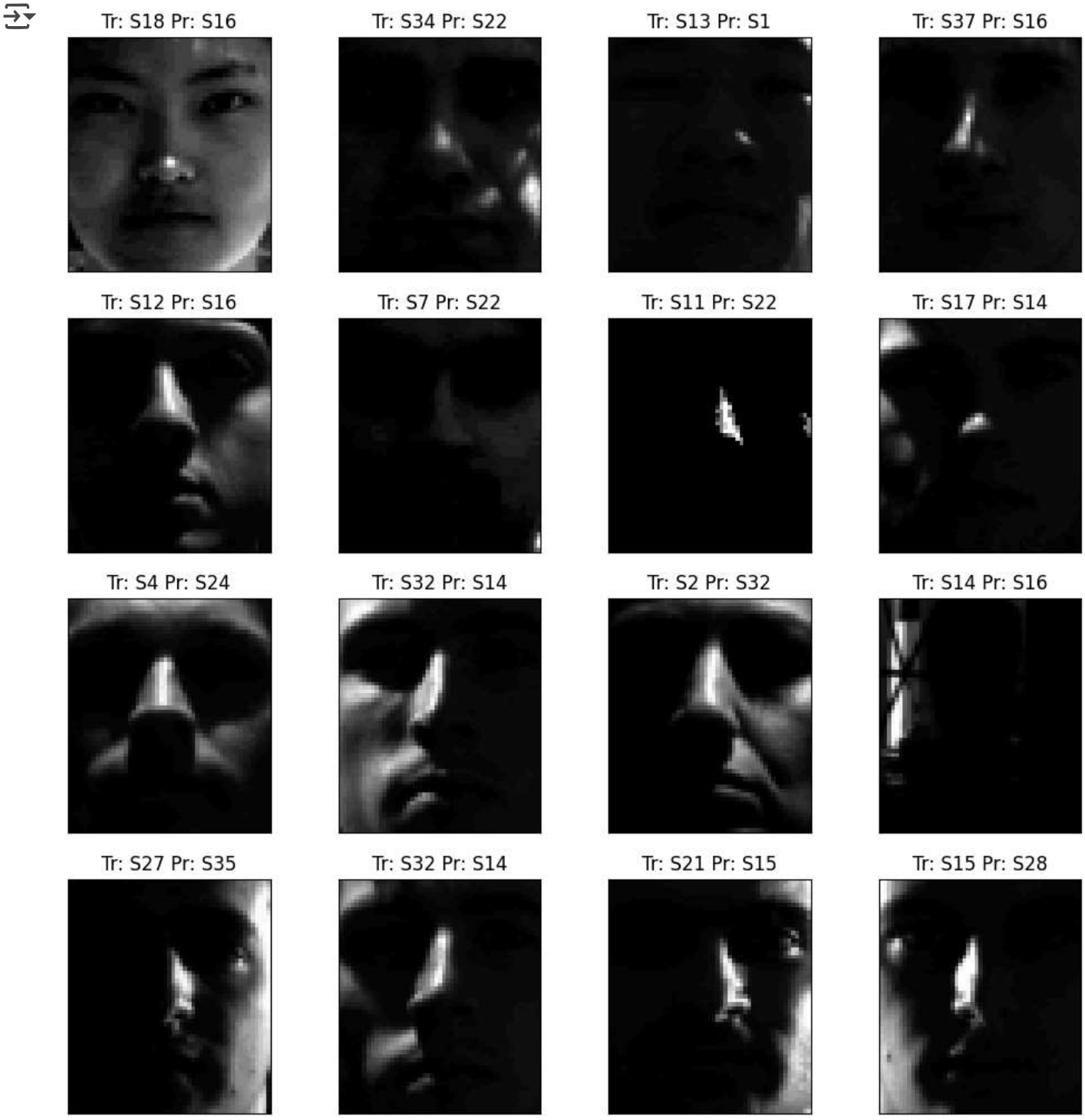
```
y_hat = model.predict(x_val)
diff = y_hat - y_val
plt.hist(diff.flatten(), bins=11)
plt.show()  #error histogram

inc_idx = []
for idx in range(y_hat.shape[0]):
  if np.argmax(y_hat[idx]) != np.argmax(y_val[idx]):
    inc_idx.append(idx)

sorted_idx = list(np.argsort(np.sum(np.abs(y_hat[inc_idx] - y_val[inc_idx]), axis=1)))

inc_idx = [inc_idx[i] for i in sorted_idx]
```

```
8/8 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
```




```
import matplotlib.pyplot as plt
import numpy as np

labels = np.argmax(y_val, axis=1, out=None)

clsmap = {}
for k in range(len(set(labels))):
  clsmap[k] = 'S' + str(k)


plt.figure(figsize=(12,6))
for i, (img, y) in enumerate(zip(x_val[inc_idx[0:8]].reshape(8, 59, 51), labels[inc_idx[0:8]
```

```
    plt.subplot(241 + i)
    plt.imshow(img, cmap='gray')
    plt.xticks([])
    plt.yticks([])
    plt.title('Tr: '+clsmap[y]+' Pr: '+clsmap[np.argmax(y_hat[inc_idx[i]])])
plt.savefig('best.errors.png', bbox_inches='tight', dpi=350)
plt.show()

plt.figure(figsize=(12,6))
for i, (img, y) in enumerate(zip(x_val[inc_idx[-8:]].reshape(8, 59, 51), labels[inc_idx[-8:]
    plt.subplot(241 + i)
    plt.imshow(img, cmap='gray')
    plt.xticks([])
    plt.yticks([])
    plt.title('Tr: '+clsmap[y]+' Pr: '+clsmap[np.argmax(y_hat[inc_idx[-8+i]])])
plt.savefig('worst.errors.png', bbox_inches='tight', dpi=350)
plt.show()
```


Tr: S18 Pr: S16
Tr: S34 Pr: S22
Tr: S13 Pr: S1
Tr: S37 Pr: S16
Tr: S12 Pr: S16
Tr: S7 Pr: S22
Tr: S11 Pr: S22
Tr: S17 Pr: S14
Tr: S4 Pr: S24
Tr: S32 Pr: S14
Tr: S2 Pr: S32
Tr: S14 Pr: S16
Tr: S27 Pr: S35
Tr: S32 Pr: S14
Tr: S21 Pr: S15
Tr: S15 Pr: S28

```
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import balanced_accuracy_score
import matplotlib.pyplot as plt
import numpy as np
import umap
import seaborn as sns

labels = np.argmax(y_val, axis=1, out=None)

y_ = labels
y_hat = model.predict(x_val)
y_pred = np.argmax(y_hat, axis=1)

print(y_hat.shape)
print(y_pred.shape)

print(y_hat[0] - y_val[0])
print(classification_report(np.argmax(y_val, axis=1), np.argmax(y_hat, axis=1), labels=list(
cm = confusion_matrix(np.argmax(y_val, axis=1), np.argmax(y_hat, axis=1), labels=list(set(la
print(cm)
ber = 1- balanced_accuracy_score(np.argmax(y_val, axis=1), np.argmax(y_hat, axis=1))
print('BER', ber)

fig, ax = plt.subplots(figsize=(14,12))
im = ax.imshow(cm, interpolation='nearest', cmap=plt.cm.Greens)
ax.figure.colorbar(im, ax=ax)

# Show ticks
ax.set(xticks=np.arange(cm.shape[1]),
       yticks=np.arange(cm.shape[0]),
       # add class labels
       xticklabels=set(labels),
       yticklabels=set(labels),
       title='Confusion Matrix  (BER = {0:0.3f})'.format(ber),
       ylabel='True Class',
       xlabel='Predicted Class')

# Rotate the tick labels and set their alignment.
# plt.setp(ax.get_xticklabels(), rotation=45, ha="right",
#          rotation_mode="anchor")

# Loop over data dimensions and create text annotations.
thresh = cm.max() / 2.
for i in range(cm.shape[0]):
    for j in range(cm.shape[1]):
        ax.text(j, i, format(cm[i, j], 'd'),
                ha="center", va="center",
                color="white" if cm[i, j] > thresh else "black")
fig.tight_layout()
np.set_printoptions(precision=2)
```

```
    plt.axis('tight')
    plt.show()
```

```
    X_ = umap.UMAP().fit_transform(y_hat)
    print(X_.shape)
```

```
    plt.figure(figsize=(10,8))
    plt.title('MLP Model')
    plt.scatter(X_[:,0], X_[:,1], s=5.0, c=y_, alpha=0.75, cmap='tab10')
    plt.xlabel('First UMAP dimension')
    plt.ylabel('Second UMAP dimension')
    plt.colorbar()
```

```
8/8 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
(245, 38)
(245,)
[ 2.50224546e-02  2.64551919e-02  1.05842378e-03  3.81335244e-02
  3.53887081e-02  4.92968364e-04  7.36522581e-03  4.33879392e-03
  8.81808228e-04  1.17752679e-04  6.93155394e-04  2.70228105e-04
  4.70103550e-04  2.77041877e-03  1.74948722e-02  2.35831190e-04
  1.61786797e-03  2.44183233e-03  3.41564454e-02  1.46230974e-03
  1.87493803e-04  1.16121024e-03  1.21114249e-06  8.52978677e-02
  1.91274151e-01  7.13271947e-05  5.59029868e-03  1.03480920e-01
  7.75130000e-03  2.85991147e-04  7.89475860e-04  4.90800850e-03
  5.13022496e-05  2.41050002e-05  6.43085630e-04 -6.22603238e-01
  7.37877330e-04  1.62676035e-03]
              precision    recall  f1-score   support

           0       0.60      1.00      0.75         6
           1       0.71      1.00      0.83         5
           2       0.86      0.75      0.80         8
           3       1.00      0.88      0.93         8
           4       1.00      0.80      0.89         5
           5       1.00      0.50      0.67         8
           6       0.83      1.00      0.91         5
           7       1.00      0.88      0.93         8
           8       1.00      1.00      1.00         2
           9       1.00      0.80      0.89        10
          10       1.00      0.88      0.93         8
          11       1.00      0.86      0.92         7
          12       1.00      0.71      0.83         7
          13       1.00      0.91      0.95        11
          14       0.62      0.71      0.67         7
          15       0.88      0.78      0.82         9
          16       0.56      1.00      0.71         5
          17       1.00      0.75      0.86         8
          18       1.00      0.75      0.86         8
          19       0.90      1.00      0.95         9
          20       1.00      1.00      1.00         7
          21       1.00      0.80      0.89         5
          22       0.57      1.00      0.73         4
          23       0.75      0.75      0.75         4
          24       0.80      1.00      0.89         4
          25       1.00      1.00      1.00         4
          26       1.00      1.00      1.00         6
          27       1.00      0.60      0.75         5
          28       0.88      1.00      0.93         7
          29       0.83      1.00      0.91         5
          30       0.50      1.00      0.67         3
          31       1.00      1.00      1.00         4
          32       0.75      0.82      0.78        11
          33       1.00      1.00      1.00         3
          34       0.67      0.86      0.75         7
          35       0.90      0.75      0.82        12
          36       0.67      1.00      0.80         2
          37       1.00      0.88      0.93         8

    accuracy                           0.86       245
   macro avg       0.88      0.88      0.86       245
```

```
weighted avg       0.89      0.86      0.86       245

[[6 0 0 ... 0 0 0]
 [0 5 0 ... 0 0 0]
 [0 0 6 ... 0 0 0]
 ...
 [2 1 0 ... 9 0 0]
 [0 0 0 ... 0 2 0]
 [0 0 0 ... 0 0 7]]
BER 0.12110769347611461
```