@icoxfog417 2017年11月27日に更新

Convolutional Neural Networkとは何なのか

Python 機械学習

機械学習の世界において、画像といえばConvolutional Neural Network(以下CNN)というのは、うどんといえば香川くらい当たり前のこととして認識されています。しかし、そのCNNとは何なのか、という解説は意外と少なかったりします。

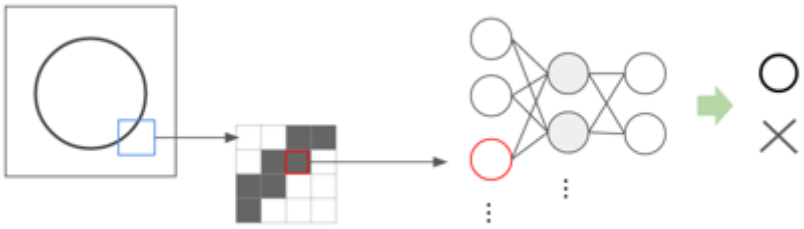
そこで、本記事ではCNNについてその仕組みとメリットの解説を行っていきたいと思います。

なお、参考文献にも記載の通り解説の内容はStanfordのCNNの講座をベースにしています。こちらの講座はNeural NetworkからCNN、はてはTensorflowによる実装まで解説される予定なので、興味がある方はそちらもご参照ください。

Convolution Neural Networkとは

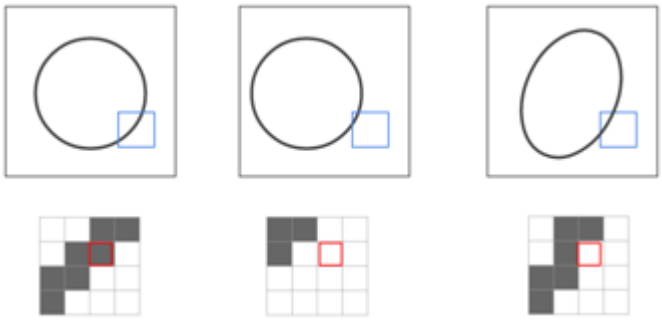
CNNはその名の通り通常のNeural NetworkにConvolutionを追加したものです。ここでは、Convolution、畳み込みとは一体なんなのか、という点と、なぜそれが画像認識に有効なのかについて説明していきます。

簡単なタスクとして、書いてある図形が○か×かを判定するタスクを考えてみます。以下は、通常のNeural Networkで行う例です。



画像の1ピクセルが1入力に対応していると思ってください。10x10の画像であれば、入力はサイズ100のベクトルになります(なお、RGB表現の場合ここにx3となります)。

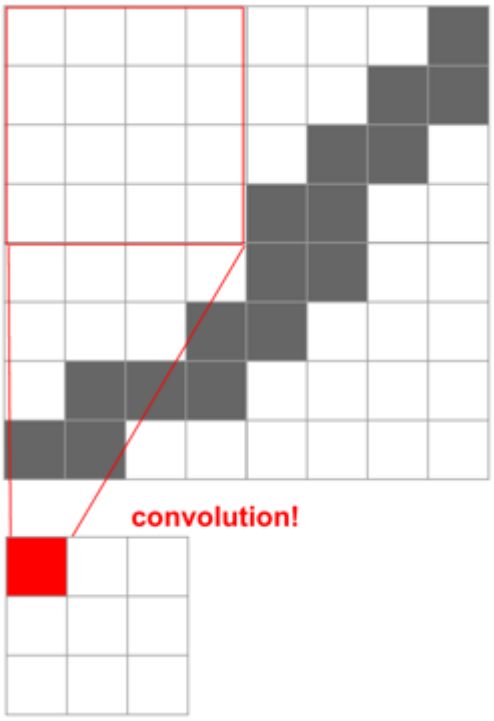
図中では円の淵の黒い部分が入力として渡っていく様子を示していますが、これを見ると、少し位置がずれていたりすると判定に大きな影響が出ることがわかります。下図のように位置や形が少し変わると、入力される情報もずれて認識されてしまうためです。



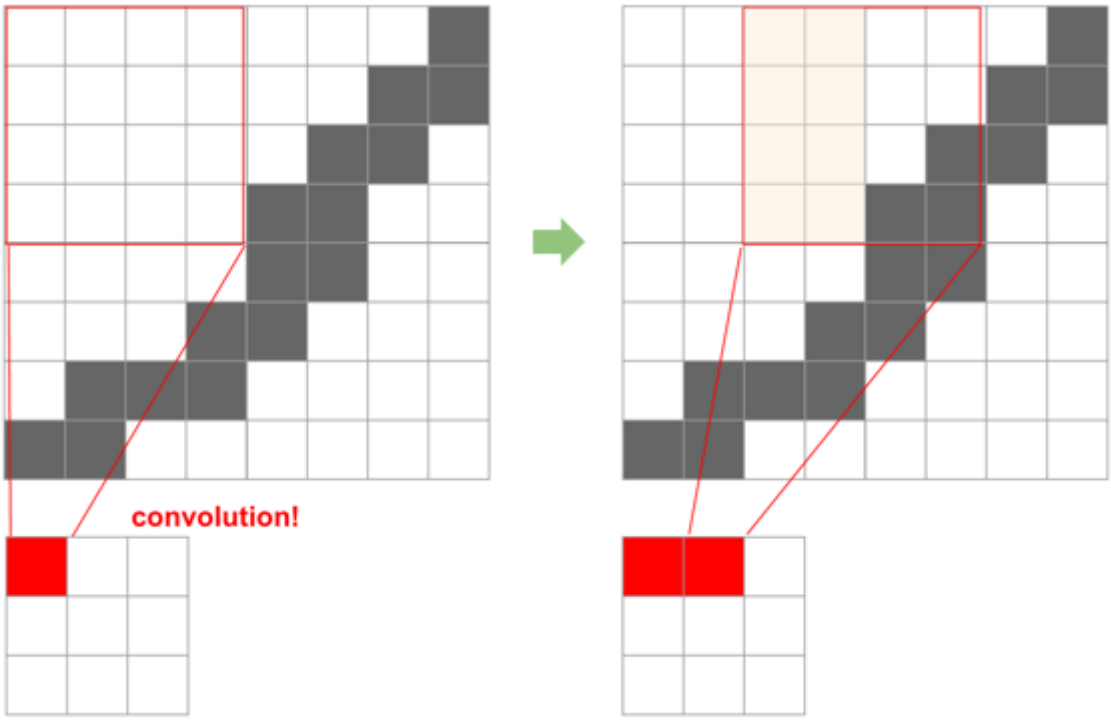
ただ、図の青い四角の中は概ね「右上から左下にかけて黒」という傾向があります。つまり、1ピクセルではなくある程度の広さの領域をまとめて入力にすることができれば、より精度の高い判定ができそうです。

このアイデアを実現するのが、CNNです。

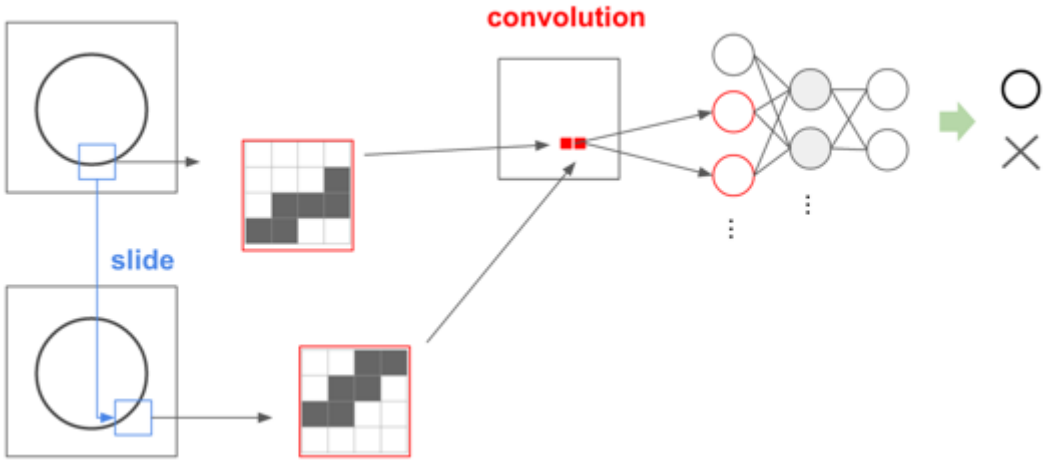
下図のように、画像上にフィルタと呼ばれる小領域(下図では赤枠の4x4のエリア)をとり、これを1つの特徴量として圧縮し(=畳み込み)ます。



この処理を、領域をスライドさせながら繰り返していきます。この結果作成されるのが、フィルタ内の情報が畳み込まれて作成されたレイヤ、Convolution Layerになります。



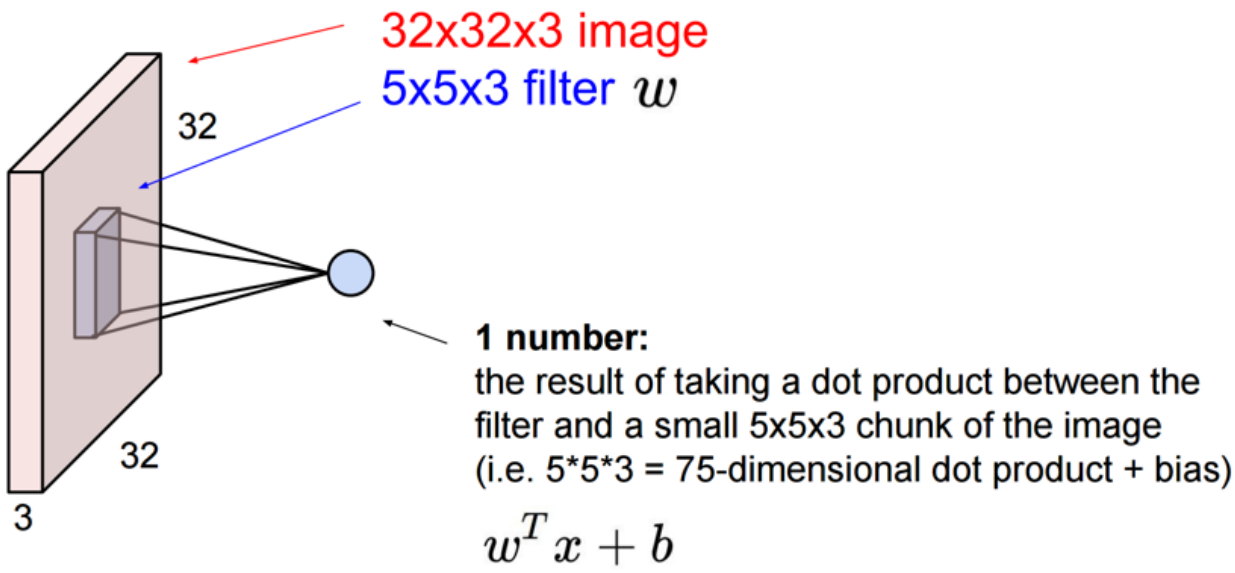
先ほどのNeural Networkの図をCNNにすると以下のようなイメージになります。



このフィルタを使った「畳み込む」という処理は、具体的には「フィルタ内の画像のベクトル」と「畳み込みに使用するベクトル」との間の掛け算、内積になります。

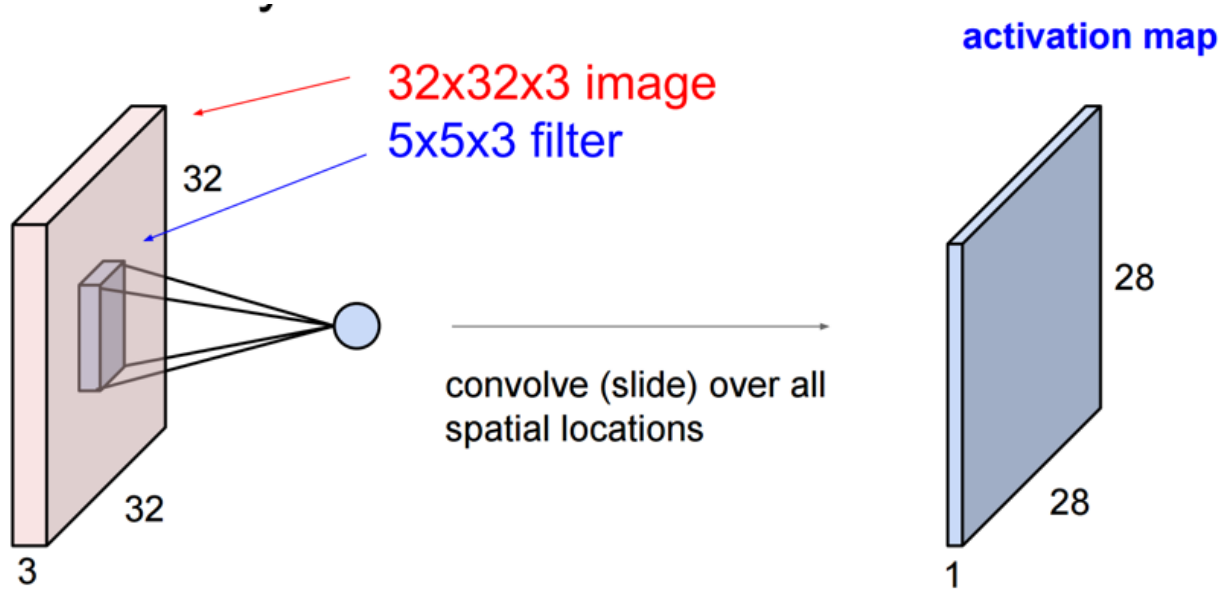
以 827 5x

す。



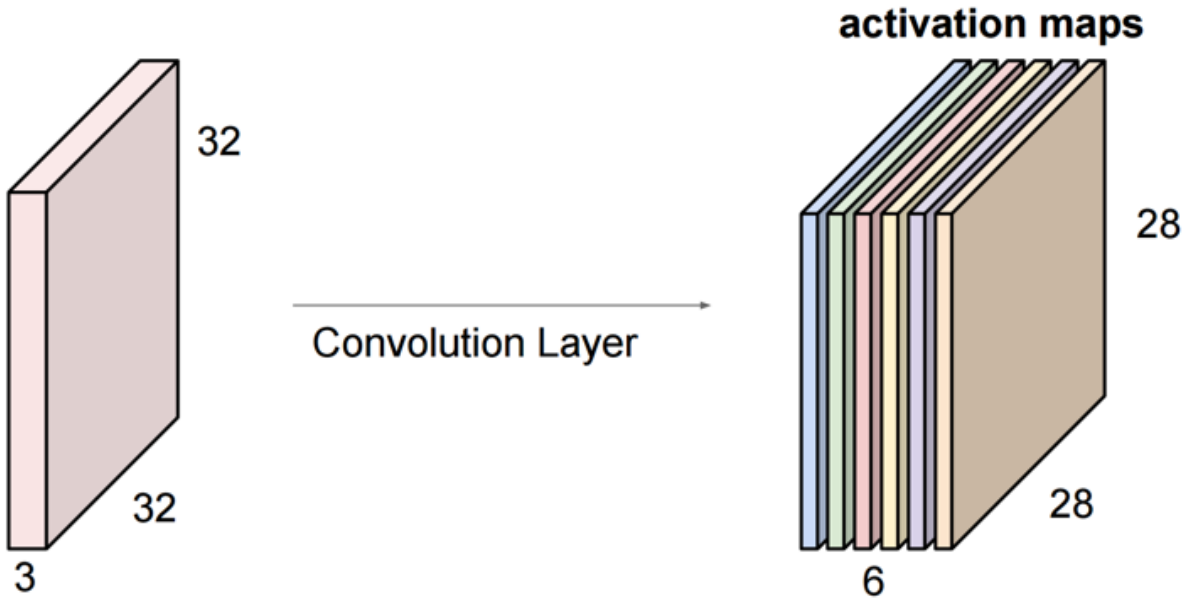
CS231n: Convolutional Neural Networks for Visual Recognition, Lecture7, p13

これにより、(スライド幅が1の場合)最終的には28x28x1のレイヤが作成されます。



CS231n: Convolutional Neural Networks for Visual Recognition, Lecture7, p14

そして、フィルタの種類を増やせばその分Convolution Layerも増えていくことになります。以下では6つのフィルタで6つのレイヤを作成しています。



CS231n: Convolutional Neural Networks for Visual Recognition, Lecture7, p16

これは、ちょうど畳み込みによって「新しい画像」を作っているとも言えます。こうして作った畳み込み層を通常のNeural Network同様、活性化関数でつないでいったものがConvolutional Neural Networkとなります(活性化関数としては、ReLUがよく使われます)。

ここまでの話をまとめておきます。

- CNNは、フィルタ内の領域の情報を畳み込んで作成するConvolution Layerを導入した、Neural Networkのことである
- Convolution Layerはフィルタを移動させながら適用することで作成し、フィルタの数だけ作成される。これを重ねて活性化関数(ReLU等)で繋いでいくことで、ネットワークを構築する。
- 畳み込みにより点ではなく領域ベースでの特徴抽出が可能になり、画像の移動や変形などに頑健になる。また、エッジなど領域ベースでないとわからない特徴抽出も可能になる。

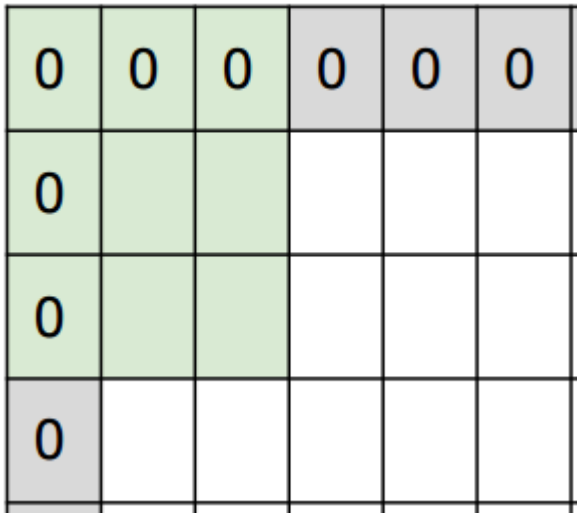
このCNNを特徴づけるのが、フィルタの設定とレイヤ構成になります。

フィルタの設定

畳み込みに使用するフィルタについて、設定しなければならないパラメーターが以下4つです。

- フィルタの数(K): 使用するフィルタの数。大体は2の累乗の値がとられる(32, 64, 128 ...)
- フィルタの大きさ(F): 使用するフィルタの大きさ
- フィルタの移動幅(S): フィルタを移動させる幅
- パディング(P): 画像の端の領域をどれくらい埋めるか

パディングは、以下のように画像の端の領域を0で埋める処理になります。



CS231n: Convolutional Neural Networks for Visual Recognition, Lecture7, p35

なぜこんなことをするかというと、普通に畳み込みを行うと端の領域はほかの領域に比べて畳み込まれる回数が少なくなってしまうためです。このように画像の端を0で埋め、そこからフィルタをかけていくことで端もほかの領域と同様に反映されるようにします。

なお、フィルタの大きさと移動幅については、きちんと画像の大きさに適合するよう調整する必要があります。以下のように、画像をはみ出てしまうようなフィルタの大きさ・移動幅は設定できないので注意してください。

これらのパラメーターの値から、Convolutional Layerのサイズを計算することが可能です。

32x32x3の画像に5x5x3のフィルタを、移動幅1、パディング2で適用するとします。ま

ず、パディングを加味すると画像のサイズは32+2*2=36となります。ここから幅5のフィルタを移動幅1でとる場合、36-5+1で32となります。つまり、最終的には32x32x3の層ができることになります。

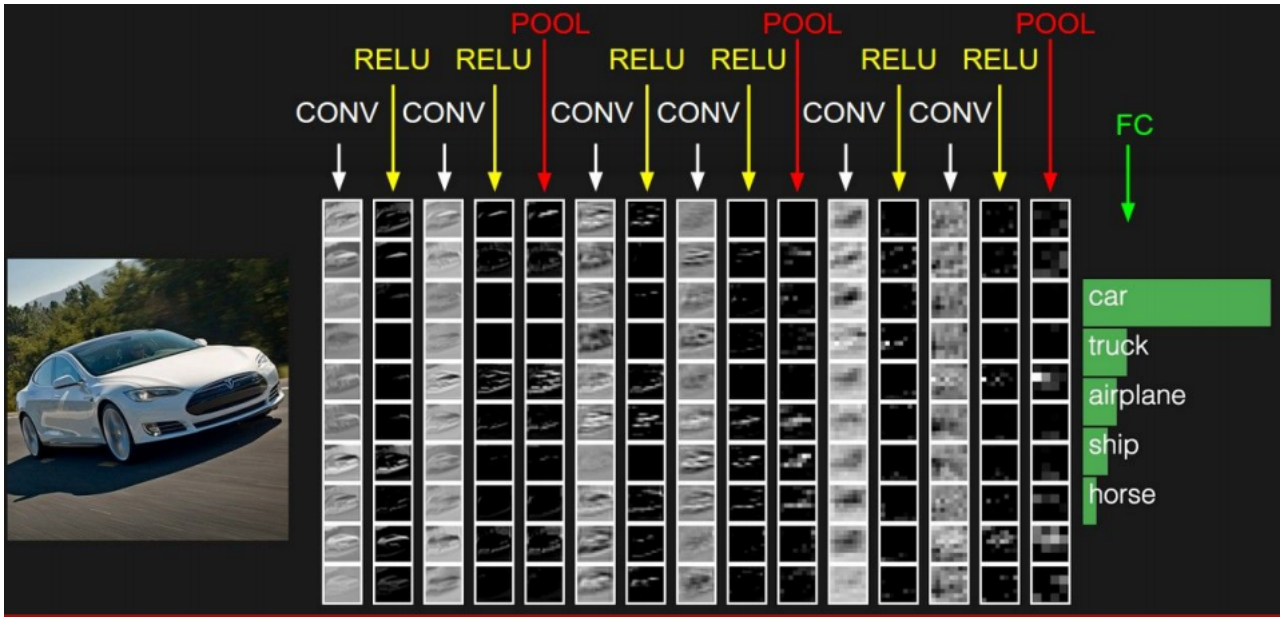
これらのパラメーターはCaffeなどのライブラリを使用する際にも設定が必要なため、その意味とサイズの計算方法を頭に入れておくとよいと思います。

レイヤ構成

CNNにおけるレイヤの種類としては、Convolutional Layerも含めて以下の3つがあります。

- Convolutional Layer: 特徴量の畳み込みを行う層
- Pooling Layer: レイヤの縮小を行い、扱いやすくするための層
- Fully Connected Layer: 特徴量から、最終的な判定を行う層

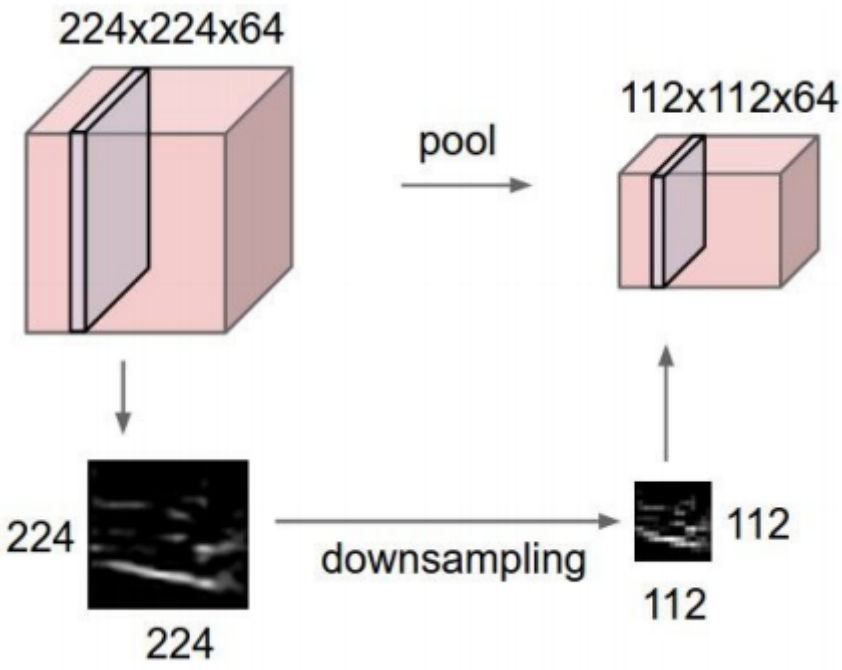
イメージ的には、以下のようになります。



CS231n: Convolutional Neural Networks for Visual Recognition, Lecture7, p22

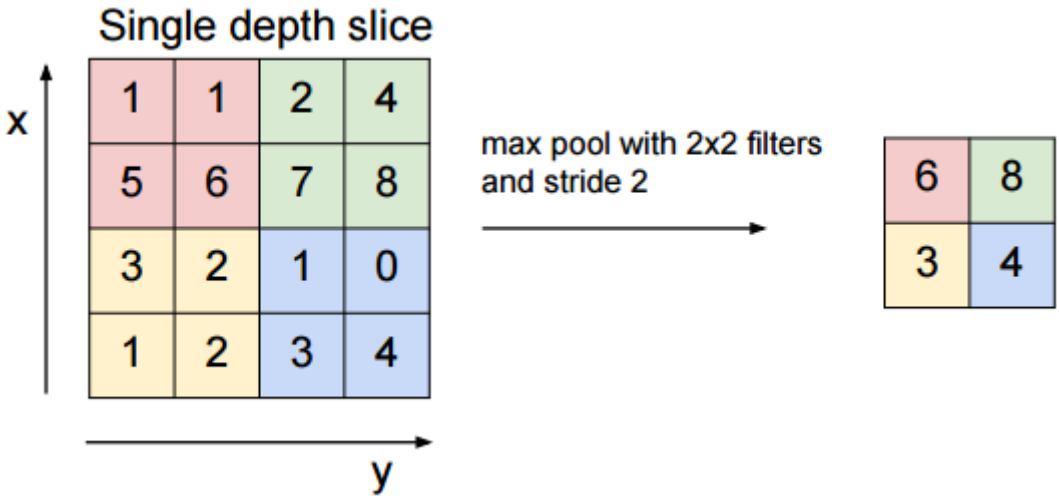
Convolutional Layer以外のレイヤについて、説明していきます。まずPooling Layerですが、これは画像の圧縮を行う層になります。画像サイズを圧縮して、後の層で扱いやす

くできるメリットがあります。



CS231n: Convolutional Neural Networks for Visual Recognition, Lecture7, p54

このPoolingを行う手段として、Max Poolingがあります。これは、各領域内の最大値をとって圧縮を行う方法です。



CS231n: Convolutional Neural Networks for Visual Recognition, Lecture7, p55

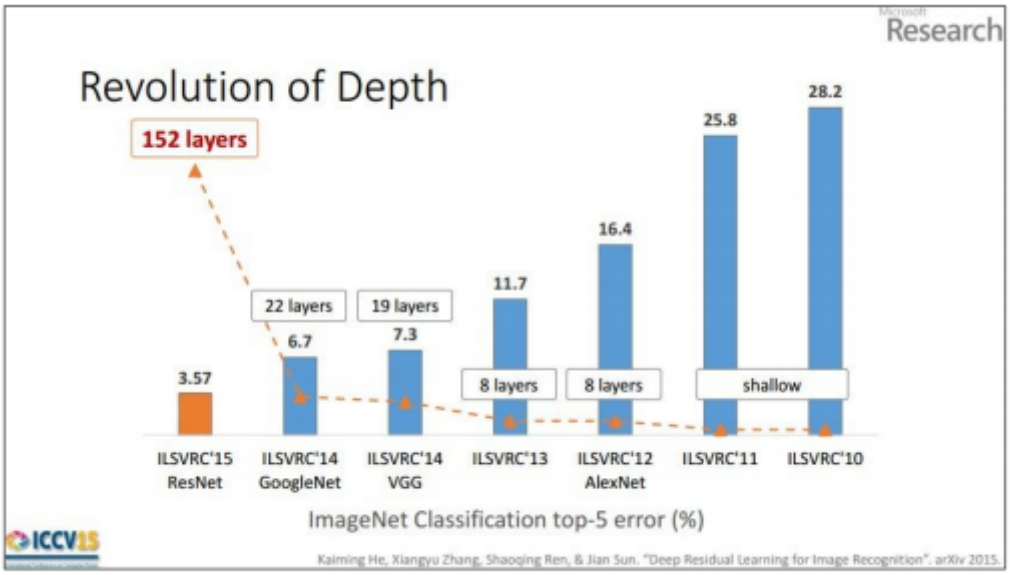
Fully Connected Layerは、前レイヤのすべての要素と接続するレイヤです。主に、最後の判定などを行う層で使用されます。
これらのレイヤを組み合わせることで、CNNを構築していきます。

CNNの進化

年が経るにつれ精度が上がってきているCNNですが、近年の構成では以下のような特徴がみられます。

- フィルタを小さくし、階層を深くする
- PoolingやFCのレイヤをなくす

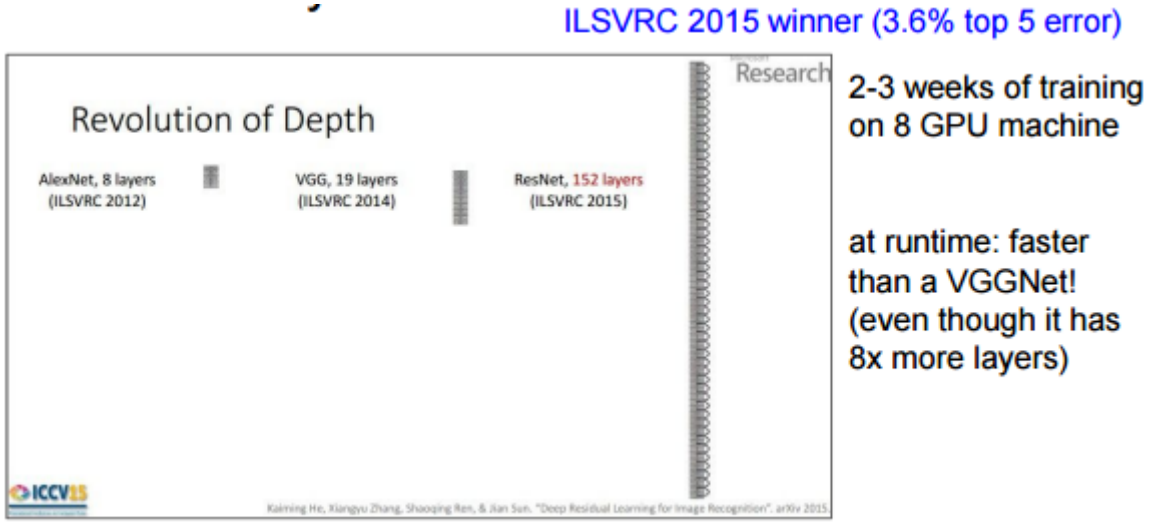
以下の図では、年々精度が上がるにつれレイヤが深くなっているのがわかります。



(slide from Kaiming He's recent presentation)

CS231n: Convolutional Neural Networks for Visual Recognition, Lecture7, p78

レイヤの深さについては、以下のほうがわかりやすいかもしれません。2012年に登場したAlexNetの8レイヤに対し、2015年の栄冠に輝いたResNetは152レイヤと大幅増となっています。



(slide from Kaiming He's recent presentation)

CS231n: Convolutional Neural Networks for Visual Recognition, Lecture7, p80

CNNの基本的な構成としては、以下のパターンが多いそうです。

(Convolution * **N** + (Pooling)) * **M** + Fully Connected * **K**

Nは~5くらいで、これを**M**層重ねて(Mは結構大きな値)、最後に判定のためのFCを**K**層(0<=K<=2)設けるという感じです(分類問題を扱うため、これにSoftMax関数を使った層をつけることもあります)。活性化関数としてはReLUが使用されることが多いです。

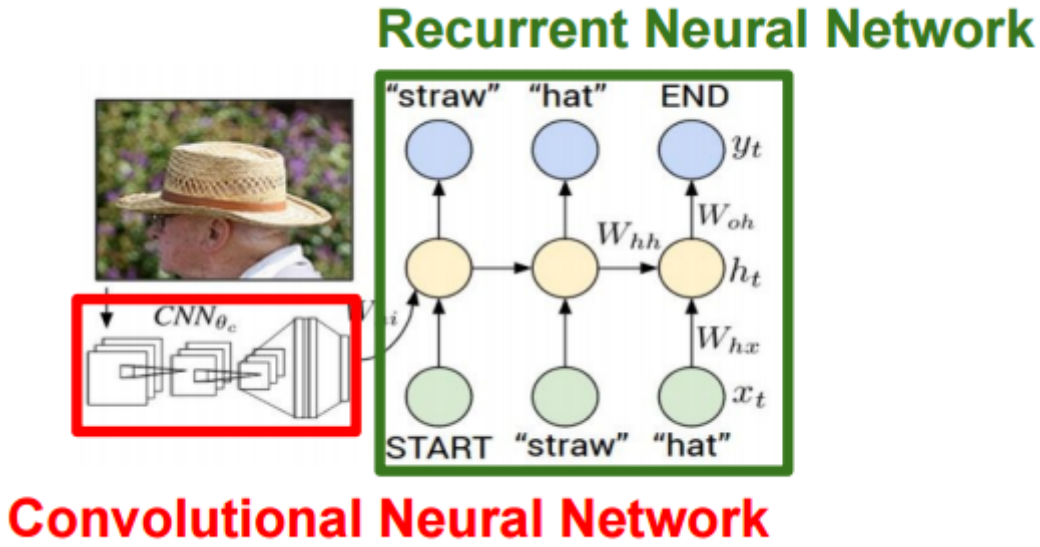
なおCNNはとても複雑そうに見えますが、重みをかけて伝播していくというNeural Networkの基本は外していないため、Neural Networkと同様Backpropagationによって学習させることが可能です。このあたりの柔軟性もNeural Networkの魅力と思います。

CNNの応用例

CNNは当初の画像はもちろん、それ以外のタスクにも応用されてきています。この応用例については以下のスライドにとてもよくまとまっているので、興味のある方は見てみてください。

Convolutional Neural Networks のトレンド

画像が識別できるCNNは、画像の特徴をよくとらえられる、とも言い換えることができます。つまり、識別の層を外したCNNは、入力された画像をその特徴を(識別が可能なほど)よく表すベクトルに変換するプロセスとも見ることができます。
応用例の幾つかはこの特徴を利用しており、特に画像に対してキャプションを付与するといった応用例は、CNNから抽出した画像の特徴量とテキスト情報を組み合わせています。



今後もいろいろな応用例が出てくると思いますし、昨今の機械学習フレームワークを利用すれば自分で試してみることも可能です。本記事が、その一助となれば幸いです。

参考文献

- CS231n: Convolutional Neural Networks for Visual Recognition
- Convolutional Neural Networks のトレンド

@icoxfog417

All my statements are from fun fancies, not a boring story that represents a company that I belonging to.

https://github.com/icoxfog417

フォロー

TIS株式会社

創業40年超のSIerです。

http://www.tis.co.jp/

この記事は以下の記事からリンクされています

 **Convolutional Neural Networkを実装する** からリンク 2016-03-24T05:00:06Z

 **画像の高速スタイル変換を行う論文の紹介** からリンク 2016-04-10T01:49:33Z

 **TensorFlowを使って顔認識器を作る** からリンク 2016-08-24T13:52:01Z

 **Amazon DSSTNEのディープラーニング設定項目一覧** からリンク 2016-09-20T08:40:42Z

 **ニューラルネットワークは電気ねずみの夢を見るか？** からリンク 2016-12-23T19:19:13Z

過去の13件を表示する

コメント

 @externvoid@github 2017-07-06 15:43

CNNが良く解りました。RNNについて知りたいので、解説論文を探してみます。どなたかポインターを示して頂けるとありがたいです。

1

 @habuyoshiaki 2017-08-17 09:35

丁寧な解説ありがとうございます。

0

@namahoge 2017-11-26 18:21

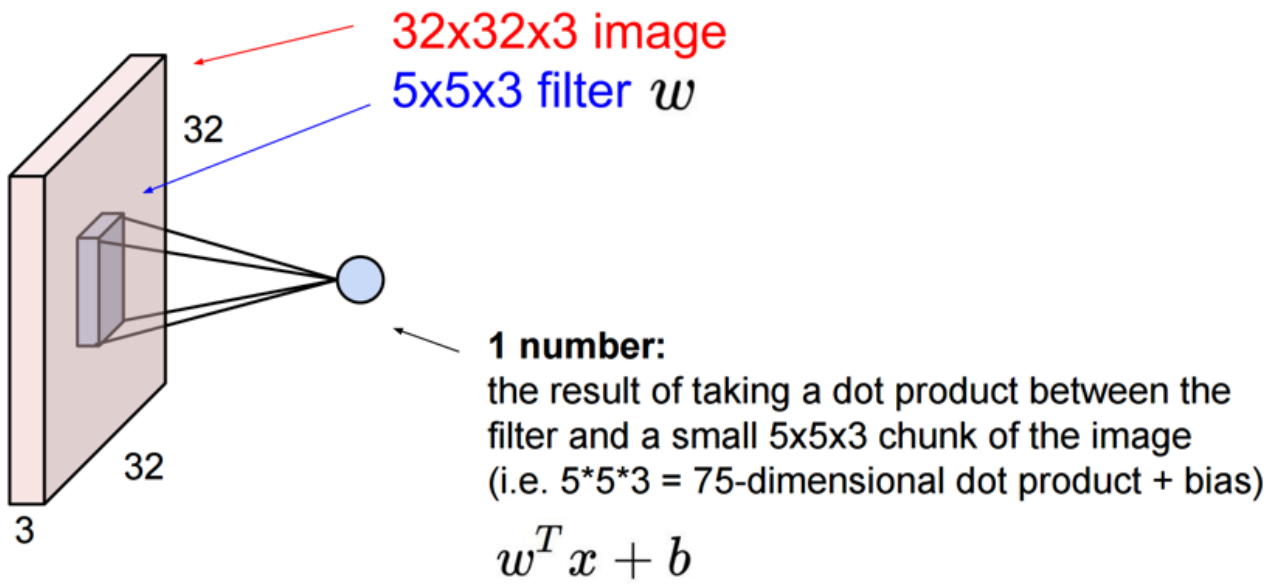
フィルタの設定
のところ、階乗じゃなくてべき乗だと思いますぉ

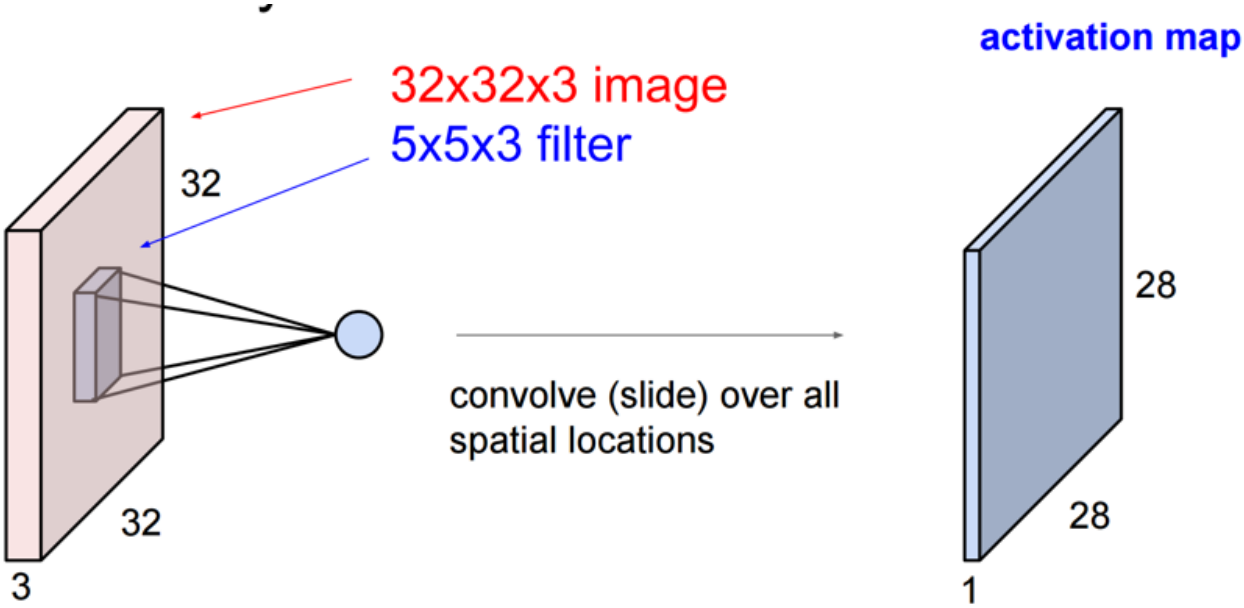
1

@Shirui 2018-06-23 17:27

僕の知識不足なのですが，以下の質問をさせて頂きたく，お願い致します．

①
カラー画像をConvolutionする際，実際に行われている具体的な計算は何でしょうか？





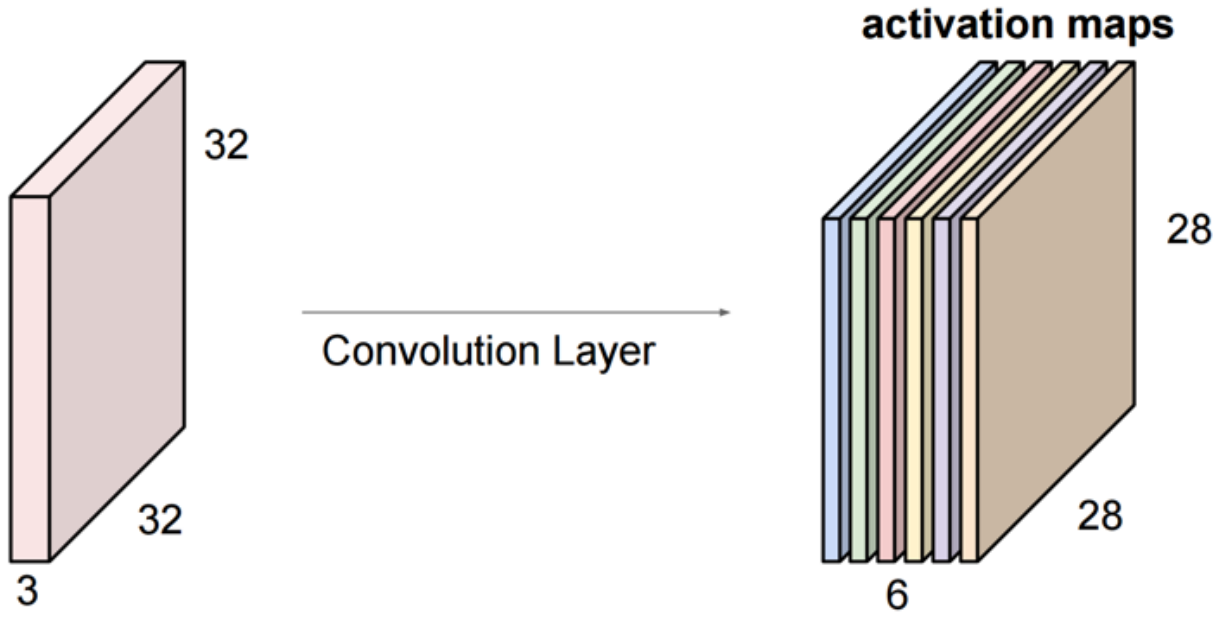
記事の元々のイメージを使用させて頂きますが，

1.
n次元の画像（x，y，n）は，n次元のフィルタ（a，b，n）を適用する，と理解して宜しいでしょうか？

2.
そして，計算することとしては，
元の画像（x，y）から（a，b）範囲のピクセルに対し，x＊y＊n個の要素を，順番に合わせて掛け算する．
（元画像[14][23][7]＊フィルタ[14][23][7]，こんな感じ）
そして，x＊y＊n個の乗算した後の要素を加算し，activation mapの1ピクセルが作成される．
（実際，生成されるactivation mapは，paddingのため，少し小さくなる．）

3.
フィルタをスライドし，最終的にactivation mapが完成．

②
また，複数のフィルタを使用した後，複数のactivation mapを得ることとなります．
「これらactivation mapを活性化関数で繋げる」とご記述ありますが，



記事の元々のイメージを使用させて頂いて下ります.

実際は，各フィルタを掛けた結果画像を，
フィルタの次元別に格納しているのではないでしょうか？

つまり，
元の画像サイズが[32][32][3]だとすると，
[5][5][3]フィルタが6個あった場合，得られる画像は，
[28][28][6]となります.

「活性化関数が使用されている箇所」は，
フィルタを掛け，得た画像ではないと思います.

「どこで使用するか」というと，先ず，画像をフラット処理します，
[28][28][6]の画像情報を，[1～28＊28*6]のデータ情報に変換します.

そして，全結合し，全結合の出力の際に，活性化関数を用い，データを扱います.
と僕は考えて下ります.

大変長くなり，申し訳ございませんが，宜しくお願い致します.

0

あなたもコメントしてみませんか :)

ユーザ登録

すでにアカウントを持っている方はログイン