Notation for the CNN layer

Layer l in a Sequential NN transforms transforms input $\mathbf{y}_{(l-1)}$ to output $\mathbf{y}_{(l)}$

- $\mathbf{y}_{(l)}$ is called a *feature map*, for all layers l
    - for each location in $\mathbf{y}_{(l-1)}$
    - it measures the intensity of the pattern match when the pattern is centered at that location

So we write the input as $\mathbf{y}_{(l-1)}$ rather than the $\mathbf{x}$ we had used previously.

The size of all quantities in the convolution can vary by layer

- so we add a parenthesized subscript to indicate the layer

We write

- the kernel size as $f_{(l)}$ (can vary by layer) rather than the f used previously
- the collection of kernels for layer l as $\mathbf{W}_{(l)}$

In general a layer l output $\mathbf{y}_{(l)}$ will have

- $N_{(l)} > 0$ non-feature dimensions
    - non-feature dimension i has length (number of indices) $d_{(l),i}$ indices
        - for dimensions $0 \leq i < N_{(l)}$
    - the set of indexes in dimension i is written as D_i
        - usually equal to $0, \ldots, d_{(l),i}$
- one feature dimension

A CNN Layer l

- preserves the non-feature dimensions (when same padding is used)

$$
\begin{aligned}
N_{(l-1)} &= N_{(l)} \\
d_{(l-1),i} &= d_{(l),i} \quad 0 \le i < N_{(l-1)}
\end{aligned}
$$

- changes the length of the feature dimension
  - from $n_{(l-1)}$ to $n_{(l)}$

Thus the shape of the input $\mathbf{y}_{(l-1)}$ and $\mathbf{y}_{(l)}$ may only differ in the length of the feature dimension

- provided padding is used
  - in the absence of padding: $\lfloor \frac{f_{(l)}}{2} \rfloor$ locations are lost at each boundary

Thus the CNN layer l

$$
\begin{aligned}
||\mathbf{y}_{(l-1)}|| &= (d_{(l-1),0} \times d_{(l-1),1} \times \ldots d_{(l-1),N_{(l-1)}}, \quad \mathbf{n_{(l-1)}}) \\
||\mathbf{y}_{(l)}|| &= (d_{(l-1),0} \times d_{(l-1),1} \times \ldots d_{(l-1),N_{(l-1)}}, \quad \mathbf{n_{(l)}})
\end{aligned}
$$

We write

$$\mathbf{y}_{(l),\mathbf{i},j}$$

to denote feature j of layer l at non-feature dimension location $\mathbf{i}$

Channel Last/First

We have adopted the convention of using the final dimension as the feature dimension.

- This is called *channel last* notation.

Alternatively: one could adopt a convention of the first channel being the feature dimension.

- This is called *channel first* notation.

When using a programming API: make sure you know which notation is the default

- Channel last is the default for TensorFlow, but other toolkits may use channel first.

Kernel, Filter

There is one pattern per output feature.

A pattern is also called a *kernel*.

The kernels of layer l are just the weights of the layer.

The vector $\mathbf{W}_{(l),1}$ above

So kernel j ($\mathbf{k}_j$) is just an element $\mathbf{W}_{(l),j}$ of the weights of layer l.

There is one kernel per output feature, so $n_{(l)}$ kernels

- $\mathbf{k}_{(l),1}, \ldots, \mathbf{k}_{(l),n_{(l)}}$

The length of the feature dimension of a kernel matches it's input, i.e., $n_{(l-1)}$

The weight vector $\mathbf{W}_{(l)}$ therefore has multiple dimensions. Our convention for each dimension is

- $\mathbf{W}_{(l),j',\ldots,j}$
    - layer l
    - output feature j
    - location: $\ldots$
        - an index into the arrangement
        - is length N (number of non-feature dimensions) so use $\ldots$ as a place-holder for N integers
    - input feature j'

Padding

Convolution centers the pattern at each location of the non-feature dimensions of the input.

But what happens when we try to center a patter over the first/last location ?

- the pattern may extend beyond the boundaries of the input

In such a case, we can choose to *pad* the input

- create a special padding input at the locations of the input beyond the original boundary

There are various options for how much to pad

- "same" padding means: add enough padding so that input and output non-feature dimensions are identical

Activation of a CNN layer

Just like the Fully Connected layer, a CNN layer is usually paired with an activation.

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 !

In [5]:

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

```
Done
```