Deep Learning HW2　112753204　陳品伃

Q1

Please download the dataset using the following link: https://drive.google.com/drive/folders/12J0JtSrqrHAjt2_olcB3tLVL6WIKlq5l?usp=sharing. The dataset includes two files: train.zip and test.zip. Utilize the training data to train two models, AlexNet and ResNet. After training, assess the performance of both models using the test data. Report the accuracy of the results obtained from testing. Additionally, please provide visualizations of the training loss changes for both models.

AlexNet

[Structure - parameters]

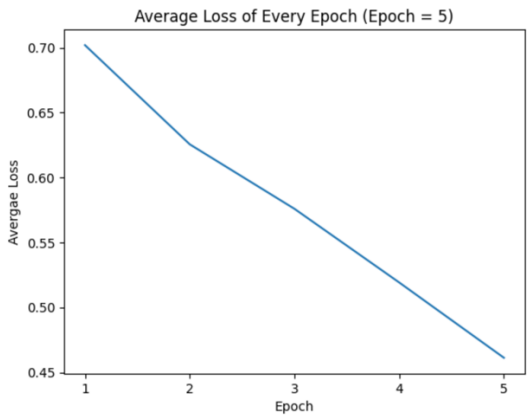
```python
# stride = 1(Default)
class AlexNet(nn.Module):
    def __init__(self, num_classes=2):
        super(AlexNet, self).__init__()
        self.features = nn.Sequential(
            nn.Conv2d(3, 32, kernel_size=3, padding=1),  # [32, 224, 224]
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=2),

            nn.Conv2d(32, 64, kernel_size=3, padding=1),  # [64, 112, 112]
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=2),
            nn.Dropout(0.5),

            nn.Conv2d(64, 64, kernel_size=3, padding=1),  # [64, 56, 56]
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=2),
        )
        # [64, 28, 28]
        self.classifier = nn.Sequential(
            nn.Linear(64 * 28 * 28, 128),
            nn.ReLU(inplace=True),
            nn.Dropout(0.5),
            nn.Linear(128, num_classes),
        )
```

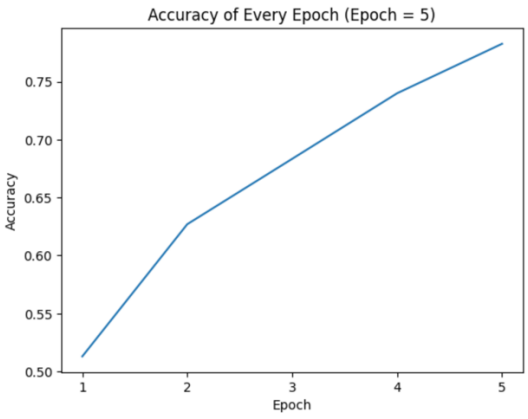
[Testing Accuracy]

```
100%|██████████| 79/79 [00:35<00:00,  2.24it/s]
Accuracy: 79.68%
```

[Training Loss & Accuracy]

```
100%|██████████| 313/313 [05:07<00:00,  1.02it/s]
Epoch 1/5, loss = 0.70303, acc = 0.51073
100%|██████████| 313/313 [05:07<00:00,  1.02it/s]
Epoch 2/5, loss = 0.65476, acc = 0.58606
100%|██████████| 313/313 [05:12<00:00,  1.00it/s]
Epoch 3/5, loss = 0.57976, acc = 0.68331
100%|██████████| 313/313 [05:16<00:00,  1.01s/it]
Epoch 4/5, loss = 0.51368, acc = 0.74775
100%|██████████| 313/313 [05:12<00:00,  1.00it/s]
Epoch 5/5, loss = 0.46618, acc = 0.78290
```





這份報告所有的Loss都是以一個Epoch為單位，在每一個Epoch內會去計算該回合內所有loss的平均。

ResNet

[Structure - parameters]

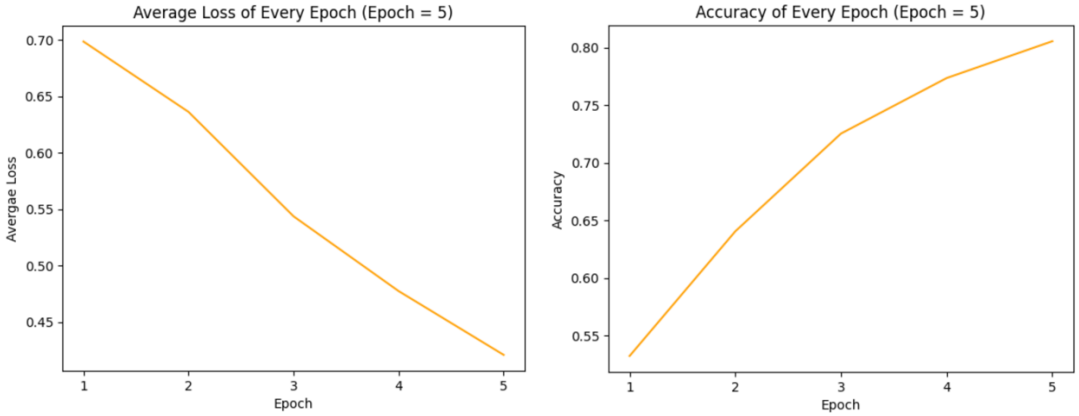
```
# stride = 1(Default)
class ReSNet(nn.Module):
    def __init__(self, num_classes=2):
        super(ReSNet, self).__init__()
        self.features = nn.Sequential(
            nn.Conv2d(3, 32, kernel_size=3, padding=1),  # [32, 224, 224]
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=2),

            nn.Conv2d(32, 64, kernel_size=3, padding=1),  # [64, 112, 112]
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=2),
            nn.Dropout(0.5),

            nn.Conv2d(64, 64, kernel_size=3, padding=1),  # [64, 56, 56]
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=2),
        )
        # [64, 28, 28]
        self.classifier = nn.Sequential(
            nn.Linear(64 * 28 * 28, 128),
            nn.ReLU(inplace=True),
            nn.Dropout(0.5),
            nn.Linear(128, num_classes),
        )
```

[Testing Accuracy]

```
100%|██████████| 79/79 [00:32<00:00,  2.47it/s]
Accuracy: 80.14%
```

[Training Loss & Accuracy]

```
100%|██████████| 313/313 [04:43<00:00,  1.11it/s]
Epoch 1/5, loss = 0.69848, acc = 0.53240
100%|██████████| 313/313 [04:27<00:00,  1.17it/s]
Epoch 2/5, loss = 0.63617, acc = 0.64053
100%|██████████| 313/313 [05:03<00:00,  1.03it/s]
Epoch 3/5, loss = 0.54367, acc = 0.72534
100%|██████████| 313/313 [04:54<00:00,  1.06it/s]
Epoch 4/5, loss = 0.47746, acc = 0.77351
100%|██████████| 313/313 [05:03<00:00,  1.03it/s]
Epoch 5/5, loss = 0.42095, acc = 0.80546
```



Q1 Summary

AlexNet與ResNet的效果從Loss以及Accuracy上看起來並沒有太大的差異，但此dataset在ResNet上擁有較好的表現，每個回合的Loss相較於AlexNet都較小，且每回合的Accuracy都較大。

Q2

Follow the format of Question 1 and utilize the training data to train two different models, including VGG and a CNN model you designed. After training, assess the performance of both models using the test data. Report the accuracy of the results obtained from testing. Additionally, please provide visualizations of the training loss changes for both models.

VGG

[Structure - parameters]

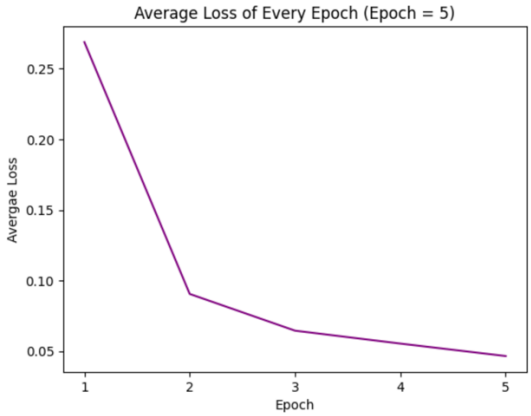
```python
# stride = 1(Default)
class VGG(nn.Module):
    def __init__(self, num_classes=2):
        super(VGG, self).__init__()
        self.features = nn.Sequential(
            nn.Conv2d(1, 32, kernel_size=3, padding=1),  # [32, 28, 28]
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=2),

            nn.Conv2d(32, 64, kernel_size=3, padding=1),  # [64, 14, 14]
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=2),
            nn.Dropout(0.5),

            nn.Conv2d(64, 128, kernel_size=3, padding=1),  # [128, 7, 7]
            nn.ReLU(inplace=True),
        )

        self.global_avg_pooling = nn.AdaptiveAvgPool2d((1, 1))

        self.classifier = nn.Sequential(
            nn.Linear(128 *7 *7 , 256),
            nn.ReLU(inplace=True),
            nn.Dropout(0.5),
            nn.Linear(256, num_classes),
        )

    def forward(self, x):
        x = self.features(x)
        x = x.view(x.size(0), -1)
        x = self.classifier(x)
        return x
```

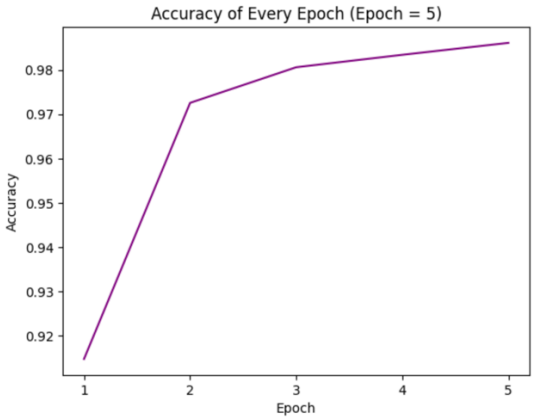
[Testing Accuracy]

```
100%|██████████| 79/79 [00:01<00:00, 52.52it/s]
Accuracy: 99.18%
```

[Training Loss & Accuracy]

```
100%|██████████| 469/469 [00:16<00:00, 28.51it/s]
Epoch 1/5, loss = 0.28156, acc = 0.91060
100%|██████████| 469/469 [00:16<00:00, 29.17it/s]
Epoch 2/5, loss = 0.08990, acc = 0.97325
100%|██████████| 469/469 [00:15<00:00, 29.64it/s]
Epoch 3/5, loss = 0.06463, acc = 0.98006
100%|██████████| 469/469 [00:15<00:00, 29.47it/s]
Epoch 4/5, loss = 0.05383, acc = 0.98374
100%|██████████| 469/469 [00:16<00:00, 29.27it/s]
Epoch 5/5, loss = 0.04835, acc = 0.98540
```





CNN

[Structure - parameters]

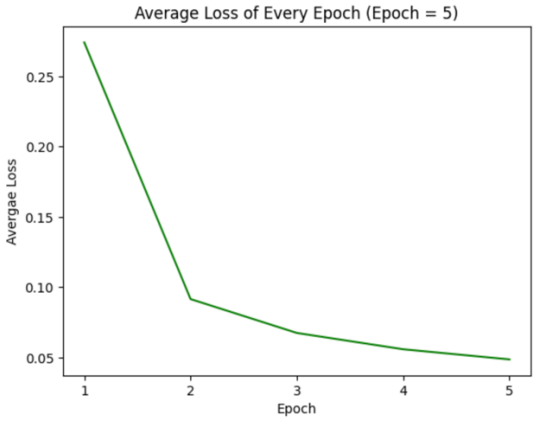
```python
# stride = 1(Default)
class CNN(nn.Module):
    def __init__(self, num_classes=2):
        super(CNN, self).__init__()
        self.features = nn.Sequential(
            nn.Conv2d(1, 32, kernel_size=3, padding=1),  # [32, 28, 28]
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=2),

            nn.Conv2d(32, 64, kernel_size=3, padding=1),  # [64, 14, 14]
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=2),
            nn.Dropout(0.5),

            nn.Conv2d(64, 128, kernel_size=3, padding=1),  # [128, 7, 7]
            nn.ReLU(inplace=True),
        )

        self.classifier = nn.Sequential(
            nn.Linear(128 * 7 * 7, 256),
            nn.ReLU(inplace=True),
            nn.Dropout(0.5),
            nn.Linear(256, num_classes),
        )
```

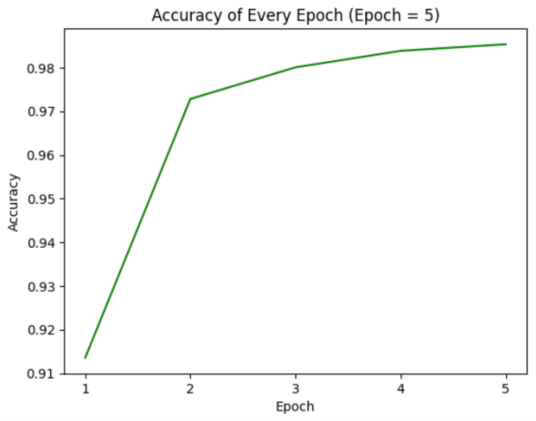
[Testing Accuracy]

```
100%|██████████| 79/79 [00:01<00:00, 58.31it/s]
Accuracy: 99.30%
```

[Training Loss & Accuracy]

```
100%|██████████| 469/469 [00:18<00:00, 25.40it/s]
Epoch 1/5, loss = 0.27422, acc = 0.91356
100%|██████████| 469/469 [00:19<00:00, 23.48it/s]
Epoch 2/5, loss = 0.09155, acc = 0.97281
100%|██████████| 469/469 [00:18<00:00, 25.16it/s]
Epoch 3/5, loss = 0.06744, acc = 0.98005
100%|██████████| 469/469 [00:18<00:00, 25.26it/s]
Epoch 4/5, loss = 0.05584, acc = 0.98383
100%|██████████| 469/469 [00:18<00:00, 25.71it/s]
Epoch 5/5, loss = 0.04862, acc = 0.98534
```





Q2 Summary

VGG與CNN的效果從Loss以及Accuracy上看起來並沒有太大的差異，兩種model在MNSIT這個資料集上都表現得很好，擁有98%的準確率，且都是於第二至三回合開始收斂。