# Interpretation by Reversing the computation

Our initial exploration of Interpretability emphasized some pretty simple methods.

We continue our quest utilizing slightly more advanced ideas.

The general flavor of these ideas is as follows:

- If we can map an individual feature (at a single location (of non-feature dimensions) of feature map $k$ of layer $l$)
- Back to the region of *input* $\mathbf{y}_{(0)}$ that affect it
- Then perhaps we can interpret the feature map $k$ of layer $l$: $\mathbf{y}_{(l),k}$
- Via how a change in the input affects the feature map

That is, we find the sensitivity of the feature map to a change in the input.

# Receptive Field: From Feature Map to Input

Mapping an element of layer $l$ back to regions of layer $0$ requires the concept of *receptive field* that was introduced in [the module on CNN (CNN_Space_and_Time.ipynb#Receptive-field)](CNN_Space_and_Time.ipynb#Receptive-field).

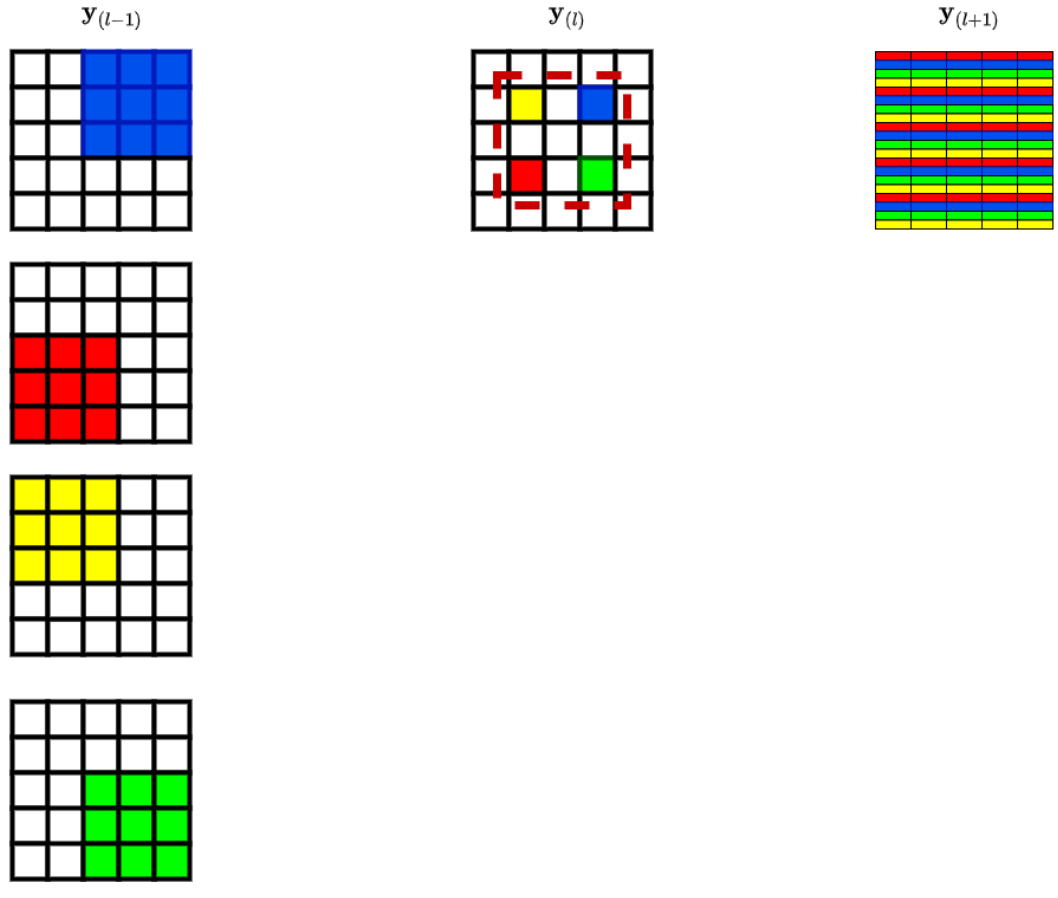
Let's review.

Since a Convolutional layer $l$

- Preserves the spatial dimension of its input (layer $(l-1)$ output (assuming full padding)
- We can relate a single feature at a particular spatial location of a feature map
- To the spatial locations of layer $0$, the input, that affect the layer $l$ feature

We can determine spatial locations of the layer $0$ features influencing this single layer $l$ location by working backwards from layer $l$.

# Conv 2D Receptive field: 2 layers



**Aside: Notes on the diagram**

**The column under layer $(l-1)$ depicts**

- **A *single* feature map at different times (i.e., when the kernel is centered at different layer $l$ spatial locations)**
- **Not different layer $(l-1)$ feature maps!**

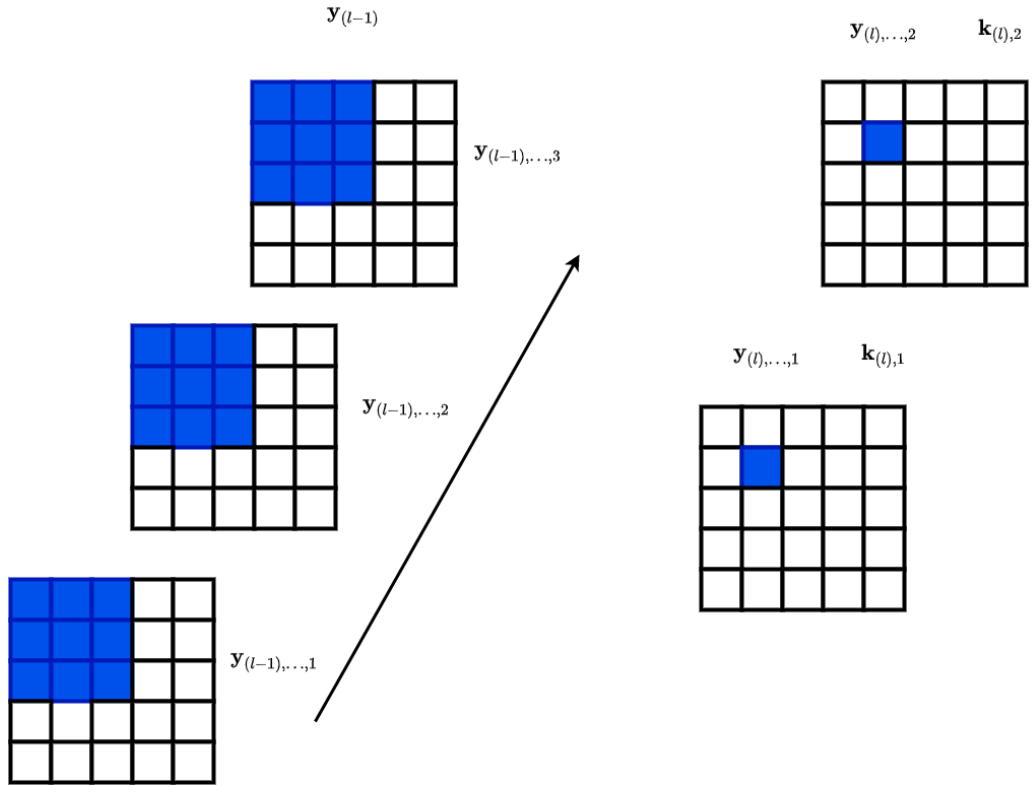
**We also omit feature map/channel subscripts (i.e., writing $\mathbf{y}_{(l)}$ rather than $\mathbf{y}_{(l),\ldots,k}$) as they are not necessary for our purpose**

- **As can be seen by reviewing the mechanics of convolution**

**This is because of the mechanics of the convolutional dot product**

- **Each feature map $k$ at layer $l$**
- **Is a function of *all* the feature maps at layer $(l-1)$**
- **So all feature maps at layer $l$ depend on the same spatial locations of layer $(l-1)$**
- **And these spatial locations are identical across all feature maps/channels of layer $(l-1)$**

**Convolution: preserve spatial dimension, change channel dimension**



Using a kernel with spatial dimension $(3 \times 3)$ for the Convolution of each layer

- The spatial locations in layer $l$
- Are color coded to match the spatial locations in layer $(l-1)$
- That affect it

So the yellow location in layer $l$ is a function of the yellow locations in layer $(l-1)$

**Moving forward one layer: the central location in layer $(l+1)$**

- **Is a function of the spatial locations in layer $l$ that are encircled by the dashed square**
- **Which in turn are a function of a larger number of layer $(l-1)$ locations**

**In general**

- **The number of layer $(l-1)$ spatial locations**
- **That affect a given spatial location in layer $l' \geq l$**
- **Grows as $l'$ increases**

**We can continue this process backwards from layer $l$ to layer $0$**

- **Finally determining the set of input features (region of the input)**
- **Affecting a single spatial location at layer $l$**

**This region of layer $0$ spatial locations**

- **Is called the receptive field of the layer $l$ spatial location**
- **They are what this single layer $l$ spatial location "sees" (i.e., depend on)**

- Let $\mathbf{idx}$ denote the spatial indices of a single location
  - Length of $\mathbf{idx}$ depends on shape of data: one-dimensional, two-dimensional
- Let

  $$\mathbf{y}_{(l),\mathbf{idx},k}$$

  denote the value of the $k^{th}$ feature of layer $l$ at spatial location $\mathbf{idx}$
- In particular, we can refer to input features as

  $$\mathbf{y}_{(0),\mathbf{idx},k}$$

The receptive field $\mathcal{R}_{(l),\mathrm{idx}}$ of spatial location $\mathbf{idx}$ of layer $l$ is

$$\mathcal{R}_{(l),\mathrm{idx}} = \left\{ \mathbf{idx' \text{ at layer } 0} \mid \mathbf{y}_{(l),\mathrm{idx},k} \mathbf{\text{ depends on }} \mathbf{y}_{(0),\mathrm{idx'},k'} \right\}$$

for some

$$\begin{aligned} 1 \leq \quad k \quad &\leq n_{(l)} \\ 1 \leq \quad k' \quad &\leq n_{(0)} \end{aligned}$$

where

$\mathbf{y}_{(l),\mathrm{idx},k}$ $\mathbf{\text{ is the feature at spatial location idx of feature map } } k \mathbf{\text{ of la}}$

(Note that $k$, $k'$ are really not necessary since the spatial locations are shared across all channels during convolution)

# Reversing a Convolution

**Suppose we pick a location in the feature map for feature $k$ of layer $l$.**

**We want to know how this location's activation**

- **was affected by each location in the input layer $0$**

**The Receptive Field is the set of layer $0$ locations that can affect the layer $l$ location.**

**In order to trace backwards, layer by layer**

- **we need to map the output of a layer**
- **to each of the layer's inputs that affect it**

**Rather than having information flowing forward (input to output)**

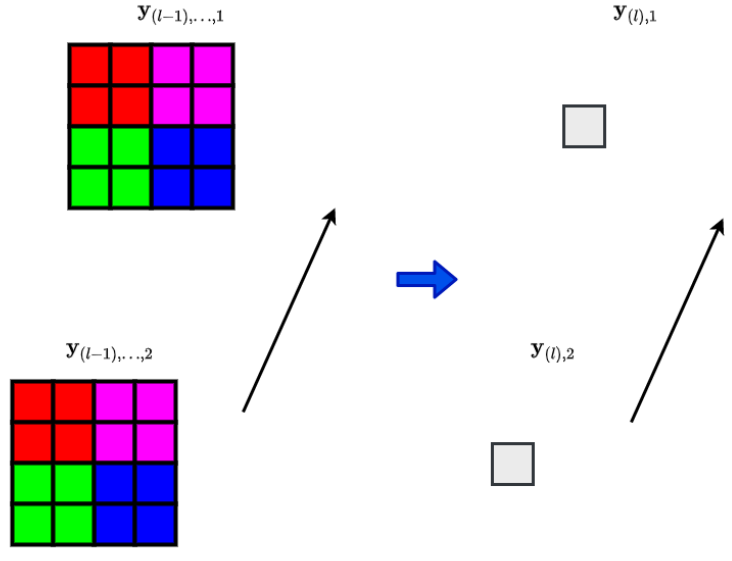
- **this tracing is "reversing" the flow (from output to input)**

**Reversing some layer types are more complicated than others.**

**Max Pooling presents a challenge**

**Recall the operation**

**Conv 2D: Global Pooling (Max/Average)**



**It selects the value of one location from the non-feature dimensions**

- **the one with maximum value**
- **but how do we "trace back" the maximum value to its originating locations ?**

**Solution: the forward computation records the location in "switches" that are used for the inversion.**

**This "tracing backwards" to reverse the flow of computation**

- **is essentially what is happening during the Backward Pass of Back propagation (Training_Neural_Network_Backprop.ipynb)**
- **and the recording of "switches" is similar to**
    - **saving values in the Forward pass of a layer**
    - **that are needed to compute the derivatives in the Backward pass**

**Naive back propagation does not always give the best results.**

**Zeiler and Fergus (and similar related papers) modify Back propagation**

- **In an attempt to get better intuition as to which input features most affect a layer $l$ feature**
- **For example: ignore the *sign* of the derivatives as they flow backwards**
    - **Look for strong positive or negative influences, not caring which**

**This is called *Guided Back propagation*.**

# Conclusion

**We explored the idea of "inverting" the Convolution process**

- **Instead of going from input (layer $0$) to layer $l$**
- **We proceeds backward from a single location in a single feature map of layer $l$**
- **In an attempt to interpret the feature that the layer $l$ feature map is recognizing**

**By mapping back to input layer $0$**

- **We avoid the difficulty that arises when trying to interpret layer $l$'s features as combinations of layer $(l-1)$'s synthetic features.**

**Detailed experiments by Zeiler and Fergus**

- **Support the hypothesis that**
- **Deeper layers recognize features of increasing complexity**

In [4]:
```
print("Done")
```

```
Done
```