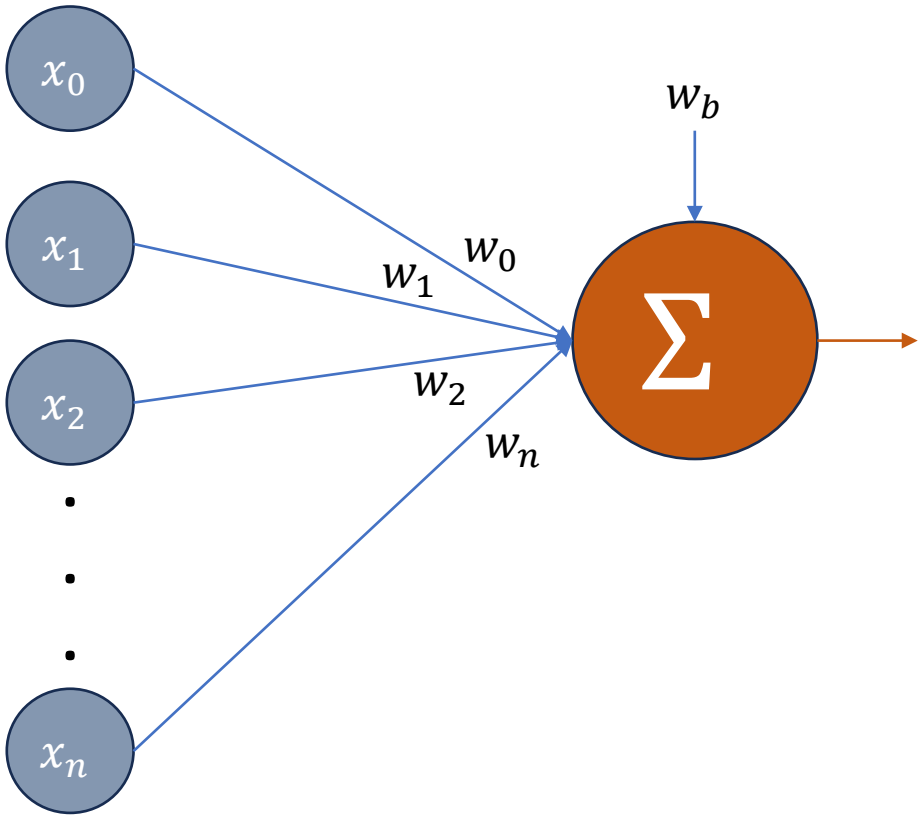
CNNs

Chaz Allegra

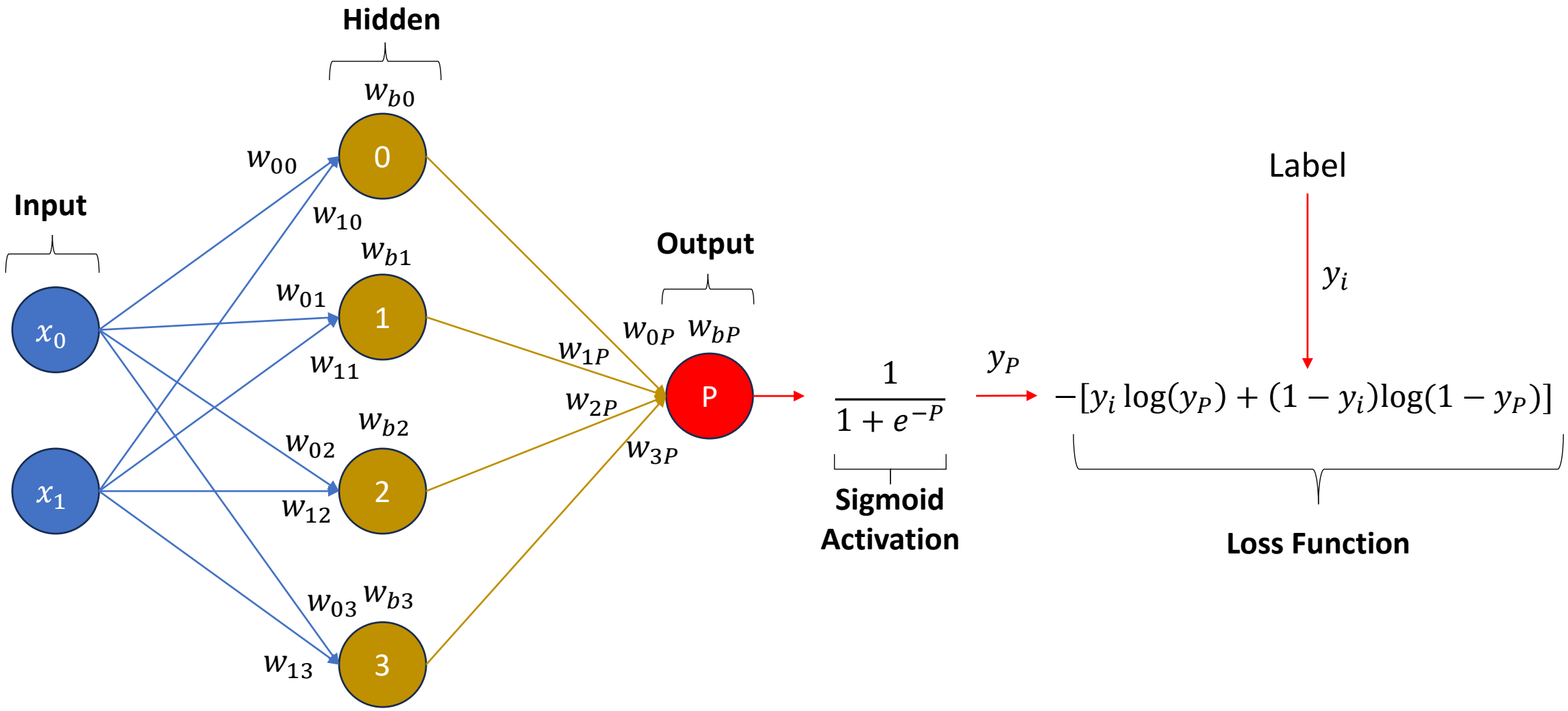
Review

Review: Neuron

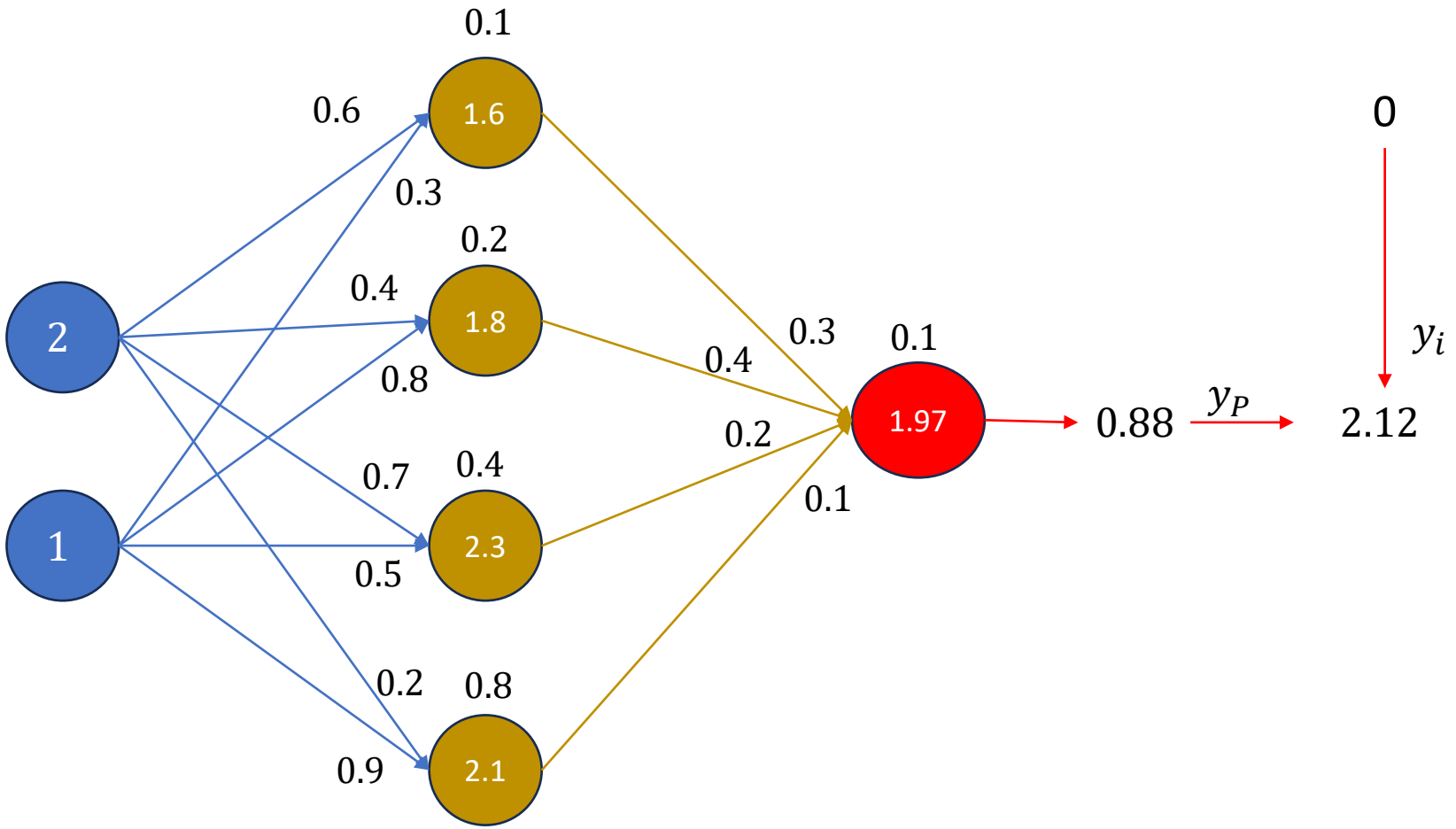


$$\Phi(w_b + \sum_{i=0}^{n} w_i x_i)$$

Review: Fully Connected

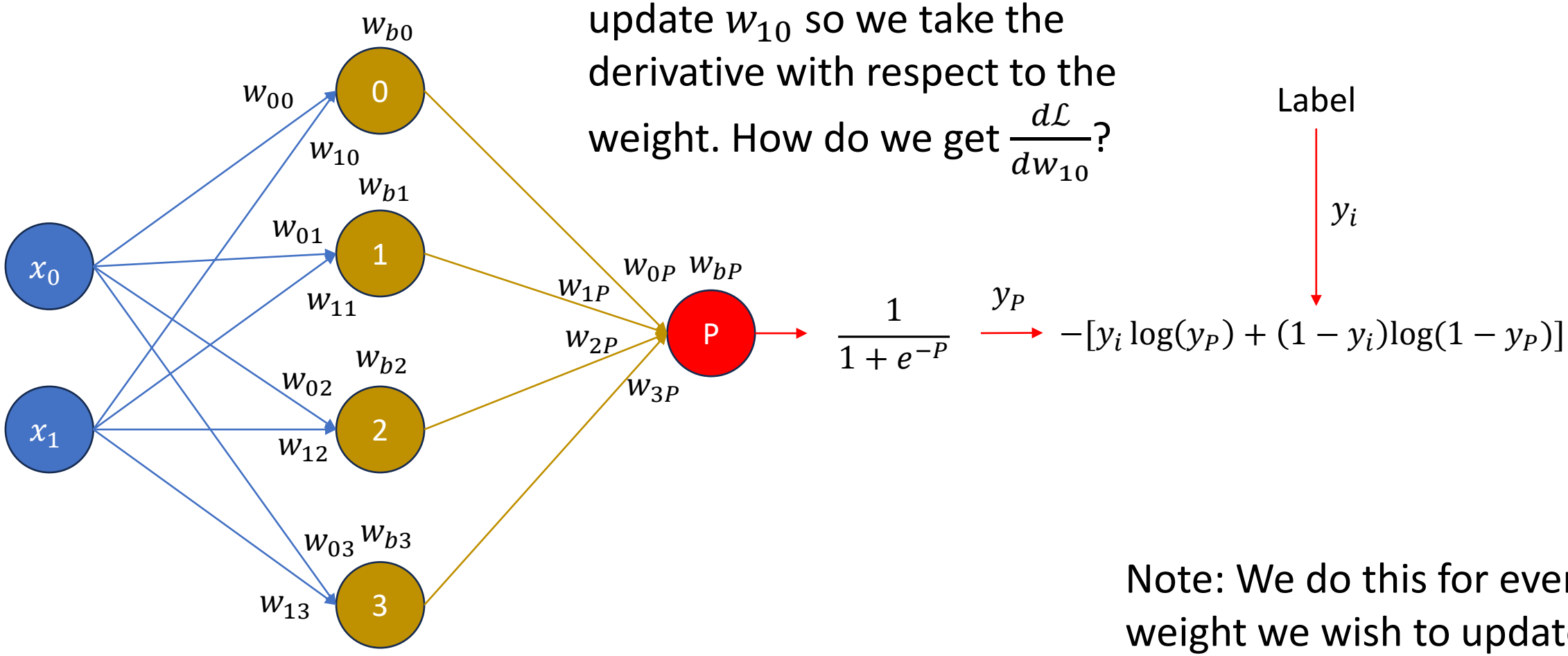


Review: Forward Propagation



Review: Back Propagation

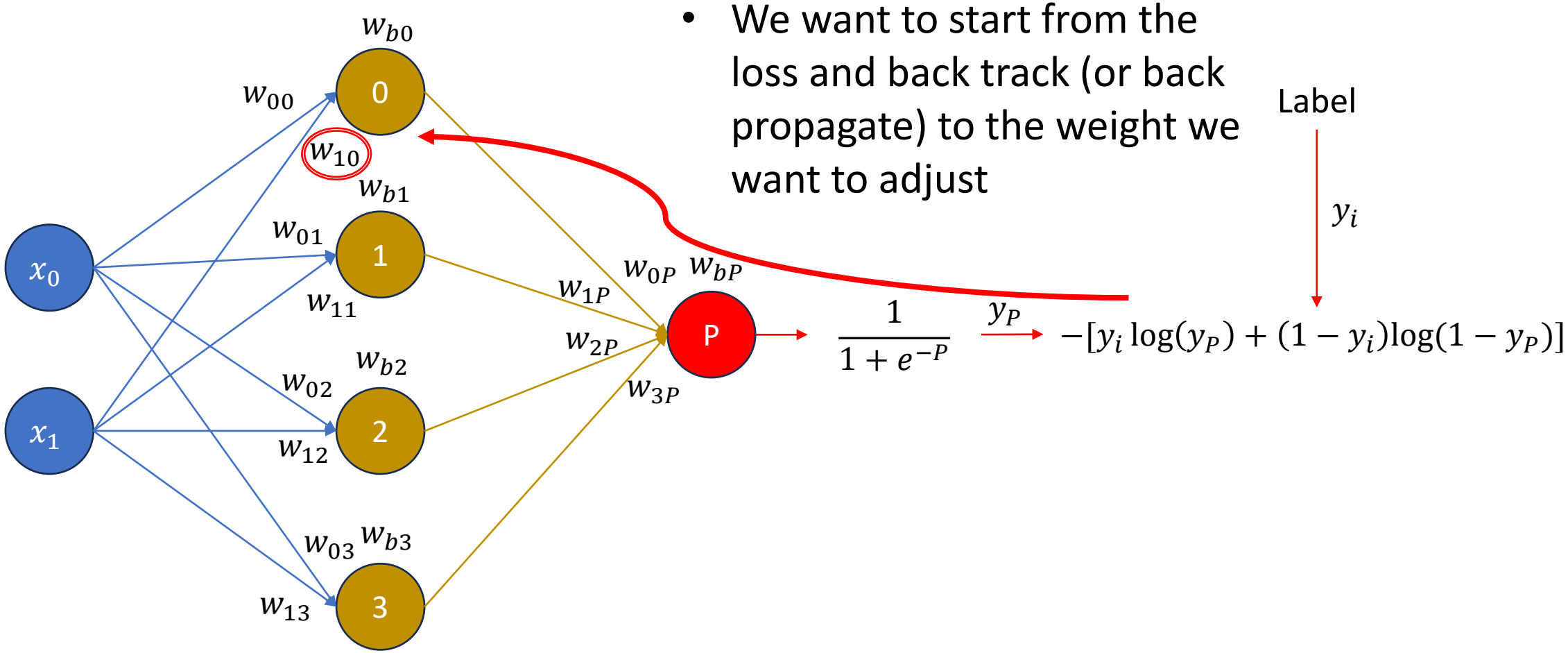
We want to use the loss to update w_{10} so we take the derivative with respect to the weight. How do we get $\frac{d\mathcal{L}}{dw_{10}}$?



Note: We do this for every weight we wish to update

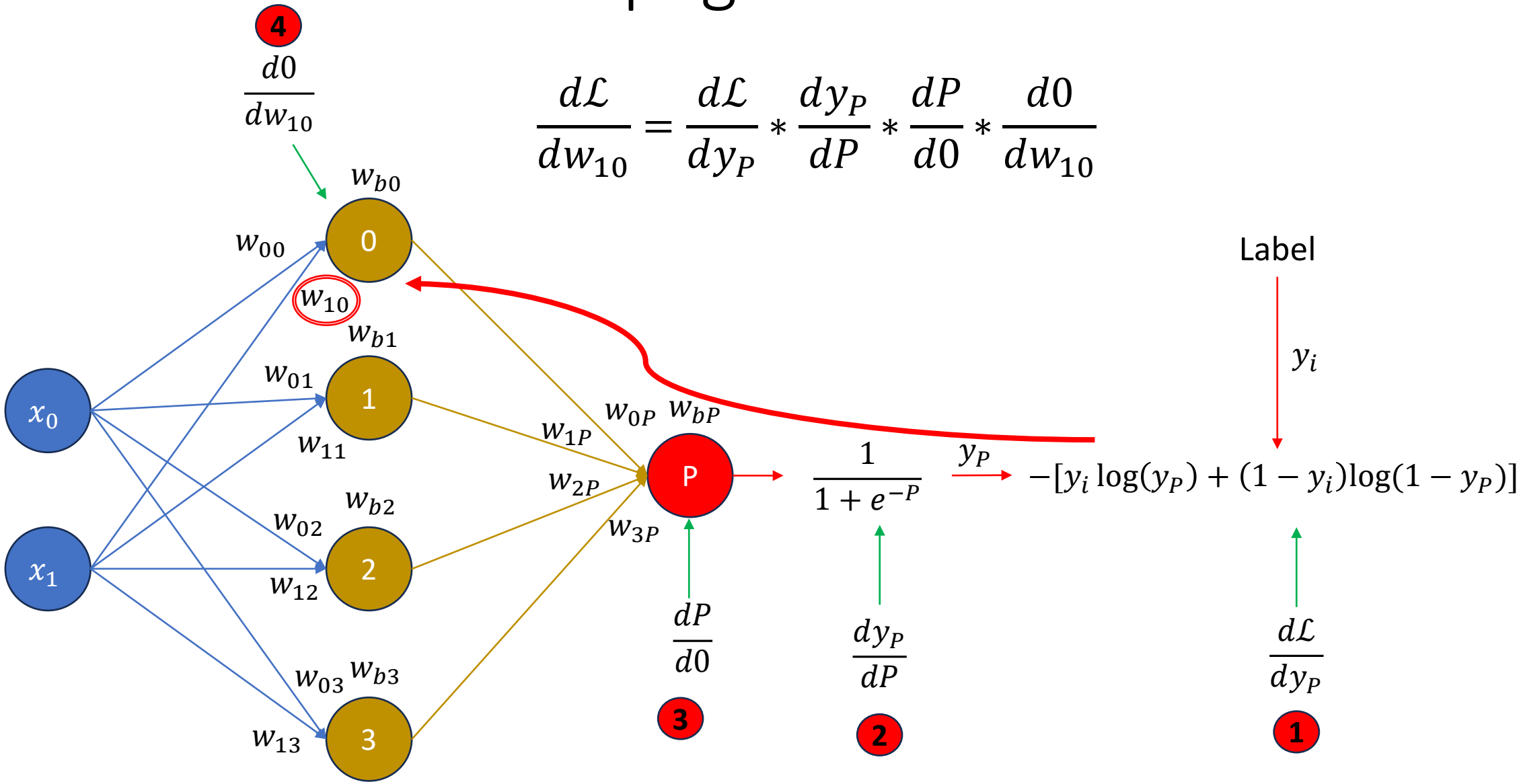
Review: Back Propagation

- The answer is the good old chain rule
- We want to start from the loss and back track (or back propagate) to the weight we want to adjust



Review: Back Propagation

$$\frac{d\mathcal{L}}{dw_{10}} = \frac{d\mathcal{L}}{dy_P} * \frac{dy_P}{dP} * \frac{dP}{d0} * \frac{d0}{dw_{10}}$$



Review: Back Propagation

- Take Derivatives first then plug in forward pass values (derivatives are always the same)
- $\frac{d\mathcal{L}}{dw_{10}} = \frac{d\mathcal{L}}{dy_P} * \frac{dy_P}{dP} * \frac{dP}{d0} * \frac{d0}{dw_{10}}$
- $\frac{d\mathcal{L}}{dy_P} = -\frac{y_i - y_P}{(1-y_P)y_P}$ = 8.33
- $\frac{dy_P}{dP} = \frac{e^{-P}}{(1+e^{-P})^2}$ = 0.107
- $\frac{dP}{d0} = w_{0P}$ = 0.3
- $\frac{d0}{dw_{10}} = x_0$ = 2
- $\frac{d\mathcal{L}}{dw_{10}}$ = 0.535

- To then update our weight, we subtract our current weight by our gradient multiplied by learning rate:

$$w_{new} = w - \eta * Gradient$$

CNN

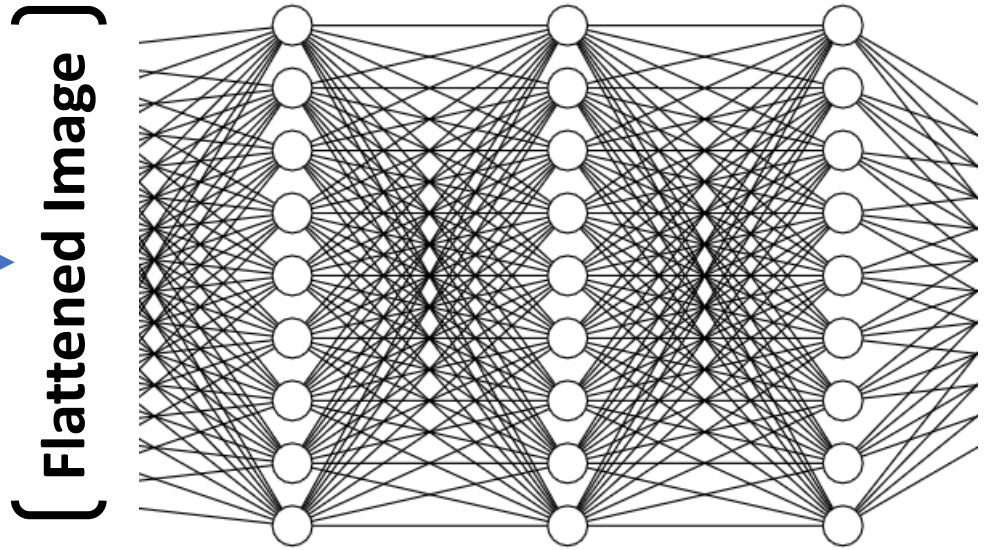
“CNNs learn to encode an image in such a way that it captures as much relevant information as possible to make a seemingly intelligent decision”

What are CNNs used for?

- CNNs tend to shine when it comes to processing images, although they are not limited to images and may be utilized in any problem with spatial structure

Why Not FCs for Image Processing?

- Flattening loses spatial structure of image
- They are FULLY connected so every single pixel of all channels are connected to every neuron. This means A LOT of parameters

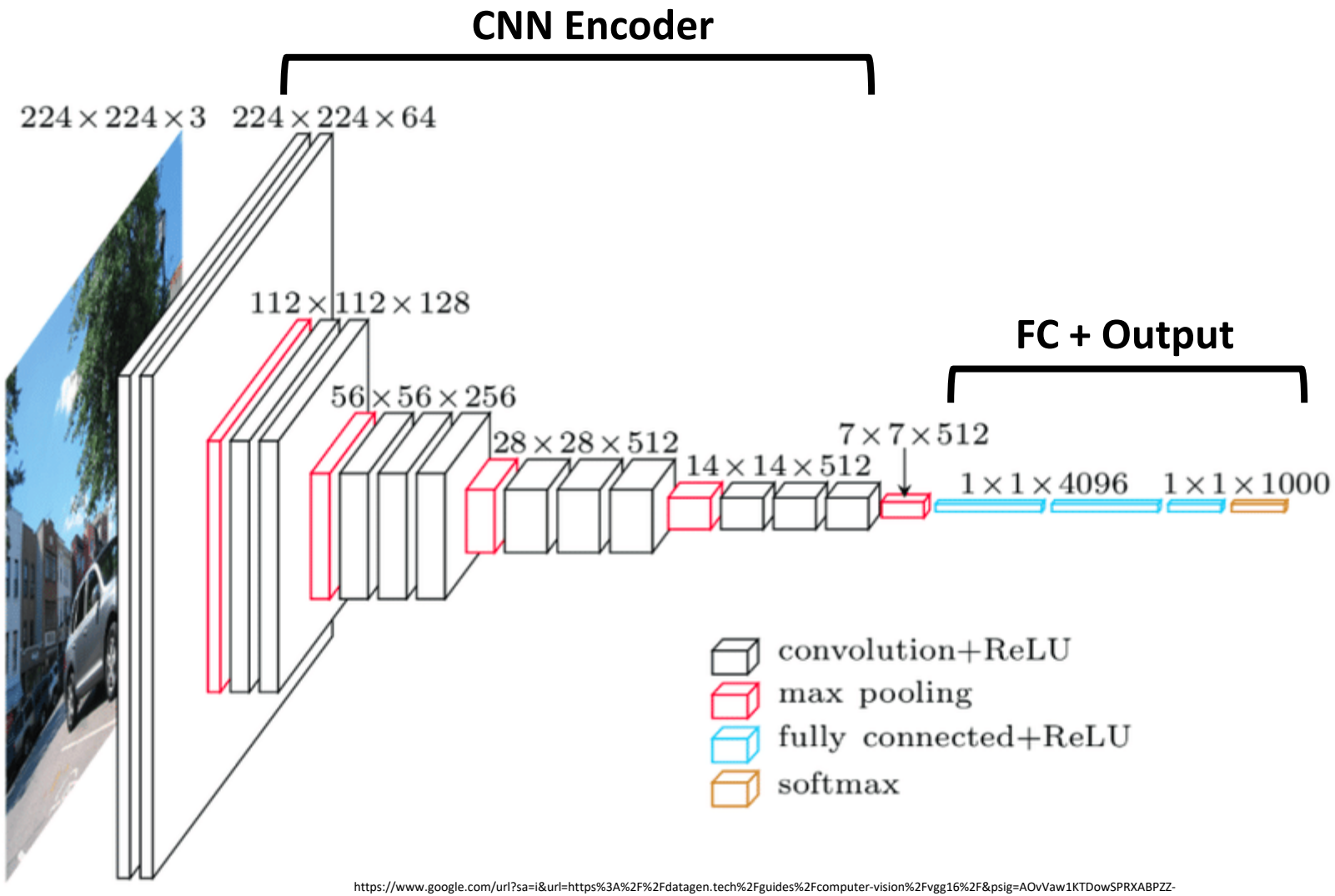


https://www.google.com/url?sa=i&url=https%3A%2F%2Fmath.stackexchange.com%2Fquestions%2F2048722%2Fa-name-for-layered-directed-graph-as-in-a-fully-connected-neural-network&psig=AOvVaw1G48-PCbwjheDQ-BOkOSC2&ust=1699727889555000&source=images&cd=vfe&opi=89978449&ved=0CBIQjRxqFwoTCKC03PmJuoIDFQAAAAAdAAAAABAE

CNN Structure

- The Convolutional Neural Network is a model that utilizes convolution layers to extract features from an image (or any input where spatial structure is important) by moving various learned filters across an image
- We usually encode the image by using stride and pooling to shrink the image along the spatial dimensions (height and width) while increasing size of the channel dimension
- These layers typically start by extracting basic features like horizontal and vertical lines at the beginning of the CNN and then end with more complicated features like wheels or eyes
- After the image has been processed, we flatten and utilize FC layer(s) to make a decision based on our learned features

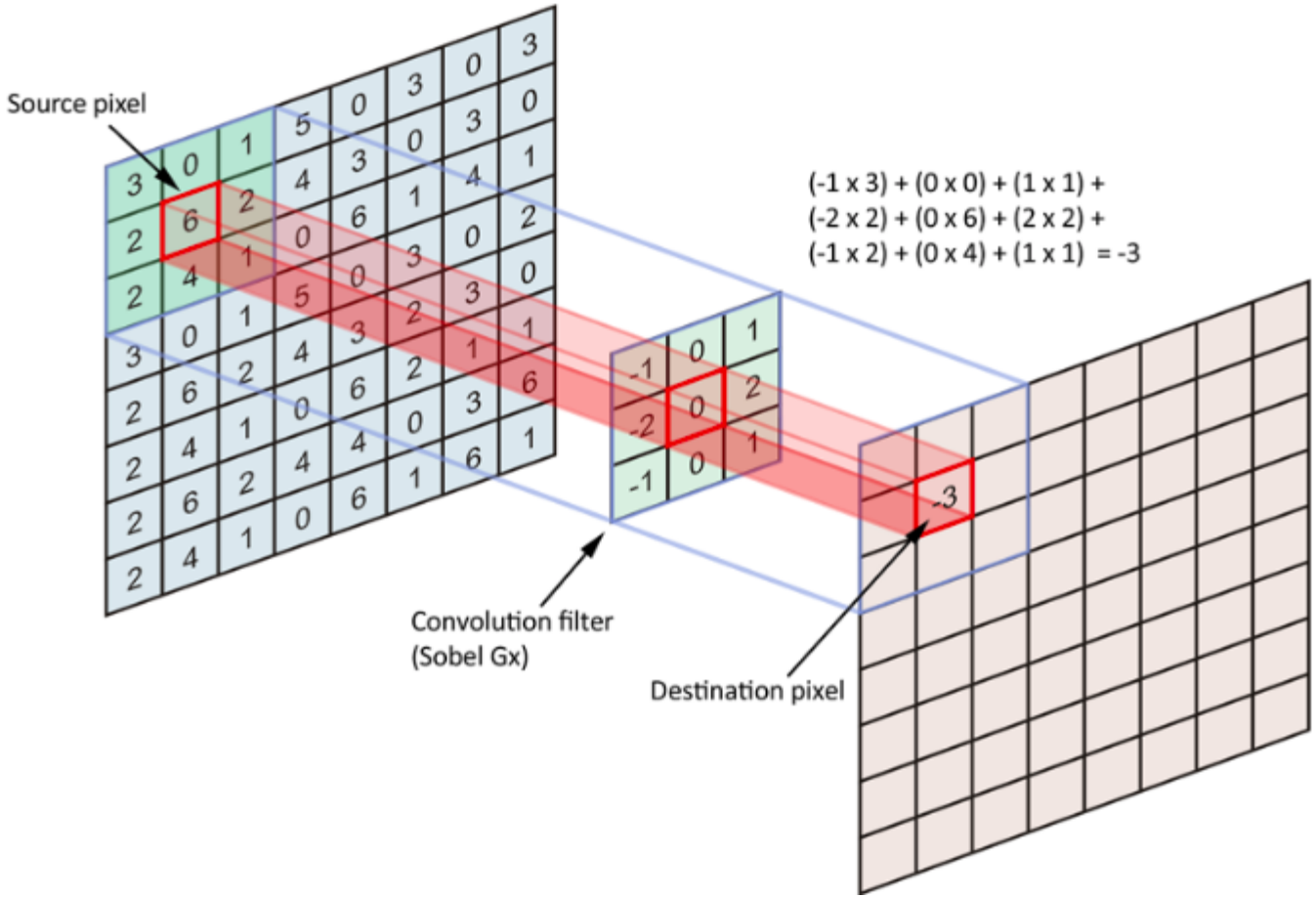
CNN Structure



https://www.google.com/url?sa=i&url=https%3A%2F%2Fdatagen.tech%2Fguides%2Fcomputer-vision%2Fvgg16%2F&psig=AOvVaw1KTDowSPRXABPZZ-sE4bnY&ust=1699727937328000&source=images&cd=vfe&opi=89978449&ved=0CBIQjRxqFwoTCLCQvZCKuoIDFQAAAAAdAAAAABAE

Kernel

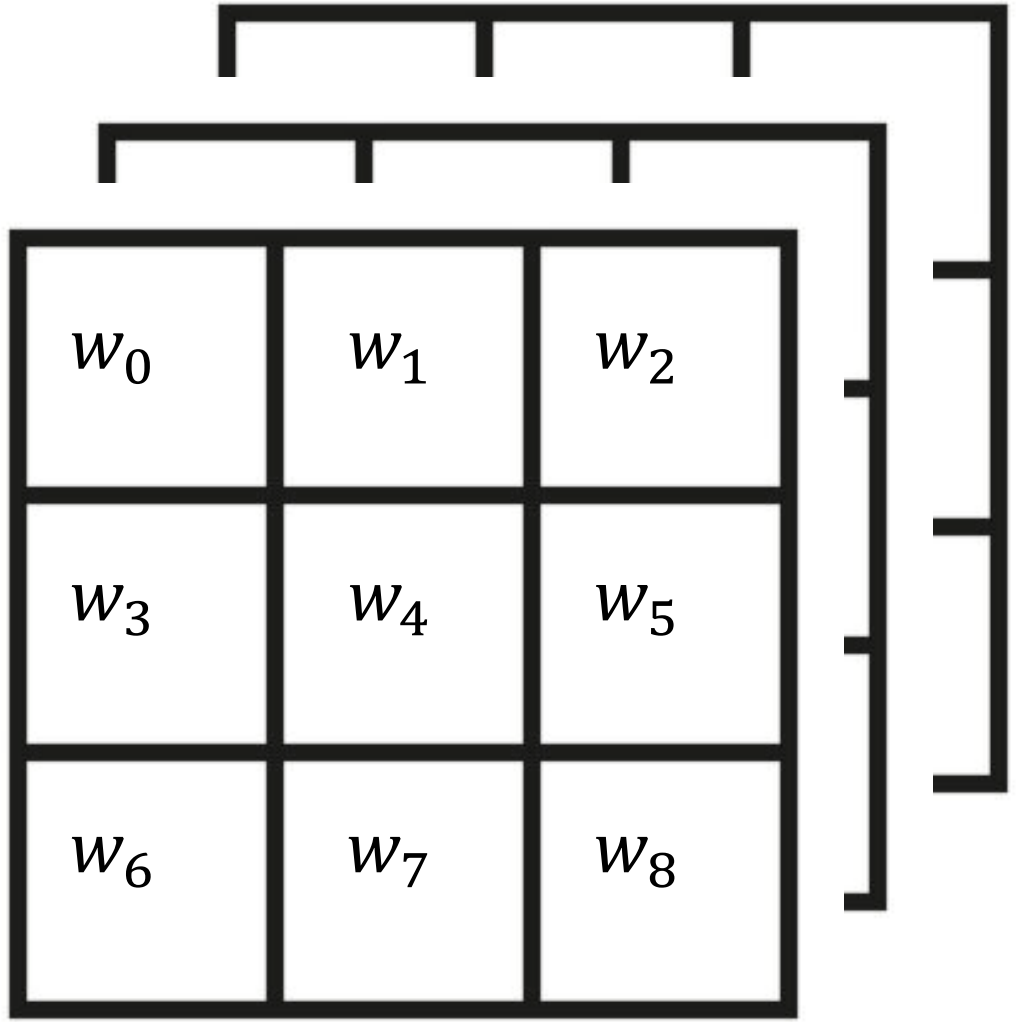
- The kernel is a filter that moves along an image
- The convolution operation is a dot product between the input image and the filter
- The output of the kernel is known as feature or activation map. Each kernel produces an activation map
- The filter's values are learned during training to capture the most relevant features



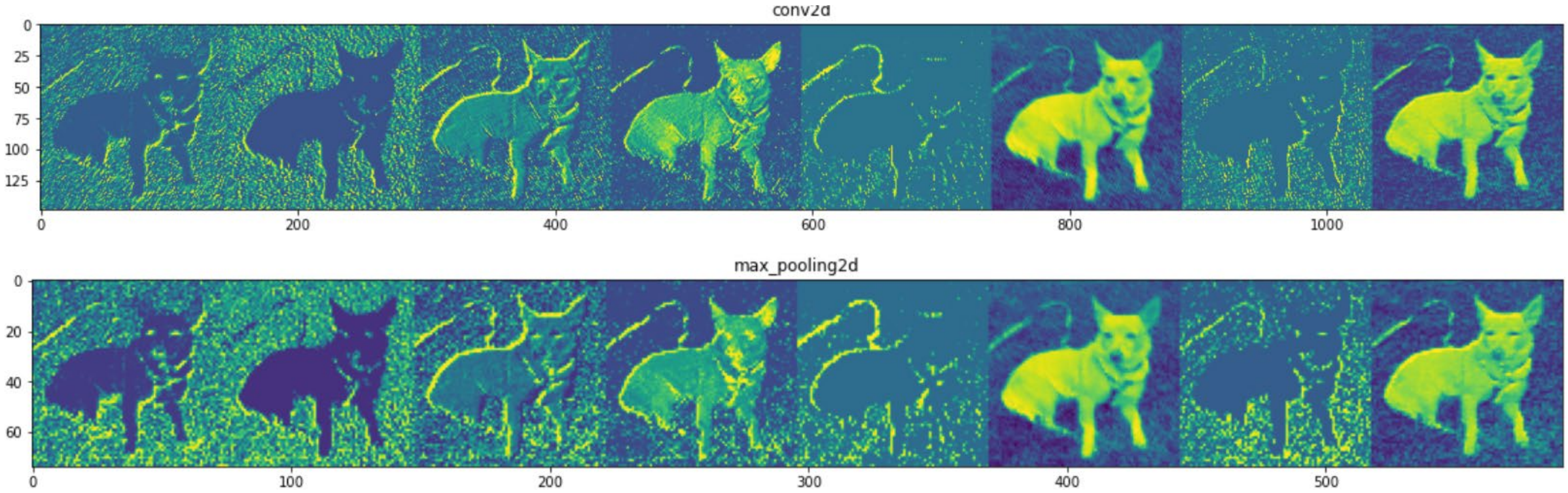
https://www.google.com/url?sa=i&url=https%3A%2F%2Fviniciuscantocosta.medium.com%2Funderstanding-the-structure-of-a-cnn-b220148e2ac4&psig=AOvVaw3ObpMOX7NxyEcYVJKz86n3&ust=1699727969252000&source=images&cd=vfe&opi=89978449&ved=0CBIQjRxqFwoTCOjM15-KuoIDFQAAAAAdAAAAABAE

Kernel Weights

- During Training we optimize the kernel in such a way that we capture relevant features
- The kernel has a learnable parameter for each spatial patch for each channel, and there is a bias weight.
- Kernel parameters = Height x Width x Channels + 1



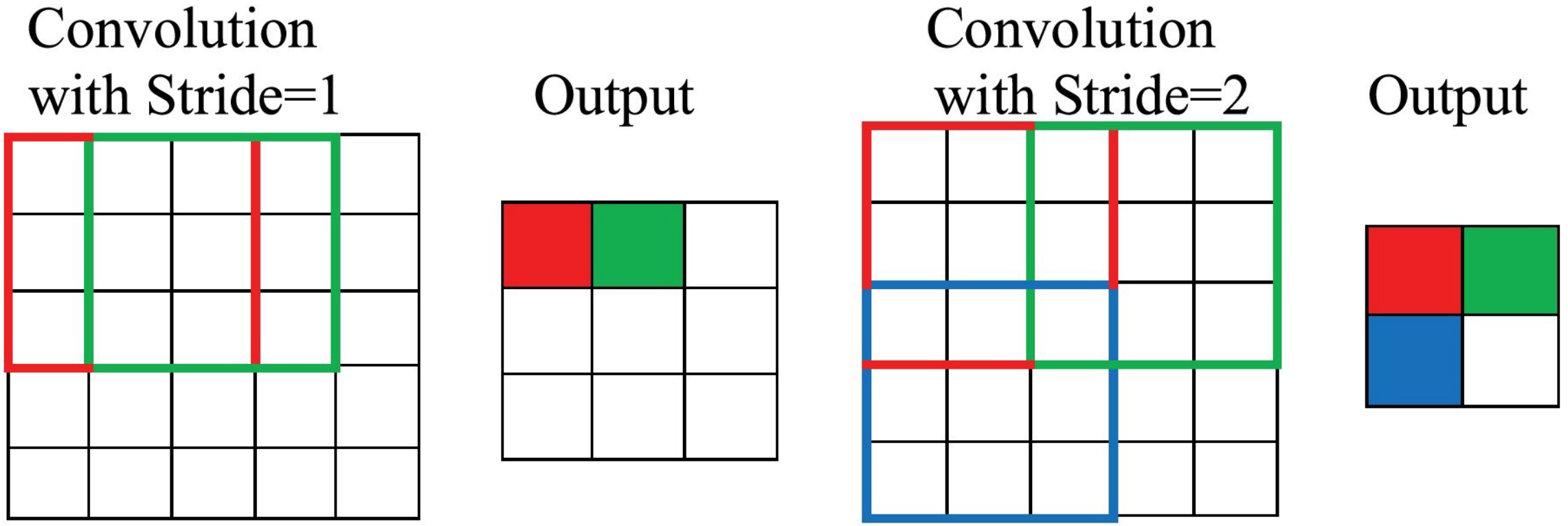
Feature Maps



https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.analyticsvidhya.com%2Fblog%2F2020%2F11%2Ftutorial-how-to-visualize-feature-maps-directly-from-cnn-layers%2F&psig=AOvVaw29tSTwhU6wfoGwiKSxx1Dw&ust=1699728194106000&source=images&cd=vfe&opi=89978449&ved=0CBIQjRxqFwoTCMCJjIuLuoIDFQAAAAAdAAAAABAE

Stride

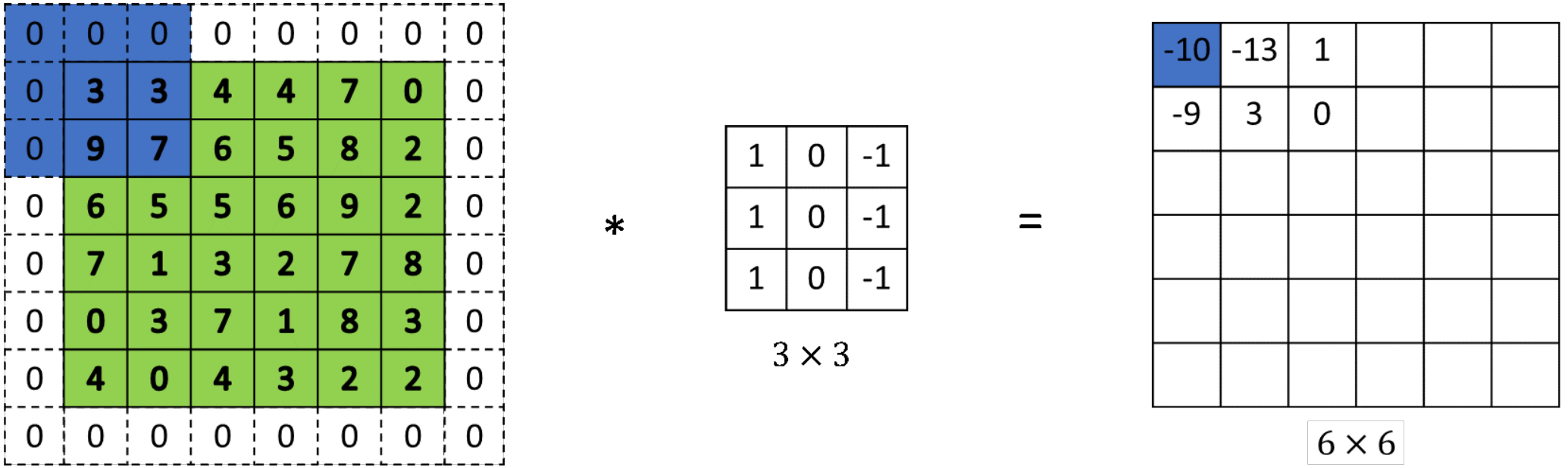
- Stride is how large of a step the kernel takes. Strides larger than 1 will reduce spatial dimensionality.



https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.analyticsvidhya.com%2Fblog%2F2022%2F03%2Fbasics-of-cnn-in-deep-learning%2F&psig=AOvVaw1mUSQ2J5aj99Fvc_ywvPGq&ust=1699728016136000&source=images&cd=vfe&opi=89978449&ved=0CBIQjRxqFwoTCKiUkLaKuoIDFQAAAAAdAAAAABAE

Padding

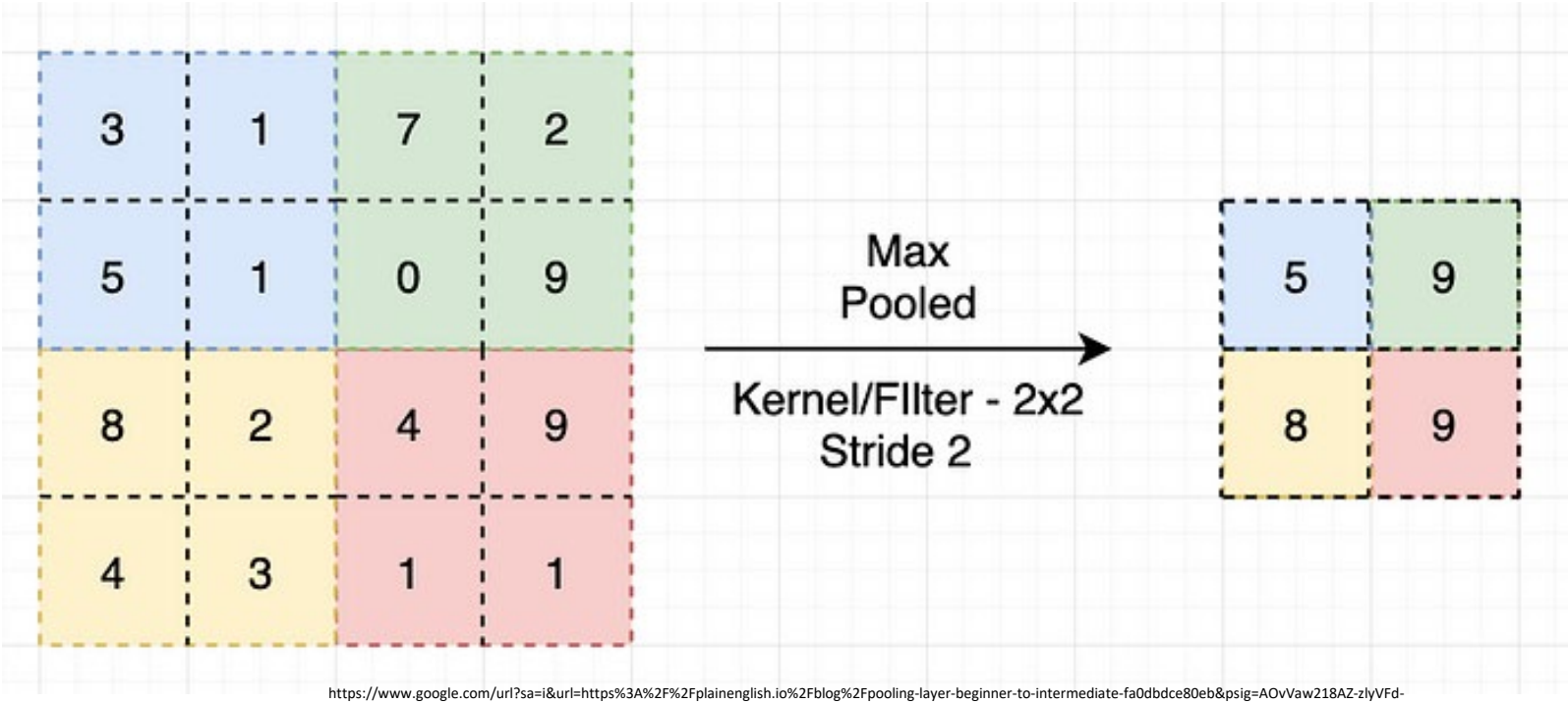
- Padding is used to maintain dimensionality. Adding zeros around the image ensures the output is the same spatial size as the input



https://www.google.com/url?sa=i&url=https%3A%2F%2Fdatahacker.rs%2Fwhat-is-padding-cnn%2F&psig=AOvVaw2n7ca73EcVvjA_Mvzy-dzr&ust=1699728119033000&source=images&cd=vfe&opi=89978449&ved=0CBIQjRxqFwoTCMDZiueKuoIDFQAAAAAdAAAAABAE

Pooling

- Pooling is another way to reduce special dimensionality. The most common types are max and average pooling.



https://www.google.com/url?sa=i&url=https%3A%2F%2Fplainenglish.io%2Fblog%2Fpooling-layer-beginner-to-intermediate-fa0dbdce80eb&psig=AOvVaw218AZ-zlyVFd-z9vTJWycU&ust=1699728147356000&source=images&cd=vfe&opi=89978449&ved=0CBIQjRxqFwoTCNjm2fSKuoIDFQAAAAAdAAAAABAE

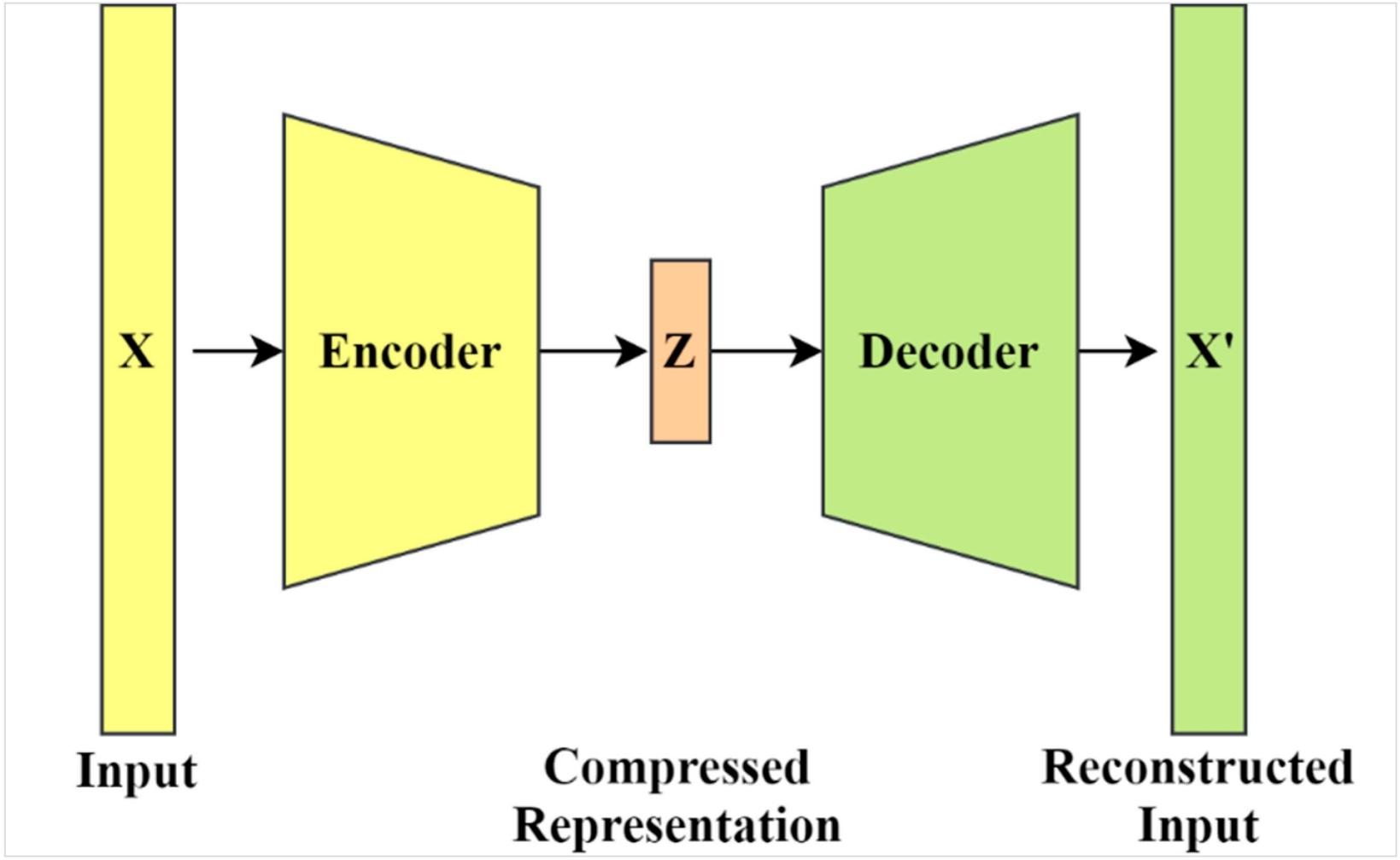
Blocks

- Blocks are groups of layers that make up a model
- You will see this often in large networks. For simplicity they may define a block and then use this block often
- When coding it is also useful to create blocks for groups of layers you will reuse
- A block can simply be two convolution layers

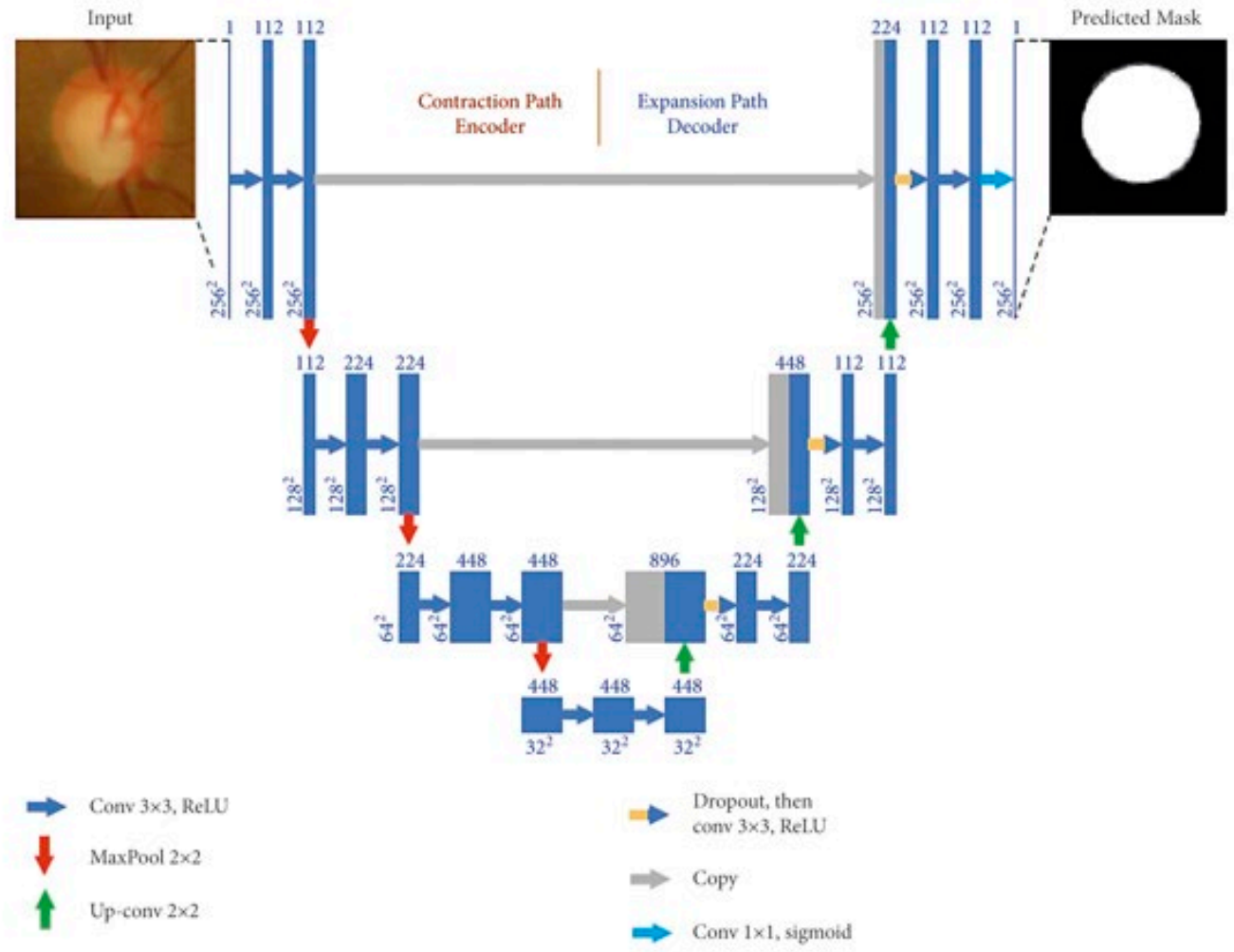
Transfer Learning

- We are often limited by hardware and resources, because of this it is useful to use models that have been pre-trained on a very large dataset and fine tune it for our purposes
- Pre-trained CNNs already learned to extract relevant features so training on a smaller dataset becomes much easier
- An example of this is to take ResNet-50, which was trained on ImageNet (1.7M images), remove the output layer and add our own for our given problem, and then train it on our dataset. You may want to lower the learning rate for pre-trained models to prevent its parameters from changing too drastically

Brief Mention: AutoEncoders



Brief Mention: U-Nets



Brief Mention: Models for Future Research

- Image Generation
  - Variational AutoEncoders
  - GANs
  - Diffusion Models
- Object Detection
  - Regional-CNNs
  - Single-Step Approaches (YOLO)
- Control Systems, Games, and Much More
  - Reinforcement Learning
- Temporal-Tasks
  - RNNs (LSTMs)
- Natural Language and All of the Above (They do it all)
  - Transformers