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=0.0001)
```

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: 2.5332211303710936, test_loss: 92.23707365989685
accuracy: 28.689999999999998%
iter:    200, train loss: 2.232292833328247, test_loss: 84.50260829925537
accuracy: 33.51%
iter:    300, train loss: 1.9840983641147614, test_loss: 78.5171263217926
accuracy: 38.98%
iter:    400, train loss: 1.9115539717674255, test_loss: 72.8037611246109
accuracy: 42.64%
iter:    500, train loss: 1.661146136522293, test_loss: 71.85437309741974
accuracy: 44.519999999999996%
iter:    600, train loss: 1.6413534426689147, test_loss: 66.95953822135925
accuracy: 46.93%
iter:    700, train loss: 1.3616656267642975, test_loss: 67.56169128417969
accuracy: 48.46%
iter:    800, train loss: 1.388744955062866, test_loss: 64.77696883678436
accuracy: 49.919999999999995%
iter:    900, train loss: 1.050610973238945, test_loss: 69.06714379787445
accuracy: 49.21%
iter:   1000, train loss: 1.1367044681310654, test_loss: 65.46756982803345
accuracy: 50.13999999999999%
iter:   1100, train loss: 0.7191055455803871, test_loss: 72.41883850097656
accuracy: 49.55%
iter:   1200, train loss: 0.8781714904308319, test_loss: 70.41666853427887
accuracy: 50.72%
iter:   1300, train loss: 0.4538443973660469, test_loss: 81.48580145835876
accuracy: 49.02%
iter:   1400, train loss: 0.6012928366661072, test_loss: 76.94457614421844
accuracy: 50.43%
iter:   1500, train loss: 0.2647048377990723, test_loss: 87.20381760597229
accuracy: 50.480000000000004%
iter:   1600, train loss: 0.38490975961089136, test_loss: 87.94103395938873
accuracy: 49.13%
iter:   1700, train loss: 0.18893719375133514, test_loss: 92.79567158222198
accuracy: 50.77%
iter:   1800, train loss: 0.24555758967995645, test_loss: 93.47587072849274
accuracy: 49.669999999999995%
iter:   1900, train loss: 0.13988042429089545, test_loss: 95.23387956619263
```

```
accuracy: 50.580000000000005%
iter:   2000, train loss: 0.17808632478117942, test_loss: 98.12761545181274
accuracy: 50.32%
iter:   2100, train loss: 0.10518449578434229, test_loss: 99.7421464920044
accuracy: 50.5%
iter:   2200, train loss: 0.12696527563035487, test_loss: 104.74409699440002
accuracy: 49.54%
iter:   2300, train loss: 0.09950956039130687, test_loss: 106.96649980545044
accuracy: 49.059999999999995%
iter:   2400, train loss: 0.15007975250482558, test_loss: 104.99172019958496
accuracy: 49.25%
iter:   2500, train loss: 0.09314730286598205, test_loss: 106.68686127662659
accuracy: 50.449999999999996%
iter:   2600, train loss: 0.11379334531724453, test_loss: 108.94511294364929
accuracy: 48.86%
iter:   2700, train loss: 0.08071077845990658, test_loss: 107.91326928138733
accuracy: 51.12%
iter:   2800, train loss: 0.10595363311469555, test_loss: 106.62535858154297
accuracy: 50.31%
iter:   2900, train loss: 0.08800137788057327, test_loss: 108.16518759727478
accuracy: 50.12%
iter:   3000, train loss: 0.11173584461212158, test_loss: 107.8587589263916
accuracy: 50.62%
iter:   3100, train loss: 0.07715226747095585, test_loss: 114.00261235237122
accuracy: 49.65%
iter:   3200, train loss: 0.09061702752485871, test_loss: 112.57902956008911
accuracy: 49.97%
iter:   3300, train loss: 0.0753959360346198, test_loss: 110.57070755958557
accuracy: 51.51%
iter:   3400, train loss: 0.10816138736903667, test_loss: 112.41074872016907
accuracy: 49.81%
iter:   3500, train loss: 0.07085177531465887, test_loss: 112.9036157131195
accuracy: 50.39%
iter:   3600, train loss: 0.09863466992974282, test_loss: 114.80265474319458
accuracy: 49.38%
iter:   3700, train loss: 0.07448856530711055, test_loss: 115.14307975769043
accuracy: 49.57%
iter:   3800, train loss: 0.08359509823843837, test_loss: 117.53816676139832
accuracy: 49.6%
iter:   3900, train loss: 0.06094980284571647, test_loss: 114.19868230819702
accuracy: 50.55%
iter:   4000, train loss: 0.07704205475747586, test_loss: 120.17772221565247
accuracy: 50.05%
iter:   4100, train loss: 0.06306493924930692, test_loss: 116.06131911277771
accuracy: 50.74999999999999%
iter:   4200, train loss: 0.0769523830153048, test_loss: 116.75829696655273
accuracy: 50.13999999999999%
iter:   4300, train loss: 0.05875626550987363, test_loss: 118.4331374168396
accuracy: 50.449999999999996%
iter:   4400, train loss: 0.08007174337282777, test_loss: 117.27180075645447
accuracy: 50.39%
iter:   4500, train loss: 0.06543203137814999, test_loss: 121.59341621398926
accuracy: 49.59%
iter:   4600, train loss: 0.0756635582819581, test_loss: 118.34532570838928
accuracy: 49.9%
iter:   4700, train loss: 0.05592101776972413, test_loss: 116.5152952671051
accuracy: 51.080000000000005%
iter:   4800, train loss: 0.07394366150721908, test_loss: 120.49987936019897
accuracy: 50.8%
iter:   4900, train loss: 0.07737955803051591, test_loss: 118.54648876190186
accuracy: 50.28%
iter:   5000, train loss: 0.07253922820091248, test_loss: 119.77606010437012
accuracy: 50.74999999999999%
iter:   5100, train loss: 0.05066114399582147, test_loss: 116.15652227401733
accuracy: 51.959999999999994%
```

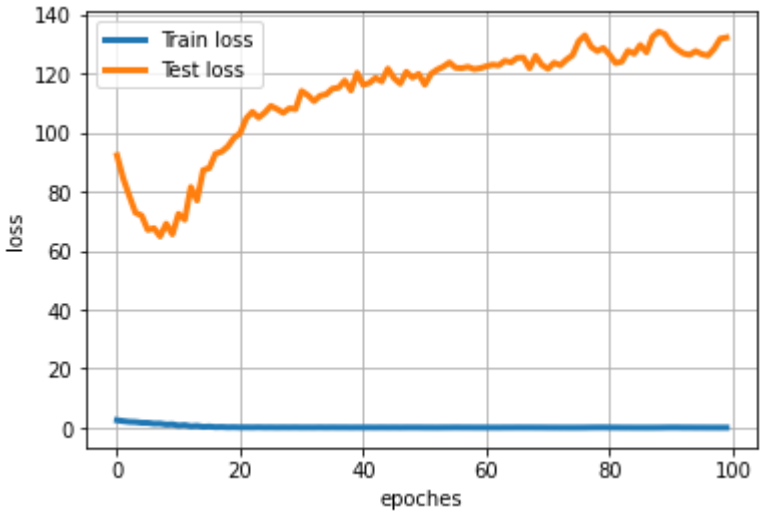
```
iter:   5200, train loss: 0.04900853645056486, test_loss: 119.81006526947021
accuracy: 51.15%
iter:   5300, train loss: 0.039287668177858, test_loss: 121.24898743629456
accuracy: 51.21%
iter:   5400, train loss: 0.060722747463732955, test_loss: 122.25502967834473
accuracy: 50.41%
iter:   5500, train loss: 0.06733125103637576, test_loss: 123.66688537597656
accuracy: 50.3%
iter:   5600, train loss: 0.0886251737177372, test_loss: 121.89406752586365
accuracy: 50.46000000000001%
iter:   5700, train loss: 0.06340759009122848, test_loss: 121.73637461662292
accuracy: 50.06%
iter:   5800, train loss: 0.06938926780596376, test_loss: 122.21899557113647
accuracy: 50.18%
iter:   5900, train loss: 0.052011940218508246, test_loss: 121.42330265045166
accuracy: 50.44%
iter:   6000, train loss: 0.05570143695920706, test_loss: 121.74571180343628
accuracy: 50.51%
iter:   6100, train loss: 0.042124906312674286, test_loss: 122.385249376297
accuracy: 50.94%
iter:   6200, train loss: 0.047991801481693984, test_loss: 122.93333768844604
accuracy: 51.080000000000005%
iter:   6300, train loss: 0.03655001156032085, test_loss: 122.68767619132996
accuracy: 51.31%
iter:   6400, train loss: 0.05844398835673928, test_loss: 124.24530863761902
accuracy: 50.4%
iter:   6500, train loss: 0.05421324741095304, test_loss: 123.74092555046082
accuracy: 50.739999999999995%
iter:   6600, train loss: 0.07581498870626092, test_loss: 125.2527072429657
accuracy: 50.239999999999995%
iter:   6700, train loss: 0.06284381326287986, test_loss: 125.35683751106262
accuracy: 50.11%
iter:   6800, train loss: 0.06939524816349149, test_loss: 121.72630286216736
accuracy: 50.62%
iter:   6900, train loss: 0.055871186628937725, test_loss: 125.91288900375366
accuracy: 49.86%
iter:   7000, train loss: 0.0580105307046324, test_loss: 122.85952544212341
accuracy: 50.74999999999999%
iter:   7100, train loss: 0.03461775164119899, test_loss: 121.54108166694641
accuracy: 51.5%
iter:   7200, train loss: 0.03237271926831454, test_loss: 123.58824157714844
accuracy: 51.28%
iter:   7300, train loss: 0.021854700751136987, test_loss: 122.74277210235596
accuracy: 51.82%
iter:   7400, train loss: 0.02370536344591528, test_loss: 124.63145160675049
accuracy: 52.12%
iter:   7500, train loss: 0.01660806266358122, test_loss: 126.14512276649475
accuracy: 51.49%
iter:   7600, train loss: 0.02498188893776387, test_loss: 130.88450264930725
accuracy: 50.79%
iter:   7700, train loss: 0.03912999338470399, test_loss: 132.87639379501343
accuracy: 50.07%
iter:   7800, train loss: 0.07376107016578316, test_loss: 129.0040729045868
accuracy: 50.57000000000001%
iter:   7900, train loss: 0.08838144985958934, test_loss: 127.5210771560669
accuracy: 49.9%
iter:   8000, train loss: 0.09556089289486408, test_loss: 128.594584941864
accuracy: 49.16%
iter:   8100, train loss: 0.06769502254202962, test_loss: 126.30320429801941
accuracy: 50.49%
iter:   8200, train loss: 0.05598935190588236, test_loss: 123.48584961891174
accuracy: 51.27%
iter:   8300, train loss: 0.02995357459411025, test_loss: 123.99334454536438
accuracy: 51.54%
iter:   8400, train loss: 0.030855697914958, test_loss: 127.6429123878479
```

```
accuracy: 51.13999999999999%
iter:   8500, train loss: 0.017450960143469273, test_loss: 126.632319688797
accuracy: 51.519999999999996%
iter:   8600, train loss: 0.018152537089772523, test_loss: 129.57845163345337
accuracy: 51.300000000000004%
iter:   8700, train loss: 0.012370810073916801, test_loss: 127.12490797042847
accuracy: 52.53%
iter:   8800, train loss: 0.018288901094347238, test_loss: 132.42096734046936
accuracy: 50.42%
iter:   8900, train loss: 0.026297238632105292, test_loss: 134.15207839012146
accuracy: 50.6%
iter:   9000, train loss: 0.06439559710212052, test_loss: 133.23156929016113
accuracy: 50.31%
iter:   9100, train loss: 0.08494882334023714, test_loss: 129.90970134735107
accuracy: 50.27%
iter:   9200, train loss: 0.08530074604786933, test_loss: 128.11973571777344
accuracy: 50.3%
iter:   9300, train loss: 0.0482060971390456, test_loss: 126.74896502494812
accuracy: 50.5%
iter:   9400, train loss: 0.05267113557085395, test_loss: 126.26752209663391
accuracy: 50.55%
iter:   9500, train loss: 0.031912611541338266, test_loss: 127.52797675132751
accuracy: 51.080000000000005%
iter:   9600, train loss: 0.03572129114530981, test_loss: 126.50971341133118
accuracy: 51.41%
iter:   9700, train loss: 0.02352637003874406, test_loss: 126.02950668334961
accuracy: 52.129999999999995%
iter:   9800, train loss: 0.024496543370187283, test_loss: 128.3332633972168
accuracy: 51.62%
iter:   9900, train loss: 0.01946283139055595, test_loss: 131.7424099445343
accuracy: 51.6%
iter:  10000, train loss: 0.02317940511275083, test_loss: 132.13578152656555
accuracy: 50.93%
```

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 [10]:
```
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[10]: <matplotlib.legend.Legend at 0x14784e69f670>



In [ ]: