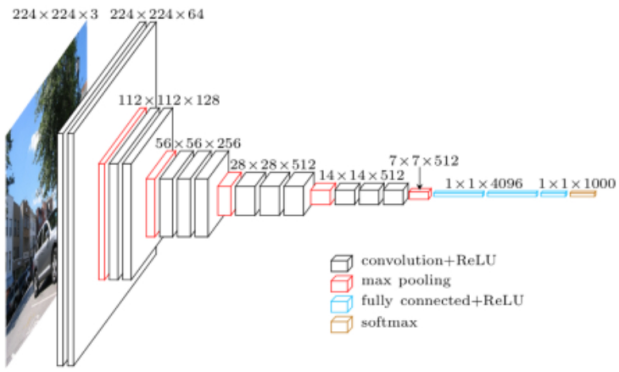
Published in Towards Data Science

Rohit Thakur
Aug 6, 2019 . 7 min read . Listen

Step by step VGG16 implementation in Keras for beginners

VGG16 is a convolution neural net (CNN) architecture which was used to win ILSVR(Imagenet) competition in 2014. It is considered to be one of the excellent vision model architecture till date. Most unique thing about VGG16 is that instead of having a large number of hyper-parameter they focused on having convolution layers of 3x3 filter with a stride 1 and always used same padding and maxpool layer of 2x2 filter of stride 2. It follows this arrangement of convolution and max pool layers consistently throughout the whole architecture. In the end it has 2 FC(fully connected layers) followed by a softmax for output. The 16 in VGG16 refers to it has 16 layers that have weights. This network is a pretty large network and it has about 138 million (approx) parameters.



Architecture of VGG16

I am going to implement full VGG16 from scratch in Keras. This implement will be done on Dogs vs Cats dataset. You can download the dataset from the link below.

> *https://www.kaggle.com/c/dogs-vs-cats/data*

Once you have downloaded the images then you can proceed with the steps written below.

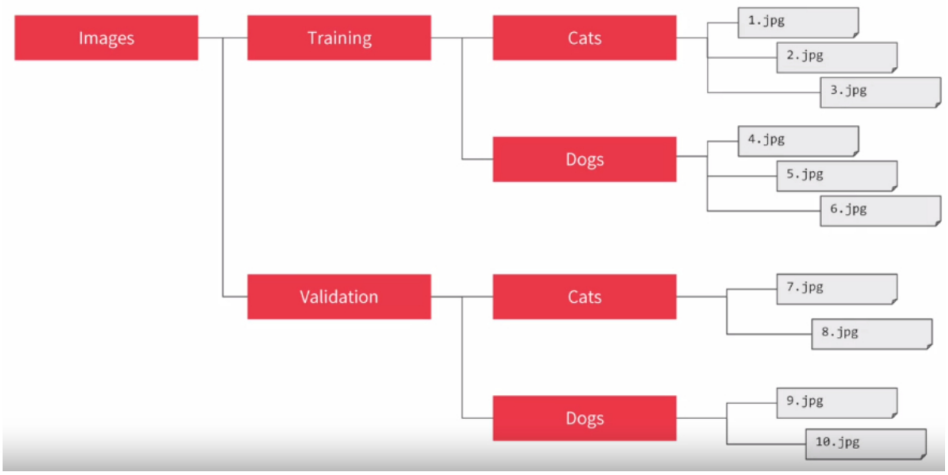
```
import keras,os
from keras.models import Sequential
from keras.layers import Dense, Conv2D, MaxPool2D , Flatten
from keras.preprocessing.image import ImageDataGenerator
```

```
from keras.preprocessing.image import ImageDataGenerator
import numpy as np
```

Here I first importing all the libraries which i will need to implement VGG16. I will be using Sequential method as I am creating a sequential model. Sequential model means that all the layers of the model will be arranged in sequence. Here I have imported ImageDataGenerator from keras.preprocessing. The objective of ImageDataGenerator is to import data with labels easily into the model. It is a very useful class as it has many function to rescale, rotate, zoom, flip etc. The most useful thing about this class is that it doesn't affect the data stored on the disk. This class alters the data on the go while passing it to the model.

```
trdata = ImageDataGenerator()
traindata = trdata.flow_from_directory(directory="data",target_size=
(224,224))
tsdata = ImageDataGenerator()
testdata = tsdata.flow_from_directory(directory="test", target_size=
(224,224))
```

Here I am creating and object of ImageDataGenerator for both training and testing data and passing the folder which has train data to the object trdata and similarly passing the folder which has test data to the object tsdata. The folder structure of the data will be as follows -



Folder structure of the data to passed to ImageDataGenerator

The ImageDataGenerator will automatically label all the data inside cat folder as cat and vis-à-vis for dog folder. In this way data is easily ready to be passed to the neural network.

```
model = Sequential()

model.add(Conv2D(input_shape=(224,224,3),filters=64,kernel_size=
(3,3),padding="same", activation="relu"))
```

```
model.add(Conv2D(filters=64,kernel_size=(3,3),padding="same",
activation="relu"))

model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))

model.add(Conv2D(filters=128, kernel_size=(3,3), padding="same",
activation="relu"))

model.add(Conv2D(filters=128, kernel_size=(3,3), padding="same",
activation="relu"))

model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))

model.add(Conv2D(filters=256, kernel_size=(3,3), padding="same",
activation="relu"))

model.add(Conv2D(filters=256, kernel_size=(3,3), padding="same",
activation="relu"))

model.add(Conv2D(filters=256, kernel_size=(3,3), padding="same",
activation="relu"))

model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))

model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same",
activation="relu"))

model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same",
activation="relu"))

model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same",
activation="relu"))

model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))

model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same",
activation="relu"))

model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same",
activation="relu"))

model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same",
activation="relu"))

model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
```

Here I have started with initialising the model by specifying that the model is a sequential model. After initialising the model I add

→ 2 x convolution layer of 64 channel of 3x3 kernal and same padding

→ 1 x maxpool layer of 2x2 pool size and stride 2x2

→ 2 x convolution layer of 128 channel of 3x3 kernal and same padding

→ 1 x maxpool layer of 2x2 pool size and stride 2x2

→ 3 x convolution layer of 256 channel of 3x3 kernal and same padding

→ 1 x maxpool layer of 2x2 pool size and stride 2x2

→ 3 x convolution layer of 512 channel of 3x3 kernal and same padding

→ 1 x maxpool layer of 2x2 pool size and stride 2x2

→ 3 x convolution layer of 512 channel of 3x3 kernal and same padding

→ 1 x maxpool layer of 2x2 pool size and stride 2x2

I also add relu(Rectified Linear Unit) activation to each layers so that all the negative values are not passed to the next layer.

```
model.add(Flatten())

model.add(Dense(units=4096,activation="relu"))

model.add(Dense(units=4096,activation="relu"))

model.add(Dense(units=2, activation="softmax"))
```

After creating all the convolution I pass the data to the dense layer so for that I flatten the vector which comes out of the convolutions and add

→ 1 x Dense layer of 4096 units

→ 1 x Dense layer of 4096 units

→ 1 x Dense Softmax layer of 2 units

I will use RELU activation for both the dense layer of 4096 units so that I stop forwarding negative values through the network. I use a 2 unit dense layer in the end with softmax activation as I have 2 classes to predict from in the end which are dog and cat. The softmax layer will output the value between 0 and 1 based on the confidence of the model that which class the images belongs to.

After the creation of softmax layer the model is finally prepared. Now I need to compile the model.

```
from keras.optimizers import Adam
opt = Adam(lr=0.001)

model.compile(optimizer=opt,
loss=keras.losses.categorical_crossentropy, metrics=['accuracy'])
```

Here I will be using Adam optimiser to reach to the global minima while training out model. If I am stuck in local minima while training then the adam optimiser will help us to get out of local minima and reach global minima. We will also specify the learning rate of the optimiser, here in this case it is set at 0.001. If our training is bouncing a lot on epochs then we need to decrease the learning rate so that we can reach global minima.

I can check the summary of the model which I created by using the code below.

```
model.summary()
```

The output of this will be the summary of the model which I just created.

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 224, 224, 64)	1792
conv2d_2 (Conv2D)	(None, 224, 224, 64)	36928
max_pooling2d_1 (MaxPooling2	(None, 112, 112, 64)	0
conv2d_3 (Conv2D)	(None, 112, 112, 128)	73856
conv2d_4 (Conv2D)	(None, 112, 112, 128)	147584
max_pooling2d_2 (MaxPooling2	(None, 56, 56, 128)	0
conv2d_5 (Conv2D)	(None, 56, 56, 256)	295168
conv2d_6 (Conv2D)	(None, 56, 56, 256)	590080
conv2d_7 (Conv2D)	(None, 56, 56, 256)	590080
max_pooling2d_3 (MaxPooling2	(None, 28, 28, 256)	0
conv2d_8 (Conv2D)	(None, 28, 28, 512)	1180160
conv2d_9 (Conv2D)	(None, 28, 28, 512)	2359808
conv2d_10 (Conv2D)	(None, 28, 28, 512)	2359808
max_pooling2d_4 (MaxPooling2	(None, 14, 14, 512)	0
conv2d_11 (Conv2D)	(None, 14, 14, 512)	2359808
conv2d_12 (Conv2D)	(None, 14, 14, 512)	2359808
conv2d_13 (Conv2D)	(None, 14, 14, 512)	2359808
max_pooling2d_5 (MaxPooling2	(None, 7, 7, 512)	0
flatten_1 (Flatten)	(None, 25088)	0
dense_1 (Dense)	(None, 4096)	102764544
dropout_1 (Dropout)	(None, 4096)	0
dense_2 (Dense)	(None, 4096)	16781312
dropout_2 (Dropout)	(None, 4096)	0
dense_3 (Dense)	(None, 2)	8194

Total params: 134,268,738
Trainable params: 134,268,738
Non-trainable params: 0

Summary of the model

```
from keras.callbacks import ModelCheckpoint, EarlyStopping

checkpoint = ModelCheckpoint("vgg16_1.h5", monitor='val_acc',
verbose=1, save_best_only=True, save_weights_only=False,
mode='auto', period=1)

early = EarlyStopping(monitor='val_acc', min_delta=0, patience=20,
verbose=1, mode='auto')

hist = model.fit_generator(steps_per_epoch=100,generator=traindata
```

```
hist = model.fit_generator(steps_per_epoch=100,generator=traindata,
validation_data= testdata, validation_steps=10,epochs=100,callbacks=
[checkpoint,early])
```

After the creation of the model I will import ModelCheckpoint and EarlyStopping method from keras. I will create an object of both and pass that as callback functions to fit_generator.

ModelCheckpoint helps us to save the model by monitoring a specific parameter of the model. In this case I am monitoring validation accuracy by passing **val_acc** to ModelCheckpoint. The model will only be saved to disk if the validation accuracy of the model in current epoch is greater than what it was in the last epoch.

EarlyStopping helps us to stop the training of the model early if there is no increase in the parameter which I have set to monitor in EarlyStopping. In this case I am monitoring validation accuracy by passing **val_acc** to EarlyStopping. I have here set **patience** to 20 which means that the model will stop to train if it doesn't see any rise in validation accuracy in 20 epochs.

I am using model.fit_generator as I am using ImageDataGenerator to pass data to the model. I will pass train and test data to fit_generator. In fit_generator steps_per_epoch will set the batch size to pass training data to the model and validation_steps will do the same for test data. You can tweak it based on your system specifications.

After executing the above line the model will start to train and you will start to see the training/validation accuracy and loss.

```
Epoch 1/100
2/2 [==============================] - 54s 27s/step - loss: 0.9895 - acc: 0.5625 - val_loss: 0.5468 - val_acc: 0.7
500

Epoch 00001: val_acc improved from -inf to 0.75000, saving model to vgg16_1.h5
Epoch 2/100
2/2 [==============================] - 24s 12s/step - loss: 0.8483 - acc: 0.6562 - val_loss: 0.1913 - val_acc: 0.9
062

Epoch 00002: val_acc improved from 0.75000 to 0.90625, saving model to vgg16_1.h5
Epoch 3/100
2/2 [==============================] - 16s 8s/step - loss: 0.3318 - acc: 0.8750 - val_loss: 0.0474 - val_acc: 1.00
00

Epoch 00003: val_acc improved from 0.90625 to 1.00000, saving model to vgg16_1.h5
Epoch 4/100
2/2 [==============================] - 14s 7s/step - loss: 0.2716 - acc: 0.9062 - val_loss: 0.2542 - val_acc: 0.84
38

Epoch 00004: val_acc did not improve from 1.00000
Epoch 5/100
2/2 [==============================] - 11s 6s/step - loss: 0.2479 - acc: 0.9219 - val_loss: 0.2267 - val_acc: 0.90
62

Epoch 00005: val_acc did not improve from 1.00000
Epoch 6/100
2/2 [==============================] - 11s 6s/step - loss: 0.1248 - acc: 0.9531 - val_loss: 0.1960 - val_acc: 0.90
62

Epoch 00006: val_acc did not improve from 1.00000
```

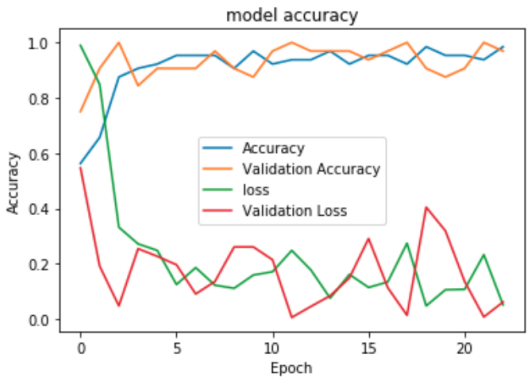
Training of the model

Once you have trained the model you can visualise training/validation accuracy and loss. As you may have noticed I am passing the output of mode.fit_generator to hist variable. All the training/validation accuracy and

loss are stored in hist and I will visualise it from there.

```
import matplotlib.pyplot as plt
plt.plot(hist.history["acc"])
plt.plot(hist.history['val_acc'])
plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.title("model accuracy")
plt.ylabel("Accuracy")
plt.xlabel("Epoch")
plt.legend(["Accuracy","Validation Accuracy","loss","Validation
Loss"])
plt.show()
```

Here I will visualise training/validation accuracy and loss using matplotlib.



training/validation accuracy and loss

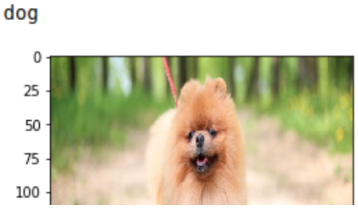
To do predictions on the trained model I need to load the best saved model and pre-process the image and pass the image to the model for output.

```
from keras.preprocessing import image

img = image.load_img("image.jpeg",target_size=(224,224))
img = np.asarray(img)
plt.imshow(img)
img = np.expand_dims(img, axis=0)

from keras.models import load_model
saved_model = load_model("vgg16_1.h5")

output = saved_model.predict(img)
if output[0][0] > output[0][1]:
    print("cat")
else:
    print('dog')
```



Output of the model

Here I have loaded the image using image method in keras and converted it to numpy array and added an extra dimension to the image to image for matching NHWC (Number, Height, Width, Channel) format of keras.

This is a complete implementation of VGG16 in keras using ImageDataGenerator. We can make this model work for any number of classes by changing the the unit of last softmax dense layer to whatever number we want based on the classes which we need to classify

Github repo link : https://github.com/1297rohit/VGG16-In-Keras

If you have less amount of data then instead of training your model from scratch you can try **Transfer Learning**. I have also written a step by step guide for beginners on performing transfer learning on VGG16 using Keras. You can check it out at : https://medium.com/@1297rohit/transfer-learning-from-scratch-using-keras-339834b153b9

If you would like to learn step by step about **Face Detection and Face Recognition** from scratch then you can head over to my article on that topic on the link : https://medium.com/@1297rohit/step-by-step-face-recognition-code-implementation-from-scratch-in-python-cc95fa041120

Enjoy Classification !

More from Towards Data Science

Follow

Your home for data science. A Medium publication sharing concepts, ideas and codes.

Alex Polyakov · Aug 6, 2019 ★

How to attack Machine Learning (Evasion, Poisoning, Inference, Trojans, Backdoors)

In my previous article i mentioned three categories of AI threats (espionage, sabotage, and fraud). If looking at a technical level, attacks…

Machine Learning 16 min read

Yorgos Askalidis · Aug 6, 2019 ★

Demystifying data science roles

How to navigate the data science job postings, and find the one that is right for you! — The “data scientist” has cemented its legacy in pop culture as the vague buzzword describing anyone who can access and…

Data Science 9 min read

Rinu Gour · Aug 6, 2019

Reasons Why You Shouldn’t Consider Data Science as an option anymore. Wait, did I say shouldn’t!

“Information is the oil of the 21st century, and analytics is the combustion engine.” — The power of big data and data science is…

Data Science 6 min read

Neil Chandarana · Aug 6, 2019 ★

4 Ways To Supercharge Your Recommendation System

4 practical steps to improve scalability and quality in front of users. — Recommender systems help users find items they like. They do so by producing a predicted likeliness score or a list of top recommended ite…

Data Science 8 min read

Brian Mwangi. · Aug 6, 2019

Using Rtweet R package to analyze the plight of Kenyans with the Kenya Power & Lighting Company(KPLC).

Introduction. The Kenya Power & Lighting Company (KPLC) transmits.

distributes and retails electricity to customers throughout Kenya. It is t…

Data Science 6 min read

Read more from Towards Data Science