Interpretation by Inverting

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 spatial location of feature map k of layer l)
- Back to the region of *input* features that affect it
- Then perhaps we can interpret the feature map k of layer l: $\mathbf{y}_{(l),k}$

We call these methods "inversion" as we map outputs (layer l representations $\mathbf{y}_{(l)}$) back to inputs ($\mathbf{x}$).

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.

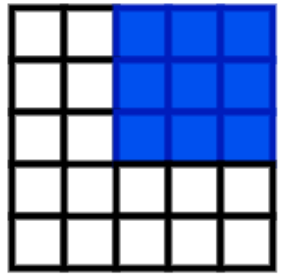
Since a Convolutional layer l

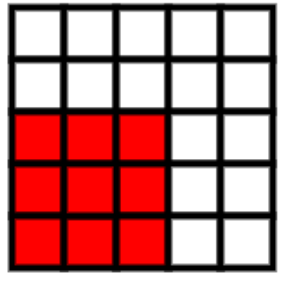
- Preserves the spatial dimension of its input (layer $(l-1)$ output (assuming full padding)
- We can relate a single feature at a particular spatial location of a feature map
- To the spatial locations of layer 0, the input, that affect the layer l feature

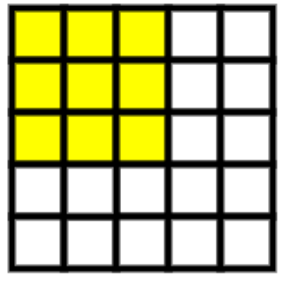
We can determine spatial locations of the layer 0 features influencing this single layer l location by working backwards from layer l.

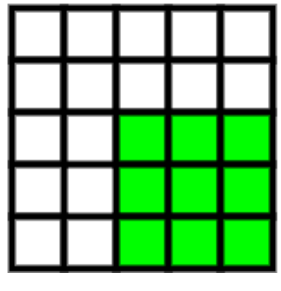
Conv 2D Receptive field: 2 layers

$\mathbf{y}_{(l-1)}$

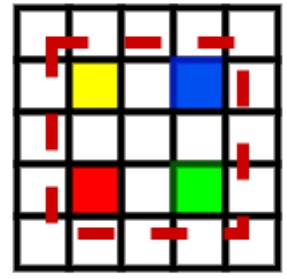




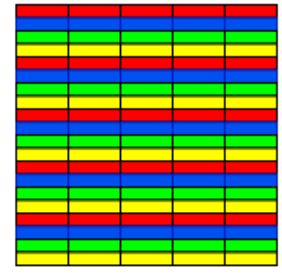




$\mathbf{y}_{(l)}$



$\mathbf{y}_{(l+1)}$



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!**

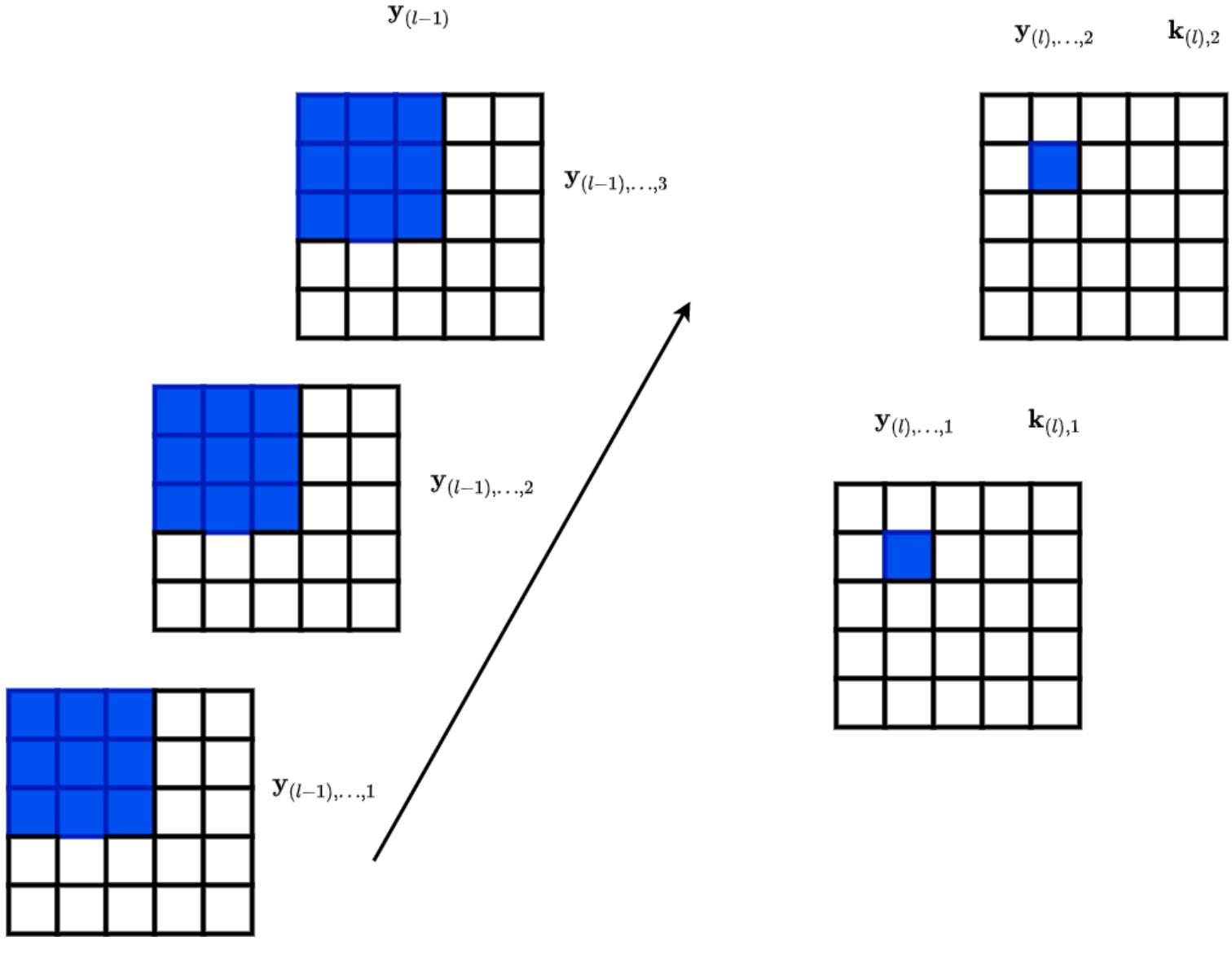
We also omit feature map/channel subscripts (i.e., writing $\mathbf{y}_{(l)}$ rather than $\mathbf{y}_{(l),\ldots,k}$) as they are not necessary for our purpose

- **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 (3×3) 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 $\mathcal{R}_{(l),\text{idx}}$ of spatial location idx of layer l is

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

for some

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

where

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

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

Saliency maps: Relating a feature to the input

The receptive field of a single location at layer l

- **Defines *which* layer 0 spatial locations affect the layer l location**
- **But it does not measure the *magnitude* of the effect**
- **Which may be different for each feature k' of layer 0 at the same spatial location**

We therefore compute the *sensitivity* of a feature

- At spatial location $\mathbf{idx}$ of feature map k of layer l of example i
- To a change in the feature at spatial location $\mathbf{idx}'$ feature map k' of layer 0

$$s^{(\mathrm{i})}_{(l),\mathbf{idx},k,(0),\mathbf{idx}',k'} = \frac{\partial \mathrm{y}^{(\mathrm{i})}_{(l),\mathbf{idx},k}}{\partial \mathrm{y}^{(\mathrm{i})}_{(0),\mathbf{idx}',k'}}$$

Note that the sensitivity

- Is *conditional* on the value of the input $\mathbf{x}^{(\mathrm{i})}$
- Since the derivative's value depends on the particular input

Saliency maps

Given layer l and feature $1 \leq k \leq n_{(l)}$

- **We can construct an image**
- **With same spatial dimension as layer 0 (the input)**
- **By creating a grid of**

  $$s^{(\mathrm{i})}_{(l),\mathrm{idx},k,(0),\mathrm{idx}',k'}$$

  for each idx' in layer 0
- **Resulting in a "heat map" of how each input pixel affects the single spatial location of feature k in layer l**

Rather than doing this for

- **each spatial location $\mathbf{idx}$ in layer l**
- **each input $\mathbf{x}^{(\mathbf{i})}$ in $\mathbf{X}$**

We will create a handful of Saliency Maps for feature k of layer l

- **Limited to the Maximally Activating Examples for feature map k of layer l**

Recall, Maximally Activating Examples are defined with respect to feature map k of layer l as follows

- For each $\mathbf{x}^{(\mathrm{i})}$: compute the (absolute value) of the maximum (across spatial locations) of the feature map
- The Maximally Activating Examples are the examples with highest maximum values
- That is: the examples to which the feature map reacts most intensely

$\mathbf{MaxAct}_{(l),k} = [i_1, \ldots, i_m]$ is the permutation of example indices, i.e., $[i | 1 \leq i \leq m]$ that sorts $\mathbf{max}^{(\mathrm{i})}_{(l),k}$

- where $\mathbf{max}^{(\mathrm{i})}_{(l),k}$ is the largest expression of the pattern anywhere in the spatial dimension of example i

$$\mathbf{max}^{(\mathrm{i})}_{(l),k} = \max_{\mathrm{idx}} \mathbf{y}^{(\mathrm{i})}_{(l),\mathrm{idx},k}$$

Note that the Saliency Map

- **Has the same number of features/channels as layer 0: $n_{(0)}$**
- **Has spatial dimension limited to the receptive field of layer l**
    - **the input pixels not in the receptive field of layer l will have undefined sensitivity**

For any input $\mathbf{x}^{(i)}$

- **We can view the Saliency Map for one layer l feature k given $\mathbf{x}^{(i)}$**
- **Along with the corresponding *patch* of $\mathbf{x}^{(i)}$ (part of $\mathbf{x}^{(i)}$ within receptive field of layer l)**

Here are visualizations of Saliency Maps and corresponding Patches of 9 images (the ones most activating the feature)

- **One layer 2 feature map**
- **On the spatial location idx with maximum intensity in each image**

Saliency Maps and Corresponding Patches
Single Layer 2 Feature Map
On multiple input images

Layer 2 Feature Map (Row 10, col 3).

Attribution: https://arxiv.org/abs/1311.2901 (https://arxiv.org/abs/1311.2901)

The images are small because the Receptive Field of layer 2 is not that large.

We can hypothesize that this Feature Map is responsible for creating the synthetic feature

"There is an eye in the input"

What is particularly interesting is that, by the time we get deeper into the network

- **More complex "patterns" are being recognized**
- **Perhaps due to the Receptive Field getting larger**

Is the interpretation of the following feature map (with high intensity "hot" colors on lips and cheeks)

"Is face with smile present"

Saliency Maps and Corresponding Patches
Single Layer 5 Feature Map
On 9 Maximally Activating Input images

Layer 5 ? Feature Map (Row 11, col 1).

Attribution: https://arxiv.org/abs/1311.2901 (https://arxiv.org/abs/1311.2901)

Video: interactive interpretation of features

There is a nice video by Yosinski (https://youtu.be/AgkfIQ4IGaM) which examines the behavior of a Neural Network's layers on video images rather than stills.

Computing the Saliency Map: Inverting a Convolution

Let's show how to compute the Saliency Map.

- **We feed $\mathbf{x}^{(\mathrm{i})}$ as input**

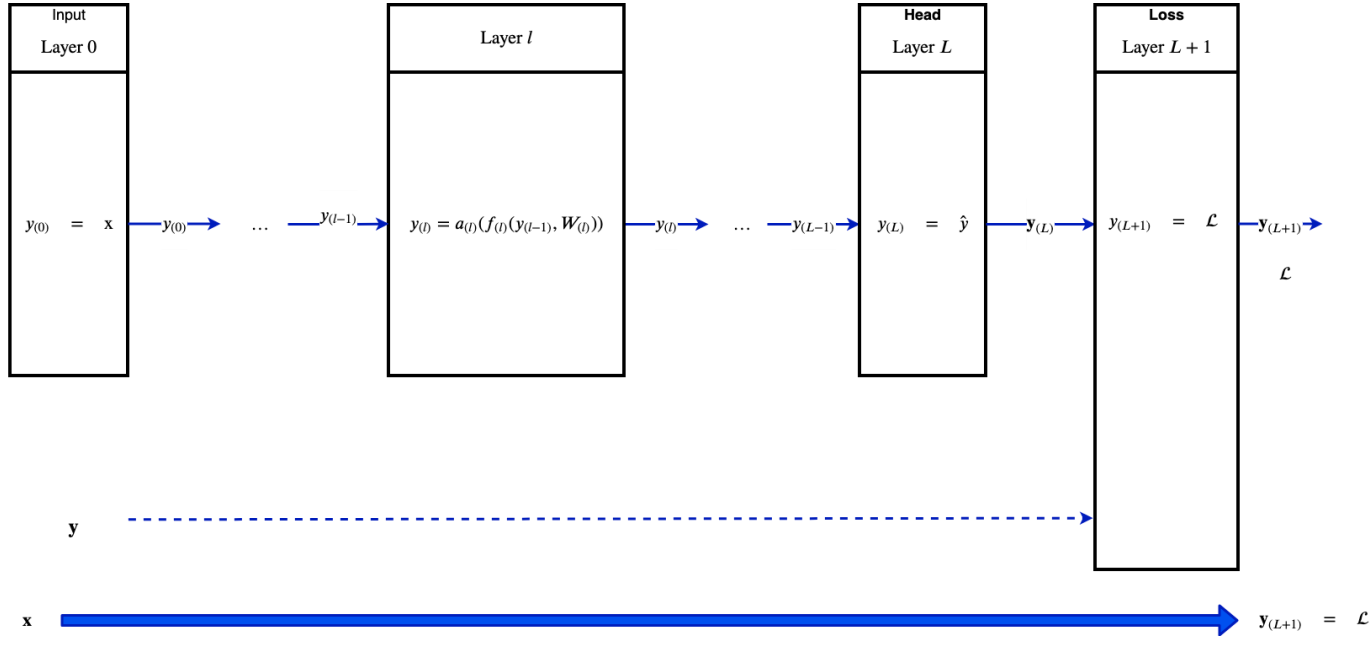
$$\mathbf{y}_{(0)} = \mathbf{x}^{(\mathrm{i})}$$

- **Compute $\mathbf{y}^{(\mathrm{i})}_{(l),\mathrm{idx},k}$ by moving left to right through the layers from 0 to l**
- **Compute the sensitivities by moving right to left, from layer l to layer 0**

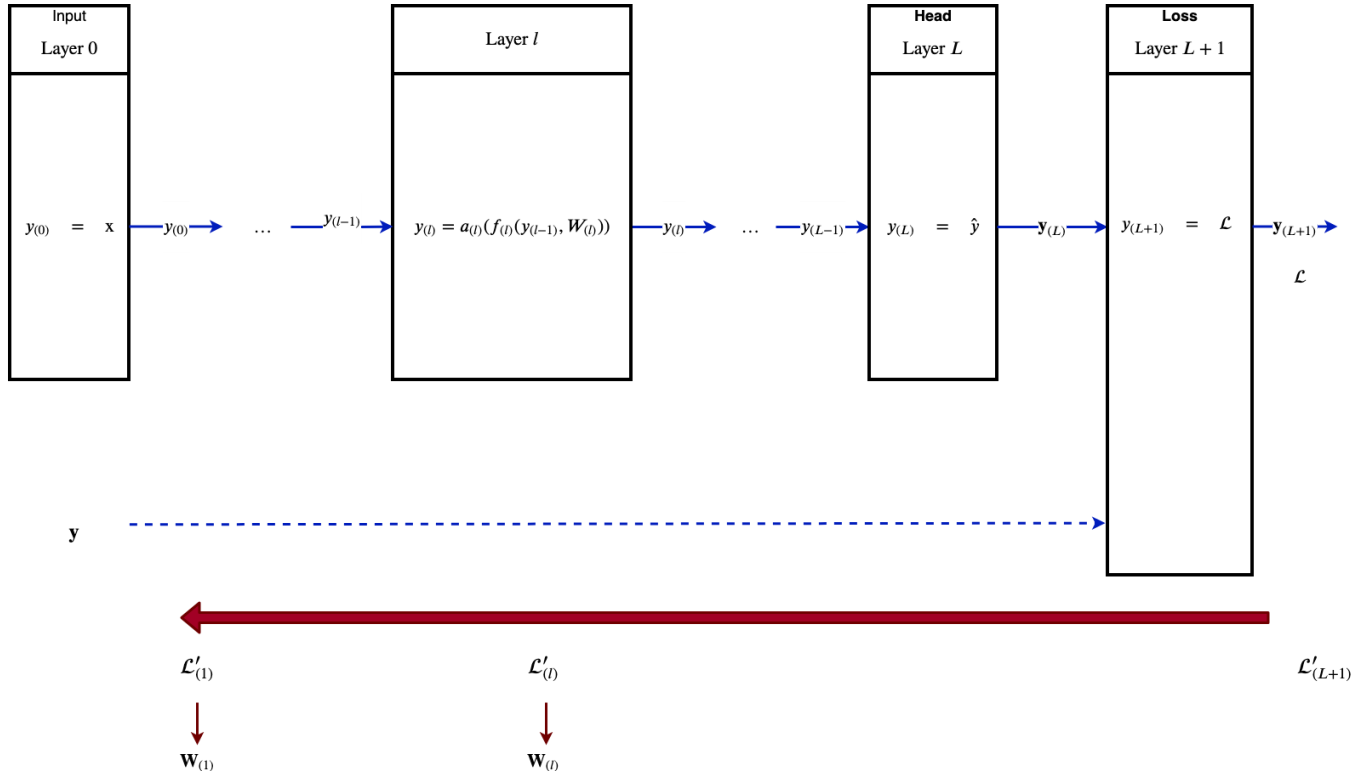
This is very much like the Forward and Backward Passes we saw in the module Back propagation (Training_Neural_Network_Backprop.ipynb)

Recall the pictures:

Forward Pass: Input to Loss



Backward pass: Loss to Weights


Input
Layer 0
$y_{(0)} = \mathbf{x}$
Layer l
$y_{(l)} = a_{(l)}(f_{(l)}(y_{(l-1)}, W_{(l)}))$
Head
Layer L
$y_{(L)} = \hat{y}$
Loss
Layer $L+1$
$y_{(L+1)} = \mathcal{L}$
$y_{(0)}$
$y_{(l-1)}$
$y_{(l)}$
$y_{(L-1)}$
$\mathbf{y}_{(L)}$
$\mathbf{y}_{(L+1)}$
$\mathcal{L}$
$\mathbf{y}$
$\mathcal{L}'_{(1)}$
$\mathcal{L}'_{(l)}$
$\mathcal{L}'_{(L+1)}$
$\mathbf{W}_{(1)}$
$\mathbf{W}_{(l)}$

The main difference is

- **We truncate the network at layer l**
- **Take the derivative of $\mathbf{y}_{(l)}^{(\mathrm{i})}$ (given $\mathbf{x}^{(\mathrm{i})}$) rather than the Loss $\mathcal{L}$**
- **Take derivatives with respect to input features $\mathbf{y}_{(0),\mathrm{idx}',k'}^{(\mathrm{i})}$ rather than weights $\mathbf{W}$**

The Forward Pass

- Mapping $\mathbf{x}^{(\mathbf{i})}$ to $\mathbf{y}_{(l)}^{(\mathbf{i})}$
- Is called *Convolution*

The Backward Pass

- Mapping $\mathbf{y}_{(l)}^{(\mathbf{i})}$ to a Saliency Map (grid of sensitivities)
- Is called *Deconvolution* or *Convolution Transpose*
- It is like inverting the Convolution

Naive back propagation does not always give the best results.

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

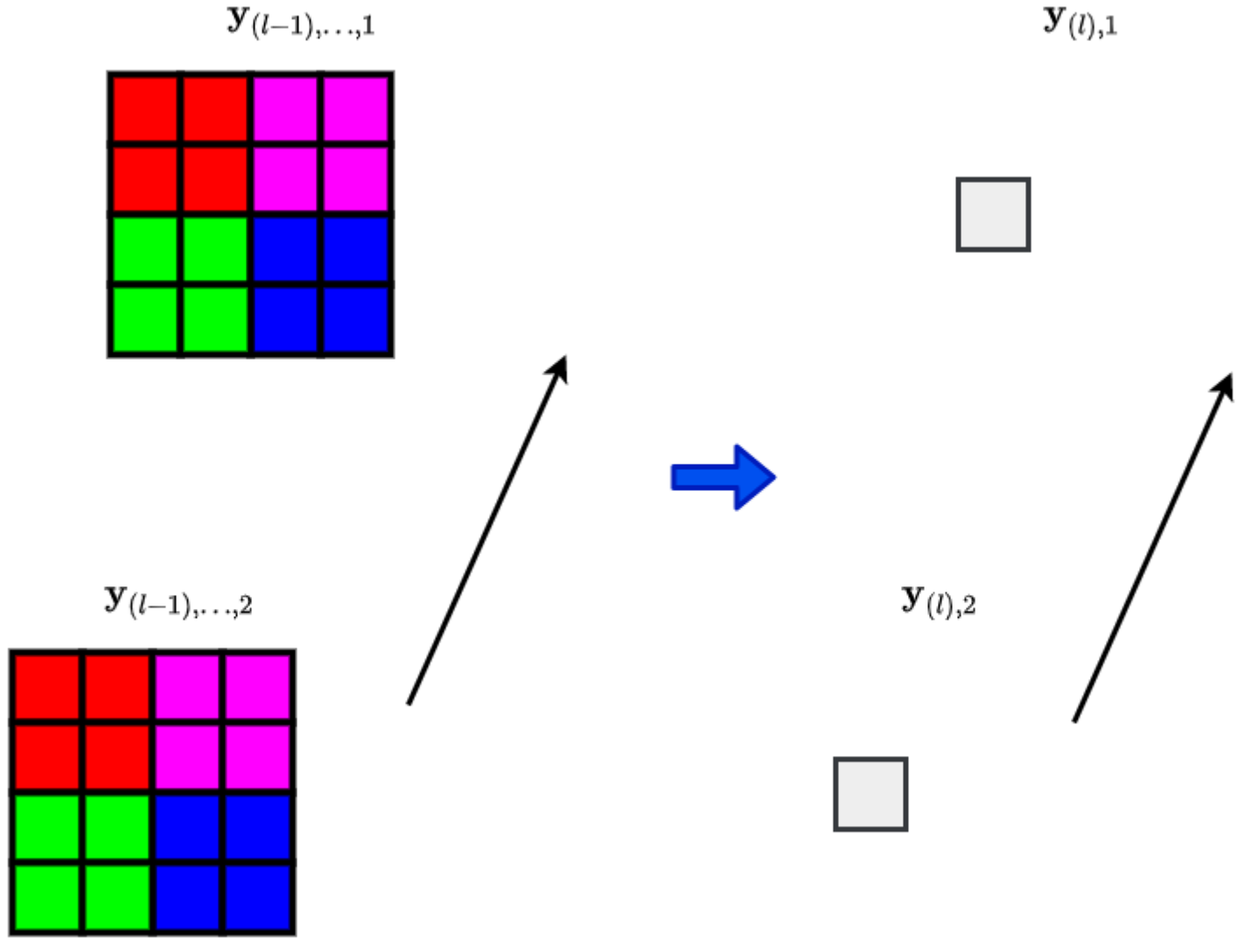
- **In an attempt to get better intuition as to which input features most affect a layer l feature**
- **For example: ignore the *sign* of the derivatives as they flow backwards**
    - **Look for strong positive or negative influences, not caring which**

This is called *Guided Back propagation*.

We also mention that back propagation through some layers is a technical challenge

- **Max Pooling selects one value among all the spatial locations**
- **Which one ?**
- **Solution: Switches to record the location of the max on the Forward Pass**

Conv 2D: Global Pooling (Max/Average)



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
```