A Simple Tutorial to Classify Images Using TensorFlow — Step by Step Guide

Prabhu Apr 28, 2018 · 3 min read

Figure: 1 → Dog Breeds Dataset from Kaggle

In this tutorial, I am going to show how easily we can train images by categories using the Tensorflow deep learning framework. For this tutorial, I have taken a simple use case from Kaggle's prediction competition "Dog Breed Identification", to determine the breed of a dog in an image.

My Github URL for this tutorial → https://github.com/RaghavPrabhu/Deep-Learning/tree/master/dogs_breed_classification

Pre-trained Models

There have been many CNNs trained models using various CNN architectures (Earlier in my blog explained these architectures LeNet, AlexNet, VGG, GoogLeNet Inception, ResNet) are created. These CNNs have been trained on the ILSVRC-2012-CLS image classification dataset.

These models, by default it can classify whether an object is a car or a truck or an elephant or an airplane or a cat or a dog and so on. To train your own dataset, we need a lot of images by its categorization to achieve high accuracy. In some cases, we may not get more training datasets from any sources. During that scenarios, the **Data augmentation** technique comes in handy to solve this dataset limitation by transform and create new images from existing images to flip, rotate, apply color variation, etc., Refer Stanford's data augmentation paper for more details.

Dataset Pre-Processing

First, to download the train and test dataset along with its labels from Kaggle by using this URL. Refer below screenshot for details

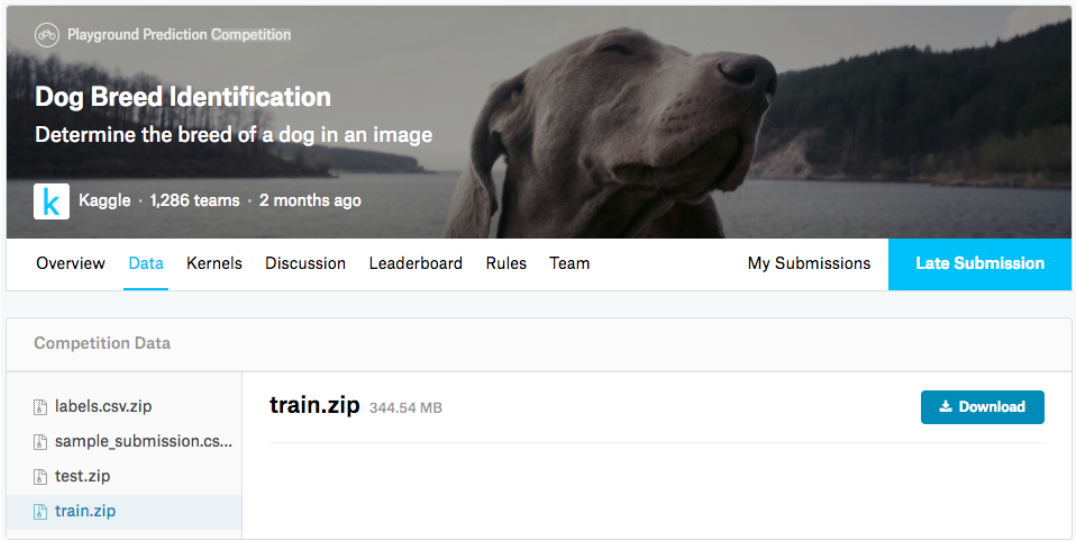
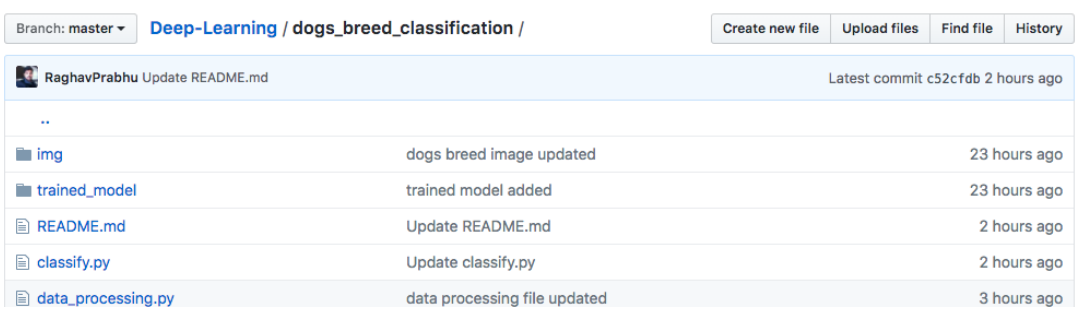


Figure: 2→ Dog Breeds Dataset Site from Kaggle

Step 1: Setup and Install

- clone the project from my GitHub repo

```
git clone git@github.com:RaghavPrabhu/Deep-Learning.git
cd Deep-Learning/dogs_breed_classification/
```



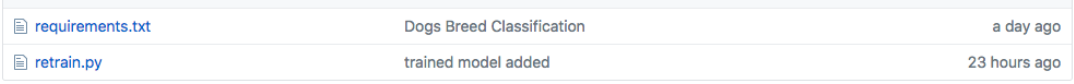


Figure : 3 → Tutorial Github Repo

- Create a virtual environment
- Install dependent libraries

```
* virtualenv -p python3 env
* source env/bin/activate
* pip install -r requirements.txt
```

- Unzip all folders which we downloaded from Kaggle's site

```
unzip train.zip
unzip test.zip
unzip labels.csv.zip
```

Step 2: Organise your train folder

- Run data processing python code to re-arrange folders by dogs breed name

```
"""
Iterate train dataset folder to create dataset folders by dog breed names.
Each 32 bit UUID file name have a equivalent reference name (dog breed) name in labels.c
"""
def organise_dataset(root_path,):
    dataset_path = root_path+'/dataset'
    train_data = root_path+'/train/'
    os.makedirs(root_path, exist_ok=True)
    df = pd.read_csv(root_path+'/labels.csv')
    files = os.listdir(train_data)
    print("Organising dataset by creating folders by dogs breeds using names in labels")
    for file in files:
        # Define folder name reference in labels csv by 32 UUID file name
        folder_name = df.loc[df['id'] == file.split('.')[0],'breed'].values[0]
        os.makedirs(dataset_path+'/'+folder_name, exist_ok=True)
        source = train_data+file
        destination = dataset_path+'/'+folder_name+'/'+file
        # Moving files from source (train folder) to detination folder under each breed
        os.rename(source, destination)
    print("Dataset folders successfully created by breed name and copied all images in c
```

data_processing.py hosted with ❤️ by GitHub view raw

```
python3 data_processing.py
```

The above python script will organize folders by dog breed names. Please refer below screenshot as an organized output processed folders by dog breed name. Each dog breed folder contains corresponding dog images inside it.

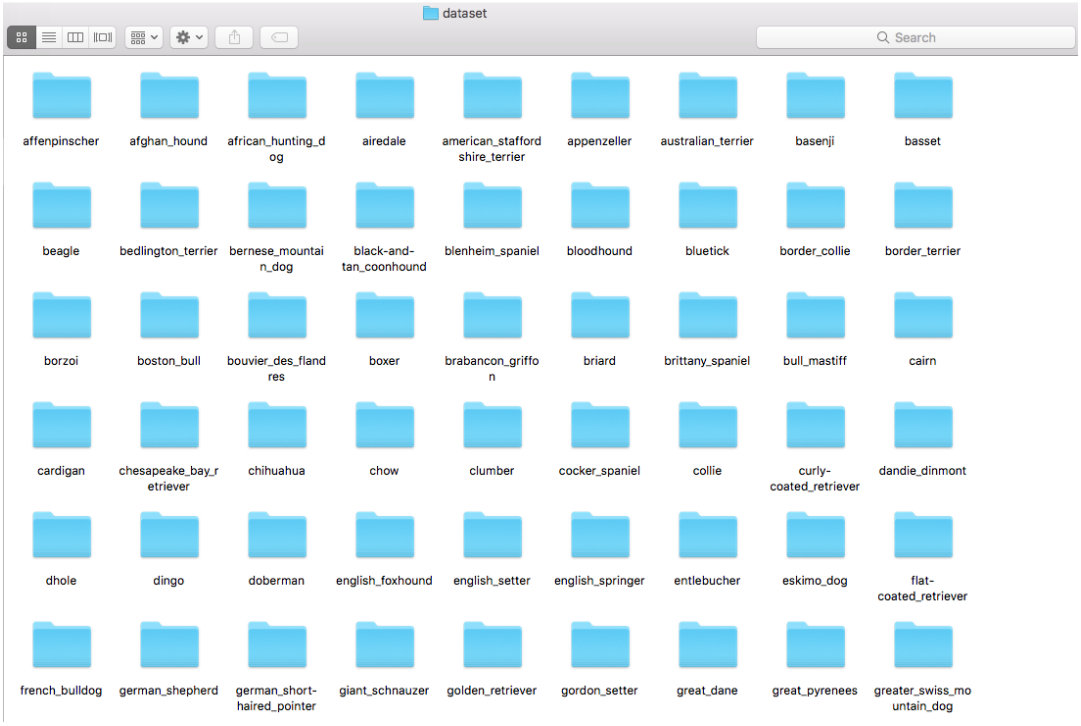


Figure: 4 → Organised output folders

Step 3: Train your model using our processed dataset

Run the below command to train your model using CNN architectures. By default, the below script will download ‘Google’s inception architecture — ‘inception-2015–12–05.tgz’.

```
python3 retrain.py --image_dir=build/dataset/ --
bottleneck_dir=build/bottleneck/ --how_many_training_steps=500 --
output_graph=build/trained_model/retrained_graph.pb --
output_lables=build/trained_model/retrained_labels.txt --
summaries_dir=build/summaries
```

The following is the processing output.

```
Downloading inception-2015-12-05.tgz ...
INFO:tensorflow:Looking for images in 'collie'
INFO:tensorflow:Looking for images in 'scotch_terrier'
```

```
INFO:tensorflow:Looking for images in 'brittany_spaniel'
INFO:tensorflow:Looking for images in 'labrador_retriever'
INFO:tensorflow:Looking for images in 'cairn'
INFO:tensorflow:Looking for images in 'norwich_terrier'
INFO:tensorflow:Looking for images in 'miniature_poodle'
INFO:tensorflow:Looking for images in 'komondor'
INFO:tensorflow:Looking for images in 'rhodesian_ridgeback'
INFO:tensorflow:Looking for images in 'saint_bernard'
INFO:tensorflow:Looking for images in 'pekinese'
INFO:tensorflow:Looking for images in 'cardigan'
INFO:tensorflow:Looking for images in 'dandie_dinmont'
INFO:tensorflow:Looking for images in 'keeshond'
INFO:tensorflow:Looking for images in 'bloodhound'
INFO:tensorflow:Looking for images in 'norfolk_terrier'
INFO:tensorflow:Looking for images in 'silky_terrier'
INFO:tensorflow:Looking for images in 'boston_bull'
....
```

Figure : 5 → Training our model

Tensorboard accuracy graph for the above trained model

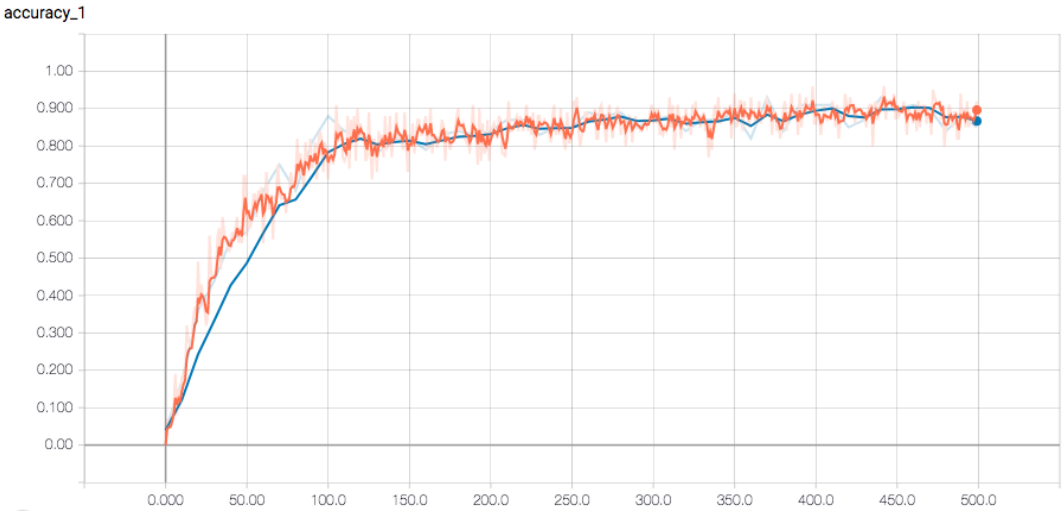


Figure 6 → TensorFlow Accuracy Graph

To run TensorBoard, run this command

```
User:/dogs_breed_classification$ tensorboard — logdir summaries/ —
host=0.0.0.0 — port=8888

TensorBoard 1.7.0 at http://0.0.0.0:8888 (Press CTRL+C to quit)#
```

Step 4: Test your model

Run the below python script to classify your test images based on our pre-trained model.

```
python classify.py
```

```
japanese_spaniel (score = 0.10750)
pekinese (score = 0.05065)
blenheim_spaniel (score = 0.03431)
papillon (score = 0.02191)
shih-tzu (score = 0.02064)
pomeranian (score = 0.01520)
lhasa (score = 0.01302)
brabancon_griffon (score = 0.01080)
maltese_dog (score = 0.01051)
chihuahua (score = 0.01032)
clumber (score = 0.01020)
pug (score = 0.01009)
sussex_spaniel (score = 0.00966)
shetland_sheepdog (score = 0.00965)
boston_bull (score = 0.00955)
keeshond (score = 0.00934)
affenpinscher (score = 0.00919)
bernese_mountain_dog (score = 0.00896)
saluki (score = 0.00841)
cocker_spaniel (score = 0.00826)
samoyed (score = 0.00804)
...
```

Figure: 6 → Output dogs breed classifier

You can skip step 3 in the above tutorial and directly use my pre-trained model (trained_model/retrained_graph.pb) available in my Github repo to test the model.

Python TensorFlow Convolutional Network Image Recognition Data Science