畳み込みニューラルネット'
視化する

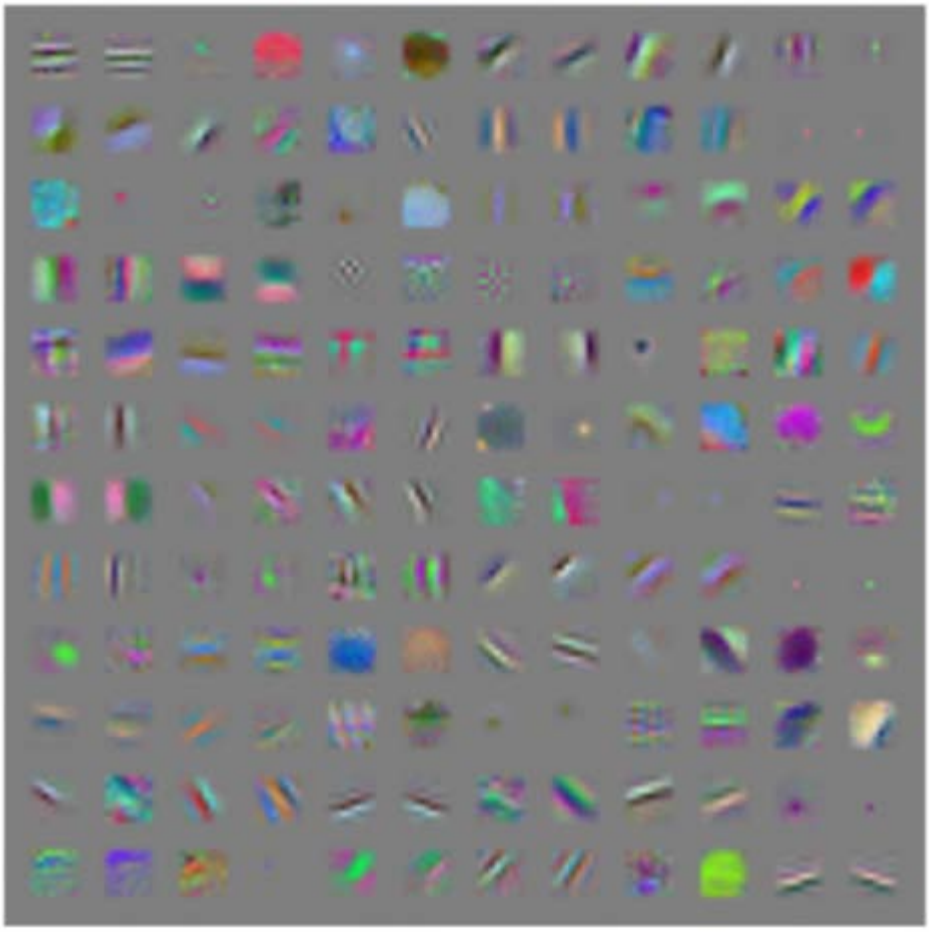
tmtk

Topics

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、Kerasをつかって学習済みResNet50の最初の畳み込み層の重みを可視化します。

はじめに

データサイエンスチームでは、Ian Goodfellow他『深層学習』の読書会をおこなっています。本書は、訳書の公式ウェブページ によれば、「深層学習の勉強のための決定版ともいえる教科書」です。訳書は今年の3月に発売されました。
本書の「第9章 畳み込みニューラルネットワーク」では、畳み込みニューラルネットワークの第1層の重みの可視化が掲載されています。



（原書ウェブ版　図9.19より引用）
この図を見ると、実際、縦・横・斜めのいろいろな色の縞々が重みの可視化として現れています。後ほど実際にためしてみますが、このように縞々状になっているカーネルで畳み込みをすると、縞々の方向のエッジが検出できます。したがって、この畳み込み層はエッジ検出器を学習している、ということになります。本書によれば、自然画像を学習した際にエッジ検出器を学習しない場合、そのアルゴリズムが悪い兆候だといえるそうです。
また、スタンフォード大学の講義資料 でも同様の図が示されています。本資料によれば、可視化した重みにノイズが多い場合は、学習が進んでいないか正則化が弱く過学習を引き起こしている兆候だということです。
この記事では、このような可視化を、具体的にどのようにすれば作成できるかを説明します。説明にあたっては、使いやすいKerasとPython 3を用います。畳み込み処理やカーネルの意味は既知であると仮定します。もし知らない場合は、ディープラーニング関連の書籍をご覧ください。

実験

実験のために、Keras、Python 3、Pillowなどがインストールされ たLinuxマシンを用意します。Amazon EC2インスタンスとDeep Learning AMIをつかい、pipでPillowをインストールすると簡単です。
可視化する対象の畳み込みニューラルネットワークを決めます。ここではResNet50を採用します。ResNet50はKerasで学習済みモデルが提供されていて、最初の畳み込み層のカーネルのサイズが大きいので、この実験に適切です。
IPythonやJupyter Notebookなどをつかい、Pythonのスクリプトが実行できる状態にします。
必要なライブラリをインポートします。

```
1 from keras.applications import ResNet50
2 import matplotlib.pyplot as plt
```

```
import numpy as np
from PIL import Image
```

ResNet50の学習済みモデルを読み込みます。初回はモデルのダウンロードにいくらかの時間がかかります。

```
resnet = ResNet50()
```

ここで`resnet.summary()`と実行すると、ネットワークの構造が出力されます。その出力をみると、最初の畳み込み層の名前が`conv1`だとわかるので、その層の重みを取得します。

```
weights = resnet.get_layer("conv1").get_weights()[0]
```

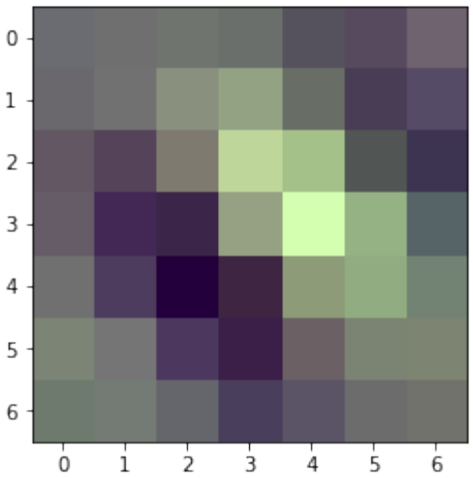
重み配列のサイズを見てみましょう。以下を実行すると、`(7, 7, 3, 64)`と出力されます。畳み込みカーネルの高さと幅がそれぞれ7, 7で、入力が3チャネル（青、緑、赤）、出力が64チャネルだということがわかります。これから、64個それぞれのチャネルのカーネルの重みを可視化していきます。

```
weights.shape
```

64個それぞれの重みをすべて可視化する前に、まずは0個目のカーネルを可視化してみます。畳み込み層の重みはどのような範囲の値をとるかわからないので、0から1の範囲に正規化します。また、KerasのResNet50のモデルはRGB形式でなくBGR形式で学習されているため、RGBからBGRに変換します。

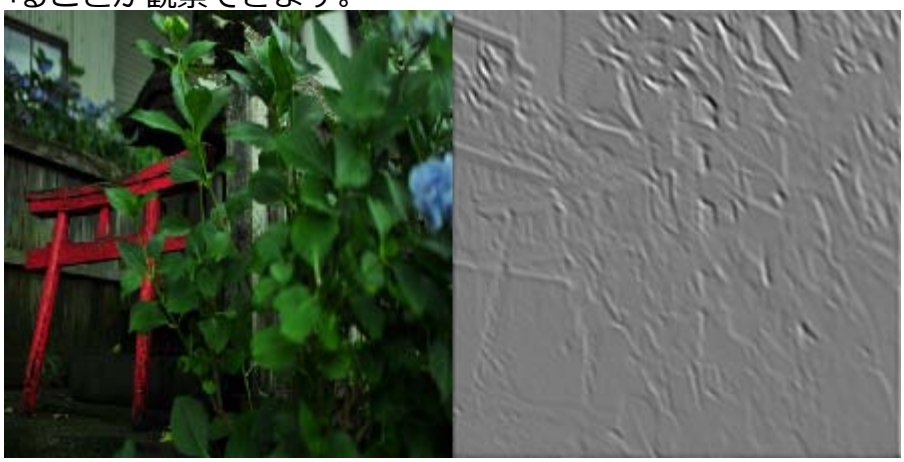
```
w = weights[:, :, ::-1, 0].copy()
m = w.min()
M = w.max()
w = (w-m)/(M-m)
plt.imshow(w)
```

すると、次のような画像が出力されます。



白い線が左上から右下に向かって走っています。この畳み込みカーネルではバックスラッシュ“＼”と同じ方向のエッジを検出できそうです。

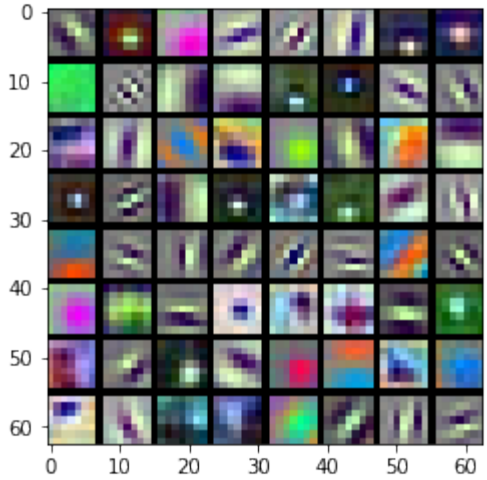
詳しい手順は省略しますが、実際にこの畳み込みカーネルを写真に対して適用すると、以下のように斜め方向のエッジが検出できていることが観察できます。



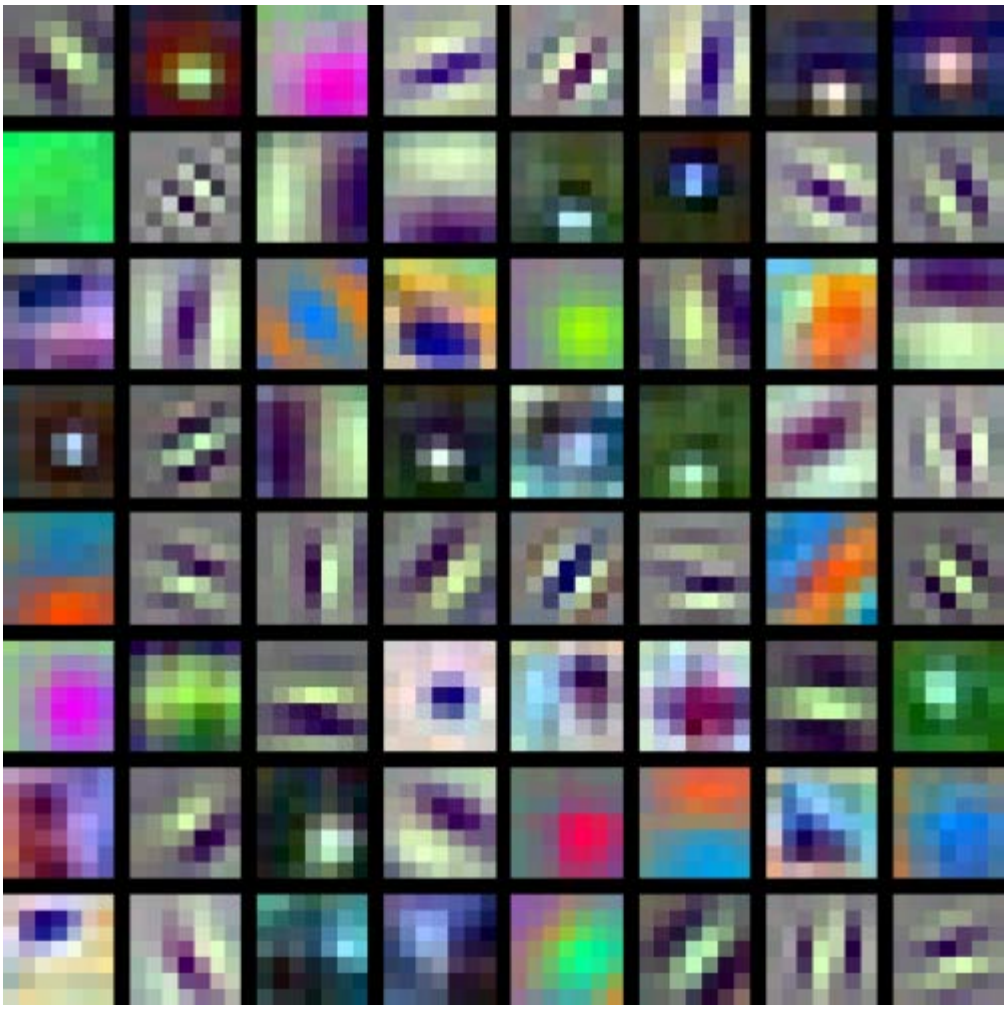
（上の畳み込みカーネルを左の写真に適用したものが右の画像。）
最後に、64個すべてのカーネルを並べて表示してみましょう。ここでは画像処理ライブラリPillowを使うことにします。
まず、大きい画像をあらわす変数`result`を用意します。そして、64枚それぞれのカーネルの可視化を`result`の各部分に貼り付けていき、画像を作成します。

```
result = Image.new("RGB", (7*8+(8-1), 7*8+(8-1)))
for i in range(64):
    w= weights[:, :, ::-1, i].copy()
    M = w.max()
    m = w.min()
    w = (w-m)/(M-m)
    w *= 255
    img = Image.fromarray(w.astype("uint8"), mode="RGB")
    result.paste(img, (7*(i//8) + (i//8), 7*(i%8)+(i%8)))
plt.imshow(result)
```

このような画像が生成されます。



このようにして、畳み込み層の重みを可視化することができました。
この画像を拡大すると、以下のようになります。



『深層学習』で述べられているように、最初の畳み込み層でエッジ検出器が学習されていることが観察できます。白と黒の縞々になっている部分がそれに該当します。

まとめ

この記事では、Kerasを用いて学習済み畳み込みニューラルネットワークの最初の畳み込み層の重みを可視化し、実際にエッジ検出器が学習されていることを観察しました。

参考

- Deep Learning （原書の公式ウェブサイト）
- Deep Learning （訳書の公式ウェブサイト）
- Keras Documentation （Kerasの公式サイト）
- CS231n Convolutional Neural Networks for Visual Recognition （スタンフォード大学の畳み込みニューラルネットワークと画像認識に関する講義の資料で、畳み込みニューラルネットワークの可視化についてのページ）

- 紫陽花と祠｜ぱくたそフリー素材　（実験に使用した写真の出典）

#Keras#Python#ニューラルネットワーク#深層学習#画像認識

データ分析と機械学習とソフトウェア開発をしています。 アルゴリズムとデータ構造が好きです。