Convolutional Neural Network for Image Recognition
Belami E-Commerce VAM Department 2018
By: Carlos Monsivais
August 28, 2018

Table of Contents

How Does Convolution Work?

Image

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

The filter is used as a sliding window on the image

Filter

$$\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

Here we are using a 3X3 filter because we are supposing that this image has all three colorways including red, green and blue.

Now the Convolution occurs by taking the dot product of the filter on the image creating a Convolution Matrix like so:

Image

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

*Calculate by using the dot product

Image **Row: 1 to 3** **Column: 1 to 3** $\begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$ × **Filter** $\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$ = 4	**Image** **Row: 1 to 3** **Column: 2 to 4** $\begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}$ × **Filter** $\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$ = 3	**Image** **Row: 1 to 3** **Column: 3 to 5** $\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}$ × **Filter** $\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$ = 4
Image **Row: 3 to 5** **Column: 1 to 3** $\begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$ × **Filter** $\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$ = 2	**Image** **Row: 3 to 5** **Column: 2 to 4** $\begin{pmatrix} 0 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix}$ × **Filter** $\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$ = 3	**Image** **Row: 3 to 5** **Column: 3 to 5** $\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$ × **Filter** $\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$ = 4

Using the results from above, we can put them together to create the Convolution Matrix below:

Convolution Matrix

$$\begin{pmatrix} 4 & 3 & 4 \\ 2 & 3 & 4 \end{pmatrix}$$

Convolution Equations

This is in a 1-Dimensional Space

$$(f \times g)(t) = \int_{-\infty}^{\infty} f(\tau) \times g(t-\tau) d\tau$$

The percentage of area the filter g that overlaps the input f at a time τ over all time t

$$(f \times g)(t) = \int_{0}^{t} f(\tau) g \times (t-\tau) d\tau$$

Since τ < 0 is meaningless and τ > t is the value of a function in the future we use this formula

For Images we use the following equations

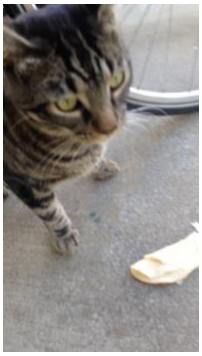
Convolved Feature Matrix

$$(I \times h)(x,y) = \int_{0}^{x} \int_{0}^{y} I(x-i, y-j) \times h(i,j) di\, dj$$

This is the convolved feature matrix where we slide the kernel over the image

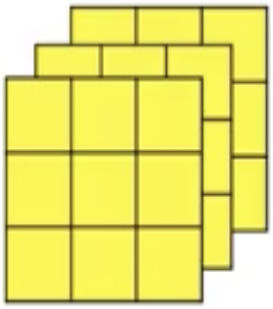
Image Example

Image



Dimensions: 1024 X 682 X 3

Filter



Dimensions: 3 X 3 X 3

Convolved 2-D Output

Dimensions: 1022 X 680

Convolution Dimensions Formula = Image – Filter +1
1024-3+1 = 1022
682-3+1 =680

More in Depth Convolutional Neural Netwroks

1. Activation layer
2. Pooling Layer
3. Fully Connected Layer
4. Convolutional Neural Networks

1. Activation Layer

We need to use a non-linear activation functions because linear funtions don't let the machine learn anything.

Activation Function Form

$$Y = \sum(weight * input) + bias$$

Example: A1 and A2 are two Convolutional Layer Filters. Since we have two Convolutional Layer Filters, we multiply them together into one Convolutional Layer Filter so then we can apply the one filter to Image X.

Activation Layer Example

$$A_1 \times (A_2 \times X)$$
$$(A_1 \times A_2) \times X$$
$$A \times X$$

This is the formula we will use, filter multiplied by the image

Filter A_1 **Filter A_2** **Image X**

$$\begin{pmatrix}1&0&0\\0&1&0\\0&0&1\end{pmatrix} \times \left[\begin{pmatrix}1&1&1\\1&1&1\\1&1&0\end{pmatrix} \times \begin{pmatrix}1&1&1&0\\0&1&0&1\\1&1&0&0\\0&0&1&0\end{pmatrix} \right]$$

Filter A_1 **Filter A_2** **Image X**

$$\left[\begin{pmatrix}1&0&0\\0&1&0\\0&0&1\end{pmatrix} \times \begin{pmatrix}1&1&1\\1&1&1\\1&1&0\end{pmatrix} \right] \times \begin{pmatrix}1&1&1&0\\0&1&0&1\\1&1&0&0\\0&0&1&0\end{pmatrix}$$

Filter A **Image X**

$$\begin{pmatrix}1&1&1\\1&1&1\\1&1&0\end{pmatrix} \times \begin{pmatrix}1&1&1&0\\0&1&0&1\\1&1&0&0\\0&0&1&0\end{pmatrix}$$

Now we calculate the Convolution Matrix using the Dot Product of the two matrixes and get the following:

Image X **Row: 1 to 3** **Column: 1 to 3** **Filter A** $\begin{pmatrix}1&1&1\\0&1&0\\1&1&0\end{pmatrix} \times \begin{pmatrix}1&1&1\\1&1&1\\1&1&0\end{pmatrix} = 6$	**Image X** **Row: 1 to 3** **Column: 2 to 4** **Filter A** $\begin{pmatrix}1&1&0\\1&0&1\\1&0&0\end{pmatrix} \times \begin{pmatrix}1&1&1\\1&1&1\\1&1&0\end{pmatrix} = 5$
Image X **Row: 2 to 4** **Column: 1 to 3** **Filter A** $\begin{pmatrix}0&1&0\\1&1&0\\0&0&1\end{pmatrix} \times \begin{pmatrix}1&1&1\\1&1&1\\1&1&0\end{pmatrix} = 3$	**Image X** **Row: 2 to 4** **Column: 2 to 4** **Filter A** $\begin{pmatrix}1&0&1\\1&0&0\\0&1&0\end{pmatrix} \times \begin{pmatrix}1&1&1\\1&1&1\\1&1&0\end{pmatrix} = 4$

Using the results from above, we can put them together to create the Convolution Matrix below:

Convolution Matrix

$$\begin{pmatrix}6&5\\3&4\end{pmatrix}$$

2. Pooling Layer

Down Sampleing Features

This is the process of reducing the samoingr rate of a signal. This is done to reduce the data rate or the size of the data. We do this to reduce the massive information in the image matrixes.

1. Dimensions of Spatial Extent

The value of n where we can take an n×n feature representation and map to a single value.

2. Stride

How many features the sliding filter skips along the width and height of the image matrix

Example: Using a maxpool with a 2×2 filter and a stride of 2

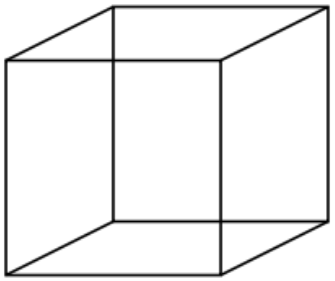
Image

$$\begin{pmatrix} 1 & 1 & 2 & 4 \\ 5 & 6 & 7 & 8 \\ 3 & 2 & 1 & 0 \\ 1 & 2 & 3 & 4 \end{pmatrix}$$

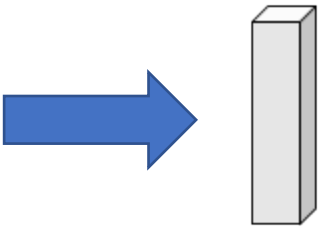
maxpool

$\begin{pmatrix} 6 & 8 \\ 3 & 4 \end{pmatrix}$ Maxpool with a 2×2 filter and a stride of 2

*This maximizes efficiency by not overlapping like in the past filters we were using where there was a lot more overlap.
*Maxpool also reduces the chances of overfitting sine there are less parameters.



$26 \times 26 \times 32$



$13 \times 13 \times 32$

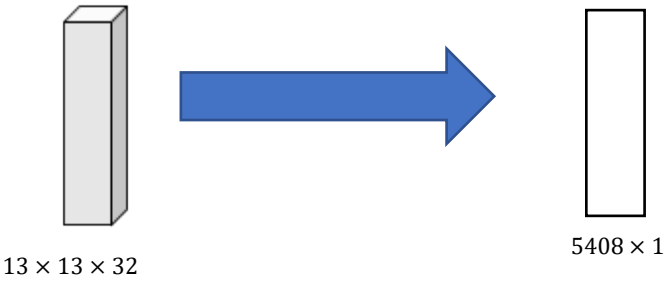
This new matrix decreases the number of parameters by a lot.

3. Fully Connected Layer

We use the Fully Connected Layer to learn non-linear combinations of features such as:

1. Convolutional Layers: Provide meaningful, low dimensional invariant future space.
2. Fully Connected Layers: Learns non-linear functions

Pooling Output: 3-D Feature Map **Fully Connected Input: 1-D Vector**



$13 \times 13 \times 32$ 5408×1

When flattening to 1-D you just multiply the dimensions of the 3-D object to get the 3-D object

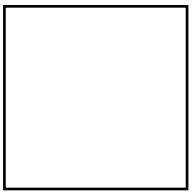
$$13 \times 13 \times 32 = 5408$$

To go from 3-D to 1-D we have to flatten the 3-D object.

4. Convolutional Neural Networks

Step 1

Image



For simplicity we are using these specific dimensions for the image matrix

$28 \times 28 \times 1$

Step 2

Convolution Round 1

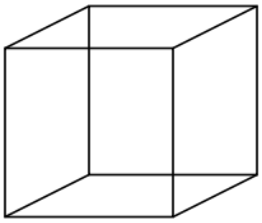
Convolution: 3×3, 32

Stride (1,1)

The Stride is (1,1) because the dimensions are $28 \times 28 \times 1$ and therefore the stride is the value in the third place holder of the dimension.

$$Outwidth = Iwidth - hwidth + 1$$
$$Outwidth = 28 - 3 + 1$$
$$Outwidth = 26$$

$Outdepth = 32$ (The number of filters to be applied)



$26 \times 26 \times 32$

Step 3

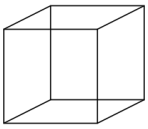
Maxpool Round 1

Maxpool: 2×2

Stride (2,2)

$$Outwidth = \frac{Iwidth - hwidth + 1}{Stride}$$
$$Outwidth = \frac{28 - 2 + 1}{2}$$
$$Outwidth = 13.5$$

$Outwidth = 13$ (Always round down, can't have half of a measurement)



$13 \times 13 \times 32$

Step 4

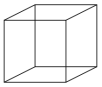
Convolution Round 2

Convolution: 3×3, 64

Stride (1,1)

$$Outwidth = Iwidth - hwidth + 1$$
$$Outwidth = 13 - 3 + 1$$
$$Outwidth = 11$$

$Outdepth = 64$ (The number of filters to be applied)



$11 \times 11 \times 64$

Step 5

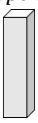
Maxpool Round 2

Maxpool: 2×2

Stride (2,2)

$$Outwidth = \frac{Iwidth - hwidth + 1}{Stride}$$
$$Outwidth = \frac{11 - 2 + 1}{2}$$
$$Outwidth = \frac{10}{2}$$
$$Outwidth = 5$$

$$Outdepth = 64$$



$5 \times 5 \times 64$

Step 6

Flatten

Multiply the dimensions of the final maxpooled output to get this in to a 1-D form and therefore a vector. Doing so we get:

$$5 \times 5 \times 64$$
$$25 \times 64$$
$$1600$$

Vector is 1600×1

1600×1

Step 7

Hidden Layers

Apply the hidden layer of neurons

512 Neurons in the Hidden Layer

Step 8

Apply Dropout

We apply dropout for regularization (regularization is the process of introducing new information in order to prevent overfitting) so there is no overfitting.

512 Neurons in the Hidden Layer

Step 9

Probability

After applying Dropout we get an output using a Softmax layer of 10 neurons. We use the Soft max function because it can calculate the probabilities of a component being a part of a certain class therefore it gives us readable probabilities, and interpretations.

Here we have the probabilities of each component being in its class.

Conclusion

In conclusion, for the example that we just completed above, we can see how for the most part the goal of this is to try and reduce the amount of information in an image to a 1-D point of view to have a sort of compression of information the image stores. We do this using a convolution form where we can sample and take the dot product of the image. The information above is a very quick guide of how Convolutional Neural Networks work in a very simplistic format however the ideas of backpropagation should be investigated along with the mathematical formulas of how SoftMax and RELUC work.