TSNE

November 23, 2021

[34]:
```
import pandas as pd
import numpy as np
import os
import glob
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib
import seaborn as sns
from sklearn.manifold import TSNE
import matplotlib as mpl
```

[10]:
```
#model_folder = 'models/02_03_21-15/architectures_cpc.cpc_combined.CPCCombined'
#model_folder = 'models/02_03_21-15/architectures_cpc.cpc_combined.CPCCombined'
#model_folder = 'models/09_03_21-18/architectures_cpc.cpc_combined.CPCCombined'
#model_folder = 'models/10_03_21-18/architectures_cpc.cpc_combined.CPCCombined1'
#model_folder = '/home/julian/Downloads/Github/contrastive-predictive-coding/
 ↪models/16_03_21-17/architectures_cpc.cpc_combined.CPCCombined0'
model_folder = '/home/julian/Downloads/Github/contrastive-predictive-coding/
 ↪models/16_08_21-10-16-test|(40x)cpc/14_08_21-15_36-train|(4x)cpc/
 ↪architectures_cpc.cpc_combined.
 ↪CPCCombined2|train-test-splits-fewer-labels60|use_weights|unfrozen|C|m:
 ↪all|cpc_downstream_cnn'
```

[11]:
```
pred_path = glob.glob(os.path.join(model_folder ,'*output.csv'))[0]
dfp = pd.read_csv(pred_path)
pred = dfp.values[:, 1:]
dfp
```

[11]:

	Unnamed: 0	10370003	11157007	\
0	/media/julian/data/data/ECG/ptbxl_challenge/HR…	0.000333	0.000956	
1	/media/julian/data/data/ECG/ptbxl_challenge/HR…	0.000930	0.004560	
2	/media/julian/data/data/ECG/ptbxl_challenge/HR…	0.000063	0.000660	
3	/media/julian/data/data/ECG/ptbxl_challenge/HR…	0.000027	0.000165	
4	/media/julian/data/data/ECG/ptbxl_challenge/HR…	0.000626	0.000553	
…	…	…	…	
8437	/media/julian/data/data/ECG/china_challenge/Q2289	0.000094	0.000204	
8438	/media/julian/data/data/ECG/china_challenge/Q2565	0.000111	0.000289	

```
8439  /media/julian/data/data/ECG/china_challenge/Q1927  0.000101  0.000261
8440  /media/julian/data/data/ECG/china_challenge/Q1845  0.000519  0.000785
8441  /media/julian/data/data/ECG/china_challenge/Q3536  0.000805  0.000747

      111975006  164861001  164865005  164867002  164873001  164884008  \
0      0.019975   0.010873   0.074232   0.010617   0.025496   0.009704
1      0.013027   0.018212   0.079470   0.008430   0.010485   0.127720
2      0.019547   0.022087   0.037891   0.019960   0.180355   0.014676
3      0.217853   0.104753   0.013420   0.010242   0.021190   0.009752
4      0.018315   0.001648   0.031800   0.009988   0.008634   0.015942
…         …          …          …          …          …          …
8437   0.027954   0.143319   0.222282   0.078769   0.272639   0.023970
8438   0.020319   0.013210   0.015619   0.007116   0.070466   0.026711
8439   0.008309   0.003686   0.096263   0.037706   0.104612   0.019850
8440   0.102021   0.126415   0.149909   0.046377   0.072142   0.052704
8441   0.082296   0.066030   0.077987   0.034551   0.048868   0.031576

      164889003  …  63593006   6374002  67198005  67741000119109  698252002  \
0      0.008610  …  0.001236  0.000215  0.000029        0.011185   0.026917
1      0.172605  …  0.024956  0.001059  0.000006        0.016636   0.011573
2      0.034115  …  0.000509  0.000015  0.000072        0.029602   0.014867
3      0.085852  …  0.000276  0.000020  0.000035        0.030892   0.013179
4      0.008793  …  0.004771  0.000741  0.000271        0.015365   0.005547
…         …  …        …         …         …               …          …
8437   0.161740  …  0.003473  0.001105  0.000012        0.072298   0.041522
8438   0.008593  …  0.001007  0.000349  0.000007        0.027856   0.005870
8439   0.018145  …  0.000458  0.000284  0.000074        0.010211   0.009146
8440   0.129777  …  0.001254  0.000253  0.001408        0.022237   0.026338
8441   0.189925  …  0.001785  0.000418  0.000003        0.054708   0.033866

      713422000  713426002  713427006  74390002  89792004
0      0.000104   0.001759   0.000052  0.000066  0.000191
1      0.000084   0.006981   0.000559  0.000108  0.001634
2      0.000359   0.025970   0.000791  0.000354  0.000275
3      0.000437   0.033761   0.006104  0.000029  0.002814
4      0.000359   0.006099   0.000131  0.000123  0.000723
…         …          …          …         …         …
8437   0.004884   0.004581   0.000344  0.000018  0.000450
8438   0.000087   0.011458   0.002290  0.000621  0.001430
8439   0.000274   0.013402   0.000762  0.000063  0.000345
8440   0.001729   0.002815   0.000633  0.000038  0.000289
8441   0.002188   0.004746   0.000119  0.000013  0.000751

[8442 rows x 68 columns]
```

[12]:
```
label_path = glob.glob(os.path.join(model_folder ,'labels-dataloader-1.csv'))[0]
print(label_path)
```

```
dfl = pd.read_csv(label_path)
LABELS = dfl.values[:, 1:].astype(int)
dfl
```

/home/julian/Downloads/Github/contrastive-predictive-coding/models/16_08_21-10-16-test|(40x)cpc/14_08_21-15_36-train|(4x)cpc/architectures_cpc.cpc_combined.CPCCombined2|train-test-splits-fewer-labels60|use_weights|unfrozen|C|m:all|cpc_downstream_cnn/labels-dataloader-1.csv

[12]:

	Unnamed: 0	10370003	11157007	\
0	/media/julian/data/data/ECG/ptbxl_challenge/HR…	0.0	0.0	
1	/media/julian/data/data/ECG/ptbxl_challenge/HR…	0.0	0.0	
2	/media/julian/data/data/ECG/ptbxl_challenge/HR…	0.0	0.0	
3	/media/julian/data/data/ECG/ptbxl_challenge/HR…	0.0	0.0	
4	/media/julian/data/data/ECG/ptbxl_challenge/HR…	0.0	0.0	
…	…	…	…	
8437	/media/julian/data/data/ECG/china_challenge/Q2289	0.0	0.0	
8438	/media/julian/data/data/ECG/china_challenge/Q2565	0.0	0.0	
8439	/media/julian/data/data/ECG/china_challenge/Q1927	0.0	0.0	
8440	/media/julian/data/data/ECG/china_challenge/Q1845	0.0	0.0	
8441	/media/julian/data/data/ECG/china_challenge/Q3536	0.0	0.0	

	111975006	164861001	164865005	164867002	164873001	164884008	\
0	0.0	0.0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	0.0	
3	0.0	1.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	
…	…	…	…	…	…	…	
8437	0.0	0.0	0.0	1.0	0.0	0.0	
8438	0.0	0.0	0.0	0.0	0.0	0.0	
8439	0.0	0.0	0.0	1.0	0.0	0.0	
8440	0.0	1.0	0.0	0.0	0.0	0.0	
8441	0.0	0.0	0.0	1.0	0.0	0.0	

	164889003	…	63593006	6374002	67198005	67741000119109	698252002	\
0	0.0	…	0.0	0.0	0.0	0.0	0.0	
1	0.0	…	0.0	0.0	0.0	0.0	0.0	
2	0.0	…	0.0	0.0	0.0	0.0	0.0	
3	0.0	…	0.0	0.0	0.0	0.0	0.0	
4	0.0	…	0.0	0.0	0.0	0.0	0.0	
…	…	…	…	…	…	…	…	
8437	0.0	…	0.0	0.0	0.0	0.0	0.0	
8438	0.0	…	0.0	0.0	0.0	0.0	0.0	
8439	0.0	…	0.0	0.0	0.0	0.0	0.0	
8440	0.0	…	0.0	0.0	0.0	0.0	0.0	
8441	0.0	…	0.0	0.0	0.0	0.0	0.0	

```
      713422000  713426002  713427006  74390002  89792004
0           0.0        0.0        0.0       0.0       0.0
1           0.0        0.0        0.0       0.0       0.0
2           0.0        0.0        0.0       0.0       0.0
3           0.0        0.0        0.0       0.0       0.0
4           0.0        0.0        0.0       0.0       0.0
…           …          …          …         …         …
8437        0.0        0.0        0.0       0.0       0.0
8438        0.0        0.0        0.0       0.0       0.0
8439        0.0        0.0        0.0       0.0       0.0
8440        0.0        0.0        0.0       0.0       0.0
8441        0.0        0.0        0.0       0.0       0.0

[8442 rows x 68 columns]
```

[13]:
```
np.set_printoptions(suppress=True)
pred_sum = np.round(np.sum(pred, axis=0).astype(float), 0).astype(int)
pred_sum
```

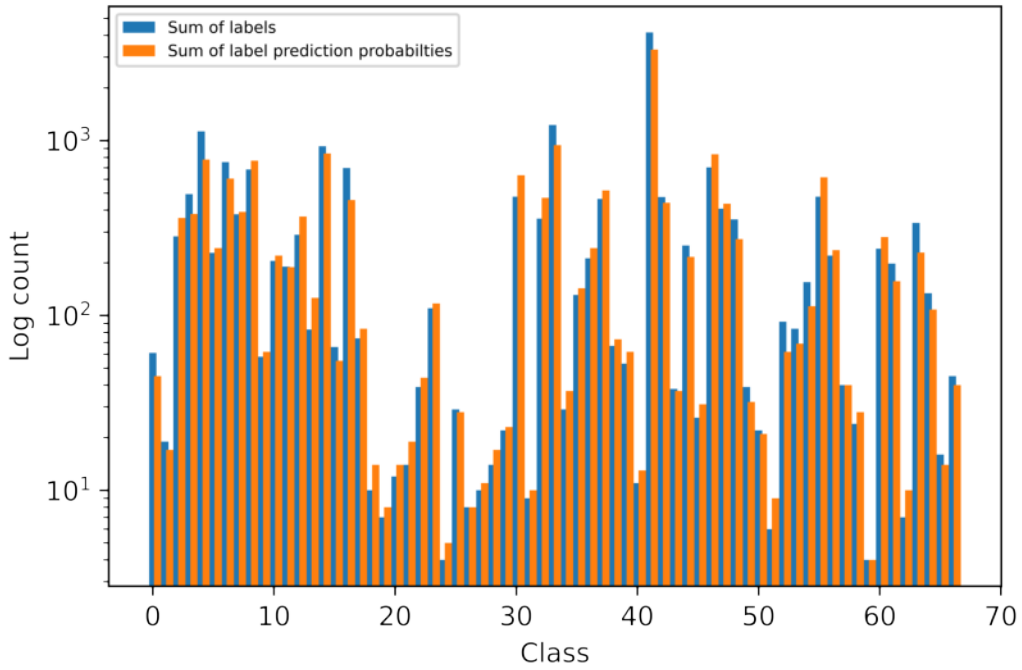
[13]:
```
array([  45,   17,  361,  380,  779,  243,  606,  391,  767,   62,  220,
        189,  368,  126,  844,   55,  457,   84,   14,    8,   14,   19,
         44,  117,    5,   28,    8,   11,   17,   23,  634,   10,  471,
        941,   37,  143,  243,  519,   73,   62,   13, 3316,  441,   37,
        216,   31,  835,  435,  273,   32,   21,    9,   62,   69,  113,
        616,  236,   40,   28,    4,  281,  157,   10,  229,  108,   14,
         40])
```

[14]:
```
label_sum = np.sum(LABELS, axis=0)
label_sum
```

[14]:
```
array([  61,   19,  283,  494, 1128,  228,  753,  379,  684,   58,  205,
        190,  289,   83,  928,   66,  696,   74,   10,    7,   12,   14,
         39,  110,    4,   29,    8,   10,   14,   22,  478,    9,  358,
       1227,   29,  131,  212,  464,   67,   53,   11, 4149,  476,   38,
        251,   26,  702,  408,  354,   39,   22,    6,   92,   84,  155,
        478,  220,   40,   24,    4,  241,  198,    7,  338,  134,   16,
         45])
```

[8]:
```
fig = plt.figure(dpi=1200)
width = 0.6
plt.bar(np.arange(len(label_sum)), label_sum, width=width, label='Sum of␣
 ↪labels')
plt.bar(np.arange(len(label_sum))+(1-width), pred_sum, width=width, label='Sum␣
 ↪of label prediction probabilties')
plt.ylabel('Log count')
plt.xlabel('Class')
```

```
plt.yscale('log')
plt.legend(prop={'size': 6})
plt.show()
```



0.1 Calculate ROC

[9]:
```
import numpy as np
import matplotlib.pyplot as plt
from itertools import cycle

from sklearn import svm, datasets
from sklearn.metrics import roc_curve, auc
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import label_binarize
from sklearn.multiclass import OneVsRestClassifier
from scipy import interp
from sklearn.metrics import roc_auc_score
```

[10]:
```
def plot_roc_singleclass(tpr, fpr, roc_auc, n, fill_area=False):
    plt.figure(dpi=1200)
    lw = 2
    plt.plot(fpr[n], tpr[n], color='darkorange',
             lw=lw, label='ROC curve')
    if fill_area:
```

```
        plt.fill_between(fpr[n], tpr[n], label='(Area Under Curve = AUC = %0.
↪2f)' % roc_auc[n])
    plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic example')
    plt.legend(loc="lower right")
    plt.show()
```

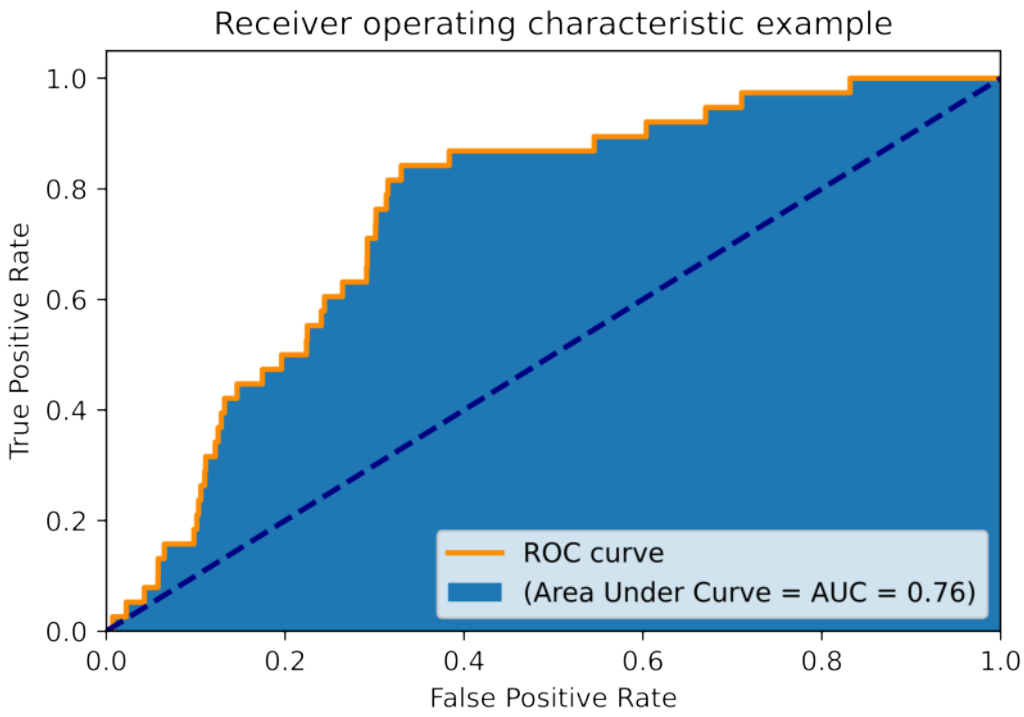
[11]:
```
def plot_roc_multiclass(tpr, fpr, roc_auc, selection=None, plot_averages=True,␣
↪plot_legend=True):
    selection = range(n_classes) if selection is None else selection
    all_fpr = np.unique(np.concatenate([fpr[i] for i in selection]))
    lw = 1
    # Then interpolate all ROC curves at this points
    mean_tpr = np.zeros_like(all_fpr)
    for i in selection:
        mean_tpr += interp(all_fpr, fpr[i], tpr[i])

    # Finally average it and compute AUC
    mean_tpr /= len(selection)

    fpr["macro"] = all_fpr
    tpr["macro"] = mean_tpr
    roc_auc["macro"] = auc(fpr["macro"], tpr["macro"])

    # Plot all ROC curves
    plt.figure(dpi=1200)
    if plot_averages:
        plt.plot(fpr["micro"], tpr["micro"],
                 label='micro-average ROC curve (area = {0:0.2f})'
                       ''.format(roc_auc["micro"]),
                 color='deeppink', linestyle=':', linewidth=4)

        plt.plot(fpr["macro"], tpr["macro"],
                 label='macro-average ROC curve (area = {0:0.2f})'
                       ''.format(roc_auc["macro"]),
                 color='navy', linestyle=':', linewidth=4)

    colors = sns.color_palette("Paired", len(selection)) #cycle(['aqua',␣
↪'darkorange', 'cornflowerblue'])
    for i, color in zip(selection, colors):
        plt.plot(fpr[i], tpr[i], color=color, lw=lw,
                 label='ROC curve of class {0} (area = {1:0.2f})'
                 ''.format(i, roc_auc[i]))
```

```
    plt.plot([0, 1], [0, 1], 'k--', lw=lw)
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Some extension of Receiver operating characteristic to␣
↪multi-class')
    plt.tight_layout()
    if plot_legend:
        plt.legend(loc="upper left", bbox_to_anchor=(1.1, 1), fontsize=6)
    plt.show()
```

Below taken (and changed) from: https://github.com/physionetchallenges/evaluation-2020/blob/master/evaluate_12ECG_score.py

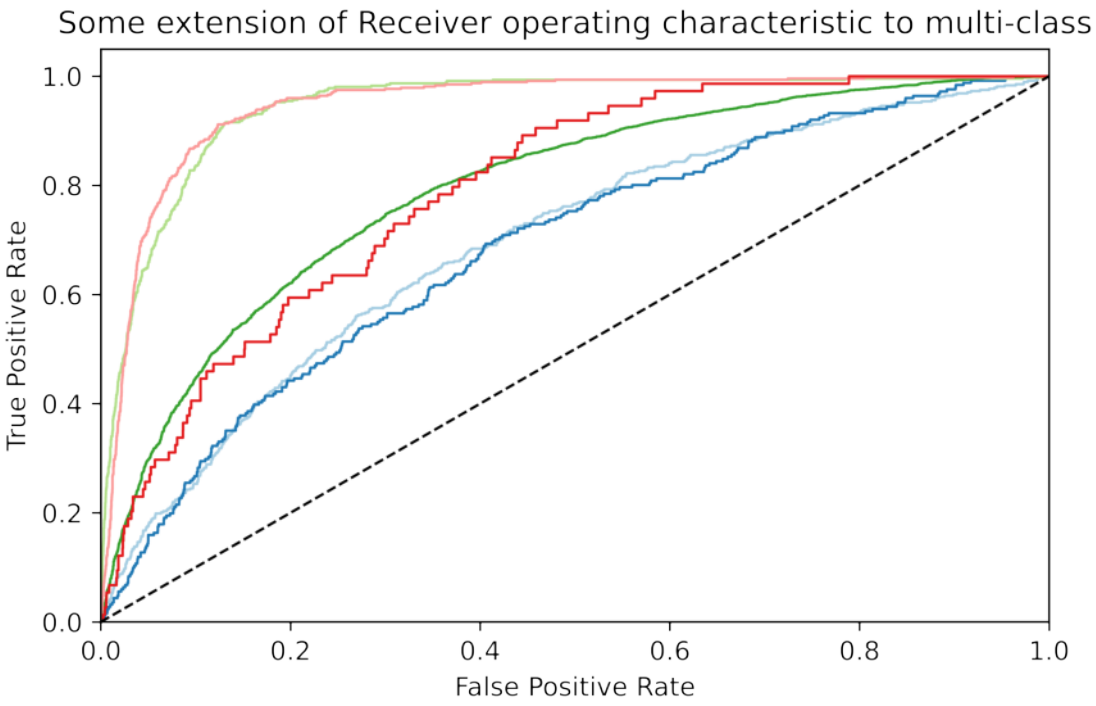
[12]:
```
n_classes = LABELS.shape[1]
fpr = dict()
tpr = dict()
thresholds = dict() #no dict because always the same
roc_auc = dict()
for i in range(n_classes):
    fpr[i], tpr[i], thresholds[i] = roc_curve(LABELS[:, i], pred[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])
fpr["micro"], tpr["micro"], _ = roc_curve(LABELS.ravel(), pred.ravel())
roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])
```

[12]:
```
plot_roc_singleclass(tpr, fpr, roc_auc, 43, fill_area=True) #normal class
```



[13]:
```
physionetchallenge_sel =␣
↪'270492004,427393009,426177001,426783006,427084000,17338001'.split(',')
physionetchallenge_sel = [dfl.columns.get_loc(s)-1 for s in␣
↪physionetchallenge_sel]
plot_roc_multiclass(tpr, fpr, roc_auc, selection=physionetchallenge_sel,␣
↪plot_averages=False, plot_legend=False)
```

```
/home/julian/anaconda3/envs/ml/lib/python3.7/site-
packages/ipykernel_launcher.py:8: DeprecationWarning: scipy.interp is deprecated
and will be removed in SciPy 2.0.0, use numpy.interp instead
```



0.2 Import latent and context data

[2]:
```
import pickle
with open('/home/julian/Downloads/Github/contrastive-predictive-coding/
 ↪experiments/data_output/17_08_21-18/14_08_21-15_36-train|(4x)cpc/
 ↪architectures_cpc.cpc_combined.
 ↪CPCCombined1|train-test-splits-fewer-labels60|use_weights|strided|frozen|C|m:
 ↪all|cpc_downstream_cnn/latent_List.pickle', 'rb') as f:
    latent_list = pickle.load(f)
```

[3]:
```
print('shape of one latent:', latent_list[0]['latents'].shape)
```

```
shape of one latent: (9, 1, 128)
```

[4]:
```
print('shape of one context:', latent_list[0]['context'].shape)
```

```
shape of one context: (1, 256)
```

[5]:
```
all_latents = np.concatenate([np.squeeze(l['latents']) for l in latent_list],␣
 ↪axis=0)
print('got latents:', all_latents.shape)
```

```
got latents: (113643, 128)
```

[6]:
```
all_contexts = np.stack([np.squeeze(l['context']) for l in latent_list], axis=0)
print('got contexts:', all_contexts.shape)
```

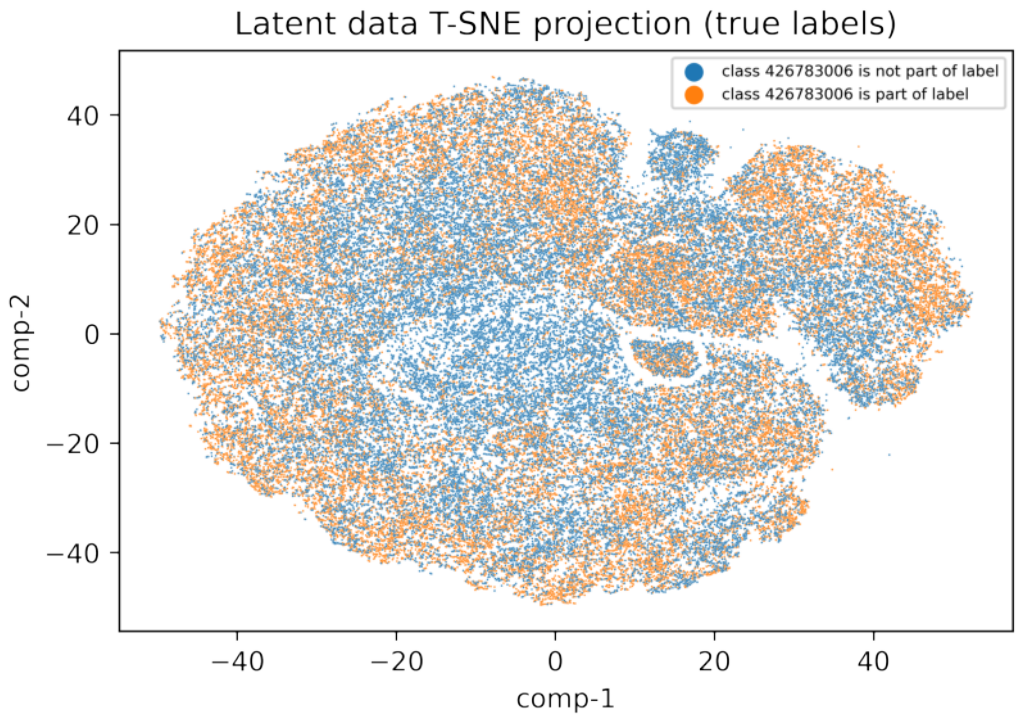
```
got contexts: (12627, 256)
```

0.3 Calc TSNE representation (latent)

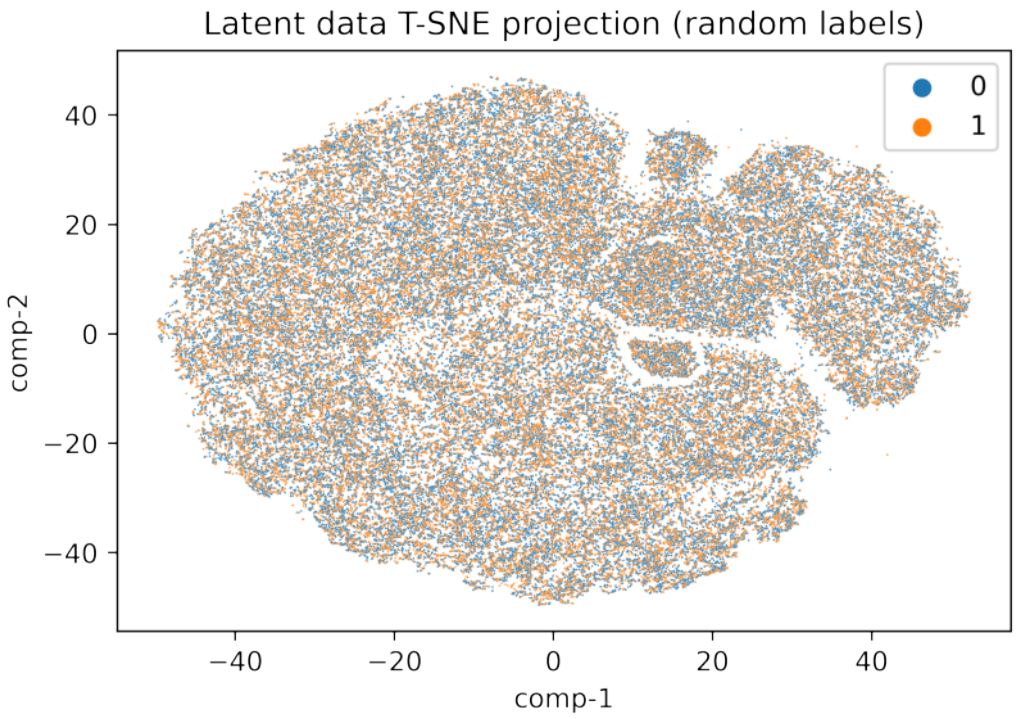
[7]:
```
np.random.seed(0)
X_embedded = TSNE(n_components=2, n_jobs=6).fit_transform(all_latents)
X_embedded.shape
```

[7]:
```
(113643, 2)
```

[17]:
```
selected_class = '426783006'
normal_class_ix = dfl.columns[1:].get_loc(selected_class)
first_labels = [item for sublist in [[1]*9 if normal_class_ix in l['labels'][1]␣
 ↪else [0]*9 for l in latent_list] for item in sublist]
df = pd.DataFrame()
df["y"] = first_labels
df["comp-1"] = X_embedded[:,0]
df["comp-2"] = X_embedded[:,1]
fig = plt.figure(dpi=600)
plt.tight_layout()
ax = sns.scatterplot(x="comp-1", y="comp-2", hue=df.y.tolist(),
                data=df, s=0.5)
ax.set(title="Latent data T-SNE projection (true labels)")
handles, labels  =  ax.get_legend_handles_labels()

ax.legend(handles, [f'class {selected_class} is not part of label', f'class␣
 ↪{selected_class} is part of label'], fontsize='xx-small')
fig.savefig('/home/julian/Downloads/tsne-latent-41.png', dpi=fig.dpi)
```



[19]:
```
import random
df = pd.DataFrame()
df["y"] = first_labels
df["comp-1"] = X_embedded[:,0]
df["comp-2"] = X_embedded[:,1]
fig = plt.figure(dpi=600)
plt.tight_layout()
random_labels = first_labels.copy()
random.shuffle(random_labels)
sns.scatterplot(x="comp-1", y="comp-2", hue=random_labels,
                data=df, s=0.5).set(title="Latent data T-SNE projection (random␣
 ↪labels)")
fig.savefig('/home/julian/Downloads/tsne-latent-(random).png', dpi=fig.dpi)
```

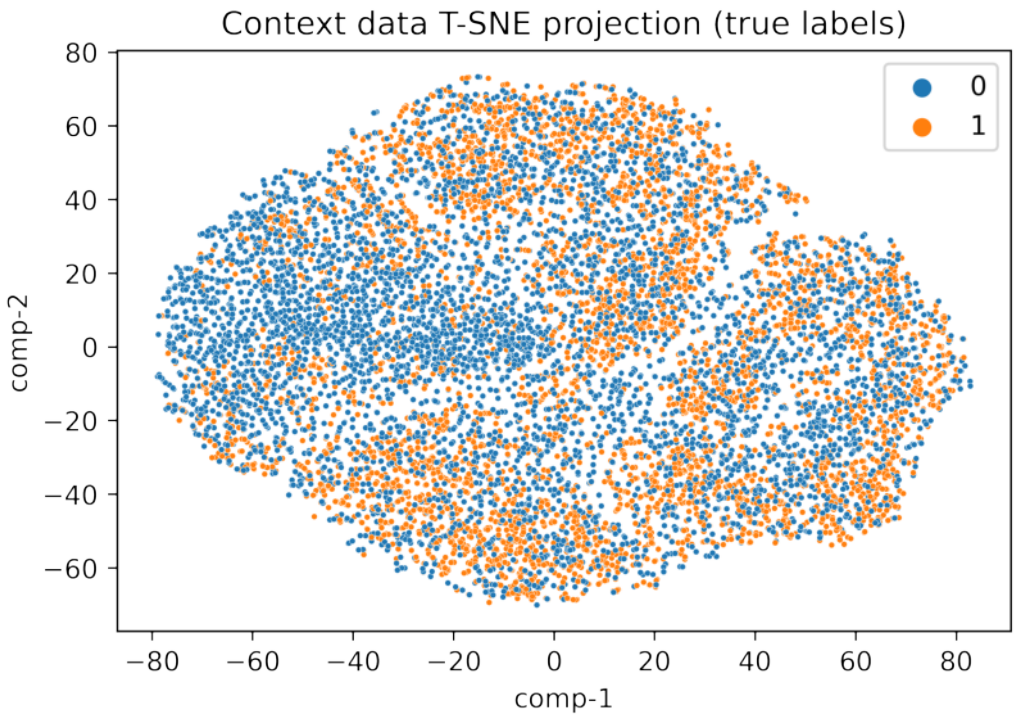


0.4 Calc TSNE representation (context)

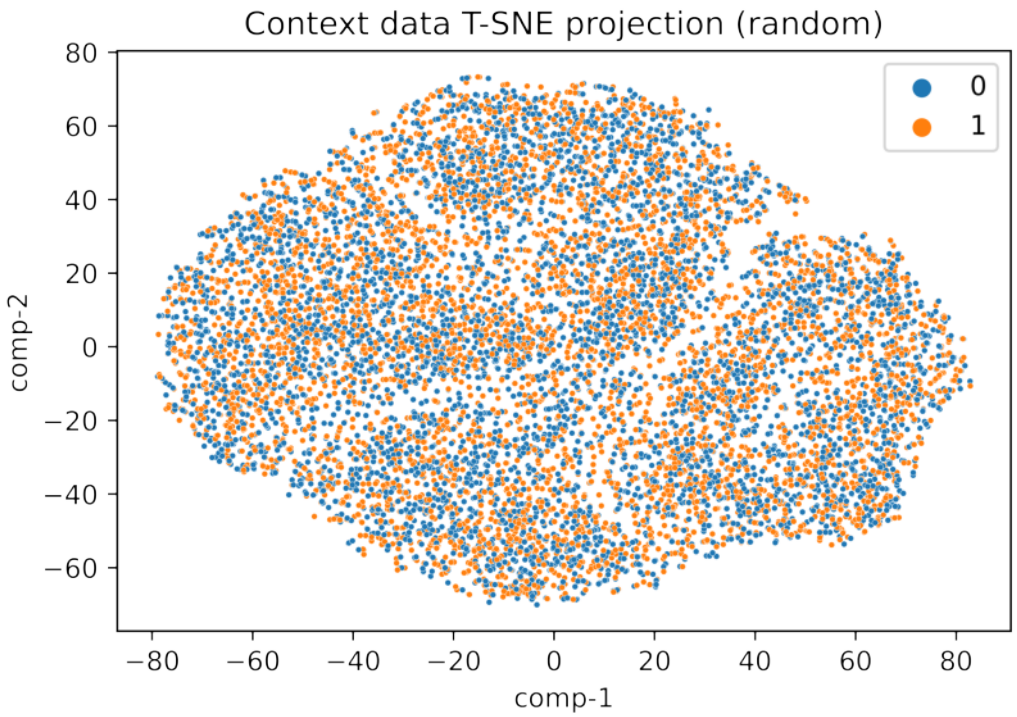
[20]:
```
np.random.seed(0)
C_embedded = TSNE(n_components=2, n_jobs=6).fit_transform(all_contexts)
C_embedded.shape
```

[20]:
```
(12627, 2)
```

[23]:
```
normal_class_ix = dfl.columns[1:].get_loc('426783006')
labels = [1 if normal_class_ix in l['labels'][1] else 0 for l in latent_list]
df = pd.DataFrame()
df["y"] = labels
df["comp-1"] = C_embedded[:,0]
df["comp-2"] = C_embedded[:,1]
fig = plt.figure(dpi=600)
sns.scatterplot(x="comp-1", y="comp-2", hue=df.y.tolist(),
                data=df, s=5).set(title="Context data T-SNE projection (true␣
 ↪labels)")
fig.savefig('/home/julian/Downloads/tsne-context-41.png', dpi=fig.dpi)
```



[24]:
```
df = pd.DataFrame()
df["y"] = labels
df["comp-1"] = C_embedded[:,0]
df["comp-2"] = C_embedded[:,1]
fig = plt.figure(dpi=600)
plt.tight_layout()
random_labels = labels.copy()
random.shuffle(random_labels)
sns.scatterplot(x="comp-1", y="comp-2", hue=random_labels,
                data=df, s=5).set(title="Context data T-SNE projection␣
↪(random)")
fig.savefig('tsne-context-(random).png', dpi=fig.dpi)
```



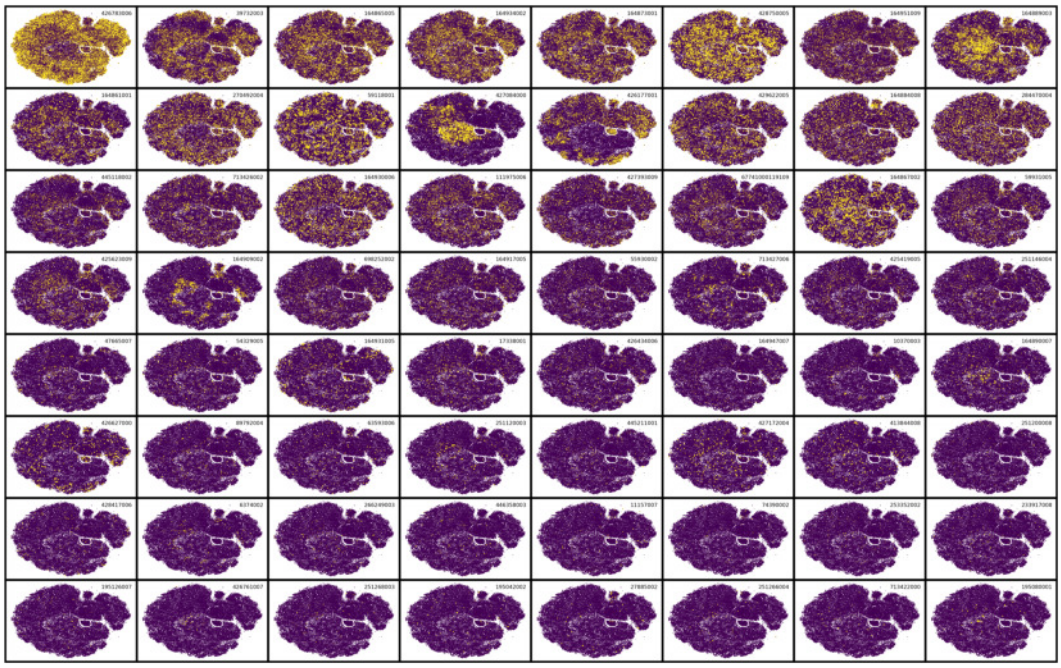
0.5 Calculate for all classes

[24]:
```
def get_label_for_label_index(all_labels, index, repeat=1):
    labels = [item for sublist in [[1]*repeat if index in l['labels'][1] else␣
 ↪[0]*repeat for l in latent_list] for item in sublist]
    return np.array(labels)
```

0.6 order by count

[26]:
```
all_labels = [l['labels'][1] for l in latent_list]
fig, axs = plt.subplots(8, 8, dpi=1200, sharex='all', gridspec_kw={'wspace':0,␣
 ↪'hspace':0})
cmap = matplotlib.colors.ListedColormap(['red' ,'black'])
label_index_order = np.argsort(-1*np.sum(LABELS, axis=0))
plt.axis('on')
plt.tight_layout()
for i in range(8):
    for j in range(8):
        ax = axs[i, j]
        ax.set_yticklabels([])
        ax.set_xticklabels([])
        ax.tick_params(axis='both', which='both', bottom=False, top=False,␣
 ↪left=False, right=False, labelbottom=False)
```

```
        n = i*8+j
        labels = get_label_for_label_index(all_labels, label_index_order[n],␣
↪repeat=9)
        ax.scatter(X_embedded[:,0], X_embedded[:,1], c=labels, s=(labels+0.3)/
↪2, linewidths=0, marker='.', label=dfl.columns[1:][label_index_order[n]])
        ax.legend(prop={'size':2}, loc=1, frameon=False)
fig.savefig('/home/julian/Downloads/tsne-latent-all.png', dpi=fig.dpi)
```

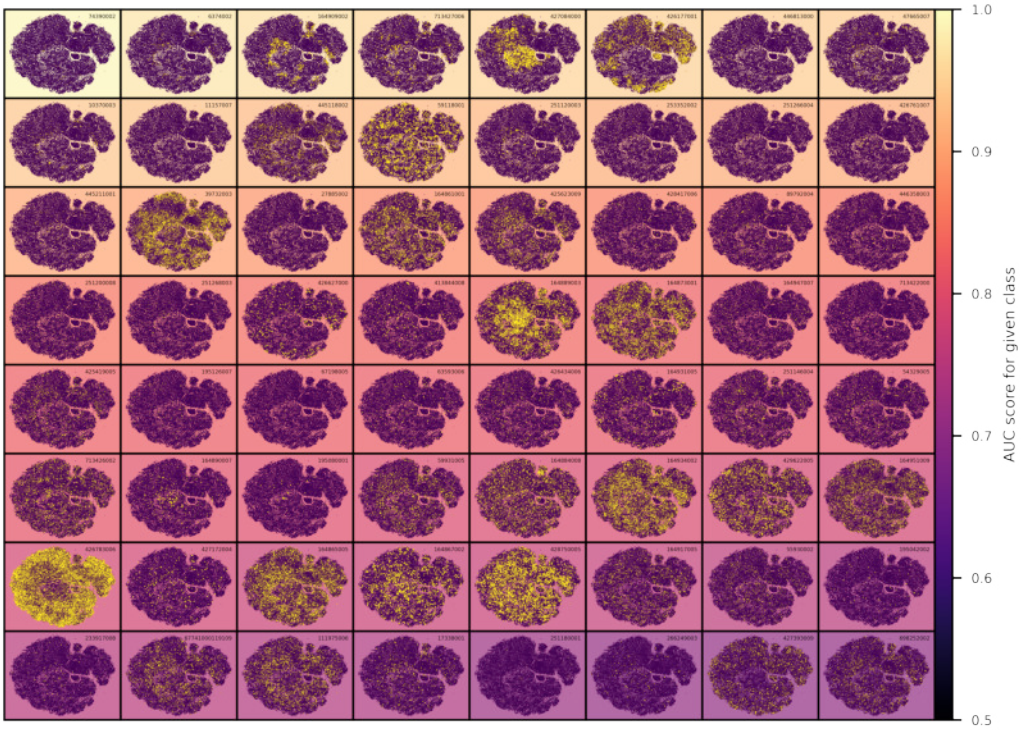


0.7 order by auc score

[78]:
```
classes = "10370003 &  11157007 &  111975006 &  164861001 &  164865005 & ␣
↪164867002 &  164873001 &  164884008 &  164889003 &  164890007 &  164909002 &␣
↪ 164917005 &  164930006 &  164931005 &  164934002 &  164947007 &  164951009␣
↪&  17338001 &  195042002 &  195080001 &  195126007 &  233917008 &  251120003␣
↪&  251146004 &  251180001 &  251200008 &  251266004 &  251268003 & ␣
↪253352002 &  266249003 &  270492004 &  27885002 &  284470004 &  39732003 & ␣
↪413844008 &  425419005 &  425623009 &  426177001 &  426434006 &  426627000 &␣
↪ 426761007 &  426783006 &  427084000 &  427172004 &  427393009 &  428417006␣
↪&  428750005 &  429622005 &  445118002 &  445211001 &  446358003 & ␣
↪446813000 &  47665007 &  54329005 &  55930002 &  59118001 &  59931005 & ␣
↪63593006 &  6374002 &  67198005 &  67741000119109 &  698252002 &  713422000␣
↪&  713426002 &  713427006 &  74390002 &  89792004".replace(' ', '').
↪split('&')
```

```
aucs = '0.940 &     0.935 &      0.743 &      0.875 &      0.773 &      0.772 &␣
 ↪     0.828 &      0.794 &      0.834 &      0.805 &      0.965 &      0.766␣
 ↪&      0.694 &      0.814 &      0.794 &      0.828 &      0.785 &      0.740␣
 ↪&      0.751 &      0.802 &      0.823 &      0.751 &      0.920 &      0.
 ↪813 &      0.719 &      0.850 &      0.912 &      0.844 &      0.914 &     ␣
 ↪0.708 &      0.685 &      0.886 &      0.638 &      0.899 &      0.835 &     ␣
 ↪0.825 &      0.874 &      0.951 &      0.817 &      0.840 &      0.909 &     ␣
 ↪ 0.784 &      0.954 &      0.783 &      0.707 &      0.864 &      0.770 &   ␣
 ↪  0.789 &      0.924 &      0.905 &      0.857 &      0.946 &      0.944 &   ␣
 ↪ 0.812 &     0.757 &     0.921 &     0.796 &     0.821 &    0.973 &     0.
 ↪823 &           0.751 &      0.707 &      0.828 &      0.810 &      0.956 & ␣
 ↪   0.992 &     0.860'.replace(' ', '').split('&')
aucs = list(map(float, aucs))
class_auc_map = dict(zip(classes, aucs))
sorted_classes = sorted(class_auc_map.items(), key=lambda x: -x[1]) #sort by␣
 ↪auc score
label_index_order = [dfl.columns[1:].get_loc(s[0]) for s in sorted_classes]
color_norm = matplotlib.colors.Normalize(.5, 1.)

cmap = sns.color_palette("magma", as_cmap=True)

all_labels = [l['labels'][1] for l in latent_list]
fig, axs = plt.subplots(8, 8, dpi=1200, sharex='all', gridspec_kw={'wspace':0,␣
 ↪'hspace':0})
plt.axis('on')
for i in range(8):
    for j in range(8):
        ax = axs[i, j]
        ax.set_yticklabels([])
        ax.set_xticklabels([])
        ax.tick_params(axis='both', which='both', bottom=False, top=False,␣
 ↪left=False, right=False, labelbottom=False)
        n = i*8+j
        labels = get_label_for_label_index(all_labels, label_index_order[n],␣
 ↪repeat=9)
        class_name = dfl.columns[1:][label_index_order[n]]
        pcm = ax.scatter(X_embedded[:,0], X_embedded[:,1], c=labels,␣
 ↪s=(labels+0.2)/2, linewidths=0, marker='.', label=class_name)
        ax.legend(prop={'size':2}, loc=1, frameon=False)
        ax.set_facecolor(cmap(color_norm(class_auc_map[class_name]), alpha=0.7))
plt.tight_layout()
cax,kw = mpl.colorbar.make_axes([ax for ax in axs.flat], aspect=40, pad=0.0)
cbar = plt.colorbar(mpl.cm.ScalarMappable(norm=color_norm, cmap=cmap), cax=cax,␣
 ↪**kw)
cbar.set_label('AUC score for given class', rotation=90, fontsize='xx-small')
cbar.ax.tick_params(labelsize=5)
```

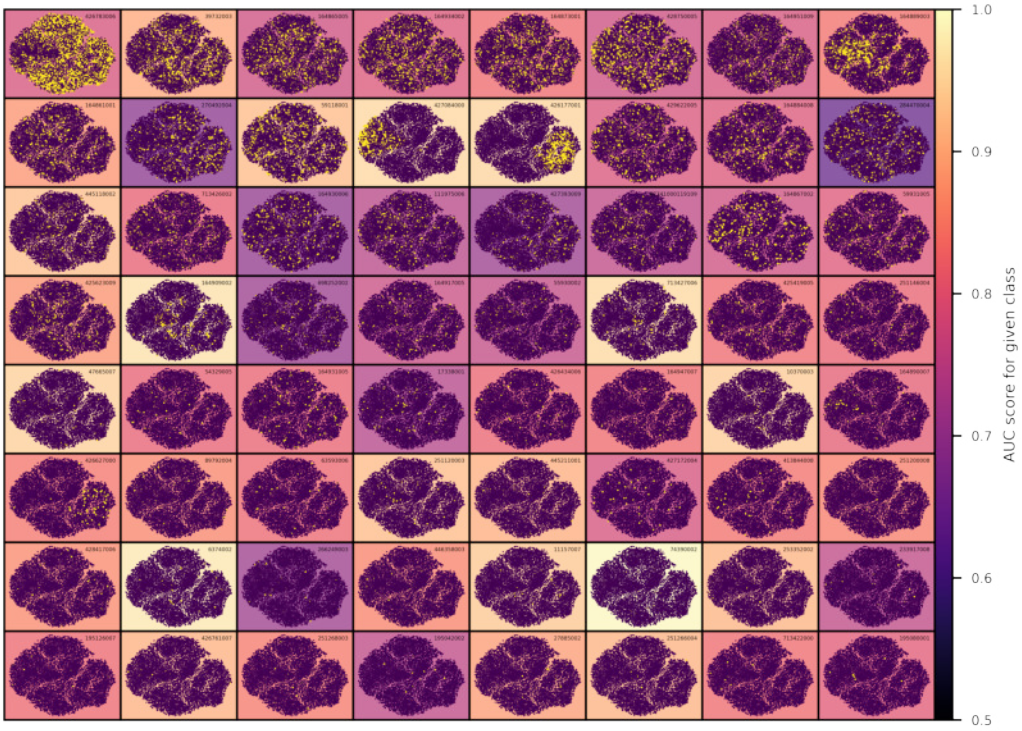
```
fig.savefig('/home/julian/Downloads/tsne-latent-all(ordered by auc).png',␣
 ↪dpi=fig.dpi, bbox_inches='tight')
```



0.8 order by count

[57]:
```
all_labels = [l['labels'][1] for l in latent_list]
fig, axs = plt.subplots(8, 8, dpi=1200, sharex='all', gridspec_kw={'wspace':0,␣
 ↪'hspace':0})
cmap = matplotlib.colors.ListedColormap(['red' ,'black'])
label_index_order = np.argsort(-1*np.sum(LABELS, axis=0))
color_norm = matplotlib.colors.Normalize(0.5, 1.0)
cmap = sns.color_palette("magma", as_cmap=True)
plt.axis('on')
#fig.suptitle('t-SNE context embedding for all classes sorted by occurences in␣
 ↪data')
for i in range(8):
    for j in range(8):
        ax = axs[i, j]
        ax.set_yticklabels([])
        ax.set_xticklabels([])
        ax.tick_params(axis='both', which='both', bottom=False, top=False,␣
 ↪left=False, right=False, labelbottom=False)
```

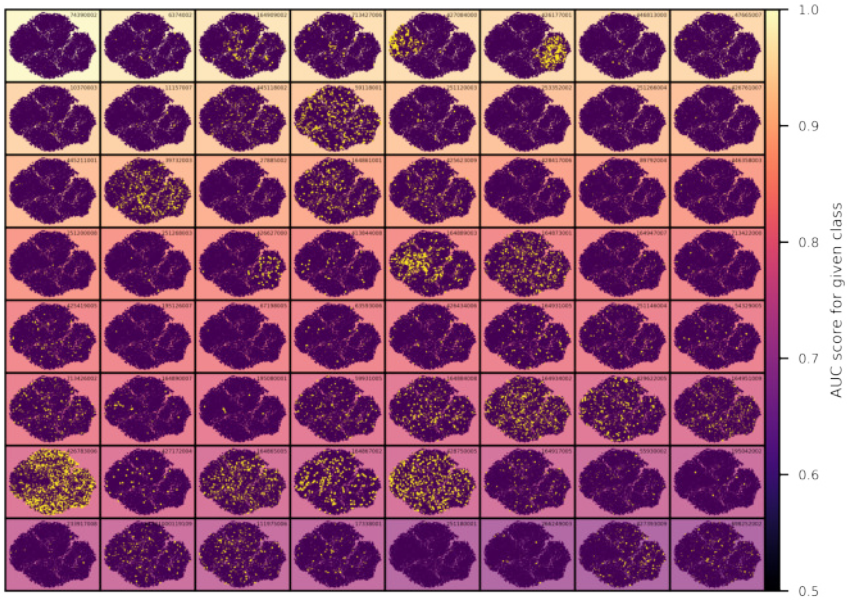
```
        n = i*8+j
        labels = get_label_for_label_index(all_labels, label_index_order[n],␣
↪repeat=1)
        class_name = dfl.columns[1:][label_index_order[n]]
        ax.scatter(C_embedded[:,0], C_embedded[:,1], c=labels, s=(labels+1),␣
↪linewidths=0, marker='.', label=class_name)
        #ax.set_facecolor('xkcd:salmon')
        ax.legend(prop={'size':2}, loc=1, frameon=False)
        ax.set_facecolor(cmap(color_norm(class_auc_map[class_name]), alpha=0.7))
plt.tight_layout()
cax,kw = mpl.colorbar.make_axes([ax for ax in axs.flat], aspect=40, pad=0.0)
cbar = plt.colorbar(mpl.cm.ScalarMappable(norm=color_norm, cmap=cmap), cax=cax,␣
↪**kw)
cbar.set_label('AUC score for given class', rotation=90, fontsize='xx-small')
cbar.ax.tick_params(labelsize=5)
fig.savefig('/home/julian/Downloads/tsne-context-all(ordererd by count).png',␣
↪dpi=fig.dpi)
```



[77]:

```
classes = "10370003 &  11157007 &  111975006 &  164861001 &  164865005 & ␣
 ↪164867002 &  164873001 &  164884008 &  164889003 &  164890007 &  164909002 &␣
 ↪ 164917005 &  164930006 &  164931005 &  164934002 &  164947007 &  164951009␣
 ↪&  17338001 &  195042002 &  195080001 &  195126007 &  233917008 &  251120003␣
 ↪&  251146004 &  251180001 &  251200008 &  251266004 &  251268003 & ␣
 ↪253352002 &  266249003 &  270492004 &  27885002 &  284470004 &  39732003 & ␣
 ↪413844008 &  425419005 &  425623009 &  426177001 &  426434006 &  426627000 &␣
 ↪ 426761007 &  426783006 &  427084000 &  427172004 &  427393009 &  428417006␣
 ↪&  428750005 &  429622005 &  445118002 &  445211001 &  446358003 & ␣
 ↪446813000 &  47665007 &  54329005 &  55930002 &  59118001 &  59931005 & ␣
 ↪63593006 &  6374002 &  67198005 &  67741000119109 &  698252002 &  713422000␣
 ↪&  713426002 &  713427006 &  74390002 &  89792004".replace(' ', '').
 ↪split('&')
aucs = '0.940 &     0.935 &      0.743 &      0.875 &      0.773 &      0.772 &␣
 ↪     0.828 &      0.794 &      0.834 &      0.805 &      0.965 &      0.766␣
 ↪&      0.694 &      0.814 &      0.794 &      0.828 &      0.785 &     0.740␣
 ↪&      0.751 &      0.802 &      0.823 &      0.751 &      0.920 &      0.
 ↪813 &      0.719 &      0.850 &      0.912 &      0.844 &      0.914 &     ␣
 ↪0.708 &      0.685 &     0.886 &      0.638 &     0.899 &      0.835 &     ␣
 ↪0.825 &      0.874 &      0.951 &      0.817 &      0.840 &      0.909 &    ␣
 ↪ 0.784 &      0.954 &      0.783 &      0.707 &      0.864 &      0.770 &   ␣
 ↪  0.789 &      0.924 &      0.905 &      0.857 &      0.946 &     0.944 &   ␣
 ↪ 0.812 &     0.757 &     0.921 &     0.796 &     0.821 &    0.973 &     0.
 ↪823 &           0.751 &      0.707 &      0.828 &      0.810 &      0.956 & ␣
 ↪   0.992 &     0.860'.replace(' ', '').split('&')
aucs = list(map(float, aucs))
class_auc_map = dict(zip(classes, aucs))
sorted_classes = sorted(class_auc_map.items(), key=lambda x: -x[1]) #sort by␣
 ↪auc score
label_index_order = [dfl.columns[1:].get_loc(s[0]) for s in sorted_classes]
all_labels = [l['labels'][1] for l in latent_list]
fig, axs = plt.subplots(8, 8, dpi=1200, sharex='all', gridspec_kw={'wspace':0,␣
 ↪'hspace':0})
plt.axis('on')
color_norm = matplotlib.colors.Normalize(0.5, 1.0)
cmap = sns.color_palette("magma", as_cmap=True)
#fig.suptitle('t-SNE context embedding for all classes sorted by occurences in␣
 ↪data')
for i in range(8):
    for j in range(8):
        ax = axs[i, j]
        ax.set_yticklabels([])
        ax.set_xticklabels([])
        ax.tick_params(axis='both', which='both', bottom=False, top=False,␣
 ↪left=False, right=False, labelbottom=False)
        n = i*8+j
```

```
        class_name = dfl.columns[1:][label_index_order[n]]
        labels = get_label_for_label_index(all_labels, label_index_order[n],␣
↪repeat=1)
        ax.scatter(C_embedded[:,0], C_embedded[:,1], c=labels, s=(labels+1),␣
↪linewidths=0, marker='.', label=class_name)
        ax.set_facecolor(cmap(color_norm(class_auc_map[class_name]), alpha=0.7))
        ax.legend(prop={'size':2}, loc=1, bbox_to_anchor=(1.02, 1.02),␣
↪frameon=False)

cax,kw = mpl.colorbar.make_axes([ax for ax in axs.flat], aspect=40, pad=0.0)
cbar = plt.colorbar(mpl.cm.ScalarMappable(norm=color_norm, cmap=cmap), cax=cax,␣
↪**kw)
cbar.set_label('AUC score for given class', rotation=90, fontsize='xx-small')
cbar.ax.tick_params(labelsize=5)

#fig.tight_layout()
fig.savefig('/home/julian/Downloads/tsne-context-all(ordered by auc).png',␣
↪dpi=fig.dpi, bbox_inches='tight')
```



0.9 Umap

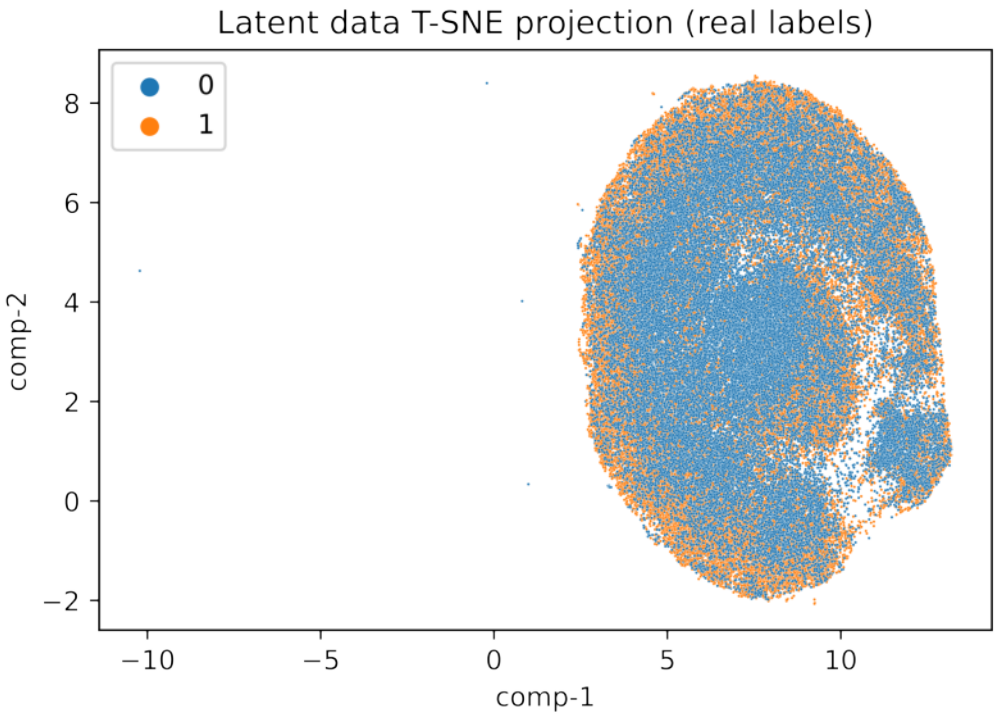
[15]:
```
import umap
from sklearn.preprocessing import StandardScaler
```

[2]:
```
reducer = umap.UMAP()
```

[16]:
```
L_embedded = reducer.fit_transform(StandardScaler().fit_transform(all_latents))
```

[18]:
```
normal_class_ix = 41
labels = [item for sublist in [[1]*9 if normal_class_ix in l['labels'][1] else␣
 ↪[0]*9 for l in latent_list] for item in sublist]
df = pd.DataFrame()
df["y"] = labels
df["comp-1"] = L_embedded[:,0]
df["comp-2"] = L_embedded[:,1]
plt.figure(dpi=1200)
sns.scatterplot(x="comp-1", y="comp-2", hue=df.y.tolist(),
                data=df, s=1).set(title="Latent data UMAP projection (real␣
 ↪labels)")
```

[18]:
```
[Text(0.5, 1.0, 'Latent data T-SNE projection (real labels)')]
```



[23]:
```
def get_label_for_label_index(all_labels, index, repeat=1):
    labels = [item for sublist in [[1]*repeat if index in l['labels'][1] else␣
 ↪[0]*repeat for l in latent_list] for item in sublist]
    return labels
all_labels = [l['labels'][1] for l in latent_list]
```

```
fig, axs = plt.subplots(8, 8, dpi=3200, sharex='all', gridspec_kw={'wspace':0,␣
↪'hspace':0})
cmap = matplotlib.colors.ListedColormap(['red' ,'black'])
label_index_order = np.argsort(-1*np.sum(LABELS, axis=0))
plt.axis('on')
for i in range(8):
    for j in range(8):
        ax = axs[i, j]
        ax.set_yticklabels([])
        ax.set_xticklabels([])
        ax.tick_params(axis='both', which='both', bottom=False, top=False,␣
↪left=False, right=False, labelbottom=False)
        n = i*8+j
        labels = get_label_for_label_index(all_labels, label_index_order[n],␣
↪repeat=9)
        ax.scatter(L_embedded[:,0], L_embedded[:,1], c=labels, s=1,␣
↪linewidths=0, marker='.')
#fig.savefig('latent-tsne-all.png', dpi=fig.dpi)
```