How to Make an Image Classifier in Python using Keras

Building and training a model that classifies CIFAR-10 dataset images which consists of airplanes, dogs, cats and other 7 objects using Keras library in Python.

Abdou Rockikz · 9 min read · Updated Jan 2020 · 1.8K · Machine Learning · Computer Vision

Image classification refers to a process in computer vision that can classify an image according to its visual content. For example, an image classification algorithm can be designed to tell if an image contains a cat or a dog. While detecting an object is trivial for humans, robust image classification is still a challenge in computer vision applications.

In this tutorial, you will learn how to successfully classify images in the `CIFAR-10` dataset which consists of airplanes, dogs, cats and other 7 objects.

Note that there is a difference between image classification and object detection, image classification is about classifying an image to some category, like in this example, the input is an image and the output is a single class label (10 classes). Object detection is about detecting, classfiying and localizing objects in real-world images, one of the main algorithms are YOLO object detection.

We will preprocess the images and labels, then train a **convolutional neural network** on all the training samples. The images will need to be **normalized** and the labels need to be **one-hot encoded**.

First, let's install the requirements for this project:

```
pip3 install keras numpy matplotlib tensorflow
```

For instance, open up an empty python file and call it **`train.py`** and follow along. Importing keras:

Topics
Python Standard Library
Computer Vision
Web Scraping
Natural Language Processing
Ethical Hacking
Machine Learning
General Python Topics
Packet Manipulation Using Scapy

New Tutorials
How to Use Threads to Speed Up your IO Tasks in Python

```
from keras.datasets import cifar10 # importing the dataset from keras
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras.callbacks import ModelCheckpoint, TensorBoard
from keras.utils import to_categorical
import os
```

Hyper Parameters

I have experimented with various parameters, and found this as an optimal ones:

```
# hyper-parameters
batch_size = 64
# 10 categories of images (CIFAR-10)
num_classes = 10
# number of training epochs
epochs = 30
```

`num_classes` just refers to the number of categories to classify, in this case `CIFAR-10` has only 10 categories of images.

Understanding and Loading CIFAR-10 Dataset

- The dataset consists of 10 classes of images which its labels ranging from 0 to 9:
  - `0`: airplane.
  - `1`: automobile.
  - `2`: bird.
  - `3`: cat.
  - `4`: deer.
  - `5`: dog.
  - `6`: frog.
  - `7`: horse.
  - `8`: ship.
  - `9`: truck.
- `50000` samples for training data, and `10000` samples for testing data.
- Each sample is an image of `32x32x3` pixels (width and height of `32` and `3` depth which are RGB values).

Let's load this:

How to Extract Script and CSS Files from Web Pages in Python

How to Perform YOLO Object Detection using OpenCV and PyTorch in Python

How to Download Torrent Files in Python

How to Predict Stock Prices in Python using TensorFlow 2 and Keras

Popular Tutorials

How to Detect Human Faces in Python using OpenCV

How to Convert Speech to Text in Python

How to Make a Speech Emotion Recognizer Using Python And Scikit-learn

How to Encrypt and Decrypt Files in Python

How to Make a Network Scanner using Scapy in 5 Minutes

```python
def load_data():
    """This function loads CIFAR-10 dataset, normalized, and labels one-hot encoded""
    # loading the CIFAR-10 dataset, splitted between train and test sets
    (X_train, y_train), (X_test, y_test) = cifar10.load_data()
    print("Training samples:", X_train.shape[0])
    print("Testing samples:", X_test.shape[0])
    print(f"Images shape: {X_train.shape[1:]}")
    # converting image labels to binary class matrices
    y_train = to_categorical(y_train, num_classes)
    y_test = to_categorical(y_test, num_classes)
    # convert to floats instead of int, so we can divide by 255
    X_train = X_train.astype("float32")
    X_test = X_test.astype("float32")
    X_train /= 255
    X_test /= 255
    return (X_train, y_train), (X_test, y_test)
```

This function loads the dataset which is built-in keras, print some statistics and then:

- One-hot encode the labels using `to_categorical` function: This process is really important as it enables the labels to be a vector of 10 numbers so it will be suitable for the neural network. For example, a horse with a value of 7 will be encoded to: `[0, 0, 0, 0, 0, 0, 0, 1, 0, 0]`.
- Normalizing the images: Since the pixel numbers ranging from 0 to 255 are too big for the neural network to train, we will be normalizing these pixels to be ranged from 0 to 1.

Constructing the Model

The following model will be used:

```
def create_model(input_shape):
    # building the model
    model = Sequential()
    model.add(Conv2D(filters=32, kernel_size=(3, 3), padding="same", input_shape=inpu
    model.add(Activation("relu"))
    model.add(Conv2D(filters=32, kernel_size=(3, 3), padding="same"))
    model.add(Activation("relu"))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))
    model.add(Conv2D(filters=64, kernel_size=(3, 3), padding="same"))
    model.add(Activation("relu"))
    model.add(Conv2D(filters=64, kernel_size=(3, 3), padding="same"))
    model.add(Activation("relu"))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))
    model.add(Conv2D(filters=128, kernel_size=(3, 3), padding="same"))
    model.add(Activation("relu"))
    model.add(Conv2D(filters=128, kernel_size=(3, 3), padding="same"))
    model.add(Activation("relu"))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))
    # flattening the convolutions
    model.add(Flatten())
    # fully-connected layer
    model.add(Dense(1024))
    model.add(Activation("relu"))
    model.add(Dropout(0.5))
    model.add(Dense(num_classes, activation="softmax"))
    # print the summary of the model architecture
    model.summary()
    # training the model using rmsprop optimizer
    model.compile(loss="categorical_crossentropy", optimizer="adam", metrics=["accura
    return model
```

That's 3 layers of 2 ConvNets with a max pooling and ReLU activation function and then a fully connected with 1024 units. This is relatively a small model comparing to ResNet50 or Xception which are the state-of-the-art. If you feel to use models that made by deep learning experts, you need to use transfer learning.

Training the Model

Now, let's train the model:

```
if __name__ == "__main__":
    # load the data
    (X_train, y_train), (X_test, y_test) = load_data()
    # constructs the model
    model = create_model(input_shape=X_train.shape[1:])
    # some nice callbacks
    tensorboard = TensorBoard(log_dir="logs/cifar10-model-v1")
    checkpoint = ModelCheckpoint("results/cifar10-loss-{val_loss:.2f}-acc-{val_acc:.2
                                save_best_only=True,
                                verbose=1)
    # make sure results folder exist
    if not os.path.isdir("results"):
        os.mkdir("results")
    # train
    model.fit(X_train, y_train,
            batch_size=batch_size,
            epochs=epochs,
            validation_data=(X_test, y_test),
            callbacks=[tensorboard, checkpoint],
            shuffle=True)
```

After loading the data and creating the model, I used some callbacks to help us in the training, `ModelCheckpoint` will save the model each time an optimal point is reached, and `TensorBoard` will be tracking the accuracy and loss in each epoch and providing us with nice visualization.

We will be using "results" folder to save our models, if you're not sure how you can handle files and directories in Python, check this tutorial: How to Handle Files in Python using OS Module.

Run this, it will take several minutes to finish, depending on your CPU/GPU.

You'll get similar result to this:

```
Epoch 1/30
50000/50000 [==============================] - 26s 520us/step - loss: 1.6408 - ac

Epoch 00001: val_loss improved from inf to 1.20628, saving model to results/cifar
Epoch 2/30
50000/50000 [==============================] - 26s 525us/step - loss: 1.1885 - ac
```

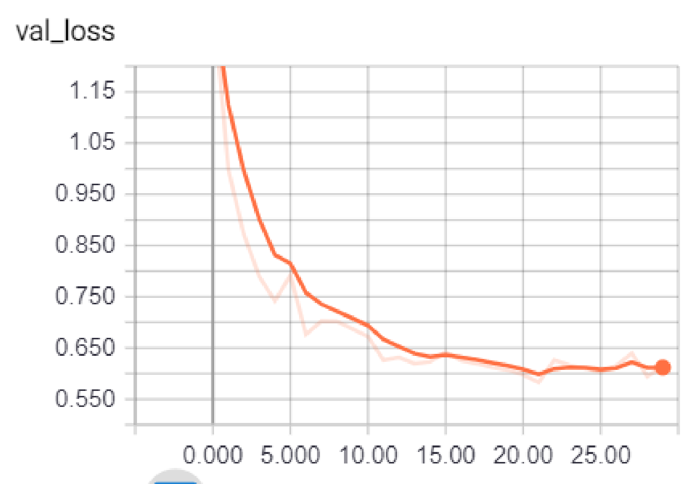
All the way to the final epoch:

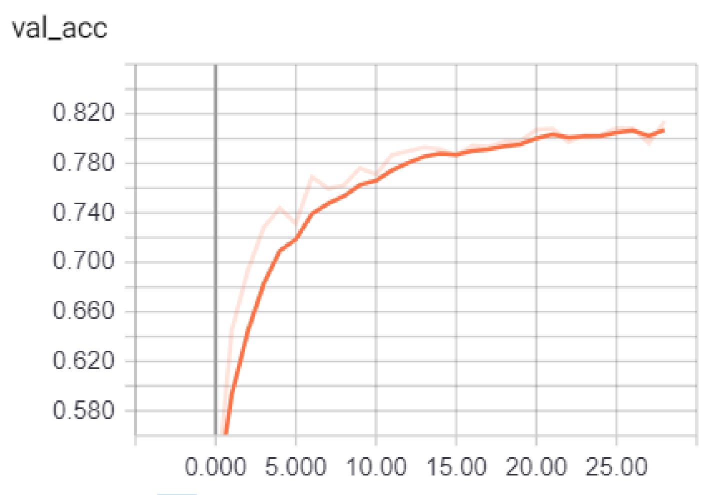
```
Epoch 29/30
50000/50000 [==============================] - 27s 534us/step - loss: 0.4225 - ac

Epoch 00029: val_loss did not improve from 0.56407
Epoch 30/30
50000/50000 [==============================] - 27s 549us/step - loss: 0.4205 - ac

Epoch 00030: val_loss did not improve from 0.56407
```

Now to open tensorboard, all you need to do is to type this command in the terminal or the command prompt in the current directory:

```
tensorboard --logdir="logs"
```

Open up a browser tab and type `localhost:6006`, you'll be redirected to tensorboard, here is my result:

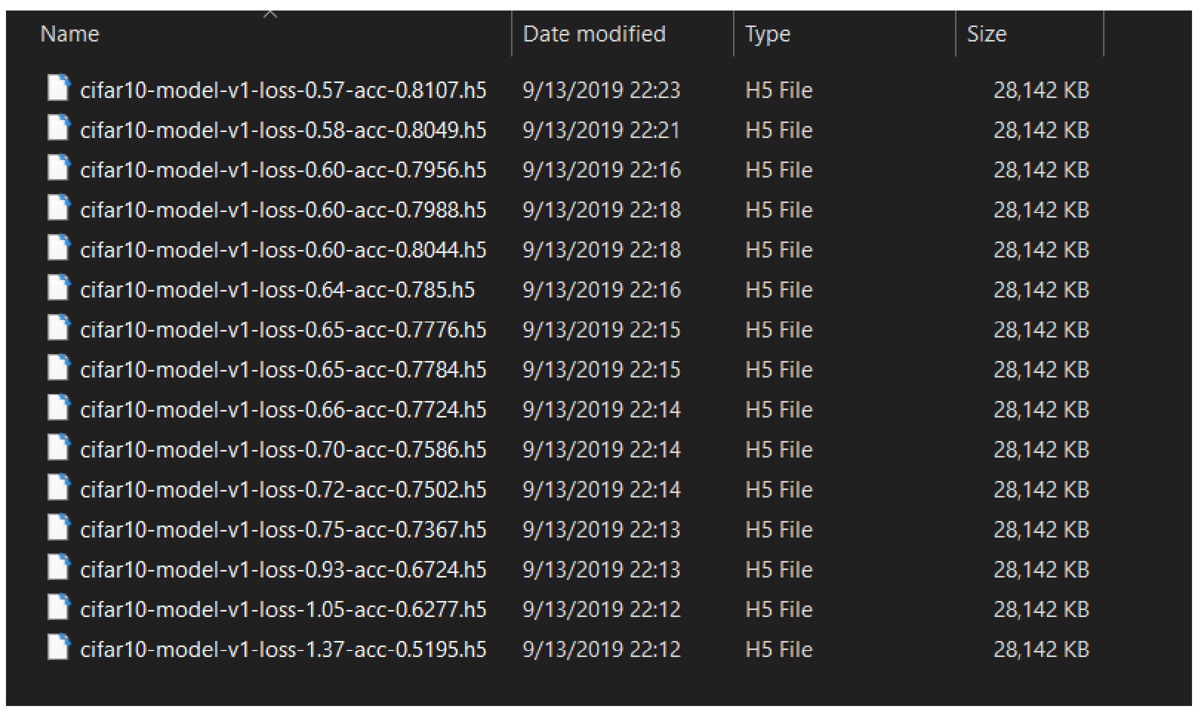




Clearly we are on the right track, validation loss is decreasing, and the accuracy is increasing all the way to about `81%`. That's great!

Testing the Model

Once training is completed, you will see various model weights saved in the results folder as shown in the following figure:



Name	Date modified	Type	Size
cifar10-model-v1-loss-0.57-acc-0.8107.h5	9/13/2019 22:23	H5 File	28,142 KB
cifar10-model-v1-loss-0.58-acc-0.8049.h5	9/13/2019 22:21	H5 File	28,142 KB
cifar10-model-v1-loss-0.60-acc-0.7956.h5	9/13/2019 22:16	H5 File	28,142 KB
cifar10-model-v1-loss-0.60-acc-0.7988.h5	9/13/2019 22:18	H5 File	28,142 KB
cifar10-model-v1-loss-0.60-acc-0.8044.h5	9/13/2019 22:18	H5 File	28,142 KB
cifar10-model-v1-loss-0.64-acc-0.785.h5	9/13/2019 22:16	H5 File	28,142 KB
cifar10-model-v1-loss-0.65-acc-0.7776.h5	9/13/2019 22:15	H5 File	28,142 KB
cifar10-model-v1-loss-0.65-acc-0.7784.h5	9/13/2019 22:15	H5 File	28,142 KB
cifar10-model-v1-loss-0.66-acc-0.7724.h5	9/13/2019 22:14	H5 File	28,142 KB
cifar10-model-v1-loss-0.70-acc-0.7586.h5	9/13/2019 22:14	H5 File	28,142 KB
cifar10-model-v1-loss-0.72-acc-0.7502.h5	9/13/2019 22:14	H5 File	28,142 KB
cifar10-model-v1-loss-0.75-acc-0.7367.h5	9/13/2019 22:13	H5 File	28,142 KB
cifar10-model-v1-loss-0.93-acc-0.6724.h5	9/13/2019 22:13	H5 File	28,142 KB
cifar10-model-v1-loss-1.05-acc-0.6277.h5	9/13/2019 22:12	H5 File	28,142 KB
cifar10-model-v1-loss-1.37-acc-0.5195.h5	9/13/2019 22:12	H5 File	28,142 KB

Obviously, we will choose the optimal one, which is `cifar10-model-v1-loss-0.57-acc-0.8107.h5` (at least in my case, choose the highest accuracy or the lowest loss).

Open up a new python file called **`test.py`** and follow along.

Importing necessary utilities:

```
from train import load_data
from keras.models import load_model
import matplotlib.pyplot as plt
import numpy as np
```

Let's make a Python dictionary that maps each integer value to its corresponding label in the dataset:

```
# CIFAR-10 classes
categories = {
    0: "airplane",
    1: "automobile",
    2: "bird",
    3: "cat",
    4: "deer",
    5: "dog",
    6: "frog",
    7: "horse",
    8: "ship",
    9: "truck"
}
```

Loading the test data and the model:

```
# load the testing set
(_, _), (X_test, y_test) = load_data()
# load the model with optimal weights
model = load_model("results/cifar10-loss-0.58-acc-0.81.h5")
```

Evaluation:

```
# evaluation
loss, accuracy = model.evaluate(X_test, y_test)
print("Test accuracy:", accuracy*100, "%")
```

Let's take a random image and make a prediction:

```
# get prediction for this image
sample_image = X_test[7500]
prediction = np.argmax(model.predict(sample_image.reshape(-1, *sample_image.shape))[0
print(categories[prediction])
```

Remember when we hot-encoded the labels ? Here, we need to reverse it from that 10 length vector into a number, that's what `np.argmax` is initially doing. After that, we map it to our dictionary to get the human readable label, here is my output:

```
10000/10000 [==============================] - 3s 331us/step
Test accuracy: 81.17999999999999 %
frog
```

The model says it's a frog, let's check it:

```
# show the image
plt.axis('off')
plt.imshow(sample_image)
plt.show()
```

Result:

Tiny little frog! The model was right!

Conclusion

Alright, we are done with this tutorial, 81% isn't bad for this little **CNN**, I highly encourage you to twick the model or check ResNet50, Xception or other state-of-the-art models to get higher performance!

If you're not sure how to use these models, I have a tutorial on this: How to Use Transfer Learning for Image Classification using Keras in Python.

You may wonder that this images are so simple, `32x32` grid isn't how the real world is, images aren't simple like that, they often contain many objects, complex patterns and so on. As a result, it is often a common practice to use image segmentation methods such as contour detection or K-Means clustering segmentation before passing to any classification techniques.

Happy Training ♥

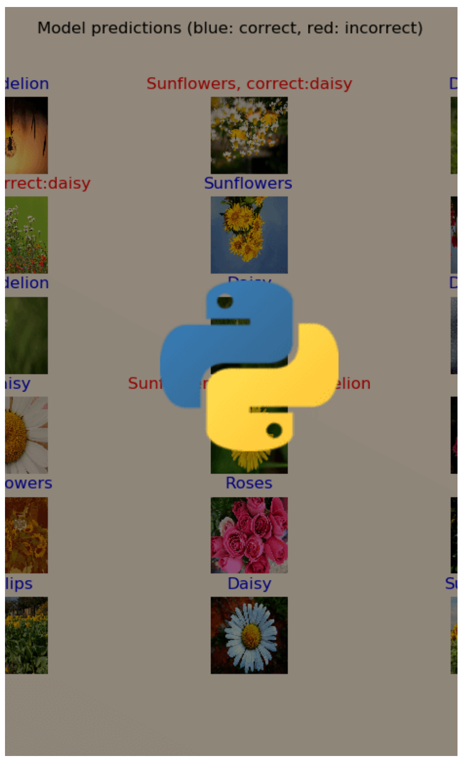
VIEW FULL CODE

Sharing is caring!

-

Read Also

How to Use Transfer Learning for Image Classification using Keras in Python


Model predictions (blue: correct, red: incorrect)
Sunflowers, correct:daisy
Sunflowers
Roses
Daisy

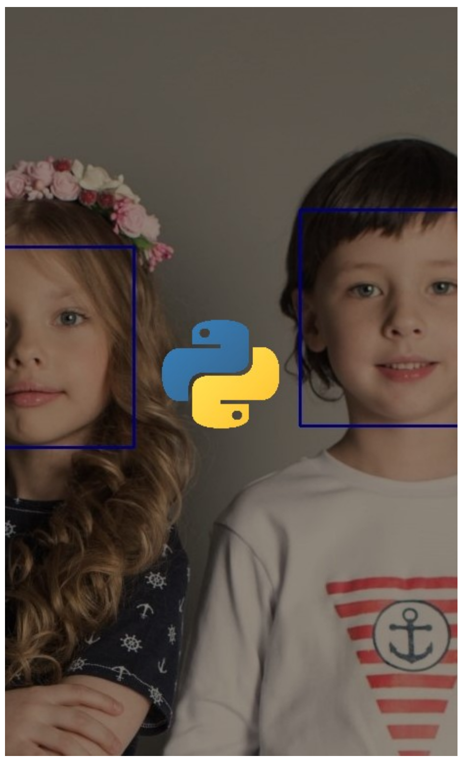
Learn what is transfer learning and how to use pre trained MobileNet model for better performance to classify flowers using Keras in Python.

VISIT →

How to Predict Stock Prices in Python using TensorFlow 2 and Keras

Predicting different stock prices using Long Short-Term Memory Recurrent Neural Network in Python using TensorFlow 2 and Keras.

VISIT →



How to Detect Human Faces in Python using OpenCV

Detecting and recognizing human faces in Python using OpenCV library which provides us with pre trained haar cascade classifiers.

VISIT →

Remote Jobs via HireRemote.io

Site Reliability Engineer

at Wikimedia Foundation, Inc. 2 days ago

python hadoop linux bigdata ruby c big data devops

DevOps Engineer

at Plutora 3 days ago

python aws devops amazon-cloudformation docker kubernetes saas cloud backend

Follow @ThePythonCode1

Like 6.3K Share

Comment panel

Comment system is still in Beta, if you find any bug, please consider contacting us here.

Paweł 4 months ago

REPLY

Hello! I really like the fact that CIFAR-10 is giving just an entry-level in image classification. And probably it's one of the most recognized and deeply studied data set in image classification. I am having two questions. First one regarding your model. Is it your own model? The second question about training. After reaching the 20th epoch I feel that if you use SGD with regularization and stop-resume method instead of ADAM you will probably get deeper on validation loss. What do you think?

Abdou Rockikz 4 months ago

REPLY

Yes, it is my own model, I have tested many combination of Conv2D/Pooling layers and this one tends to outperform them on this dataset (although it's not optimal, you can construct a better one). For the second question, I would like to know how deeper your validation loss went, can you tell me how much accuracy you achieved ? I just chose ADAM for training speed !

Pawel Rolbiecki 4 months ago

REPLY

I didn't to it in CIFAR-10. 2 years ago I have made an experiment with distinguishing "dogs or cats breeds" like data set. I mean a set in which every class is similar to other - like dogs breeds. I don't remember the numbers of validation loss - sorry. But one thing left in my had. That training a model with ADAM couldn't get as lo as "brake-resume" SGD both in accuracy and validation loss. I hope I will provide you an example on my blog post in 2-3 weeks. I wan to finish my current series.

teimoor 2 months ago

REPLY

How to load an actual dataset that is not packaged with keras? please give an example for this particular case since i don't know how to create the training and test set from raw images

Abdou Rockikz 2 months ago

REPLY

Hello teimoor, in that case, you need to manually download, preprocess and prepare your images, you can check tons of tutorials on the web. Also, I did an example in the tutorial where I used transfer learning in which I manually downloaded and prepared flower datasets: https://www.thepythoncode.com/article/use-transfer-learning-for-image-flower-classification-keras-python

Abdou Rockikz 2 months ago

REPLY

Also, check this sentdex's video tutorial for further help: https://www.youtube.com/watch?v=j-3vuBynnOE

Your email address will not be published.

Enter your name

Enter Email

Write a comment...

☐ I'm not a robot

reCAPTCHA
Privacy - Terms

☑ Subscribe for our newsletter

COMMENT