In [3]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

import cnn_helper
%aimport cnn_helper

nn_ch = cnn_helper.Charts_Helper(save_dir="/tmp", visible=False)
```

Receptive Field

Consider one element in some layer..

Let v denote its value.

The *receptive field* for the element

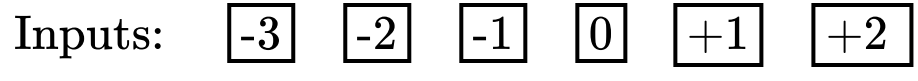
- are the set of *layer 0 locations*
- that affect the value v

We illustrate with inputs with

- one non-feature dimension
- one feature

We will index the locations

- so that the indices are all relative to a center location which we define as 0



Let's consider

- a convolutional layer with $f = 3$
- show the receptive field for the central input location and its neighbors

The right column explains which locations contributed to the receptive field for the central location

Inputs:	-3	-2	-1	0	+1	+2	
Layer 1 field			[-2: 0]	[-1: +1]	[0:+2]	left:	-1

The receptive field for the center location are input locations -1 through $+1$.

This is because

- when we center the kernel of size $f = 3$ at location 0
- locations $[-1 : +1]$ are included in the dot product

We similarly show the receptive field for the two neighbors

- lower and upper boundaries are offset by 1 relative to the central location

Adding a second convolutional layer

Let us add a second convolutional layer with $f = 3$ and look at the receptive field for the central location.

Again: the explanation for the central receptive field is given in the last column.

Inputs:	-3	-2	-1	0	+1	+2		
Layer 1 field			[-2: 0]	[-1: +1]	[0:+2]		left:	-1
Layer 2 field				[-2:+2]			left:	[-2:0

When the filter of the Layer 2 Convolution is centered over location 0 the following elements appear in the dot product

- Layer 1 element at location -1 is included; it's receptive field is $[-2:0]$
- Layer 1 element at location 0 is included; it's receptive field is $[-1:+1]$
- Layer 1 element at location $+1$ is included; it's receptive field is $[0:+2]$

Thus the Layer 2 value at location 0

- has a receptive field $[-2:+2]$

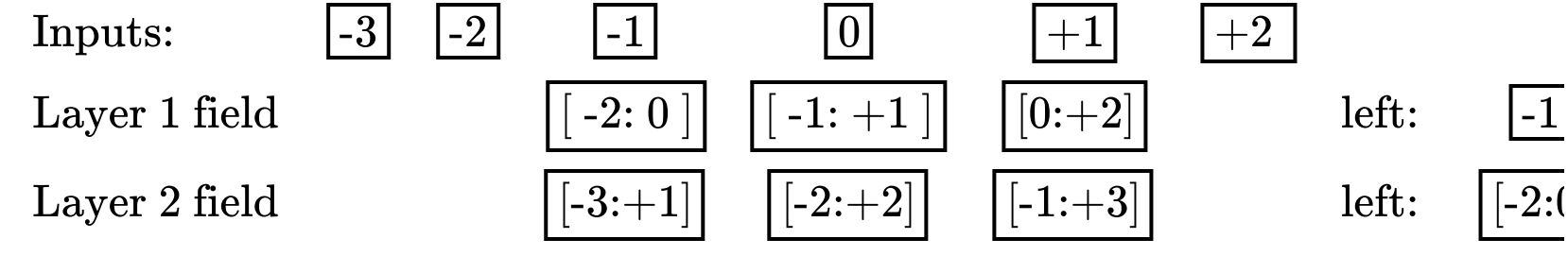
Remember

The receptive field are the **Layer 0** locations affected by the dot product.

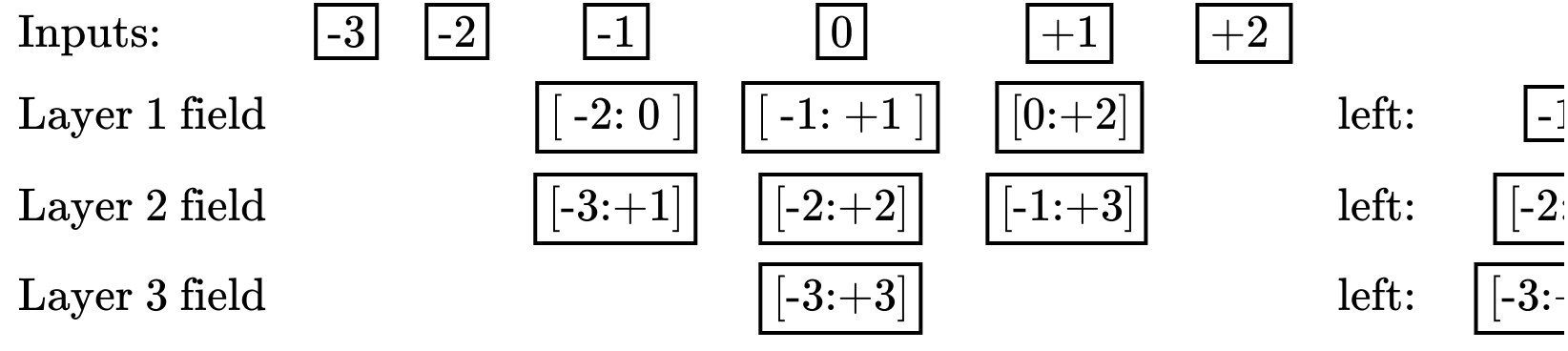
Adding a third Convolutional layer

Before we expand to one more layer:

- Let's expand our diagram to include the receptive field of the left and right neighbor of the center location



Now we can see the Layer 3 receptive field for the central location.



Width of Receptive Field is a function of the depth of the location

The number of locations in the receptive field of a location is called the *width or size* of the receptive field

We can see from our illustration

- that the width of the receptive field for a location
- increases with the depth
    - layer in which the location appears

Layer	Receptive field
1	3
2	5
3	7
⋮	⋮

All locations in the same layer have the same width of receptive field

- assuming "same" padding

Thus

- any location at a depth after 3 Convolutional layers with $f = 3$
- has receptive field width 7

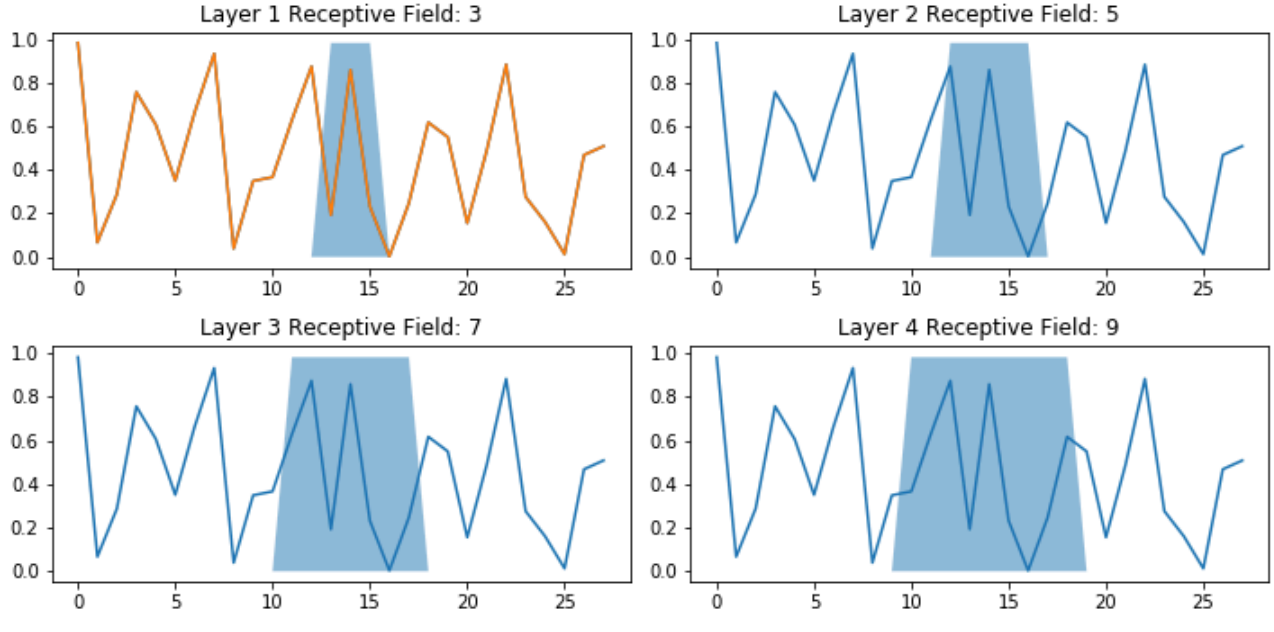
Similarly

- any location at a depth after **a single** convolutional layer with $f = 7$
- also have receptive field width 7.

We can visualize how multiple Convolutional layers would be used

- on timeseries data

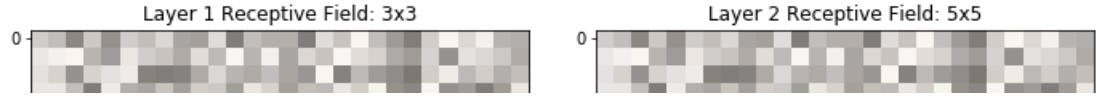
Receptive field varies by depth: N=1 non-spatial dimension



And we can visualize how multiple Convolutional layers would be used

- on two dimensional inputs

Receptive field varies by depth: N=2 non-spatial dimension


Layer 1 Receptive Field: 3x3
0
Layer 2 Receptive Field: 5x5
0

Trading filter size for depth

This means

- that a shallow (one Convolutional layer with $f = 7$) Neural Network
- computes a function over the same input locations
- as a deeper (3 Convolutional layers) with a smaller filter ($f = 3$).

As a result, we rarely see very big values for f

- it is preferred to use a smaller f with more layers

We give the mathematical reason for this below.

Let's compare the number of weights

- in a NN with a single Convolutional layer with larger filter $f = 5$
- to a NN with two Convolutional layer with small filter $f = 3$

The final Convolutional layer of each network has receptive field with of 5.

We illustrate with $N = 2$ non-spatial dimensions.

- the difference in number of weights of the two networks
- increases with N

The number of weights of a Convolutional layer l

- are the sizes of the filters
- which have length $f_{(l)}$ for each non-spatial dimension
- $n_{(l-1)}$ features (matches the number of features of it's input: layer $(l-1)$ output)
- there are $n_{(l)}$ output features (with an associated filter for each)

In terms of number of weights:

- The one layer network uses

$$
\begin{aligned}
||\mathbf{W}|| &= n_{(l)} * (n_{(l-1)} * f'(l) * f'(l)) \\
&= 25 * n_{(l)} * n_{(l-1)} \qquad \text{when } f'(l) = 5
\end{aligned}
$$

- The two layer network uses $$ \begin{array}[lll]\ || \W*llp* || & = & *n*\llp *(n_{(\ll-1)}* f*llp* * *f*\llp) \ || \W*{(\ll+1)}* || & = & *n*{(\ll+1)} *(n_*llp* f*{(\ll+1)}* * *f*{(\ll+1)}) \ || \W || & = & || \W*llp* || *+* || *W*{(\ll+1)} || \

```
   & = & (9 * n_\llp * n_{(\ll-1)}) + 9 *( n_\llp * n_{(\ll+1)}) ) & \te
xt{ when } f_\llp = f_{(\ll+1)} = 3 \\
```

\end{array} $$

The two layer network uses *fewer* weights when

$$9 * (n_{(l)} * n_{(l+1)})) < (25 - 9) * n_{(l)} * n_{(l-1)}$$

This will be the case when the number of feature maps in all layers is roughly the same.

- The advantage of the smaller network increases as $f'_{(l)} - f_l$ increases
  - For example: $f'_{(l)} = 7$
  - Versus 3 Convolutional Layers

Thus

- for less space (number of weights)
- we can achieve an identical receptive field
- with the smaller filter
- and more layers

In [5]:
```
print("Done")
```

```
Done
```