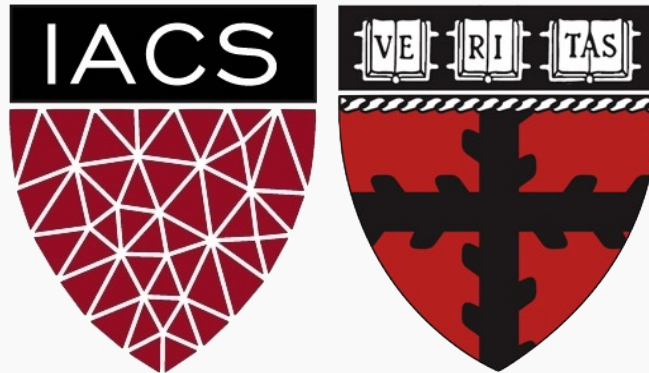


# Convolutional Neural Networks IV

## Saliency Maps

CS109B Data Science 2  
Pavlos Protopapas, Mark Glickman



# Outline

---

1. Gradient Base
2. Deconvolution, Guided Backpropagation Algorithm
3. Class Activation Map (CAM), Grad-CAM and Guided Grad-CAM

# Outline

---

1. **Gradient Base**
2. Deconvolution, Guided Backpropagation Algorithm
3. Class Activation Map (CAM), Grad-CAM and Guided Grad-CAM

# Saliency maps

If you are given the image below and you are asked to classify it,



# Saliency maps

If you are given the image below and you are asked to classify it, most probably you will answer immediately – **Dog!**



# Saliency maps

If you are given the image below and you are asked to classify it, most probably you will answer immediately – **Dog!**

But your Deep Learning Network might not be as smart as you. It might classify it as a cat, a lion or Pavlos!



# Saliency maps

If you are given the image below and you are asked to classify it, most probably you will answer immediately – **Dog!**

But your Deep Learning Network might not be as smart as you. It might classify it as a cat, a lion or Pavlos!

What are the reasons for that?



# Saliency maps

If you are given the image below and you are asked to classify it, most probably you will answer immediately – **Dog!**

But your Deep Learning Network might not be as smart as you. It might classify it as a cat, a lion or Pavlos!

What are the reasons for that?

- bias in training data
- no regularization
- or your network has seen too many celebrities [therefore Pavlos]





# Saliency maps

---

We want to understand what made my network output a certain class.

# Saliency maps

---

We want to understand what made my network output a certain class.

**Saliency Maps**, are a way to measure the spatial support of a particular class in each image.

*“Find me pixels responsible for the class  $C$  having score  $S(C)$  when the image  $I$  is passed through my network”.*

# Saliency maps

---

We want to understand what made my network output a certain class.

**Saliency Maps**, are a way to measure the spatial support of a particular class in each image.

*“Find me pixels responsible for the class  $C$  having score  $S(C)$  when the image  $I$  is passed through my network”.*

**Note:** In the previous lecture, we looked at the **occlusion** method. Occlusion method is a **forward** pass attribution method. In this lecture, we will be looking at **backward** methods for decision attribution.

# Saliency maps: Methods

Saliency map is the oldest and most frequently used explanation method for interpreting the predictions of convolutional neural networks (CNNs).

There are five main approaches to getting the saliency map:

1. **Gradient Based Backpropagation**, [Symonian et al. 2013](#)
2. **Deconvolutional Networks**, [Zeiler and Fergus 2013](#)
3. **Guided Backpropagation Algorithm**, [Springenberg et al. 2014](#)
4. **Class Activation Maps**, [Zhou et al. 2016](#)
5. **Grad-CAM and Guided Grad-CAM**, [Selvaraju et al. 2016](#)

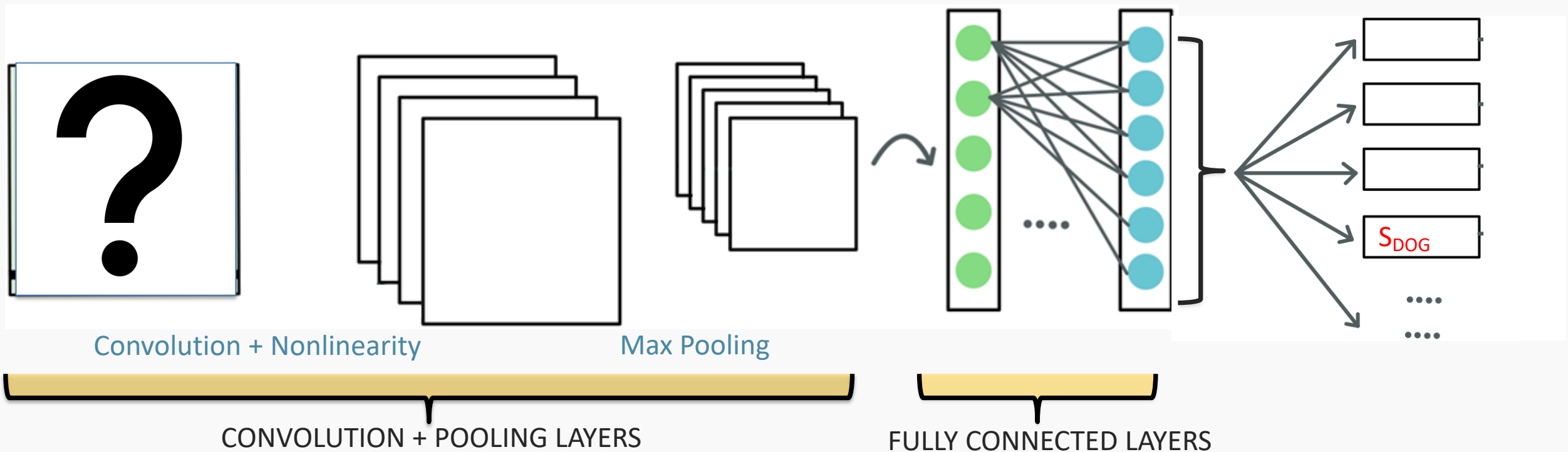
# Saliency maps: Notes

---

**Note:** The method which is referred to as “Gradients” is sometimes also referred to as “backpropagation” or even just “saliency mapping” – although the other techniques are also ways to accomplish “saliency mapping”.

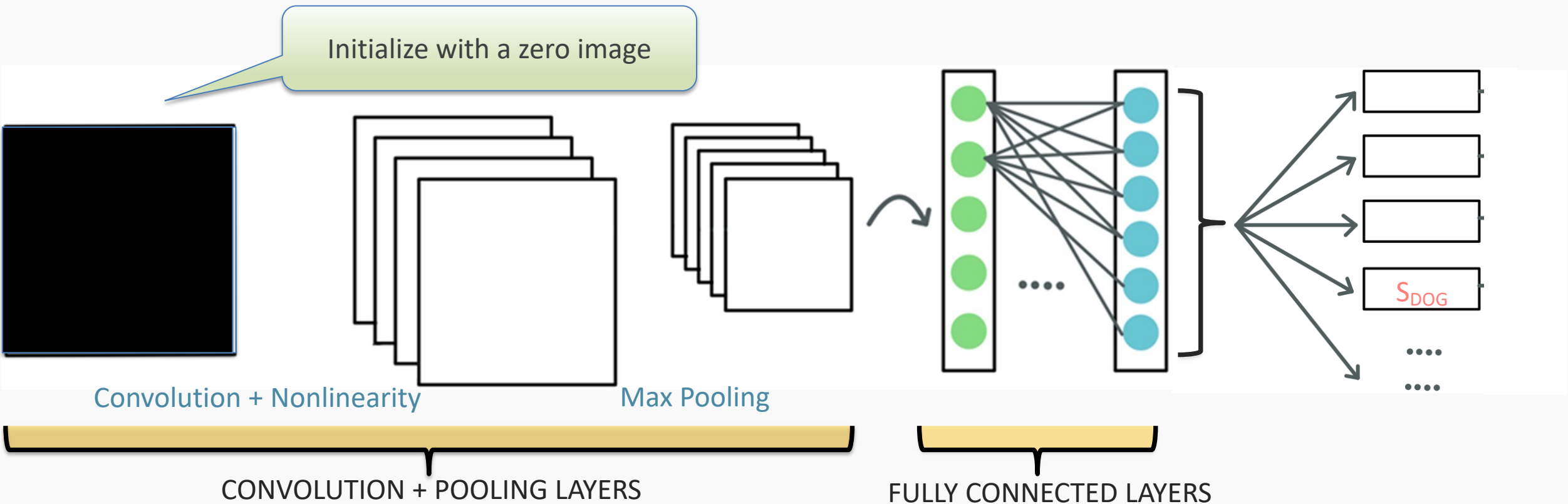
# Saliency Maps: Class Model Visualization

For a **trained CNN**, the visualization method consists of numerically **generating an image** representing a class.



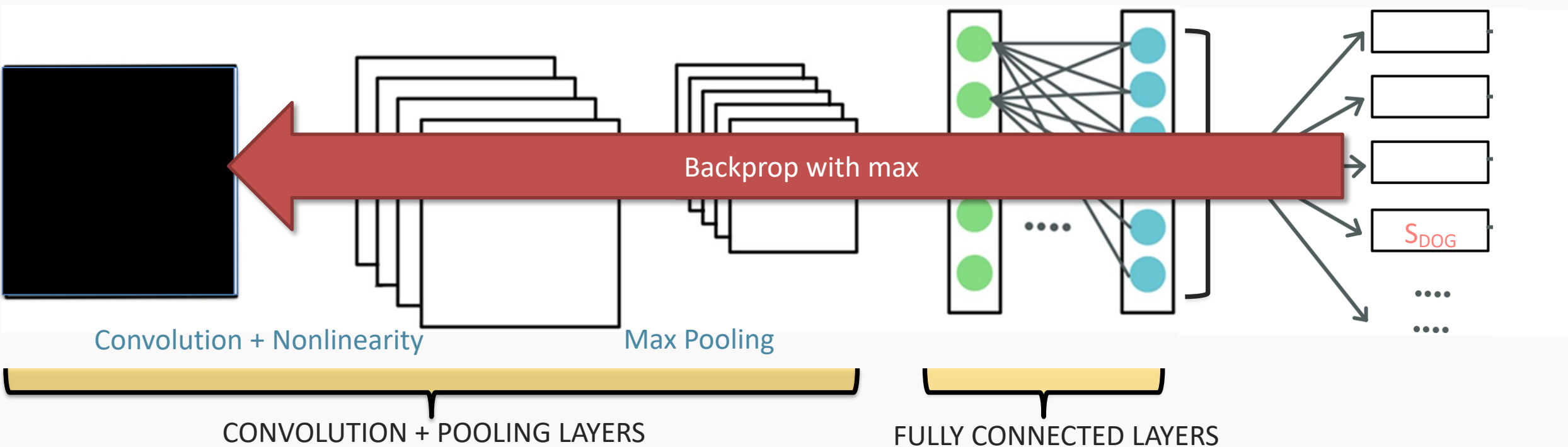
# Saliency Maps: Class Model Visualization

This means inverting the problem. We want an image that maximizes a specific class score, **fixing** the values of **the trained weights**.



# Saliency Maps: Class Model Visualization

For a given class and the score function  $S_c(I)$  (given by the logits), we **maximize** the input image.



**Note:** We don't use the softmax values, as the maximization of the class can be achieved by minimizing the scores of other classes.



# Saliency Maps: Class Model Visualization

Formally, we want to find the image  $I$  such that

$$I^* = \arg \max_I S_c(I)$$

Finding the image that maximizes the logits or any element of the feature map is called [activation maximization](#).

We need to L2-regularise, such that  $I$  is bounded:

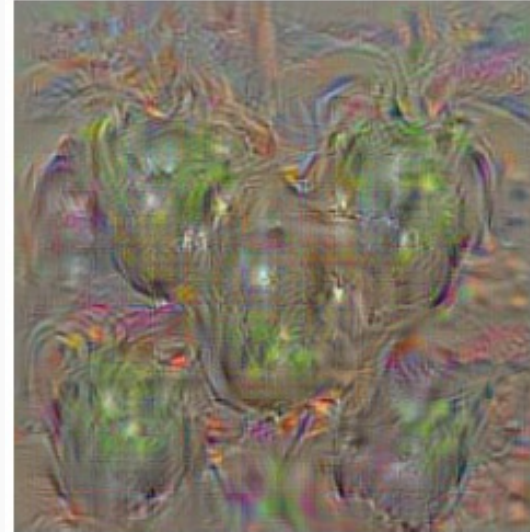
$$I^* = \arg \max_I S_c(I) - \lambda \|I\|_2^2$$

Where  $S_c(I)$  are the logits of the class  $c$  given the image  $I$ .

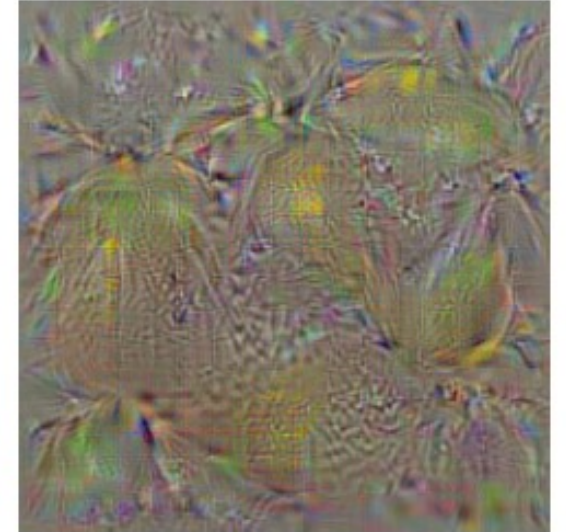
# Saliency Maps: Class Model Visualization

The results are images that represent a local maximum.

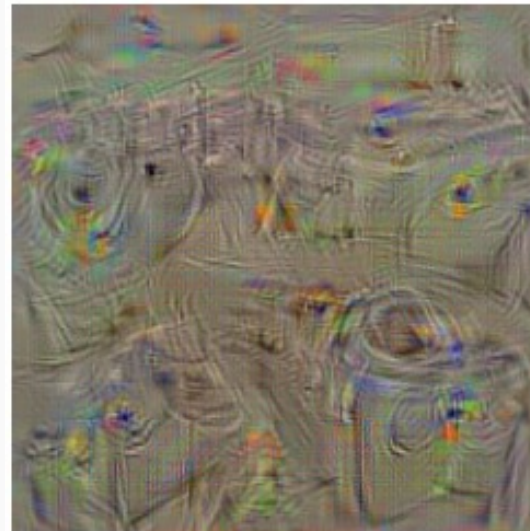
Since the logits of the network have a receptive field as big as the image, we can identify features of the specific class all over the image.



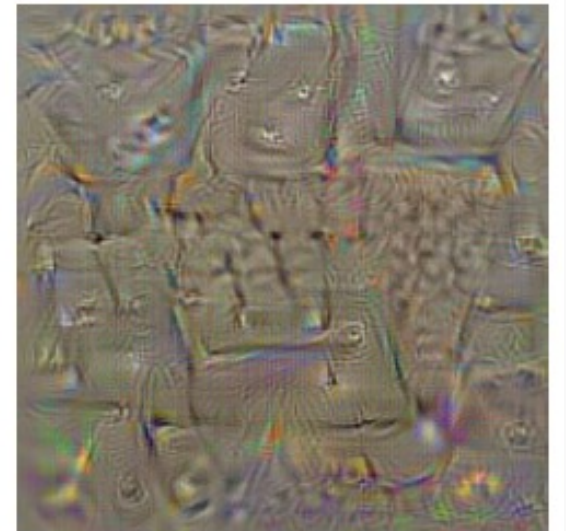
**bell pepper**



**lemon**



**washing machine**

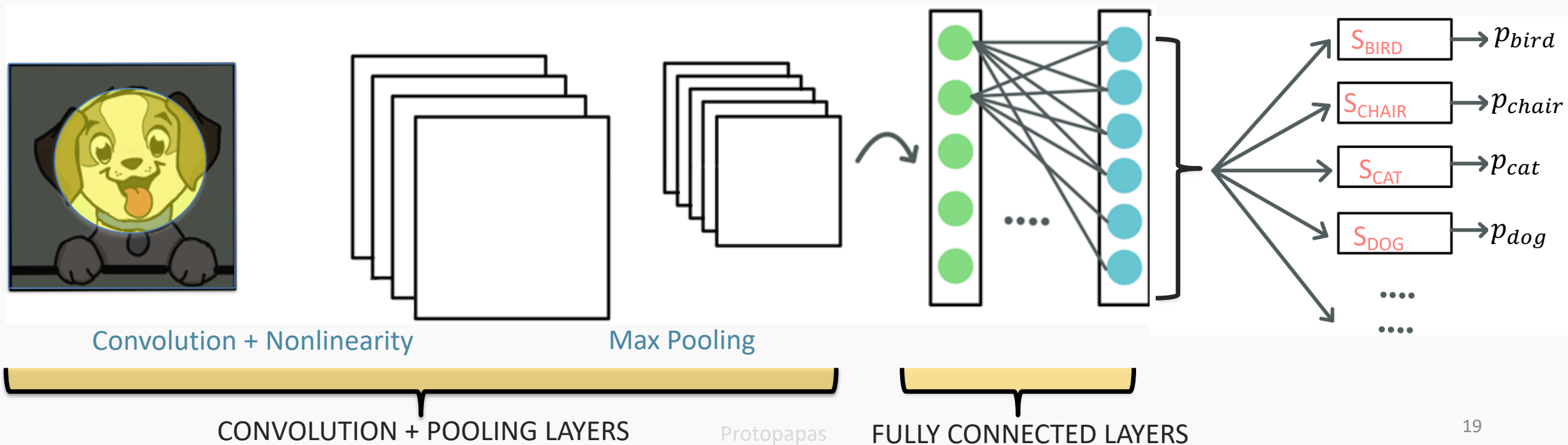


**computer keyboard**

# Saliency Maps with Gradient Based Backprop

What if we want to detect what part of a **specific image**  $I_0$  that contains the relevant information?

Intuitively, the relevant pixels are located where the object is. As such, that part should be highlighted.



# Saliency Maps with Gradient Based Backprop

$S_c(I)$  are the logits,  
not the output of the  
softmax

[Symoniam et al.](#) were the first to propose a method that uses the backpropagation algorithm to compute the gradients of **logits** w.r.t. to the input of the network while the weights are fixed (for training the network, the gradients are computed w.r.t. the parameters of the network). Using Taylor expansions, we can show that:

$$S_c(I) \approx w^T I + b$$

where

$$w = \left. \frac{\partial S_c}{\partial I} \right|_{I_0}$$

Using backpropagation, we can **highlight pixels** of the input image based on the amount of the **gradient they receive**, which shows their contribution to the final score.

# Saliency Maps with Gradient Based Backprop

The maps were extracted using a single back-propagation pass through a classification ConvNet.

No additional annotation (except for the image labels) was used in training.

**Note:** For color images like the one shown here, we take the maximum derivative of the three derivatives.



Symonian et al, [Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps](#), 2013



# Outline

---

1. Gradient Base
- 2. Deconvolution, Guided Backpropagation Algorithm**
3. Class Activation Map (CAM), Grad-CAM and Guided Grad-CAM



# Saliency Maps with DeconvNet

---

[Zeiler and Fergus](#) proposed a method able to recognize what features in the input image, an intermediate layer of the network is looking for.

The method is known as **Deconvolutional Network** (DeconvNet) because it inverts the operations from a particular layer to the input using that specific layer's activation.

To invert the process, the authors used:



# Saliency Maps with DeconvNet

---

[Zeiler and Fergus](#) proposed a method able to recognize what features in the input image an intermediate layer of the network is looking for.

The method is known as **Deconvolutional Network** (DeconvNet) because it inverts the operations from a particular layer to the input using that specific layer's activation.

To invert the process, the authors used:

- **Unpooling** as the inverse of pooling

# Saliency Maps with DeconvNet

---

[Zeiler and Fergus](#) proposed a method able to recognize what features in the input image an intermediate layer of the network is looking for.

The method is known as **Deconvolutional Network** (DeconvNet) because it inverts the operations from a particular layer to the input using that specific layer's activation.

To invert the process, the authors used:

- **Unpooling** as the inverse of pooling
- **Deconvolution** to backpropagate the derivatives

# Saliency Maps with DeconvNet

---

[Zeiler and Fergus](#) proposed a method able to recognize what features in the input image an intermediate layer of the network is looking for.

The method is known as **Deconvolutional Network** (DeconvNet) because it inverts the operations from a particular layer to the input using that specific layer's/filter activation.

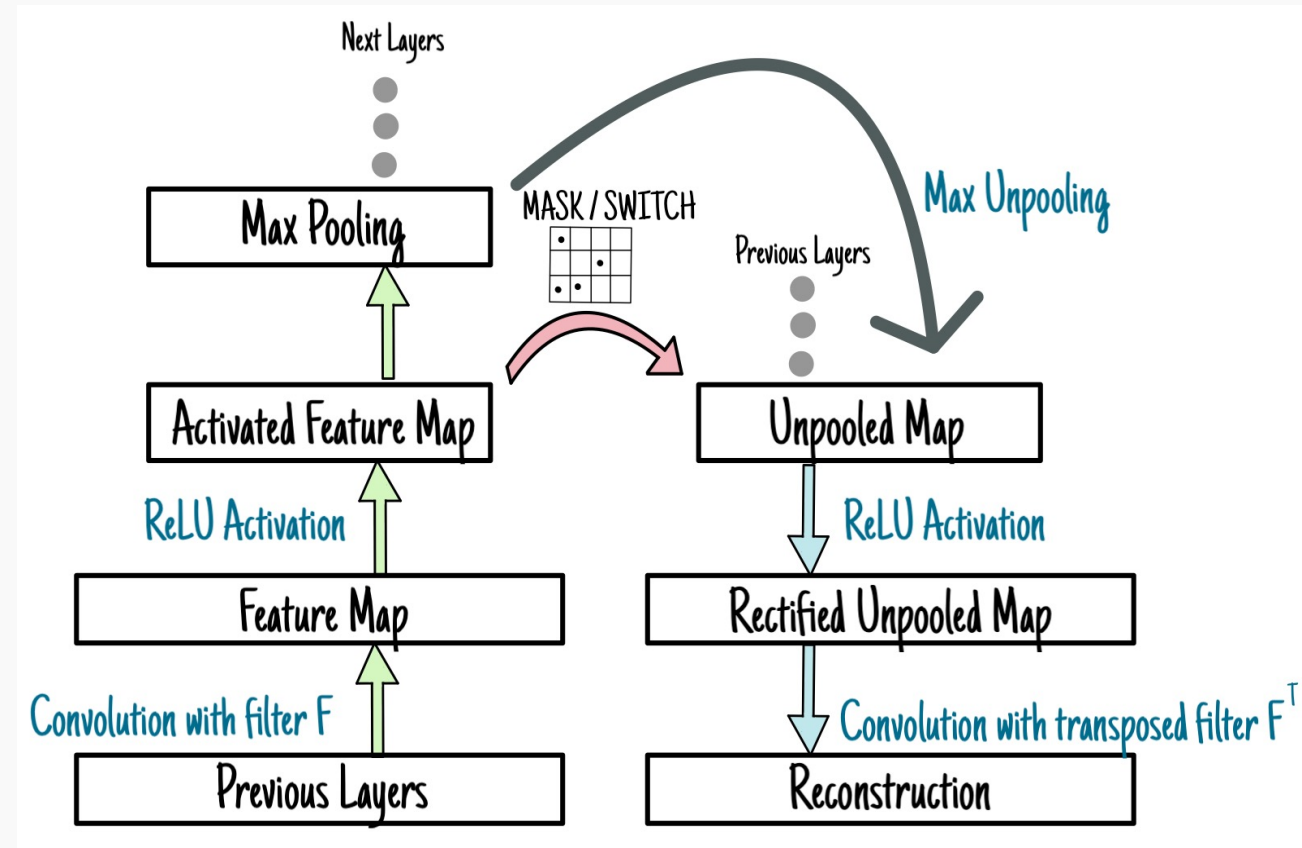
To invert the process, the authors used:

- **Unpooling** as the inverse of pooling
- **Deconvolution** to backpropagate the derivatives
- **Inverse ReLU** to remove the negative values as the inverse of itself

# Saliency Maps with DeconvNet

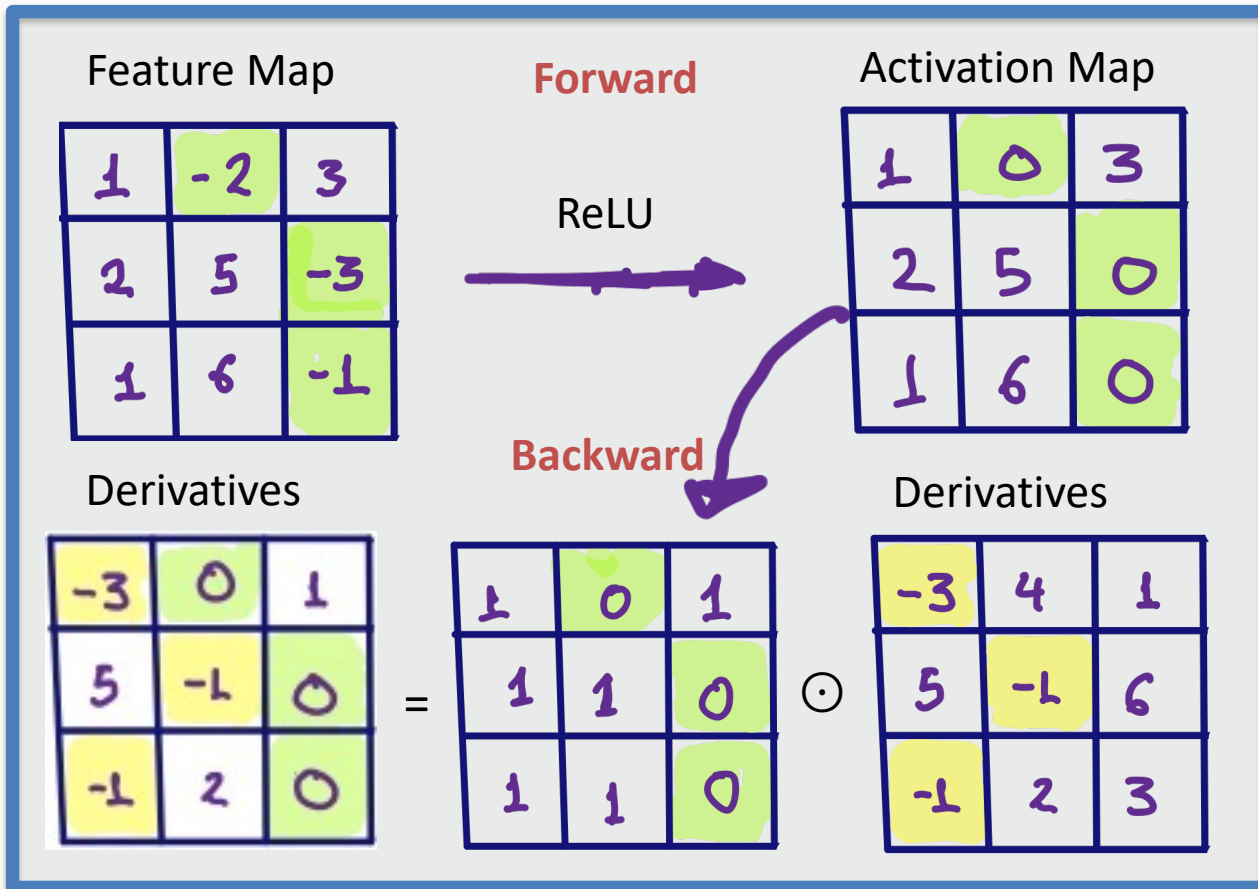
The whole process is illustrated in the figure on the right.

The authors used a module called **switch (mask)** to recover maxima positions in the forward pass because the pooling operation is non-invertible.



# Saliency Maps with DeconvNet: Inverse ReLU

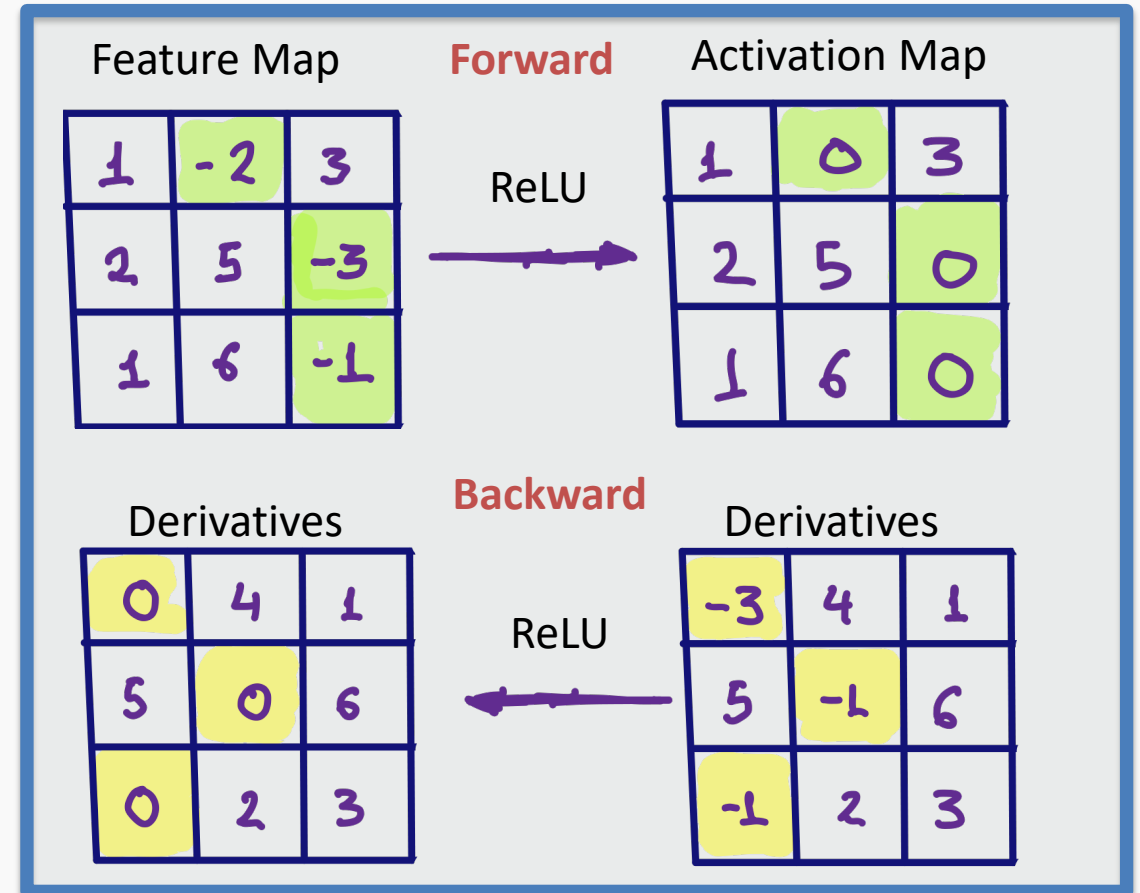
## Vanilla BackPropagation



Given an input image, perform the forward pass to the layer we are interested in, set to zero all activations except one and propagate back to the input image.

Protopapas

## Deconvolution

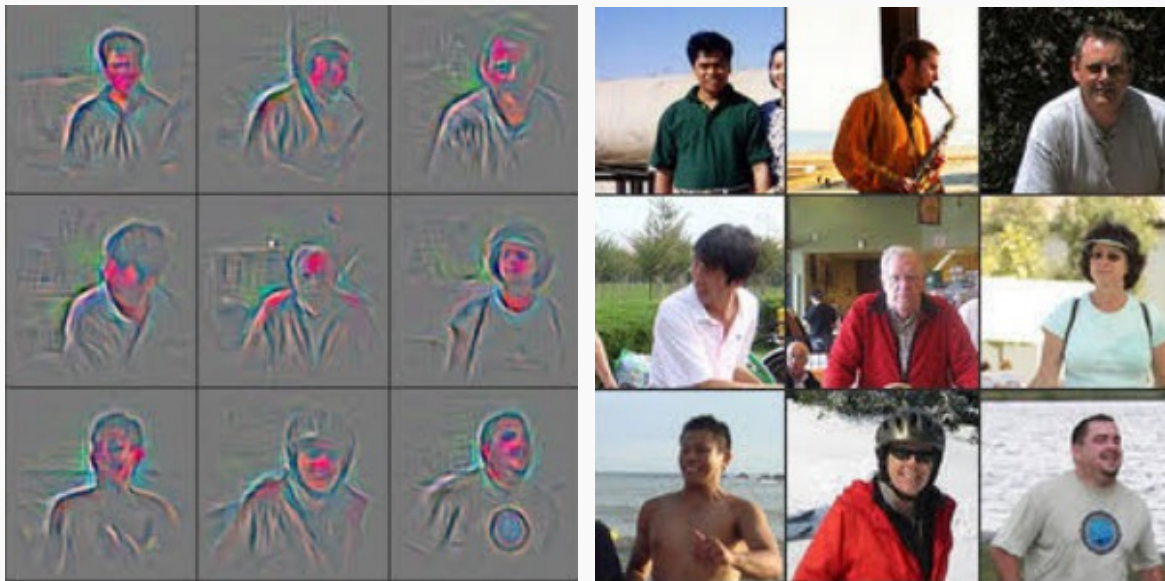


Different methods of propagating back through a ReLU nonlinearity.

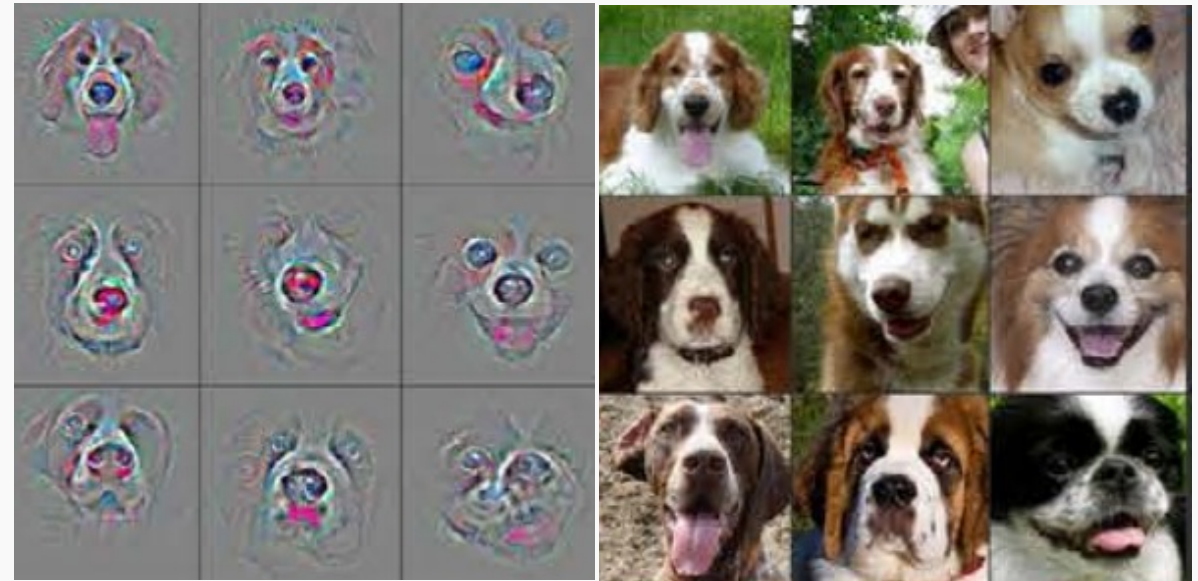
# Saliency Maps with DeconvNet

Two examples of patterns that cause high activations in feature maps of layer 4 and layer 5.

Layer 4



Layer 5



Right panels, image patches showing which patterns from the training set activate the feature map.

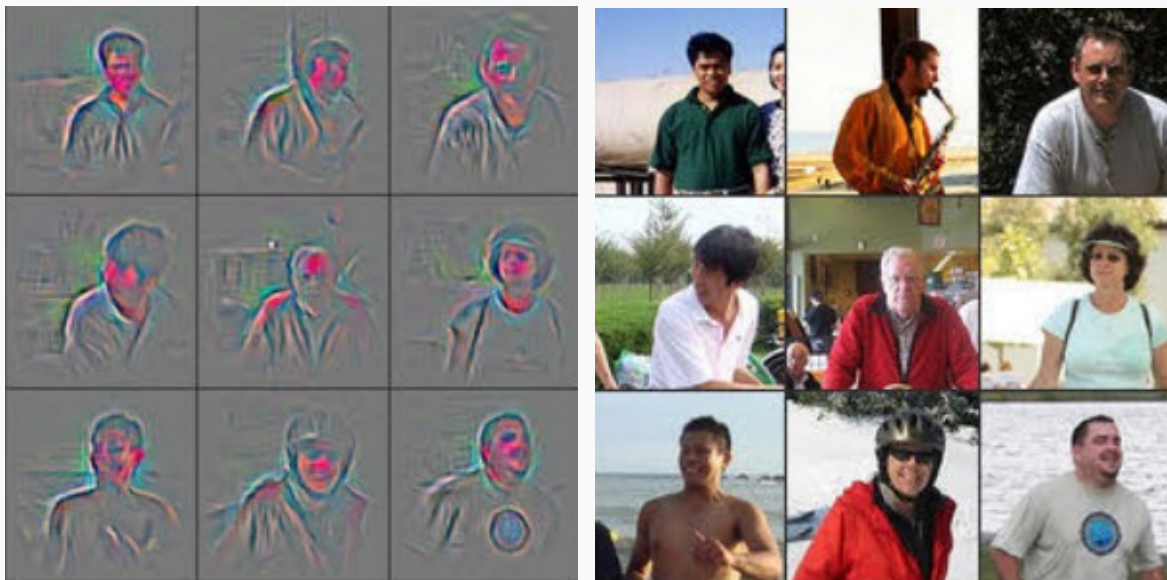


# Saliency Maps with DeconvNet

Two examples of post-deconv and inputs that give high activation and layer 5.

Layer4/one feature map. 9 deconv and inputs that give the highest activations.

Layer 4



Layer5/one feature map. 9 deconv and inputs that give the highest activations.

Layer 5



# Saliency Maps Guided Backpropagation Algorithm

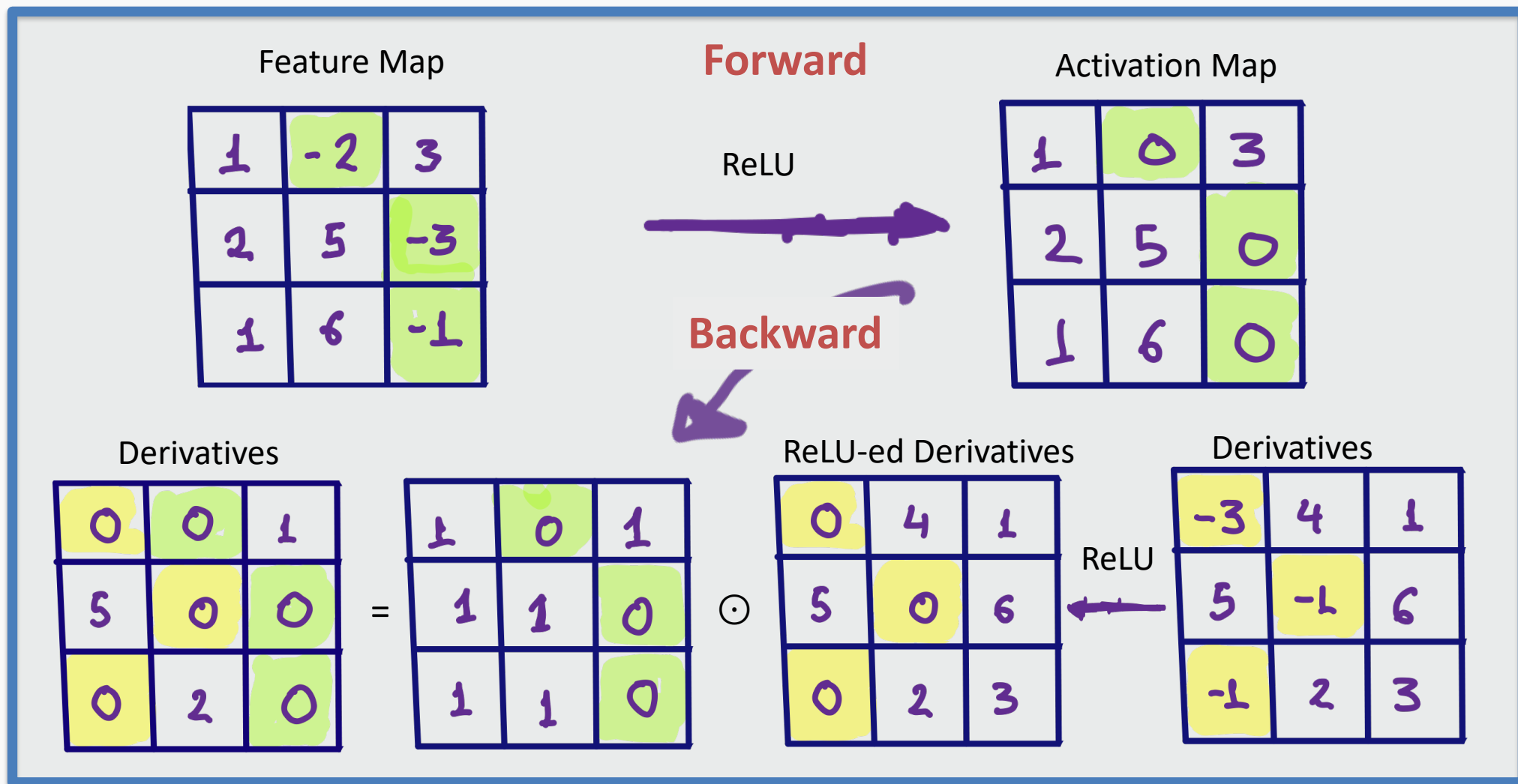
---

[Springenberg et al.](#) **combined** DeconvNet and Gradient-Based Backpropagation and proposed the **Guided Backpropagation Algorithm** as another way of getting saliency maps.

Instead of masking the importance of the signal based on the positions of negative values of the input in forward-pass or the negative values from the reconstruction signal flowing from top to bottom (deconvolution), they **mask the signal for both cases.**

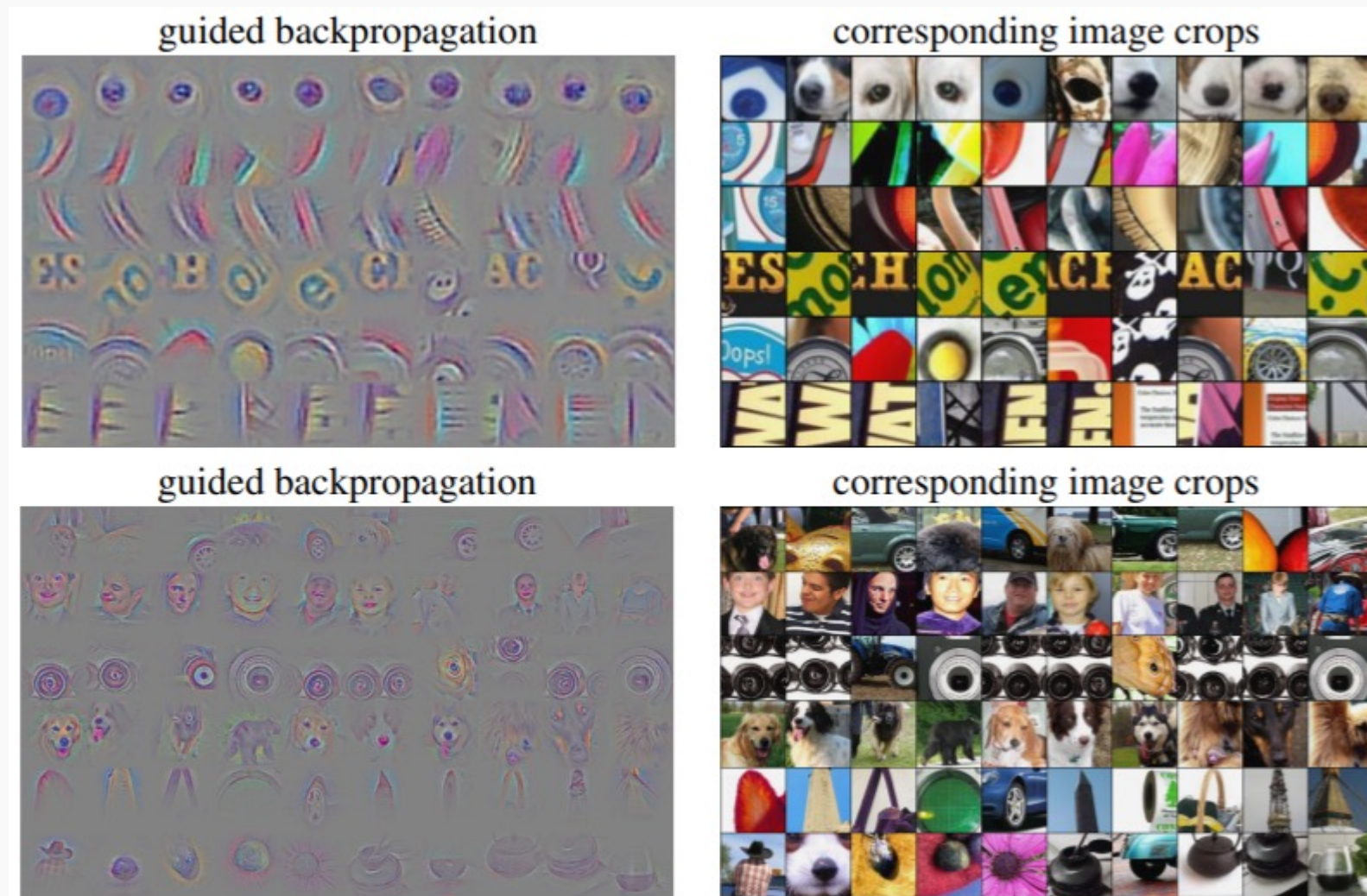


# Saliency Maps Guided Backpropagation Algorithm



# Saliency Maps Guided Backpropagation Algorithm

Visualization of patterns learned by the layer conv6 (top) and layer conv9 (bottom) of the network trained on ImageNet. Each row corresponds to one filter. Again, we show 10 examples that produces the maximum activations.



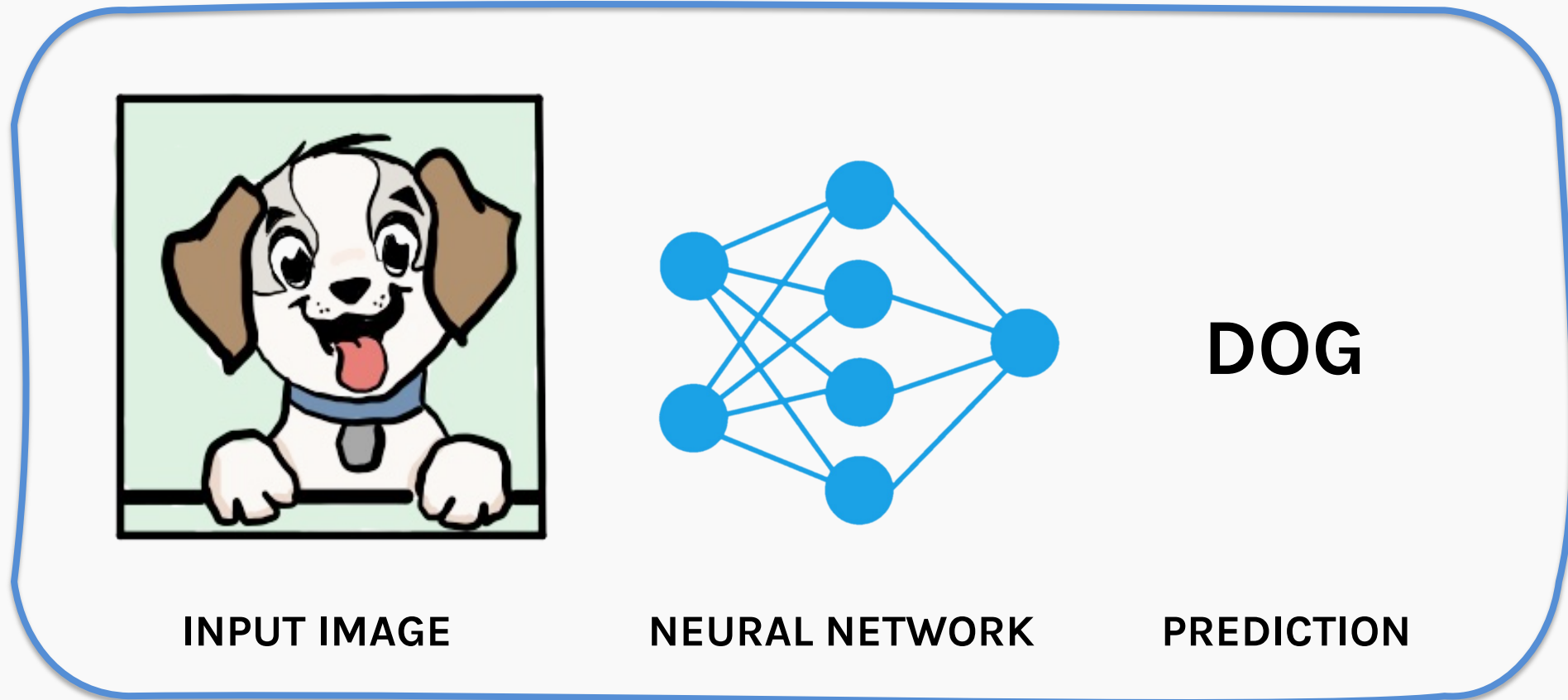
Springenberg et al., [Striving for Simplicity](#), 2014

# Outline

---

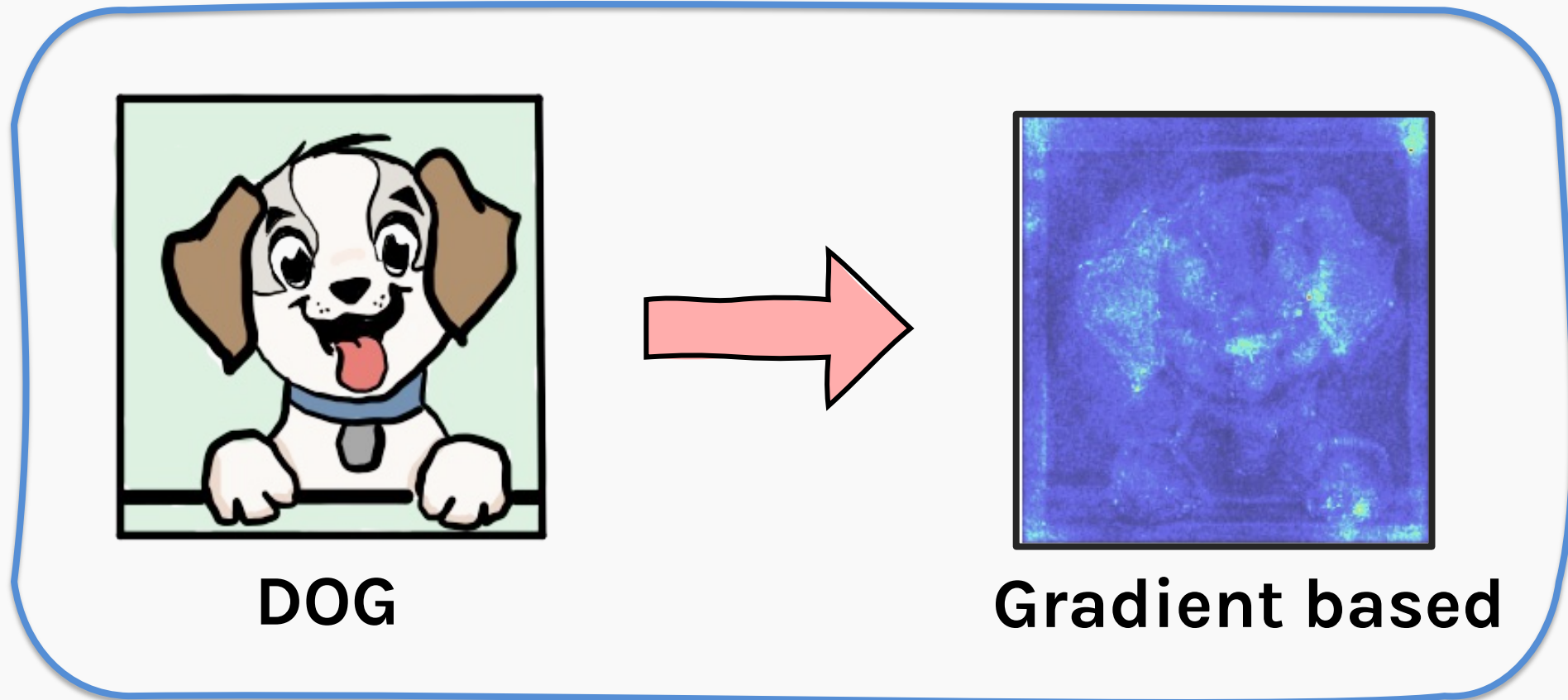
1. Gradient Base
2. Deconvolution, Guided Backpropagation Algorithm
3. **Class Activation Map (CAM)**, Grad-CAM and Guided Grad-CAM

# Class Activation Mapping (CAM)



An image classification CNN takes an input image and outputs the image label. We want to know what part of the image the model is “looking” at, while making the prediction?

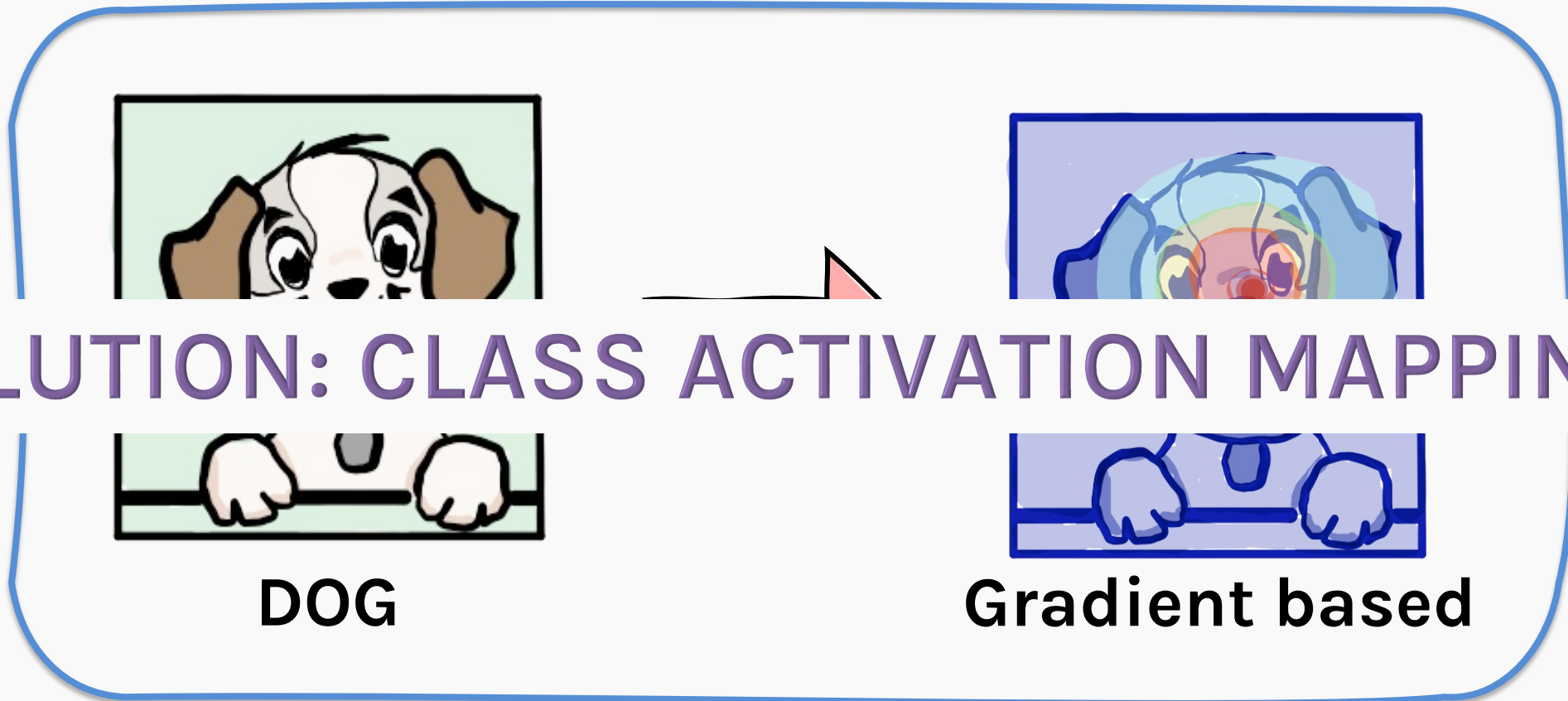
# Class Activation Mapping (CAM)



Gradient based methods give us pixel by pixel importance. However, these are not very useful for localized feature display, for e.g., the dog in the image.



# Class Activation Mapping (CAM)



## WHAT WE WANT?

A method to estimate which sub-part of the image the model focuses at when making a particular prediction. For example, Dog in the above image.

# Class Activation Mapping (CAM)

Class Activation Mapping (CAM) is another explanation method for interpreting convolutional neural networks (CNNs) introduced by [Zhou et al. 2016](#).

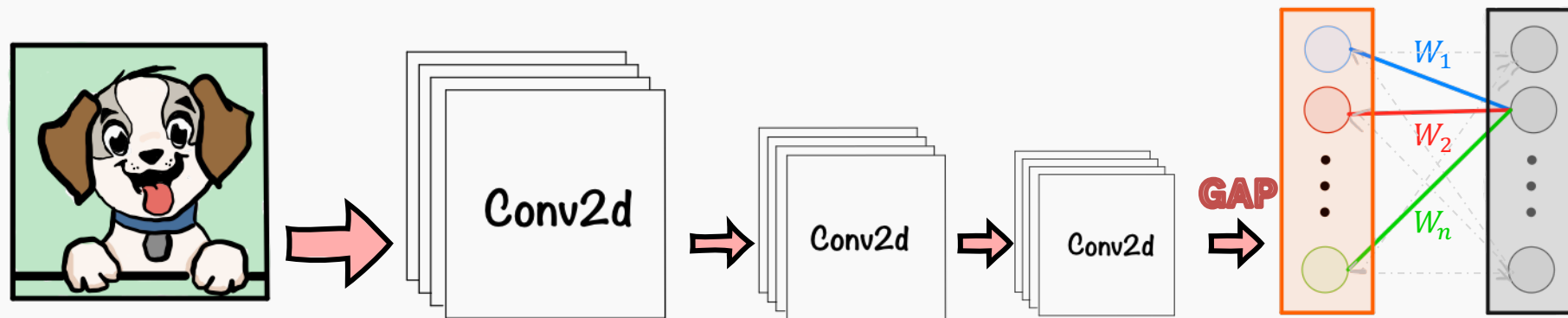
They proposed a network where the fully connected layers at the very end of the model have been replaced by a layer named **Global Average Pooling (GAP)** and combined with a class activation mapping (CAM) technique.



# Class Activation Mapping (CAM)

## HOW DO WE DO IT?

- Instead of Dense layer, we use a **Global Average Pooling (GAP)** to **average** the activations of each feature map.
- Each of the averages is concatenated into a single vector (shown in color red below).
- Then, a weighted sum of the resulted vector is fed to the final softmax layer.

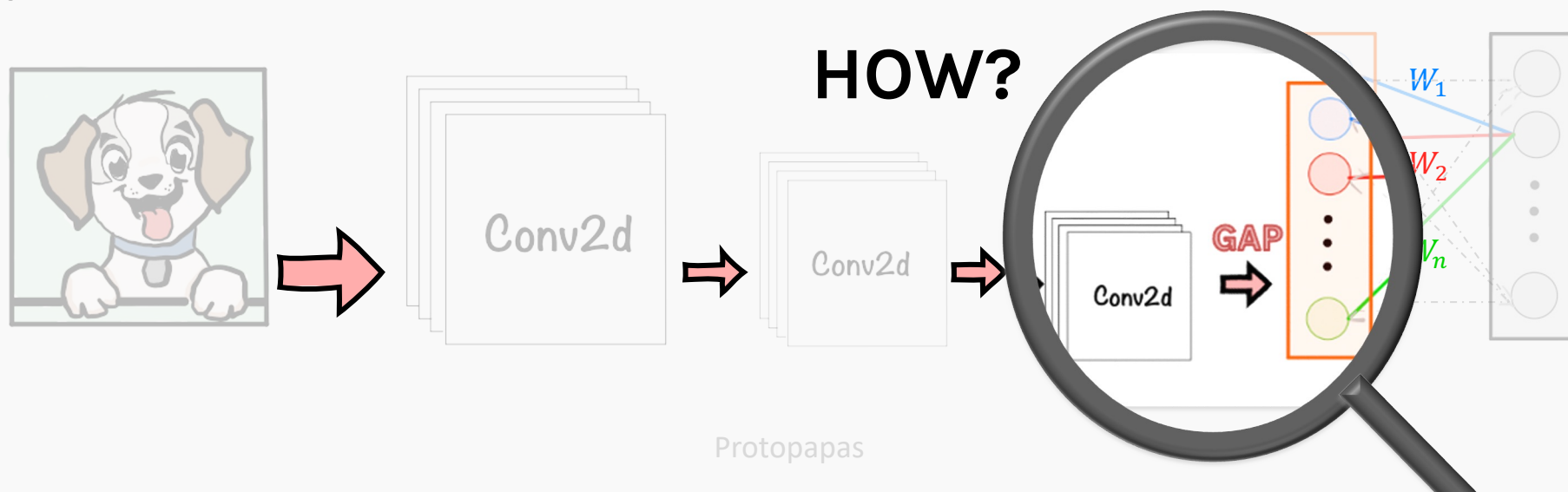




# Class Activation Mapping (CAM)

## HOW DO WE DO IT?

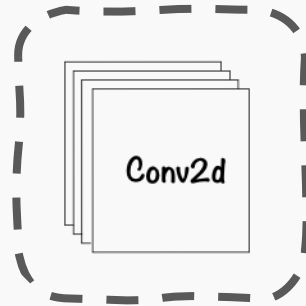
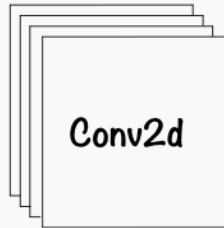
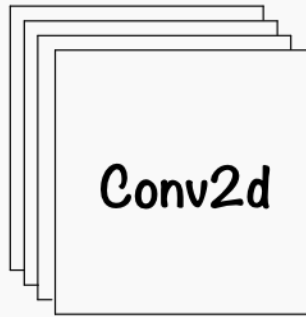
- Instead of Dense layer, we use a **Global Average Pooling (GAP)** to **average** the activations of each feature map.
- Each of the averages is concatenated into a single vector (shown in color red below).
- Then, a weighted sum of the resulted vector is fed to the final softmax layer.



# Global Average Pooling

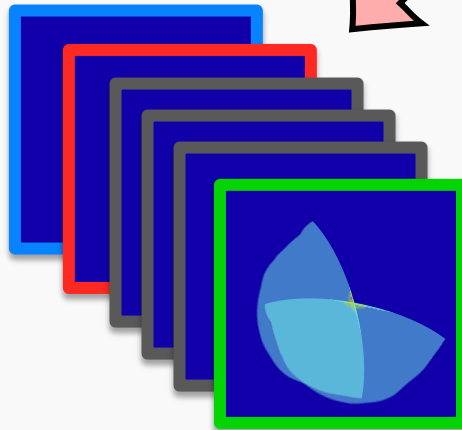
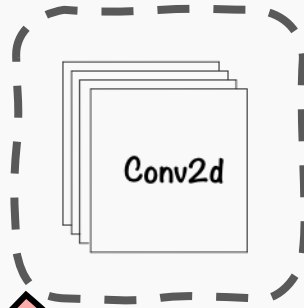
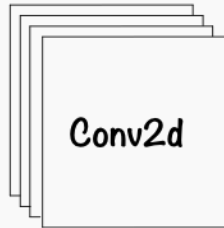
## HOW DO WE DO IT?

LAST CONVOLUTION LAYER



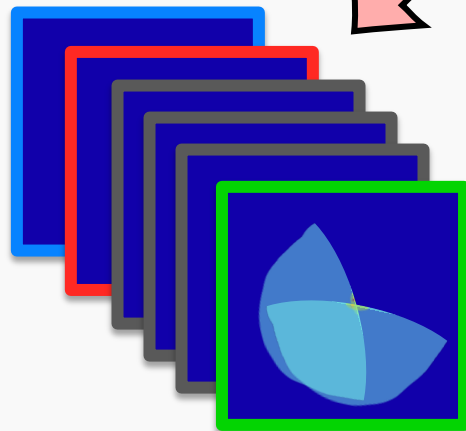
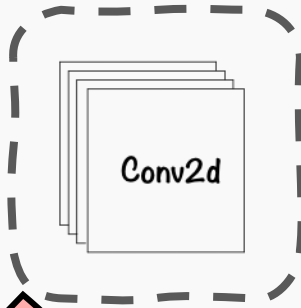
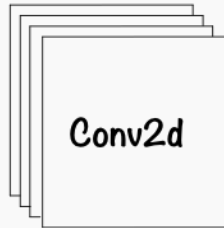
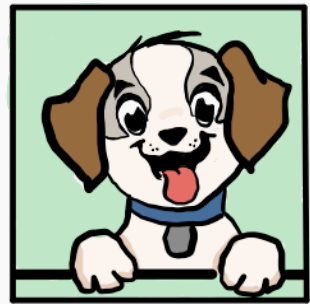
# Global Average Pooling

## HOW DO WE DO IT?

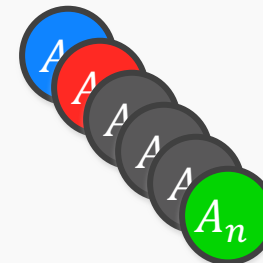


# Global Average Pooling

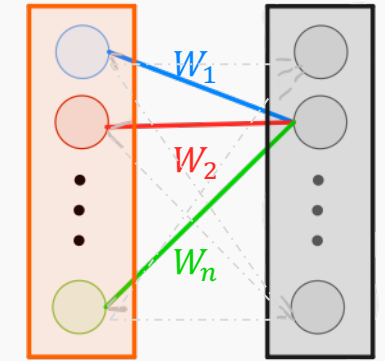
## HOW DO WE DO IT?



$$A_i = \sum_{j,k} a_{i,j,k}$$



LAST CONVOLUTION LAYER

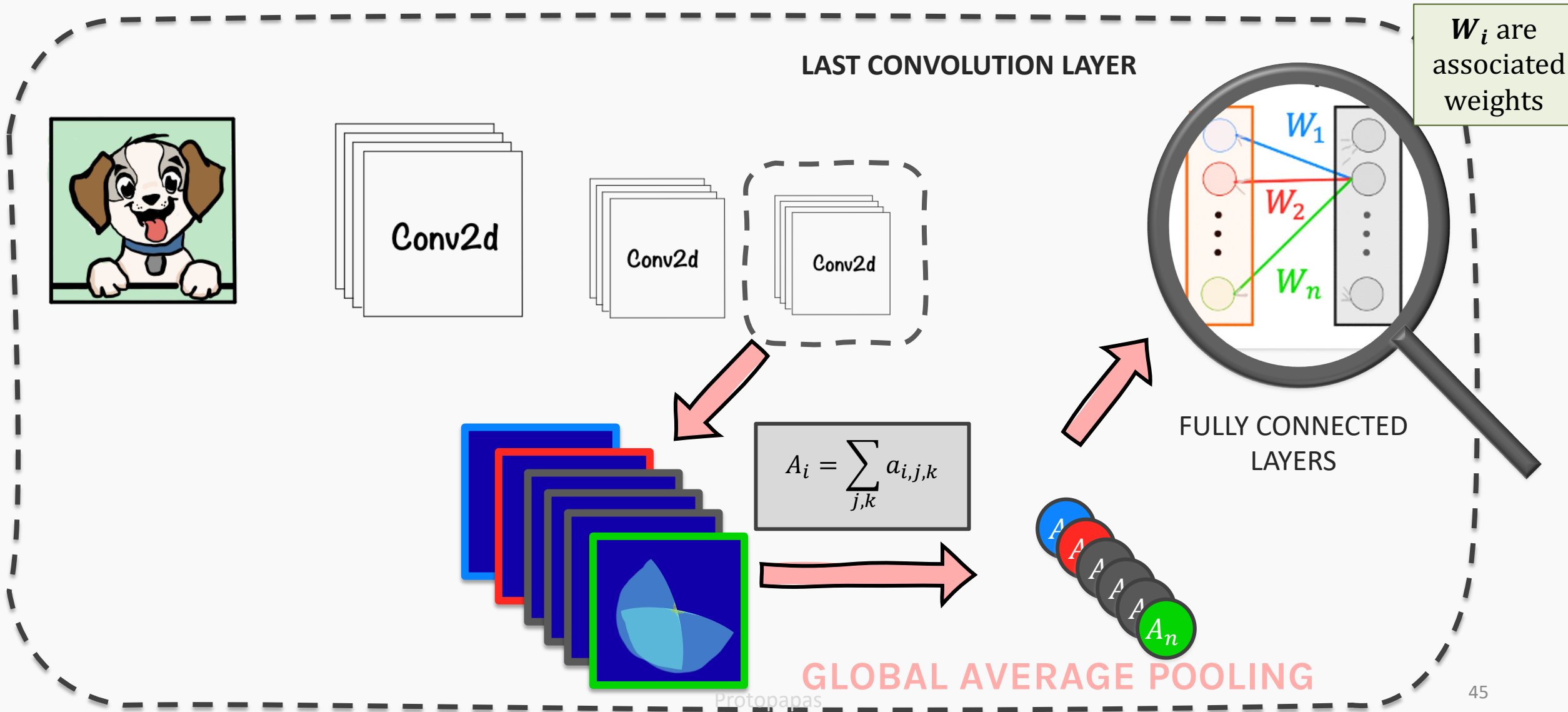


FULLY CONNECTED LAYERS

GLOBAL AVERAGE POOLING

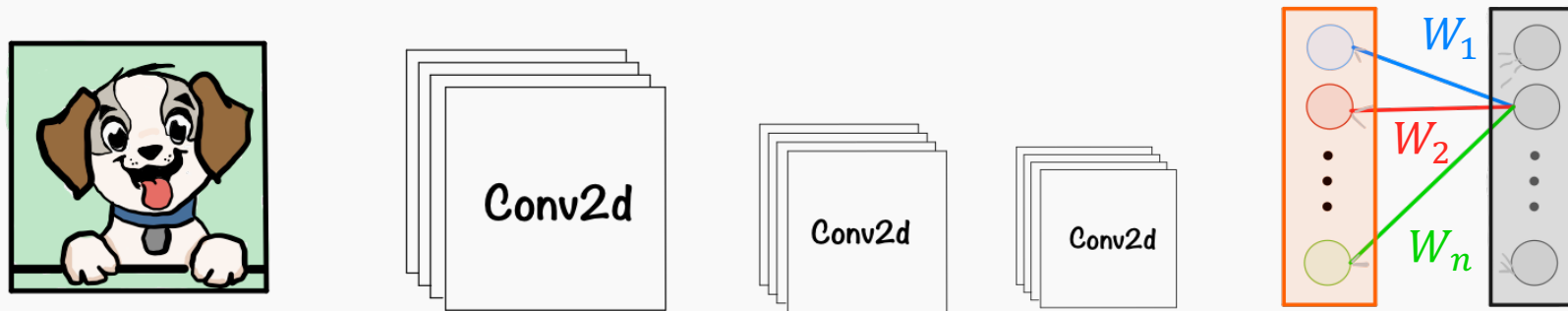
# Global Average Pooling

## HOW DO WE DO IT?



# Class Activation Mapping (CAM)

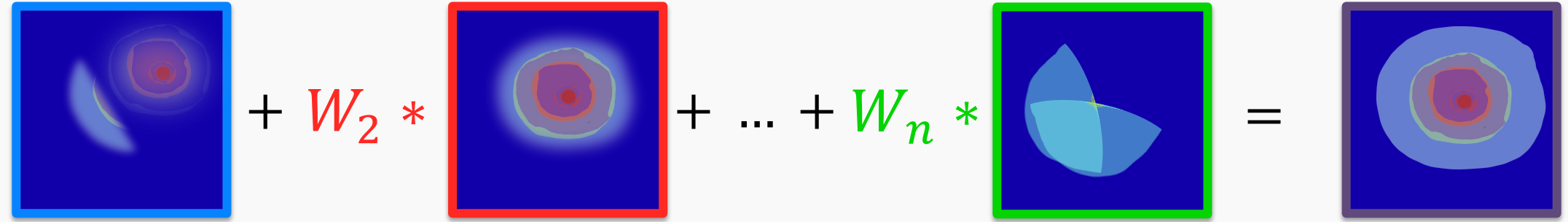
Finally, for a particular prediction, we take the weighted sum of the feature maps (where  $W_i$  comes from the GAP of the  $i^{th}$  feature map



$$W_1 * \text{Feature Map}_1 + W_2 * \text{Feature Map}_2 + \dots + W_n * \text{Feature Map}_n = \text{CAM Map}$$

**CLASS ACTIVATION MAPPING**

# Class Activation Mapping (CAM)

$$W_1 * \text{img}_1 + W_2 * \text{img}_2 + \dots + W_n * \text{img}_n = \text{output}$$


We then upsample the *weighted* feature map to the image dimensions to get the output.

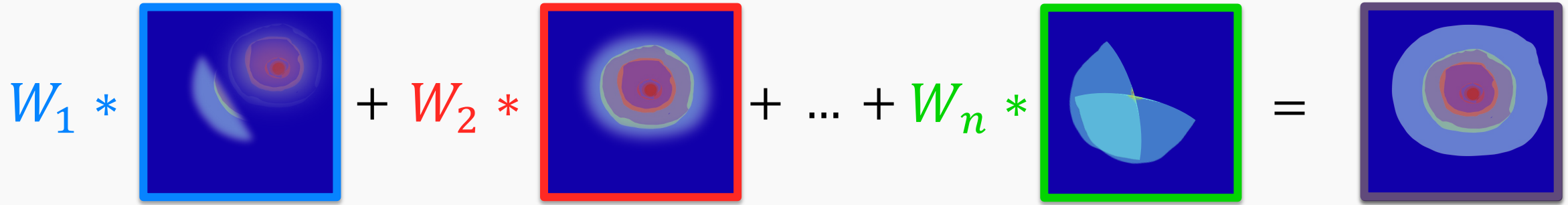
# Class Activation Mapping (CAM)

$$W_1 * \text{[Heatmap 1]} + W_2 * \text{[Heatmap 2]} + \dots + W_n * \text{[Heatmap n]} = \text{[Output Heatmap]}$$

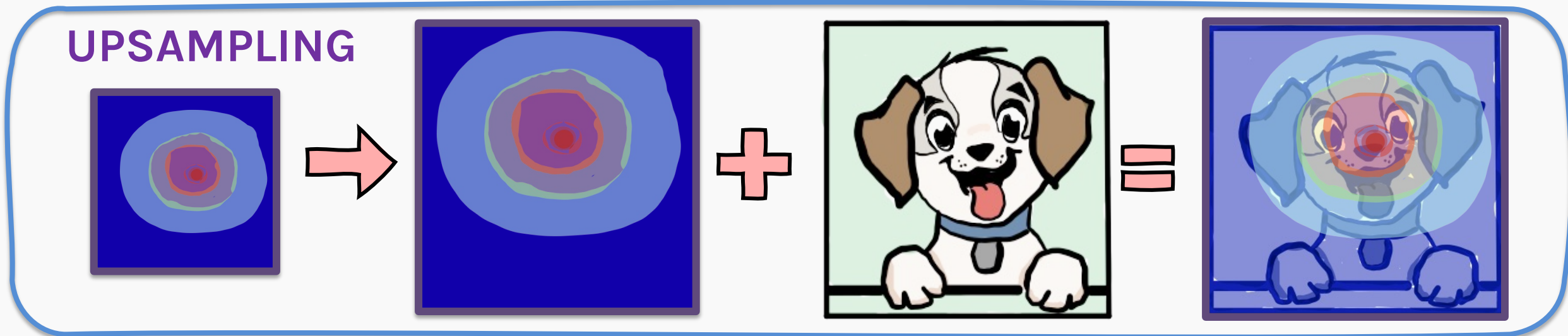
We then upsample the *weighted* feature map to the image dimensions to get the output.



# Class Activation Mapping (CAM)

$$W_1 * \text{[Heatmap 1]} + W_2 * \text{[Heatmap 2]} + \dots + W_n * \text{[Heatmap n]} = \text{[Weighted Heatmap]}$$


We then upsample the *weighted* feature map to the image dimensions to get the output.





# Outline

---

1. Gradient Base
2. Deconvolution, Guided Backpropagation Algorithm
3. Class Activation Map (CAM), **Grad-CAM** and Guided Grad-CAM

# More on Class Activation Mapping (CAM)

---

Other approaches to Class Activation Mapping have been developed by [Selvaraju et al. 2016](#):

- **Grad-CAM:** is a more versatile version of CAM that can produce visual explanations for any **arbitrary** CNN, even if the network contains a stack of fully connected layers as well (e.g. the VGG networks);
- **Guided Grad-CAM:** by adding an element-wise multiplication of guided-backpropagation visualization.

# Class Activation Mapping (CAM): **GRAD-CAM**

---

The basic idea behind Grad-CAM is the same as the basic idea behind CAM: we want to exploit the spatial information that is preserved through convolutional layers, in order to understand which parts of an input image were important for a classification decision.

# Class Activation Mapping (CAM): **GRAD-CAM**

---

The basic idea behind Grad-CAM is the same as the basic idea behind CAM: we want to exploit the spatial information that is preserved through convolutional layers, in order to understand which parts of an input image were important for a classification decision.

**Grad-CAM** is applied to a neural network that is done training; in other words, the weights of the neural network are fixed. We feed an image into the network to calculate the Grad-CAM heatmap for that image for a chosen class of interest.

# Class Activation Mapping (CAM): GRAD-CAM

---

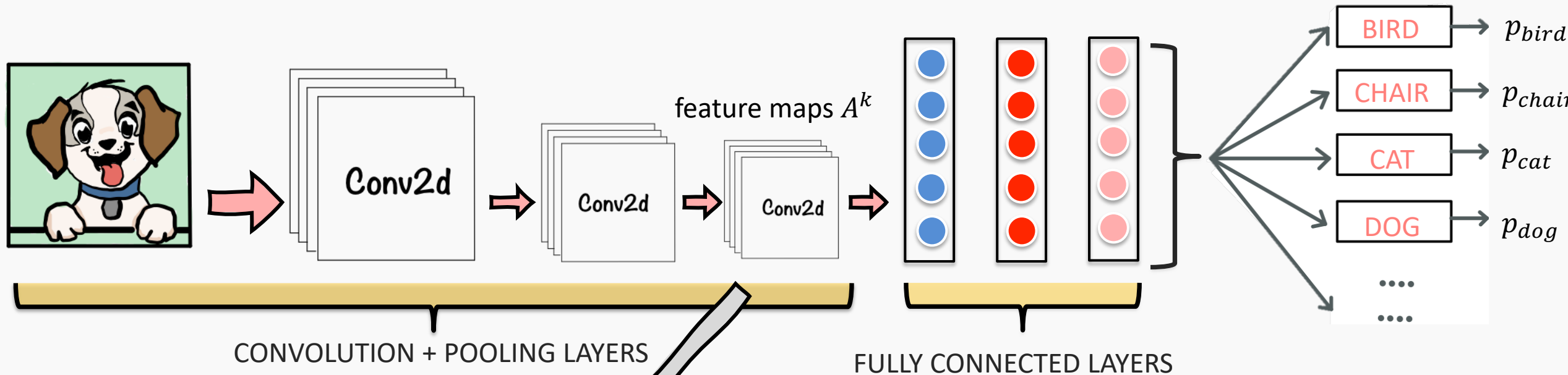
The basic idea behind Grad-CAM is the same as the basic idea behind CAM: we want to exploit the spatial information that is preserved through convolutional layers, in order to understand which parts of an input image were important for a classification decision.

**Grad-CAM** is applied to a neural network that is done training; in other words, the weights of the neural network are fixed. We feed an image into the network to calculate the Grad-CAM heatmap for that image for a chosen class of interest.

Both CAM and Grad-CAM are local backpropagation-based interpretation methods. They are model-specific as they can be used exclusively for interpretation of convolutional neural networks.



# Class Activation Mapping (CAM): GRAD-CAM



$$\alpha_k^c = \frac{1}{Z} \sum_{i,j} \frac{\partial y^c}{\partial A_{ij}^k}$$

Instead of the Global Average Pooling layer, we compute the derivative of the logit with respect to the feature maps using the equation on the left



# Class Activation Mapping (CAM): GRAD-CAM

First, the gradient of the logits,  $y^c$ , of the class  $c$  w.r.t the activations maps of the final convolutional layer is computed and then the gradients are averaged across each feature map to give us an importance score.

$$\alpha_k^c = \frac{1}{Z} \sum_{i,j} \frac{\partial y^c}{\partial A_{ij}^k}$$

$\alpha_k^c$  is the importance of feature map  $k$  for the class  $c$

$Z$  is some normalization

Summing over all elements of the  $k$  activation map (aka **global average pooling**)

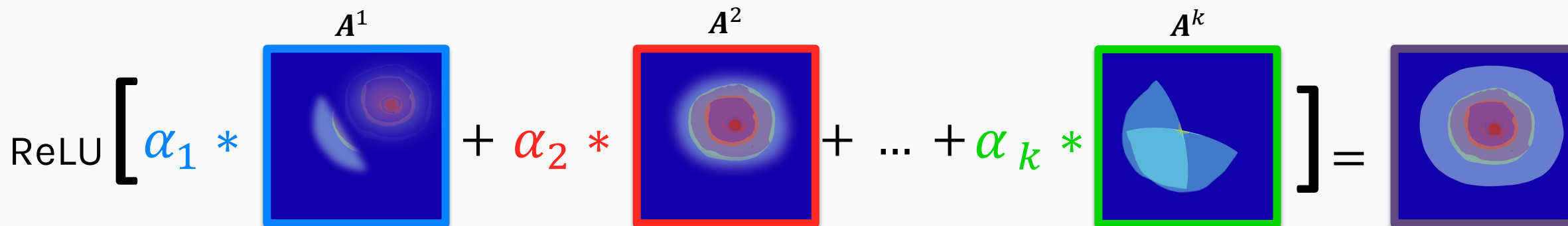
$y^c$  is the logit for class  $c$

$A_{ij}^k$  is the  $i,j$  element of the  $k$  activation map of the last convolutional activation layer

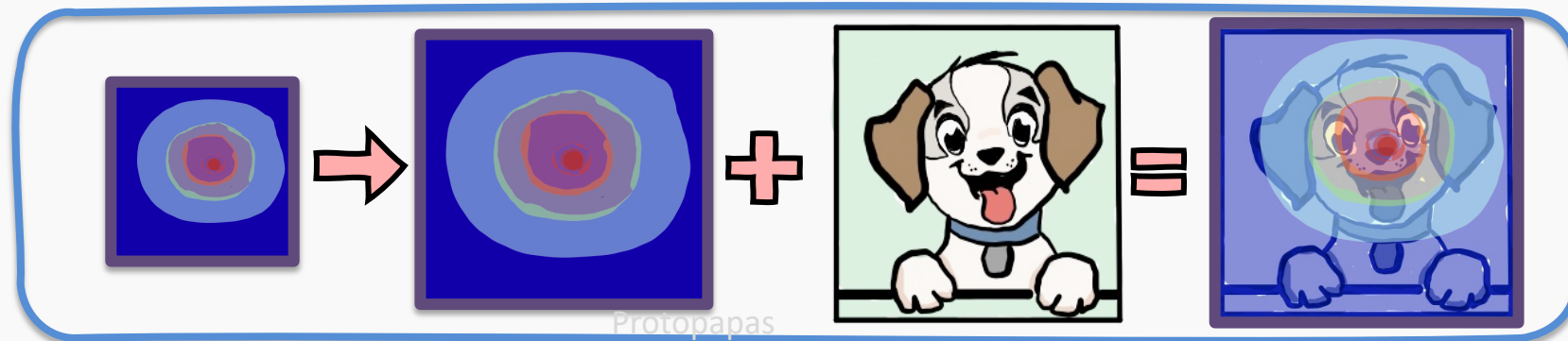
# Class Activation Mapping (CAM): GRAD-CAM

Combine the feature maps as we did in the CAM before except that here, we use the  $\alpha$ 's instead of the  $W$  and we also activate the resulting weighted sum.

$$S_{Grad-CAM}^c = \text{ReLU} \left( \sum_k \alpha_k^c A^k \right)$$



UPSAMPLING



# Outline

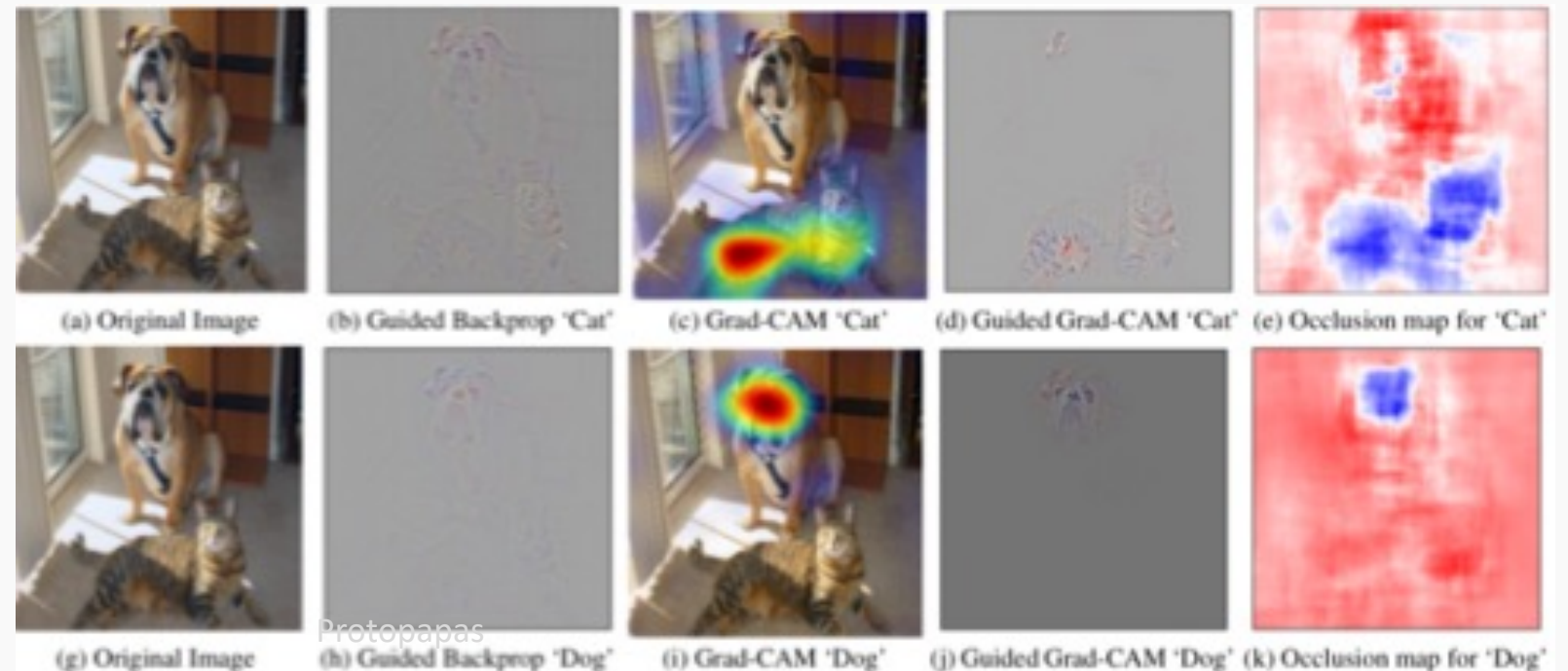
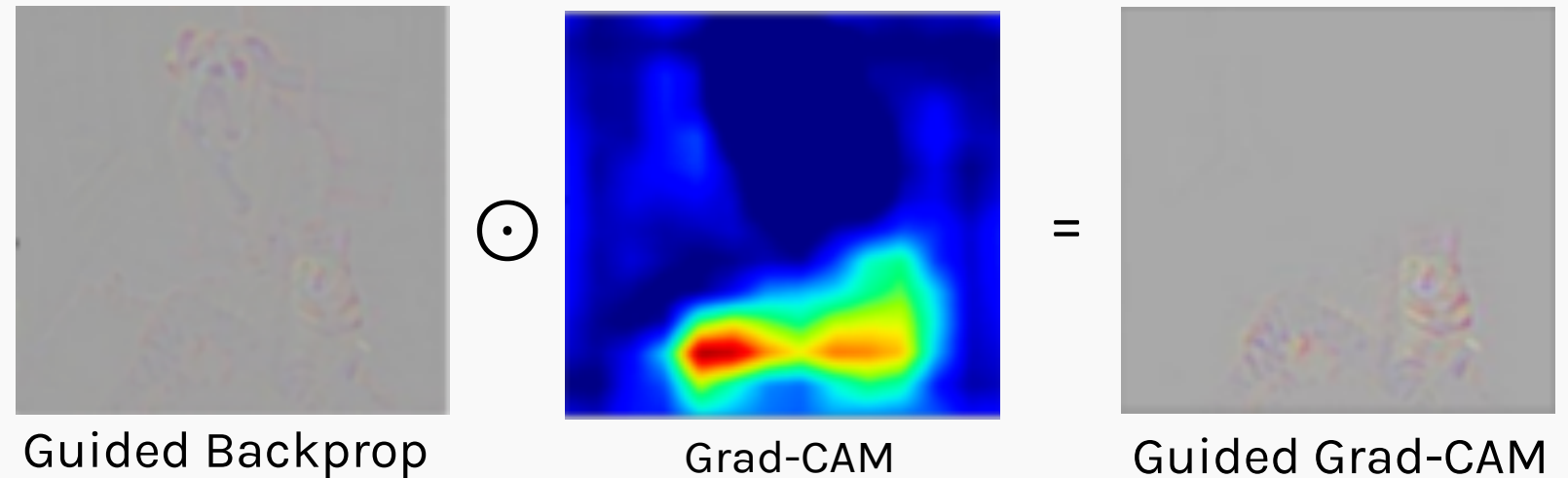
---

1. Gradient Base
2. Deconvolution, Guided Backpropagation Algorithm
3. Class Activation Map (CAM), Grad-CAM and **Guided Grad-CAM**

# Class Activation Mapping (CAM): Guided GRAD-CAM

Grad-CAM can only produce coarse-grained visualizations.

Guided Grad-CAM combines Guided-Backpropagation with Grad-CAM by simply perform an **element-wise** multiplication of Guided-Backpropagation with Grad-CAM.



# Saliency Maps: **Limitations**

Saliency maps are interpretable techniques to investigate hidden layers in CNNs. They are based on a **local gradient-based backpropagation interpretation method**, and it could be used for any arbitrary artificial neural network (**model-agnostic**).

However, some **limitations** of the methods has been raised because:

- Saliency maps are not always **reliable**. Indeed, subtracting the mean and normalizations, can make undesirable changes in saliency maps as shown by [Kindermans et al. 2018](#);
- Saliency maps are vulnerable to adversarial attacks [Ghorbani et al. 2019](#).
- [Adebayo et al. 2018](#) tested many saliency maps techniques and found that Grad-CAM and gradient base are the most reliable.

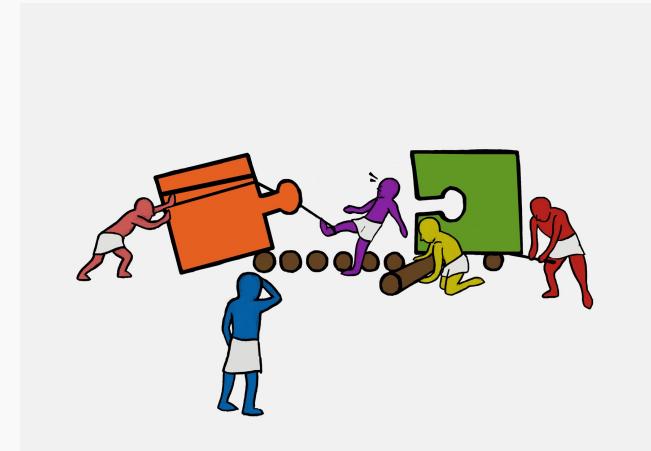
## Exercise:

The goal of this exercise is to build a saliency map using Grad-CAM.

Your final image may resemble the one on the right.

An important skill to learn from this exercise is how to use `tf.GradientTape()` to find the gradients of the output with respect to the activations.

Knowing how to use `GradientTape()` is like having the key to the kingdom of DeepLearning.



Predicted class: tabby, tabby cat

