AAI Project: Train classifier for modified MNIST with unstable (spurious) features

This project requires training an image classification model from scratch to classify a preprocessed MNIST dataset.

This project will be conducted in teams, **allowing for groups of 2-4 people**. The size of the team will have an impact on the final evaluation. The deadline for forming teams is **2023/12/11**. Students who do not form teams by the deadline will be randomly grouped into teams of four.

The DDL of this project is 2024/01/07.

Dataset:

Description

The MNIST dataset has a training set of 60,000 samples and a test set of 10,000 samples. For this project, we have added some unstable (spurious) features to the original MNIST dataset. We need you to train a classifier that can filter out these spurious features to create a stable and robust classifier.

Directory Structure

The data directory structure is shown below.

```
.
├── test
|   ├── 0.npy
|   ├── ...
├── train
|   ├── 0
|   |   ├── 0.npy
|   |   ├── ...
|   ├── 1
|   |   ├── 1.npy
|   |   ├── ...
|   ├── ...
├── val
|   ├── 0
|   |   ├── 0.npy
|   |   ├── ...
|   ├── 1
|   |   ├── 1.npy
|   |   ├── ...
|   ├── ...
```

The train directory, which contains the training set, has two levels: label and sample. The directories within train represent different labels, and their names should be used as prediction labels. All the samples belonging to the same label are corresponding training samples. The validation directory has similar structure with the train directory and the test directory has samples without labels.

You should manually separate the dataset into training and validation subsets. Since the distribution of spurious features is different in the training and test subsets, if your model fails to filter them out, the accuracy will be very poor.

Download

The dataset can be downloaded from Blackboard. The access requires your SUSTech identity.

Submission

There are three items you should submit: 1) prediction results from your model, 2) the source code used for training and inference, and 3) a report. Below are the details:

Prediction Results

After training your model, you need to generate labels on the test set. The results should be stored in a .txt file, and the content should look like

```
0.npy 0
1.npy 1
...
```

Source Code

The whole process will involve data pre-processing, model training, and prediction. You will definitely have to pre-process the data on your own. You are encouraged to have your own model implementation, at least an interface combining open resources. All the code should be packed and submitted. Probably you will use Python, and you can attach a simple environment configuration file, e.g., requirement.txt.

Report

Several key elements should be included in your report: 1) model description, 2) experimental details, 3) the training records, 4) member contribution. In the model description, you have to describe the model structure that you use. For example, the model can be a combination of CNNs, Transformer layers, and a pooling head. The computation process should be explained. For the experimental details, you should mention how you implement your model, what features your model takes, the open resources that you use, and the hyperparameters including the learning rate, model size, etc. As for the training records, you can provide the loss curves on the training and validation sets. If you improve your model gradually, you can give further explanations. Finally, the project should be finished in individual.

Scoring Criterion

The evaluation of projects in this course will consider accuracy on the test set, code readability, as well as the ability to execute, and report for an overall assessment.

Please note that the number of team members will affect the overall score. The final score for a team of two will be the standard score multiplied by 1.1, the final score for a team of three will remain unchanged, and the final score for a team of four will be the standard score multiplied by 0.9.

References

[1] Bao, Yujia, Shiyu Chang, and Regina Barzilay. "Learning stable classifiers by transferring unstable features." International Conference on Machine Learning. PMLR, 2022.

[2] Arjovsky, Martin, et al. "Invariant risk minimization." arXiv preprint arXiv:1907.02893 (2019).

[3] LeCun, Yann, et al. "Gradient-based learning applied to document recognition." Proceedings of the IEEE 86.11 (1998): 2278-2324.