Create Your Own Image Classifier

REVIEW

CODE REVIEW 5

HISTORY

Meets Specifications

Great work! 👏

Congratulations on completing your project!

- I certainly enjoyed walking through your code. It's very clean and I can clearly see the effort that has been put into this.
- It was a really a Great first submission, less than 10% of students pass their project in the first attempt. You should give yourself a pat on the back and I am really happy with your work.
- Additionally, Do check out this it is a great playlist to watch for neural networks

Here are some additional links that might help you further your understanding:

- https://machinelearningmastery.com/transfer-learning-for-deep-learning/
- http://cs231n.github.io/transfer-learning/

Specific to this flower classification problem:

- https://arxiv.org/ftp/arxiv/papers/1708/1708.03763.pdf
- https://www.robots.ox.ac.uk/~vgg/research/flowers_demo/docs/Chai11.pdf

EXTRA LINKS(Optional)

- I wanted to refer you to some famous Ted Talks and other content to inspire you for a great career ahead
- Geoffrey Hinton the man most noted for his work in neural networks
- Some practical tips by Andrew NG on how to advance your career in AI

Good Luck in your AI journey, keep learning. 😄

Files Submitted

The submission includes all required files. (Model checkpoints not required.)

- All required files are included.

Part 1 - Development Notebook

All the necessary packages and modules are imported in the first cell of the notebook

- Good job importing everything into one cell. This is just a good practice and helps one to go through the project in one go, check out this thread to learn more about some of the advantages of doing so.

`torchvision` transforms are used to augment the training data with random scaling, rotations, mirroring, and/or cropping

- Good job using transform.torchvision to enhance the volume of data by augmenting it. We do random resize and crop just to bring variety in our training set and making it more robust.
- Check out this article on data augmentation.

The training, validation, and testing data is appropriately cropped and normalized

Good job

- It is always good to resize the image before cropping otherwise we could end up with a really zoomed image. 👏🏻

The data for each set is loaded with torchvision's DataLoader

- Dataloader helps us to load data in mini batches and speeds up the training process.

A pretrained network such as VGG16 is loaded from torchvision.models and the parameters are frozen

- Good job using `nn.Sequential` to create a new classifier on top of pretrained model. By creating a new classifier we have repurposed the pre-trained model to our task(in our case flower category). Refer to this article to read more about how to create nets.
- Checkout this article to understand why we use activation functions such as ReLU.
- This is a Great link to article on dropout

A new feedforward network is defined for use as a classifier using the features as input

- Great job loading pre-trained model and using transfer learning. (Try loading densenet also it works a little better for this case)
- You should also refer to this link for a better understanding of transfer learning.

The data for each set (train, validation, test) is loaded with torchvision's ImageFolder

- The code looks good here as well. Well done!
- Your code is well commented and clean, hence easy to go through.

The parameters of the feedforward classifier are appropriately trained, while the parameters of the feature network are left static

- This is perfect! You are only training the classier layers and not the layers from the pretrained models using this code block: `optim.Adam(classifier.parameters(), lr=l_r)`
- Checkout this link on different Loss Functions
- We have the option to use many optimizers like SGD, Adamax, RMSprop. You must explore them as well, here is a link to it.
- Here is a great link on choosing the best learning rate

The network's accuracy is measured on the test data

- Great job in printing the logs and checking for the validation set accuracy after each epoch, we can check for overfitting or underfitting in this manner 👏
- Refer this link to learn more about each dataset (Train, Validation and Test Sets)
- Checkout this link for more understanding on overfitting vs underfitting

During training, the validation loss and accuracy are displayed

- Great Job with 82% accuracy on the test set, which is great!! 😄

The trained model is saved as a checkpoint along with associated hyperparameters and the class_to_idx dictionary

There is a function that successfully loads a checkpoint and rebuilds the model

The process_image function successfully converts a PIL image into an object that can be used as input to a trained model

- You should use PIL library and python native functions to complete this, so you could learn about most used python libraries in image processing.
- This function should process the images in the same manner used for training. As it is mentioned in your jupyter notebook-

First, resize the images where the shortest side is 256 pixels, keeping the aspect ratio. This can be done with the thumbnail or resize methods. Then you'll need to crop out the center 224x224 portion of the image.

- You are getting the right result, but in this section we have to do all this without using transforms.
- Kindly resize the image such that shortest side is 256 pixels and keeping the aspect ratio. You have resized to a square of size 256x256 every time.
  For example, if you have an image of size **512x2048**(Aspect ratio is 1:4), you need to resize the image to 256x1024, and not 256x256

You are using resize(256), which will give you 256x256 square instead we just want to resize the images where the shortest side is 256 pixels. Refer below code:

```
HINT -

       width, height = image.size
       size = 256, 256

       if width > height:
              ratio = float(width) / float(height)
              newheight = ratio * size[0]



       else:
        ratio = float(height) / float(width)
        newwidth = ratio * size[0]
```

- In case you want to have a look at the torchvision transforms source code. (Optional)
- Let me know in the comments section how you feel about this change and if you have fully understood my explanation of this concept 😄

The predict function successfully takes the path to an image and a checkpoint, then returns the top K most probably classes for that image

A matplotlib figure is created displaying an image and its associated top 5 most probable classes with actual flower names

- The bar-graph with the correct class-names are being printed for the top k probabilities.

Part 2 - Command Line Application

train.py successfully trains a new network on a dataset of images and saves the model to a checkpoint

- Leaving comments for this section in code review

The training loss, validation loss, and validation accuracy are printed out as a network trains

The training script allows users to choose from at least two different architectures available from torchvision.models

The training script allows users to choose training the model on a GPU

The training script allows users to set hyperparameters for learning rate, number of hidden units, and training epochs

The predict.py script successfully reads in an image and a checkpoint then prints the most likely image class and it's associated probability

The predict.py script allows users to print out the top K classes along with associated probabilities

The predict.py script allows users to load a JSON file that maps the class values to other category names

The predict.py script allows users to use the GPU to calculate the predictions

DOWNLOAD PROJECT

5 CODE REVIEW COMMENTS ›