https://blog.keras.io/building-powerful-image-classification

All Images Videos Books News More Tools

About 4,02,000 results (0.51 seconds)

Keras
https://blog.keras.io › building-powerful-image-classif...

Building powerful image classification models using very ...

05-Jun-2016 — In this tutorial, we will present a few simple yet effective methods that you can use to **build** a **powerful image classifier**, using only very ...

https://blog.keras.io

The Keras Blog

Building a simple **Keras** + deep learning REST API ... Vision with Python, a new book on deep learning for computer vision and **image recognition** using **Keras**.

People also ask

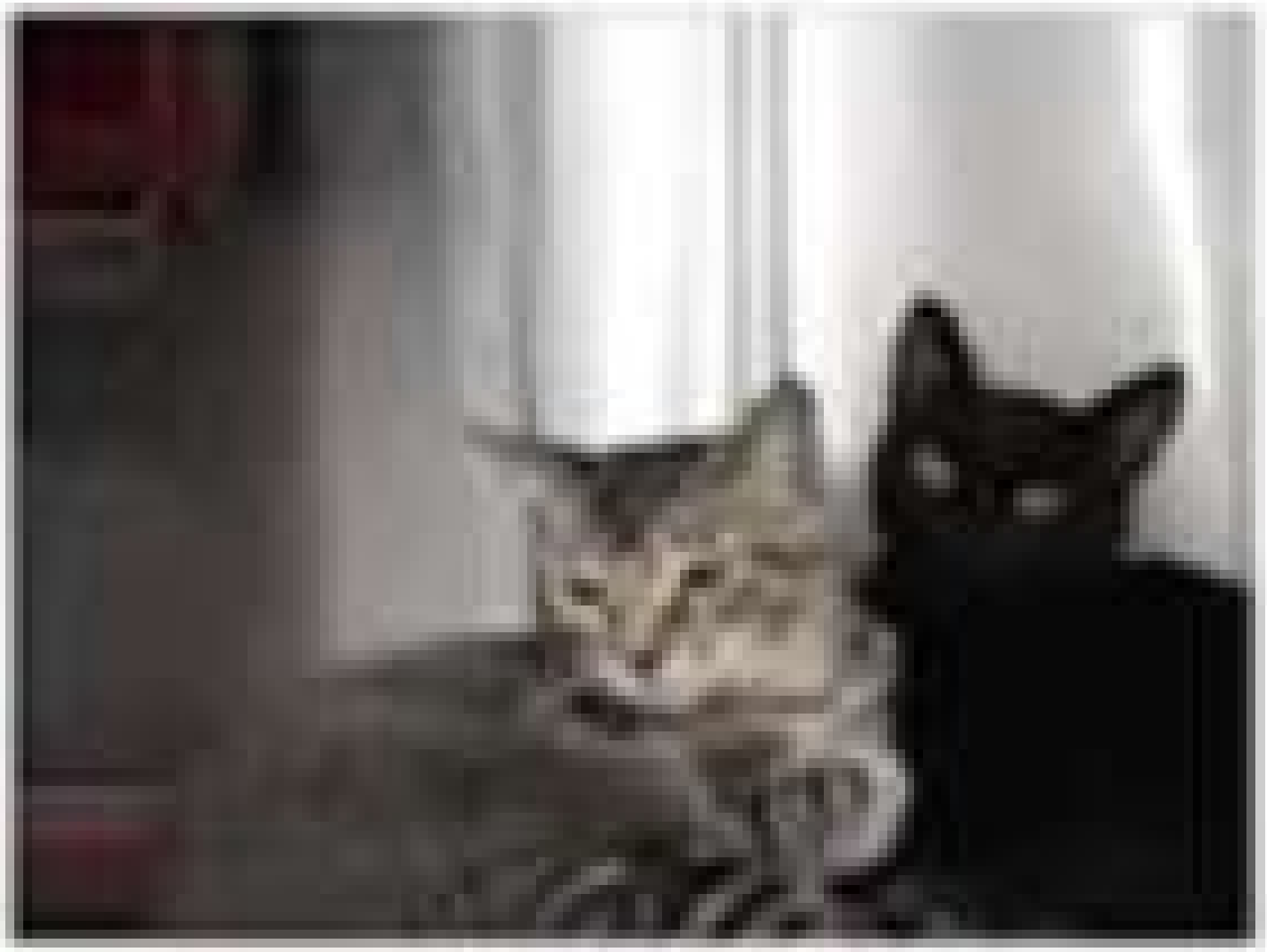
Which Keras model is best for image classification?

What is the best model for image classification?

How to do image classification using Keras?

How do you get high accuracy in image classification?

Feedback



jupyter single-image-data-augumentation Last Checkpoint: 4 hours ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted Python 3

Run Code

In [2]:
```
from keras.preprocessing import image
from keras.preprocessing.image import ImageDataGenerator
```

```
Using TensorFlow backend.
```

In [3]:
```
img = image.load_img('train/cat.18.jpg',target_size=(200,200))
```

In [5]:
```
type(img)
```

Out[5]:
```
PIL.Image.Image
```

In [36]:
```
datagen = ImageDataGenerator(
    rotation_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    vertical_flip=True,
    width_shift_range=0.2,
    height_shift_range=0.2,
)
```

In [7]:
```
img = image.img_to_array(img)
```

In [8]:
```
type(img)
```

Out[8]:
```
numpy.ndarray
```

In [38]:
```
img.shape
```

Out[38]:
```
(200, 200, 3)
```

In [39]:
```
input_batch = img.reshape(1,200,200,3)
```

In [40]:
```
i=0

for output in datagen.flow(input_batch,batch_size=1,save_to_dir='aug'):

    i = i + 1

    if i == 10:
        break
```

jupyter cat-dog-normal Last Checkpoint: 2 hours ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted Python 3

Run Code

In [1]:
```
import cv2
import os
import random
import numpy as np
```

In [2]:
```
mydir = r'C:\Users\91842\batch 6\cnn-lecture-1\data\train'
```

In [3]:
```
categories = ['cats','dogs']
```

In [4]:
```
data = []

for i in categories:

    folder_path = os.path.join(mydir,i)

    if i == 'cats':
        label = 0
    else:
        label = 1
```

In [3]:
```
categories = ['cats','dogs']
```

In [4]:
```
data = []

for i in categories:

    folder_path = os.path.join(mydir,i)

    if i == 'cats':
        label = 0
    else:
        label = 1

    for j in os.listdir(folder_path):

        img_path = os.path.join(folder_path,j)
        img = cv2.imread(img_path)
        img = cv2.resize(img,(150,150))
        data.append([img,label])
```

```
        data.append([img,label])
```

In [5]:
```
random.shuffle(data)
```

In [6]:
```
X = []
y = []
for i in data:

    X.append(i[0])
    y.append(i[1])
```

In [7]:
```
y = np.array(y)
```

In [8]:
```
X = np.array(X)
```

In [9]:
```
X.shape
```

Out[9]:
```
(2000, 150, 150, 3)
```

In [7]:
```
y = np.array(y)
```

In [8]:
```
X = np.array(X)
```

In [9]:
```
X.shape
```

Out[9]:
```
(2000, 150, 150, 3)
```

In [10]:
```
X = X/255
```

```
from keras import Sequential
from keras.layers import Conv2D,MaxPooling2D,Dense,Flatten,Activation,Dropout
```

```
Using TensorFlow backend.
```

In [12]:

```
model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=(150, 150,3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(32, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
```

In [13]:
```
model.add(Flatten())  # this converts our 3D feature maps to 1D feature vectors
model.add(Dense(64))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(1))
model.add(Activation('sigmoid'))

model.compile(loss='binary_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

In [14]:
```
model.fit(X,y,epochs=5,validation_split=0.1)
```

```
Train on 1800 samples, validate on 200 samples
Epoch 1/5
1800/1800 [==============================] - 12s 6ms/step - loss: 0.7444 - accuracy: 0.5244 - val_loss: 0.6911 - val_accuracy:
0.5050
Epoch 2/5
1800/1800 [==============================] - 5s 3ms/step - loss: 0.6903 - accuracy: 0.5561 - val_loss: 0.6895 - val_accuracy:
```

In [14]:
```
model.fit(X,y,epochs=5,validation_split=0.1)
```

```
Train on 1800 samples, validate on 200 samples
Epoch 1/5
1800/1800 [==============================] - 12s 6ms/step - loss: 0.7444 - accuracy: 0.5244 - val_loss: 0.6911 - val_accuracy:
0.5050
Epoch 2/5
1800/1800 [==============================] - 5s 3ms/step - loss: 0.6903 - accuracy: 0.5561 - val_loss: 0.6895 - val_accuracy:
0.5350
Epoch 3/5
1800/1800 [==============================] - 5s 3ms/step - loss: 0.6558 - accuracy: 0.6217 - val_loss: 0.6386 - val_accuracy:
0.6600
Epoch 4/5
1800/1800 [==============================] - 5s 3ms/step - loss: 0.6338 - accuracy: 0.6817 - val_loss: 0.6079 - val_accuracy:
0.7150
Epoch 5/5
1800/1800 [==============================] - 5s 3ms/step - loss: 0.5640 - accuracy: 0.7261 - val_loss: 0.7446 - val_accuracy:
0.5750
```

Out[14]:
```
<keras.callbacks.callbacks.History at 0x27bd62d2088>
```

In [ ]:

jupyter cat-dogs-with-data-augmentation Last Checkpoint: 10 hours ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted Python 3

Run Code

In [15]:
```
from keras.preprocessing.image import ImageDataGenerator
```

In [16]:
```
batch_size = 16

# this is the augmentation configuration we will use for training
train_datagen = ImageDataGenerator(
        rescale=1./255,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True)

# this is the augmentation configuration we will use for testing:
# only rescaling
test_datagen = ImageDataGenerator(rescale=1./255)

# this is a generator that will read pictures found in
# subfolers of 'data/train', and indefinitely generate
# batches of augmented image data
```

```
# this is a generator that will read pictures found in
# subfolers of 'data/train', and indefinitely generate
# batches of augmented image data
train_generator = train_datagen.flow_from_directory(
        'data/train',  # this is the target directory
        target_size=(150, 150),  # all images will be resized to 150x150
        batch_size=batch_size,
        class_mode='binary')  # since we use binary_crossentropy loss, we need binary labels

# this is a similar generator, for validation data
validation_generator = test_datagen.flow_from_directory(
        'data/valid',
        target_size=(150, 150),
        batch_size=batch_size,
        class_mode='binary')
```

```
Found 2000 images belonging to 2 classes.
Found 1000 images belonging to 2 classes.
```

In [17]:

```
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Activation, Dropout, Flatten, Dense

model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=(150, 150,3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(32, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
```

```
Found 1000 images belonging to 2 classes.
```

In [17]:
```
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Activation, Dropout, Flatten, Dense

model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=(150, 150,3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(32, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
```

In [18]:
```
model.add(Flatten())  # this converts our 3D feature maps to 1D feature vectors
model.add(Dense(64))
model.add(Activation('relu'))
```

```
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(1))
model.add(Activation('sigmoid'))

model.compile(loss='binary_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

In [19]:

```
model.fit_generator(
        train_generator,
        steps_per_epoch=2000 // batch_size,
        epochs=25,
        validation_data=validation_generator,
        validation_steps=800 // batch_size)
```

```
Epoch 1/25
125/125 [==============================] - 37s 293ms/step - loss: 0.7147 - accuracy: 0.5465 - val_loss: 0.6834 - val_accuracy:
0.5088
Epoch 2/25
125/125 [==============================] - 34s 271ms/step - loss: 0.6867 - accuracy: 0.5770 - val_loss: 0.6183 - val_accuracy:
0.6338
Epoch 3/25
125/125 [==============================] - 34s 272ms/step - loss: 0.6357 - accuracy: 0.6365 - val_loss: 0.8259 - val_accuracy:
0.5770
```

```
Epoch 1/25
125/125 [==============================] - 37s 293ms/step - loss: 0.7147 - accuracy: 0.5465 - val_loss: 0.6834 - val_accuracy:
0.5088
Epoch 2/25
125/125 [==============================] - 34s 271ms/step - loss: 0.6867 - accuracy: 0.5770 - val_loss: 0.6183 - val_accuracy:
0.6338
Epoch 3/25
125/125 [==============================] - 34s 272ms/step - loss: 0.6357 - accuracy: 0.6365 - val_loss: 0.8259 - val_accuracy:
0.5770
Epoch 4/25
125/125 [==============================] - 31s 250ms/step - loss: 0.6191 - accuracy: 0.6745 - val_loss: 0.3901 - val_accuracy:
0.7172
Epoch 5/25
125/125 [==============================] - 34s 275ms/step - loss: 0.6031 - accuracy: 0.6955 - val_loss: 0.6000 - val_accuracy:
0.6925
```

```
Epoch 20/25
125/125 [==============================] - 54s 436ms/step - loss: 0.4234 - accuracy: 0.8125 - val_loss: 0.6448 - val_accuracy:
0.7400
Epoch 21/25
125/125 [==============================] - 58s 462ms/step - loss: 0.4118 - accuracy: 0.8220 - val_loss: 1.1208 - val_accuracy:
0.7449
Epoch 22/25
125/125 [==============================] - 56s 445ms/step - loss: 0.4186 - accuracy: 0.8040 - val_loss: 0.4275 - val_accuracy:
0.7576
Epoch 23/25
125/125 [==============================] - 54s 435ms/step - loss: 0.4092 - accuracy: 0.8220 - val_loss: 0.6022 - val_accuracy:
0.7348
Epoch 24/25
125/125 [==============================] - 53s 423ms/step - loss: 0.4064 - accuracy: 0.8275 - val_loss: 0.5712 - val_accuracy:
0.7399
Epoch 25/25
125/125 [==============================] - 40s 322ms/step - loss: 0.4065 - accuracy: 0.8230 - val_loss: 0.3671 - val_accuracy:
0.7487
```

Out[19]: <keras.callbacks.callbacks.History at 0x29276314ec8>

In [ ]: