UDACITY

‹ Return to Classroom

DISCUSS ON STUDENT HUB

Create Your Own Image Classifier - TensorFlow

REVIEW

CODE REVIEW 2

HISTORY

▼ image_classifier/predict.py 2

```
1  import argparse
2  import json
3  import numpy as np
4  import os
5  import tensorflow as tf
6  import tensorflow_hub as hub
7  from PIL import Image
8
9  os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2' # Avoid warning
10
11 # Parse arguments
12 parser = argparse.ArgumentParser(
13     description='Predict flowers from images',
14 )
15
16 parser.add_argument('image_file', action="store", help="Image file to predic
17 parser.add_argument('model_file', action="store", help="Model to use for pre
18 parser.add_argument('--top_k', action="store", type=int, default=1, help="To
19 parser.add_argument('--category_names', action="store", help="Name of file v
20
21 args = parser.parse_args()
22
23 # Define functions
24
25 def process_image(image):
```

```
    image_size = 224
    res = tf.convert_to_tensor(image, np.float32)
    res = tf.image.resize(res, (image_size, image_size))
    res /= 255
    return res.numpy()

def predict(image_path, model, top_k, names = None):
    image = Image.open(image_path)
    image = np.asarray(image)
    image = np.expand_dims(image, 0)
    image = process_image(image)
    res = model.predict(image)[0]
    top = np.flip(res.argsort()[-top_k:])
    cat = []
    for t in top:
        if names:
            cat.append(class_names[str(t + 1)])
        else:
            cat.append(str(t + 1))
    return res[top], cat

```

AWESOME

Good work. Correctly configured. The expressions for the predict steps aligns with the requirement

```
# Main

reloaded_model = tf.keras.models.load_model(args.model_file, custom_objects=

class_names = None
if args.category_names:
    with open(args.category_names, 'r') as f:
        class_names = json.load(f)
```

AWESOME

Correctly done. JSON files loaded to map class values

```

probs, classes = predict(args.image_file, reloaded_model, args.top_k, class_

print(f'\nThe picture most probably shows a {classes[0]} (with {probs[0] * 1

if args.top_k > 1:
    print(f'The top {args.top_k} most probable flowers:\n')
    for i in range(args.top_k):
        print(f'{i + 1:2}. {classes[i]:30}   {probs[i] * 100:.2f}%')
else:
    print("Use the --top_k option to see more than one alternative")

print("\n")

if not class_names:
    print("Use the --category_names option to show names for the flowers")
    print("This may work well:")
    print("  --category_names label_map.json")
    print("\n")


```

- image_classifier/README.md
- finding_donors/visuals.py
- finding_donors/Udacity Reviews_files/udacity-base.min.css
- finding_donors/Udacity Reviews_files/materialize.min.js
- finding_donors/Udacity Reviews_files/jquery.min.js
- finding_donors/Udacity Reviews_files/grading_481bb2da.js
- finding_donors/Udacity Reviews_files/blueshift.js
- finding_donors/Udacity Reviews_files/angular.min.js
- finding_donors/Udacity Reviews.html
- finding_donors/README.md
- README.md

RETURN TO PATH

Rate this review

START