Watson Studio democratizes machine learning and deep learning to accelerate infusion of AI in your business to drive innovation. Watson Studio provides a suite of tools and a collaborative environment for data scientists, developers and domain experts.

COGNITIVE CLASS

Pre-trained-Models with PyTorch ¶

In this lab, we will use pre-trained models to classify between the negative and positive samples; you will be provided with the dataset object. The particular pre-trained model will be resnet18; you will have three questions:

- change the output layer
- train the model
- identify several misclassified samples

You will take several screenshots of your work and share your notebook.

Table of Contents

1. Download Data
2. Imports and Auxiliary Functions
3. Dataset Class
4. Question 1
5. Question 2
6. Question 3

Estimated Time Needed: 120 min

Mount your Google Drive files by running the following code snippet

from google.colab import drive drive.mount('/content/drive')

Change current working directory

%cd "/content/drive/My Drive/Colab Notebooks/data"# Verify the current working directory %pwd# List the content in given directory path !ls "/content/drive/My Drive/Colab Notebooks/data"

Download Data

Download the dataset and unzip the files in your data directory, unlike the other labs, all the data will be deleted after you close the lab, this may take some time:

#!wget https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-data/CognitiveClass/DL0321EN/data/images/Positive_tensors.zip #!unzip -q Positive_tensors.zip #! wget https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-data/CognitiveClass/DL0321EN/data/images/Negative_tensors.zip #!unzip -q Negative_tensors.zip

We will install torchvision:

#!pip install torchvision

In [1]:
```
import os

current_directory_path = os.getcwd()
current_directory_path

# Path in local machine
# full input corpus
data_path = current_directory_path + "/../../../../../../Downloads/zTaiVeMay/data/"
model_path = "./model/"

#data_path = "./data-capstone/"  # small dataset

#os.path.exists(data_path)
!ls $data_path
```

```
Negative_tensors
Negative_tensors.zip
Positive_tensors
Positive_tensors.zip
```

Verify the number of files in input corpus

In [2]:
```
#data_path = "/content/drive/My Drive/Colab Notebooks/data/"
#model_path = "/content/drive/My Drive/Colab Notebooks/model/"

output_dir = data_path + "/Positive_tensors"
if os.path.exists(output_dir):
    print("Number of files in directory '{}' : {}".format(output_dir, len([name for name in os.listdir(output_dir)])))

output_dir = data_path + "/Negative_tensors"
if os.path.exists(output_dir):
    print("Number of files in directory '{}' : {}".format(output_dir, len([name for name in os.listdir(output_dir)])))
```

```
Number of files in directory 'C:\Users\TienLe\Develops\Solution\CoBan\Solution\notebook\AICapstone/../../../../../../Downloads/zTaiVeMay/data//Positive_tensors' : 20000
Number of files in directory 'C:\Users\TienLe\Develops\Solution\CoBan\Solution\notebook\AICapstone/../../../../../../Downloads/zTaiVeMay/data//Negative_tensors' : 20000
```

Imports and Auxiliary Functions

The following are the libraries we are going to use for this lab. The `torch.manual_seed()` is for forcing the random function to give the same number every time we try to recompile it.

In [3]:
```
# These are the libraries will be used for this lab.
import torchvision.models as models
from PIL import Image
import pandas
from torchvision import transforms
import torch.nn as nn
import time
import torch
import matplotlib.pylab as plt
import numpy as np
from torch.utils.data import Dataset, DataLoader
import h5py
import os
import glob
torch.manual_seed(0)
```

Out[3]:
```
<torch._C.Generator at 0x24a4b901190>
```

In [4]:
```
from matplotlib.pyplot import imshow
import matplotlib.pylab as plt
from PIL import Image
import pandas as pd
import os
```

In [5]:
```
print("PyTorch version : ", torch.__version__)
```

```
PyTorch version :  1.3.0
```

Change Default Device - GPU / CPU

In [6]:
```
def get_default_device():
  """ Pick GPU if available else CPU. In Google Colab, we should change para
meter "Hardware Accelerator" in menu item "Edit > Notebook Settings"
  """
  if torch.cuda.is_available():
    print("Using GPU")
    return torch.device("cuda")

  print("Using CPU")
  return torch.device("cpu")
# end def

use_cuda = True   # Use GPU
device = torch.device("cpu")
if use_cuda:
  device = get_default_device()
```

```
Using CPU
```

Dataset Class

This dataset class is essentially the same dataset you build in the previous section, but to speed things up, we are going to use tensors instead of jpeg images. Therefor for each iteration, you will skip the reshape step, conversion step to tensors and normalization step.

In [7]:
```python
# Create your own dataset object

class Dataset(Dataset):

    # Constructor
    def __init__(self, transform=None, train=True):
        directory = data_path
        positive = "Positive_tensors"
        negative = "Negative_tensors"

        positive_file_path = os.path.join(directory, positive)
        negative_file_path = os.path.join(directory, negative)
        positive_files = [os.path.join(positive_file_path, file) for file in
os.listdir(positive_file_path) if file.endswith(".pt")]
        negative_files = [os.path.join(negative_file_path, file) for file in
os.listdir(negative_file_path) if file.endswith(".pt")]

        number_of_samples = len(positive_files)+len(negative_files)

        self.all_files = [None]*number_of_samples
        self.all_files[::2] = positive_files
        self.all_files[1::2] = negative_files

        # The transform is goint to be used on image
        self.transform = transform

        #torch.LongTensor
        self.Y = torch.zeros([number_of_samples]).type(torch.LongTensor)
        self.Y[::2] = 1
        self.Y[1::2] = 0

        NUM_OF_ITEMS_FOR_SPLITTING = 30000
        if train:
            self.all_files = self.all_files[:NUM_OF_ITEMS_FOR_SPLITTING]
            self.Y = self.Y[:NUM_OF_ITEMS_FOR_SPLITTING]
            self.len = len(self.all_files)
        else:
            self.all_files = self.all_files[NUM_OF_ITEMS_FOR_SPLITTING:]
            self.Y = self.Y[NUM_OF_ITEMS_FOR_SPLITTING:]
            self.len = len(self.all_files)

    # Get the length
    def __len__(self):
        return self.len

    # Getter
    def __getitem__(self, idx):
        image = torch.load(self.all_files[idx])
        y = self.Y[idx]

        # If there is any transform method, apply it onto the image
        if self.transform:
            image = self.transform(image)
        return image, y

print("done")
```

```
done
```

We create two dataset objects, one for the training data and one for the validation data.

In [8]:
```
train_dataset = Dataset(train=True)
validation_dataset = Dataset(train=False)
print("done")
```

```
done
```

Question 1

Prepare a pre-trained resnet18 model :

Step 1: Load the pre-trained model `resnet18` Set the parameter `pretrained` to true:

In [9]:
```
# Step 1: Load the pre-trained model resnet18
import torchvision.models as models

model = models.resnet18(pretrained=True)
```

In [10]:
```
model = model.to(device)
model
```

Out[10]:
```
ResNet(
  (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3),
bias=False)
  (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running
_stats=True)
  (relu): ReLU(inplace=True)
  (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_
mode=False)
  (layer1): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_run
ning_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_run
ning_stats=True)
    )
    (1): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_run
ning_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_run
ning_stats=True)
    )
  )
  (layer2): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=
(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_ru
nning_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_ru
nning_stats=True)
      (downsample): Sequential(
        (0): Conv2d(64, 128, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_ru
nning_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_ru
nning_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_ru
nning_stats=True)
    )
  )
  (layer3): Sequential(
    (0): BasicBlock(
```

```
      (conv1): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2), padding=
(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_ru
nning_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_ru
nning_stats=True)
      (downsample): Sequential(
        (0): Conv2d(128, 256, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_ru
nning_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_ru
nning_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_ru
nning_stats=True)
    )
  )
  (layer4): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(256, 512, kernel_size=(3, 3), stride=(2, 2), padding=
(1, 1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_ru
nning_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_ru
nning_stats=True)
      (downsample): Sequential(
        (0): Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_ru
nning_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_ru
nning_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_ru
nning_stats=True)
    )
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
  (fc): Linear(in_features=512, out_features=1000, bias=True)
)
```

Step 2: Set the attribute `requires_grad` to `False`. As a result, the parameters will not be affected by training.

In [11]:

```
# Step 2: Set the parameter cannot be trained for the pre-trained model
for param in model.parameters():
    param.requires_grad = False
```

`resnet18` is used to classify 1000 different objects; as a result, the last layer has 1000 outputs. The 512 inputs come from the fact that the previously hidden layer has 512 outputs.

Step 3: Replace the output layer `model.fc` of the neural network with a `nn.Linear` object, to classify 2 different classes. For the parameters `in_features` remember the last hidden layer has 512 neurons.

In [12]:

```
# Dimension of last layer
d_hidden = 512
d_out = 2

model.fc = nn.Linear(d_hidden, d_out)  # replace the output layer
```

Print out the model in order to show whether you get the correct answer.
(Your peer reviewer is going to mark based on what you print here.)

In [13]:
```
print(model)
```

```
ResNet(
  (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), 
bias=False)
  (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running
_stats=True)
  (relu): ReLU(inplace=True)
  (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_
mode=False)
  (layer1): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 
1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_run
ning_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 
1), bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_run
ning_stats=True)
    )
    (1): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 
1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_run
ning_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 
1), bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_run
ning_stats=True)
    )
  )
  (layer2): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=
(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_ru
nning_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_ru
nning_stats=True)
      (downsample): Sequential(
        (0): Conv2d(64, 128, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_ru
nning_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_ru
nning_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_ru
nning_stats=True)
    )
  )
  (layer3): Sequential(
    (0): BasicBlock(
```
/

```
      (conv1): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2), padding=
(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_ru
nning_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_ru
nning_stats=True)
      (downsample): Sequential(
        (0): Conv2d(128, 256, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_ru
nning_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_ru
nning_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_ru
nning_stats=True)
    )
  )
  (layer4): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(256, 512, kernel_size=(3, 3), stride=(2, 2), padding=
(1, 1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_ru
nning_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_ru
nning_stats=True)
      (downsample): Sequential(
        (0): Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_ru
nning_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_ru
nning_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_ru
nning_stats=True)
    )
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
  (fc): Linear(in_features=512, out_features=2, bias=True)
)
```

Question 2: Train the Model

/

In this question you will train your, model:

Step 1: Create a cross entropy criterion function

In [14]:
```
# Step 1: Create the loss function
criterion = nn.CrossEntropyLoss()
```

Step 2: Create a training loader and validation loader object, the batch size should have 100 samples each.

In [15]:
```
batch_size = 100

train_loader = torch.utils.data.DataLoader(dataset=train_dataset, batch_size
=batch_size)
validation_loader = torch.utils.data.DataLoader(dataset=validation_dataset,
batch_size=batch_size)
```

Step 3: Use the following optimizer to minimize the loss

In [16]:
```
# Verify the parameters whose requires_grad are True
for param in model.parameters():
  if param.requires_grad:
    print(param)
```

```
Parameter containing:
tensor([[ 0.0342, -0.0255,  0.0249,  ...,  0.0130, -0.0365, -0.0426],
        [ 0.0411, -0.0204,  0.0384,  ...,  0.0156, -0.0340,  0.0069]],
       requires_grad=True)
Parameter containing:
tensor([0.0074, 0.0091], requires_grad=True)
```

In [17]:
```
optimizer = torch.optim.Adam([parameters for parameters in model.parameters
() if parameters.requires_grad],
                             lr=0.001)
```

Complete the following code to calculate the accuracy on the validation data for one epoch; this should take about 45 minutes. Make sure you calculate the accuracy on the validation data.

```
In [18]: n_epochs = 1
loss_list = []
accuracy_list = []
accuracy = 0
correct = 0
N_test = len(validation_dataset)
N_train = len(train_dataset)
start_time = time.time()
#n_epochs

print("Number of items in training set : ", N_train)
print("Number of items in testing set : ", N_test)

running_loss = 0
start_time = time.time()
for epoch in range(n_epochs):
    for i, (x, y) in enumerate(train_loader):
        print('-' * 30)
        print('Iteration (train phase) {}/{}'.format(i+1, int(N_train/batch_
size)))
        i_start_time = time.time()

        x = x.to(device)
        y = y.to(device)

        # set model to train
        model.train()

        # clear gradient
        optimizer.zero_grad()

        # make a prediction
        z = model(x)

        # calculate loss
        loss = criterion(z, y)
        # loss.requires_grad = True

        # calculate gradients of parameters
        loss.backward()

        # update parameters
        optimizer.step()

        loss_list.append(loss.data)
        print("Finished in {} (s)".format(time.time()-i_start_time))
    # end for

    correct=0
    for i, (x_test, y_test) in enumerate(validation_loader):
        print('-' * 30)
        print('Iteration (validation phase) {}/{}'.format(i+1, int(N_test/ba
tch_size)))
        i_start_time = time.time()

        x_test = x_test.to(device)
        y_test = y_test.to(device)

        # set model to eval
        model.eval()

        # make a prediction
```

```
        z = model(x_test)

        # find max
        _, yhat = torch.max(z.data, 1)


        #Calculate misclassified  samples in mini-batch
        #hint +=(yhat==y_test).sum().item()
        correct += (yhat==y_test).sum().item()

        print("Finished in {} (s)".format(time.time()-i_start_time))
    # end for

    accuracy=correct/N_test
    print("Epoch %d - accuracy: %.3f" % (epoch+1, accuracy))

    accuracy_list.append(accuracy)
    print("-" * 72)

    # Save model
    model_file_path = model_path + "resnet18_trained_model_epoch_{}.pth".for
mat(epoch+1)
    torch.save(model.state_dict(), model_file_path)

    # Duration for epoch
    print("Finished epoch {} in {} (s).".format(epoch+1, time.time()-start_t
ime))
# end for
```

```
Number of items in training set :  30000
Number of items in testing set :  10000
------------------------------
Iteration (train phase) 1/300
Finished in 6.096280813217163 (s)
------------------------------
Iteration (train phase) 2/300
Finished in 5.291006803512573 (s)
------------------------------
Iteration (train phase) 3/300
Finished in 5.274720907211304 (s)
------------------------------
Iteration (train phase) 4/300
Finished in 5.415050268173218 (s)
------------------------------
Iteration (train phase) 5/300
Finished in 5.065397500991821 (s)
------------------------------
Iteration (train phase) 6/300
Finished in 5.100115060806274 (s)
------------------------------
Iteration (train phase) 7/300
Finished in 6.011878252029419 (s)
------------------------------
Iteration (train phase) 8/300
Finished in 5.400213241577148 (s)
------------------------------
Iteration (train phase) 9/300
Finished in 5.936899423599243 (s)
------------------------------
Iteration (train phase) 10/300
Finished in 5.564206838607788 (s)
------------------------------
Iteration (train phase) 11/300
Finished in 5.352254152297974 (s)
------------------------------
Iteration (train phase) 12/300
Finished in 5.678906440734863 (s)
------------------------------
Iteration (train phase) 13/300
Finished in 5.841312646865845 (s)
------------------------------
Iteration (train phase) 14/300
Finished in 5.576798915863037 (s)
------------------------------
Iteration (train phase) 15/300
Finished in 6.100056409835815 (s)
------------------------------
Iteration (train phase) 16/300
Finished in 5.649114608764648 (s)
------------------------------
Iteration (train phase) 17/300
Finished in 5.451741695404053 (s)
------------------------------
Iteration (train phase) 18/300
Finished in 5.183423757553101 (s)
------------------------------
Iteration (train phase) 19/300
Finished in 5.700443744659424 (s)
------------------------------
Iteration (train phase) 20/300
Finished in 5.385800361633301 (s)
------------------------------
```

/

```
Iteration (train phase) 21/300
Finished in 5.393368244171143 (s)
------------------------------
Iteration (train phase) 22/300
Finished in 5.949162721633911 (s)
------------------------------
Iteration (train phase) 23/300
Finished in 7.269596576690674 (s)
------------------------------
Iteration (train phase) 24/300
Finished in 9.072009563446045 (s)
------------------------------
Iteration (train phase) 25/300
Finished in 5.535938024520874 (s)
------------------------------
Iteration (train phase) 26/300
Finished in 5.935414552688599 (s)
------------------------------
Iteration (train phase) 27/300
Finished in 6.786778211593628 (s)
------------------------------
Iteration (train phase) 28/300
Finished in 6.188758611679077 (s)
------------------------------
Iteration (train phase) 29/300
Finished in 6.571410417556763 (s)
------------------------------
Iteration (train phase) 30/300
Finished in 6.635863542556763 (s)
------------------------------
Iteration (train phase) 31/300
Finished in 7.159437417984009 (s)
------------------------------
Iteration (train phase) 32/300
Finished in 7.692087650299072 (s)
------------------------------
Iteration (train phase) 33/300
Finished in 9.985524415969849 (s)
------------------------------
Iteration (train phase) 34/300
Finished in 6.561969041824341 (s)
------------------------------
Iteration (train phase) 35/300
Finished in 5.990538597106934 (s)
------------------------------
Iteration (train phase) 36/300
Finished in 6.647655248641968 (s)
------------------------------
Iteration (train phase) 37/300
Finished in 7.714814901351929 (s)
------------------------------
Iteration (train phase) 38/300
Finished in 7.829302072525024 (s)
------------------------------
Iteration (train phase) 39/300
Finished in 5.970057487487793 (s)
------------------------------
Iteration (train phase) 40/300
Finished in 5.603570222854614 (s)
------------------------------
Iteration (train phase) 41/300
Finished in 5.557733535766602 (s)
------------------------------
```

```
Iteration (train phase) 42/300
Finished in 8.772055387496948 (s)
------------------------------
Iteration (train phase) 43/300
Finished in 6.333737373352051 (s)
------------------------------
Iteration (train phase) 44/300
Finished in 6.644757986068726 (s)
------------------------------
Iteration (train phase) 45/300
Finished in 8.870732069015503 (s)
------------------------------
Iteration (train phase) 46/300
Finished in 6.0540220737457275 (s)
------------------------------
Iteration (train phase) 47/300
Finished in 5.632509469985962 (s)
------------------------------
Iteration (train phase) 48/300
Finished in 5.727102041244507 (s)
------------------------------
Iteration (train phase) 49/300
Finished in 6.580753326416016 (s)
------------------------------
Iteration (train phase) 50/300
Finished in 6.089967250823975 (s)
------------------------------
Iteration (train phase) 51/300
Finished in 6.28475284576416 (s)
------------------------------
Iteration (train phase) 52/300
Finished in 6.308919668197632 (s)
------------------------------
Iteration (train phase) 53/300
Finished in 5.768669128417969 (s)
------------------------------
Iteration (train phase) 54/300
Finished in 6.2007856369018555 (s)
------------------------------
Iteration (train phase) 55/300
Finished in 5.8240156173706055 (s)
------------------------------
Iteration (train phase) 56/300
Finished in 5.916966915130615 (s)
------------------------------
Iteration (train phase) 57/300
Finished in 5.887775897979736 (s)
------------------------------
Iteration (train phase) 58/300
Finished in 6.281780004501343 (s)
------------------------------
Iteration (train phase) 59/300
Finished in 6.531958818435669 (s)
------------------------------
Iteration (train phase) 60/300
Finished in 6.125633478164673 (s)
------------------------------
Iteration (train phase) 61/300
Finished in 5.887486457824707 (s)
------------------------------
Iteration (train phase) 62/300
Finished in 5.966558933258057 (s)
------------------------------
```

```
Iteration (train phase) 63/300
Finished in 6.187785387039185 (s)
------------------------------
Iteration (train phase) 64/300
Finished in 6.241070508956909 (s)
------------------------------
Iteration (train phase) 65/300
Finished in 5.978247880935669 (s)
------------------------------
Iteration (train phase) 66/300
Finished in 5.9279937744140625 (s)
------------------------------
Iteration (train phase) 67/300
Finished in 5.866018295288086 (s)
------------------------------
Iteration (train phase) 68/300
Finished in 6.288495063781738 (s)
------------------------------
Iteration (train phase) 69/300
Finished in 6.271265268325806 (s)
------------------------------
Iteration (train phase) 70/300
Finished in 6.225990295410156 (s)
------------------------------
Iteration (train phase) 71/300
Finished in 6.094849348068237 (s)
------------------------------
Iteration (train phase) 72/300
Finished in 5.74773645401001 (s)
------------------------------
Iteration (train phase) 73/300
Finished in 6.079614877700806 (s)
------------------------------
Iteration (train phase) 74/300
Finished in 5.775083065032959 (s)
------------------------------
Iteration (train phase) 75/300
Finished in 5.977657079696655 (s)
------------------------------
Iteration (train phase) 76/300
Finished in 6.141481637954712 (s)
------------------------------
Iteration (train phase) 77/300
Finished in 6.29436993598938 (s)
------------------------------
Iteration (train phase) 78/300
Finished in 5.847611904144287 (s)
------------------------------
Iteration (train phase) 79/300
Finished in 6.041842222213745 (s)
------------------------------
Iteration (train phase) 80/300
Finished in 5.823960542678833 (s)
------------------------------
Iteration (train phase) 81/300
Finished in 6.095324754714966 (s)
------------------------------
Iteration (train phase) 82/300
Finished in 5.925950288772583 (s)
------------------------------
Iteration (train phase) 83/300
Finished in 6.078890085220337 (s)
------------------------------
```

/

```
Iteration (train phase) 84/300
Finished in 6.132469654083252 (s)
------------------------------
Iteration (train phase) 85/300
Finished in 5.7160561084747314 (s)
------------------------------
Iteration (train phase) 86/300
Finished in 6.052163362503052 (s)
------------------------------
Iteration (train phase) 87/300
Finished in 6.667537212371826 (s)
------------------------------
Iteration (train phase) 88/300
Finished in 6.415132284164429 (s)
------------------------------
Iteration (train phase) 89/300
Finished in 6.596692323684692 (s)
------------------------------
Iteration (train phase) 90/300
Finished in 6.3155317306518555 (s)
------------------------------
Iteration (train phase) 91/300
Finished in 5.919970273971558 (s)
------------------------------
Iteration (train phase) 92/300
Finished in 6.119394540786743 (s)
------------------------------
Iteration (train phase) 93/300
Finished in 5.667450189590454 (s)
------------------------------
Iteration (train phase) 94/300
Finished in 5.415510892868042 (s)
------------------------------
Iteration (train phase) 95/300
Finished in 5.429065465927124 (s)
------------------------------
Iteration (train phase) 96/300
Finished in 5.890809059143066 (s)
------------------------------
Iteration (train phase) 97/300
Finished in 5.57244873046875 (s)
------------------------------
Iteration (train phase) 98/300
Finished in 5.7650251388549805 (s)
------------------------------
Iteration (train phase) 99/300
Finished in 5.466165542602539 (s)
------------------------------
Iteration (train phase) 100/300
Finished in 5.497646331787109 (s)
------------------------------
Iteration (train phase) 101/300
Finished in 5.755497455596924 (s)
------------------------------
Iteration (train phase) 102/300
Finished in 5.8093695640563965 (s)
------------------------------
Iteration (train phase) 103/300
Finished in 5.833518743515015 (s)
------------------------------
Iteration (train phase) 104/300
Finished in 5.860225677490234 (s)
------------------------------
```

```
Iteration (train phase) 105/300
Finished in 6.2519755363464355 (s)
------------------------------
Iteration (train phase) 106/300
Finished in 5.866231679916382 (s)
------------------------------
Iteration (train phase) 107/300
Finished in 6.005195140838623 (s)
------------------------------
Iteration (train phase) 108/300
Finished in 6.217177152633667 (s)
------------------------------
Iteration (train phase) 109/300
Finished in 5.579162836074829 (s)
------------------------------
Iteration (train phase) 110/300
Finished in 5.861521005630493 (s)
------------------------------
Iteration (train phase) 111/300
Finished in 6.0043323040008545 (s)
------------------------------
Iteration (train phase) 112/300
Finished in 5.760803937911987 (s)
------------------------------
Iteration (train phase) 113/300
Finished in 6.0886147022247314 (s)
------------------------------
Iteration (train phase) 114/300
Finished in 5.70722508430481 (s)
------------------------------
Iteration (train phase) 115/300
Finished in 6.242507457733154 (s)
------------------------------
Iteration (train phase) 116/300
Finished in 5.611908197402954 (s)
------------------------------
Iteration (train phase) 117/300
Finished in 5.7134270668029785 (s)
------------------------------
Iteration (train phase) 118/300
Finished in 6.071218013763428 (s)
------------------------------
Iteration (train phase) 119/300
Finished in 5.454803705215454 (s)
------------------------------
Iteration (train phase) 120/300
Finished in 6.079091787338257 (s)
------------------------------
Iteration (train phase) 121/300
Finished in 5.469616413116455 (s)
------------------------------
Iteration (train phase) 122/300
Finished in 5.998258829116821 (s)
------------------------------
Iteration (train phase) 123/300
Finished in 5.5267815589904785 (s)
------------------------------
Iteration (train phase) 124/300
Finished in 5.993518590927124 (s)
------------------------------
Iteration (train phase) 125/300
Finished in 5.768739223480225 (s)
------------------------------
```

```
Iteration (train phase) 126/300
Finished in 6.061080455780029 (s)
------------------------------
Iteration (train phase) 127/300
Finished in 5.7711992263793945 (s)
------------------------------
Iteration (train phase) 128/300
Finished in 5.873793840408325 (s)
------------------------------
Iteration (train phase) 129/300
Finished in 6.02105975151062 (s)
------------------------------
Iteration (train phase) 130/300
Finished in 5.965259075164795 (s)
------------------------------
Iteration (train phase) 131/300
Finished in 6.0554468631744385 (s)
------------------------------
Iteration (train phase) 132/300
Finished in 5.427324056625366 (s)
------------------------------
Iteration (train phase) 133/300
Finished in 5.98679256439209 (s)
------------------------------
Iteration (train phase) 134/300
Finished in 5.8559699058532715 (s)
------------------------------
Iteration (train phase) 135/300
Finished in 5.974087476730347 (s)
------------------------------
Iteration (train phase) 136/300
Finished in 5.078265428543091 (s)
------------------------------
Iteration (train phase) 137/300
Finished in 5.531055688858032 (s)
------------------------------
Iteration (train phase) 138/300
Finished in 6.005217790603638 (s)
------------------------------
Iteration (train phase) 139/300
Finished in 5.776670455932617 (s)
------------------------------
Iteration (train phase) 140/300
Finished in 5.963910102844238 (s)
------------------------------
Iteration (train phase) 141/300
Finished in 6.018723487854004 (s)
------------------------------
Iteration (train phase) 142/300
Finished in 7.120019435882568 (s)
------------------------------
Iteration (train phase) 143/300
Finished in 6.397792816162109 (s)
------------------------------
Iteration (train phase) 144/300
Finished in 5.5633063316345215 (s)
------------------------------
Iteration (train phase) 145/300
Finished in 5.3338234424591064 (s)
------------------------------
Iteration (train phase) 146/300
Finished in 5.49887228012085 (s)
------------------------------
```

```
Iteration (train phase) 147/300
Finished in 5.103135585784912 (s)
------------------------------
Iteration (train phase) 148/300
Finished in 5.254668235778809 (s)
------------------------------
Iteration (train phase) 149/300
Finished in 5.2755303382873535 (s)
------------------------------
Iteration (train phase) 150/300
Finished in 5.841545104980469 (s)
------------------------------
Iteration (train phase) 151/300
Finished in 9.736835718154907 (s)
------------------------------
Iteration (train phase) 152/300
Finished in 7.886337518692017 (s)
------------------------------
Iteration (train phase) 153/300
Finished in 6.926172733306885 (s)
------------------------------
Iteration (train phase) 154/300
Finished in 9.399559020996094 (s)
------------------------------
Iteration (train phase) 155/300
Finished in 9.198180675506592 (s)
------------------------------
Iteration (train phase) 156/300
Finished in 8.488030433654785 (s)
------------------------------
Iteration (train phase) 157/300
Finished in 5.33989405632019 (s)
------------------------------
Iteration (train phase) 158/300
Finished in 5.329319477081299 (s)
------------------------------
Iteration (train phase) 159/300
Finished in 5.4890406131744385 (s)
------------------------------
Iteration (train phase) 160/300
Finished in 5.642747402191162 (s)
------------------------------
Iteration (train phase) 161/300
Finished in 5.537738561630249 (s)
------------------------------
Iteration (train phase) 162/300
Finished in 5.234736204147339 (s)
------------------------------
Iteration (train phase) 163/300
Finished in 5.459437847137451 (s)
------------------------------
Iteration (train phase) 164/300
Finished in 5.24165940284729 (s)
------------------------------
Iteration (train phase) 165/300
Finished in 5.396859169006348 (s)
------------------------------
Iteration (train phase) 166/300
Finished in 8.374356031417847 (s)
------------------------------
Iteration (train phase) 167/300
Finished in 5.8076088428497314 (s)
------------------------------
```

```
Iteration (train phase) 168/300
Finished in 7.338224649429321 (s)
------------------------------
Iteration (train phase) 169/300
Finished in 11.567046880722046 (s)
------------------------------
Iteration (train phase) 170/300
Finished in 6.571920394897461 (s)
------------------------------
Iteration (train phase) 171/300
Finished in 7.339379787445068 (s)
------------------------------
Iteration (train phase) 172/300
Finished in 7.062859296798706 (s)
------------------------------
Iteration (train phase) 173/300
Finished in 5.680261611938477 (s)
------------------------------
Iteration (train phase) 174/300
Finished in 5.75756311416626 (s)
------------------------------
Iteration (train phase) 175/300
Finished in 5.648321866989136 (s)
------------------------------
Iteration (train phase) 176/300
Finished in 5.933973789215088 (s)
------------------------------
Iteration (train phase) 177/300
Finished in 6.695343255996704 (s)
------------------------------
Iteration (train phase) 178/300
Finished in 5.744401693344116 (s)
------------------------------
Iteration (train phase) 179/300
Finished in 11.187623500823975 (s)
------------------------------
Iteration (train phase) 180/300
Finished in 6.367698431015015 (s)
------------------------------
Iteration (train phase) 181/300
Finished in 6.497730493545532 (s)
------------------------------
Iteration (train phase) 182/300
Finished in 5.435657739639282 (s)
------------------------------
Iteration (train phase) 183/300
Finished in 10.08749532699585 (s)
------------------------------
Iteration (train phase) 184/300
Finished in 7.05037260055542 (s)
------------------------------
Iteration (train phase) 185/300
Finished in 5.900851488113403 (s)
------------------------------
Iteration (train phase) 186/300
Finished in 5.379602670669556 (s)
------------------------------
Iteration (train phase) 187/300
Finished in 9.832329988479614 (s)
------------------------------
Iteration (train phase) 188/300
Finished in 6.583111047744751 (s)
------------------------------
```

```
Iteration (train phase) 189/300
Finished in 5.646862983703613 (s)
------------------------------
Iteration (train phase) 190/300
Finished in 7.470922231674194 (s)
------------------------------
Iteration (train phase) 191/300
Finished in 5.677636384963989 (s)
------------------------------
Iteration (train phase) 192/300
Finished in 6.842303991317749 (s)
------------------------------
Iteration (train phase) 193/300
Finished in 8.373835325241089 (s)
------------------------------
Iteration (train phase) 194/300
Finished in 5.965176343917847 (s)
------------------------------
Iteration (train phase) 195/300
Finished in 5.358096122741699 (s)
------------------------------
Iteration (train phase) 196/300
Finished in 5.760464429855347 (s)
------------------------------
Iteration (train phase) 197/300
Finished in 5.258824110031128 (s)
------------------------------
Iteration (train phase) 198/300
Finished in 7.717464447021484 (s)
------------------------------
Iteration (train phase) 199/300
Finished in 5.250720739364624 (s)
------------------------------
Iteration (train phase) 200/300
Finished in 5.348538875579834 (s)
------------------------------
Iteration (train phase) 201/300
Finished in 5.66180944442749 (s)
------------------------------
Iteration (train phase) 202/300
Finished in 5.297524452209473 (s)
------------------------------
Iteration (train phase) 203/300
Finished in 5.872174024581909 (s)
------------------------------
Iteration (train phase) 204/300
Finished in 5.412132263183594 (s)
------------------------------
Iteration (train phase) 205/300
Finished in 6.5067408084869385 (s)
------------------------------
Iteration (train phase) 206/300
Finished in 6.84074854850769 (s)
------------------------------
Iteration (train phase) 207/300
Finished in 6.763948440551758 (s)
------------------------------
Iteration (train phase) 208/300
Finished in 8.722794532775879 (s)
------------------------------
Iteration (train phase) 209/300
Finished in 7.294142961502075 (s)
------------------------------
```

```
Iteration (train phase) 210/300
Finished in 5.30419135093689 (s)
------------------------------
Iteration (train phase) 211/300
Finished in 5.466709852218628 (s)
------------------------------
Iteration (train phase) 212/300
Finished in 6.003605365753174 (s)
------------------------------
Iteration (train phase) 213/300
Finished in 6.413337707519531 (s)
------------------------------
Iteration (train phase) 214/300
Finished in 6.802649974822998 (s)
------------------------------
Iteration (train phase) 215/300
Finished in 6.514224052429199 (s)
------------------------------
Iteration (train phase) 216/300
Finished in 6.22156548500061 (s)
------------------------------
Iteration (train phase) 217/300
Finished in 8.891314029693604 (s)
------------------------------
Iteration (train phase) 218/300
Finished in 10.359421014785767 (s)
------------------------------
Iteration (train phase) 219/300
Finished in 5.43654990196228 (s)
------------------------------
Iteration (train phase) 220/300
Finished in 5.229981184005737 (s)
------------------------------
Iteration (train phase) 221/300
Finished in 7.1301116943359375 (s)
------------------------------
Iteration (train phase) 222/300
Finished in 9.384384632110596 (s)
------------------------------
Iteration (train phase) 223/300
Finished in 7.324803352355957 (s)
------------------------------
Iteration (train phase) 224/300
Finished in 9.055775165557861 (s)
------------------------------
Iteration (train phase) 225/300
Finished in 8.169331073760986 (s)
------------------------------
Iteration (train phase) 226/300
Finished in 6.247869968414307 (s)
------------------------------
Iteration (train phase) 227/300
Finished in 5.643316984176636 (s)
------------------------------
Iteration (train phase) 228/300
Finished in 5.380385637283325 (s)
------------------------------
Iteration (train phase) 229/300
Finished in 6.037240743637085 (s)
------------------------------
Iteration (train phase) 230/300
Finished in 5.93189811706543 (s)
------------------------------
```

```
Iteration (train phase) 231/300
Finished in 6.5027735233306885 (s)
------------------------------
Iteration (train phase) 232/300
Finished in 5.664273738861084 (s)
------------------------------
Iteration (train phase) 233/300
Finished in 7.014066696166992 (s)
------------------------------
Iteration (train phase) 234/300
Finished in 8.006085634231567 (s)
------------------------------
Iteration (train phase) 235/300
Finished in 6.640657424926758 (s)
------------------------------
Iteration (train phase) 236/300
Finished in 5.778226375579834 (s)
------------------------------
Iteration (train phase) 237/300
Finished in 6.294759035110474 (s)
------------------------------
Iteration (train phase) 238/300
Finished in 5.336493492126465 (s)
------------------------------
Iteration (train phase) 239/300
Finished in 5.909621000289917 (s)
------------------------------
Iteration (train phase) 240/300
Finished in 7.684089422225952 (s)
------------------------------
Iteration (train phase) 241/300
Finished in 7.334156036376953 (s)
------------------------------
Iteration (train phase) 242/300
Finished in 6.8365089893341064 (s)
------------------------------
Iteration (train phase) 243/300
Finished in 6.667696237564087 (s)
------------------------------
Iteration (train phase) 244/300
Finished in 6.047314167022705 (s)
------------------------------
Iteration (train phase) 245/300
Finished in 5.404781103134155 (s)
------------------------------
Iteration (train phase) 246/300
Finished in 5.294153690338135 (s)
------------------------------
Iteration (train phase) 247/300
Finished in 5.678259611129761 (s)
------------------------------
Iteration (train phase) 248/300
Finished in 5.360722303390503 (s)
------------------------------
Iteration (train phase) 249/300
Finished in 6.003880023956299 (s)
------------------------------
Iteration (train phase) 250/300
Finished in 6.808876991271973 (s)
------------------------------
Iteration (train phase) 251/300
Finished in 6.50197958946228 (s)
------------------------------
```

```
Iteration (train phase) 252/300
Finished in 6.181591749191284 (s)
------------------------------
Iteration (train phase) 253/300
Finished in 6.313189506530762 (s)
------------------------------
Iteration (train phase) 254/300
Finished in 5.07785177230835 (s)
------------------------------
Iteration (train phase) 255/300
Finished in 6.054868936538696 (s)
------------------------------
Iteration (train phase) 256/300
Finished in 7.540154933929443 (s)
------------------------------
Iteration (train phase) 257/300
Finished in 7.8357017040252686 (s)
------------------------------
Iteration (train phase) 258/300
Finished in 5.093387603759766 (s)
------------------------------
Iteration (train phase) 259/300
Finished in 7.356971263885498 (s)
------------------------------
Iteration (train phase) 260/300
Finished in 5.652416706085205 (s)
------------------------------
Iteration (train phase) 261/300
Finished in 5.332224369049072 (s)
------------------------------
Iteration (train phase) 262/300
Finished in 8.442997455596924 (s)
------------------------------
Iteration (train phase) 263/300
Finished in 5.418361663818359 (s)
------------------------------
Iteration (train phase) 264/300
Finished in 6.155961513519287 (s)
------------------------------
Iteration (train phase) 265/300
Finished in 5.105592727661133 (s)
------------------------------
Iteration (train phase) 266/300
Finished in 5.569375276565552 (s)
------------------------------
Iteration (train phase) 267/300
Finished in 5.229057550430298 (s)
------------------------------
Iteration (train phase) 268/300
Finished in 5.240102767944336 (s)
------------------------------
Iteration (train phase) 269/300
Finished in 5.450912952423096 (s)
------------------------------
Iteration (train phase) 270/300
Finished in 5.322052240371704 (s)
------------------------------
Iteration (train phase) 271/300
Finished in 5.082930564880371 (s)
------------------------------
Iteration (train phase) 272/300
Finished in 5.293299913406372 (s)
------------------------------
```

```
Iteration (train phase) 273/300
Finished in 5.340768098831177 (s)
------------------------------
Iteration (train phase) 274/300
Finished in 5.099598407745361 (s)
------------------------------
Iteration (train phase) 275/300
Finished in 5.204756736755371 (s)
------------------------------
Iteration (train phase) 276/300
Finished in 5.19101095199585 (s)
------------------------------
Iteration (train phase) 277/300
Finished in 5.211349248886108 (s)
------------------------------
Iteration (train phase) 278/300
Finished in 5.085863351821899 (s)
------------------------------
Iteration (train phase) 279/300
Finished in 5.1344475746154785 (s)
------------------------------
Iteration (train phase) 280/300
Finished in 5.378952741622925 (s)
------------------------------
Iteration (train phase) 281/300
Finished in 5.369287729263306 (s)
------------------------------
Iteration (train phase) 282/300
Finished in 5.192302703857422 (s)
------------------------------
Iteration (train phase) 283/300
Finished in 5.179342985153198 (s)
------------------------------
Iteration (train phase) 284/300
Finished in 5.136267900466919 (s)
------------------------------
Iteration (train phase) 285/300
Finished in 5.200512170791626 (s)
------------------------------
Iteration (train phase) 286/300
Finished in 5.039432525634766 (s)
------------------------------
Iteration (train phase) 287/300
Finished in 4.982022285461426 (s)
------------------------------
Iteration (train phase) 288/300
Finished in 5.057367563247681 (s)
------------------------------
Iteration (train phase) 289/300
Finished in 5.140556812286377 (s)
------------------------------
Iteration (train phase) 290/300
Finished in 5.159147500991821 (s)
------------------------------
Iteration (train phase) 291/300
Finished in 5.1426026821136475 (s)
------------------------------
Iteration (train phase) 292/300
Finished in 5.0284857749938965 (s)
------------------------------
Iteration (train phase) 293/300
Finished in 5.706006050109863 (s)
------------------------------
```

```
Iteration (train phase) 294/300
Finished in 5.534398794174194 (s)
------------------------------
Iteration (train phase) 295/300
Finished in 5.866415977478027 (s)
------------------------------
Iteration (train phase) 296/300
Finished in 5.289555072784424 (s)
------------------------------
Iteration (train phase) 297/300
Finished in 5.491362810134888 (s)
------------------------------
Iteration (train phase) 298/300
Finished in 6.023866415023804 (s)
------------------------------
Iteration (train phase) 299/300
Finished in 5.90197491645813 (s)
------------------------------
Iteration (train phase) 300/300
Finished in 5.944220781326294 (s)
------------------------------
Iteration (validation phase) 1/100
Finished in 5.099977016448975 (s)
------------------------------
Iteration (validation phase) 2/100
Finished in 5.046879053115845 (s)
------------------------------
Iteration (validation phase) 3/100
Finished in 5.066048622131348 (s)
------------------------------
Iteration (validation phase) 4/100
Finished in 5.091193437576294 (s)
------------------------------
Iteration (validation phase) 5/100
Finished in 5.369806289672852 (s)
------------------------------
Iteration (validation phase) 6/100
Finished in 5.217925310134888 (s)
------------------------------
Iteration (validation phase) 7/100
Finished in 5.272547245025635 (s)
------------------------------
Iteration (validation phase) 8/100
Finished in 4.870260953903198 (s)
------------------------------
Iteration (validation phase) 9/100
Finished in 5.068841934204102 (s)
------------------------------
Iteration (validation phase) 10/100
Finished in 4.773751258850098 (s)
------------------------------
Iteration (validation phase) 11/100
Finished in 4.9500861167907715 (s)
------------------------------
Iteration (validation phase) 12/100
Finished in 4.89833927154541 (s)
------------------------------
Iteration (validation phase) 13/100
Finished in 5.129556894302368 (s)
------------------------------
Iteration (validation phase) 14/100
Finished in 4.659253120422363 (s)
------------------------------
```

```
Iteration (validation phase) 15/100
Finished in 4.692891836166382 (s)
------------------------------
Iteration (validation phase) 16/100
Finished in 4.668203830718994 (s)
------------------------------
Iteration (validation phase) 17/100
Finished in 4.949375629425049 (s)
------------------------------
Iteration (validation phase) 18/100
Finished in 4.93558669090271 (s)
------------------------------
Iteration (validation phase) 19/100
Finished in 4.947063207626343 (s)
------------------------------
Iteration (validation phase) 20/100
Finished in 5.294849872589111 (s)
------------------------------
Iteration (validation phase) 21/100
Finished in 5.816081523895264 (s)
------------------------------
Iteration (validation phase) 22/100
Finished in 5.275944232940674 (s)
------------------------------
Iteration (validation phase) 23/100
Finished in 5.239357233047485 (s)
------------------------------
Iteration (validation phase) 24/100
Finished in 5.076871633529663 (s)
------------------------------
Iteration (validation phase) 25/100
Finished in 5.213708877563477 (s)
------------------------------
Iteration (validation phase) 26/100
Finished in 5.121939182281494 (s)
------------------------------
Iteration (validation phase) 27/100
Finished in 6.0119147300720215 (s)
------------------------------
Iteration (validation phase) 28/100
Finished in 5.38561749458313 (s)
------------------------------
Iteration (validation phase) 29/100
Finished in 5.416457176208496 (s)
------------------------------
Iteration (validation phase) 30/100
Finished in 5.021626234054565 (s)
------------------------------
Iteration (validation phase) 31/100
Finished in 5.1901726722717285 (s)
------------------------------
Iteration (validation phase) 32/100
Finished in 5.107229709625244 (s)
------------------------------
Iteration (validation phase) 33/100
Finished in 4.856112241744995 (s)
------------------------------
Iteration (validation phase) 34/100
Finished in 4.79146409034729 (s)
------------------------------
Iteration (validation phase) 35/100
Finished in 5.271399259567261 (s)
------------------------------
```

```
Iteration (validation phase) 36/100
Finished in 5.1068572998046875 (s)
------------------------------
Iteration (validation phase) 37/100
Finished in 5.039779186248779 (s)
------------------------------
Iteration (validation phase) 38/100
Finished in 5.02919864654541 (s)
------------------------------
Iteration (validation phase) 39/100
Finished in 5.743528127670288 (s)
------------------------------
Iteration (validation phase) 40/100
Finished in 5.651045560836792 (s)
------------------------------
Iteration (validation phase) 41/100
Finished in 6.639713287353516 (s)
------------------------------
Iteration (validation phase) 42/100
Finished in 7.045636892318726 (s)
------------------------------
Iteration (validation phase) 43/100
Finished in 4.891063690185547 (s)
------------------------------
Iteration (validation phase) 44/100
Finished in 6.47954535484314 (s)
------------------------------
Iteration (validation phase) 45/100
Finished in 5.052016735076904 (s)
------------------------------
Iteration (validation phase) 46/100
Finished in 4.792670726776123 (s)
------------------------------
Iteration (validation phase) 47/100
Finished in 5.361390829086304 (s)
------------------------------
Iteration (validation phase) 48/100
Finished in 5.7336461544036865 (s)
------------------------------
Iteration (validation phase) 49/100
Finished in 5.70596718788147 (s)
------------------------------
Iteration (validation phase) 50/100
Finished in 5.5974342823028564 (s)
------------------------------
Iteration (validation phase) 51/100
Finished in 7.418220043182373 (s)
------------------------------
Iteration (validation phase) 52/100
Finished in 6.984492778778076 (s)
------------------------------
Iteration (validation phase) 53/100
Finished in 5.965245962142944 (s)
------------------------------
Iteration (validation phase) 54/100
Finished in 5.23383259773254 (s)
------------------------------
Iteration (validation phase) 55/100
Finished in 5.116316080093384 (s)
------------------------------
Iteration (validation phase) 56/100
Finished in 8.716162919998169 (s)
------------------------------
```

/

```
Iteration (validation phase) 57/100
Finished in 5.20356559753418 (s)
------------------------------
Iteration (validation phase) 58/100
Finished in 4.8426690101623535 (s)
------------------------------
Iteration (validation phase) 59/100
Finished in 6.2191550731658936 (s)
------------------------------
Iteration (validation phase) 60/100
Finished in 5.22513747215271 (s)
------------------------------
Iteration (validation phase) 61/100
Finished in 5.125848293304443 (s)
------------------------------
Iteration (validation phase) 62/100
Finished in 4.9857399463653564 (s)
------------------------------
Iteration (validation phase) 63/100
Finished in 4.70170521736145 (s)
------------------------------
Iteration (validation phase) 64/100
Finished in 5.737640619277954 (s)
------------------------------
Iteration (validation phase) 65/100
Finished in 4.995473384857178 (s)
------------------------------
Iteration (validation phase) 66/100
Finished in 4.743335723876953 (s)
------------------------------
Iteration (validation phase) 67/100
Finished in 4.680486440658569 (s)
------------------------------
Iteration (validation phase) 68/100
Finished in 4.644129753112793 (s)
------------------------------
Iteration (validation phase) 69/100
Finished in 4.694386720657349 (s)
------------------------------
Iteration (validation phase) 70/100
Finished in 4.642921209335327 (s)
------------------------------
Iteration (validation phase) 71/100
Finished in 4.705610275268555 (s)
------------------------------
Iteration (validation phase) 72/100
Finished in 5.241948127746582 (s)
------------------------------
Iteration (validation phase) 73/100
Finished in 4.7558019161224365 (s)
------------------------------
Iteration (validation phase) 74/100
Finished in 4.787576198577881 (s)
------------------------------
Iteration (validation phase) 75/100
Finished in 4.848999261856079 (s)
------------------------------
Iteration (validation phase) 76/100
Finished in 4.988003253936768 (s)
------------------------------
Iteration (validation phase) 77/100
Finished in 5.319436073303223 (s)
------------------------------
```

```
Iteration (validation phase) 78/100
Finished in 4.871276140213013 (s)
------------------------------
Iteration (validation phase) 79/100
Finished in 5.046750545501709 (s)
------------------------------
Iteration (validation phase) 80/100
Finished in 4.742498159408569 (s)
------------------------------
Iteration (validation phase) 81/100
Finished in 4.642536878585815 (s)
------------------------------
Iteration (validation phase) 82/100
Finished in 5.060824871063232 (s)
------------------------------
Iteration (validation phase) 83/100
Finished in 5.099695444107056 (s)
------------------------------
Iteration (validation phase) 84/100
Finished in 5.080451488494873 (s)
------------------------------
Iteration (validation phase) 85/100
Finished in 5.236719369888306 (s)
------------------------------
Iteration (validation phase) 86/100
Finished in 5.70662784576416 (s)
------------------------------
Iteration (validation phase) 87/100
Finished in 4.829598665237427 (s)
------------------------------
Iteration (validation phase) 88/100
Finished in 4.687036037445068 (s)
------------------------------
Iteration (validation phase) 89/100
Finished in 5.235410451889038 (s)
------------------------------
Iteration (validation phase) 90/100
Finished in 4.87119197845459 (s)
------------------------------
Iteration (validation phase) 91/100
Finished in 5.319596290588379 (s)
------------------------------
Iteration (validation phase) 92/100
Finished in 4.893118619918823 (s)
------------------------------
Iteration (validation phase) 93/100
Finished in 5.6469128131866455 (s)
------------------------------
Iteration (validation phase) 94/100
Finished in 5.013097763061523 (s)
------------------------------
Iteration (validation phase) 95/100
Finished in 7.902310132980347 (s)
------------------------------
Iteration (validation phase) 96/100
Finished in 5.994199752807617 (s)
------------------------------
Iteration (validation phase) 97/100
Finished in 8.37505459785461 (s)
------------------------------
Iteration (validation phase) 98/100
Finished in 7.913624048233032 (s)
------------------------------
```

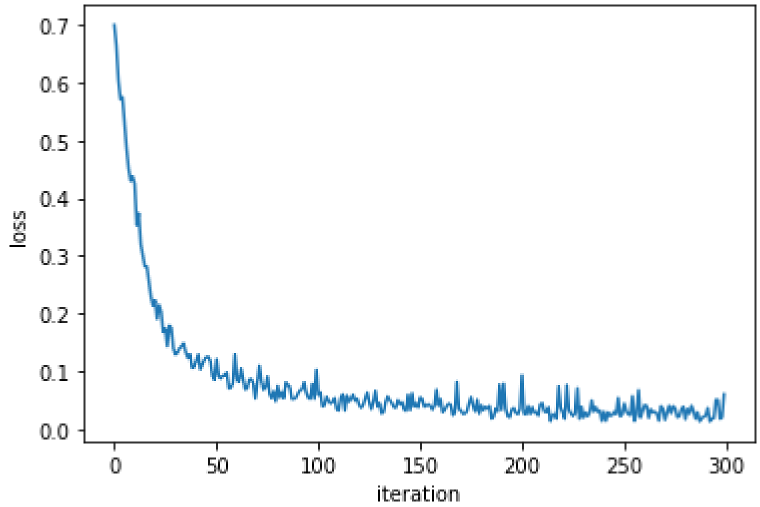
```
Iteration (validation phase) 99/100
Finished in 4.801631450653076 (s)
------------------------------
Iteration (validation phase) 100/100
Finished in 4.793451547622681 (s)
Epoch 1 - accuracy: 0.994
------------------------------------------------------------------------
Finished epoch 1 in 2609.5270249843597 (s).
```

Print out the Accuracy and plot the loss stored in the list `loss_list` for every iteration and take a screen shot.

In [19]:
```python
accuracy
```

Out[19]: 0.9943

In [20]:
```python
plt.plot(loss_list)
plt.xlabel("iteration")
plt.ylabel("loss")
plt.show()
```



Question 3:Find the misclassified samples

Identify the first four misclassified samples using the validation data:

```
In [22]: count = 0
max_num_of_items = 4  # first four mis-classified samples
validation_loader_batch_one = torch.utils.data.DataLoader(dataset=validation
_dataset, batch_size=1)

for i, (x_test, y_test) in enumerate(validation_loader_batch_one):
    # set model to eval
    model.eval()

    # make a prediction
    z = model(x_test)

    # find max
    _, yhat = torch.max(z.data, 1)

    # print mis-classified samples
    if yhat != y_test:
        print("Sample : {}; Expected Label: {}; Obtained Label: {}".format(s
tr(i), str(y_test), str(yhat)))
        count += 1
        if count >= max_num_of_items:
            break
    # end if
# end for
```

```
Sample : 22; Expected Label: tensor([1]); Obtained Label: tensor([0])
Sample : 101; Expected Label: tensor([0]); Obtained Label: tensor([1])
Sample : 182; Expected Label: tensor([1]); Obtained Label: tensor([0])
Sample : 213; Expected Label: tensor([0]); Obtained Label: tensor([1])
```

CLICK HERE Click here to see how to share your notebook.

About the Authors

Joseph Santarcangelo has a PhD in Electrical Engineering, his research focused on using machine learning, signal processing, and computer vision to determine how videos impact human cognition. Joseph has been working for IBM since he completed his PhD.