# 計算機視覺

# Computer Vision

## Homework III

國立清華大學


**系所:電子所碩二**

**中文姓名:李聖謙**

**學號:111063517**


**授課老師:Min Sun, 孫民**

## Content

# Problems

1. (30%) Finish the rest of the codes for Problem 1 and Problem 2 according to the hint. (2 code cells in total.)
2. Train small model (resnet18) and big model (resnet50) from scratch on ‘sixteenth train dataloader’, ‘half train dataloader’, and ‘train dataloader’ respectively.

I. Load models

```python
###########################################
#############    Model_1~4   #############
###########################################
model_1 = torch.hub.load('pytorch/vision:v0.10.0', 'resnet18', weights= None)          # 有4種情況 18*none, 18*v1, 50*none, 50*v1
model_2 = torch.hub.load('pytorch/vision:v0.10.0', 'resnet18', weights='IMAGENET1K_V1')
model_3 = torch.hub.load('pytorch/vision:v0.10.0', 'resnet50', weights= None)
model_4 = torch.hub.load('pytorch/vision:v0.10.0', 'resnet50', weights='IMAGENET1K_V1')

model_1.fc = torch.nn.Linear(model_1.fc.in_features, 10)
model_2.fc = torch.nn.Linear(model_2.fc.in_features, 10)
model_3.fc = torch.nn.Linear(model_3.fc.in_features, 10)
model_4.fc = torch.nn.Linear(model_4.fc.in_features, 10)

print('done')
```

Model_1 is small mode & weight = None

Model_2 is small mode & weight = IMAGENET1K_V1

Model_3 is big mode & weight = None

Model_4 is big mode & weight = IMAGENET1K_V1

II. Training and testing models

```python
#=====================================================================
#                        Training model def
#=====================================================================

###########################################
##########    loss & optimizer   ##########
###########################################
loss_fn = nn.CrossEntropyLoss()
optimizer_1 = torch.optim.Adam(model_1.parameters(), lr=1e-3)
optimizer_2 = torch.optim.Adam(model_2.parameters(), lr=1e-3)
optimizer_3 = torch.optim.Adam(model_3.parameters(), lr=1e-3)
optimizer_4 = torch.optim.Adam(model_4.parameters(), lr=1e-3)
```

Loss & Optimizer

```python
###########################################
#############    Model_1   #############
###########################################
def train_1(dataloader, model_1, loss_fn, optimizer_1):
    num_batches = len(dataloader)
    size = len(dataloader.dataset)
    epoch_loss = 0
    correct = 0

    model_1.train()

    for X, Y in tqdm(dataloader):
        # X, Y = X.to(device), Y.to(device)

        # Compute prediction error
        pred = model_1(X)
        loss = loss_fn(pred, Y)

        # Backpropagation
        optimizer_1.zero_grad()
        loss.backward()
        optimizer_1.step()

        epoch_loss += loss.item()
        pred = pred.argmax(dim = 1, keepdim = True)
        correct += pred.eq(Y.view_as(pred)).sum().item()

    avg_epoch_loss = epoch_loss / num_batches
    avg_acc = correct / size
    return avg_epoch_loss, avg_acc
```

Training definition

```python
def test_1(dataloader, model_1, loss_fn):
    num_batches = len(dataloader)
    size = len(dataloader.dataset)
    epoch_loss = 0
    correct = 0

    model_1.eval()

    with torch.no_grad():
        for X, Y in tqdm(dataloader):
            # X, Y = X.to(device), Y.to(device)

            pred = model_1(X)

            epoch_loss += loss_fn(pred, Y).item()
            pred = pred.argmax(dim = 1, keepdim = True)
            correct += pred.eq(Y.view_as(pred)).sum().item()

    avg_epoch_loss = epoch_loss / num_batches
    avg_acc = correct / size
    return avg_epoch_loss, avg_acc
```

Testing definition

Sixteenth

Half

All

```python
#======================================================================
#                              Training
#======================================================================
epochs = 5

print("==========Adam==========\n")

##########################################
##############    Model_1   ##############
##########################################
print("==========model_1==========\n")

print("==========sixteenth_train_dataloader==========\n")
acc_sixteenth_list_1 = []
acc_test_sixteenth_list_1 = []
for epoch in range(epochs):
  train_loss, train_acc = train_1(sixteenth_train_dataloader, model_1, loss_fn, optimizer_1)
  test_loss, test_acc = test_1(valid_dataloader, model_1, loss_fn)
  print(f"Epoch {epoch+1:2d}:Loss = {train_loss:.4f} Acc = {train_acc:.2f} Test_Loss = {test_loss:.4f} Test_Acc = {test_acc:.2f}")
  acc_sixteenth_list_1.append(train_acc)
  acc_test_sixteenth_list_1.append(test_acc)
print("Done!")

print("==========half_train_dataloader==========\n")
acc_half_list_1 = []
acc_test_half_list_1 = []
for epoch in range(epochs):
  train_loss, train_acc = train_1(half_train_dataloader, model_1, loss_fn, optimizer_1)
  test_loss, test_acc = test_1(valid_dataloader, model_1, loss_fn)
  print(f"Epoch {epoch+1:2d}:Loss = {train_loss:.4f} Acc = {train_acc:.2f} Test_Loss = {test_loss:.4f} Test_Acc = {test_acc:.2f}")
  acc_half_list_1.append(train_acc)
  acc_test_half_list_1.append(test_acc)
print("Done!")

print("==========train_dataloader==========\n")
acc_all_list_1 = []
acc_test_all_list_1 = []
for epoch in range(epochs):
  train_loss, train_acc = train_1(train_dataloader, model_1, loss_fn, optimizer_1)
  test_loss, test_acc = test_1(valid_dataloader, model_1, loss_fn)
  print(f"Epoch {epoch+1:2d}:Loss = {train_loss:.4f} Acc = {train_acc:.2f} Test_Loss = {test_loss:.4f} Test_Acc = {test_acc:.2f}")
  acc_all_list_1.append(train_acc)
  acc_test_all_list_1.append(test_acc)
print("Done!")
```

Only model_1 is taken as a representation, and so on for other models.

III. Matplotlib

```python
#======================================================================
#                              Matplotlib
#======================================================================
###########################################
################   none   ################
###########################################
plt.subplots_adjust(hspace=16, wspace=16)
plt.figure(figsize=(10, 15))
###########################################
################    acc    ################
###########################################
plt.subplot2grid((16,40), (0, 0), rowspan=4, colspan=16)
plt.xlim(0, 1)
plt.ylim(0, 1)
x_ticks = np.arange(0, 1.1, 0.2)
y_ticks = np.arange(0, 1.1, 0.2)
plt.xticks(x_ticks)
plt.xlabel("Data Size")
plt.yticks(y_ticks)
plt.ylabel("Train Accuracy")
plt.title("                                                              " "weights = Nano\n""Data Size vs Accuracy ")
data_x = [0, 1/16, 0.5, 1]
data_y_small = [0, acc_sixteenth_list_1[epochs-1], acc_half_list_1[epochs-1], acc_all_list_1[epochs-1]]
plt.plot(data_x, data_y_small, label="small Model")
data_y_big = [0, acc_sixteenth_list_3[epochs-1], acc_half_list_3[epochs-1], acc_all_list_3[epochs-1]]
plt.plot(data_x, data_y_big, label="Big Model")
plt.legend()
###########################################
##############    test acc    #############
###########################################
plt.subplot2grid((16,40), (0, 20), rowspan=4, colspan=16)
plt.xlim(0, 1)
plt.ylim(0, 1)
x_ticks = np.arange(0, 1.1, 0.2)
y_ticks = np.arange(0, 1.1, 0.2)
plt.xticks(x_ticks)
plt.xlabel("Data Size")
plt.yticks(y_ticks)
plt.ylabel("Test Accuracy")
plt.title("Data Size vs Accuracy ")
data_x = [0, 1/16, 0.5, 1]
data_y_small = [0, acc_test_sixteenth_list_1[epochs-1], acc_test_half_list_1[epochs-1], acc_test_all_list_1[epochs-1]]
plt.plot(data_x, data_y_small, label="small Model")
data_y_big = [0, acc_test_sixteenth_list_3[epochs-1], acc_test_half_list_3[epochs-1], acc_test_all_list_3[epochs-1]]
plt.plot(data_x, data_y_big, label="Big Model")
plt.legend()
```

Python

Using matplotlib, plot shows the relationship between training accuracy and test accuracy and data size.

3. (30%) Achieve the best performance given all training data using whatever model and training strategy. (You cannot use the model that was pretrained on CIFAR10)

To achieve better performance, the hyperparameter must be adjusted. Since the training result is better when adding weight to the model, I chose to adjust the hyperparameter starting from the model with weight = IMAGENET1K_V1 with big model.

I tried three training strategies. The first one is to reduce the learning rate of optimizer to reduces the overfitting phenomenon during testing. The second is to reduce the batch size so that it can make more gradient adjustments. The third is that I tried another optimizer's gradient calculation. Method, changed from Adam to SGD, but the training effect of the third method was not very good, or even worse, so in the end I chose to combine the first two training strategies. Below I will explain in detail why I chose these three training strategies.

I. Modify the learning rate of the optimizer.

Because under the original optimizer, lr=1e-3 will cause overfitting and distortion in the test results, so I set lr to 1e-4 and use a smaller learning rate to prevent overfitting.

```
loss_fn = nn.CrossEntropyLoss()
optimizer_1 = torch.optim.Adam(model_1.parameters(), lr=1e-4)
optimizer_2 = torch.optim.Adam(model_2.parameters(), lr=1e-4)
optimizer_3 = torch.optim.Adam(model_3.parameters(), lr=1e-4)
optimizer_4 = torch.optim.Adam(model_4.parameters(), lr=1e-4)
```

The results are as follows:

```
==========sixteenth_train_dataloader==========

100%|██████████| 13/13 [00:14<00:00,  1.14s/it]
100%|██████████| 40/40 [00:12<00:00,  3.22it/s]
Epoch  1:Loss = 2.1873 Acc = 0.01 Test_Loss = 2.0313 Test_Acc = 0.35
100%|██████████| 13/13 [00:14<00:00,  1.11s/it]
100%|██████████| 40/40 [00:12<00:00,  3.26it/s]
Epoch  2:Loss = 1.7713 Acc = 0.02 Test_Loss = 1.6447 Test_Acc = 0.46
100%|██████████| 13/13 [00:14<00:00,  1.10s/it]
100%|██████████| 40/40 [00:12<00:00,  3.31it/s]
Epoch  3:Loss = 1.4838 Acc = 0.03 Test_Loss = 1.3747 Test_Acc = 0.52
100%|██████████| 13/13 [00:14<00:00,  1.09s/it]
100%|██████████| 40/40 [00:11<00:00,  3.35it/s]
Epoch  4:Loss = 1.2763 Acc = 0.03 Test_Loss = 1.2405 Test_Acc = 0.56
100%|██████████| 13/13 [00:14<00:00,  1.09s/it]
100%|██████████| 40/40 [00:12<00:00,  3.30it/s]
Epoch  5:Loss = 1.1579 Acc = 0.04 Test_Loss = 1.1528 Test_Acc = 0.59
Done!
```

```
==========half_train_dataloader==========

100%|██████████| 98/98 [01:54<00:00,  1.17s/it]
100%|██████████| 40/40 [00:11<00:00,  3.37it/s]
Epoch  1:Loss = 0.9487 Acc = 0.33 Test_Loss = 0.7795 Test_Acc = 0.73
100%|██████████| 98/98 [01:53<00:00,  1.15s/it]
100%|██████████| 40/40 [00:12<00:00,  3.28it/s]
Epoch  2:Loss = 0.7320 Acc = 0.37 Test_Loss = 0.6772 Test_Acc = 0.77
100%|██████████| 98/98 [01:53<00:00,  1.16s/it]
100%|██████████| 40/40 [00:12<00:00,  3.33it/s]
Epoch  3:Loss = 0.6418 Acc = 0.39 Test_Loss = 0.6256 Test_Acc = 0.78
100%|██████████| 98/98 [01:53<00:00,  1.16s/it]
100%|██████████| 40/40 [00:12<00:00,  3.27it/s]
Epoch  4:Loss = 0.5674 Acc = 0.40 Test_Loss = 0.5693 Test_Acc = 0.80
100%|██████████| 98/98 [01:52<00:00,  1.15s/it]
100%|██████████| 40/40 [00:11<00:00,  3.40it/s]
Epoch  5:Loss = 0.5209 Acc = 0.41 Test_Loss = 0.5462 Test_Acc = 0.81
Done!
```

```
==========train_dataloader==========

100%|██████████| 196/196 [03:45<00:00,  1.15s/it]
100%|██████████| 40/40 [00:12<00:00,  3.26it/s]
Epoch  1:Loss = 0.4702 Acc = 0.84 Test_Loss = 0.4899 Test_Acc = 0.83
100%|██████████| 196/196 [03:49<00:00,  1.17s/it]
100%|██████████| 40/40 [00:12<00:00,  3.27it/s]
Epoch  2:Loss = 0.4053 Acc = 0.86 Test_Loss = 0.4633 Test_Acc = 0.84
100%|██████████| 196/196 [03:44<00:00,  1.15s/it]
100%|██████████| 40/40 [00:12<00:00,  3.25it/s]
Epoch  3:Loss = 0.3585 Acc = 0.88 Test_Loss = 0.4656 Test_Acc = 0.85
100%|██████████| 196/196 [03:44<00:00,  1.15s/it]
100%|██████████| 40/40 [00:12<00:00,  3.24it/s]
Epoch  4:Loss = 0.3221 Acc = 0.89 Test_Loss = 0.4579 Test_Acc = 0.85
100%|██████████| 196/196 [03:44<00:00,  1.15s/it]
100%|██████████| 40/40 [00:11<00:00,  3.40it/s]
Epoch  5:Loss = 0.2908 Acc = 0.90 Test_Loss = 0.4448 Test_Acc = 0.86
Done!
```

Big model all data:

Training:

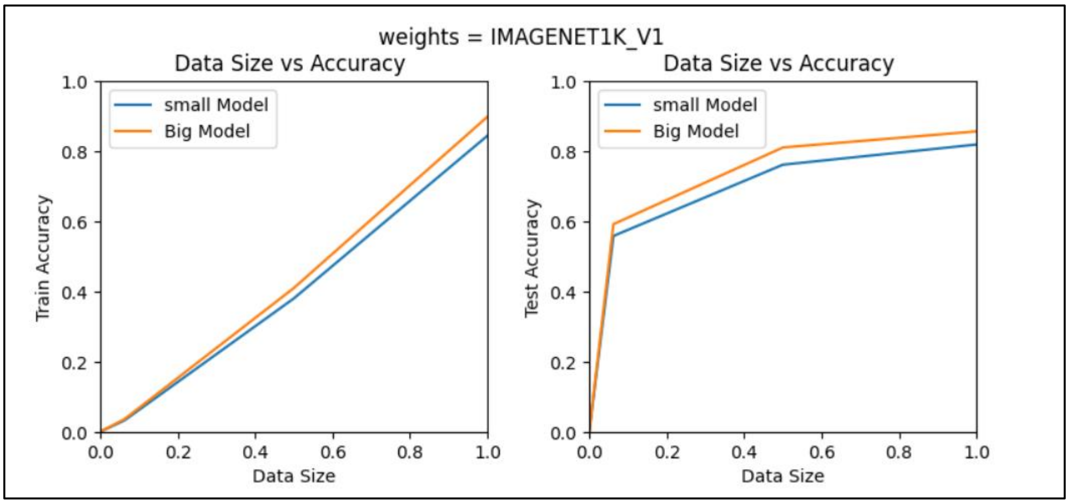
Loss = 0.2908

Accuracy = 0.90

Testing:

Loss = 0.4448

Accuracy = 0.86

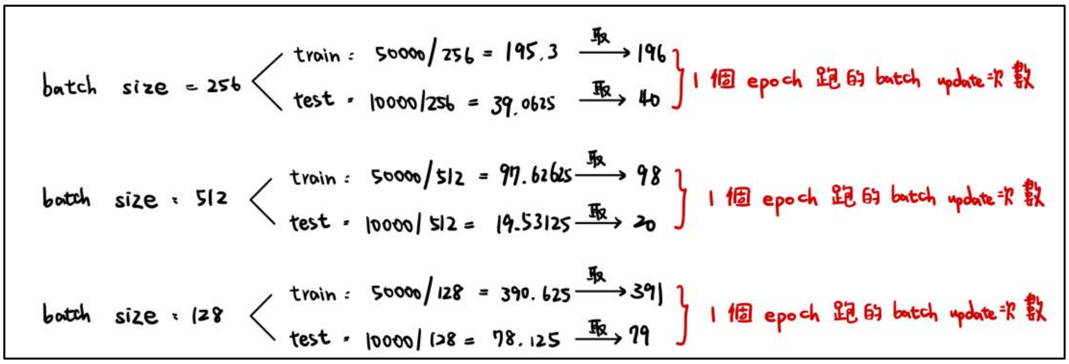



It can be found from the results that the training accuracy of lr=1e-4 is improved by 0.1 compared with lr=1e-3, and the testing accuracy is improved by 0.06. Although the loss has also declined, the difference between training loss and testing loss is quite large, which means that this model The training is good, but the test may still have overfitting.

II. Modify the batch size.

To solve the problem that the test loss is still too high, I tried to change the batch size. Before displaying the results, I first explain the principle of modifying the batch size.

Each epoch will determine how to train according to the batch size set at the beginning. Assuming batch size =data size=60000, that epoch will read all 60000 data before calculating the gradient and updating. But if batch size=1, then in gradient will be updated 60,000 times in an epoch.




As shown in the figure above, if the batch size becomes larger, the number of gradient updates in an epoch will become smaller. The advantage is that the training time will be shorter, and the gradient will be more stable. On the contrary, if the batch size becomes smaller, although the training time will be longer.The number of gradient updates is more and more noise are produced. I think there is a higher chance of training a good model.

Batch size will not only affect training loss, but also test loss. The principle is that loss will encounter many local minima during calculation, but not all gradient=0 values are local minima and may also be saddle points. If the local minima represent finding the correct relative low point, the resulting loss will be reduced. However, if you encounter a saddle point, it will cause errors in model judgment and increase the loss. Therefore, by changing the batch size, a small batch size can increase the probability of obtaining the correct local minima due to the greater number of gradient updates, which will prevent the model from falling into a saddle point and reduce the loss.

The batch size=32 results are as follows:

```
==========sixteenth_train_dataloader==========

100%|██████████| 98/98 [00:32<00:00,  2.97it/s]
100%|██████████| 313/313 [00:22<00:00, 14.08it/s]
Epoch  1:Loss = 1.9520 Acc = 0.02 Test_Loss = 1.5138 Test_Acc = 0.47
100%|██████████| 98/98 [00:32<00:00,  2.98it/s]
100%|██████████| 313/313 [00:23<00:00, 13.44it/s]
Epoch  2:Loss = 1.4279 Acc = 0.03 Test_Loss = 1.2079 Test_Acc = 0.58
100%|██████████| 98/98 [00:32<00:00,  3.04it/s]
100%|██████████| 313/313 [00:22<00:00, 13.64it/s]
Epoch  3:Loss = 1.2630 Acc = 0.03 Test_Loss = 1.0452 Test_Acc = 0.64
100%|██████████| 98/98 [00:32<00:00,  3.04it/s]
100%|██████████| 313/313 [00:22<00:00, 13.77it/s]
Epoch  4:Loss = 1.1517 Acc = 0.04 Test_Loss = 0.9750 Test_Acc = 0.67
100%|██████████| 98/98 [00:32<00:00,  3.01it/s]
100%|██████████| 313/313 [00:22<00:00, 13.75it/s]
Epoch  5:Loss = 1.0486 Acc = 0.04 Test_Loss = 0.9088 Test_Acc = 0.69
Done!
```

```
==========half_train_dataloader==========

100%|██████████| 782/782 [04:21<00:00,  2.99it/s]
100%|██████████| 313/313 [00:22<00:00, 13.80it/s]
Epoch  1:Loss = 0.9024 Acc = 0.34 Test_Loss = 0.7000 Test_Acc = 0.76
100%|██████████| 782/782 [04:20<00:00,  3.00it/s]
100%|██████████| 313/313 [00:22<00:00, 13.98it/s]
Epoch  2:Loss = 0.7417 Acc = 0.37 Test_Loss = 0.5921 Test_Acc = 0.80
100%|██████████| 782/782 [04:20<00:00,  3.00it/s]
100%|██████████| 313/313 [00:22<00:00, 14.09it/s]
Epoch  3:Loss = 0.6675 Acc = 0.39 Test_Loss = 0.5771 Test_Acc = 0.80
100%|██████████| 782/782 [04:20<00:00,  3.00it/s]
100%|██████████| 313/313 [00:22<00:00, 13.91it/s]
Epoch  4:Loss = 0.6050 Acc = 0.40 Test_Loss = 0.5018 Test_Acc = 0.83
100%|██████████| 782/782 [04:18<00:00,  3.03it/s]
100%|██████████| 313/313 [00:21<00:00, 14.24it/s]
Epoch  5:Loss = 0.5616 Acc = 0.41 Test_Loss = 0.4895 Test_Acc = 0.83
Done!
```

```
==========train_dataloader==========

100%|██████████| 1563/1563 [08:35<00:00,  3.03it/s]
100%|██████████| 313/313 [00:21<00:00, 14.42it/s]
Epoch  1:Loss = 0.5188 Acc = 0.82 Test_Loss = 0.4681 Test_Acc = 0.84
100%|██████████| 1563/1563 [08:36<00:00,  3.03it/s]
100%|██████████| 313/313 [00:22<00:00, 13.98it/s]
Epoch  2:Loss = 0.4659 Acc = 0.84 Test_Loss = 0.4273 Test_Acc = 0.86
100%|██████████| 1563/1563 [08:35<00:00,  3.03it/s]
100%|██████████| 313/313 [00:22<00:00, 14.16it/s]
Epoch  3:Loss = 0.4270 Acc = 0.85 Test_Loss = 0.4160 Test_Acc = 0.86
100%|██████████| 1563/1563 [08:36<00:00,  3.03it/s]
100%|██████████| 313/313 [00:22<00:00, 14.02it/s]
Epoch  4:Loss = 0.3858 Acc = 0.87 Test_Loss = 0.4166 Test_Acc = 0.86
100%|██████████| 1563/1563 [08:39<00:00,  3.01it/s]
100%|██████████| 313/313 [00:22<00:00, 14.01it/s]
Epoch  5:Loss = 0.3688 Acc = 0.87 Test_Loss = 0.3817 Test_Acc = 0.87
Done!
```

Big model all data:

Training:

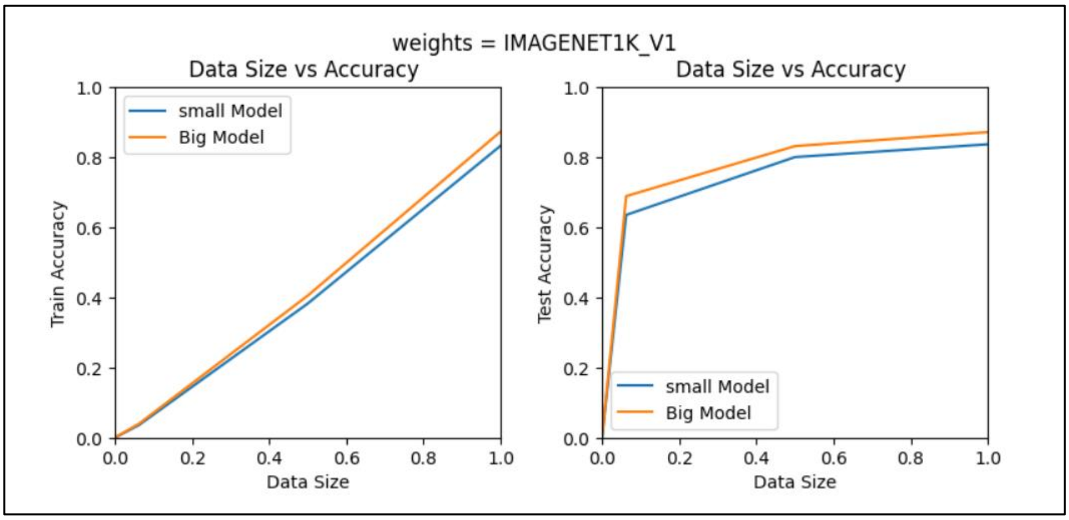
Loss = 0.3688

Accuracy = 0.87

Testing:

Loss = 0.3817

Accuracy = 0.87



From the results, it can be found that when the accuracy of training and testing is significantly improved, compared with the first time when only the learning rate was changed, there was overfitting. After reducing the batch size, the testing loss also showed a downward trend, which is consistent with the training loss.

III. Modify the optimizer from Adam to SGD

Although the previous two methods of adjusting hyperparameter have achieved better performance, I think there should be ways to improve performance, so I tried to change the entire optimizer to the SGD system, but the result was not as expected. The changed code is as follows.

```
loss_fn = nn.CrossEntropyLoss()
optimizer_1 = torch.optim.SGD(model_1.parameters(), lr=1e-3, momentum=0.9)
optimizer_2 = torch.optim.SGD(model_2.parameters(), lr=1e-3, momentum=0.9)
optimizer_3 = torch.optim.SGD(model_3.parameters(), lr=1e-3, momentum=0.9)
optimizer_4 = torch.optim.SGD(model_4.parameters(), lr=1e-3, momentum=0.9)
```

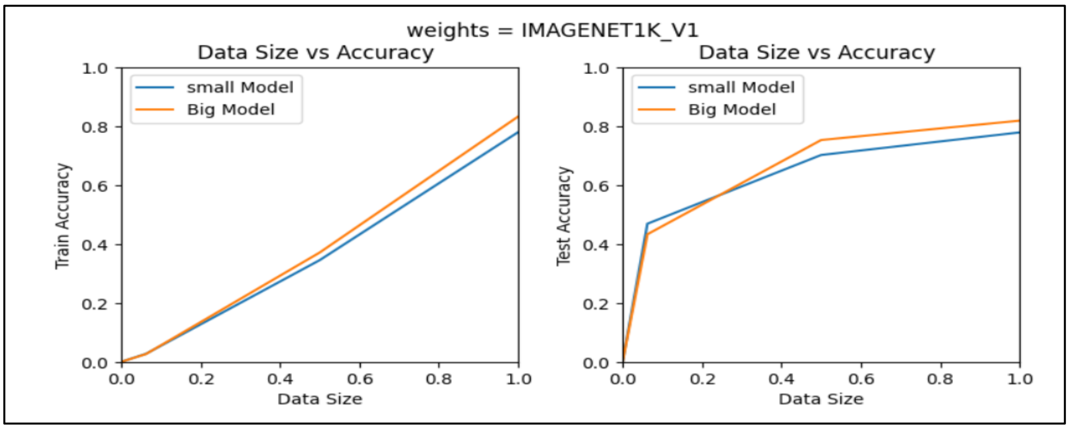
The significance of adding momentum is that the previous gradient value will be recorded during training to prevent the inability to continue training when gradient=0 is encountered during calculation, but it is not local minima but saddle points. If the momentum value is not set, the built-in preset value of pytorch will be set to zero and the training effect will be worse (The moment parameter in Adam is already built-in and does not need to be adjusted.). But this result is close to the goal in the question. When the data size is small, the small model is better. When the data size is large, the big model is better.

The results are as follows:

```
==========train_dataloader==========

100%|██████████| 196/196 [17:37<00:00,  5.40s/it]
100%|██████████| 40/40 [00:56<00:00,  1.41s/it]
Epoch  1:Loss = 0.6744 Acc = 0.76 Test_Loss = 0.6562 Test_Acc = 0.77
100%|██████████| 196/196 [17:48<00:00,  5.45s/it]
100%|██████████| 40/40 [00:56<00:00,  1.40s/it]
Epoch  2:Loss = 0.6046 Acc = 0.79 Test_Loss = 0.6094 Test_Acc = 0.79
100%|██████████| 196/196 [17:37<00:00,  5.39s/it]
100%|██████████| 40/40 [00:56<00:00,  1.40s/it]
Epoch  3:Loss = 0.5551 Acc = 0.80 Test_Loss = 0.5800 Test_Acc = 0.80
100%|██████████| 196/196 [17:46<00:00,  5.44s/it]
100%|██████████| 40/40 [00:57<00:00,  1.43s/it]
Epoch  4:Loss = 0.5091 Acc = 0.82 Test_Loss = 0.5562 Test_Acc = 0.81
100%|██████████| 196/196 [17:35<00:00,  5.38s/it]
100%|██████████| 40/40 [00:55<00:00,  1.39s/it]
Epoch  5:Loss = 0.4769 Acc = 0.83 Test_Loss = 0.5335 Test_Acc = 0.82
Done!
```

Big model all data:
Training:
Loss = 0.4769
Accuracy = 0.83
Testing:
Loss = 0.5335
Accuracy = 0.82



# Discussion

1. (30%) The relationship between the accuracy, model size, and the training dataset size. (Total 6 models. Small model trains on the sixteenth, half, and all data. Big model trains on the sixteenth, half, and all data. If the result is different from Fig. 1, please explain the possible reasons.)

   I. Small mode & weight = None (Model_1)

```
==========sixteenth_train_dataloader==========

100%|██████████| 13/13 [00:04<00:00,  3.04it/s]
100%|██████████| 40/40 [00:10<00:00,  3.80it/s]
Epoch  1:Loss = 2.1476 Acc = 0.02 Test_Loss = 2.2104 Test_Acc = 0.19
100%|██████████| 13/13 [00:03<00:00,  3.42it/s]
100%|██████████| 40/40 [00:05<00:00,  7.90it/s]
Epoch  2:Loss = 1.9254 Acc = 0.02 Test_Loss = 2.4530 Test_Acc = 0.27
100%|██████████| 13/13 [00:04<00:00,  2.98it/s]
100%|██████████| 40/40 [00:04<00:00,  8.10it/s]
Epoch  3:Loss = 1.7895 Acc = 0.02 Test_Loss = 1.6900 Test_Acc = 0.39
100%|██████████| 13/13 [00:03<00:00,  3.35it/s]
100%|██████████| 40/40 [00:04<00:00,  8.78it/s]
Epoch  4:Loss = 1.6628 Acc = 0.02 Test_Loss = 1.6550 Test_Acc = 0.39
100%|██████████| 13/13 [00:04<00:00,  3.18it/s]
100%|██████████| 40/40 [00:04<00:00,  8.61it/s]
Epoch  5:Loss = 1.6442 Acc = 0.02 Test_Loss = 1.6073 Test_Acc = 0.40
Done!
```

Sixteenth:

Training:

Loss = 1.6442

Accuracy = 0.02

Testing:

Loss = 1.6073

Accuracy = 0.40

```
==========half_train_dataloader==========

100%|██████████| 98/98 [00:31<00:00,  3.10it/s]
100%|██████████| 40/40 [00:04<00:00,  8.77it/s]
Epoch  1:Loss = 1.4727 Acc = 0.23 Test_Loss = 1.3647 Test_Acc = 0.50
100%|██████████| 98/98 [00:31<00:00,  3.12it/s]
100%|██████████| 40/40 [00:04<00:00,  8.65it/s]
Epoch  2:Loss = 1.3138 Acc = 0.26 Test_Loss = 1.2090 Test_Acc = 0.56
100%|██████████| 98/98 [00:31<00:00,  3.14it/s]
100%|██████████| 40/40 [00:04<00:00,  8.69it/s]
Epoch  3:Loss = 1.1734 Acc = 0.29 Test_Loss = 1.1495 Test_Acc = 0.58
100%|██████████| 98/98 [00:31<00:00,  3.11it/s]
100%|██████████| 40/40 [00:04<00:00,  8.69it/s]
Epoch  4:Loss = 1.0927 Acc = 0.30 Test_Loss = 1.0737 Test_Acc = 0.62
100%|██████████| 98/98 [00:31<00:00,  3.13it/s]
100%|██████████| 40/40 [00:04<00:00,  8.60it/s]
Epoch  5:Loss = 1.0284 Acc = 0.32 Test_Loss = 1.1848 Test_Acc = 0.60
Done!
```

Half:

Training:

Loss = 1.0284

Accuracy = 0.32

Testing:

Loss = 1.1848

Accuracy = 0.60

```
==========train_dataloader==========

100%|██████████| 196/196 [01:02<00:00,  3.16it/s]
100%|██████████| 40/40 [00:05<00:00,  7.90it/s]
Epoch  1:Loss = 0.9567 Acc = 0.66 Test_Loss = 0.8889 Test_Acc = 0.69
100%|██████████| 196/196 [01:02<00:00,  3.14it/s]
100%|██████████| 40/40 [00:04<00:00,  8.80it/s]
Epoch  2:Loss = 0.8718 Acc = 0.69 Test_Loss = 0.8235 Test_Acc = 0.71
100%|██████████| 196/196 [01:02<00:00,  3.13it/s]
100%|██████████| 40/40 [00:04<00:00,  8.73it/s]
Epoch  3:Loss = 0.8207 Acc = 0.71 Test_Loss = 0.8278 Test_Acc = 0.71
100%|██████████| 196/196 [01:02<00:00,  3.12it/s]
100%|██████████| 40/40 [00:04<00:00,  8.12it/s]
Epoch  4:Loss = 0.7711 Acc = 0.73 Test_Loss = 0.8163 Test_Acc = 0.72
100%|██████████| 196/196 [01:02<00:00,  3.15it/s]
100%|██████████| 40/40 [00:04<00:00,  8.62it/s]
Epoch  5:Loss = 0.7285 Acc = 0.74 Test_Loss = 0.7340 Test_Acc = 0.74
Done!
```

All:

Training:

Loss = 0.7285

Accuracy = 0.74

Testing:

Loss = 0.7340

Accuracy = 0.74

II. Small mode & weight = IMAGENET1K_V1 (Model_2)

```
==========sixteenth_train_dataloader==========

100%|██████████| 13/13 [00:04<00:00,  3.08it/s]
100%|██████████| 40/40 [00:04<00:00,  8.71it/s]
Epoch  1:Loss = 1.9112 Acc = 0.02 Test_Loss = 2.0604 Test_Acc = 0.40
100%|██████████| 13/13 [00:03<00:00,  3.40it/s]
100%|██████████| 40/40 [00:04<00:00,  8.28it/s]
Epoch  2:Loss = 1.4164 Acc = 0.03 Test_Loss = 1.4847 Test_Acc = 0.50
100%|██████████| 13/13 [00:03<00:00,  3.38it/s]
100%|██████████| 40/40 [00:04<00:00,  8.59it/s]
Epoch  3:Loss = 1.2531 Acc = 0.04 Test_Loss = 1.3728 Test_Acc = 0.55
100%|██████████| 13/13 [00:04<00:00,  3.12it/s]
100%|██████████| 40/40 [00:04<00:00,  8.66it/s]
Epoch  4:Loss = 1.1466 Acc = 0.04 Test_Loss = 1.3930 Test_Acc = 0.56
100%|██████████| 13/13 [00:04<00:00,  3.21it/s]
100%|██████████| 40/40 [00:04<00:00,  8.55it/s]
Epoch  5:Loss = 1.0469 Acc = 0.04 Test_Loss = 1.0434 Test_Acc = 0.64
Done!
```

Sixteenth:

Training:

Loss = 1.0469

Accuracy = 0.04

Testing:

Loss = 1.0434

Accuracy = 0.64

```
==========half_train_dataloader==========

100%|██████████| 98/98 [00:31<00:00,  3.15it/s]
100%|██████████| 40/40 [00:04<00:00,  8.40it/s]
Epoch  1:Loss = 0.8949 Acc = 0.35 Test_Loss = 0.8260 Test_Acc = 0.72
100%|██████████| 98/98 [00:30<00:00,  3.16it/s]
100%|██████████| 40/40 [00:04<00:00,  8.44it/s]
Epoch  2:Loss = 0.7405 Acc = 0.37 Test_Loss = 0.8702 Test_Acc = 0.71
100%|██████████| 98/98 [00:31<00:00,  3.15it/s]
100%|██████████| 40/40 [00:04<00:00,  8.21it/s]
Epoch  3:Loss = 0.6913 Acc = 0.38 Test_Loss = 0.6901 Test_Acc = 0.77
100%|██████████| 98/98 [00:30<00:00,  3.18it/s]
100%|██████████| 40/40 [00:04<00:00,  8.10it/s]
Epoch  4:Loss = 0.6328 Acc = 0.39 Test_Loss = 0.6746 Test_Acc = 0.77
100%|██████████| 98/98 [00:31<00:00,  3.11it/s]
100%|██████████| 40/40 [00:04<00:00,  8.58it/s]
Epoch  5:Loss = 0.5959 Acc = 0.40 Test_Loss = 0.6241 Test_Acc = 0.79
Done!
```

Half:

Training:

Loss = 0.5959

Accuracy = 0.40

Testing:

Loss = 0.6241

Accuracy = 0.79

```
==========train_dataloader==========

100%|██████████| 196/196 [01:02<00:00,  3.14it/s]
100%|██████████| 40/40 [00:05<00:00,  7.94it/s]
Epoch  1:Loss = 0.5678 Acc = 0.81 Test_Loss = 0.6114 Test_Acc = 0.80
100%|██████████| 196/196 [01:02<00:00,  3.14it/s]
100%|██████████| 40/40 [00:04<00:00,  8.87it/s]
Epoch  2:Loss = 0.5277 Acc = 0.82 Test_Loss = 0.6739 Test_Acc = 0.79
100%|██████████| 196/196 [01:02<00:00,  3.12it/s]
100%|██████████| 40/40 [00:04<00:00,  8.21it/s]
Epoch  3:Loss = 0.4903 Acc = 0.83 Test_Loss = 0.5922 Test_Acc = 0.81
100%|██████████| 196/196 [01:01<00:00,  3.16it/s]
100%|██████████| 40/40 [00:04<00:00,  8.79it/s]
Epoch  4:Loss = 0.4666 Acc = 0.84 Test_Loss = 0.5617 Test_Acc = 0.81
100%|██████████| 196/196 [01:02<00:00,  3.12it/s]
100%|██████████| 40/40 [00:04<00:00,  8.78it/s]
Epoch  5:Loss = 0.4430 Acc = 0.85 Test_Loss = 0.5397 Test_Acc = 0.82
Done!
```

All:

Training:

Loss = 0.4430

Accuracy = 0.85

Testing:

Loss = 0.5397

Accuracy = 0.82

III. Big mode & weight = None (Model_3)

```
==========sixteenth_train_dataloader==========

100%|██████████| 13/13 [00:10<00:00,  1.20it/s]
100%|██████████| 40/40 [00:09<00:00,  4.24it/s]
Epoch  1:Loss = 2.8531 Acc = 0.01 Test_Loss = 2.3809 Test_Acc = 0.11
100%|██████████| 13/13 [00:10<00:00,  1.21it/s]
100%|██████████| 40/40 [00:09<00:00,  4.41it/s]
Epoch  2:Loss = 2.3735 Acc = 0.01 Test_Loss = 2.1382 Test_Acc = 0.18
100%|██████████| 13/13 [00:10<00:00,  1.19it/s]
100%|██████████| 40/40 [00:09<00:00,  4.23it/s]
Epoch  3:Loss = 2.1039 Acc = 0.02 Test_Loss = 1.9496 Test_Acc = 0.27
100%|██████████| 13/13 [00:10<00:00,  1.21it/s]
100%|██████████| 40/40 [00:09<00:00,  4.21it/s]
Epoch  4:Loss = 2.0923 Acc = 0.02 Test_Loss = 1.9196 Test_Acc = 0.29
100%|██████████| 13/13 [00:11<00:00,  1.18it/s]
100%|██████████| 40/40 [00:09<00:00,  4.20it/s]
Epoch  5:Loss = 1.9655 Acc = 0.02 Test_Loss = 1.8475 Test_Acc = 0.32
Done!
```

Sixteenth:

Training:

Loss = 1.9655

Accuracy = 0.02

Testing:

Loss = 1.8475

Accuracy = 0.32

```
==========half_train_dataloader==========

100%|██████████| 98/98 [01:24<00:00,  1.16it/s]
100%|██████████| 40/40 [00:09<00:00,  4.39it/s]
Epoch  1:Loss = 1.8089 Acc = 0.18 Test_Loss = 1.5872 Test_Acc = 0.41
100%|██████████| 98/98 [01:24<00:00,  1.16it/s]
100%|██████████| 40/40 [00:09<00:00,  4.24it/s]
Epoch  2:Loss = 1.6364 Acc = 0.21 Test_Loss = 4.4134 Test_Acc = 0.43
100%|██████████| 98/98 [01:25<00:00,  1.15it/s]
100%|██████████| 40/40 [00:09<00:00,  4.10it/s]
Epoch  3:Loss = 1.5839 Acc = 0.22 Test_Loss = 5.4166 Test_Acc = 0.45
100%|██████████| 98/98 [01:24<00:00,  1.16it/s]
100%|██████████| 40/40 [00:09<00:00,  4.22it/s]
Epoch  4:Loss = 1.5257 Acc = 0.23 Test_Loss = 12.3747 Test_Acc = 0.36
100%|██████████| 98/98 [01:24<00:00,  1.16it/s]
100%|██████████| 40/40 [00:09<00:00,  4.18it/s]
Epoch  5:Loss = 1.7516 Acc = 0.20 Test_Loss = 1.5808 Test_Acc = 0.42
Done!
```

Half:

Training:

Loss = 1.7516

Accuracy = 0.20

Testing:

Loss = 1.5808

Accuracy = 0.42

```
==========train_dataloader==========

100%|██████████| 196/196 [02:48<00:00,  1.16it/s]
100%|██████████| 40/40 [00:09<00:00,  4.28it/s]
Epoch  1:Loss = 1.5916 Acc = 0.44 Test_Loss = 1.4180 Test_Acc = 0.49
100%|██████████| 196/196 [02:48<00:00,  1.17it/s]
100%|██████████| 40/40 [00:09<00:00,  4.20it/s]
Epoch  2:Loss = 1.6318 Acc = 0.42 Test_Loss = 1.6026 Test_Acc = 0.41
100%|██████████| 196/196 [02:48<00:00,  1.16it/s]
100%|██████████| 40/40 [00:09<00:00,  4.34it/s]
Epoch  3:Loss = 1.5985 Acc = 0.43 Test_Loss = 1.4417 Test_Acc = 0.47
100%|██████████| 196/196 [02:47<00:00,  1.17it/s]
100%|██████████| 40/40 [00:09<00:00,  4.21it/s]
Epoch  4:Loss = 1.4871 Acc = 0.47 Test_Loss = 1.2941 Test_Acc = 0.52
100%|██████████| 196/196 [02:48<00:00,  1.16it/s]
100%|██████████| 40/40 [00:09<00:00,  4.21it/s]
Epoch  5:Loss = 1.3854 Acc = 0.51 Test_Loss = 1.3050 Test_Acc = 0.53
Done!
```

All:

Training:

Loss = 1.3854

Accuracy = 0.51

Testing:

Loss = 1.3050

Accuracy = 0.53

IV. Big mode & weight = IMAGENET1K_V1(Model_4)

```
==========sixteenth_train_dataloader==========

100%|██████████| 13/13 [00:10<00:00,  1.21it/s]
100%|██████████| 40/40 [00:09<00:00,  4.38it/s]
Epoch  1:Loss = 1.9672 Acc = 0.02 Test_Loss = 5.0815 Test_Acc = 0.24
100%|██████████| 13/13 [00:10<00:00,  1.22it/s]
100%|██████████| 40/40 [00:09<00:00,  4.18it/s]
Epoch  2:Loss = 1.3572 Acc = 0.03 Test_Loss = 4.4669 Test_Acc = 0.46
100%|██████████| 13/13 [00:10<00:00,  1.22it/s]
100%|██████████| 40/40 [00:09<00:00,  4.17it/s]
Epoch  3:Loss = 1.1853 Acc = 0.04 Test_Loss = 2.5890 Test_Acc = 0.52
100%|██████████| 13/13 [00:10<00:00,  1.24it/s]
100%|██████████| 40/40 [00:09<00:00,  4.28it/s]
Epoch  4:Loss = 1.0990 Acc = 0.04 Test_Loss = 1.0292 Test_Acc = 0.65
100%|██████████| 13/13 [00:10<00:00,  1.23it/s]
100%|██████████| 40/40 [00:09<00:00,  4.17it/s]
Epoch  5:Loss = 1.0114 Acc = 0.04 Test_Loss = 1.0275 Test_Acc = 0.66
Done!
```

Sixteenth:

Training:

Loss = 1.0114

Accuracy = 0.04

Testing:

Loss = 1.0275

Accuracy = 0.66

```
==========half_train_dataloader==========

100%|██████████| 98/98 [01:24<00:00,  1.16it/s]
100%|██████████| 40/40 [00:09<00:00,  4.25it/s]
Epoch  1:Loss = 0.8559 Acc = 0.35 Test_Loss = 0.8118 Test_Acc = 0.73
100%|██████████| 98/98 [01:24<00:00,  1.16it/s]
100%|██████████| 40/40 [00:09<00:00,  4.14it/s]
Epoch  2:Loss = 0.7433 Acc = 0.37 Test_Loss = 0.7029 Test_Acc = 0.76
100%|██████████| 98/98 [01:24<00:00,  1.16it/s]
100%|██████████| 40/40 [00:09<00:00,  4.34it/s]
Epoch  3:Loss = 0.6191 Acc = 0.39 Test_Loss = 0.6207 Test_Acc = 0.80
100%|██████████| 98/98 [01:24<00:00,  1.16it/s]
100%|██████████| 40/40 [00:09<00:00,  4.22it/s]
Epoch  4:Loss = 0.5868 Acc = 0.40 Test_Loss = 0.6437 Test_Acc = 0.79
100%|██████████| 98/98 [01:24<00:00,  1.17it/s]
100%|██████████| 40/40 [00:09<00:00,  4.12it/s]
Epoch  5:Loss = 0.5411 Acc = 0.41 Test_Loss = 0.5926 Test_Acc = 0.80
Done!
```

Half:

Training:

Loss = 0.5411

Accuracy = 0.41

Testing:

Loss = 0.5926

Accuracy = 0.80

```
==========train_dataloader==========

100%|██████████| 196/196 [02:50<00:00,  1.15it/s]
100%|██████████| 40/40 [00:09<00:00,  4.13it/s]
Epoch  1:Loss = 0.5100 Acc = 0.82 Test_Loss = 0.5158 Test_Acc = 0.83
100%|██████████| 196/196 [02:51<00:00,  1.14it/s]
100%|██████████| 40/40 [00:09<00:00,  4.15it/s]
Epoch  2:Loss = 0.4664 Acc = 0.84 Test_Loss = 0.5659 Test_Acc = 0.81
100%|██████████| 196/196 [02:50<00:00,  1.15it/s]
100%|██████████| 40/40 [00:09<00:00,  4.16it/s]
Epoch  3:Loss = 0.4440 Acc = 0.85 Test_Loss = 0.5110 Test_Acc = 0.83
100%|██████████| 196/196 [02:49<00:00,  1.15it/s]
100%|██████████| 40/40 [00:09<00:00,  4.21it/s]
Epoch  4:Loss = 0.5048 Acc = 0.83 Test_Loss = 0.5115 Test_Acc = 0.83
100%|██████████| 196/196 [02:49<00:00,  1.16it/s]
100%|██████████| 40/40 [00:09<00:00,  4.12it/s]
Epoch  5:Loss = 0.6015 Acc = 0.80 Test_Loss = 0.5791 Test_Acc = 0.80
Done!
```

All:

Training:

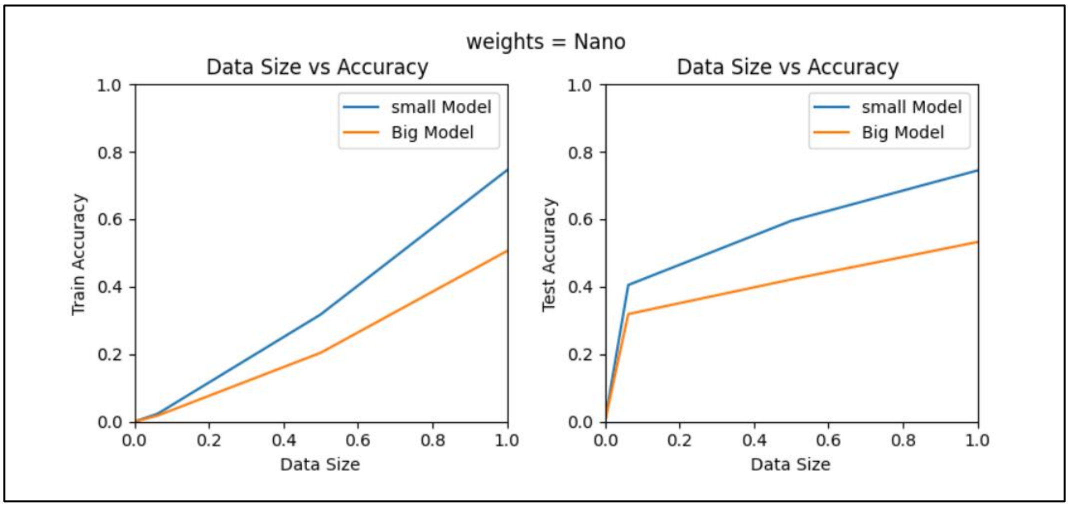
Loss = 0.6015

Accuracy = 0.80

Testing:

Loss = 0.5791

Accuracy = 0.80




It can be seen from the results that when weight = Nano, the performance of the small model in accuracy will be better than that of the big model, and the larger the data size, the more obvious the gap between the two. Although the small model does have better performance when the data size is small, there should be a reversal as the data size increases, but it does not happen.

I think there are several possibilities as following:

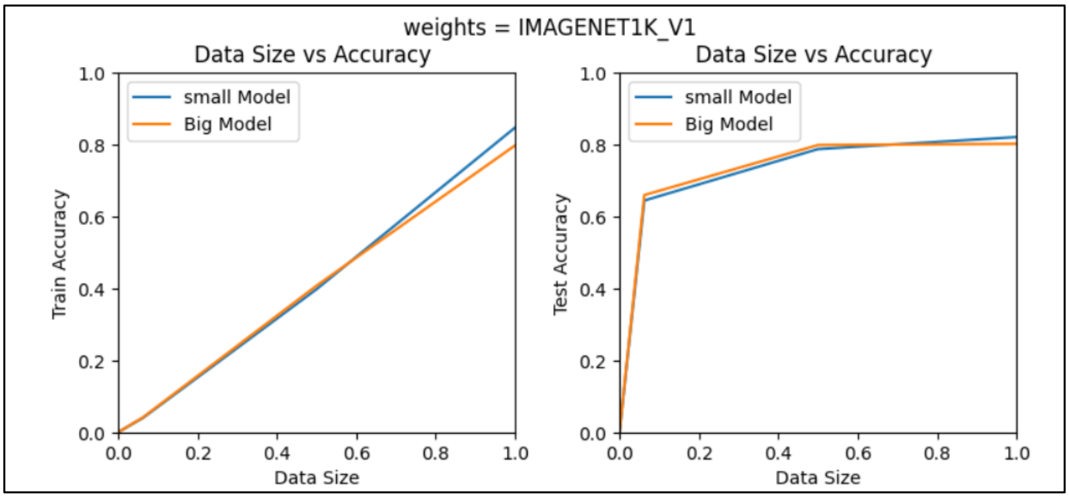
First, because training is in the case of weights=none, so there are no good training initial points and good pretrained feature points. In the big model (resnet50), because the neural network model is more complex, there will be overfitting during testing.

Second, we can find that the accuracy of the big model (resnet50) on the training side is already worse than that of the small model (resnet18). This means that training has failed, and the test result is not necessarily overfitting, but due to insufficient training. Insufficient training may be caused by insufficient number of epochs. Because there is no pretrained weight to assist, it may take more training times before the model can find the best solution.

Third, the influence of hyperparameter. The learning rate in the optimizer may be too large, and the training effect of the inverted big model is not good.

The above problems can be improved through the methods in discussion 2 and problem 3.

2. (10%) What if we train the ResNet with ImageNet initialized weights (weights="IMAGENET1K V1"). Please explain why the relationship changed this way?




IMAGENET1K_V1 is a pretrained model provided by pytorch for image classification. It can be seen that compared to weight = None, the accuracy becomes relatively in line with the expected image. Because IMAGENET1K_V1 provides a good initial point, the model can converge faster. When training the CIFAR10 dataset, the feature points that may be needed already have calculated weights in the pretrained model, so that better accuracy can be obtained. In addition, pretrained models can also prevent overfitting, but big models will still experience overfitting when the hyperparameter is not adjusted and the data size is large.