Intro to Neural Nets

Session 4: Image Data (CNNs)

Session Agenda

Convolutional Neural Networks (CNNs)

- What CNNs try to accomplish
- What is a convolution?
    - Padding, strides, filters
- What is pooling?
    - Max, min, avg pooling.

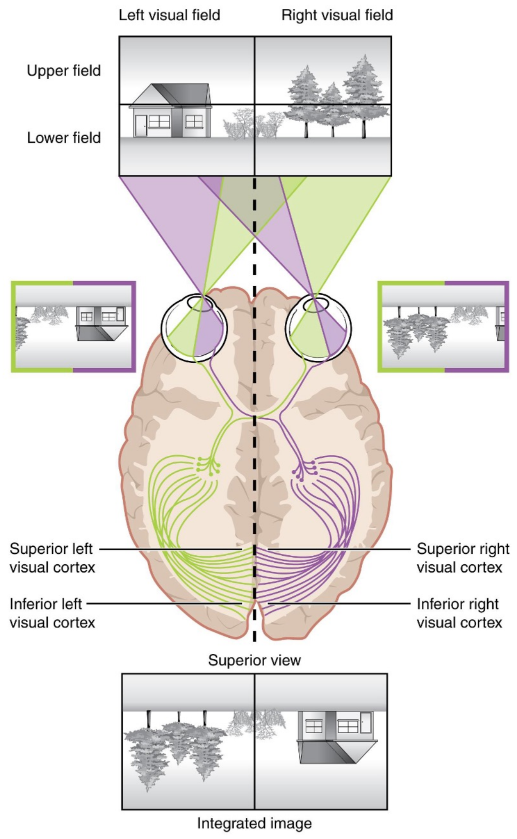
Other Stuff

- CNN specific techniques to avoid overfitting (data augmentation).
- Extracting feature representations from your trained model.
- Adapting pre-trained models (transfer learning).

Inspiration for Convnets

Our Visual System

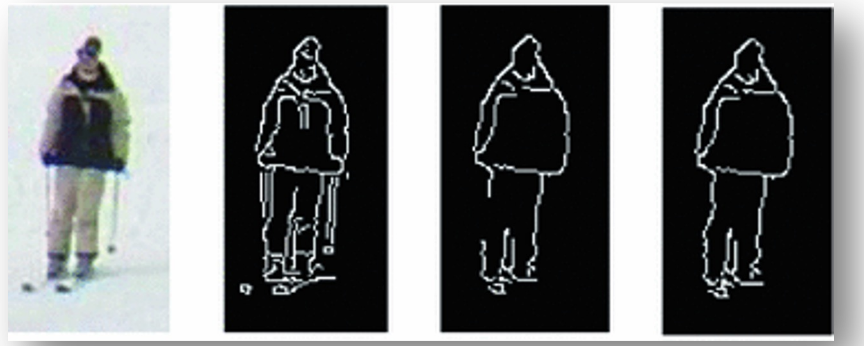
- Human eye is basically a 576-megapixel video camera.
- For comparison, the Pixel 6 camera is 50-megapixels.
- The human field of vision is not a square; something like a video camera that records individual image frames comprised of 24,000 x 24,000 pixels.


Left visual field
Right visual field
Upper field
Lower field
Superior left visual cortex
Superior right visual cortex
Inferior left visual cortex
Inferior right visual cortex
Superior view
Integrated image

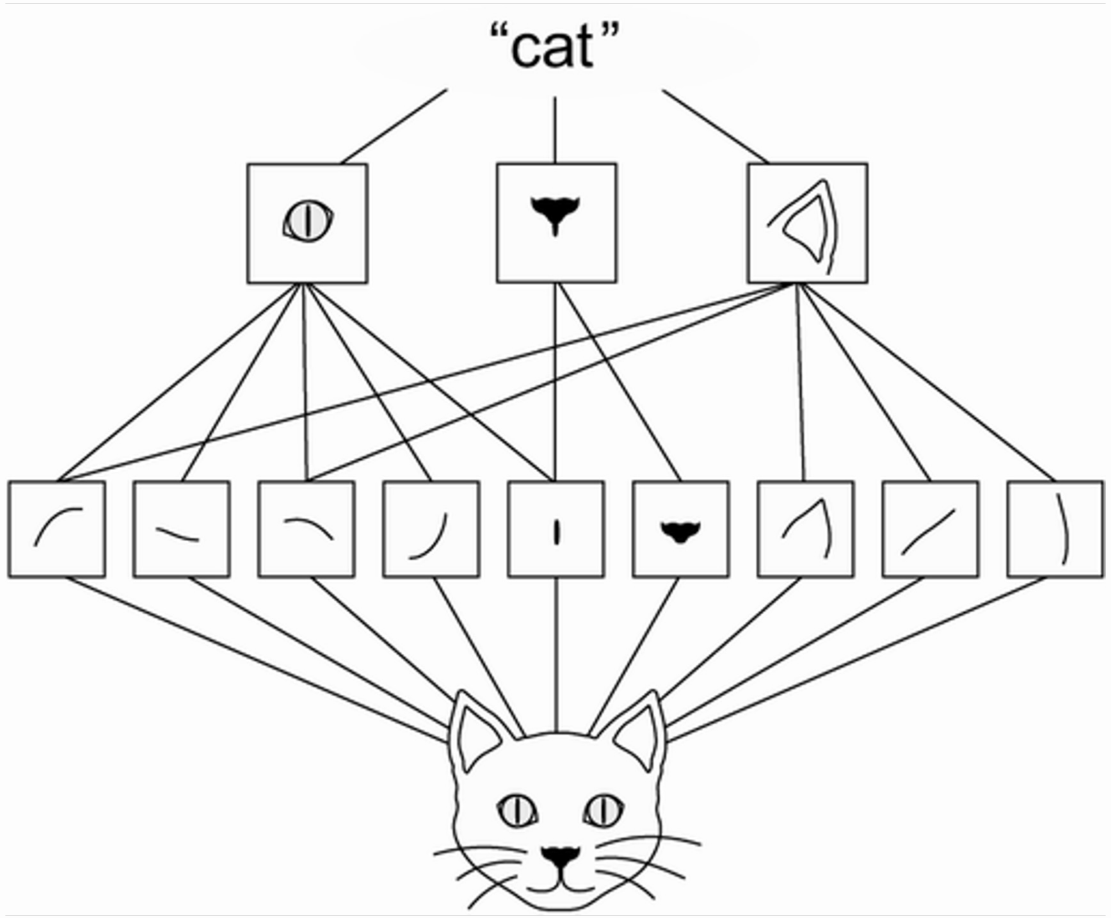
Feature Detection

How Does Your Visual System Work?

- We think that your brain processes individual visual receptors in groups, identifies combinations of inputs in proximity to one another that imply something like an edge (edge detection), combines that with color and so on. These low-level features are then processed together to arrive at higher level objects (e.g., a nose, a mouth, an eye).
- Those higher-level features are then processed together to yield a face (perhaps someone we know or do not know). Hence why you might have a hard time recognizing someone who has a new haircut, or who is wearing a facemask!



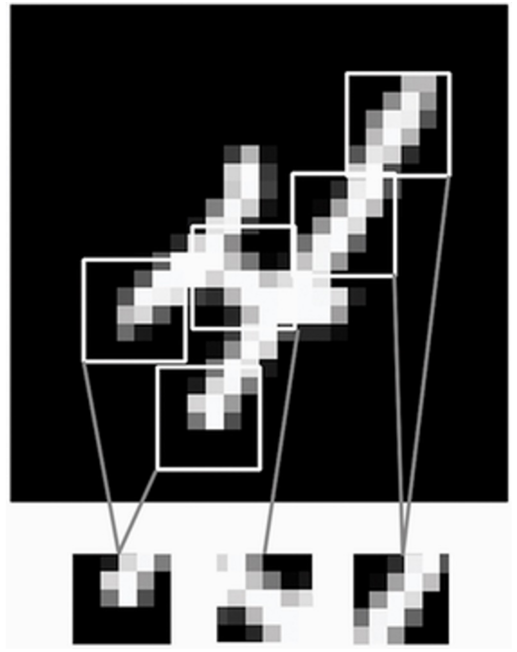
Feature Detection / Aggregation



What is Convolution?

Hone-in On Sub-sections of the Image

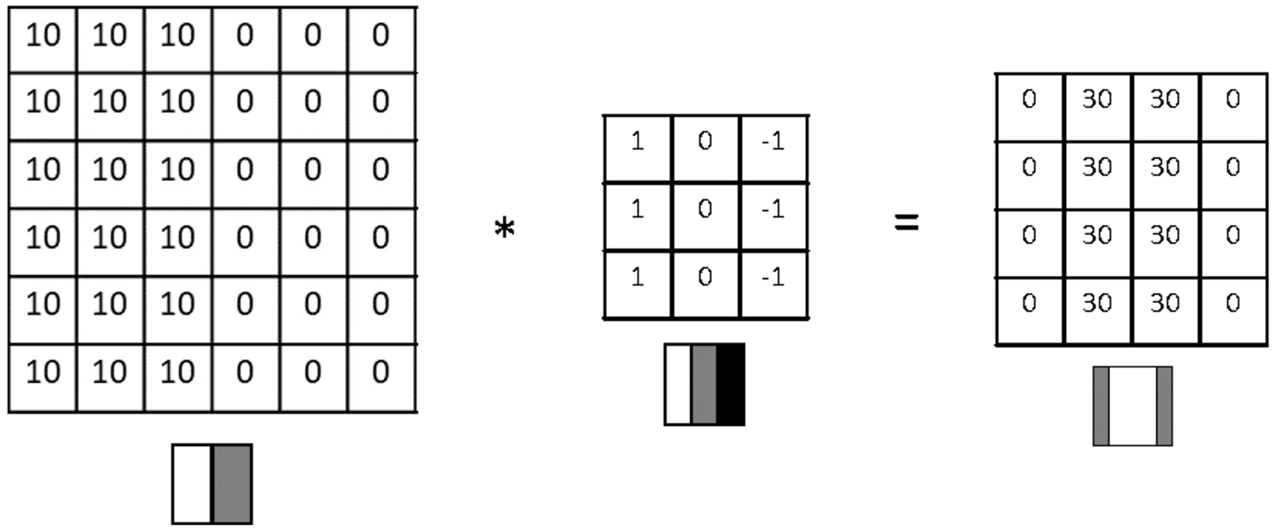
- So, if we have a 28x28 image, we might separately consider 3x3 pixel subsection of that image. Each subsection (they can be overlapping) is represented by its own node in the first hidden layer.
- That local input matrix (subfield) is considered in tandem with a ‘filter’ a matrix of weights. A filter might be something like



The Convolution Operation

Consider in Matrix Representation

- We have the raw image data (a.k.a. input feature map), the filter, and the result of passing our filter over our image (a.k.a. output feature map). We will have one output feature map for a given image, per filter (each filter is intended to detect a different type of feature).
- The filter elements are just weights for the Conv layer; we learn the filter values as part of the backpropagation process. So, the CNN will figure out what features to look for to predict the label (probably what a baby does when its first board and first learning how to process visual information).



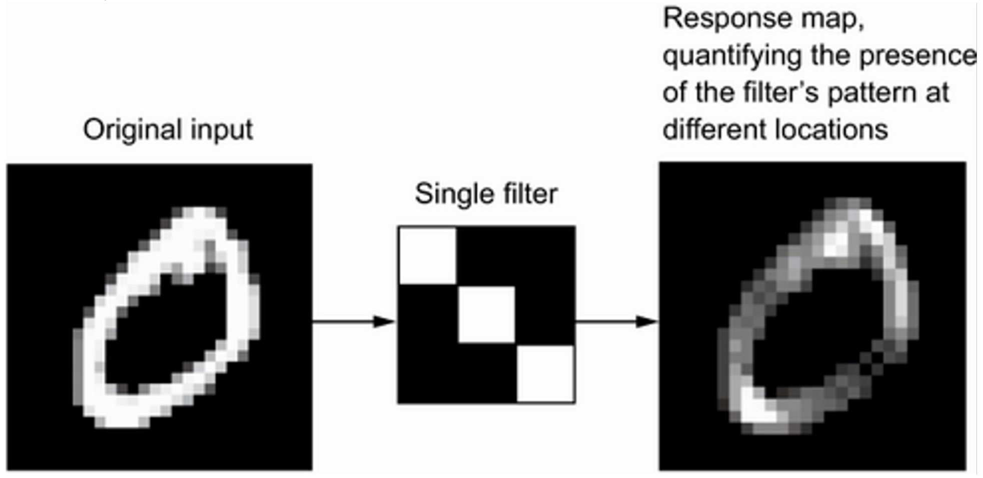
$$\begin{bmatrix} 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \end{bmatrix}$$

The Convolution Operation

Consider in Matrix Representation

- We have the raw image data (a.k.a. input feature map), the filter, and the result of passing our filter over our image (a.k.a. output feature map). We will have one output feature map for a given image, per filter (each filter is intended to detect a different type of feature).

- The filter elements are just weights for the Conv layer; we learn the filter values as part of the backpropagation process. So, the CNN will figure out what features to look for to predict the label (probably what a baby does when its first board and first learning how to process visual information).

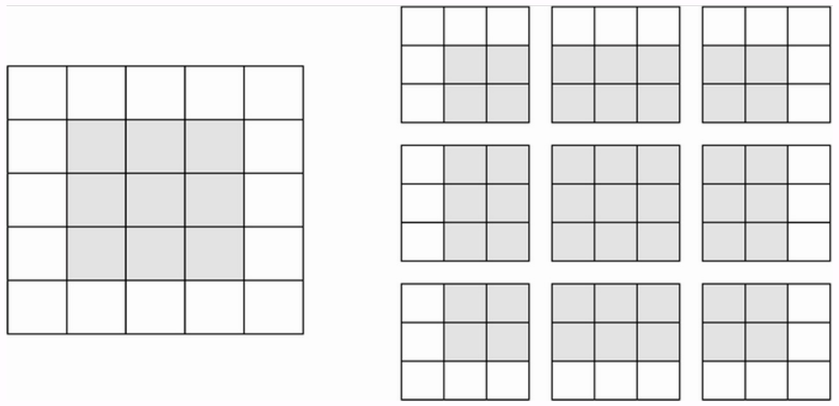


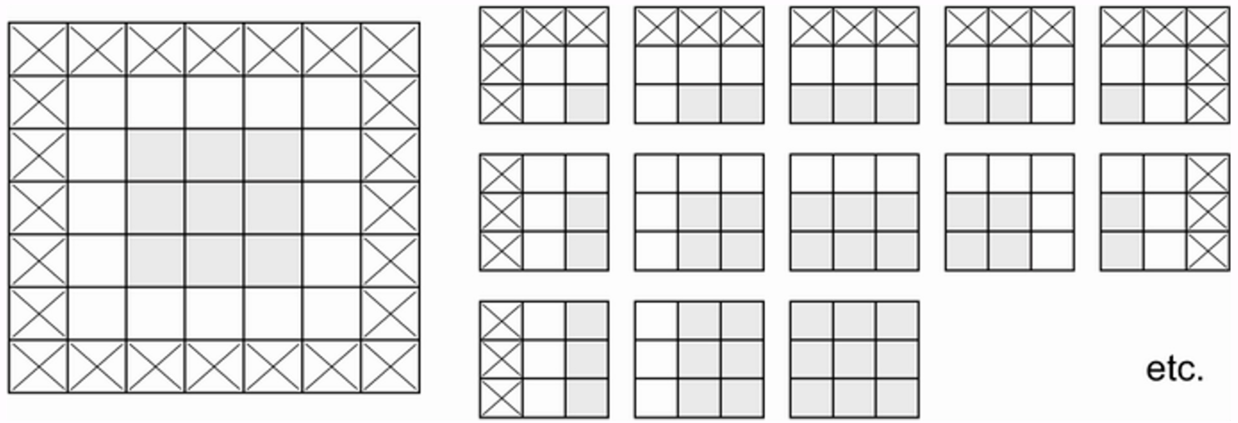
Original input
Single filter
Response map, quantifying the presence of the filter's pattern at different locations

Padding

Padding

- To prevent the transformation from down-sampling (reducing the size of the matrix during convolution to output), we can pad the edges of the image with 0's.

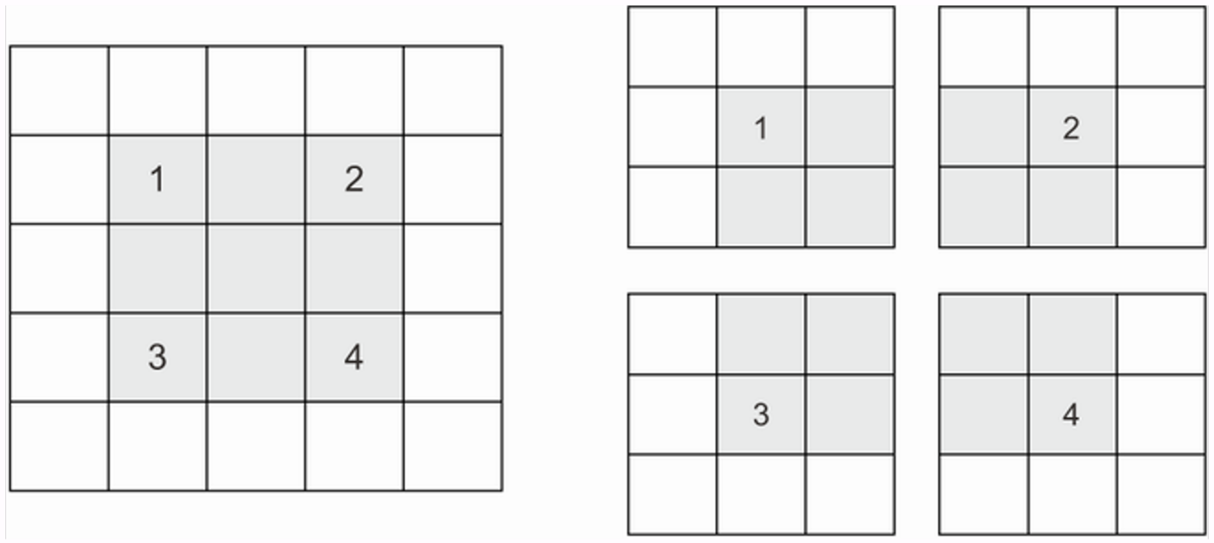




Strides

Strides

- Often, we will pass the filter over every pixel cell, but we don't have to; we might pass over every other cell. This is what strides refers to (skipping).



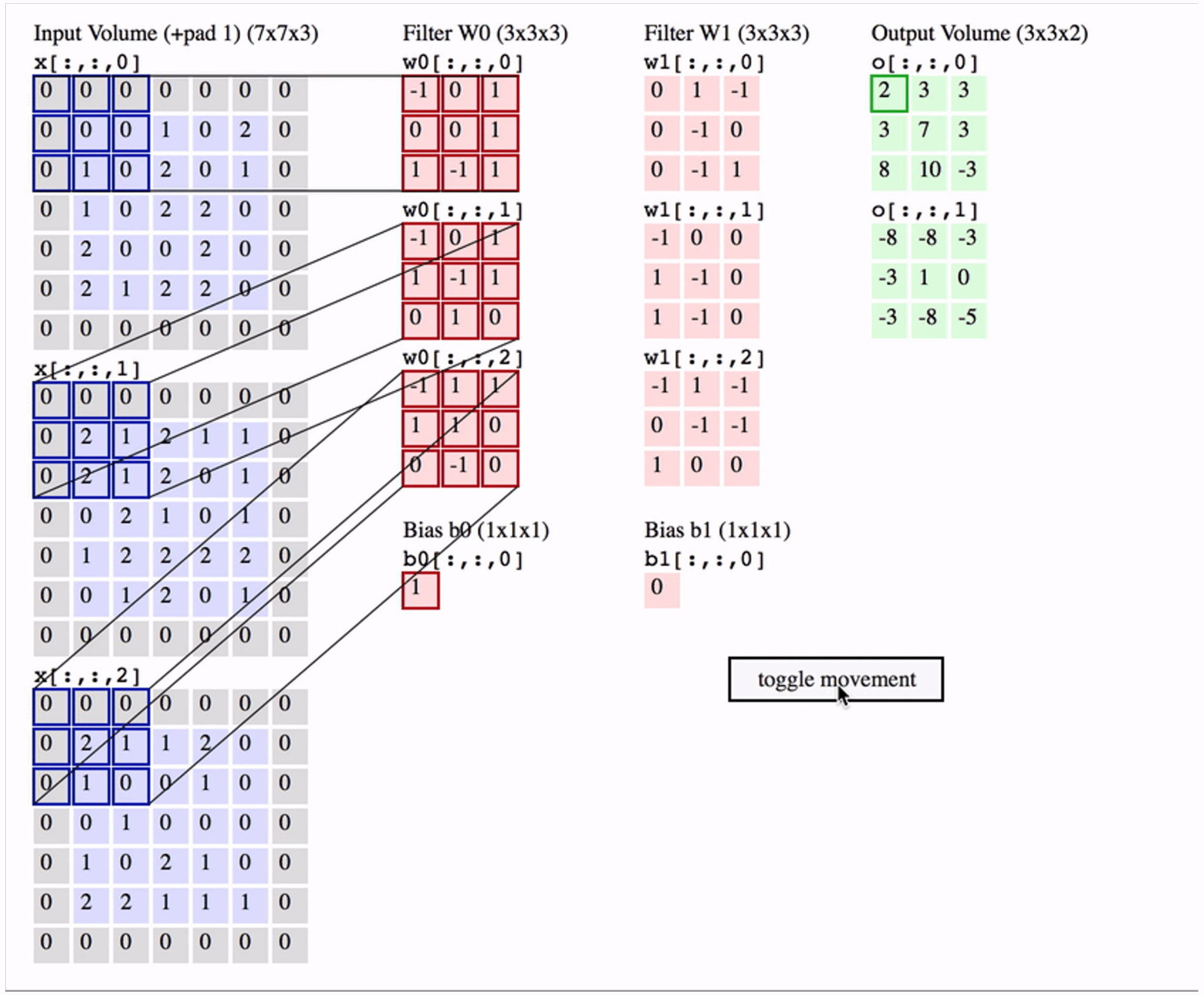
Padding and Strides

Padding

- To prevent the transformation from down-sampling (reducing the size of the matrix during convolution to output), we can pad the edges of the image with 0's.

Strides

- Often, we will pass the filter over every pixel cell, but we don't have to; we might pass over every other cell.
- This is what strides refers to (skipping).



Input Volume (+pad 1) (7x7x3)

x[:,:,0]

0	0	0	0	0	0	0
0	0	0	1	0	2	0
0	1	0	2	0	1	0
0	1	0	2	2	0	0
0	2	0	0	2	0	0
0	2	1	2	2	0	0
0	0	0	0	0	0	0

x[:,:,1]

0	0	0	0	0	0	0
0	2	1	2	1	1	0
0	2	1	2	0	1	0
0	0	2	1	0	1	0
0	1	2	2	2	2	0
0	0	1	2	0	1	0
0	0	0	0	0	0	0

x[:,:,2]

0	0	0	0	0	0	0
0	2	1	1	2	0	0
0	1	0	0	1	0	0
0	0	1	0	0	0	0
0	1	0	2	1	0	0
0	2	2	1	1	1	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

w0[:,:,0]

-1	0	1
0	0	1
1	-1	1

w0[:,:,1]

-1	0	1
1	-1	1
0	1	0

w0[:,:,2]

-1	1	1
1	1	0
0	-1	0

Bias b0 (1x1x1)

b0[:,:,0]

1

Filter W1 (3x3x3)

w1[:,:,0]

0	1	-1
0	-1	0
0	-1	1

w1[:,:,1]

-1	0	0
1	-1	0
1	-1	0

w1[:,:,2]

-1	1	-1
0	-1	-1
1	0	0

Bias b1 (1x1x1)

b1[:,:,0]

0

Output Volume (3x3x2)

o[:,:,0]

2	3	3
3	7	3
8	10	-3

o[:,:,1]

-8	-8	-3
-3	1	0
-3	-8	-5

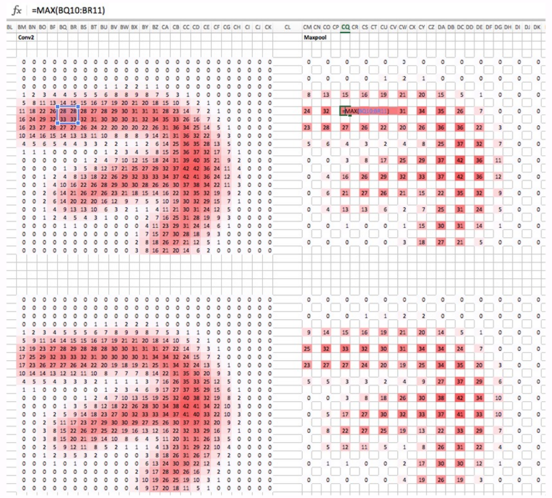
What is Pooling?

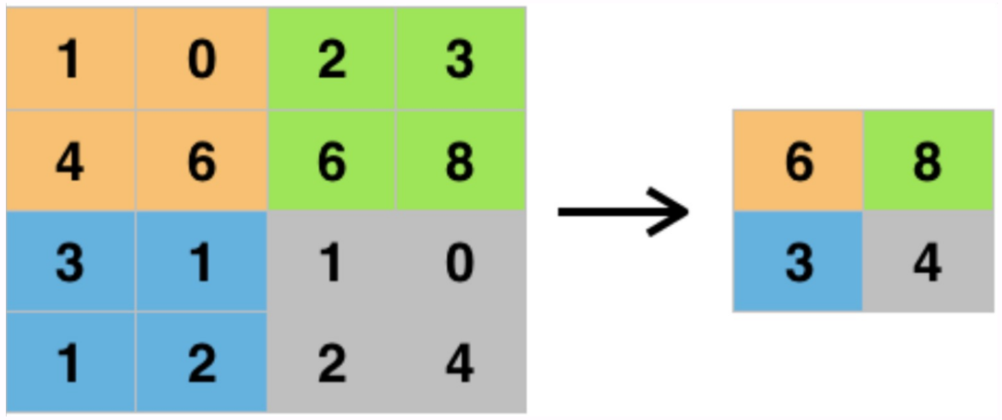
Down-sampling Detected Features

- The idea is to compress the resulting data down into a coarser representation, to reduce model complexity, and to also force attention toward a broader section of the original image (helps reduce overfitting).

Forcing Attention to Larger Blocks of the Original Image

- Because we typically use stride = pool width, the pooling output is aggregating over segments of the input.





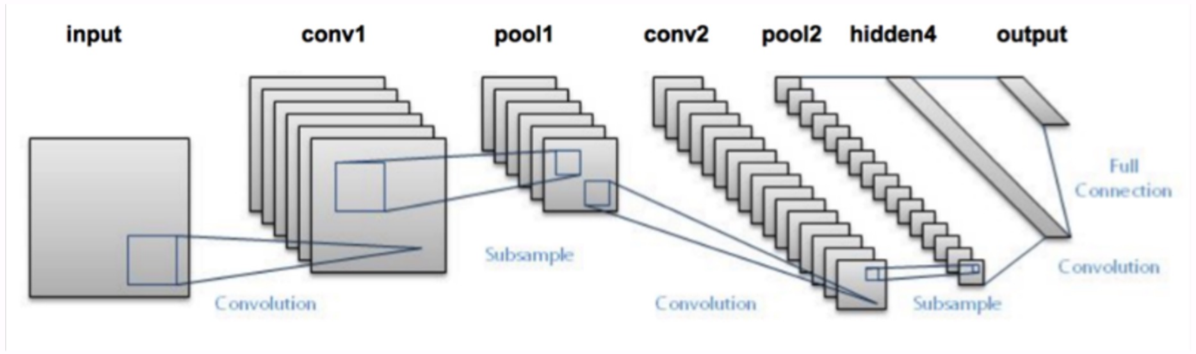
Basic Image Labeling Topology

Progressively More Filters

- As you move through the network, the number of features rises exponentially.
- More filters as you move along means it allows more permutations / combinations

Progressively Smaller Filter Maps

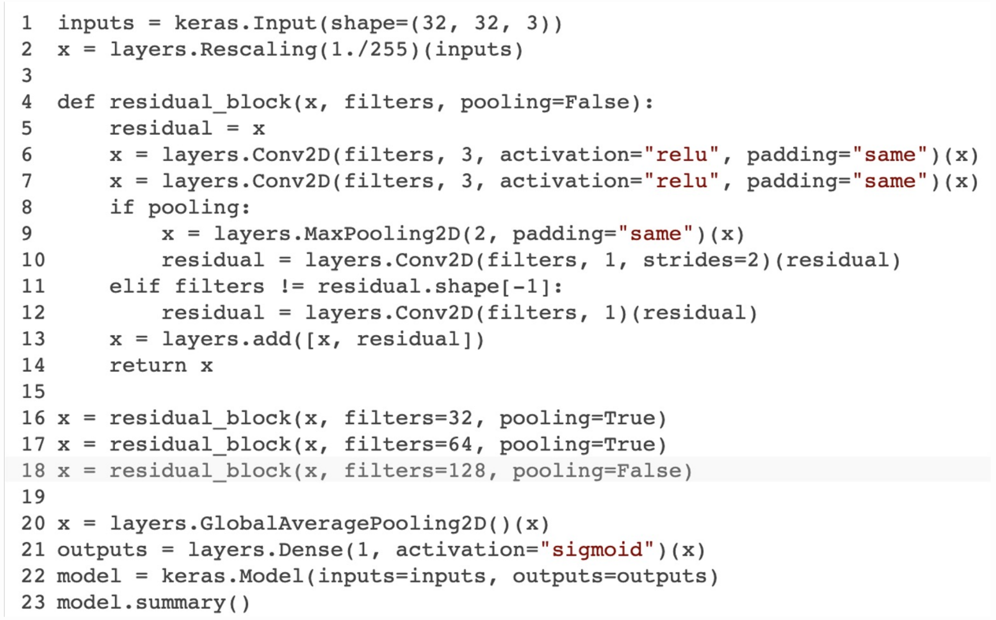
- Smaller filter map arises from the pooling steps, which means that each element of the final map distills features (high level features, derived from low level features, derived from raw pixels) derived from a larger segment of the original picture.



Residual Connections

Construct Repeating 'Blocks' of Layers

- Convolution, Pooling, Residual Connection, Convolution, Pooling, Residual Connection…
- In deep networks, vanishing gradient problem arises because there is noise in every step, and if we go far enough back the noise overwhelms the signal. The residual connections help ensure we have more signal than noise.

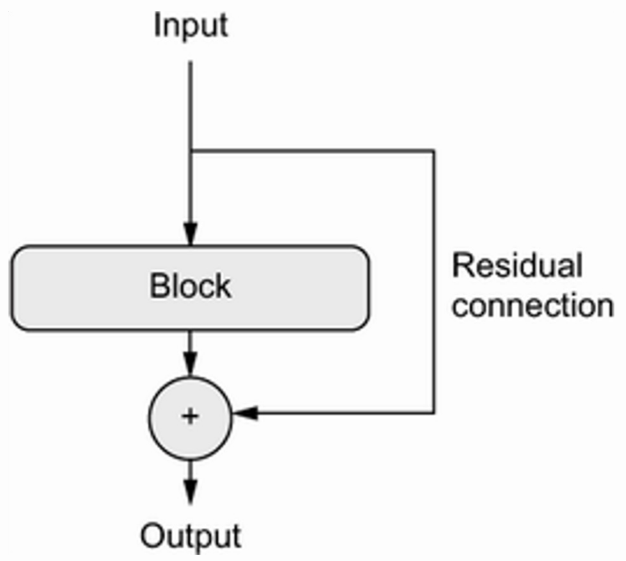


```python
inputs = keras.Input(shape=(32, 32, 3))
x = layers.Rescaling(1./255)(inputs)

def residual_block(x, filters, pooling=False):
    residual = x
    x = layers.Conv2D(filters, 3, activation="relu", padding="same")(x)
    x = layers.Conv2D(filters, 3, activation="relu", padding="same")(x)
    if pooling:
        x = layers.MaxPooling2D(2, padding="same")(x)
        residual = layers.Conv2D(filters, 1, strides=2)(residual)
    elif filters != residual.shape[-1]:
        residual = layers.Conv2D(filters, 1)(residual)
    x = layers.add([x, residual])
    return x

x = residual_block(x, filters=32, pooling=True)
x = residual_block(x, filters=64, pooling=True)
x = residual_block(x, filters=128, pooling=False)

x = layers.GlobalAveragePooling2D()(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs=inputs, outputs=outputs)
model.summary()
```

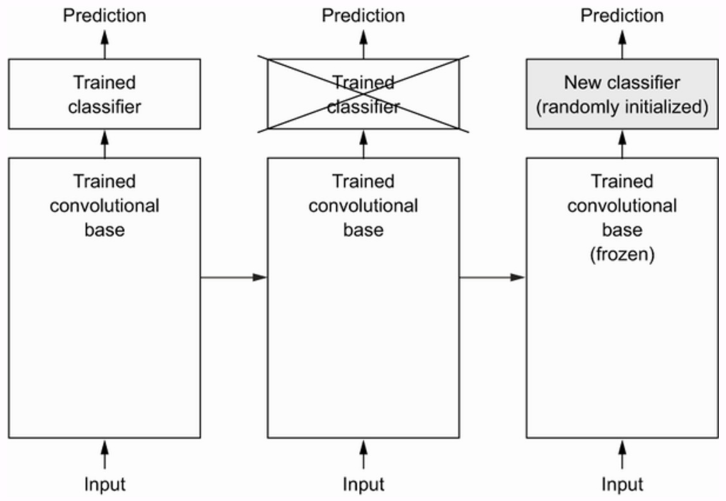


Pre-Trained Models: Feature Extraction

Take the convolutional base layers from someone else's model, then…

Two Options

- *Feed Data Through Model Base:* feed your images through convolutional base, take the outputs, and then use those as your predictors, feeding them into a network of dense layers.
- *Freeze Model Base and Include in Network:* Take the convolutional base layers from someone else's model and freeze them (make parameters non-trainable), then stack your (trainable) Dense layers onto the end. This lets you add data-augmentation to the front of the model.

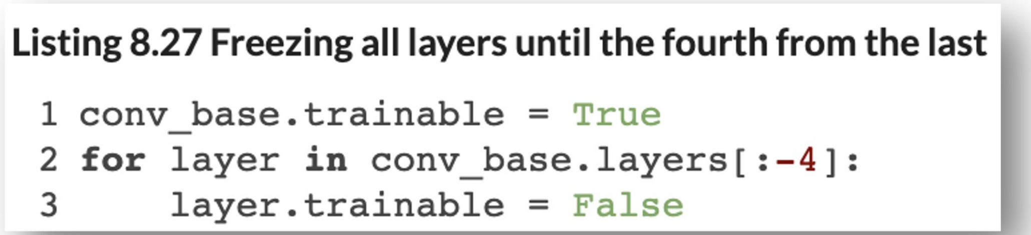

Prediction
Trained classifier
Trained convolutional base
Input
Prediction
Trained classifier
Trained convolutional base
Input
Prediction
New classifier (randomly initialized)
Trained convolutional base (frozen)
Input

Pre-Trained Models: Fine Tuning

Take the convolutional base layers from someone else's model, then...

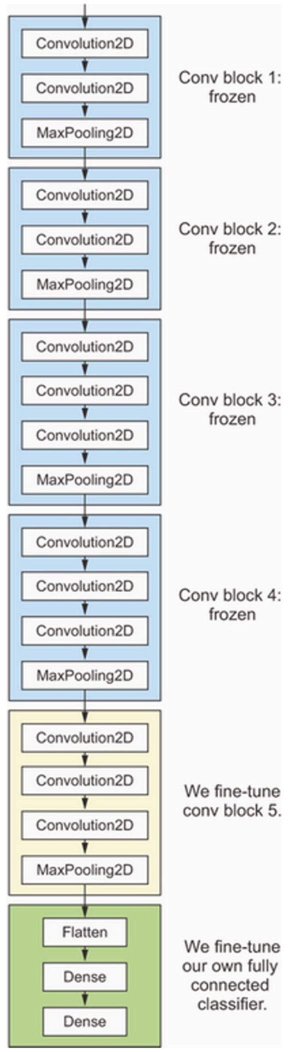
Freeze Only the First Several Layers

- Allow your network to modify / update the last few convolutional base layers as part of training, along with your own Dense layers...
- Iterate over the layers in the network and set the last few to be trainable.



Listing 8.27 Freezing all layers until the fourth from the last

```
conv_base.trainable = True
for layer in conv_base.layers[:-4]:
    layer.trainable = False
```



Questions?