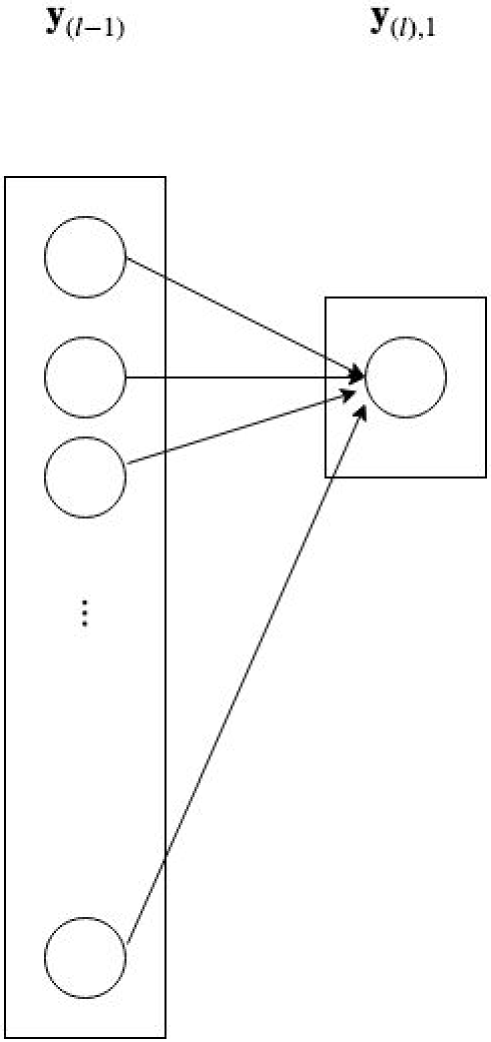
Convolutional Neural Networks

A Fully Connected/Dense Layer with a single unit producing a single feature at layer l computes

$$\mathbf{y}_{(l),1} = a_{(l)}(\mathbf{y}_{(l-1)} \cdot \mathbf{W}_{(l),1})$$

Fully connected, single feature



That is:

- It recognizes one new synthetic feature
- In the entirety ("fully" connected) of $\mathbf{y}_{(l-1)}$
- Using pattern $\mathbf{W}_{(l),1}$ (same size as $\mathbf{y}_{(l-1)}$)
- To reduce $\mathbf{y}_{(l-1)}$ to a single feature.

The pattern being matched spans the entirety of the input

- Might it be useful to recognize a smaller feature that spanned only *part* of the input ?
- What if this smaller feature could occur *anywhere* in the input rather than at a fixed location ?

For example

- A "spike" in a timeseries
- The eye in a face

A pattern whose length was that of the entire input could recognize the smaller feature only in a *specific* place

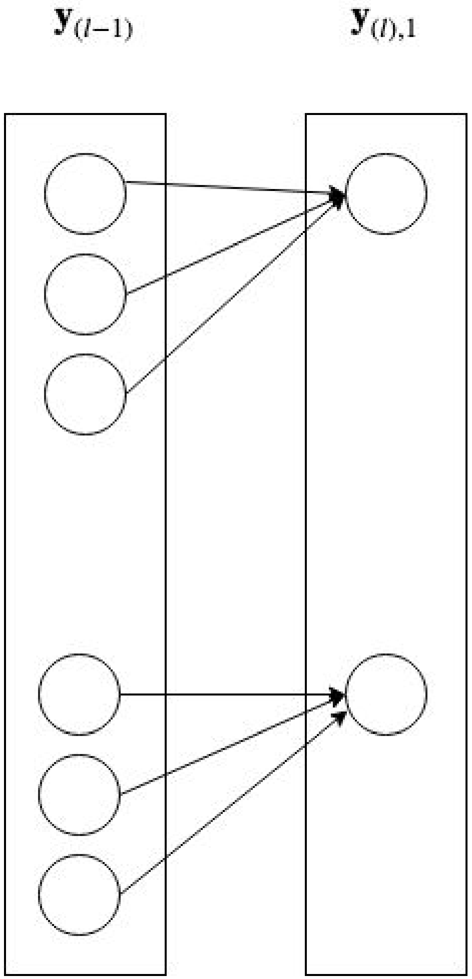
This motivates some of the key ideas behind a Convolutional Layer.

- Recognize smaller features within the whole
- Using small patterns
- That are "slid" over the entire input
- Localizing the specific part of the input containing the smaller feature

Here is the connectivity diagram of a Convolutional Layer producing a **single** feature at at layer l

- Using a pattern of length 3
- Eventually we will show how to produce *multiple* featres
- Hence the subscript "1" in $\mathbf{y}_{(l),1}$ to denote the first output feature
- The output $\mathbf{y}_{(l),1}$ is called a *feature map* as it attempts to match a feature at each input location

Convolutional layer, single feature



The important differences of a Convolutional Layer from a Fully Connected Layer:

- Produces a new *single* feature *for each location* in $\mathbf{y}_{(l-1)}$
- $\mathbf{y}_{(l),1}$ is thus a *vector* (first feature map) of the same length as $\mathbf{y}_{(l-1)}$
- $y_{(l)}$ is a vector of $n_{(l)}$ feature maps, one feature map per output feature
- The output feature at location j is **not** fully connected to $\mathbf{y}_{(l-1)}$
    - Only a subsequence of $\mathbf{y}_{(l-1)}$

The lack of full connectivity is significant.

In a Fully Connected network the relationship between

- Feature j and eatures $(j-1), (j+1)$
- Is no more significant than the relationship between feature j and feature $k \gg j$

That is: spatial locality does not matter.

To see the lack of relationship:

Let perm be a random ordering of the integers in the range $[1 \ldots n]$.

Then

- $\mathbf{x}[\text{perm}]$ is a permutation of input $\mathbf{x}$
- $\Theta[\text{perm}]$ is the corresponding permutation of parameters Θ.

$$\Theta^T \cdot \mathbf{x} = \Theta[\text{perm}]^T \cdot \mathbf{x}[\text{perm}]]$$

But for certain types of inputs (e.g. images) it is easy to imagine that spatial locality is important.

By using a small pattern (and restricting connectivity), we emphasize the importance of neighboring features over far way features.

Mathematically, the One Dimensional Convolutional Layer (Conv1d) we have shown computes $\mathbf{y}_{(l)}$

$$\mathbf{y}_{(l),1} = \begin{pmatrix} a_{(l)} \left(\; N(\mathbf{y}_{(l-1)}, \mathbf{W}_{(l),1}, 1) \cdot \mathbf{W}_{(l),1} \; \right) \\ a_{(l)} \left(\; N(\mathbf{y}_{(l-1)}, \mathbf{W}_{(l),1}, 2) \cdot \mathbf{W}_{(l),1} \; \right) \\ \vdots \\ a_{(l)} \left(\; N(\mathbf{y}_{(l-1)}, \mathbf{W}_{(l),1}, n_{(l-1)} \cdot \mathbf{W}_{(l),1} \; \right) \end{pmatrix}$$

where $N(\; \mathbf{y}_{(l-1)}, \mathbf{W}_{(l),1}, j \;)$

- selects a subsequence of $\mathbf{y}_{(l-1)}$ centered at $\mathbf{y}_{(l-1),j}$

Note that

- The *same* weight matrix $\mathbf{W}_{(l),1}$ is used for the first feature at *all*locations j
- The size of $\mathbf{W}_{(l),1}$ is the same as the size of the subsequence $N(\; \mathbf{y}_{(l-1)}, \mathbf{W}_{(l),1}, j)$
    - Since dot product is element-wise multiplication

So $\mathbf{W}_{(l),1}$

- Is a smaller pattern
- That is applied to *each* location j in $\mathbf{y}_{(l-1)}$
- $\mathbf{y}_{(l),1,j}$ recognizes the match/non-match of the smaller first feature at $\mathbf{y}_{(l-1),j}$

$\mathbf{W}_{(l),1}$ is called a convolutional *filter* or *kernel*

- We will often denote it $\mathbf{k}_{(l),1}$
- But it is just a part of the weights $\mathbf{W}$ of the multi-layer NN.
- We use $f_{(l)}$ to denote the size of the smaller pattern called the *filter size*

Note

The default activation $a_{(l)}$ in Keras is "linear"

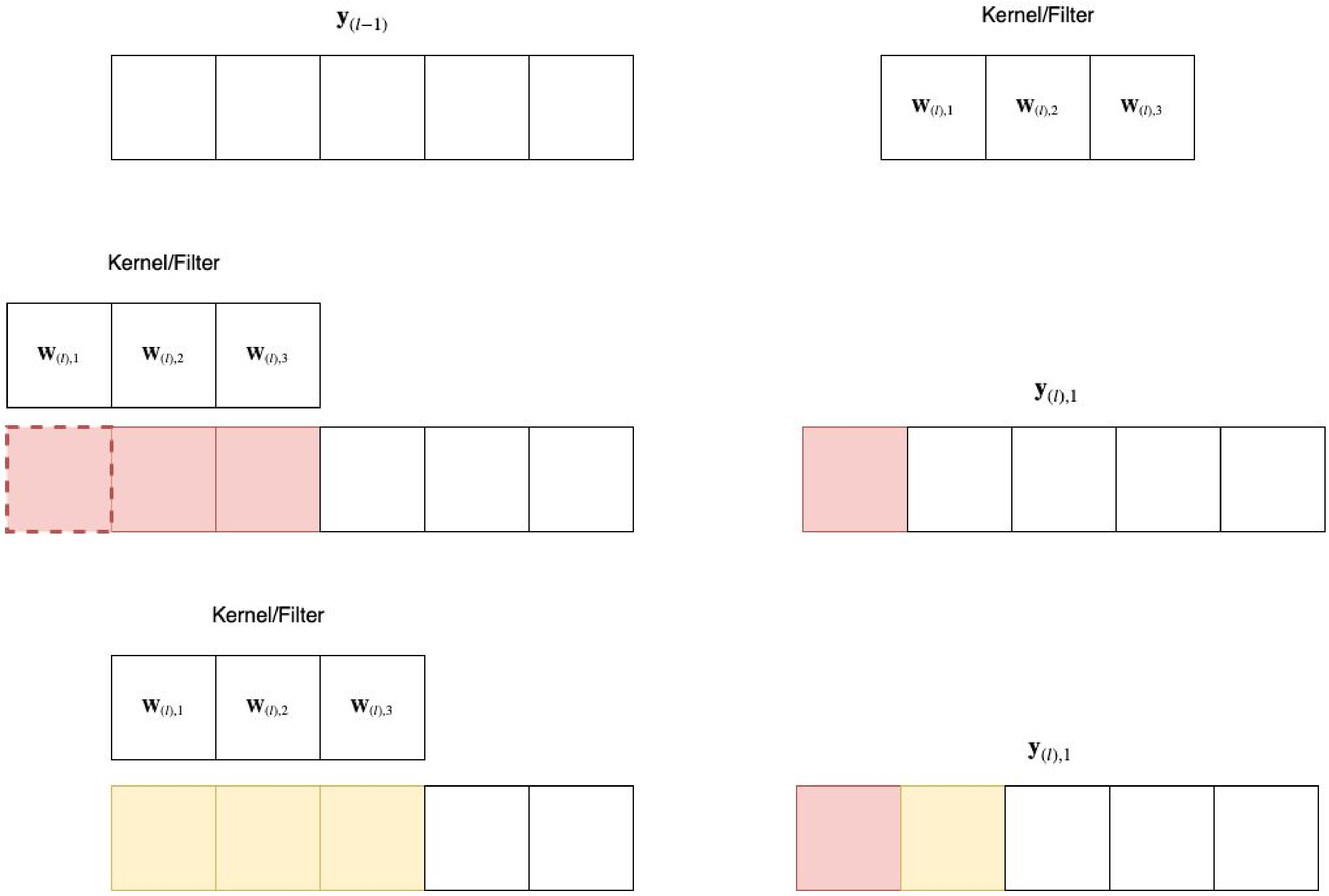
- That is: it returns the dot product input unchanged
- Always know what is the default activation for a layer; better yet: always specify !

A *Convolution* is often depicted as

- A filter/kernel
- That is slid over each location in the input
- Producing a corresponding output for that location

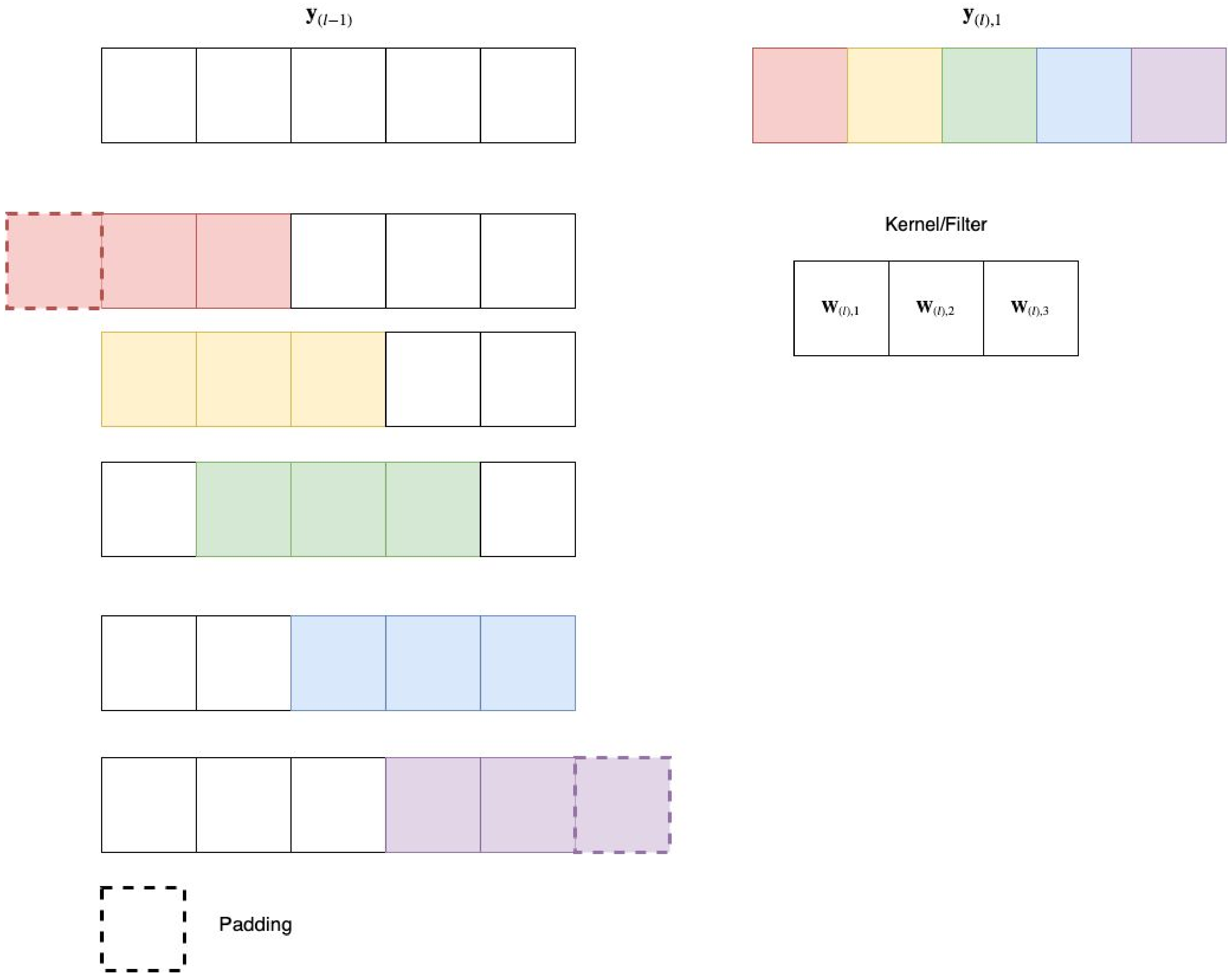
Here's a picture with a kernel of size $f_{(l)} = 3$

Conv 1D, single feature: sliding the filter



After sliding the Kernel over the whole $\mathbf{y}_{(l-1)}$ we get:

Conv 1D, single feature



$\mathbf{y}_{(l-1)}$
$\mathbf{y}_{(l),1}$
Kernel/Filter
$\mathbf{w}_{(l),1}$
$\mathbf{w}_{(l),2}$
$\mathbf{w}_{(l),3}$
Padding

Element j of output $\mathbf{y}_{(l).1}$ (i.e., $\mathbf{y}_{(l),1,j}$)

- Is colored (e.g., $j = 1$ is colored Red)
- Is computed by applying the *same* $\mathbf{W}_{(l),1}$ to
  - The $f_{(l)}$ elements of $\mathbf{y}_{(l-1)}$, centered at $\mathbf{y}_{(l-1),j}$
  - Which have the same color as the output

Note however that, at the "ends" of $\mathbf{y}_{(l-1)}$ the kernel may extend beyond the input vector.

In that case $\mathbf{y}_{(l-1)}$ may be extended with *padding* (elements with 0 value typically)

In [ ]:

```
print("Done")
```