# Resnet 18 on faces

In [1]:
```
import torch
import torchvision
import torchvision.transforms as transforms
import resnet
import torch.optim as optim
import torch.nn as nn
import numpy as np
import matplotlib.pyplot as plt
import random
import resnet
```

In [2]:
```
batchsize = 75
rate = 0.1
epochs = 200
lr_decay = 0.85
lr_stride = 5
```

In [3]:
```
class FaceDataset(torch.utils.data.Dataset):

    def __init__(self, transform, train=True):
        self.image_prefix = "face_renders/face"
        self.image_suffix = ".jpg"
        self.vertex_prefix = "processed_faces/face"
        self.vertex_suffix = ".txt"
        self.count = 5000
        self.trainn = 4500

        self.train = train
        self.transform = transform

        shape = np.loadtxt(self.vertex_prefix + str(1) + self.vertex_suffix).s
hape
        tmp = np.zeros((self.count, shape[0], shape[1]))
        for i in range(self.count):
            tmp[i] = np.loadtxt(self.vertex_prefix + str(i + 1) + self.vertex_
suffix)

        self.mean = np.mean(tmp, axis=0)
        self.outputdim = shape[0] * shape[1]
        self.labels = [torch.from_numpy((lab - self.mean).reshape(self.outputd
im)).float() for lab in tmp]

        # simple version for working with CWD


    def __len__(self):
        if self.train:
            return self.trainn
        else:
            return self.count - self.trainn

    def __getitem__(self, idx):
        if not train:
            idx += self.trainn
        y = self.labels[idx]
        x = plt.imread(self.image_prefix + str(idx + 1) + self.image_suffix)

        sample = (x,y)
        sample = (self.transform(sample[0]), sample[1])

        return sample
```

In [4]:
```
transform = transforms.Compose(
    [transforms.ToTensor(),
     transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))])

trainset = FaceDataset(transform, train=True)
trainloader = torch.utils.data.DataLoader(trainset, batch_size=batchsize,
                                          shuffle=True, num_workers=0)

testset = FaceDataset(transform, train=False)
testloader = torch.utils.data.DataLoader(trainset, batch_size=batchsize,
                                          shuffle=True, num_workers=0)
```

In [5]:
```
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
print("torch.cuda.is_available()   =", torch.cuda.is_available())
print("torch.cuda.device_count()   =", torch.cuda.device_count())
print("torch.cuda.device('cuda')   =", torch.cuda.device(0))
print("torch.cuda.current_device() =", torch.cuda.current_device())

def to_device(data, device):
    if isinstance(data, (list, tuple)):
        return [to_device(x, device) for x in data]
    return data.to(device, non_blocking=True)

class DeviceDataLoader():
    def __init__(self, dl, device):
        self.dl = dl
        self.device = device

    def __iter__(self):
        for b in self.dl:
            yield to_device(b, self.device)

    def __len__(self):
        return len(self.dl)

trainloader = DeviceDataLoader(trainloader, device)
testloader = DeviceDataLoader(testloader, device)
```

```
torch.cuda.is_available()   = True
torch.cuda.device_count()   = 1
torch.cuda.device('cuda')   = <torch.cuda.device object at 0x0000016199917A58
>
torch.cuda.current_device() = 0
```

In [6]:
```
model = resnet.resnet18(output_size=trainset.outputdim)
model.to(device)
optimizer = optim.SGD(model.parameters(), lr=rate)

criterion = nn.MSELoss()

def adjust_learning_rate(optimizer, epoch, decay, stride):
    lr = rate * (decay ** (epoch // stride))
    for param_group in optimizer.param_groups:
        param_group['lr'] = lr
```

In [7]:
```
def train(model, optimizer, criterion, epochs, trainloader, testloader):
    model.train()
    samples = 1
    losses = []
    test_losses = []
    k = len(trainloader)// samples

    for epoch in range(epochs):  # loop over the dataset multiple times
        running_loss = 0.0
        for i, data in enumerate(trainloader, 0):
            # get the inputs
            inputs, labels = data

            # zero the parameter gradients
            optimizer.zero_grad()

            # forward + backward + optimize
            outputs = model(inputs)
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()

            # print statistics
            running_loss += loss.item()
            if i % k == k - 1:

                losses.append(running_loss / k)

                testloss = 0
                total = 0
                iterations = 0
                with torch.no_grad():
                    for data in testloader:
                        images, labels = data
                        outputs = model(images)
                        testloss += criterion(outputs, labels)
                        total += labels.size(0)
                        iterations += 1
                        if total > 200:
                            break
                test_losses.append(testloss / iterations)

                print('[%d, %5d] loss: %.3f test_loss: %.3f' %(epoch + 1, i +
1,losses[-1],test_losses[-1]))

                running_loss = 0.0

        adjust_learning_rate(optimizer, epoch+1, lr_decay, lr_stride)

    print('Finished Training')
    plt.plot(np.arange(0, len(losses)/samples, 1.0/samples), losses)
    plt.title("loss")
    plt.xlabel("epoch")
    plt.ylabel("loss")
    plt.show()
```

```
    plt.plot(np.arange(0, len(test_losses)/samples, 1.0/samples), test_losses)
    plt.title("test_loss")
    plt.xlabel("epoch")
    plt.ylabel("test_losses")
    plt.show()
    model.eval()
```

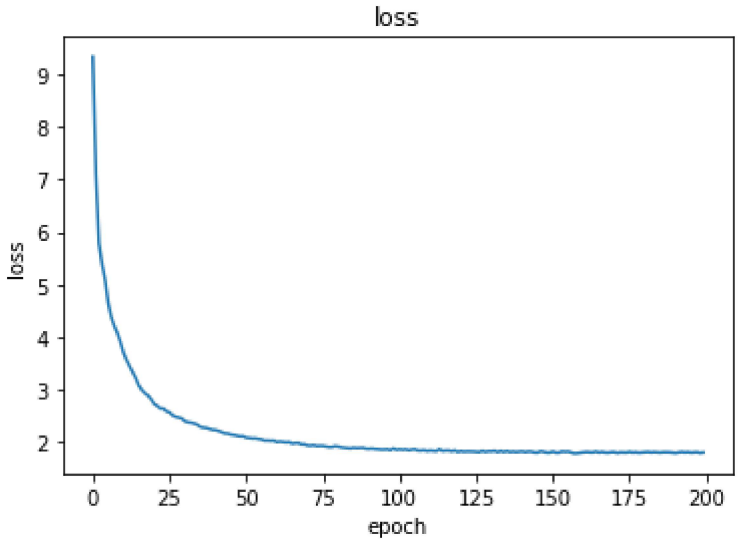
In [8]:
```
train(model, optimizer, criterion, epochs, trainloader, testloader)
```

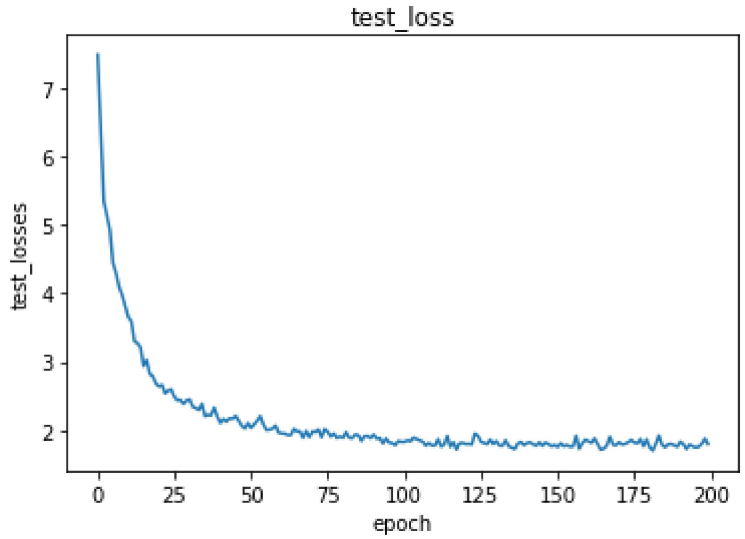
```
[1,    60] loss: 9.339 test_loss: 7.488
[2,    60] loss: 7.093 test_loss: 6.447
[3,    60] loss: 5.784 test_loss: 5.349
[4,    60] loss: 5.392 test_loss: 5.143
[5,    60] loss: 5.093 test_loss: 4.929
[6,    60] loss: 4.658 test_loss: 4.444
[7,    60] loss: 4.383 test_loss: 4.296
[8,    60] loss: 4.213 test_loss: 4.098
[9,    60] loss: 4.079 test_loss: 3.975
[10,    60] loss: 3.906 test_loss: 3.811
[11,    60] loss: 3.702 test_loss: 3.654
[12,    60] loss: 3.562 test_loss: 3.593
[13,    60] loss: 3.445 test_loss: 3.304
[14,    60] loss: 3.333 test_loss: 3.271
[15,    60] loss: 3.220 test_loss: 3.213
[16,    60] loss: 3.078 test_loss: 2.941
[17,    60] loss: 2.994 test_loss: 3.029
[18,    60] loss: 2.933 test_loss: 2.827
[19,    60] loss: 2.889 test_loss: 2.789
[20,    60] loss: 2.820 test_loss: 2.685
[21,    60] loss: 2.731 test_loss: 2.643
[22,    60] loss: 2.693 test_loss: 2.666
[23,    60] loss: 2.642 test_loss: 2.540
[24,    60] loss: 2.644 test_loss: 2.582
[25,    60] loss: 2.587 test_loss: 2.599
[26,    60] loss: 2.570 test_loss: 2.493
[27,    60] loss: 2.513 test_loss: 2.439
[28,    60] loss: 2.482 test_loss: 2.446
[29,    60] loss: 2.471 test_loss: 2.388
[30,    60] loss: 2.452 test_loss: 2.439
[31,    60] loss: 2.401 test_loss: 2.450
[32,    60] loss: 2.380 test_loss: 2.346
[33,    60] loss: 2.374 test_loss: 2.327
[34,    60] loss: 2.359 test_loss: 2.298
[35,    60] loss: 2.336 test_loss: 2.390
[36,    60] loss: 2.295 test_loss: 2.210
[37,    60] loss: 2.278 test_loss: 2.227
[38,    60] loss: 2.273 test_loss: 2.222
[39,    60] loss: 2.255 test_loss: 2.332
[40,    60] loss: 2.235 test_loss: 2.204
[41,    60] loss: 2.233 test_loss: 2.109
[42,    60] loss: 2.214 test_loss: 2.166
[43,    60] loss: 2.189 test_loss: 2.126
[44,    60] loss: 2.168 test_loss: 2.177
[45,    60] loss: 2.165 test_loss: 2.170
[46,    60] loss: 2.144 test_loss: 2.210
[47,    60] loss: 2.138 test_loss: 2.140
[48,    60] loss: 2.123 test_loss: 2.066
[49,    60] loss: 2.113 test_loss: 2.029
[50,    60] loss: 2.114 test_loss: 2.107
[51,    60] loss: 2.085 test_loss: 2.036
[52,    60] loss: 2.081 test_loss: 2.090
[53,    60] loss: 2.071 test_loss: 2.140
[54,    60] loss: 2.075 test_loss: 2.208
[55,    60] loss: 2.061 test_loss: 2.091
[56,    60] loss: 2.053 test_loss: 2.006
[57,    60] loss: 2.030 test_loss: 2.012
```

```
[58,    60] loss: 2.033 test_loss: 2.026
[59,    60] loss: 2.027 test_loss: 2.071
[60,    60] loss: 2.031 test_loss: 1.967
[61,    60] loss: 2.001 test_loss: 1.956
[62,    60] loss: 2.012 test_loss: 1.955
[63,    60] loss: 2.004 test_loss: 1.931
[64,    60] loss: 1.995 test_loss: 1.933
[65,    60] loss: 2.003 test_loss: 2.020
[66,    60] loss: 1.974 test_loss: 1.986
[67,    60] loss: 1.984 test_loss: 1.984
[68,    60] loss: 1.986 test_loss: 1.893
[69,    60] loss: 1.957 test_loss: 1.992
[70,    60] loss: 1.962 test_loss: 1.902
[71,    60] loss: 1.937 test_loss: 1.984
[72,    60] loss: 1.934 test_loss: 1.977
[73,    60] loss: 1.943 test_loss: 2.013
[74,    60] loss: 1.927 test_loss: 1.898
[75,    60] loss: 1.931 test_loss: 2.019
[76,    60] loss: 1.929 test_loss: 1.980
[77,    60] loss: 1.918 test_loss: 1.910
[78,    60] loss: 1.908 test_loss: 1.953
[79,    60] loss: 1.908 test_loss: 1.889
[80,    60] loss: 1.925 test_loss: 1.911
[81,    60] loss: 1.917 test_loss: 1.893
[82,    60] loss: 1.901 test_loss: 1.975
[83,    60] loss: 1.894 test_loss: 1.895
[84,    60] loss: 1.884 test_loss: 1.887
[85,    60] loss: 1.880 test_loss: 1.940
[86,    60] loss: 1.894 test_loss: 1.930
[87,    60] loss: 1.886 test_loss: 1.848
[88,    60] loss: 1.889 test_loss: 1.914
[89,    60] loss: 1.893 test_loss: 1.919
[90,    60] loss: 1.877 test_loss: 1.888
[91,    60] loss: 1.872 test_loss: 1.938
[92,    60] loss: 1.880 test_loss: 1.880
[93,    60] loss: 1.869 test_loss: 1.888
[94,    60] loss: 1.867 test_loss: 1.809
[95,    60] loss: 1.859 test_loss: 1.882
[96,    60] loss: 1.859 test_loss: 1.823
[97,    60] loss: 1.866 test_loss: 1.815
[98,    60] loss: 1.847 test_loss: 1.784
[99,    60] loss: 1.882 test_loss: 1.845
[100,    60] loss: 1.854 test_loss: 1.833
[101,    60] loss: 1.865 test_loss: 1.834
[102,    60] loss: 1.855 test_loss: 1.854
[103,    60] loss: 1.863 test_loss: 1.842
[104,    60] loss: 1.846 test_loss: 1.894
[105,    60] loss: 1.859 test_loss: 1.875
[106,    60] loss: 1.865 test_loss: 1.857
[107,    60] loss: 1.841 test_loss: 1.835
[108,    60] loss: 1.840 test_loss: 1.777
[109,    60] loss: 1.849 test_loss: 1.816
[110,    60] loss: 1.832 test_loss: 1.779
[111,    60] loss: 1.843 test_loss: 1.781
[112,    60] loss: 1.837 test_loss: 1.871
[113,    60] loss: 1.829 test_loss: 1.765
[114,    60] loss: 1.867 test_loss: 1.797
```

```
[115,    60] loss: 1.833 test_loss: 1.919
[116,    60] loss: 1.841 test_loss: 1.761
[117,    60] loss: 1.843 test_loss: 1.829
[118,    60] loss: 1.826 test_loss: 1.721
[119,    60] loss: 1.845 test_loss: 1.811
[120,    60] loss: 1.814 test_loss: 1.819
[121,    60] loss: 1.831 test_loss: 1.800
[122,    60] loss: 1.819 test_loss: 1.804
[123,    60] loss: 1.825 test_loss: 1.795
[124,    60] loss: 1.824 test_loss: 1.955
[125,    60] loss: 1.822 test_loss: 1.920
[126,    60] loss: 1.818 test_loss: 1.834
[127,    60] loss: 1.812 test_loss: 1.817
[128,    60] loss: 1.836 test_loss: 1.799
[129,    60] loss: 1.816 test_loss: 1.866
[130,    60] loss: 1.828 test_loss: 1.801
[131,    60] loss: 1.826 test_loss: 1.845
[132,    60] loss: 1.837 test_loss: 1.779
[133,    60] loss: 1.812 test_loss: 1.777
[134,    60] loss: 1.834 test_loss: 1.857
[135,    60] loss: 1.814 test_loss: 1.771
[136,    60] loss: 1.832 test_loss: 1.747
[137,    60] loss: 1.813 test_loss: 1.729
[138,    60] loss: 1.822 test_loss: 1.809
[139,    60] loss: 1.813 test_loss: 1.832
[140,    60] loss: 1.821 test_loss: 1.785
[141,    60] loss: 1.825 test_loss: 1.816
[142,    60] loss: 1.812 test_loss: 1.827
[143,    60] loss: 1.823 test_loss: 1.797
[144,    60] loss: 1.821 test_loss: 1.777
[145,    60] loss: 1.804 test_loss: 1.826
[146,    60] loss: 1.802 test_loss: 1.775
[147,    60] loss: 1.828 test_loss: 1.823
[148,    60] loss: 1.817 test_loss: 1.800
[149,    60] loss: 1.802 test_loss: 1.768
[150,    60] loss: 1.802 test_loss: 1.789
[151,    60] loss: 1.823 test_loss: 1.759
[152,    60] loss: 1.800 test_loss: 1.813
[153,    60] loss: 1.801 test_loss: 1.771
[154,    60] loss: 1.825 test_loss: 1.788
[155,    60] loss: 1.817 test_loss: 1.763
[156,    60] loss: 1.822 test_loss: 1.768
[157,    60] loss: 1.806 test_loss: 1.921
[158,    60] loss: 1.786 test_loss: 1.734
[159,    60] loss: 1.791 test_loss: 1.810
[160,    60] loss: 1.801 test_loss: 1.864
[161,    60] loss: 1.808 test_loss: 1.842
[162,    60] loss: 1.815 test_loss: 1.819
[163,    60] loss: 1.808 test_loss: 1.882
[164,    60] loss: 1.805 test_loss: 1.803
[165,    60] loss: 1.823 test_loss: 1.719
[166,    60] loss: 1.803 test_loss: 1.734
[167,    60] loss: 1.816 test_loss: 1.779
[168,    60] loss: 1.803 test_loss: 1.907
[169,    60] loss: 1.800 test_loss: 1.791
[170,    60] loss: 1.810 test_loss: 1.783
[171,    60] loss: 1.812 test_loss: 1.826
```

```
[172,    60] loss: 1.795 test_loss: 1.793
[173,    60] loss: 1.808 test_loss: 1.803
[174,    60] loss: 1.804 test_loss: 1.828
[175,    60] loss: 1.802 test_loss: 1.862
[176,    60] loss: 1.817 test_loss: 1.823
[177,    60] loss: 1.802 test_loss: 1.813
[178,    60] loss: 1.810 test_loss: 1.874
[179,    60] loss: 1.799 test_loss: 1.770
[180,    60] loss: 1.803 test_loss: 1.870
[181,    60] loss: 1.808 test_loss: 1.755
[182,    60] loss: 1.815 test_loss: 1.705
[183,    60] loss: 1.805 test_loss: 1.803
[184,    60] loss: 1.812 test_loss: 1.924
[185,    60] loss: 1.806 test_loss: 1.791
[186,    60] loss: 1.802 test_loss: 1.755
[187,    60] loss: 1.810 test_loss: 1.796
[188,    60] loss: 1.810 test_loss: 1.803
[189,    60] loss: 1.808 test_loss: 1.783
[190,    60] loss: 1.801 test_loss: 1.762
[191,    60] loss: 1.794 test_loss: 1.836
[192,    60] loss: 1.803 test_loss: 1.796
[193,    60] loss: 1.815 test_loss: 1.729
[194,    60] loss: 1.805 test_loss: 1.793
[195,    60] loss: 1.812 test_loss: 1.768
[196,    60] loss: 1.801 test_loss: 1.751
[197,    60] loss: 1.808 test_loss: 1.763
[198,    60] loss: 1.810 test_loss: 1.807
[199,    60] loss: 1.795 test_loss: 1.880
[200,    60] loss: 1.804 test_loss: 1.801
Finished Training
```






In [9]:
```
torch.save(model.state_dict(), "res18b" + str(batchsize) + "r" + str(rate) +
"e" + str(epochs) + ".statedict")
```

In [11]:
```
with torch.no_grad():
    d = next(testloader.__iter__())
    images, labels = d
    outputs = model(images)

    print(trainset.mean)
    print(outputs[0])
    print(labels[0])
    print(np.linalg.norm((labels[0] - outputs[0]).to('cpu').numpy()))
```

```
[[-57.42849171  43.50585649  81.09827847]
 [-57.10330025  40.76217867  80.77729131]
 [-56.4013172   36.99098729  79.87123442]
 ...
 [ 54.11570628 -43.99206555  66.38817782]
 [ 55.6071899  -46.20434955  60.50756982]
 [ 56.74632444 -47.9137754   53.69290953]]
tensor([-0.8509, -1.7231, -4.8252,  ..., -2.9921,  2.7223,  0.9377], device
='cuda:0')
tensor([-0.5952, -1.8055, -7.2621,  ..., -1.7967,  2.6882,  0.4817], device
='cuda:0')
56.56403
```