Imagenet_validation_pretrained

November 27, 2018

This code is to load a pretrained model from Pytorch modelzoo and check it's performance on Imagenet validation set

```
In [ ]: import torch
        import torchvision
        import torchvision.transforms as transforms
        import torchvision.models as models
        from torchvision import datasets
        import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
```

```
In [ ]: %reload_ext autoreload
        %autoreload 2
        %matplotlib inline
```

```
In [ ]: import sys
        sys.path.insert(0, '/data/ali/')
```

```
In [ ]: PATH = "/data/ali/imagenet_validation/"
```

Just to display an image if we want to check out some image.

```
In [ ]: image_name = 'ILSVRC2012_val_'+str(5300).zfill(8)+'.JPEG'
        folder_name = '197'
        img = plt.imread(f'{PATH}{folder_name}/{image_name}')
        plt.imshow(img);
```

To convert an image tensor used for validation back to a format which can be displayed.

```
In [ ]: def imshow(inp, title=None):
            inp = inp.cpu().numpy().transpose((1, 2, 0))
            mean = np.array([0.485, 0.456, 0.406])
            std = np.array([0.229, 0.224, 0.225])
            inp = std * inp + mean
            inp = np.clip(inp, 0, 1)
            plt.imshow(inp)
            if title is not None:
                plt.title(title)
            plt.pause(3)
```

To load the dataset using the Pytorch DataLoader and the datasets.ImageFolder classes. Since we used the validation_set_creator to split the validation set into subfolders we can now use datasets.ImageFolder.

```
In [ ]: batch_sz = 8
        img_sz =224 #the target size after cropping
        val_dataset = datasets.ImageFolder(PATH,  transforms.Compose([
                    transforms.CenterCrop(img_sz),
                    transforms.ToTensor(),
                    transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
                ])) #loading the dataset
        val_loader = torch.utils.data.DataLoader(val_dataset, batch_size=batch_sz, shuffle=True,
        class_names = val_dataset.classes #Getting class labels in order to use it later for acc
```

Loading the pretrained model from pytorch modelzoo. Note that we need to put it on GPU and set it to evaluation mode instead of training mode.

```
In [ ]: Resnet = models.resnet50(pretrained=True)
        Resnet.to('cuda')
        Resnet.eval()
```

This is the class to do validation testing by cycling through all images and checking the accuracy of the model.

```
In [ ]: def test(model, test_loader, batch_size, class_names):

            # Accuracy counter
            correct = 0

            # Loop over all examples in test set
            for data, target in test_loader:
                #obtaining the folder names (which is the target class label) for the minibatch
                target_classes = [class_names[x] for x in target]
                # Send the data and label to the gpu
                data, target = data.to('cuda'), target.to('cuda')
                # Forward pass the data through the model
                output = model(data)
                init_pred = output.max(1)[1] # get the index of the max log-probability
                predicted_classes = [str(x.cpu().numpy()) for x in init_pred]
                for i in range(len(target)):
                    if predicted_classes[i] == target_classes[i]:
                        correct += 1
            print(f'Correct {correct}')

            # Return the accuracy
            return correct

In [ ]: acc = test(Resnet, val_loader, batch_sz, class_names)
```