Artificial Intelligence(STA303) Assignment 2 Report

12110208 宋子烨

Test code

We use Exercise_03.ipynb as a template and finish all the questions to accomplish the assignment.We will use distinct datesets to check the performance of CLIP. The code is omited. You can check them with opening file Assignment2.ipynb.
Another significance I want to emphasize is that we choose VGG, resnet50 as the baseline model. For different datasets we modify last classification layer to match the classes.

Dataset1

Introduction about the dataset1

We choose a picture dataset based on road pothole identification. The raw pictures are preserved in path"./data/data". You can check first. There are two folders which classify the pictures through the only difference: whether the road picture has pothole or not.
There are totally 301 pictures, which constitute a small dataset. It is definitely a binary classification problem and has 266 normal pictures and 35 pothole pictures.the proportion is almost 8:1. we demonstrate the loading dataset code below.

```
def pothole_dataset():# define transform
    transform = transforms.Compose([
        transforms.ToTensor(),
        transforms.Resize((224,224),antialias=True),
    ])

    # loading dataset
    dataset = ImageFolder(root='data/data', transform=transform)
    test_dataloader = torch.utils.data.DataLoader(dataset, batch_size=BATCH_SIZE,
shuffle=True, num_workers=2)
    class_names = ['normal', 'pothole']
    dataset_name = 'Pothole identification'
    return dataset,test_dataloader,class_names,dataset_name
```

Output

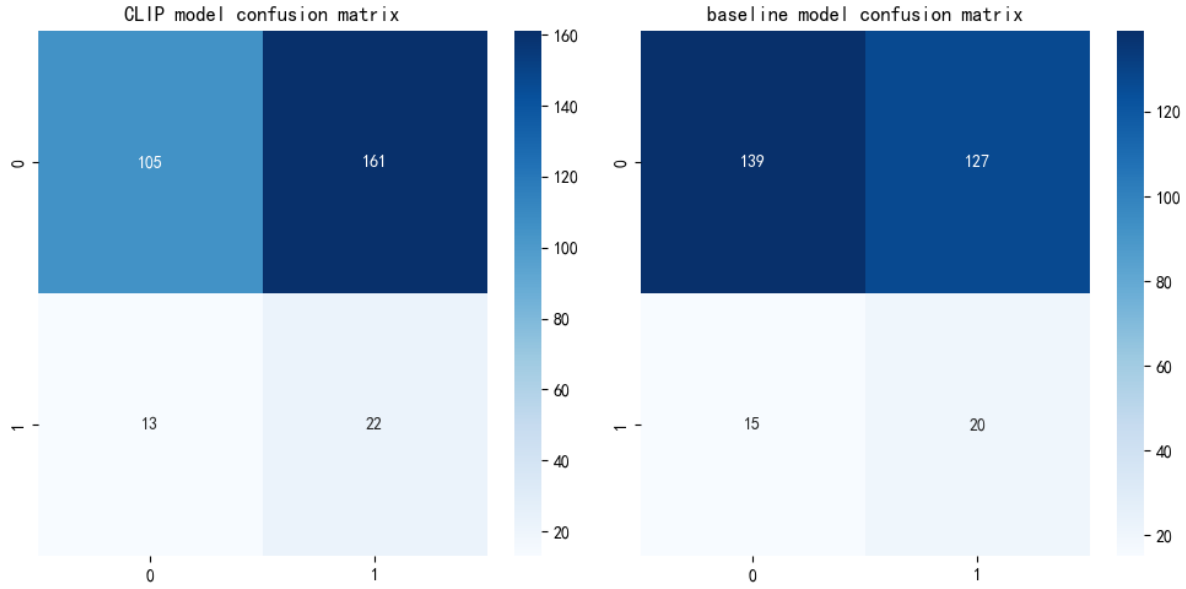
The accuracies of these two models is as below:

```
the zero-shot performance on Pothole identification is 50.83%, visual encoder is ViT-B/16.
```

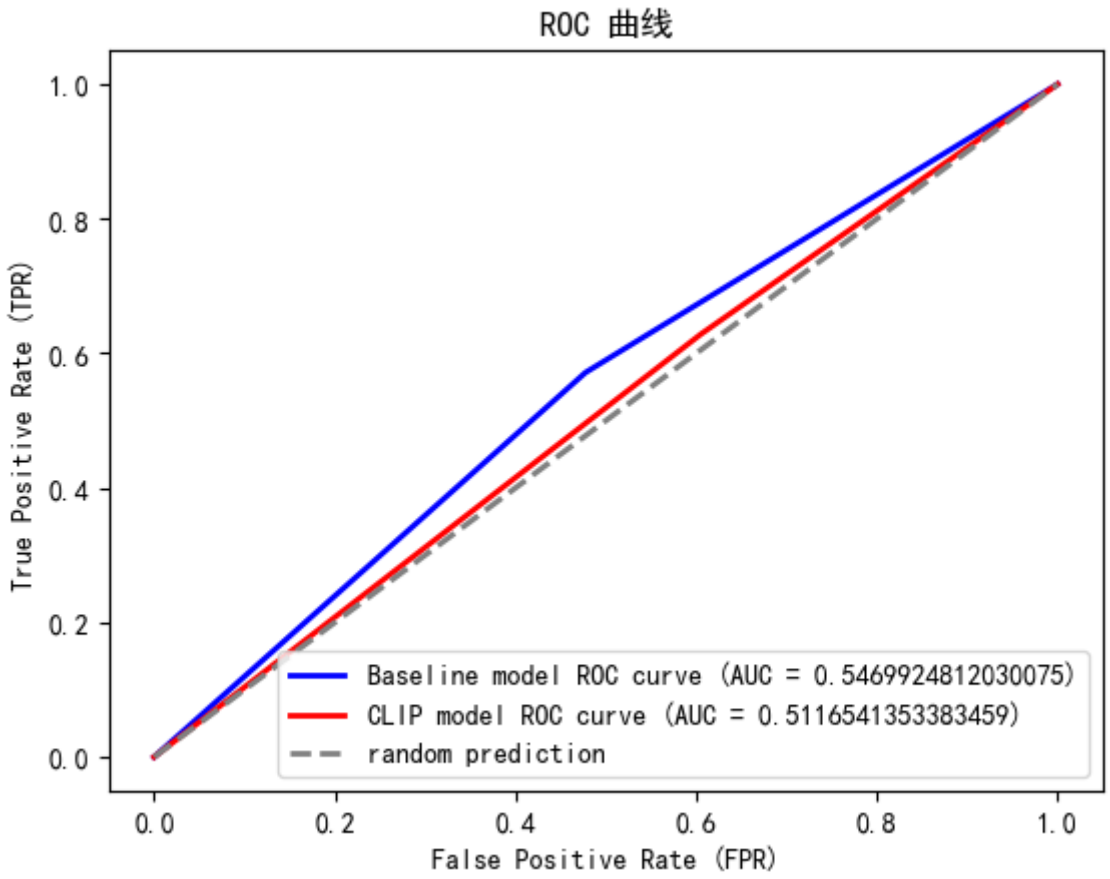
```
Baseline model Accuracy: 55.48%
```

We choose the best visual encoder "ViT-B/16" to make the accuracy to be the highest.

We also derive the confusion matrix and ROC curve to check the output.
The confusion matrix:



The ROC curve:



Brief Conclusion

Through these outputs we can conclude that there are few difference between CLIP and other baseline model (like `resnet50`, `VGG`, pre-trained model).

My assumption is that due to the small sample size, it's unable to demonstrate the difference between the 'clip' model and other baseline models.

Another point which is deserved to be discussed is that when performing zero-shot learning with `resNet50`, the accuracy results can vary across different runs.

I consider there are a few reasons:

- *Randomness*: The models has stochastic elements during training, such as weight initialization or data augmentation, which can cause slight variations in performance with each training run.
- *Data Distribution*: Zero-shot learning typically relies on generalizing the model to previously unseen classes. Variations in the test sample distribution across different runs may result in different outcomes.
- *The last classification layer*: In this case, we shrink the last layer to match the binary classification problem. However, we have no idea about whether the model recognize the picture or not. We just let the higher one to be output.

dataset2

introduction about the dataset2

The CIFAR-10 dataset consists of 60,000 32x32 color images in 10 classes, with each class containing 6,000 images. These images are evenly distributed across the ten categories, including common objects such as airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks.

Each image in CIFAR-10 has a resolution of 32x32 pixels, which is relatively small compared to many other image datasets. This dataset's small image size and diverse classes make it a standard benchmark for evaluating image classification algorithms and models.

The CIFAR-10 dataset's modest image resolution and variety of objects present challenges for machine learning models to learn and generalize from the provided images. Its balanced distribution of classes contributes to its usefulness in assessing the performance and robustness of classification algorithms across different categories.

we demonstrate the loading dataset code below.

```
def CIFAR10_dataset():
    transform_cifar10_test = transforms.Compose([
        transforms.Resize(size=224),
        transforms.CenterCrop(size=(224, 224)),
        transforms.ToTensor(),
        transforms.Normalize((0.4914, 0.4822, 0.4465), (0.2023, 0.1994, 0.2010)),
    ])

    test_set = torchvision.datasets.CIFAR10(root='/shareddata', train=False,
                                            download=True,
transform=transform_cifar10_test)
    test_dataloader = torch.utils.data.DataLoader(test_set,
batch_size=BATCH_SIZE,
                                              shuffle=True, num_workers=2)

    class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog',
'frog', 'horse', 'ship', 'truck']
    dataset_name = 'CIFAR10'
    return test_set,test_dataloader,class_names,dataset_name
```

Output

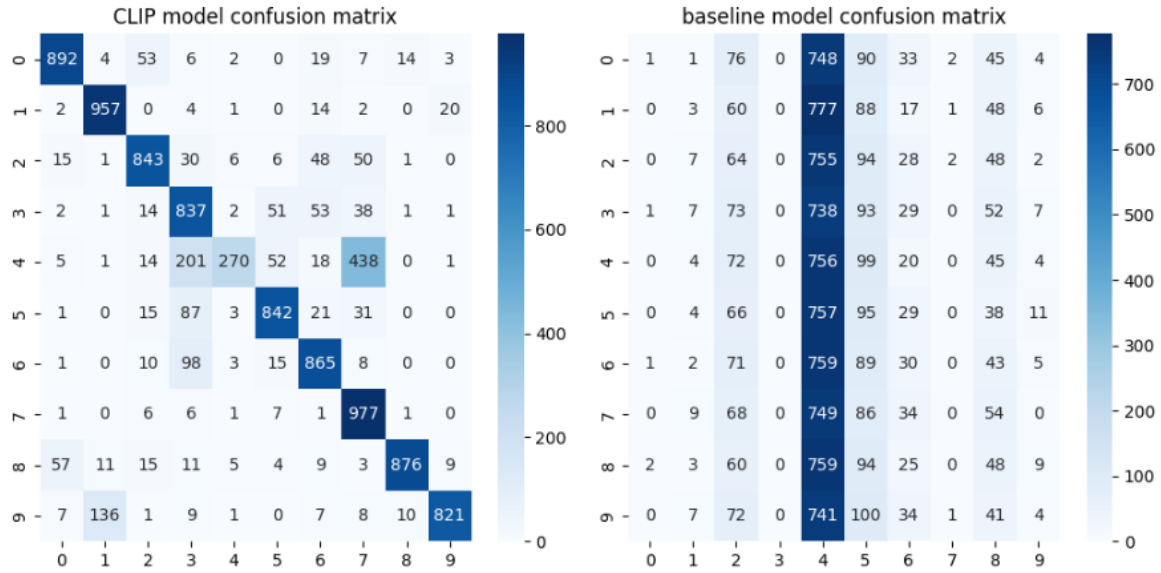
The accuracies of these two models is as below:

```
the zero-shot performance on CIFAR10 is 81.80%, visual encoder is ViT-B/16.

Baseline model Accuracy: 11.41%
```

We choose the best visual encoder "ViT-B/16" to make the accuracy to be the highest.

We also derive the confusion matrix to check the output.
The confusion matrix:



Brief Conclusion

It is hard to believe that CLIP make what a great prediction on CIFAR10 dataset, and resnet50 only predicts almost 10%. Check the confusion matrix we can find that our baseline model classify a large quantity of label 4.

My assumption is since the low picture quality, ResNet50_Weights.IMAGENET1K_V1 may be difficult to recognize this pictures. As for high accuracy with CLIP, maybe CLIP use CIFAR10 as the trainset. As a result, CLIP show a great performance on the CIFAR10 dataset.

dataset3

Introduction about the dataset3

This time we try DTD dataset, which is available in the folder `shareddata`. The Describable Textures Dataset (DTD) comprises around 5,640 images, covering 47 distinct texture categories. The dataset showcases a diverse array of textures encompassing various materials such as burlap, corduroy, cracked surfaces, furry textures, and more. DTD images exhibit considerable variation in scale, lighting, and appearance, offering a broad spectrum of texture appearances for analysis. The dataset's richness and diversity make it a valuable resource for texture recognition tasks, providing ample opportunities to evaluate algorithms' abilities to distinguish and classify a wide range of texture patterns. ***After my observation, the classification job is extremely difficult, since it seems like the AI need a few strategies in disputing and recognizing texture categories.***

we demonstrate the loading dataset code below.

```
def DTD_dataset():
    transform = transforms.Compose([
        transforms.ToTensor(),
        transforms.Resize((224,224),antialias=True),
    ])
    test_set = torchvision.datasets.DTD(root='/shareddata',
                                        download=True, transform=transform)
    test_dataloader = torch.utils.data.DataLoader(test_set,
batch_size=BATCH_SIZE,
                                          shuffle=True, num_workers=47)
    class_names = ["banded", "blotchy", "braided", "bubbly", "bumpy",
"chequered", "cobwebbed", "cracked", "crosshatched", "crystalline", "dotted",
"fibrous", "flecked", "freckled", "frilly", "gauzy", "grid", "grooved",
"honeycombed", "interlaced", "knitted", "lacelike", "lined", "marbled", "matted",
"meshed", "paisley", "perforated", "pitted", "pleated", "polka-dotted", "porous",
"potholed", "scaly", "smeared", "spiralled", "sprinkled", "stained",
"stratified", "striped", "studded", "swirly", "veined", "waffled", "woven",
"wrinkled", "zigzagged"]
    dataset_name = 'DTD'
    return test_set, test_dataloader,class_names,dataset_name
```

Output

The accuracies of these two models is as below:

```
the zero-shot performance on DTD is 11.81%, visual encoder is ViT-B/16.

Baseline model Accuracy: 1.70%
```

since we got too many classes, it is hard for us to make the confusion matrix clear to check. As a result, we omit this procedure. You can check it in the code.

Brief Conclusion

Through these outputs we can conclude that there are few difference between CLIP and other baseline model (like `resnet50`, `VGG`, pre-trained model).

My assumption is that due to the large sample size and so many classes, it's unable for two models to make a comparably correct classification.

I consider if we have a dataset in a large scale, it is better to use CLIP to make a prediction. The output by baseline model is awful when there are so many classes and difficulties in classification.

By the way, the running time of both CLIP and baseline model is very long.

Summary

- Binary Classification:

CLIP, known for its ability to understand images and text jointly, performs well in binary classification tasks. Leveraging its pre-training on large-scale data, CLIP demonstrates robustness and generalization capabilities. By utilizing a prompt, users can fine-tune CLIP for specific binary classification tasks, achieving high accuracy and reliable performance.

- Multi-Class Classification:

In multi-class classification, CLIP showcases promising results across various domains. Its ability to associate images and text across a wide range of classes enables effective classification in scenarios with multiple categories. Despite its high performance, fine-tuning CLIP for multi-class tasks might demand a well-structured prompt and dataset, along with careful tuning of hyperparameters to ensure optimal performance across diverse classes.

In both binary and multi-class classification scenarios, CLIP's strength lies in its capability to understand the semantic relationships between images and text. However, achieving optimal results often requires domain-specific fine-tuning, data preprocessing, and tailored prompts to harness CLIP's full potential for different classification tasks.

My suggestion is when we need to do a binary classification or a multi-classification with a few classes, for example 10, we can choose CLIP to make this job. However, when the number of classes is much higher, the effect of CLIP may be extremely low.

Another dificiency is ***Computational Resource Demands***. Regardless of the size of datasets, the time which is consumed by running CLIP is probably very long.

From my own perspective, after observing the performance in the three datasets, I conclude that if the pictures are clear enough and the classification job is not too strategic and difficult, the performance of CLIP may be acceptable.