Assignment 4

BS6207

ZHOU RUIMENG

First Assignment: https://github.com/cyberpunk-newman/Master/tree/main/BS6207

Second Assignment: https://github.com/cyberpunk-newman/Master/tree/main/BS6207/Assignment2

Third Assignment: https://github.com/cyberpunk-newman/Master/tree/main/BS6207/Assignment3

Fourth Assignment: https://github.com/cyberpunk-newman/Master/tree/main/BS6207/Assignment4

Q1

Dataset: MNIST

Declared: In the paper, the total iteration of the training part is 1,000,000 times and owing to the time limit and hardware limit, I changed it 2000 times so the accuracy would not be as high as Ucc original model.

For model 2, model 3, and model 4, the differences are in loss functions and their training bags. Model 1 and model 3 are trained on bags with labels of ucc1 to ucc4. Model 2 and model 4 are trained on bags with labels ucc2 to ucc4. These parts of code can be modified in train.py which is represented by ucc start and ucc end.

Meanwhile, for model 3 and model 4, these two models have no autoencoder branch and are optimized over ucc loss function only. Model 1 and model 2 are both optimized by ucc loss and autoencoder loss. In this paper, the 'autoencoder loss' is 'mean square error loss' and 'ucc loss' is 'cross-entropy loss'. Loss function can be changed in model.py

	Min JS divergence	Ucc acc	Clustering acc
UCC (model 1)	0.23	**0.9993**	**0.9773**
UCC^{2+} (model 2)	0.23	0.7503	0.7902
$UCC_{a=1}$ (model 3)	0.23	**0.7907**	**0.8874**

$UCC_{a=1}^{2+}$ (model 4)	0.23	0.7842	0.6916

Table 1. Test Result

From table 1 we can see that except UCC model, other models' Ucc accuracy and Clustering Accuracy is quite close, similar to the paper. The highest two models are model 1 and model 3. I think the reason is that these two models both use the training bag from ucc1 to ucc4. The difference is that Model 1 has two loss functions so the backwards have a better effect. With diverse training data, these two models surpass model 2 and model 4.

One thing that needs to be noticed is that the Min JS divergence are all 0.23, quite similar to the result on paper but I think in fact they have different but the results are set to two decimal places.

I used Ubuntu with virtual environment to run this code, and here is some of my running process.

```
(myvenv) zrm@LAPTOP-O6HL6CL8:/mnt/d/PG_study/BS6207/Assignment/4/mnist$ python3 train.py
CNN model parameters:
patch_size = 28
num_instances = 32
num_features = 10
batch_size = 20
num_samples_per_class = 5
num_classes = 4
subset_length = 10
ucc_start = 1
ucc_end = 4
learning_rate = 0.0001
num_steps = 1500
metrics_file = loss_data/step_loss_acc_metrics__2022_01_18__20_04_08__0123456789.txt
init_model_file = None
2022-01-18 20:04:08.958754: E tensorflow/stream_executor/cuda/cuda_driver.cc:271] failed call to cuInit: UNKNOWN ERROR (100)
2022-01-18 20:04:08.958815: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:156] kernel driver does not appear to be running on this host (LAPTOP-O6HL6CL8): /proc/driver/nvidia/versio
n does not exist
2022-01-18 20:04:08.959195: I tensorflow/core/platform/cpu_feature_guard.cc:151] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CP
U instructions in performance-critical operations:  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
x0_img_shape:[28, 28]
x1_img_shape:[28, 28]
x0 shape:(None, 28, 28, 16)
x1 shape:(None, 28, 28, 32)
x0_img_shape:[28, 28]
x1_img_shape:[14, 14]
x0 shape:(None, 28, 28, 32)
x1 shape:(None, 14, 14, 64)
x0_img_shape:[14, 14]
x1_img_shape:[7, 7]
x0 shape:(None, 14, 14, 64)
x1 shape:(None, 7, 7, 128)
flatten shape:(None, 6272)
x0 shape:(None, 14, 14, 128)
x1 shape:(None, 14, 14, 128)
x0 shape:(None, 28, 28, 128)
x1 shape:(None, 28, 28, 64)
x0 shape:(None, 28, 28, 64)
x1 shape:(None, 28, 28, 32)
reconstructed shape:(None, 28, 28, 1)
concatenated shape:(None, 32, 10)
ae_concatenated shape:(None, 32, 28, 28, 1)
y shape:(20, 110)
/mnt/d/PG_study/BS6207/Assignment/4/myvenv/lib/python3.8/site-packages/keras/optimizer_v2/adam.py:105: UserWarning: The `lr` argument is deprecated, use `learning_rate` instead.
  super(Adam, self).__init__(name, **kwargs)
/mnt/d/PG_study/BS6207/Assignment/4/myvenv/lib/python3.8/site-packages/keras/engine/functional.py:1410: CustomMaskWarning: Custom mask layers require a config and must override get_config.
When loading, the custom mask layer must be passed to the custom_objects argument.
  layer_config = serialize_layer_fn(layer)
x_train_shape: (50000, 28, 28, 1)
```

Figure 1. training part

```
10000 val samples
digit0:0, num_train:4936, num_val:987
digit1:1, num_train:5619, num_val:1123
digit2:2, num_train:4965, num_val:993
digit3:3, num_train:5110, num_val:1021
digit4:4, num_train:4869, num_val:973
digit5:5, num_train:4514, num_val:907
digit6:6, num_train:4932, num_val:986
digit7:7, num_train:5221, num_val:1044
digit8:8, num_train:4876, num_val:975
digit9:9, num_train:4958, num_val:991
Subset=0123456789, Step=0 ### training: weighted_loss=1.488, ucc_acc=0.250, ucc_loss=1.389, ae_loss=1.588 ### validation: weighted_loss=1.244, ucc_acc=0.250, ucc_loss=1.389, ae_loss=1.100
Subset=0123456789, Step=10 ### training: weighted_loss=1.157, ucc_acc=0.250, ucc_loss=1.386, ae_loss=0.927 ### validation: weighted_loss=1.151, ucc_acc=0.200, ucc_loss=1.387, ae_loss=0.915
Subset=0123456789, Step=20 ### training: weighted_loss=1.099, ucc_acc=0.250, ucc_loss=1.388, ae_loss=0.810 ### validation: weighted_loss=1.095, ucc_acc=0.250, ucc_loss=1.388, ae_loss=0.802
Subset=0123456789, Step=30 ### training: weighted_loss=1.061, ucc_acc=0.250, ucc_loss=1.387, ae_loss=0.735 ### validation: weighted_loss=1.057, ucc_acc=0.250, ucc_loss=1.388, ae_loss=0.726
Subset=0123456789, Step=40 ### training: weighted_loss=1.050, ucc_acc=0.250, ucc_loss=1.387, ae_loss=0.713 ### validation: weighted_loss=1.051, ucc_acc=0.250, ucc_loss=1.386, ae_loss=0.717
Subset=0123456789, Step=50 ### training: weighted_loss=1.044, ucc_acc=0.350, ucc_loss=1.386, ae_loss=0.703 ### validation: weighted_loss=1.048, ucc_acc=0.250, ucc_loss=1.384, ae_loss=0.711
Subset=0123456789, Step=60 ### training: weighted_loss=1.041, ucc_acc=0.300, ucc_loss=1.386, ae_loss=0.697 ### validation: weighted_loss=1.045, ucc_acc=0.300, ucc_loss=1.385, ae_loss=0.704
Subset=0123456789, Step=70 ### training: weighted_loss=1.046, ucc_acc=0.300, ucc_loss=1.384, ae_loss=0.707 ### validation: weighted_loss=1.045, ucc_acc=0.250, ucc_loss=1.385, ae_loss=0.704
Subset=0123456789, Step=80 ### training: weighted_loss=1.047, ucc_acc=0.300, ucc_loss=1.385, ae_loss=0.708 ### validation: weighted_loss=1.041, ucc_acc=0.200, ucc_loss=1.386, ae_loss=0.697
Subset=0123456789, Step=90 ### training: weighted_loss=1.039, ucc_acc=0.200, ucc_loss=1.387, ae_loss=0.691 ### validation: weighted_loss=1.040, ucc_acc=0.350, ucc_loss=1.382, ae_loss=0.698
Model weights saved in file:  saved_models/model_weights__2022_01_18__20_04_08__0123456789__100.h5
Subset=0123456789, Step=100 ### training: weighted_loss=1.042, ucc_acc=0.400, ucc_loss=1.381, ae_loss=0.703 ### validation: weighted_loss=1.039, ucc_acc=0.350, ucc_loss=1.382, ae_loss=0.697
Subset=0123456789, Step=110 ### training: weighted_loss=1.037, ucc_acc=0.200, ucc_loss=1.385, ae_loss=0.690 ### validation: weighted_loss=1.034, ucc_acc=0.350, ucc_loss=1.381, ae_loss=0.688
Subset=0123456789, Step=120 ### training: weighted_loss=1.035, ucc_acc=0.350, ucc_loss=1.384, ae_loss=0.685 ### validation: weighted_loss=1.031, ucc_acc=0.350, ucc_loss=1.378, ae_loss=0.685
Subset=0123456789, Step=130 ### training: weighted_loss=1.025, ucc_acc=0.400, ucc_loss=1.374, ae_loss=0.675 ### validation: weighted_loss=1.032, ucc_acc=0.500, ucc_loss=1.375, ae_loss=0.689
Subset=0123456789, Step=140 ### training: weighted_loss=1.029, ucc_acc=0.550, ucc_loss=1.375, ae_loss=0.682 ### validation: weighted_loss=1.019, ucc_acc=0.450, ucc_loss=1.375, ae_loss=0.664
Subset=0123456789, Step=150 ### training: weighted_loss=1.016, ucc_acc=0.400, ucc_loss=1.377, ae_loss=0.655 ### validation: weighted_loss=1.012, ucc_acc=0.400, ucc_loss=1.373, ae_loss=0.652
Subset=0123456789, Step=160 ### training: weighted_loss=1.001, ucc_acc=0.350, ucc_loss=1.364, ae_loss=0.638 ### validation: weighted_loss=1.007, ucc_acc=0.400, ucc_loss=1.368, ae_loss=0.646
Subset=0123456789, Step=170 ### training: weighted_loss=0.994, ucc_acc=0.400, ucc_loss=1.365, ae_loss=0.622 ### validation: weighted_loss=0.993, ucc_acc=0.450, ucc_loss=1.363, ae_loss=0.623
Subset=0123456789, Step=180 ### training: weighted_loss=0.983, ucc_acc=0.550, ucc_loss=1.365, ae_loss=0.601 ### validation: weighted_loss=0.982, ucc_acc=0.350, ucc_loss=1.363, ae_loss=0.601
Subset=0123456789, Step=190 ### training: weighted_loss=0.978, ucc_acc=0.300, ucc_loss=1.363, ae_loss=0.593 ### validation: weighted_loss=0.976, ucc_acc=0.350, ucc_loss=1.363, ae_loss=0.589
Model weights saved in file:  saved_models/model_weights__2022_01_18__20_04_08__0123456789__200.h5
Subset=0123456789, Step=200 ### training: weighted_loss=0.960, ucc_acc=0.450, ucc_loss=1.346, ae_loss=0.574 ### validation: weighted_loss=0.961, ucc_acc=0.400, ucc_loss=1.353, ae_loss=0.570
Subset=0123456789, Step=210 ### training: weighted_loss=0.954, ucc_acc=0.400, ucc_loss=1.356, ae_loss=0.551 ### validation: weighted_loss=0.953, ucc_acc=0.450, ucc_loss=1.349, ae_loss=0.558
Subset=0123456789, Step=220 ### training: weighted_loss=0.954, ucc_acc=0.350, ucc_loss=1.366, ae_loss=0.542 ### validation: weighted_loss=0.957, ucc_acc=0.300, ucc_loss=1.371, ae_loss=0.543
Subset=0123456789, Step=230 ### training: weighted_loss=0.940, ucc_acc=0.550, ucc_loss=1.342, ae_loss=0.538 ### validation: weighted_loss=0.937, ucc_acc=0.300, ucc_loss=1.339, ae_loss=0.535
Subset=0123456789, Step=240 ### training: weighted_loss=0.932, ucc_acc=0.400, ucc_loss=1.342, ae_loss=0.521 ### validation: weighted_loss=0.934, ucc_acc=0.550, ucc_loss=1.337, ae_loss=0.531
Subset=0123456789, Step=250 ### training: weighted_loss=0.923, ucc_acc=0.350, ucc_loss=1.320, ae_loss=0.525 ### validation: weighted_loss=0.923, ucc_acc=0.500, ucc_loss=1.325, ae_loss=0.520
Subset=0123456789, Step=260 ### training: weighted_loss=0.911, ucc_acc=0.500, ucc_loss=1.317, ae_loss=0.505 ### validation: weighted_loss=0.929, ucc_acc=0.350, ucc_loss=1.343, ae_loss=0.515
Subset=0123456789, Step=270 ### training: weighted_loss=0.916, ucc_acc=0.350, ucc_loss=1.313, ae_loss=0.518 ### validation: weighted_loss=0.897, ucc_acc=0.550, ucc_loss=1.297, ae_loss=0.498
Subset=0123456789, Step=280 ### training: weighted_loss=0.883, ucc_acc=0.600, ucc_loss=1.272, ae_loss=0.494 ### validation: weighted_loss=0.906, ucc_acc=0.450, ucc_loss=1.316, ae_loss=0.495
Subset=0123456789, Step=290 ### training: weighted_loss=0.903, ucc_acc=0.450, ucc_loss=1.310, ae_loss=0.497 ### validation: weighted_loss=0.899, ucc_acc=0.350, ucc_loss=1.292, ae_loss=0.505
Model weights saved in file:  saved_models/model_weights__2022_01_18__20_04_08__0123456789__300.h5
Subset=0123456789, Step=300 ### training: weighted_loss=0.903, ucc_acc=0.350, ucc_loss=1.302, ae_loss=0.503 ### validation: weighted_loss=0.898, ucc_acc=0.400, ucc_loss=1.306, ae_loss=0.490
Subset=0123456789, Step=310 ### training: weighted_loss=0.877, ucc_acc=0.450, ucc_loss=1.273, ae_loss=0.481 ### validation: weighted_loss=0.872, ucc_acc=0.500, ucc_loss=1.250, ae_loss=0.493
Subset=0123456789, Step=320 ### training: weighted_loss=0.882, ucc_acc=0.400, ucc_loss=1.286, ae_loss=0.478 ### validation: weighted_loss=0.886, ucc_acc=0.450, ucc_loss=1.284, ae_loss=0.488
Subset=0123456789, Step=330 ### training: weighted_loss=0.870, ucc_acc=0.450, ucc_loss=1.252, ae_loss=0.487 ### validation: weighted_loss=0.833, ucc_acc=0.650, ucc_loss=1.205, ae_loss=0.460
Subset=0123456789, Step=340 ### training: weighted_loss=0.861, ucc_acc=0.250, ucc_loss=1.249, ae_loss=0.473 ### validation: weighted_loss=0.860, ucc_acc=0.400, ucc_loss=1.245, ae_loss=0.474
Subset=0123456789, Step=350 ### training: weighted_loss=0.852, ucc_acc=0.500, ucc_loss=1.227, ae_loss=0.476 ### validation: weighted_loss=0.836, ucc_acc=0.550, ucc_loss=1.206, ae_loss=0.465
```

Figure 2. training part

```
(myvenv) zrm@LAPTOP-O6HL6CL8:/mnt/d/PG_study/BS6207/Assignment/4/mnist$ ./evaluate_model.sh
2019_09_05__18_43_15__128000
test.py
__2019_09_05__18_43_15__0123456789__128000
CNN model parameters:
batch_size = 28
subset_length = 10
num_classes = 4
batch_size = 40
2022-01-18 21:55:00.424241: E tensorflow/stream_executor/cuda/cuda_driver.cc:271] failed call to cuInit: UNKNOWN ERROR (100)
2022-01-18 21:55:00.424293: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:156] kernel driver does not appear to be running on this host (LAPTOP-O6HL6CL8): /proc/driver/nvidia/versio
n does not exist
2022-01-18 21:55:00.424684: I tensorflow/core/platform/cpu_feature_guard.cc:151] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CP
U instructions in performance-critical operations:  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
x0_img_shape:[28, 28]
x1_img_shape:[28, 28]
x0 shape:(None, 28, 28, 16)
x1 shape:(None, 28, 28, 32)
x0_img_shape:[28, 28]
x1_img_shape:[14, 14]
x0 shape:(None, 28, 28, 32)
x1 shape:(None, 14, 14, 64)
x0_img_shape:[14, 14]
x1_img_shape:[7, 7]
x0 shape:(None, 14, 14, 64)
x1 shape:(None, 7, 7, 128)
flatten shape:(None, 6272)
x0 shape:(None, 14, 14, 128)
x1 shape:(None, 14, 14, 128)
x0 shape:(None, 28, 28, 128)
x1 shape:(None, 28, 28, 64)
x0 shape:(None, 28, 28, 64)
x1 shape:(None, 28, 28, 32)
reconstructed shape:(None, 28, 28, 1)
concatenated shape:(None, 32, 10)
ae_concatenated shape:(None, 32, 28, 28, 1)
y shape:(40, 110)
/mnt/d/PG_study/BS6207/Assignment/4/myvenv/lib/python3.8/site-packages/keras/optimizer_v2/adam.py:105: UserWarning: The `lr` argument is deprecated, use `learning_rate` instead.
  super(Adam, self).__init__(name, **kwargs)
weights loaded successfully!!!
model_weights_filename: saved_models/2019_09_05__18_43_15__128000/model_weights__2019_09_05__18_43_15__0123456789__128000.h5
x_test shape: (10000, 28, 28, 1)
10000 test samples
y_test shape: (10000,)
digit0:0, num_test:980
digit1:1, num_test:1135
digit2:2, num_test:1032
digit3:3, num_test:1010
digit4:4, num_test:982
```

Figure 3. test part

1. I tried different feature extractors and get some results. The result of Sigmoid is bad.

	Min JS divergence	Ucc acc	Clustering acc
$UCC_{a=1}^{2+}$ (model 4)	0.23	0.7842	0.6916
Sigmoid	0.23	0.3775	0.3007

Table2. Different Feature Extractors

2. Also here is my result of training different sizes of data on Ucc model.

Training size	Ucc acc	Clustering acc
60,000	0.7984	0.7486
20,000	0.5684	0.4860
5,000	0.5038	0.5203
500	ValueError: cannot get result of training and testing	

Table3. Different Training Size

3. I think one way to improve this algorithm is to add or remove some convolutional or pooling layers to better extract features and get a good result. Also, we can use a larger different kind of dataset to raises the ability to extend of BP neural network.