【高精度な画像分類器作りに挑戦！】(2)ファインチューニングで高精度化

tmtk

Topics

- はじめに

- 実験

- まとめ

- 参考

こんにちは。データサイエンスチーム tmtkです。
この記事では、桜とコスモスの写真を分類する機械学習モデルを、**ファインチューニングを用いて作成**します。

はじめに

前回の記事では、畳み込みニューラルネットワークをゼロから作成し、学習させることで、桜とコスモスの写真を分類する機械学習モデルを作成しました。
今回は、ImageNetのために作られた**VGG16**という畳み込みニューラルネットワークの一部を改変・再学習することで、桜とコスモスの写真を分類する機械学習モデルを作成します。このような方法を、**ファインチューニング**といいます。

実験

前回の記事と同じ準備の下で実験をします。
IPythonを起動します。

```
ipython3
```

まずは、念のため乱数のシードを固定します。

```
import numpy as np
import tensorflow as tf
import random as rn
import os
from keras import backend as K
os.environ['PYTHONHASHSEED'] = '0'
np.random.seed(0)
rn.seed(0)
session_conf = tf.ConfigProto(intra_op_parallelism_threads=1, inter_op_parallelism_threads=1)
tf.set_random_seed(0)
sess = tf.Session(graph=tf.get_default_graph(), config=session_conf)
K.set_session(sess)
```

次に、画像データを読み込みます。前回と違うところとして、VGG16用の前処理を画像データに施しています。

```
from keras.preprocessing import image
from keras.applications.vgg16 import preprocess_input
from sklearn.model_selection import train_test_split
import keras
import numpy as np
import os
input_shape = (224, 224, 3)
batch_size = 128
epochs = 12
num_classes = 2
x = []
y = []
for f in os.listdir("sakura"):
    x.append(image.img_to_array(image.load_img("sakura/"+f, target_size=input_shape[:2])))
    y.append(0)
for f in os.listdir("cosmos"):
    x.append(image.img_to_array(image.load_img("cosmos/"+f, target_size=input_shape[:2])))
    y.append(1)
x = np.asarray(x)
x = preprocess_input(x)
y = np.asarray(y)
y = keras.utils.to_categorical(y, num_classes)
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.33, random_state= 3)
```

```
0
2
1
2
2
2
3
```

VGG16のモデルを最後の全結合層を除いて読み込み、かわりに新しく別の全結合層を付け加えます。VGG16の部分の畳み込み層は再学習しないように設定します。

```
1  from keras.models import Sequential, Model
2  from keras.layers import Dense, GlobalAveragePooling2D
3  from keras.applications.vgg16 import VGG16
4  base_model = VGG16(weights='imagenet', include_top=False)
5  x = base_model.output
6  x = GlobalAveragePooling2D()(x)
7  x = Dense(1024, activation='relu')(x)
8  predictions = Dense(num_classes, activation='softmax')(x)
9  model = Model(inputs=base_model.input, outputs=predictions)
1  for layer in base_model.layers:
0      layer.trainable = False
1  model.compile(loss=keras.losses.categorical_crossentropy,
1                optimizer="rmsprop",
1                metrics=['accuracy'])
2
1
3
1
4
```

自分で付け加えた全結合層の部分を学習させます。

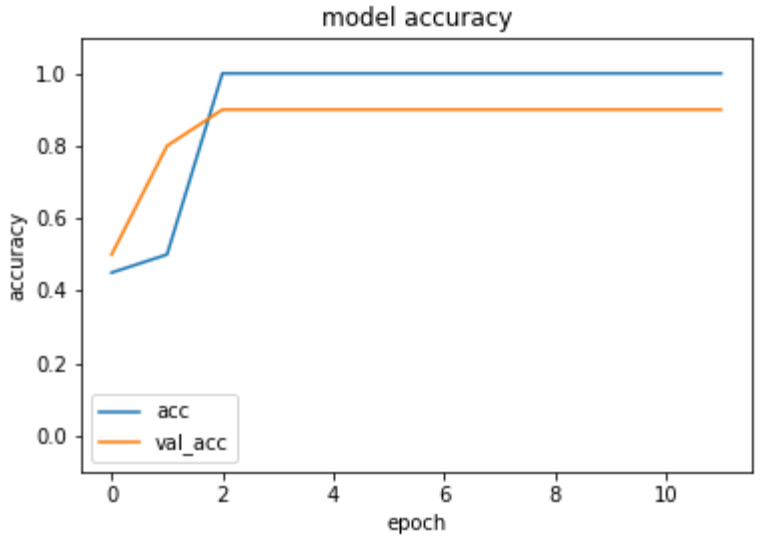
```
1  history = model.fit(x_train, y_train,
2            batch_size=batch_size,
3            epochs=epochs,
4            verbose=1,
5            validation_data=(x_test, y_test))
```

以下のように学習が進みます。

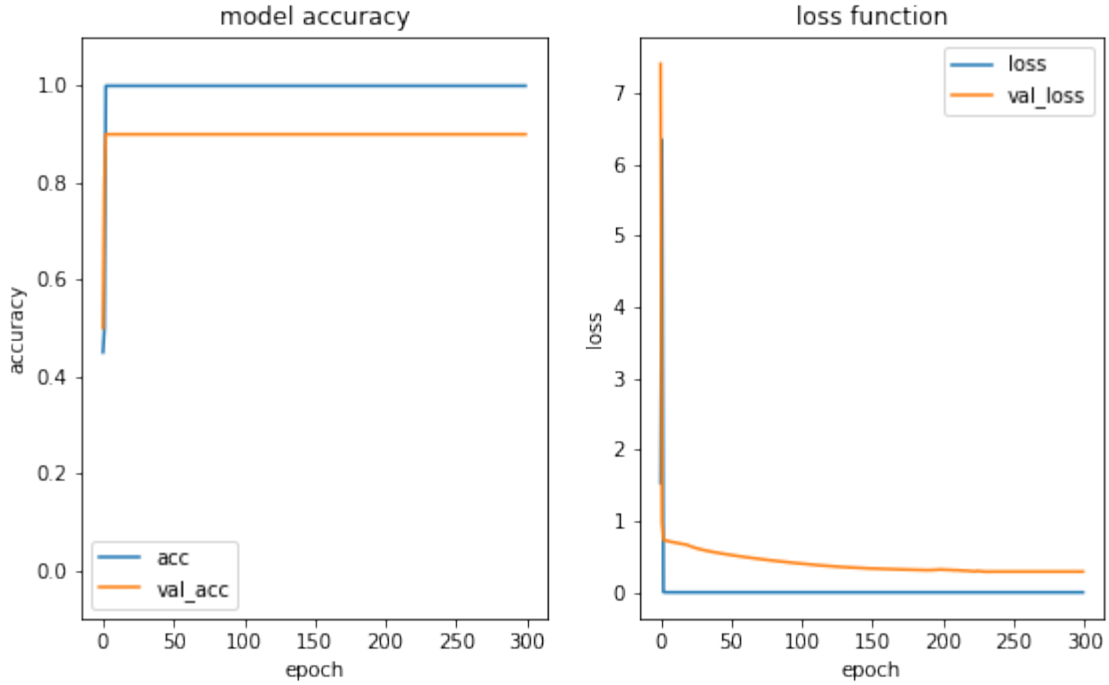
```
1  Train on 20 samples, validate on 10 samples
2  Epoch 1/12
3  20/20 [==============================] - 4s 204ms/step - loss: 1.5333 - acc: 0.4500 - val_loss: 7.412
4  3 - val_acc: 0.5000
5  Epoch 2/12
6  20/20 [==============================] - 0s 20ms/step - loss: 6.3482 - acc: 0.5000 - val_loss: 0.9846
7  - val_acc: 0.8000
8  Epoch 3/12
9  20/20 [==============================] - 0s 20ms/step - loss: 0.0029 - acc: 1.0000 - val_loss: 0.7341
1  - val_acc: 0.9000
0  ...
1  Epoch 10/12
1  20/20 [==============================] - 0s 20ms/step - loss: 4.4952e-05 - acc: 1.0000 - val_loss: 0.
1  7017 - val_acc: 0.9000
2  Epoch 11/12
1  20/20 [==============================] - 0s 21ms/step - loss: 4.1572e-05 - acc: 1.0000 - val_loss: 0.
```

```
6978 - val_acc: 0.9000
Epoch 12/12
20/20 [==============================] - 0s 20ms/step - loss: 3.8545e-05 - acc: 1.0000 - val_loss: 0.
6941 - val_acc: 0.9000
```

今回は、学習した結果、最終的に**訓練データに対する精度が100%、バリデーションデータに対する精度が90%**になっています。前回の記事でゼロから学習をしたのとは違い、VGG16というImageNetのために作られた汎用性の高いパラメータとモデルを使うことで、汎化性能と精度を得ることができました。また、学習時間も学習エポック数も前回の1/10程度になり、学習も前回より非常に速く進んでいることがわかります。



（訓練データとバリデーションデータに対する精度の推移）
また、エポック数を増やしてもこれ以上あまり学習は進みませんが、前回とは違って過学習には陥らないことがわかります。



（300エポックまで学習した場合）

まとめ

この記事では、桜とコスモスを分類する畳み込みニューラルネットワークを、VGG16をファインチューニングすることにより作成しました。
前回の記事でゼロから学習したのと比べて、学習も高速に進み、汎化性能も獲得することができました。
次回の記事では、VGG16を活用して、SVMへの転移学習によって機械学習モデルを作成し、100%の精度を達成します。

参考

- Building powerful image classification models using very little data
- Applications – Keras Documentation

2020.5.18 **GCP**2020.3

About us会社情報
セミナー・イベント
採用情報
執筆者への取材依頼
フォトギャラリー

Category AWS
Data Science
Tech
Event
Column
Tags
Members

商標について 個人情報保護方針 ISMS認証