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).

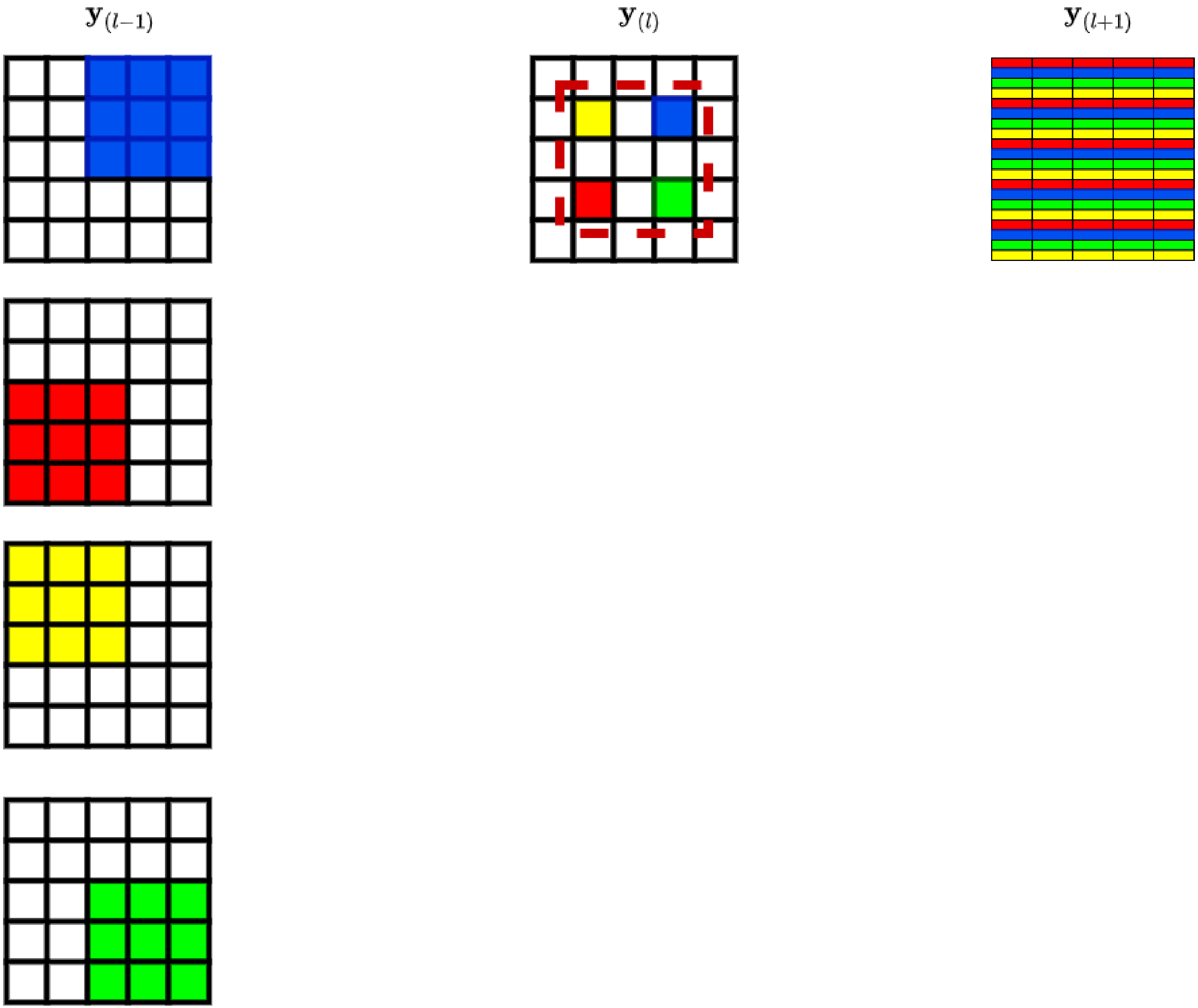
Let's review.

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!

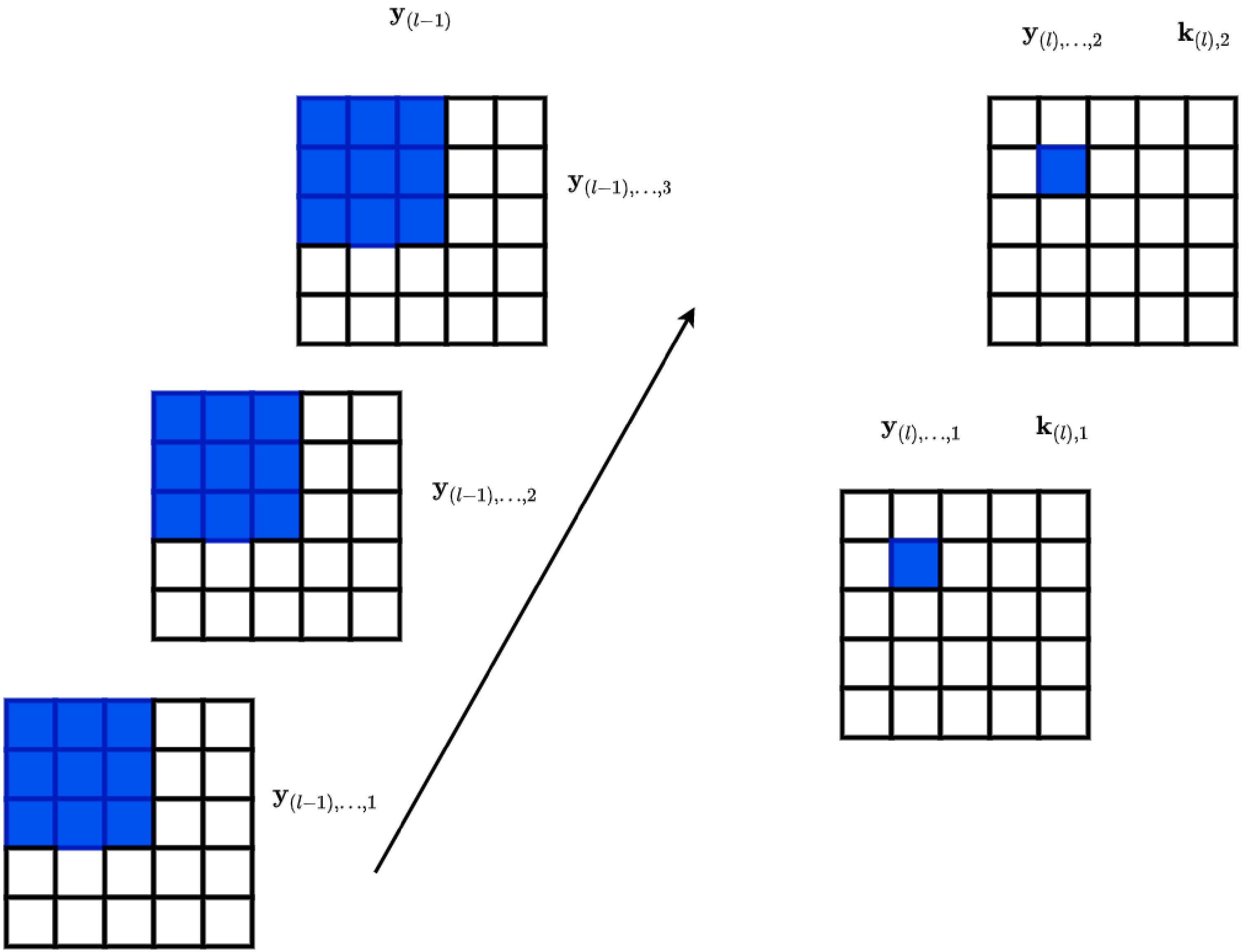
We also omit feature map/channel subscripts (i.e., writing $\mathbf{y}_{(l)}$ rather than $\mathbf{y}_{(l),\ldots,k}$) as they are not necessary for our purpose

- As can be seen by reviewing the mechanics of convolution

This is because of the mechanics of the convolutional dot product

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

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 $(\mathbf{l}+1)$

- Is a function of the spatial locations in layer $\mathbf{l}$ that are encircled by the dashed square
- Which in turn are a function of a larger number of layer $(\mathbf{l}-1)$ 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 date: one-dimensional, two-dimensional

- Let

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

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

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

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

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

for some

$$\begin{aligned} 1 \le & \quad \boldsymbol{k} & \le \boldsymbol{n}_{(l)} \\ 1 \le & \quad \boldsymbol{k}' & \le \boldsymbol{n}_{(0)} \end{aligned}$$

where

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

(Note that $\boldsymbol{k}, \boldsymbol{k}'$ are really not necessary, as we explained due to the mechanics of convolution)

Deconvolution

The receptive field of a single location at layer $\mathbf{l}$

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

We therefore compute the *sensitivity* of a feature

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

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

for each $\mathbf{idx}' \in \mathcal{R}_{(\boldsymbol{l}),\mathbf{idx}}$

For a *single spatial location* $\mathbf{idx}$ of example $\boldsymbol{i}$

- We can arrange the sensitivities of the feature at location $\mathbf{idx}$ of map $\boldsymbol{k}$ of layer $\boldsymbol{l}$ in a grid
- Called a *Saliency Map*
- Which has the same spatial dimensions as the Receptive Field $\mathcal{R}_{(l),\mathbf{idx}}$
- And the same number of features/channels as layer $\mathbf{0}$: $\boldsymbol{n}_{(0)}$

Moreover, the Receptive Field is a contiguous region of the spatial dimensions of layer 0.

So we can *visualize* the Saliency Map on input $\mathbf{x}^{(\mathbf{i})}$

- **Just as we would visualize the *patch* (region) of $\mathbf{x}^{(\mathbf{i})}$ under the Receptive Field**

Note that the Saliency Map is *conditional* on the value of the input $\mathbf{x}^{(\mathbf{i})}$

- Since the value $\mathbf{y}_{(l),\text{idx},k}^{(\mathbf{i})}$ is a function of the particular input

Here are visualizations of 9 Saliency Maps alongside the corresponding patches of the input.

- On the *same* layer 2 feature map
- Using 9 different images
- On the spatial location $\mathbf{idx}$ 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"***

Computing the Saliency Map

Let's show how to compute the Saliency Map.

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

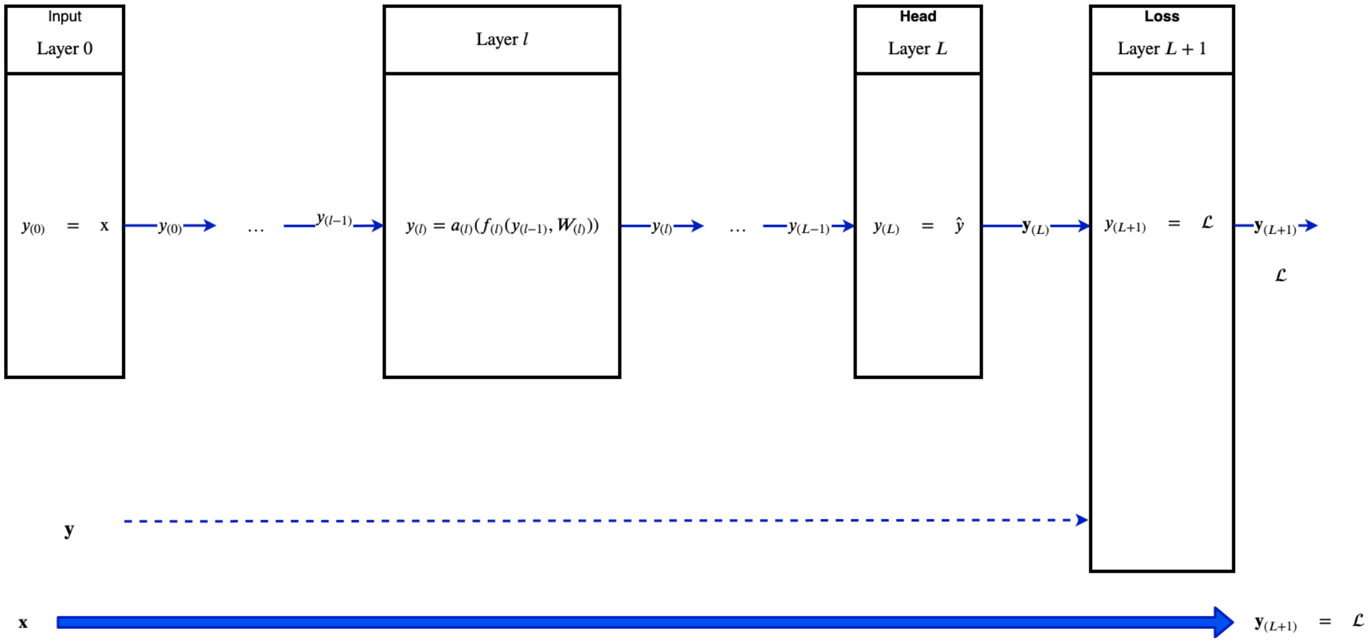
$$\mathbf{y}_{(0)} = \mathbf{x}^{(\mathbf{i})}$$

- Compute $\mathbf{y}_{(l),\text{idx},k}^{(\mathbf{i})}$ 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

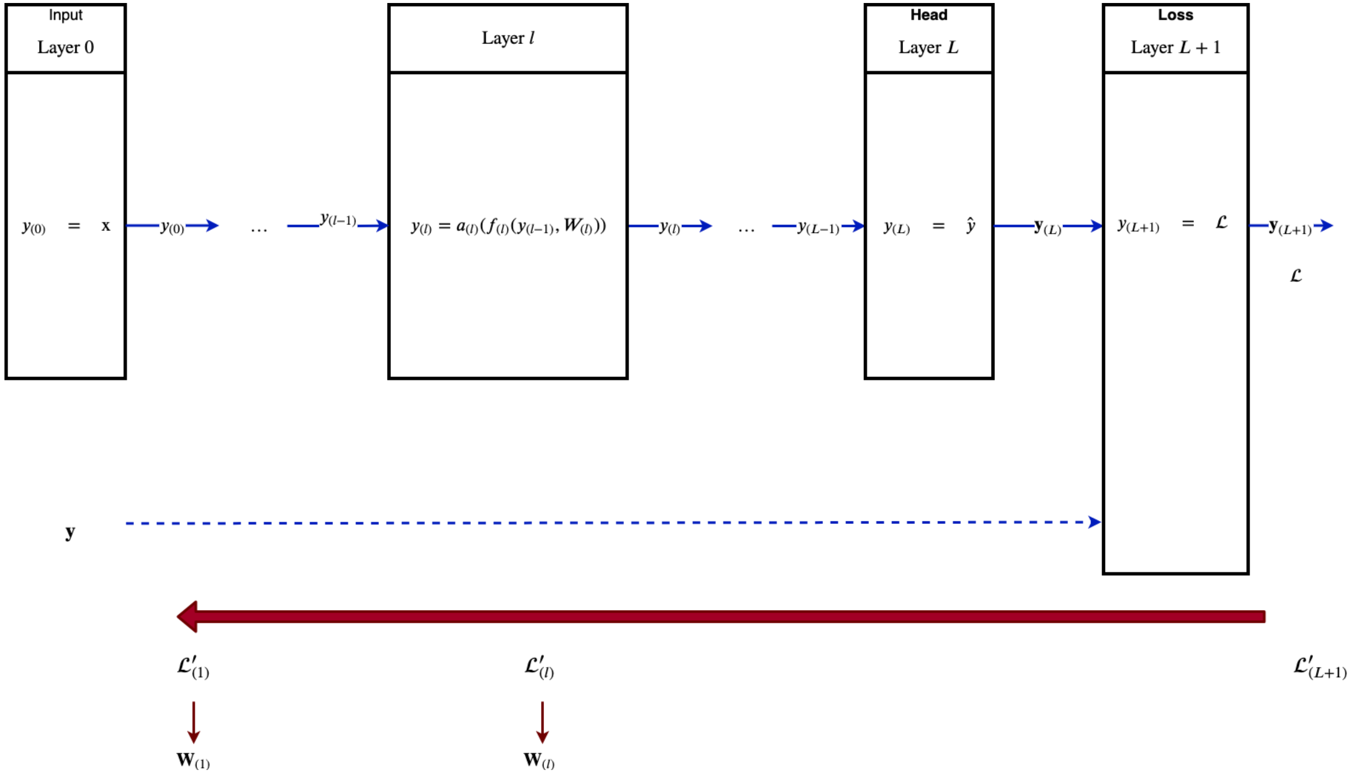
This is very much like the Forward and Backward Passes we saw in the module [Back propagation (Training_Neural_Network_Backprop.ipynb)](Training_Neural_Network_Backprop.ipynb)

Recall the pictures:

Forward Pass: Input to Loss



Backward pass: Loss to Weights



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

The main difference is

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

The Forward Pass

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

Zeiler and Fergus (and similar related papers)

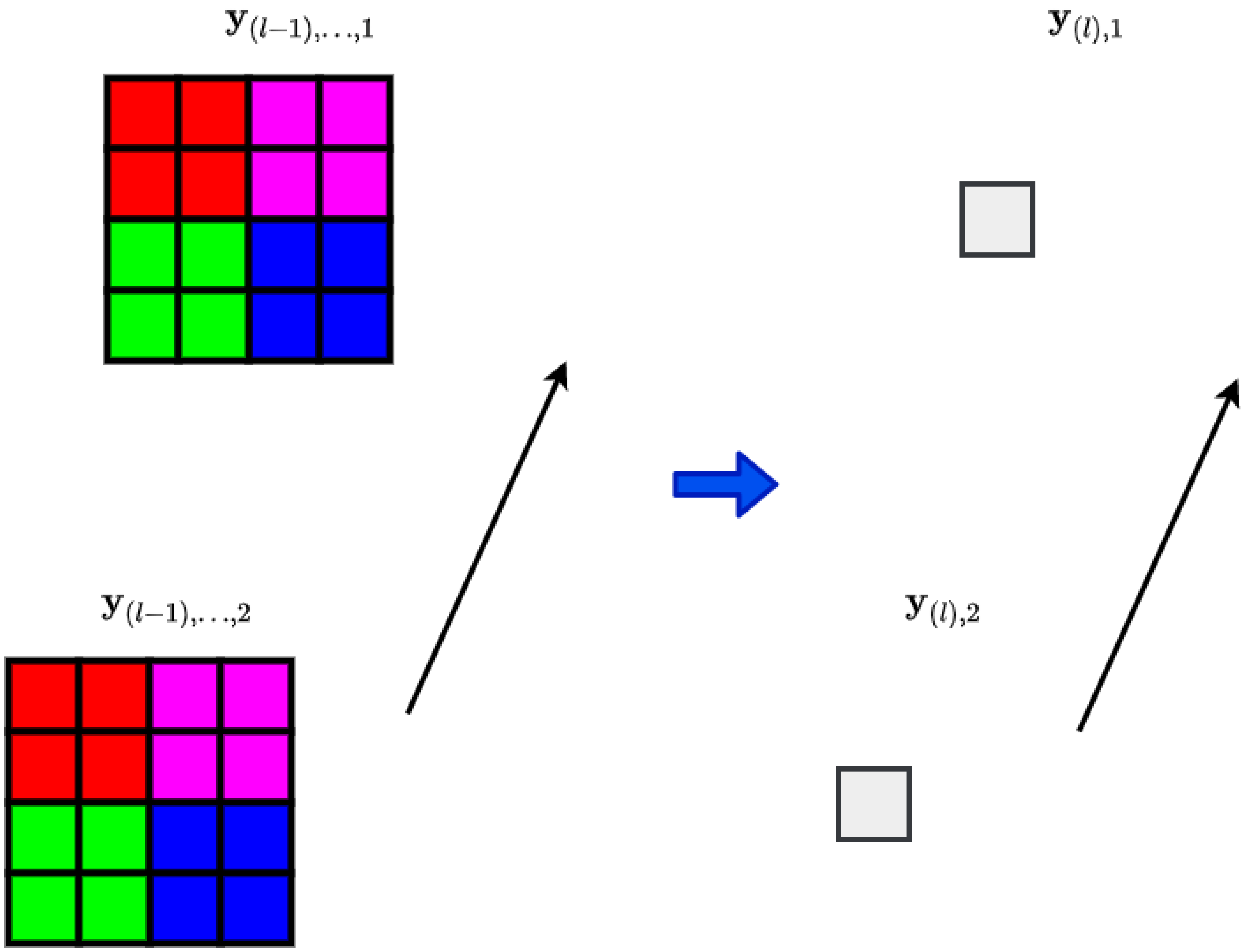
- Modify Back propagation
- In an attempt to get better intuition as to which input features most affect a layer l feature
- For example: ignore the *sign* of the derivatives as they flow backwards
    - Look for strong positive or negative influences, not caring which

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)



Relating a feature to the input

Both Saliency Maps and Patches

- Relate a feature in map k of layer l
- At a single spatial location $\mathbf{idx}$

to a region of input example $\mathbf{x}^{(\mathbf{i})}$.

There are lots of inputs (m of them) and lots of spatial locations.

It is impractical to interpret feature map k by examining all inputs and spatial locations.

We typically compute saliency maps

- For the Maximally Activating Examples for feature map k of layer l
- As these are the examples that maximally excite the feature

(the Saliency maps are computed with respect to the spatial location $\mathbf{idx}$ with greatest intensity for each example)

Zeiler and Fergus create interesting images for each layer using this technique

- For each feature map $\boldsymbol{k}$ at layer $\boldsymbol{l}$
- Find the $\mathbf{9}$ inputs in $\mathbf{X}$ that are maximally activating for the feature map
- Show the Saliency Map and Patch for each of the 9 images

As a reminder, Maximally Activating Examples are defined by

- $\mathbf{MaxAct}_{(l),k} = [\boldsymbol{i}_1, \ldots, \boldsymbol{i}_m]$ is the permutation of example indices, i.e., $[\boldsymbol{i}|\mathbf{1} \leq \boldsymbol{i} \leq \boldsymbol{m}]$ 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 $\boldsymbol{i}$

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

So the 9 inputs chosen are $\mathbf{MaxAct}_{(l),k}[:\mathbf{9}]$ in Python notation

Here are the Saliency Maps and corresponding input image patches

- For a single feature map
- At Layer 2
- Choosing the index $\mathbf{idx}$ of each of the 9 examples that are maximally activating
    - i.e., examples $\mathbf{MaxAct}_{(l),k}[:\mathbf{9}]$ in Python notation

Saliency Maps and Corresponding Patches
Single Layer 2 Feature Map
On 9 Maximally Activating Input images

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

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

If the 9 maximally activating inputs have patches with a common, recognizable patten

- Interpret feature map k of layer l as being responsible for creating the feature
    - "Is pattern present in input"

Perhaps this feature map is recognizing an "eye" ?

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)

Further exploration

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

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
```