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 model
import data
```

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

In [3]:
```
net = model.resnet34().to(device)
```

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

```
Files already downloaded and verified
```

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.float().cuda())
    #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_inputs, train_labels = train_inputs.to(device), train_labels.to(devic
            train_outputs = model(train_inputs)
```

```
            l = loss_fn(train_outputs, train_labels)
            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_inputs, test_labels = test_inputs.to(device), test_labels.to(d
                    test_outputs = model(test_inputs)
                    l = loss_fn(test_outputs, test_labels)
                    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]:

```
%%time
train_loss_his = []
test_loss_his = []
train(net,trainDataLoader,testDataLoader,loss,optimizer,100)
```

```
iter:    100, train loss: 4.024906370639801, test_loss: 147.24758911132812
accuracy: 13.48%
iter:    200, train loss: 3.5079910254478452, test_loss: 133.46297931671143
accuracy: 19.15%
iter:    300, train loss: 3.175412175655365, test_loss: 120.45579671859741
accuracy: 26.26%
iter:    400, train loss: 2.9666999769210816, test_loss: 115.60265684127808
accuracy: 27.839999999999996%
iter:    500, train loss: 2.733630211353302, test_loss: 106.96102118492126
accuracy: 32.5%
iter:    600, train loss: 2.5678416967391966, test_loss: 101.10906195640564
accuracy: 34.410000000000004%
iter:    700, train loss: 2.349814684391022, test_loss: 95.4348771572113
accuracy: 37.63%
iter:    800, train loss: 2.2757661724090577, test_loss: 90.31295824050903
accuracy: 40.58%
iter:    900, train loss: 2.093131390810013, test_loss: 87.75786983966827
accuracy: 42.52%
iter:   1000, train loss: 2.0226086473464964, test_loss: 83.69384062290192
accuracy: 43.38%
iter:   1100, train loss: 1.8799833965301513, test_loss: 79.19804322719574
accuracy: 46.75%
iter:   1200, train loss: 1.8263429117202759, test_loss: 76.57946300506592
accuracy: 47.94%
iter:   1300, train loss: 1.6842893755435944, test_loss: 74.7001291513443
accuracy: 49.830000000000005%
iter:   1400, train loss: 1.687008823156357, test_loss: 73.14329957962036
accuracy: 50.8%
iter:   1500, train loss: 1.5399150121212006, test_loss: 71.4420838356018
accuracy: 51.239999999999995%
iter:   1600, train loss: 1.5656410813331605, test_loss: 70.2825163602829
accuracy: 52.290000000000006%
iter:   1700, train loss: 1.4218439769744873, test_loss: 69.4449074268341
accuracy: 52.72%
iter:   1800, train loss: 1.4409341669082643, test_loss: 66.23213982582092
```

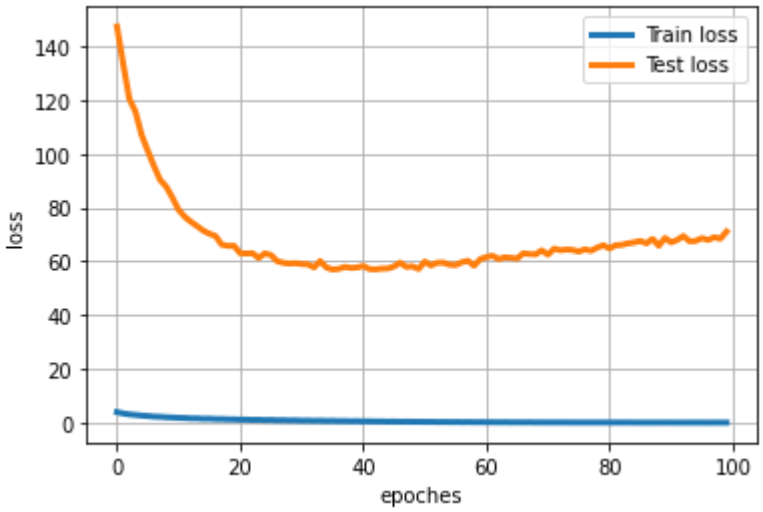
```
accuracy: 54.99000000000001%
iter:   1900, train loss: 1.3207526850700377, test_loss: 65.75469648838043
accuracy: 54.92%
iter:   2000, train loss: 1.3279290342330932, test_loss: 65.83396506309509
accuracy: 54.54%
iter:   2100, train loss: 1.2058181089162827, test_loss: 63.061524748802185
accuracy: 56.89999999999999%
iter:   2200, train loss: 1.2363215291500091, test_loss: 62.93902146816254
accuracy: 57.04%
iter:   2300, train loss: 1.1228331553936004, test_loss: 63.06032729148865
accuracy: 57.089999999999996%
iter:   2400, train loss: 1.1608581531047821, test_loss: 61.25407087802887
accuracy: 57.68%
iter:   2500, train loss: 1.034962671995163, test_loss: 62.955403566360474
accuracy: 56.95%
iter:   2600, train loss: 1.0807666432857514, test_loss: 62.37264609336853
accuracy: 57.54%
iter:   2700, train loss: 0.9645453882217407, test_loss: 60.11874461174011
accuracy: 58.809999999999995%
iter:   2800, train loss: 1.0055160957574845, test_loss: 59.462440490722656
accuracy: 59.14%
iter:   2900, train loss: 0.890141636133194, test_loss: 59.22799217700958
accuracy: 59.97%
iter:   3000, train loss: 0.9375342756509781, test_loss: 59.359928369522095
accuracy: 59.489999999999995%
iter:   3100, train loss: 0.8192101627588272, test_loss: 59.0440673828125
accuracy: 59.75%
iter:   3200, train loss: 0.8609080469608307, test_loss: 58.92434239387512
accuracy: 60.34%
iter:   3300, train loss: 0.762772381901741, test_loss: 57.76142382621765
accuracy: 61.1%
iter:   3400, train loss: 0.8196233797073365, test_loss: 60.14807450771332
accuracy: 59.98%
iter:   3500, train loss: 0.709559742808342, test_loss: 57.756112456321716
accuracy: 61.150000000000006%
iter:   3600, train loss: 0.7609703266620635, test_loss: 56.998082756996155
accuracy: 61.28%
iter:   3700, train loss: 0.6518784487247467, test_loss: 57.19835913181305
accuracy: 61.63999999999999%
iter:   3800, train loss: 0.7055479198694229, test_loss: 57.94878709316254
accuracy: 61.18%
iter:   3900, train loss: 0.60459666877985, test_loss: 57.58298063278198
accuracy: 62.28%
iter:   4000, train loss: 0.6437634426355362, test_loss: 57.735482931137085
accuracy: 61.99%
iter:   4100, train loss: 0.5580968996882438, test_loss: 58.30563187599182
accuracy: 61.75000000000001%
iter:   4200, train loss: 0.5996719118952751, test_loss: 57.156277894973755
accuracy: 62.56%
iter:   4300, train loss: 0.518047130703926, test_loss: 57.04919958114624
accuracy: 62.64999999999999%
iter:   4400, train loss: 0.548522855937481, test_loss: 57.33073854446411
accuracy: 62.56%
iter:   4500, train loss: 0.4695066991448402, test_loss: 57.38788425922394
accuracy: 62.64999999999999%
iter:   4600, train loss: 0.5114386633038521, test_loss: 58.11634945869446
accuracy: 62.44%
iter:   4700, train loss: 0.4287231248617172, test_loss: 59.53158748149872
accuracy: 62.050000000000004%
iter:   4800, train loss: 0.48393256902694703, test_loss: 57.95026230812073
accuracy: 62.96000000000001%
iter:   4900, train loss: 0.4014679515361786, test_loss: 58.16990411281586
accuracy: 62.86000000000001%
iter:   5000, train loss: 0.4448962053656578, test_loss: 57.1872581243515
accuracy: 63.36000000000001%
```

```
iter:   5100, train loss: 0.3718532872200012, test_loss: 59.96992588043213
accuracy: 62.69%
iter:   5200, train loss: 0.4065892821550369, test_loss: 58.491350293159485
accuracy: 63.849999999999994%
iter:   5300, train loss: 0.3340849670767784, test_loss: 59.5131995677948
accuracy: 63.24999999999999%
iter:   5400, train loss: 0.3783061960339546, test_loss: 59.5409471988678
accuracy: 63.55%
iter:   5500, train loss: 0.3124420160055161, test_loss: 58.798670530319214
accuracy: 63.83%
iter:   5600, train loss: 0.35119822472333906, test_loss: 58.649481654167175
accuracy: 63.959999999999994%
iter:   5700, train loss: 0.28850904762744906, test_loss: 59.67970657348633
accuracy: 63.81%
iter:   5800, train loss: 0.335436322838068, test_loss: 60.23436117172241
accuracy: 63.2%
iter:   5900, train loss: 0.2587211388349533, test_loss: 58.41147470474243
accuracy: 64.29%
iter:   6000, train loss: 0.3095052482187748, test_loss: 60.83417475223541
accuracy: 63.04%
iter:   6100, train loss: 0.25386717572808265, test_loss: 61.605520606040955
accuracy: 63.739999999999995%
iter:   6200, train loss: 0.27093087777495384, test_loss: 62.115352749824524
accuracy: 63.24999999999999%
iter:   6300, train loss: 0.23024898752570153, test_loss: 60.84813833236694
accuracy: 64.24%
iter:   6400, train loss: 0.27051698610186575, test_loss: 61.50290262699127
accuracy: 63.92%
iter:   6500, train loss: 0.21670550309121608, test_loss: 61.32792890071869
accuracy: 63.739999999999995%
iter:   6600, train loss: 0.23861306115984918, test_loss: 61.06890952587128
accuracy: 64.25%
iter:   6700, train loss: 0.20906426437199116, test_loss: 62.99718225002289
accuracy: 63.93%
iter:   6800, train loss: 0.2352594715356827, test_loss: 62.74605369567871
accuracy: 63.77%
iter:   6900, train loss: 0.18822875328361988, test_loss: 62.622225880622864
accuracy: 64.44%
iter:   7000, train loss: 0.2113390089571476, test_loss: 64.08211755752563
accuracy: 63.43%
iter:   7100, train loss: 0.17679747141897678, test_loss: 62.489580154418945
accuracy: 64.74%
iter:   7200, train loss: 0.20617523424327375, test_loss: 64.73842680454254
accuracy: 63.55%
iter:   7300, train loss: 0.16958268463611603, test_loss: 64.15803277492523
accuracy: 63.519999999999996%
iter:   7400, train loss: 0.2022206000983715, test_loss: 64.39790773391724
accuracy: 63.63999999999999%
iter:   7500, train loss: 0.16531826853752135, test_loss: 64.24014556407928
accuracy: 64.69%
iter:   7600, train loss: 0.18443458691239356, test_loss: 63.658273816108704
accuracy: 64.09%
iter:   7700, train loss: 0.14994367882609366, test_loss: 64.4768979549408
accuracy: 64.2%
iter:   7800, train loss: 0.16965111784636974, test_loss: 63.93740403652191
accuracy: 64.32%
iter:   7900, train loss: 0.15518925733864308, test_loss: 65.11812829971313
accuracy: 63.81%
iter:   8000, train loss: 0.1640006621927023, test_loss: 66.01484298706055
accuracy: 64.02%
iter:   8100, train loss: 0.14568025812506677, test_loss: 64.81221663951874
accuracy: 64.73%
iter:   8200, train loss: 0.16540457785129548, test_loss: 65.98684179782867
accuracy: 64.1%
iter:   8300, train loss: 0.1438988158851862, test_loss: 66.15814983844757
```

```
accuracy: 64.42999999999999%
iter:   8400, train loss: 0.17356188997626304, test_loss: 66.6974526643753
accuracy: 64.33%
iter:   8500, train loss: 0.13970071367919445, test_loss: 67.06410276889801
accuracy: 64.29%
iter:   8600, train loss: 0.13995911806821823, test_loss: 67.5802047252655
accuracy: 63.75999999999999%
iter:   8700, train loss: 0.11408874921500683, test_loss: 66.69949781894684
accuracy: 64.91%
iter:   8800, train loss: 0.13797770373523235, test_loss: 68.37976408004761
accuracy: 63.67%
iter:   8900, train loss: 0.12315288439393043, test_loss: 65.80476927757263
accuracy: 64.7%
iter:   9000, train loss: 0.15021535955369472, test_loss: 68.73481154441833
accuracy: 63.839999999999996%
iter:   9100, train loss: 0.12576601654291153, test_loss: 67.08365738391876
accuracy: 64.99000000000001%
iter:   9200, train loss: 0.1282997052371502, test_loss: 67.96765995025635
accuracy: 64.5%
iter:   9300, train loss: 0.11717251647263766, test_loss: 69.43722140789032
accuracy: 63.61%
iter:   9400, train loss: 0.13486560493707656, test_loss: 67.4457619190216
accuracy: 64.37%
iter:   9500, train loss: 0.1134812767803669, test_loss: 67.60335505008698
accuracy: 64.48%
iter:   9600, train loss: 0.13506418585777283, test_loss: 68.64422035217285
accuracy: 64.36%
iter:   9700, train loss: 0.10465744726359844, test_loss: 67.93462371826172
accuracy: 65.03999999999999%
iter:   9800, train loss: 0.13748703561723233, test_loss: 68.97268617153168
accuracy: 64.62%
iter:   9900, train loss: 0.11382203981280327, test_loss: 68.40832006931305
accuracy: 64.74%
iter:  10000, train loss: 0.12435924790799618, test_loss: 71.06939160823822
accuracy: 63.56%
CPU times: user 32min 5s, sys: 10min 47s, total: 42min 53s
Wall time: 43min 53s
```

In [9]:
```
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[9]:
```
<matplotlib.legend.Legend at 0x14c1b0487520>
```



In [10]:
```
print('accuracy: ' + str(evaluate(net,testDataLoader)*100) + '%')
```

```
accuracy: 63.56%
```