ChatGPT prompt “minimalist landscape painting of a deep underwater scene with a blue tang fish in the bottom right corner”
Recap

Building multi-layer neural networks

- Hidden layers
- What a one-hidden layer network can learn
- What a multi-layer network can learn
- Partially connected networks are useful (e.g., for images!)
- Fully connected networks are not transitionally invariant
- Convolutional filter

Introduction to CNNs
Today’s goal – continue to learn about CNNs

(1) Convolution (contd.) – stride

(2) Learning convolutional filters – connection to partially connected networks

(3) Convolution in Tensorflow – padding and other considerations
What Convolution Does (Visually)

In summary:

\[
\begin{array}{cccc}
2 & 0 & 1 & 3 \\
7 & 1 & 1 & 0 \\
0 & 2 & 5 & 0 \\
0 & 5 & 1 & 4 \\
\end{array}
\otimes
\begin{array}{ccc}
1 & 1 & 1 \\
0 & 0 & 0 \\
-1 & -1 & -1 \\
\end{array}
= \begin{array}{cc}
-4 & -3 \\
3 & -8 \\
\end{array}
\]
What Convolution Does (Mathematically)

\[ V(x, y) = (I \otimes K)(x, y) = \sum_{m} \sum_{n} I(x + m, y + n)K(m, n) \]

- The output at pixel \((x, y)\)
- "Image \(I\) convolved with kernel \(K\)"
- Sum over kernel columns
- Sum over kernel rows
- Multiply kernel value with corresponding image pixel value
What Convolution Does (Mathematically)

![Diagram showing the process of convolution]

**Image**

<table>
<thead>
<tr>
<th>y = 0</th>
<th>x = 0</th>
<th>x = 1</th>
<th>x = 2</th>
<th>x = 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>y = 0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>y = 1</td>
<td>7</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>y = 2</td>
<td>0</td>
<td>2</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>y = 3</td>
<td>0</td>
<td>5</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>

**Filter/Kernel**

<table>
<thead>
<tr>
<th>m = 0</th>
<th>m = 1</th>
<th>m = 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>n = 0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>n = 1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>n = 2</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

**Output**

<table>
<thead>
<tr>
<th>x = 0</th>
<th>x = 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>y = 0</td>
<td>-4</td>
</tr>
<tr>
<td>y = 1</td>
<td>3</td>
</tr>
</tbody>
</table>
What Convolution Does (Mathematically)

\[ V(0, 0) = (I \otimes K)(0, 0) = \sum_{m=0}^{2} \sum_{n=0}^{2} I(0 + m, 0 + n)K(m, n) \]
What Convolution Does (Mathematically)

\[
V(1, 0) = (I \otimes K)(1, 0) = \sum_{m=0}^{2} \sum_{n=0}^{2} I(1 + m, 0 + n)K(m, n)
\]

<table>
<thead>
<tr>
<th>x = 0</th>
<th>x = 1</th>
<th>x = 2</th>
<th>x = 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>y = 0</td>
<td>2</td>
<td>0x1</td>
<td>1x1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(K(0, 0))</td>
<td>(K(1, 0))</td>
</tr>
<tr>
<td>y = 1</td>
<td>7</td>
<td>1x0</td>
<td>1x0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(K(0, 1))</td>
<td>(K(1, 1))</td>
</tr>
<tr>
<td>y = 2</td>
<td>0</td>
<td>2x-1</td>
<td>5x-1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(K(0, 2))</td>
<td>(K(1, 2))</td>
</tr>
<tr>
<td>y = 3</td>
<td>0</td>
<td>5</td>
<td>1</td>
</tr>
</tbody>
</table>

Image:

Output:

\[
0x1 + 1x1 + 3x1 + 0x0 + 1x0 + 0x0 + 2x-1 + 5x-1 + 0x-1
\]
What Convolution Does (Mathematically)

$$V(0, 1) = (I \otimes K)(0, 1) = \sum_{m=0}^{2} \sum_{n=0}^{2} I(0 + m, 1 + n)K(m, n)$$
What Convolution Does (Mathematically)

\[ V(1, 1) = (I \otimes K)(1, 1) = \sum_{m=0}^{2} \sum_{n=0}^{2} I(1 + m, 1 + n)K(m, n) \]
What Convolution Does (In Code)

```plaintext
// Input: Image I, Kernel K, Output V, pixel index x,y
// Assumes K is 3x3
function apply_kernel(I, K, V, x, y)
    for m = 0 to 2:
        for n = 0 to 2:
            V(x,y) += K(m,n) * I(m+x, n+y)
```

**Equation:** \[ V(x,y) = (I \otimes K)(x,y) = \sum_{m} \sum_{n} I(x + m, y + n)K(m,n) \]
Different filters = different effects

https://setosa.io/ev/image-kernels/

Blur

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1/9</td>
<td>1/9</td>
<td>1/9</td>
</tr>
<tr>
<td>1/9</td>
<td>1/9</td>
<td>1/9</td>
</tr>
<tr>
<td>1/9</td>
<td>1/9</td>
<td>1/9</td>
</tr>
</tbody>
</table>

Edge Detection / Outline Kernel

<table>
<thead>
<tr>
<th></th>
<th>-1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>-1</td>
<td>5</td>
<td>-1</td>
</tr>
<tr>
<td>0</td>
<td>-1</td>
<td>0</td>
</tr>
</tbody>
</table>

Shift

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Stride

- We don’t just have to slide the filter by one pixel every time.
- The distance we slide a filter by is called \textit{stride}.
  - All the examples we’ve seen thus far have been stride = 1.
Stride in Action

Stride: 1

Stride: 2

http://deeplearning.net/software/theano_versions/0.9.X/tutorial/conv_arithmetic.html
Why would we want stride > 1?

Stride: 1

Stride: 2

Any connection between input and output size?
Why would we want stride > 1?

Larger stride turns a bigger input into the same size output.
Why would we want stride > 1?

Stride: 1  
Output size: 2x2  
Input size: 4x4

Stride: 2  
Output size: 2x2  
Input size: 5x5

Larger stride turns a bigger input into the same size output

**Corollary:** Larger stride turns the same size input into a *smaller* output

Use this to (controllably) decrease image resolution!
OK but...where’s the *learning*?

Can you guess what do we learn in CNNs? (what are our parameters?)
Key Idea 1: Filters are Learnable

![Image of convolution operation with an image, a filter/kernel, and the resulting output.]
Key Idea 1: Filters are \textit{Learnable}

$\begin{array}{cccc}
2 & 0 & 1 & 3 \\
7 & 1 & 1 & 0 \\
0 & 2 & 5 & 0 \\
0 & 5 & 1 & 4 \\
\end{array}$ $\times$ $\begin{array}{ccc}
k_{1,1} & k_{1,2} & k_{1,3} \\
k_{2,1} & k_{2,2} & k_{2,3} \\
k_{3,1} & k_{3,2} & k_{3,3} \\
\end{array}$ = $\begin{array}{cc}
? & ? \\
? & ? \\
\end{array}$

$k_{i,j}$ are learnable parameters
Key Idea 1: Filters are *Learnable*

Label="Mouse"
Detecting patterns using learned filters

Image courtesy: https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/
Detecting patterns using learned filters

How to detect other patterns?
Key Idea 2: Learn *many* filters

This block of filters is called a *filter bank*.
Key Idea 2: Learn *many* filters

The output is now a multi-channel image.
Key Idea 2: Learn *many* filters

• Why are multiple filters a good idea?
  • Can learn to extract different *features* of the image

You will explore this more in lab!
How is convolution “partially connected?”
Only certain input pixels are “connected” to certain output pixels

```
2 0 1 3
7 1 1 0
0 2 5 0
0 5 1 4
```

```
1 1 1
0 0 0
-1 -1 -1
```

\( \times \)

```
-4 -3
2 -9
```
Only certain input pixels are “connected” to certain output pixels

Colored dots in the input pixels represent which output pixels that input pixel contributes to.

If this were fully connected, every input pixel would have all four output colors.
Convolution in Tensorflow

```
tf.nn.conv2d(input, filter, strides, padding)
```

Input Image (4-D Tensor)
Shape:
```
[batchSz, input_height, input_width, input_channels]
```

Can you guess the shape?

Full documentation here: [https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/conv2d](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/conv2d)
Output Size of a Convolution Layer

The output size of a convolution layer depends on 4 Hyperparameters:

• Number of filters, $N$

$N = 3$
Output Size of a Convolution Layer

The output size of a convolution layer depends on 4 Hyperparameters:

- Number of filters, $N$
- The size of these filters, $F$

Filter size = 3
Convolution in Tensorflow

```
tf.nn.conv2d(input, filter, strides, padding)
```

Kernel (4-D Tensor)

Shape:

```
[f_height, f_width, in_channels, out_channels]
```

Can you guess the shape?

Full documentation here:  
[https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/conv2d](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/conv2d)
Output Size of a Convolution Layer

The output size of a convolution layer depends on 4 Hyperparameters:

- Number of filters, $N$
- The size of these filters, $F$
- The stride, $S$

Stride = 2
Convolution in Tensorflow

```
tf.nn.conv2d(input, filter, strides, padding)
```

List of ints of length 4
Represents the strides along each dimension of the input

```
[batch_stride, stride_along_height, stride_along_width, stride_along_input_channels]
```

Full documentation here: [https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/conv2d](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/conv2d)
Convolution in Tensorflow

\[ \text{tf.nn.conv2d(input, filter, strides, padding)} \]

Full documentation here: [https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/conv2d](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/conv2d)
“Problem” With Convolution

- Output of convolution is always smaller than the input
- Why might we want the output size to be the same?
  - To avoid the filter “eating at the border” of the image when applying multiple conv layers

\[
\begin{array}{ccc}
2 & 0 & 1 \\
0 & 1 & 1 \\
0 & 0 & 2 \\
0 & 1 & 1 \\
\end{array}
\otimes
\begin{array}{ccc}
1 & 1 & 1 \\
0 & 0 & 0 \\
-1 & -1 & -1 \\
\end{array}
= \begin{array}{ccc}
1 & 2 \\
0 & -1 \\
\end{array}
\]
Solution: Padding

Apply the kernel to ‘imaginary’ pixels surrounding the image

```
2 0 3 1 1
1 1 0 0 2
4 3 2 0 1
1 0 5 2 0
0 1 0 3 0
```
Solution: Padding

Apply the kernel to ‘imaginary’ pixels surrounding the image

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>?</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>?</td>
</tr>
<tr>
<td>?</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>?</td>
</tr>
<tr>
<td>?</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>?</td>
</tr>
<tr>
<td>?</td>
<td>1</td>
<td>0</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>?</td>
</tr>
<tr>
<td>?</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>?</td>
</tr>
</tbody>
</table>
What Values to Use For These Pixels?

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
What Values to Use For These Pixels?

Standard practice: fill with zeroes
What Values to Use For These Pixels?

Standard practice: fill with zeroes

• Zero-valued padding pixels just result in some terms in the convolution sum being zero

\[ V(x, y) = (I \otimes K)(x, y) = \sum_m \sum_n I(x + m, y + n)K(m, n) \]

• End result: equivalent to applying a ‘masked’ version of the filter that only covers the valid pixels

This is zero for a padding pixel
2 available options: ‘VALID’ and ‘SAME’:

**Valid**
Filter only slides over “Valid” regions of the data

**Same**
Filter slides over the bounds of the data, ensuring output size is the “Same” as input size (when stride = 1)
VALID Padding in Tensorflow

\[
\text{tf.nn.conv2d}(\text{input}, \text{filter}, \text{strides}, \\
\text{padding}='\text{VALID}')
\]

2 0 1 3
0 1 1 0
0 0 2 0
0 1 1 1
VALID Padding in Tensorflow

tf.nn.conv2d(input, filter, strides, padding='VALID')

```
\begin{array}{cccc}
2 & 0 & 1 & 3 \\
0 & 1 & 1 & 0 \\
0 & 0 & 2 & 0 \\
0 & 1 & 1 & 1 \\
\end{array}
```
VALID Padding in Tensorflow

tf.nn.conv2d(input, filter, strides, padding='VALID')

```plaintext
  2 0 1 3
0 1 1 0
0 0 2 0
0 1 1 1
```
VALID Padding in Tensorflow

```
tf.nn.conv2d(input, filter, strides,
             padding='VALID')
```

```
2 0 1 3
0 1 1 0
0 0 2 0
0 1 1 1
```
We already tried this! (reduced output size)

Stride = 1

"VALID"
SAME Padding in Tensorflow

```python
tf.nn.conv2d(input, filter, strides, padding='SAME')
```

![Diagram of padding examples](image)
SAME Padding in Tensorflow

tf.nn.conv2d(input, filter, strides, padding='SAME')

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>8</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
SAME Padding in Tensorflow

tf.nn.conv2d(input, filter, strides, padding='SAME')

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>8</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
SAME Padding in Tensorflow

tf.nn.conv2d(input, filter, strides, padding='SAME')

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>8</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
SAME padding Example (Try it as HW)

\[
\begin{array}{cccc}
2 & 0 & 3 & 1 \\
1 & 1 & 0 & 0 \\
1 & 0 & 2 & 0 \\
1 & 0 & 1 & 2 \\
\end{array}
\]

\[
\begin{array}{ccc}
1 & 0 & -1 \\
2 & 0 & -2 \\
1 & 0 & -1 \\
\end{array}
\]

\[
\begin{array}{cccc}
-1 & -1 & -1 & 6 \\
-2 & 0 & 1 & 5 \\
-1 & -1 & -1 & 5 \\
0 & -1 & -4 & 4 \\
\end{array}
\]

"Same"  
Stride = 1
Convolution in Tensorflow

```
tf.nn.conv2d(input, filter, strides, padding)
```

The mode of padding to use (String)
Either “Valid” or “Same”
Case-insensitive

Full documentation here: https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/conv2d
Output Size of a Convolution Layer

The output size of a convolution layer depends on 4 Hyperparameters:

- Number of filters, $N$
- The size of these filters, $F$
- The stride, $S$
- The amount of padding, $P$

![Diagram showing output size calculation]

Padding = 2
Output Size of a Convolution Layer

Suppose we know the number of filters, their size, the stride, and padding \((n,f,s,p)\).

Then for a convolution layer with input dimension \(w \times h \times d\), the output dimensions \(w' \times h' \times d'\) are:

\[
w' = \frac{w - f + 2p}{s} + 1
\]

\[
h' = \frac{h - f + 2p}{s} + 1
\]

\[d' = n\]
Output Size for “VALID” Padding

\[ w' = \frac{w - f + 2p}{s} + 1 \]

Let \( w = 4 \)

\[ w' = \frac{4 - 3 + 2 \cdot 0}{1} + 1 \]

\[ = 1 + 1 = 2 \]

num filters \( n = 1 \)
filter size \( f = 3 \)
stride \( s = 1 \)
padding \( p = 0 \)
Output Size for “VALID” Padding

\[ w' = \frac{w - f + 2p}{s} + 1 \]

num filters \( n = 1 \)
filter size \( f = 3 \)
stride \( s = 1 \)
padding \( p = 0 \)
Output Size for “VALID” Padding

\[ w' = \frac{w - f + 2p}{s} + 1 \]

num filters \( n = 1 \),
filter size \( f = 3 \),
stride \( s = 1 \),
padding \( p = 0 \)
Output Size for “VALID” Padding

\[ w' = \frac{w - f + 2p}{s} + 1 \]

num filters \( n = 1 \)
filter size \( f = 3 \)
stride \( s = 1 \)
padding \( p = 0 \)
Output Size for “VALID” Padding

\[ w' = \frac{w - f + 2p}{s} + 1 \]

- num filters \( n = 1 \)
- filter size \( f = 3 \)
- stride \( s = 1 \)
- padding \( p = 0 \)
Output Size for “SAME” Padding

\[ w' = \frac{w - f + 2p}{s} + 1 \]

num filters \( n = 1 \)
filter size \( f = 3 \)
stride \( s = 1 \)
padding \( p = 1^* \)

Let \( w = 4 \)

\[ w' = \frac{4 - 3 + 2 \cdot 1}{1} + 1 \]

\[ = 3 + 1 = 4 \]

*Chosen so output size is the same
Output Size for “SAME” Padding

\[ w' = \frac{w - f + 2p}{s} + 1 \]

- num filters \( n = 1 \)
- filter size \( f = 3 \)
- stride \( s = 1 \)
- padding \( p = 1^* \)

*Chosen so output size is the same
Output Size for “SAME” Padding

\[ w' = \frac{w - f + 2p}{s} + 1 \]

num filters \( n = 1 \)
filter size \( f = 3 \)
stride \( s = 1 \)
padding \( p = 1^* \)

*Chosen so output size is the same
Output Size for “SAME” Padding

\[ w' = \frac{w - f + 2p}{s} + 1 \]

num filters \( n = 1 \)
filter size \( f = 3 \)
stride \( s = 1 \)
padding \( p = 1^* \)

*Chosen so output size is the same
Output Size for “SAME” Padding

\[ w' = \frac{w - f + 2p}{s} + 1 \]

num filters \( n = 1 \)
filter size \( f = 3 \)
stride \( s = 1 \)
padding \( p = 1 \)*

*Chosen so output size is the same
Convolution in Tensorflow

```python
tf.nn.conv2d(input, filter, strides, padding)
```

- **Input Image**: (4-D Tensor)
- **Filter/Kernel**: (4-D Tensor)
- **Strides along each dimension**
- **Type of Padding**: (String “Valid” or “Same”)

Full documentation here: [https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/conv2d](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/conv2d)
Application to Real World Data (MNIST)

# Should be of shape (batch_sz, 28, 28, 1) for MNIST
inputs = MNIST_image_batch

# Sets up a 5x5 filter with 1 input channels and 16 output channels
self.filter = tf.Variable(tf.random.normal([5, 5, 1, 16], stddev=0.1))

# Convolves the input batch with our defined filter
conv = tf.nn.conv2d(inputs, self.filter, [1, 2, 2, 1], padding="SAME")
Application to Real World Data (CIFAR)

# Should be of shape (batch_sz, 32, 32, 3) for CIFAR10
inputs = CIFAR_image_batch

# Sets up a 5x5 filter with ? input channels and 16 output channels
self.filter = tf.Variable(tf.random.normal([?,?,?,?], stddev=0.1))

# Convolves the input batch with our defined filter
conv = tf.nn.conv2d(?,?,?,?)
Application to Real World Data (CIFAR)

# Should be of shape (batch_sz, 32, 32, 3) for CIFAR10
inputs = CIFAR_image_batch

# Sets up a 5x5 filter with 3 input channels and 16 output channels
self.filter = tf.Variable(tf.random.normal([5, 5, 3, 16], stddev=0.1))

# Convolves the input batch with our defined filter
conv = tf.nn.conv2d(inputs, self.filter, [1, 2, 2, 1], padding="SAME")
2D Convolution for 3D Image

See also: Understanding Convolution by hand vs TensorFlow
Recap

Filters/Kernels and Stride

Learning filters

CNNs are partially connected networks

Tensorflow conv2d function

Padding

Application to MNIST/CIFAR

Convolution in Tensorflow

Convolution