In [1]:
```
import torch
import torchvision
import numpy as np
import matplotlib.pyplot as plt
import torch.nn as nn
import torch.nn.functional as F
from torchvision.datasets import CIFAR100
import torchvision.transforms as transforms
from torchvision.utils import make_grid
from torch.utils.data.dataloader import DataLoader
from torch.utils.data import random_split,ConcatDataset

import ResNet
import data
import CNN
```

```
Files already downloaded and verified
```

In [2]:
```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

In [3]:
```
net = ResNet.resnet34(num_classes=20).to(device)
```

In [4]:
```
trainDataLoader, testDataLoader =  data.loadData_byBigClass(250)
```

In [5]:
```
loss = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(net.parameters(), lr=0.00005)
```

In [6]:
```
def evaluate(model,dataloader):
  acc = 0.0
  rights = 0
  wrongs = 0
  for i, test_examples in enumerate(dataloader, 0):
    #predicting using the nets
    inputs, labels = test_examples
    predicted_outputs = model(inputs.to(device))
    #Selecting the label which has the largest outputs
    outputs = torch.argmax(predicted_outputs, 1)

    #Counting successfully and unsuccessfully predicted cases
    for j, n in enumerate(outputs):
      if n == labels[j]:
        rights += 1
      else:
        wrongs += 1
  #calculate accuracy with the cases we recorded
  acc = rights/(rights+wrongs)
  #return the accuracy
  return acc
```

In [7]:
```
def train(model,train,test,loss_fn,optimizer,watch_iter):
    total_iter = 0
    loss = 0.0

    while total_iter < 10000:
        for batch in train:
            total_iter += 1
            train_inputs, train_labels = batch
            train_outputs = model(train_inputs.to(device))
```

```
            l = loss_fn(train_outputs, train_labels.to(device))
            loss += l.item()
            optimizer.zero_grad()
            l.backward()
            optimizer.step()

            if total_iter % watch_iter == 0:
                train_loss = loss / watch_iter
                train_loss_his.append(train_loss)
                loss = 0.0
                for batch in test:
                    test_inputs, test_labels = batch
                    test_outputs = model(test_inputs.to(device))
                    l = loss_fn(test_outputs, test_labels.to(device))
                    loss += l.item()
                test_loss_his.append(loss)
                txt = f'iter: {total_iter: 6d}, train loss: {train_loss}, test_loss: {l
                print(txt)
                print('accuracy: ' + str(evaluate(model,test)*100) + '%')
                loss = 0.0
    return
```

In [8]:
```
train_loss_his = []
test_loss_his = []
train(net,trainDataLoader,testDataLoader,loss,optimizer,100)
```

```
iter:    100, train loss: 2.588987829685211, test_loss: 93.46189141273499
accuracy: 27.16%
iter:    200, train loss: 2.288800039291382, test_loss: 87.14319729804993
accuracy: 32.550000000000004%
iter:    300, train loss: 2.051382590532303, test_loss: 82.8116979598999
accuracy: 35.57%
iter:    400, train loss: 1.9816999208927155, test_loss: 78.50823628902435
accuracy: 38.73%
iter:    500, train loss: 1.693417261838913, test_loss: 77.33368825912476
accuracy: 39.660000000000004%
iter:    600, train loss: 1.7227002274990082, test_loss: 74.36561298370361
accuracy: 42.49%
iter:    700, train loss: 1.3315027332305909, test_loss: 80.35701668262482
accuracy: 40.07%
iter:    800, train loss: 1.4230734670162202, test_loss: 75.99516558647156
accuracy: 42.57%
iter:    900, train loss: 0.8945182180404663, test_loss: 87.41694021224976
accuracy: 39.73%
iter:   1000, train loss: 1.0532421284914018, test_loss: 83.85055327415466
accuracy: 41.53%
iter:   1100, train loss: 0.500490782558918, test_loss: 98.13025283813477
accuracy: 39.96%
iter:   1200, train loss: 0.6394453766942024, test_loss: 96.5655448436737
accuracy: 40.699999999999996%
iter:   1300, train loss: 0.27211894288659094, test_loss: 106.43516087532043
accuracy: 39.82%
iter:   1400, train loss: 0.32847372174263, test_loss: 108.86777114868164
accuracy: 39.78%
iter:   1500, train loss: 0.15630148068070412, test_loss: 114.65765857696533
accuracy: 39.81%
iter:   1600, train loss: 0.17863095194101333, test_loss: 117.01314115524292
accuracy: 40.11%
iter:   1700, train loss: 0.10455941826105118, test_loss: 121.9595696926117
accuracy: 39.879999999999995%
iter:   1800, train loss: 0.13636657383292913, test_loss: 126.43559980392456
accuracy: 38.92%
iter:   1900, train loss: 0.10010721765458584, test_loss: 128.3287308216095
```

```
accuracy: 39.01%
iter:   2000, train loss: 0.1356023544818163, test_loss: 127.78088045120239
accuracy: 40.26%
iter:   2100, train loss: 0.10521364998072386, test_loss: 129.51181554794312
accuracy: 40.300000000000004%
iter:   2200, train loss: 0.1387629686295986, test_loss: 131.7462456226349
accuracy: 39.73%
iter:   2300, train loss: 0.09982798960059881, test_loss: 132.85077285766602
accuracy: 39.800000000000004%
iter:   2400, train loss: 0.12549008667469025, test_loss: 134.47439575195312
accuracy: 39.73%
iter:   2500, train loss: 0.09940264612436295, test_loss: 136.8858358860016
accuracy: 39.53%
iter:   2600, train loss: 0.14465889558196068, test_loss: 134.04464483261108
accuracy: 39.839999999999996%
iter:   2700, train loss: 0.10157572329044343, test_loss: 138.44389748573303
accuracy: 40.150000000000006%
iter:   2800, train loss: 0.11598061837255955, test_loss: 135.86789059638977
accuracy: 39.44%
iter:   2900, train loss: 0.07749187240377069, test_loss: 138.3361532688141
accuracy: 40.2%
iter:   3000, train loss: 0.10244607876986266, test_loss: 140.32854795455933
accuracy: 40.6%
iter:   3100, train loss: 0.07228214355185628, test_loss: 141.28221321105957
accuracy: 39.96%
iter:   3200, train loss: 0.07423256969079375, test_loss: 141.82456231117249
accuracy: 39.989999999999995%
iter:   3300, train loss: 0.06343984687700868, test_loss: 142.82974863052368
accuracy: 40.03%
iter:   3400, train loss: 0.08272552896291017, test_loss: 145.74136352539062
accuracy: 39.6%
iter:   3500, train loss: 0.07063958127051592, test_loss: 146.6438913345337
accuracy: 39.51%
iter:   3600, train loss: 0.10027904000133275, test_loss: 145.87724351882935
accuracy: 39.0%
iter:   3700, train loss: 0.08552527775987982, test_loss: 146.92946076393127
accuracy: 39.47%
iter:   3800, train loss: 0.10785311441868543, test_loss: 145.0497705936432
accuracy: 40.089999999999996%
iter:   3900, train loss: 0.08443156350404024, test_loss: 144.99008202552795
accuracy: 40.62%
iter:   4000, train loss: 0.10237992182374, test_loss: 148.57360816001892
accuracy: 39.87%
iter:   4100, train loss: 0.06393072888255119, test_loss: 148.24876713752747
accuracy: 40.28%
iter:   4200, train loss: 0.07198920987546444, test_loss: 148.72429656982422
accuracy: 40.23%
iter:   4300, train loss: 0.051738427486270666, test_loss: 150.89701867103577
accuracy: 39.660000000000004%
iter:   4400, train loss: 0.05594304327853024, test_loss: 150.09100317955017
accuracy: 40.29%
iter:   4500, train loss: 0.048977455971762535, test_loss: 152.60643601417542
accuracy: 40.160000000000004%
iter:   4600, train loss: 0.05990700338035822, test_loss: 153.51472425460815
accuracy: 40.23%
iter:   4700, train loss: 0.05213420808315277, test_loss: 151.93729519844055
accuracy: 40.79%
iter:   4800, train loss: 0.08929777858778834, test_loss: 152.22503876686096
accuracy: 39.77%
iter:   4900, train loss: 0.09220706164836884, test_loss: 153.0599524974823
accuracy: 39.739999999999995%
iter:   5000, train loss: 0.1269132711738348, test_loss: 149.15477442741394
accuracy: 40.03%
iter:   5100, train loss: 0.08587750017642976, test_loss: 153.58955097198486
accuracy: 39.79%
```

```
iter:   5200, train loss: 0.09555836889892816, test_loss: 151.00081777572632
accuracy: 39.97%
iter:   5300, train loss: 0.05303983698599041, test_loss: 150.01779580116272
accuracy: 40.69%
iter:   5400, train loss: 0.05118360232561827, test_loss: 152.223375082016
accuracy: 40.61%
iter:   5500, train loss: 0.027120996117591858, test_loss: 153.04109740257263
accuracy: 40.949999999999996%
iter:   5600, train loss: 0.031241668285802007, test_loss: 152.6897406578064
accuracy: 41.89%
iter:   5700, train loss: 0.018845220166258513, test_loss: 152.37044262886047
accuracy: 41.42%
iter:   5800, train loss: 0.02074489984661341, test_loss: 155.0032389163971
accuracy: 41.77%
iter:   5900, train loss: 0.019577863242011516, test_loss: 156.28093338012695
accuracy: 41.449999999999996%
iter:   6000, train loss: 0.026854052343405783, test_loss: 159.1444535255432
accuracy: 40.86%
iter:   6100, train loss: 0.04152850623242557, test_loss: 161.35222029685974
accuracy: 40.28%
iter:   6200, train loss: 0.1023405335098505, test_loss: 162.14966583251953
accuracy: 39.160000000000004%
iter:   6300, train loss: 0.16006116904318332, test_loss: 154.68674087524414
accuracy: 39.35%
iter:   6400, train loss: 0.19140281610190868, test_loss: 148.21221566200256
accuracy: 39.519999999999996%
iter:   6500, train loss: 0.09587398894131184, test_loss: 145.96292209625244
accuracy: 41.13%
iter:   6600, train loss: 0.079124117475003, test_loss: 147.100848197937
accuracy: 41.39%
iter:   6700, train loss: 0.035296878246590495, test_loss: 148.04418897628784
accuracy: 41.69%
iter:   6800, train loss: 0.027641991302371025, test_loss: 152.1803538799286
accuracy: 41.120000000000005%
iter:   6900, train loss: 0.011793556483462453, test_loss: 151.11974930763245
accuracy: 41.78%
iter:   7000, train loss: 0.010732201212085783, test_loss: 152.27709007263184
accuracy: 41.699999999999996%
iter:   7100, train loss: 0.006872306306613609, test_loss: 153.35126399993896
accuracy: 41.69%
iter:   7200, train loss: 0.00623724511009641, test_loss: 155.84460520744324
accuracy: 41.94%
iter:   7300, train loss: 0.0029734156955964863, test_loss: 153.91102743148804
accuracy: 42.63%
iter:   7400, train loss: 0.002565257928799838, test_loss: 154.19953632354736
accuracy: 42.67%
iter:   7500, train loss: 0.001268526484782342, test_loss: 155.46264910697937
accuracy: 42.79%
iter:   7600, train loss: 0.0011912362498696894, test_loss: 154.83665823936462
accuracy: 42.78%
iter:   7700, train loss: 0.0005717541078047361, test_loss: 154.4448516368866
accuracy: 43.519999999999996%
iter:   7800, train loss: 0.0005720722008845768, test_loss: 155.18482971191406
accuracy: 43.08%
iter:   7900, train loss: 0.00037320297313272023, test_loss: 155.30243825912476
accuracy: 42.980000000000004%
iter:   8000, train loss: 0.0003437360786483623, test_loss: 155.48675394058228
accuracy: 42.9%
iter:   8100, train loss: 0.00027084950270364063, test_loss: 155.95464706420898
accuracy: 43.07%
iter:   8200, train loss: 0.0002326358480786439, test_loss: 156.10128617286682
accuracy: 43.14%
iter:   8300, train loss: 0.00020981269284675363, test_loss: 156.18726205825806
accuracy: 43.15%
iter:   8400, train loss: 0.0001979276317433687, test_loss: 156.30847144126892
```

```
accuracy: 43.13%
iter:   8500, train loss: 0.0001876077154156519, test_loss: 156.4802484512329
accuracy: 43.04%
iter:   8600, train loss: 0.00018014327768469228, test_loss: 156.6173644065857
accuracy: 43.02%
iter:   8700, train loss: 0.00014893153602315578, test_loss: 156.72763633728027
accuracy: 42.970000000000006%
iter:   8800, train loss: 0.00013999283768498572, test_loss: 157.12383604049683
accuracy: 42.96%
iter:   8900, train loss: 0.00014419981285755058, test_loss: 157.3787477016449
accuracy: 42.870000000000005%
iter:   9000, train loss: 0.00013632561516715213, test_loss: 157.67504000663757
accuracy: 42.88%
iter:   9100, train loss: 0.00012116293633880559, test_loss: 157.8825855255127
accuracy: 42.809999999999995%
iter:   9200, train loss: 0.00012415992277965416, test_loss: 158.0439076423645
accuracy: 42.91%
iter:   9300, train loss: 0.00011050346111005638, test_loss: 158.20290207862854
accuracy: 43.03%
iter:   9400, train loss: 0.00011112603578803828, test_loss: 158.36382484436035
accuracy: 42.93%
iter:   9500, train loss: 0.00010103093416546471, test_loss: 158.58765745162964
accuracy: 43.01%
iter:   9600, train loss: 9.607782514649443e-05, test_loss: 158.7808609008789
accuracy: 43.03%
iter:   9700, train loss: 0.00014567002283001783, test_loss: 159.90488982200623
accuracy: 43.4%
iter:   9800, train loss: 0.00012859608821599977, test_loss: 159.67563652992249
accuracy: 43.5%
iter:   9900, train loss: 9.663858512794832e-05, test_loss: 159.7446644306183
accuracy: 43.32%
iter:  10000, train loss: 9.445611260161968e-05, test_loss: 160.01647901535034
accuracy: 43.13%
```

In [9]:
```
class_2 =       [['beaver', 'dolphin', 'otter', 'seal', 'whale'],
        ['aquarium_fish', 'flatfish', 'ray', 'shark', 'trout'],
        ['orchid', 'poppy', 'rose', 'sunflower', 'tulip'],
    ['bottle', 'bowl', 'can', 'cup', 'plate'],
        ['apple', 'mushroom', 'orange', 'pear', 'sweet_pepper'],
        ['clock', 'keyboard', 'lamp', 'telephone', 'television'],
    ['bed', 'chair', 'couch', 'table', 'wardrobe'],
        ['bee', 'beetle', 'butterfly', 'caterpillar', 'cockroach'],
        ['bear', 'leopard', 'lion', 'tiger', 'wolf'],
    ['bridge', 'castle', 'house', 'road', 'skyscraper'],
        ['cloud', 'forest', 'mountain', 'plain', 'sea'],
        ['camel', 'cattle', 'chimpanzee', 'elephant', 'kangaroo'],
        ['fox', 'porcupine', 'possum', 'raccoon', 'skunk'],
        ['crab', 'lobster', 'snail', 'spider', 'worm'],
        ['baby', 'boy', 'girl', 'man', 'woman'],
        ['crocodile', 'dinosaur', 'lizard', 'snake', 'turtle'],
        ['hamster', 'mouse', 'rabbit', 'shrew', 'squirrel'],
        ['maple_tree', 'oak_tree', 'palm_tree', 'pine_tree', 'willow_tree'],
        ['bicycle', 'bus', 'motorcycle', 'pickup_truck', 'train'],
        ['lawn_mower', 'rocket', 'streetcar', 'tank', 'tractor']]
```

In [ ]:
```
plt.plot(range(len(train_loss_his)),train_loss_his,'-',linewidth=3,label='Train loss')
plt.plot(range(len(train_loss_his)),test_loss_his,'-',linewidth=3,label='Test loss')
plt.xlabel('epoches')
plt.ylabel('loss')
plt.grid(True)
plt.legend()
```

Out[ ]: <matplotlib.legend.Legend at 0x148e2fe9dfa0>

In [ ]: