

סיווג בעזרת רשת ניוונים תמונת כלבים מול חתולים  
15.12.2023

מבוא


בתרגיל זה היה עליכם להגדיר רשת קונבולוציה בסיסית ולנסות לסווג סט נתונים המורכב משתי מחלקות - תמונות כלבים / חתולים

התרגיל

נתון לכם סט הנתונים הייעודי, תמונות משני סוגים כלבים / חתולים, ניתן להוריד מהלינק הבא:  
<https://www.microsoft.com/en-us/download/details.aspx?id=54765>  
בתקיה יש 9999 תמונות כלבים ו 9999 תמונות חתולים. כל סטודנט יקבל 500 תמונות מכל קטגוריה ע"פ הרשימה הבאה::

אני ממליץ לעבוד עם האתר <https://colab.research.google.com>. תפתחו שם מחברת פיתון חדשה ותעלו את קובץ הנתונים ותפתחו אותו. זכרו לבחור GPU במחברת

Notebook settings

Hardware accelerator  
GPU 

To get the most out of Colab, avoid using a GPU unless you need one. [Learn more](#)

☐ Omit code cell output when saving this notebook

CANCEL SAVE

הקוד מורכב מהשלבים הבאים

1. קריאת הנתונים (התמונות) והכנתם לאימון הרשת
2. הגדרת הרשת
3. אימון והבדיקה

### **1. קריאת הנתונים (התמונות) והכנתם לאימון הרשת**

את התמונות אפשר לדחוס שוב ולהעלות ל colab חלקו את ה 500 תמונות שלכם ל400 תמונות לאימון ו 100 לבדיקה.

קובץ הזיפ מגיע מחולק ל- train ו- test אנו נרצה לאחד את כל הקבצים מ- train ו- test לתיקיה אחת את זה נעשה על ידי הפקודות הבאות (שכדאי להכיר גם באופן כללי)

```
CAT_DOG_data/train/cat/  
CAT_DOG_data/train/dog/  
CAT_DOG_data/test/cat/  
CAT_DOG_data/test/dog/
```

הקוד ההתחלתי של התרגיל – אלו הספריות שיש להשתמש בהן:

```
import torch  
import torchvision  
import torchvision.transforms as transforms  
import torch.nn as nn  
import torch.optim as optim  
import torch.nn.functional as F  
import os  
import shutil  
import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
from matplotlib import image as mp_image  
import seaborn as sns  
%matplotlib inline  
  
from sklearn import metrics  
from sklearn.metrics import accuracy_score, confusion_matrix  
from sklearn.model_selection import train_test_split
```

מכיוון שהתמונות בקבצים המקוריים הן בגדלים שונים, ראשית נרצה לעבור עליהן ולשנות את גודלן לגודל אחיד. הקוד הבא עושה זאת עבורכם, עובר על התיקייה ועושה resize לכל הקבצים [טעינת הנתונים והכנתם לאימון הוא חלק בלתי נפרד מאימון רשתות, אני נותן לכם כאן רק טעימה קטנה:]

```
from PIL import Image  
def resize_image(src_image, size=(128,128), bg_color="white"):  
    from PIL import Image, ImageOps  
  
    # resize the image so the longest dimension matches our target size  
    src_image.thumbnail(size, Image.ANTIALIAS)  
  
    # Create a new square background image  
    new_image = Image.new("RGB", size, bg_color)  
  
    # Paste the resized image into the center of the square background  
    new_image.paste(src_image, (int((size[0] - src_image.size[0]) / 2), int((size[1] - src_image.size[1]) / 2)))  
  
    return new_image  
  
training_folder_name = '../CAT_DOG_data/all'
```

```

# New location for the resized images
train_folder = './DATA_OUT/'

# Create resized copies of all of the source images
size = (128,128)
# Create the output folder if it doesn't already exist
if os.path.exists(train_folder):
    shutil.rmtree(train_folder)

for root, folders, files in os.walk(training_folder_name):
    for sub_folder in folders:
        print('processing folder ' + sub_folder)
        # Create a subfolder in the output location
        saveFolder = os.path.join(train_folder,sub_folder)
        if not os.path.exists(saveFolder):
            os.makedirs(saveFolder)
        # Loop through files in the subfolder (Open each & resize & save
        file_names = os.listdir(os.path.join(root,sub_folder))
        for file_name in file_names:
            file_path = os.path.join(root,sub_folder, file_name)
            image = Image.open(file_path)
            resized_image = resize_image(image, size)
            saveAs = os.path.join(saveFolder, file_name)
            resized_image.save(saveAs)

```

כעת, לאחר שהכנו את הקבצים הנמצאים בתיקיות והקבצים בגודל אחיד, הפונקציה `load_dataset` מכינה וקוראת את הנתונים,

```

def load_dataset(data_path):
    import torch
    import torchvision
    import torchvision.transforms as transforms
    # Load all the images
    transformation = transforms.Compose([
        transforms.RandomHorizontalFlip(0.5),
        # Random vertical flip
        transforms.RandomVerticalFlip(0.3),
        # transform to tensors
        transforms.ToTensor(),
        # Normalize the pixel values (in R, G, and B channels)
        transforms.Normalize(mean=[0.5, 0.5, 0.5], std=[0.5, 0.5, 0.5])
    ])

    # Load all of the images and transforming them
    full_dataset = torchvision.datasets.ImageFolder(
        root=data_path,
        transform=transformation)

    # Split into training (70% and testing (30%) datasets)

```

```

train_size = int(0.7 * len(full_dataset))
test_size = len(full_dataset) - train_size

train_dataset, test_dataset = torch.utils.data.random_split(full_dataset, [train_size,
test_size])

# training data , 50-image batches
train_loader = torch.utils.data.DataLoader(
    train_dataset,
    batch_size=50,
    num_workers=0,
    shuffle=False
)

# testing data
test_loader = torch.utils.data.DataLoader(
    test_dataset,
    batch_size=50,
    num_workers=0,
    shuffle=False
)

return train_loader, test_loader

#####
# Get the iterative dataloaders for test and training data
train_loader, test_loader = load_dataset(train_folder)
batch_size = train_loader.batch_size
print("Data loaders ready to read", train_folder)

```

## 2. גדרת הרשת

המחלקה הבאה `net()` מגדירה את שכבות הרשת ב-`contractor` (יש להגדיר את השכבות השונות בפונקציה) והפונקציה `forward` משרשרת אותם לכדי רשת אחת מתפקדת.

```
# Create a neural net class
class Net(nn.Module):
    # Defining the Constructor
    def __init__(self, num_classes=3):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(

        # your code here

        self.fc = nn.Linear(in_features=, out_features=num_classes)

    def forward(self, x):

        # your code here

        return torch.log_softmax(x, dim=1)
device = "cpu"
if (torch.cuda.is_available()):
    device = "cuda"
classes = 2
model = Net(num_classes=classes).to(device)

print(model)
```

### 3. אימון והבדיקה

כעת, לאחר קריאת הנתונים והגדרת הרשת, נוכל לאמן ולבדוק את ביצועי הרשת.

```
loss_criteria = [ see here ]

def train(model, device, train_loader, optimizer, epoch):
    # Set the model to training mode
    model.train()
    train_loss = 0
    print("Epoch:", epoch)
    # Process the images in batches
    for batch_idx, (data, target) in enumerate(train_loader):
        data, target = data.to(device), target.to(device)
        optimizer.zero_grad() # Reset the optimizer
        # Push the data forward through the model layers
        output = model(data)
        # Get the loss
        loss = loss_criteria(output, target)
        # Keep a running total
        train_loss += loss.item()
        # Backpropagate
        loss.backward()
        optimizer.step()
        # Print metrics so we see some progress
        print('\tTraining batch {} Loss: {:.6f}'.format(batch_idx + 1, loss.item()))

    # return average loss for the epoch
    avg_loss = train_loss / (batch_idx+1)
    print('Training set: Average loss: {:.6f}'.format(avg_loss))
    return avg_loss
```

```
def test(model, device, test_loader):
    # Switch the model to evaluation mode (so we don't backpropagate or drop)
    model.eval()
    test_loss = 0
    correct = 0
    with torch.no_grad():
        batch_count = 0
        for data, target in test_loader:
            batch_count += 1
            data, target = data.to(device), target.to(device)
            # Get the predicted classes for this batch
            output = model(data)
            # Calculate the loss for this batch
            test_loss += loss_criteria(output, target).item()
```

```

        # Calculate the accuracy for this batch
        _, predicted = torch.max(output.data, 1)
        correct += torch.sum(target==predicted).item()
    # Calculate the average loss and total accuracy for this epoch
    avg_loss = test_loss / batch_count
    print('Validation set: Average loss: {:.6f}, Accuracy: {}/{} ({:.0f}%)\n'.format(
        avg_loss, correct, len(test_loader.dataset),
        100. * correct / len(test_loader.dataset)))

# return average loss for the epoch
return avg_loss

```

האימון עצמו:

```

# Train over 10 epochs (We restrict to 10 for time issues)
Network = net()
optimizer = [ see here ]
Device = 'cuda'
epochs = 10
print('Training on', device)
for epoch in range(1, epochs + 1):
    train_loss = train(model, device, train_loader, optimizer, epoch)
    test_loss = test(model, device, test_loader)
    epoch_nums.append(epoch)
    training_loss.append(train_loss)
    validation_loss.append(test_loss)

```



מה יש לעשות בתרגיל ?

1. ראשית העתיקו את הקוד למחברת חדשה, והשלימו את קטעי הקוד החסרים,
  - 1.1. השלימו את המחלקה `Net()` (התחילו עם רשת בסיסית ונסו אחרות, ראו סעיף 3)
  - 1.2. הגידרו `optimizer` – ראו קישור בשורה המתאימה שיסביר לכם איך/מה אפשר להשלים.
  - 1.3. `loss_criteria` – ראו קישור בשורה המתאימה שיסביר לכם איך/מה אפשר להשלים
  - 1.4. השלימו קוד אשר מצייר בגרף של ערכי ה-Loss בבדיקה ובאימון.
2. נסו לאמן את הרשת במספר דרכים שונות:
  - 2.1. הגדילו והקטינו את ה-`batch-size`.
  - 2.2. נסו שיטות אופטימיזציה שונות
  - 2.3. ~~נסו להוסיף עוד `transforms`~~
  - 2.4. שחקו עם גדלי רשתות שונות (יותר קונבולוציות / פחות).
  - 2.5. הקוד שלכם קובע את גודל התמונות ל-`128x128`. נסו להגדיל או להקטין את התמונות.
  - 2.6. ועוד לשיקולכן....