File Name: T-ICML-O_2_I1_introduction

Format: Presenter in Studio

Presenter: Carl Osipov

Google Cloud

Implementing Convolutional Neural Networks for Image Classification Tasks

Carl Osipov

Agenda

Convolutional Neural Networks

Understanding Convolutions

Parameters (padding, stride)

Pooling layers

Implementing CNNs

Architecture walkthrough

Agenda

Convolutional Neural Networks

Understanding Convolutions

Parameters (padding, stride)

Pooling layers

Implementing CNNs

Architecture walkthrough

Agenda

Convolutional Neural Networks

Understanding Convolutions

Parameters (padding, stride)

Pooling layers

Implementing CNNs

Architecture walkthrough

Agenda

Convolutional Neural Networks

Understanding Convolutions

Parameters (padding, stride)

Pooling layers

Implementing CNNs

Architecture walkthrough

Agenda

Convolutional Neural Networks

Understanding Convolutions

Parameters (padding, stride)

Pooling layers

Implementing CNNs

Architecture walkthrough

Agenda

Convolutional Neural Networks

Understanding Convolutions

Parameters (padding, stride)

Pooling layers

Implementing CNNs

Architecture walkthrough

What are Convolutional Neural Networks (CNNs)?

← **3264px** Width →

↕ **2448px** Height →

8 Megapixel resolution

3264 (w) x 2448 (h) x 3 (RGB) =

23,970,816 per image

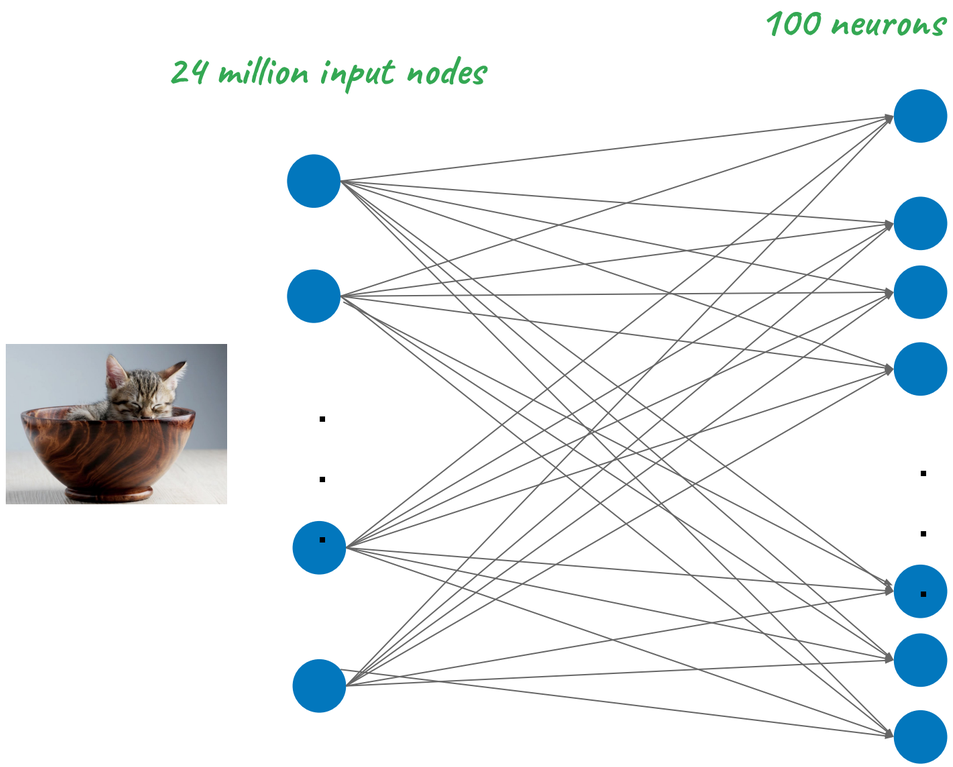
← **3264px** Width →

←**2448px** Height →

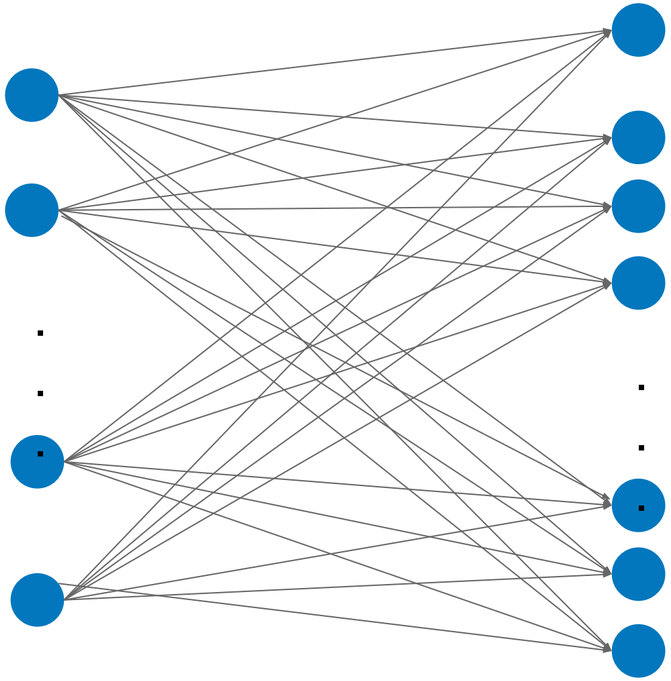
23,970,816 * 100 neurons =

23 Billion weights

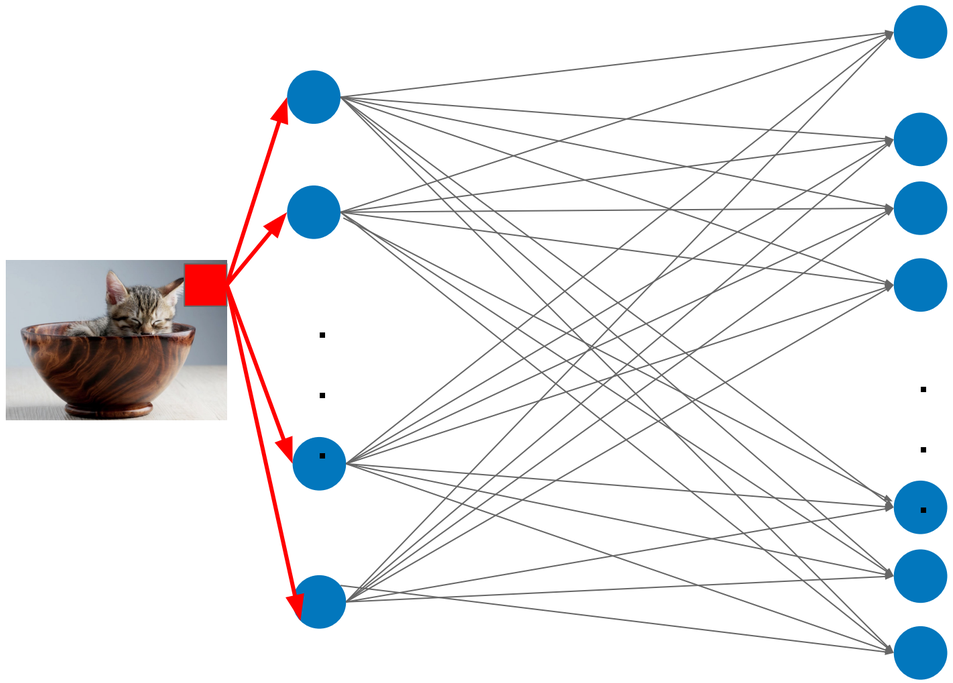
Using only a DNN for image classification could have billions of weights



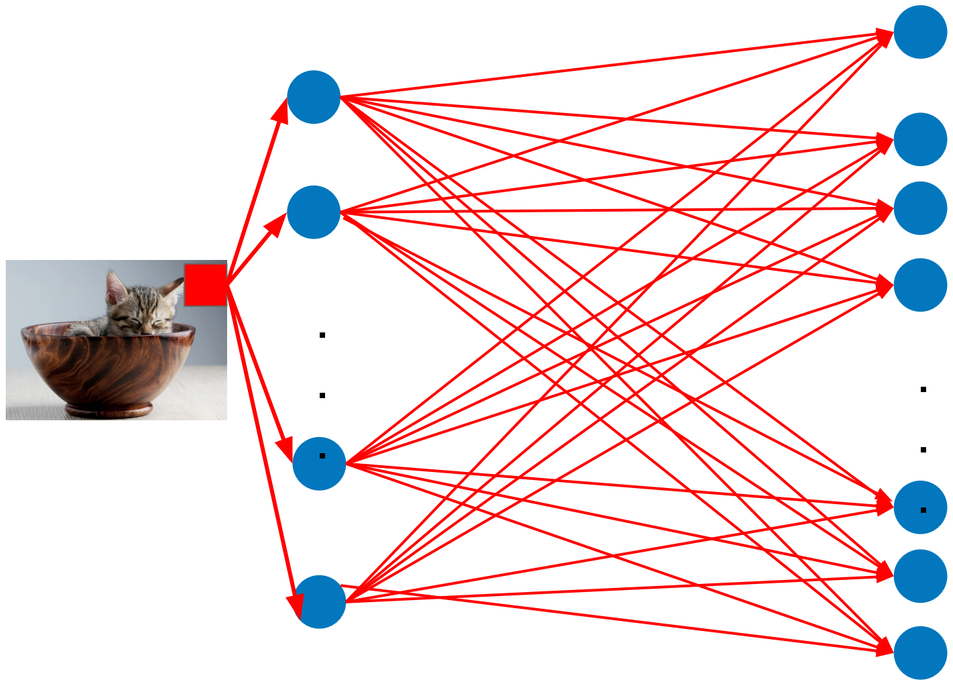
Using only a DNN for image classification could have billions of weights



Using only a DNN for image classification could have billions of weights

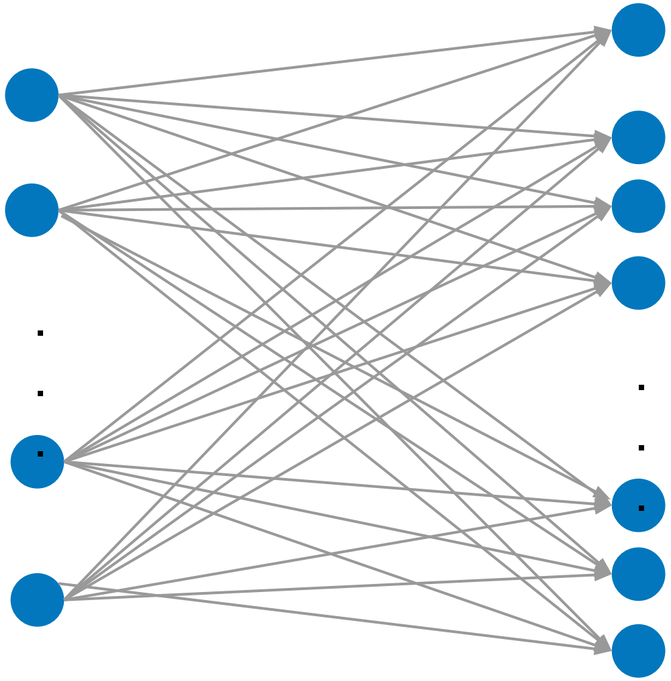


Using only a DNN for image classification could have billions of weights



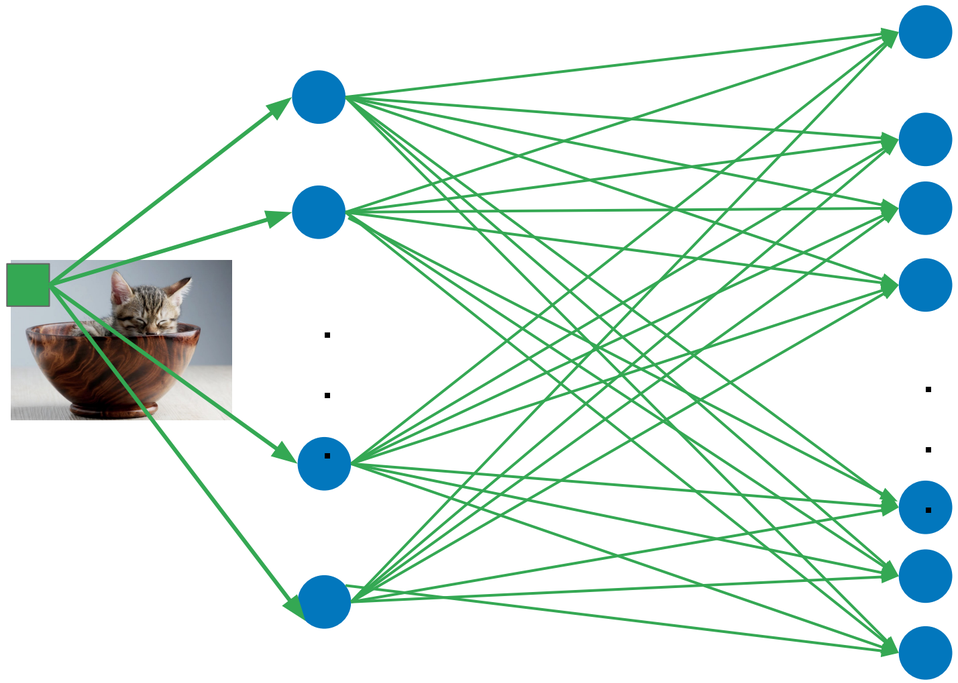
(Note: Many neurons not shown)

Using only a DNN for image classification could have billions of weights



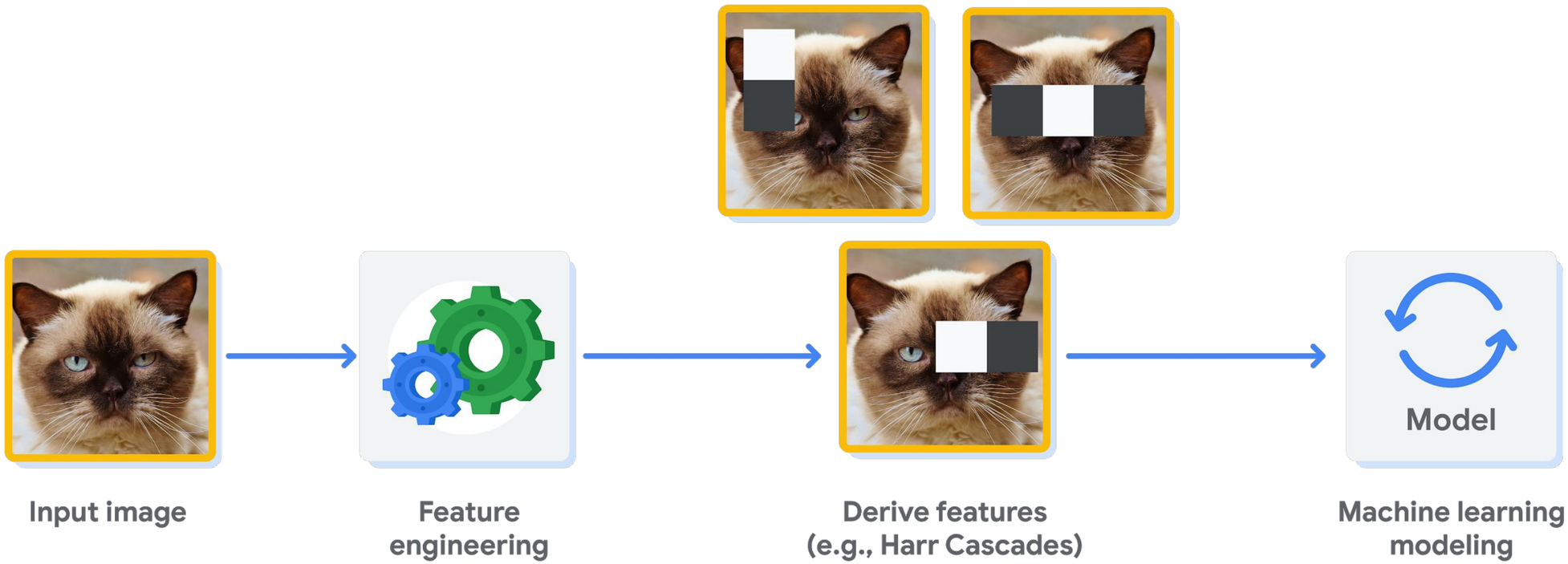
(Note: Many neurons not shown)

Using only a DNN for image classification could have billions of weights

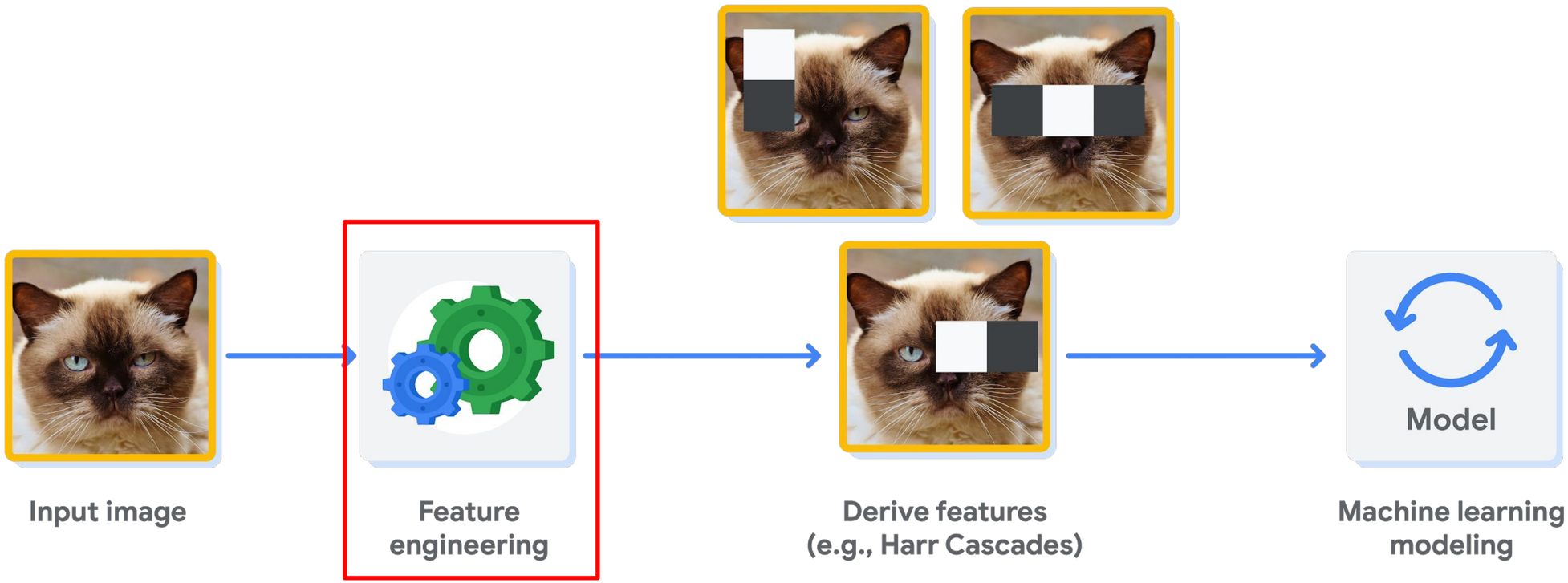


(Note: Many neurons not shown)

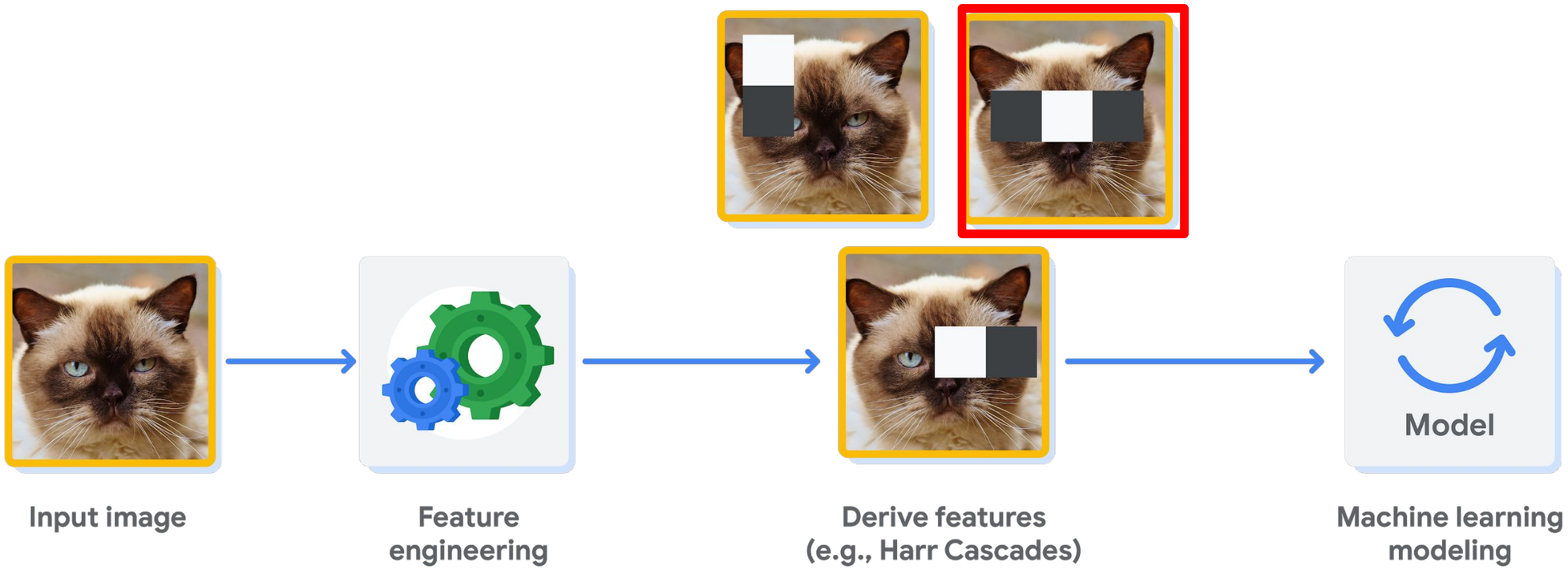
Traditionally image classification tasks utilize a feature engineering step



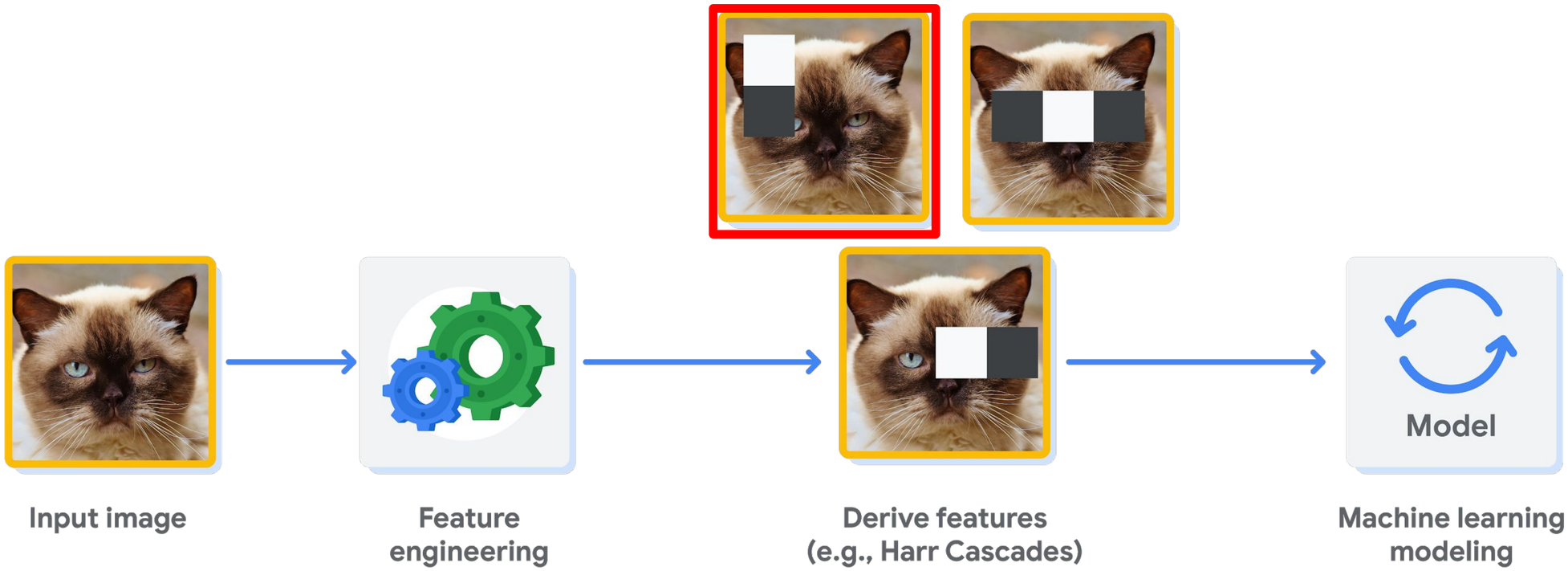
Traditionally image classification tasks utilize a feature engineering step



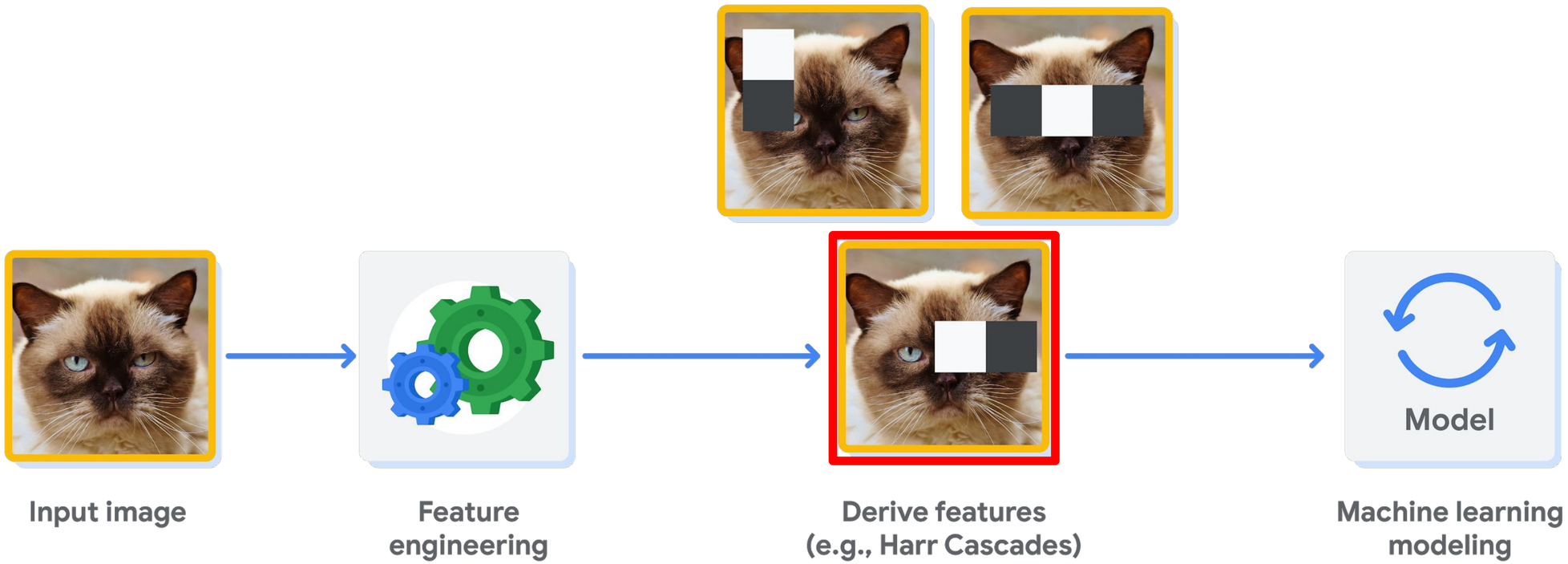
Pre-2012, ML practitioners tell the model which features to identify



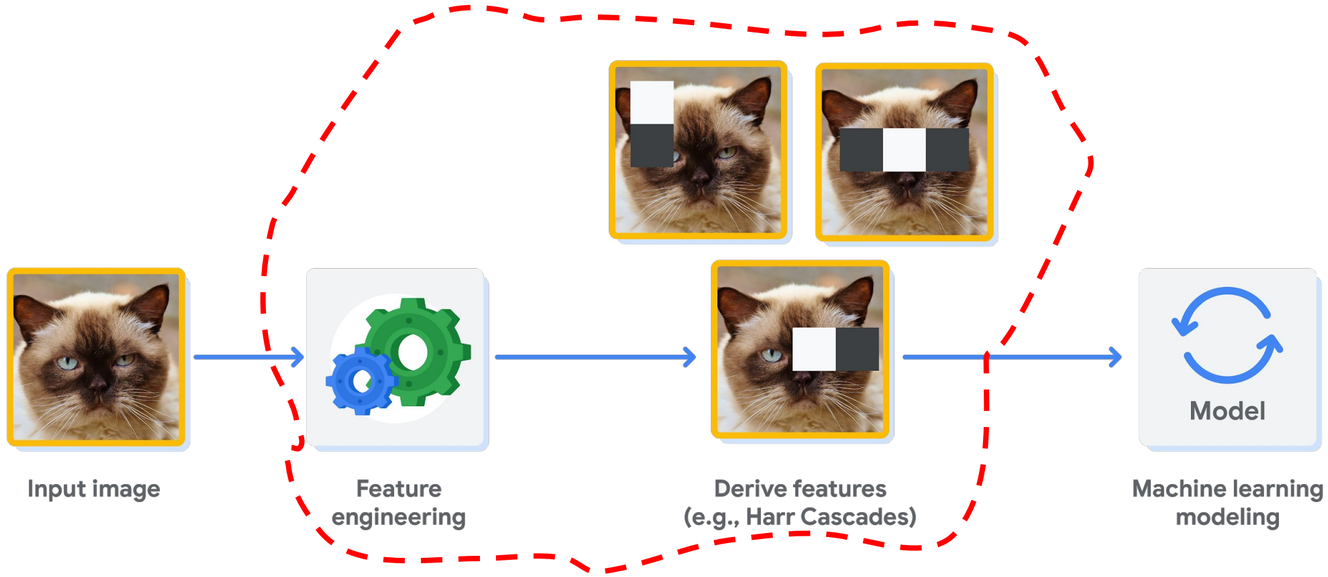
Pre-2012, ML practitioners tell the model which features to identify



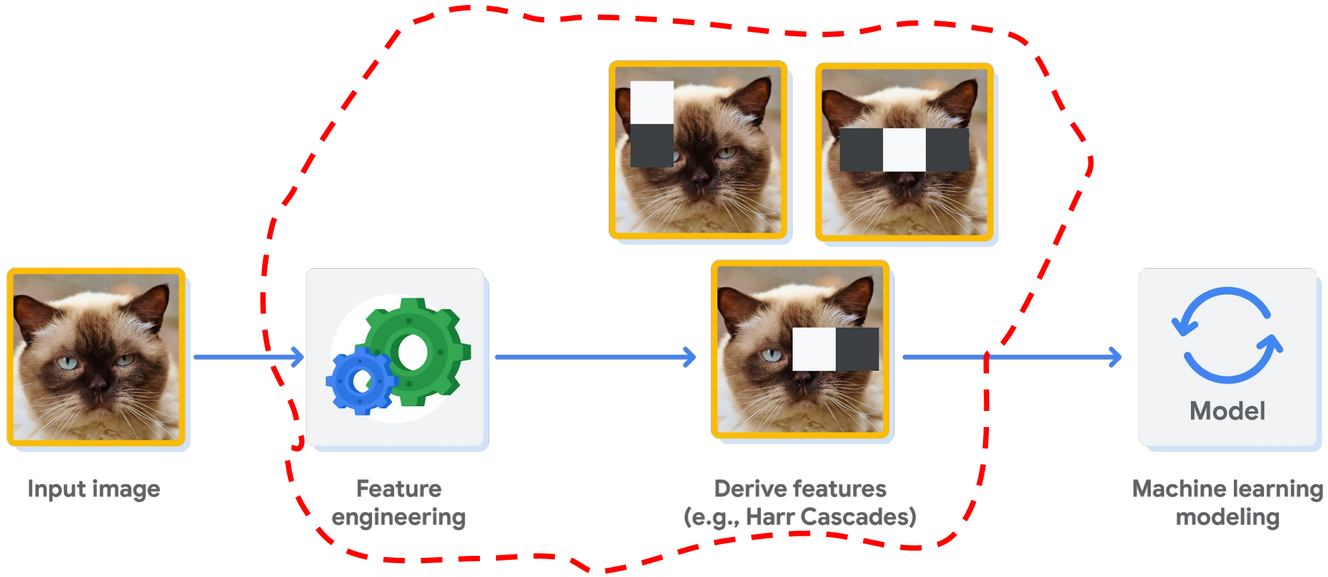
Pre-2012, ML practitioners tell the model which features to identify



Pre-2012

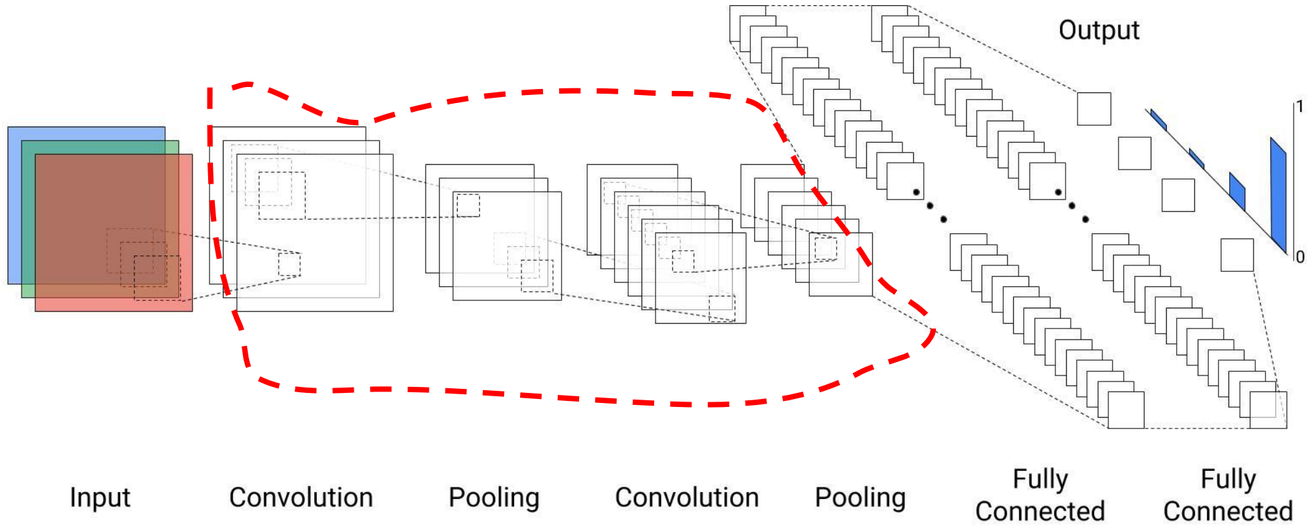

Model
Input image
Feature engineering
Derive features (e.g., Harr Cascades)
Machine learning modeling

Pre-2012

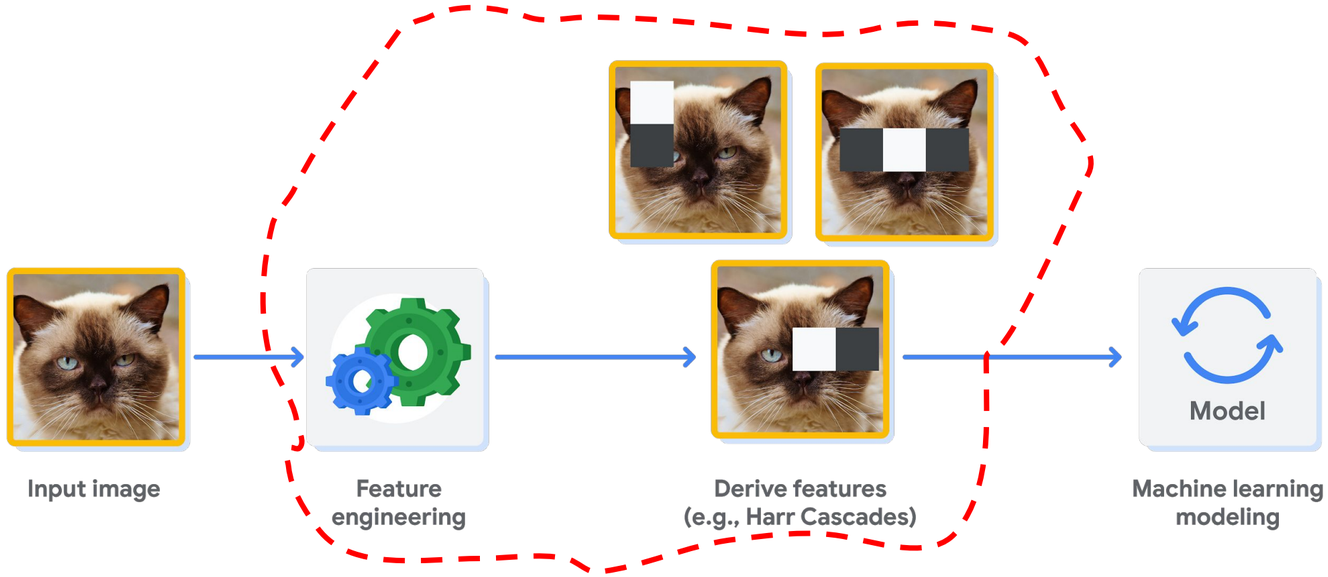


Post-2012

CNN model learns features itself during training

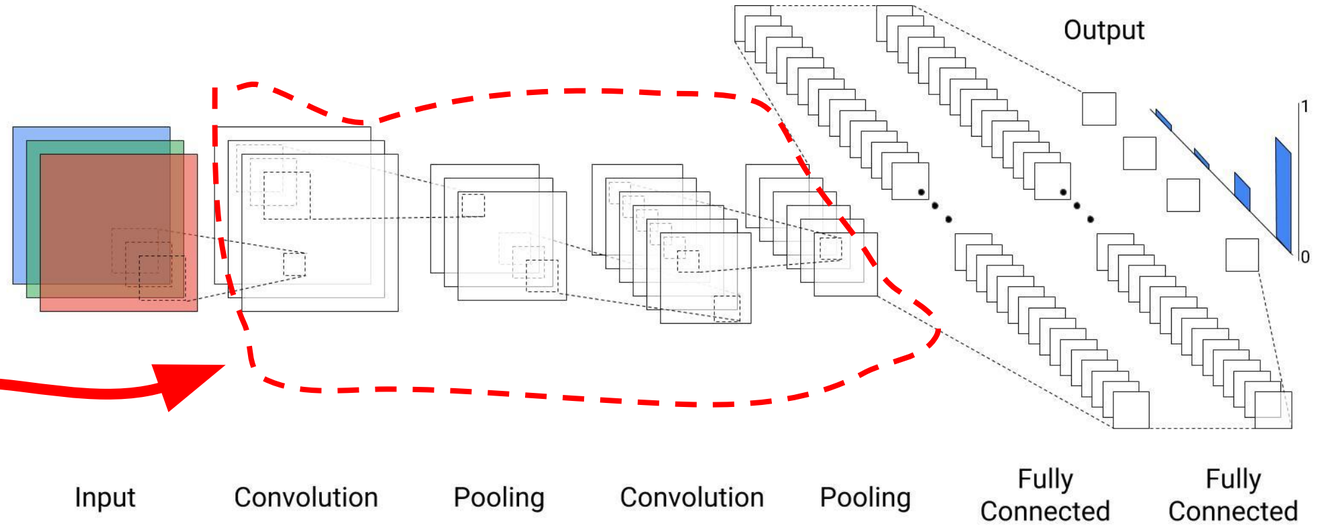


Pre-2012

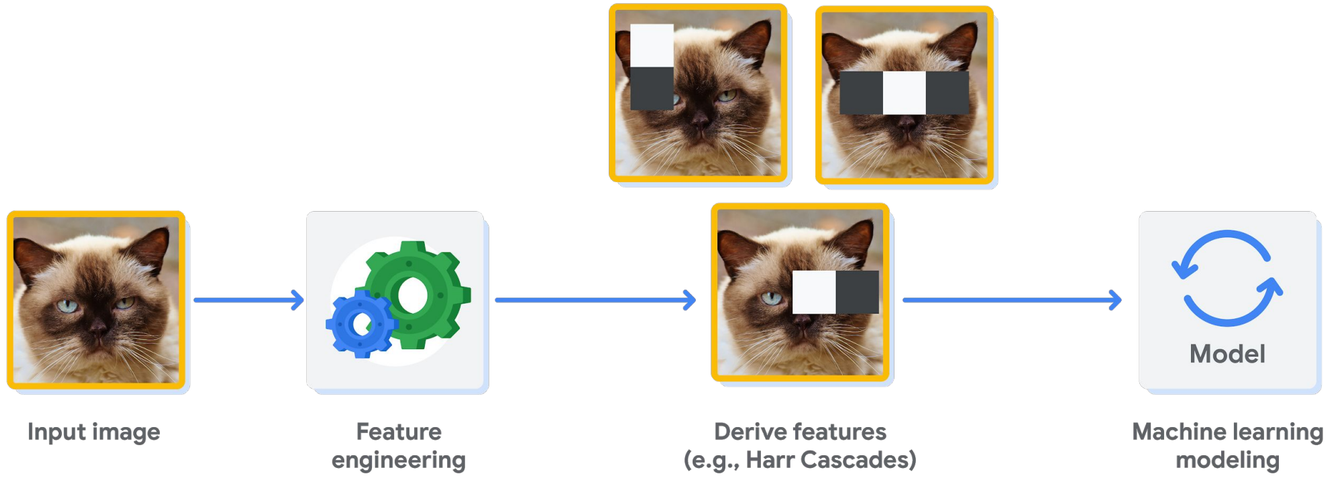


Post-2012

CNN model learns features itself during training

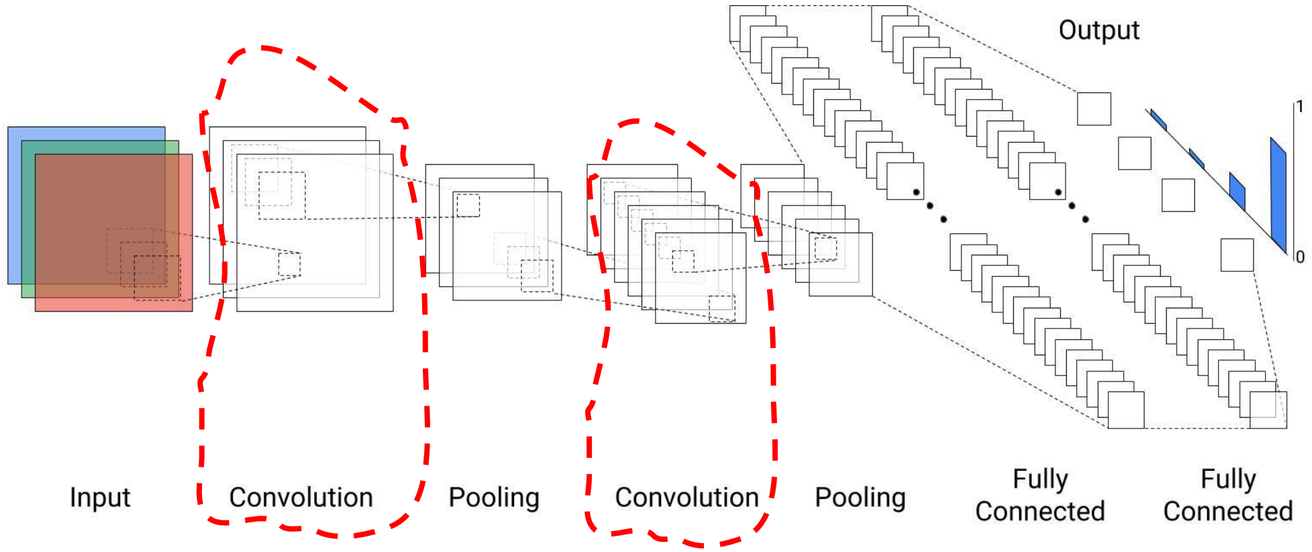


Pre-2012

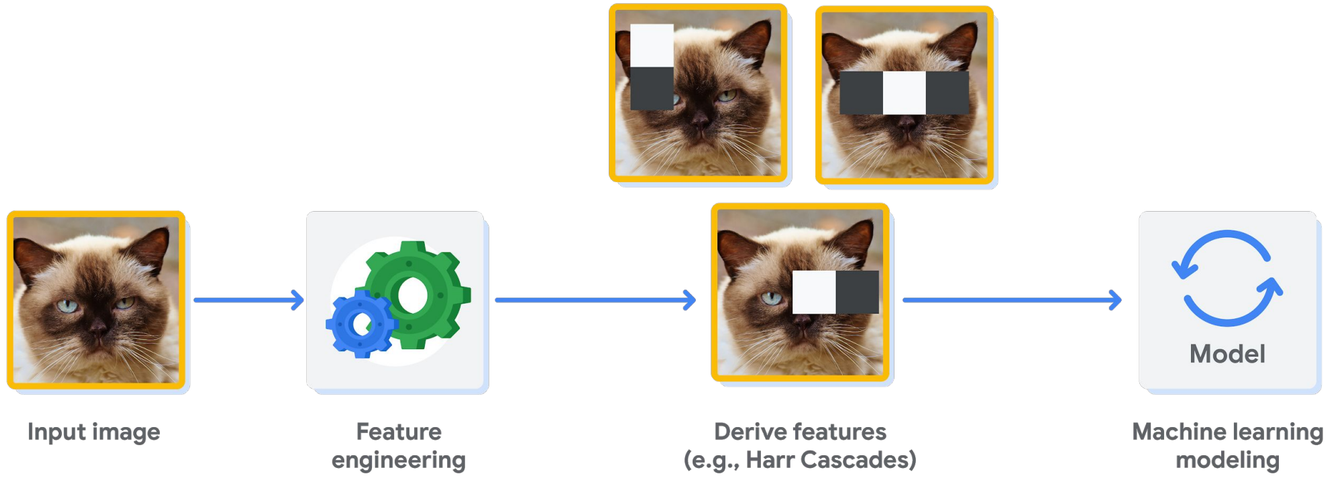


Post-2012

CNN model learns features itself during training

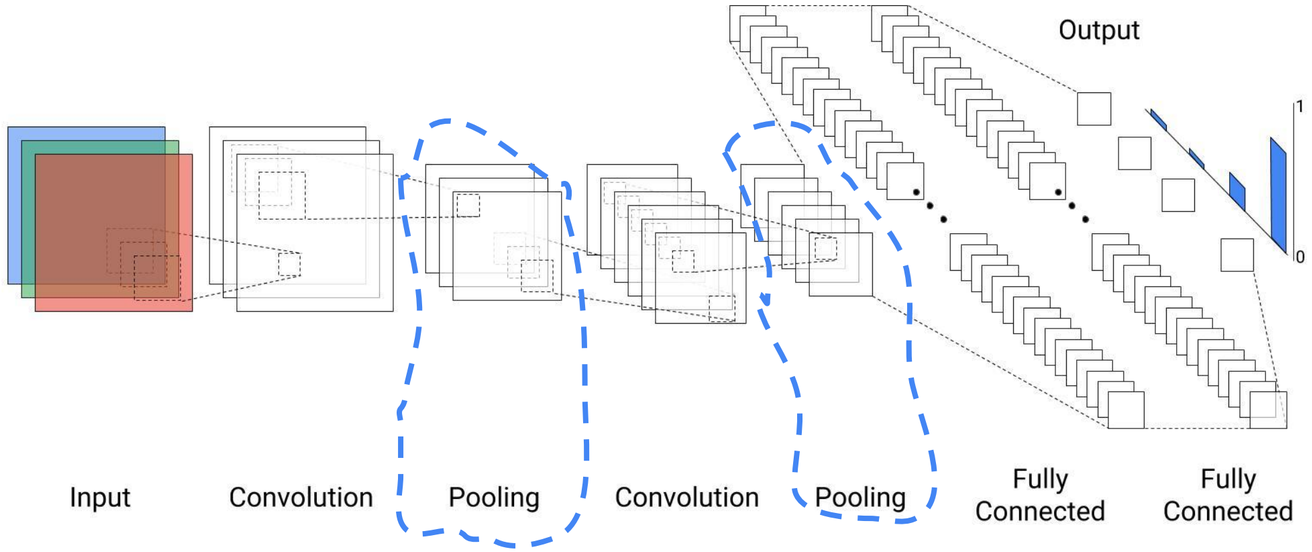


Pre-2012



Post-2012

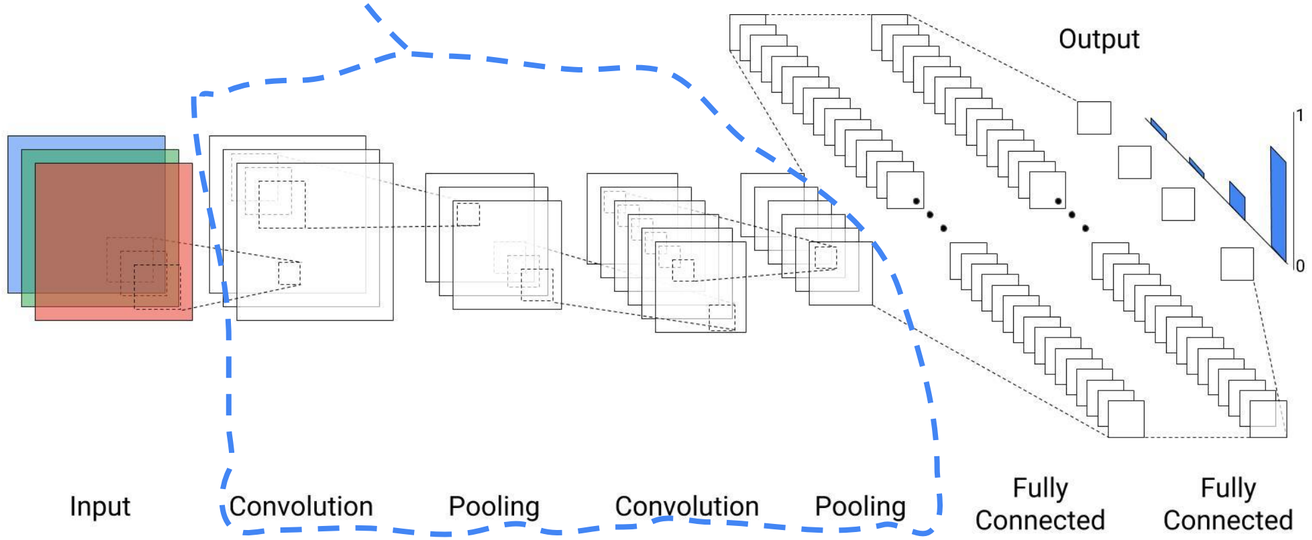
CNN model learns features itself during training



Post-2012

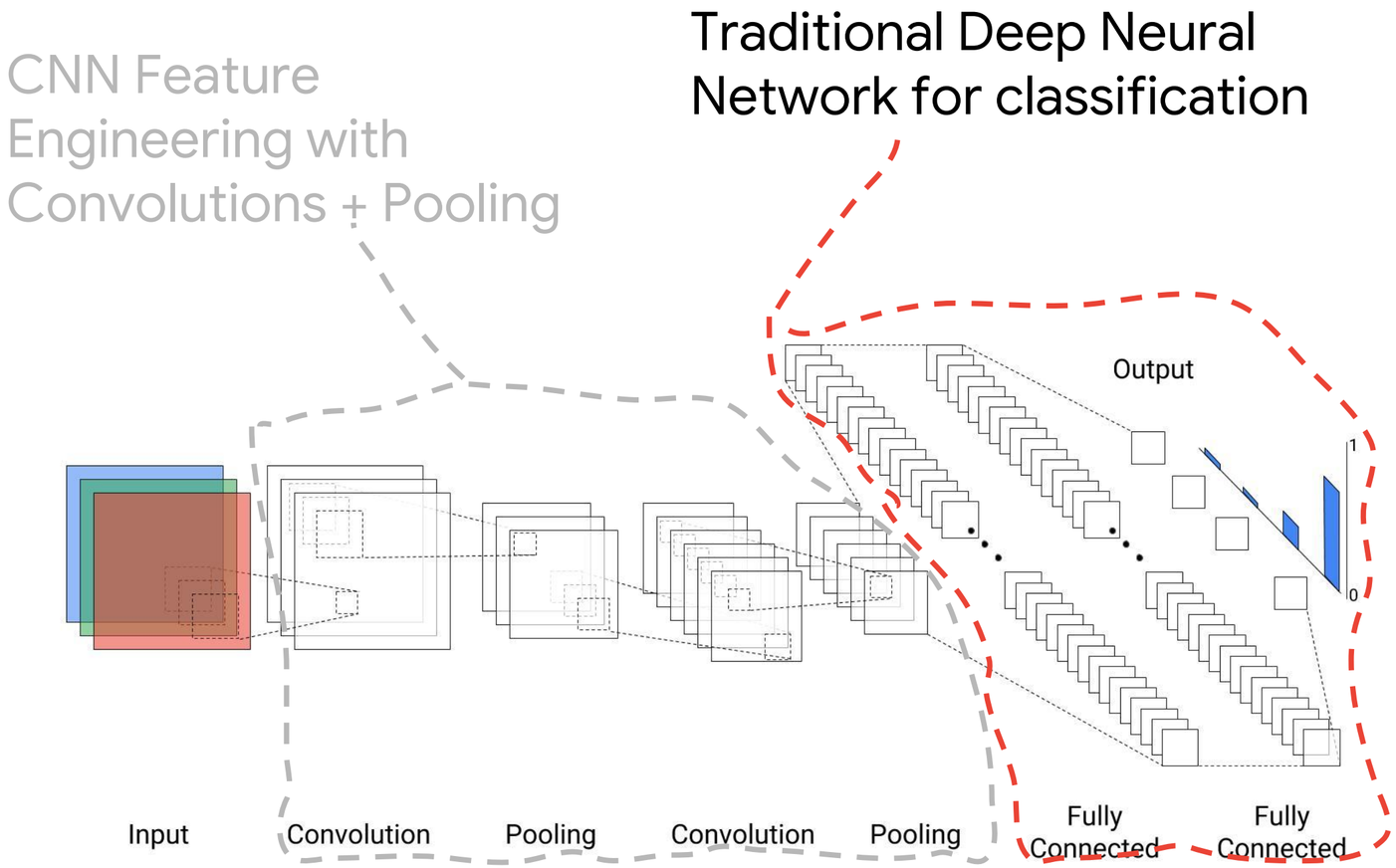
CNN model learns features itself during training

CNN Feature Engineering with Convolutions + Pooling



Post-2012

CNN model learns features itself during training



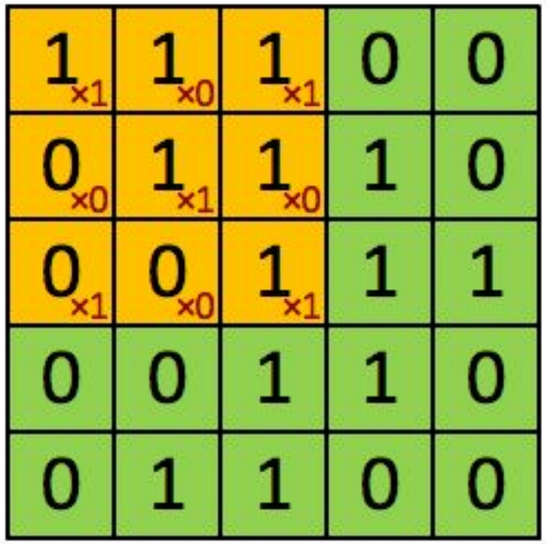
File Name: T-ICML-O_2_l2_understanding_convolutions

Format: Presenter in Studio

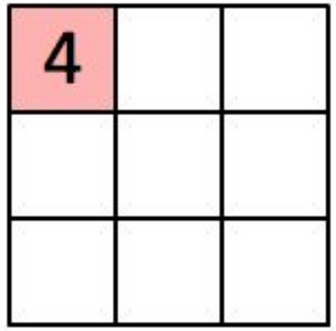
Presenter: Carl Osipov

A **convolution** is an operation that processes groups of nearby pixels

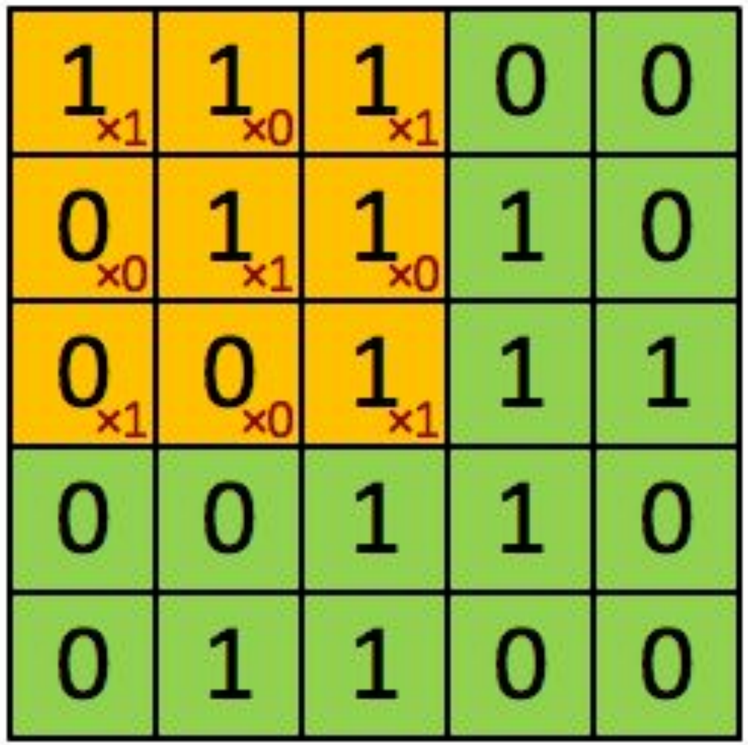
A **convolution** is an operation that processes groups of nearby pixels



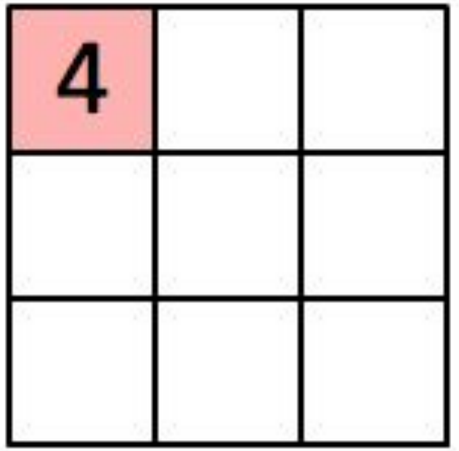
Image



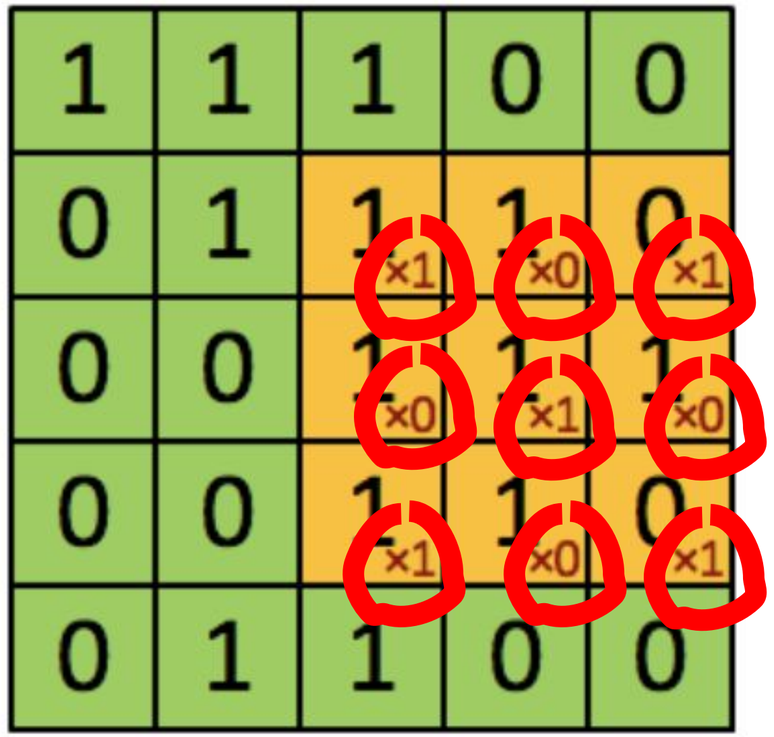
Convolved Feature



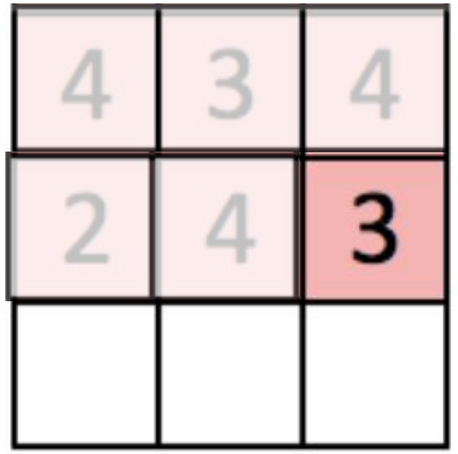
Image



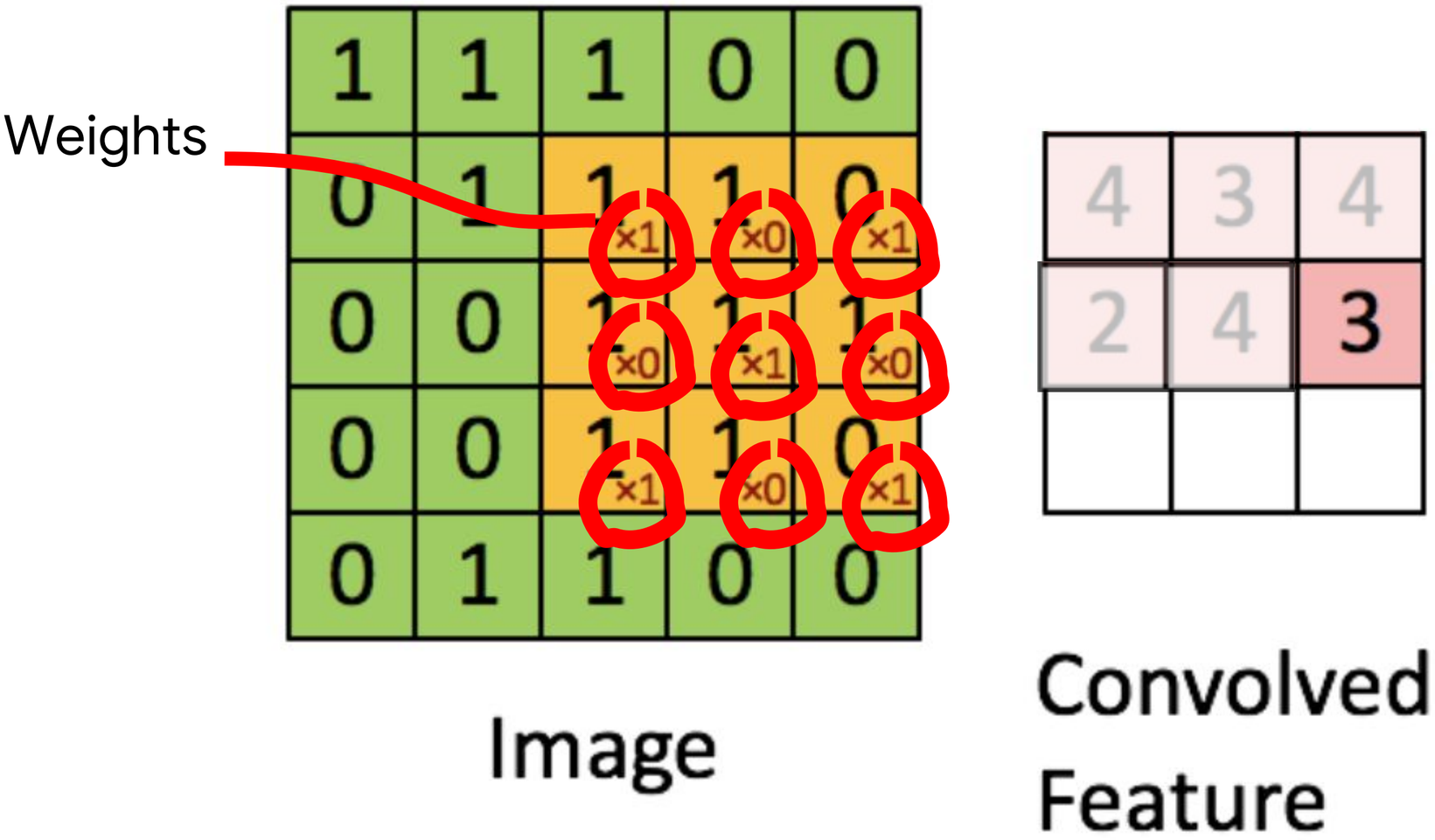
Convolved Feature

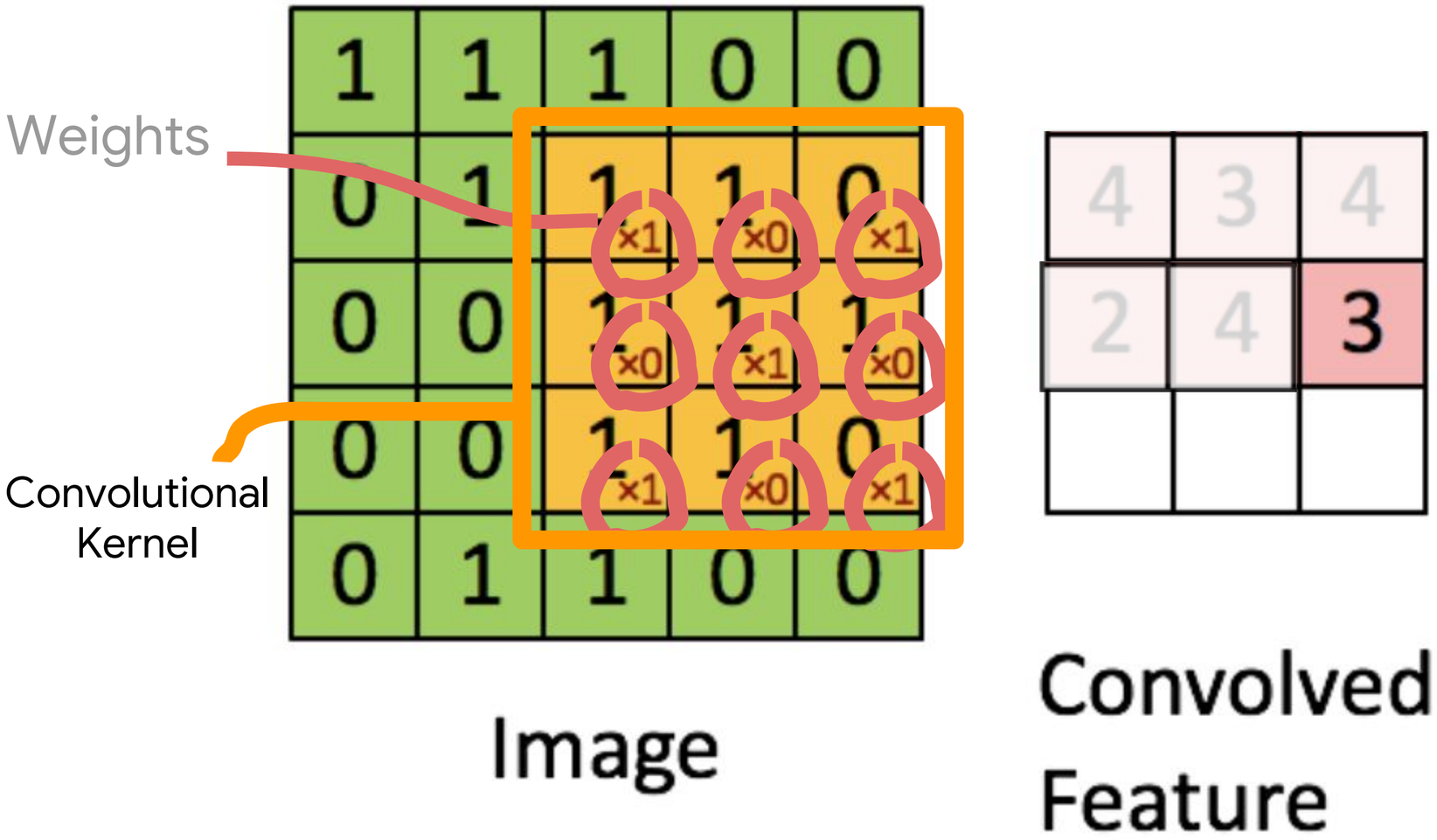


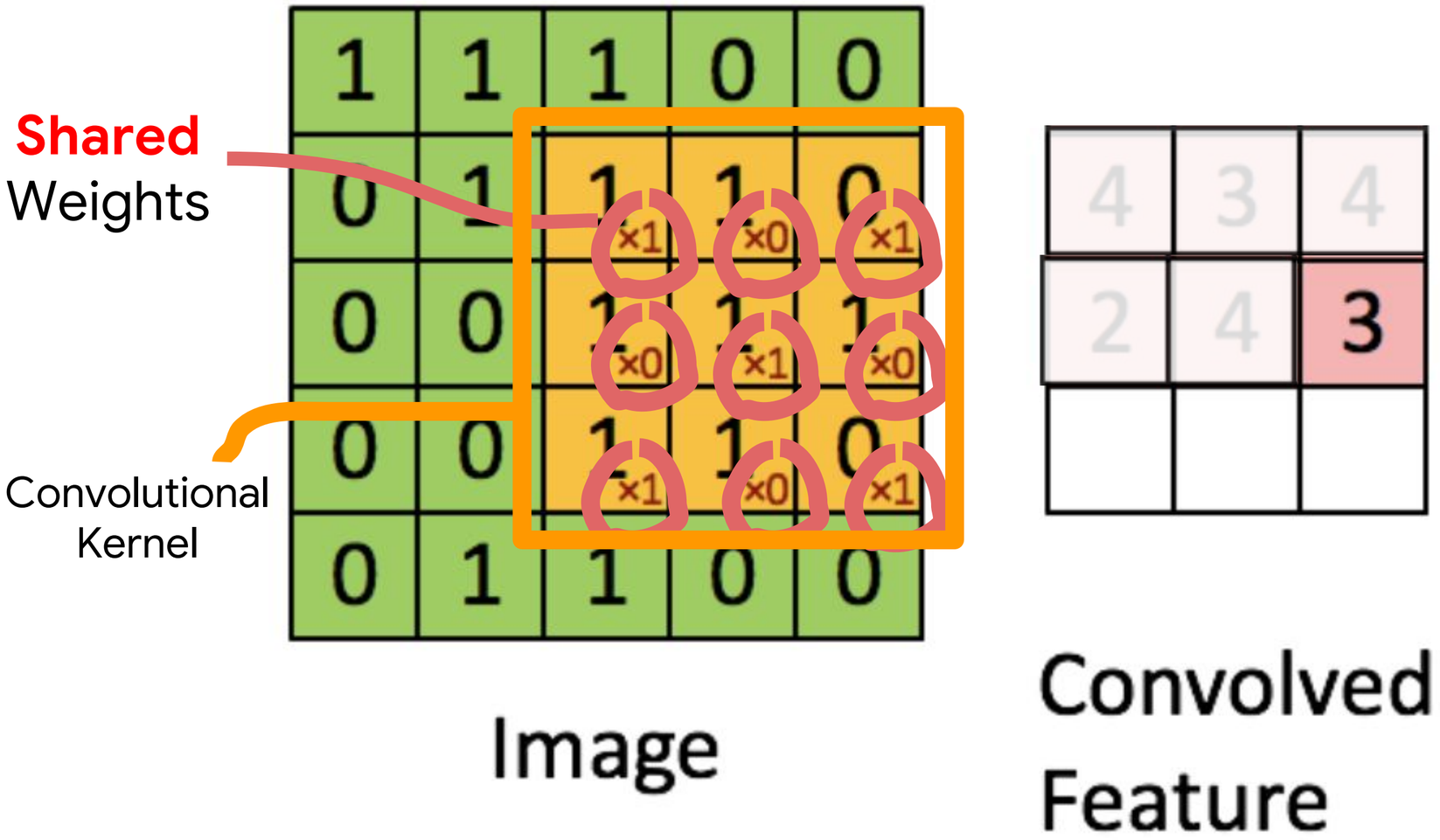
Image

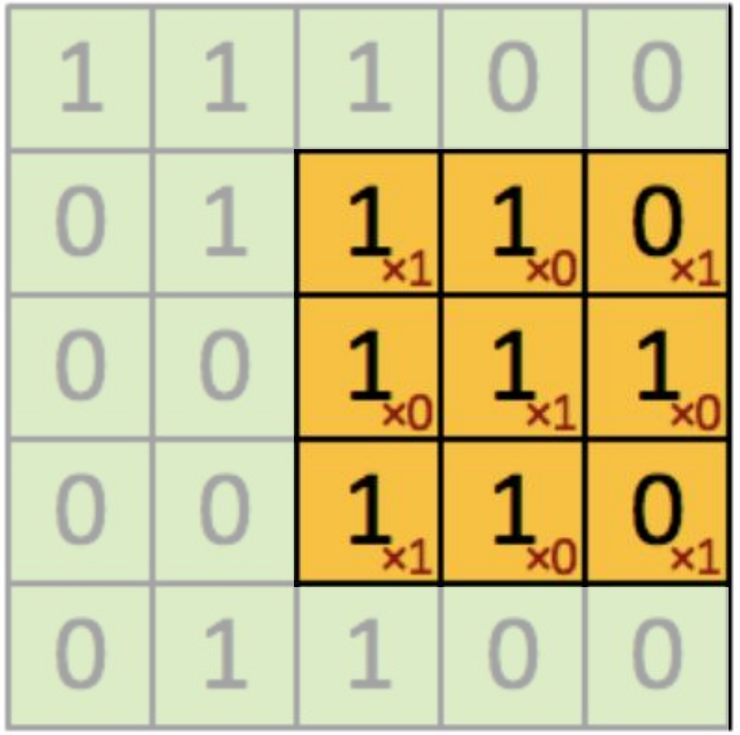


Convolved Feature

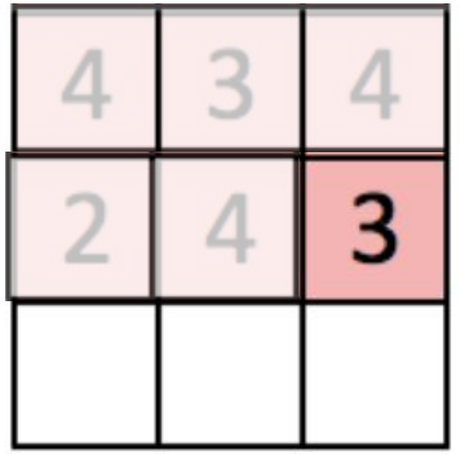

Weights
1 1 1 0 0
0 1 1 1 0
0 0 1 1 1
0 0 1 1 0
0 1 1 0 0
×1 ×0 ×1
×0 ×1 ×0
×1 ×0 ×1
4 3 4
2 4 3
Image
Convolved Feature


Weights
Convolutional Kernel
1 1 1 0 0
0 1 1 1 0
0 0 1 1 1
0 0 1 1 0
0 1 1 0 0
×1 ×0 ×1
×0 ×1 ×0
×1 ×0 ×1
4 3 4
2 4 3
Image
Convolved Feature


Shared
Weights
Convolutional
Kernel
1 1 1 0 0
0 1 1 1 0
0 0 1 1 1
0 0 1 1 0
0 1 1 0 0
×1 ×0 ×1
×0 ×1 ×0
×1 ×0 ×1
4 3 4
2 4 3
Image
Convolved
Feature

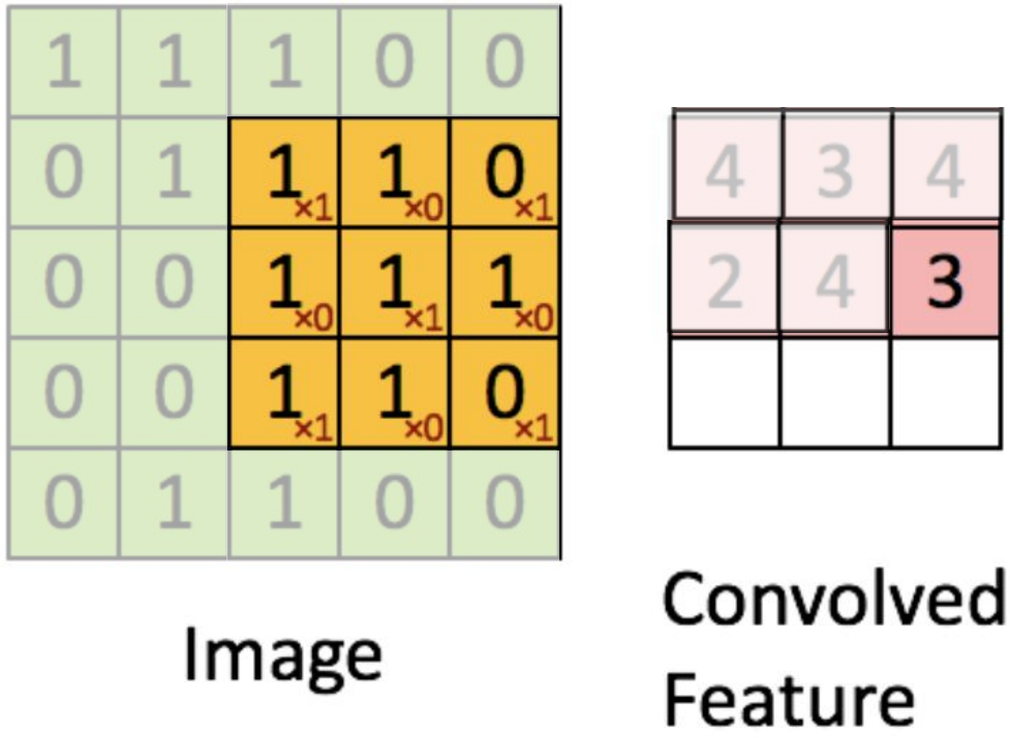


Image

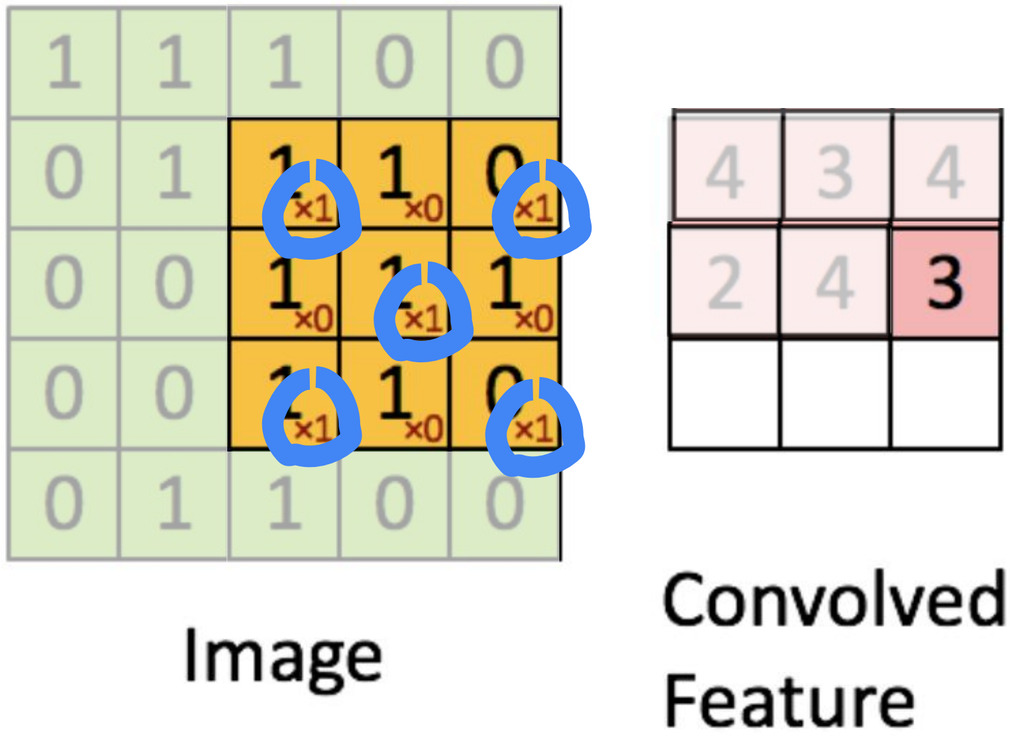


Convolved Feature

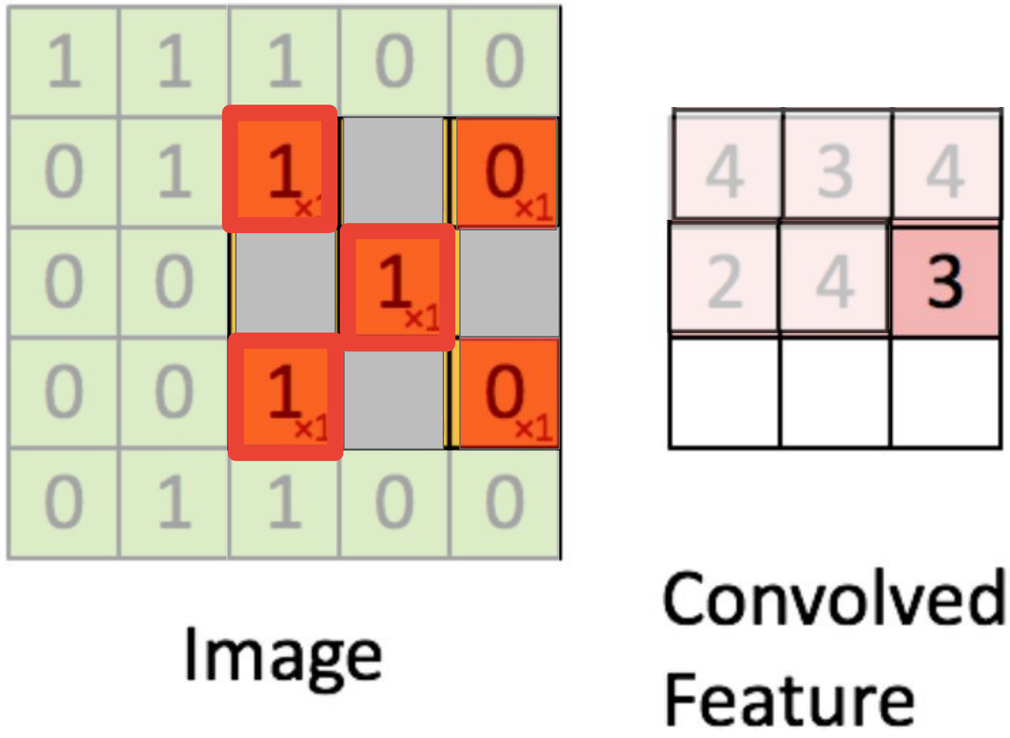
A convolved feature map is created by multiplying the kernel weights by the features



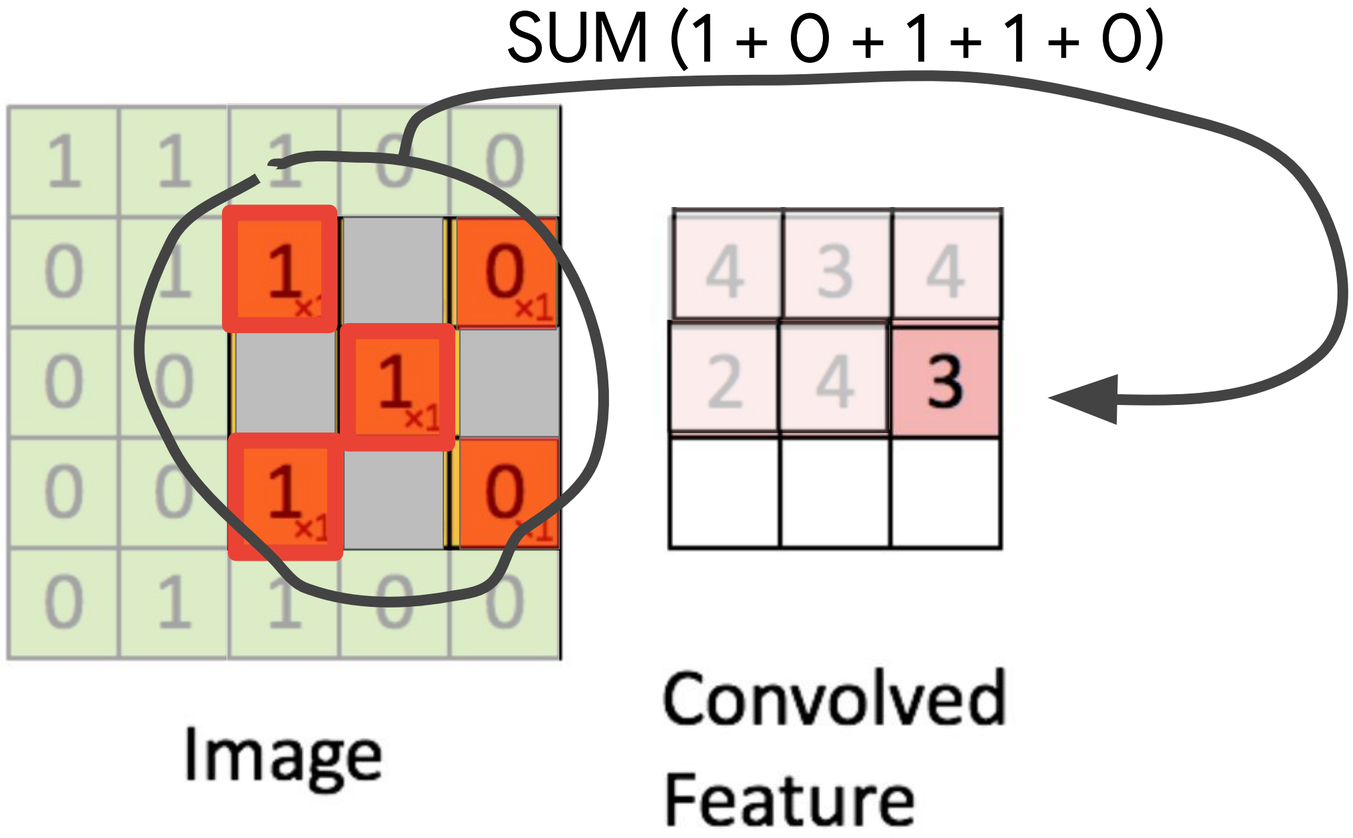
A convolved feature map is created by multiplying the kernel weights by the features



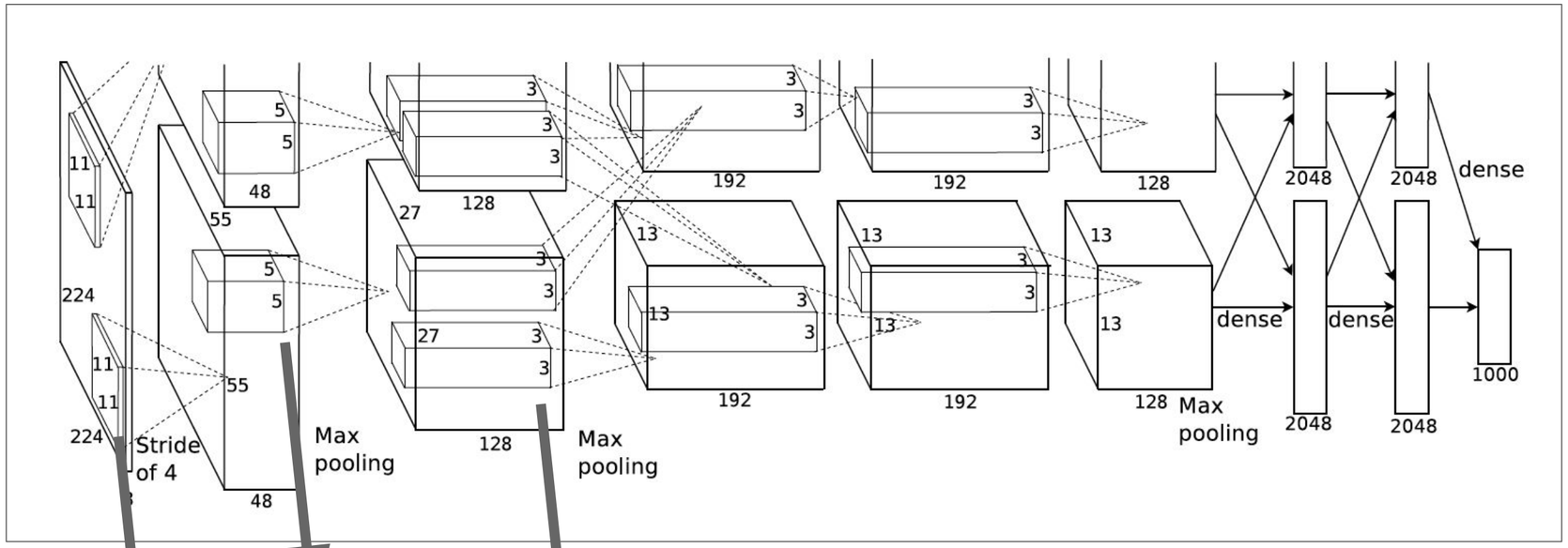
A convolved feature map is created by multiplying the kernel weights by the features



A convolved feature map is created by multiplying the kernel weights by the features



Convolutional Neural Networks commonly use multiple kernels



11x11

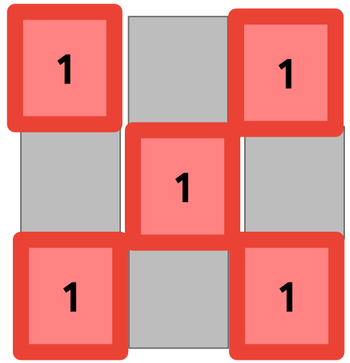
5x5

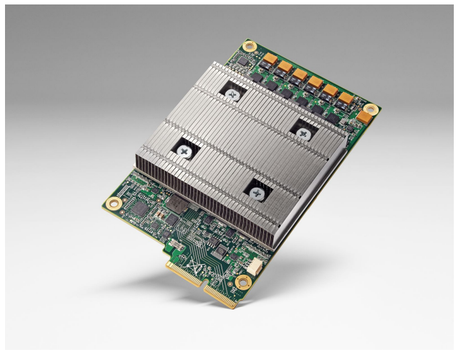
3x3

AlexNet CNN Architecture 2012

The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches
https://arxiv.org/abs/1803.01164

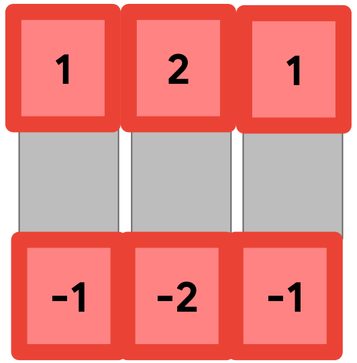
Convolutional Neural Networks commonly use multiple kernels



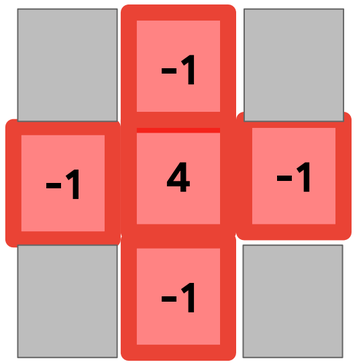


TPU

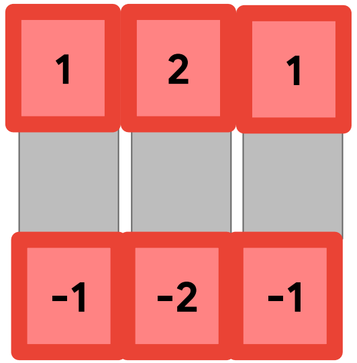
Convolutional Neural Networks commonly use multiple kernels



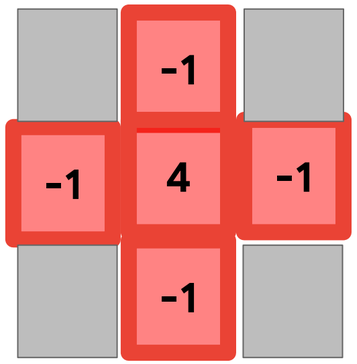
Detects horizontal edges



Convolutional Neural Networks commonly use multiple kernels

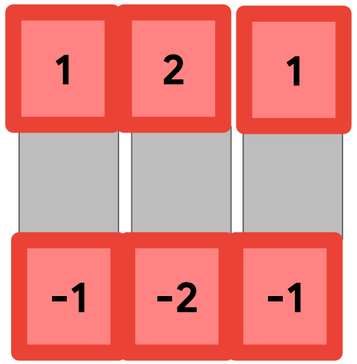


Detects horizontal edges

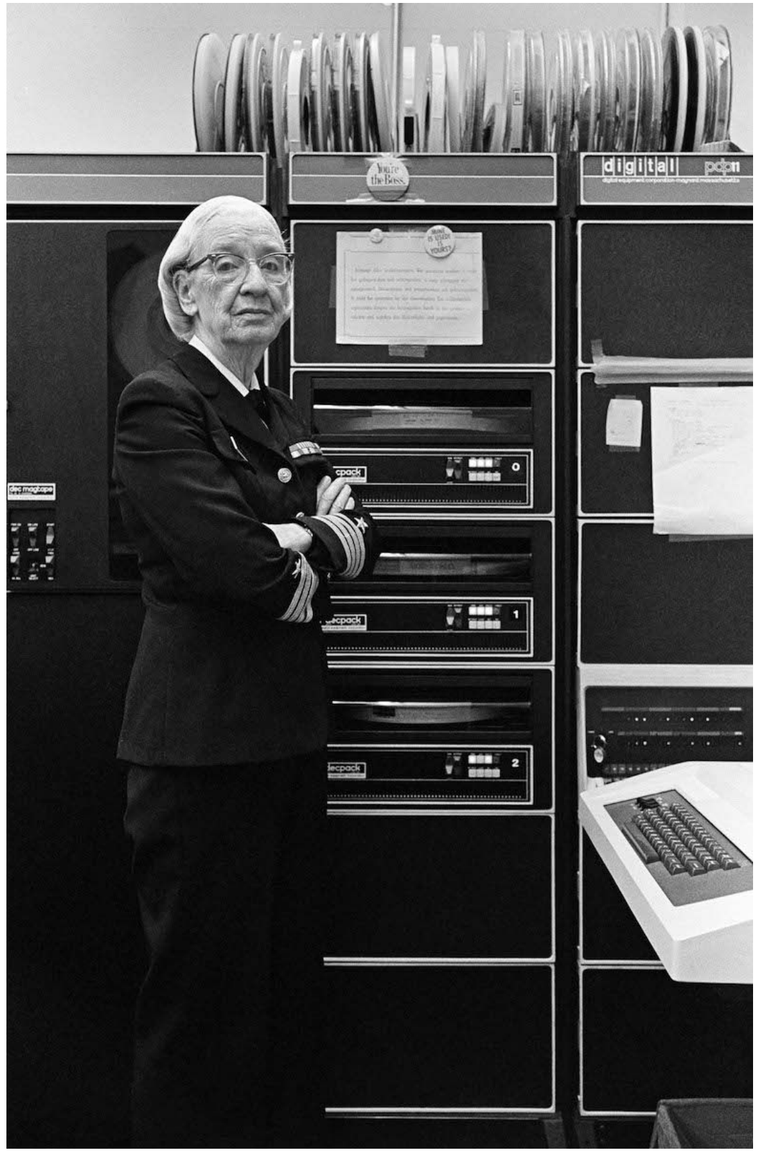


Detects bright spots

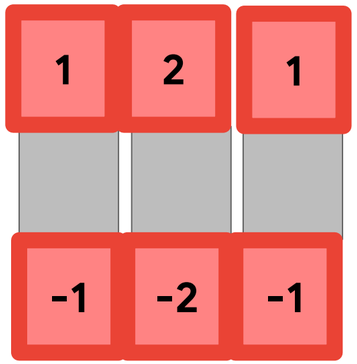
Different weights
detect different features



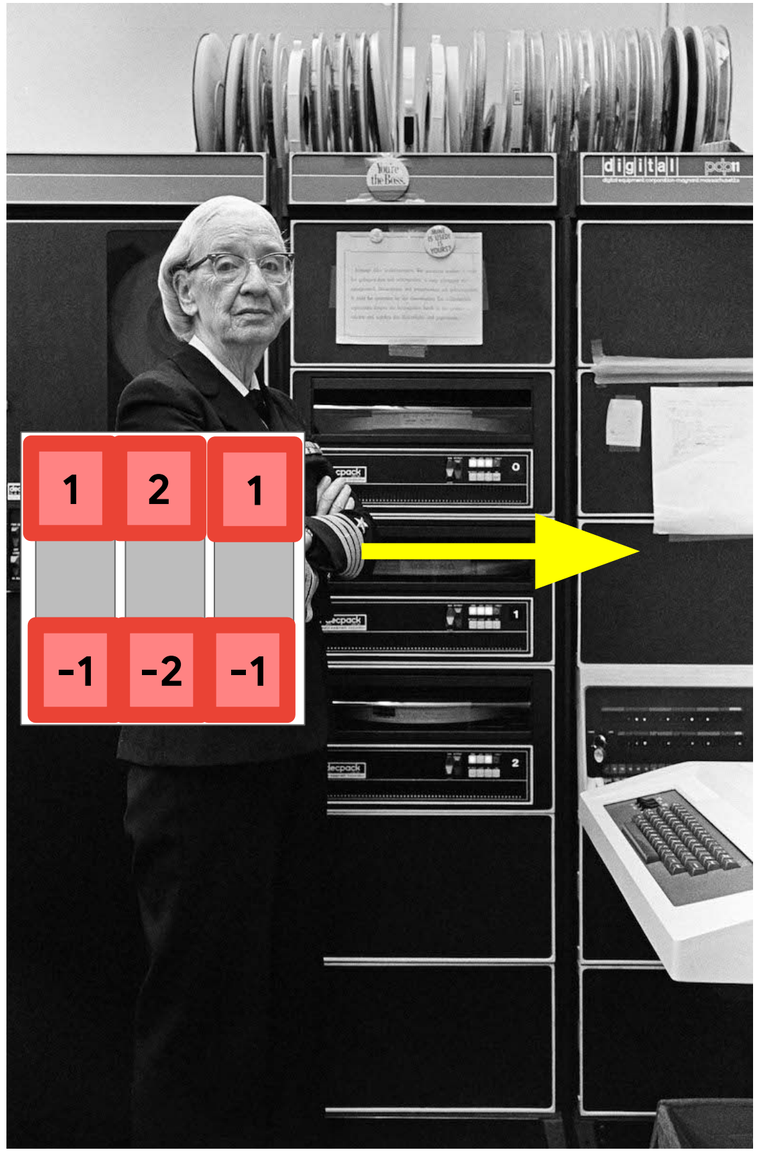
Detects
horizontal edges

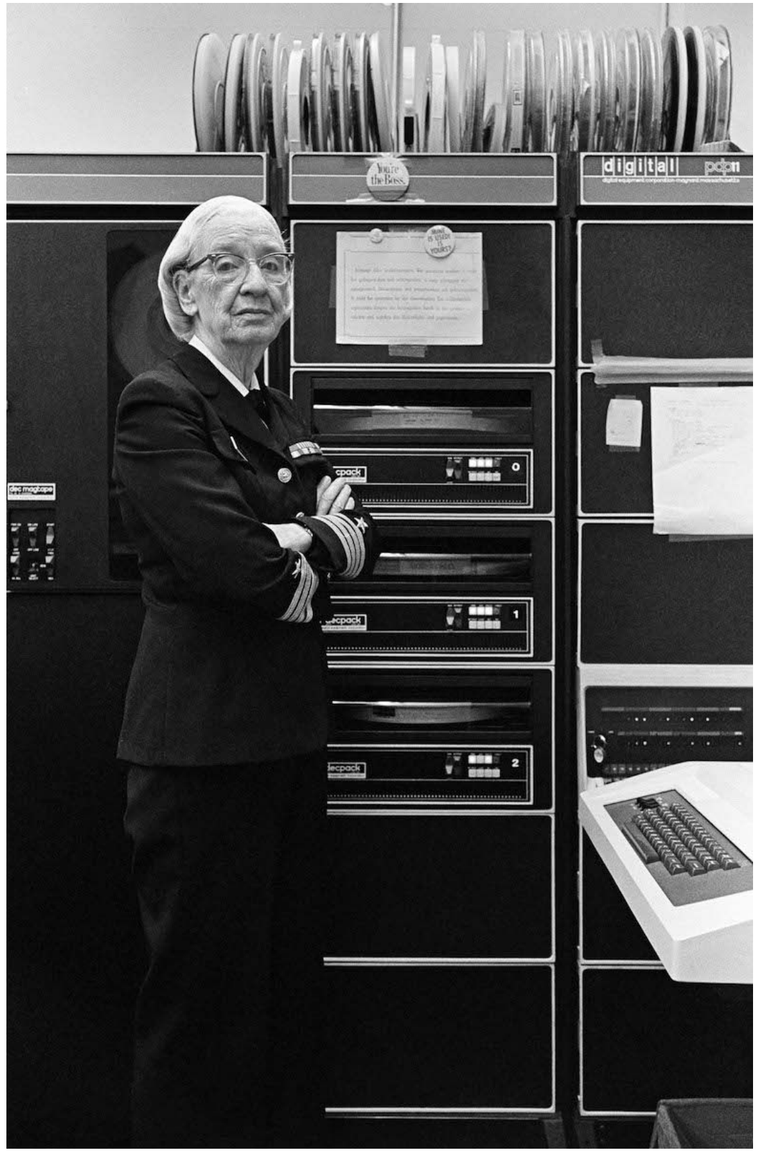
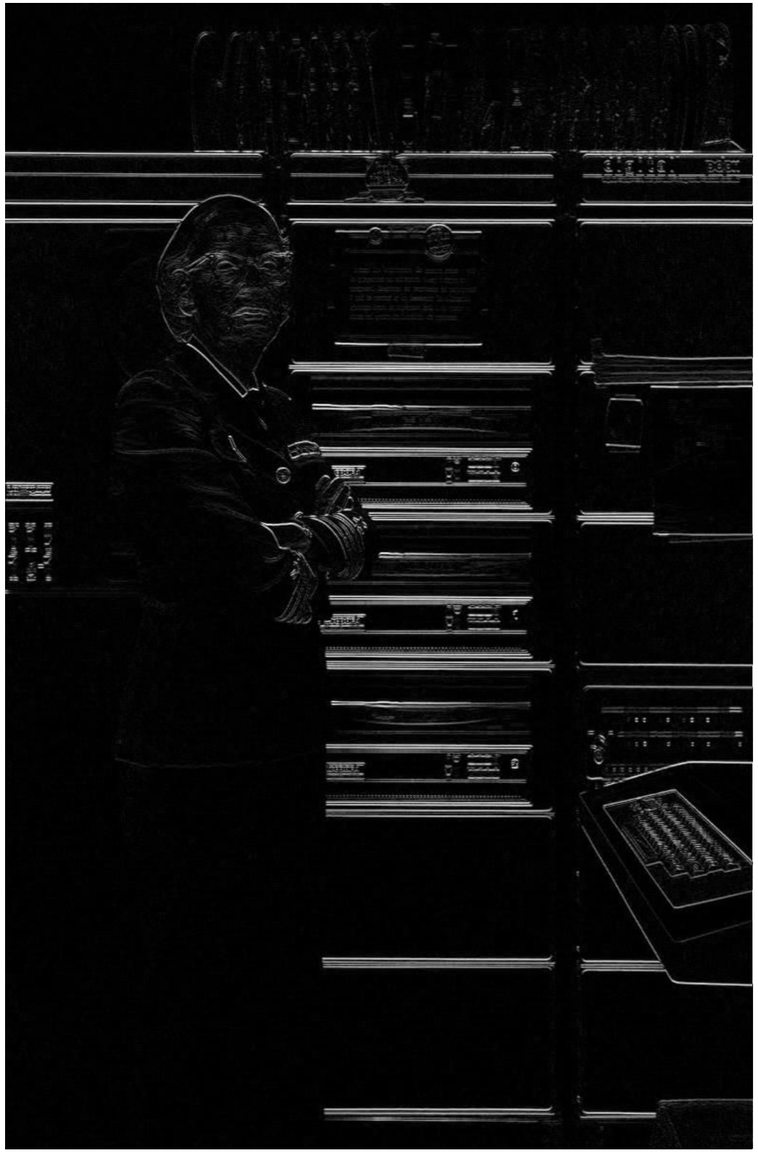


Different weights
detect different features

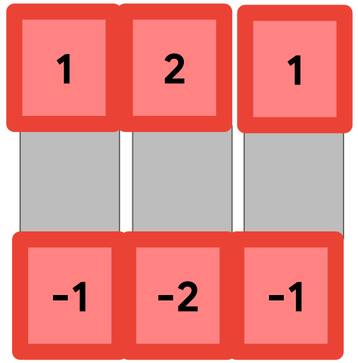


Detects
horizontal edges

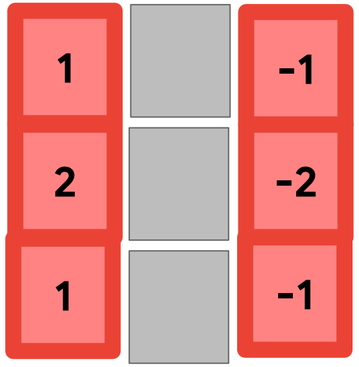



digital
pdp-11
You're the Boss
cpack
0
1
2


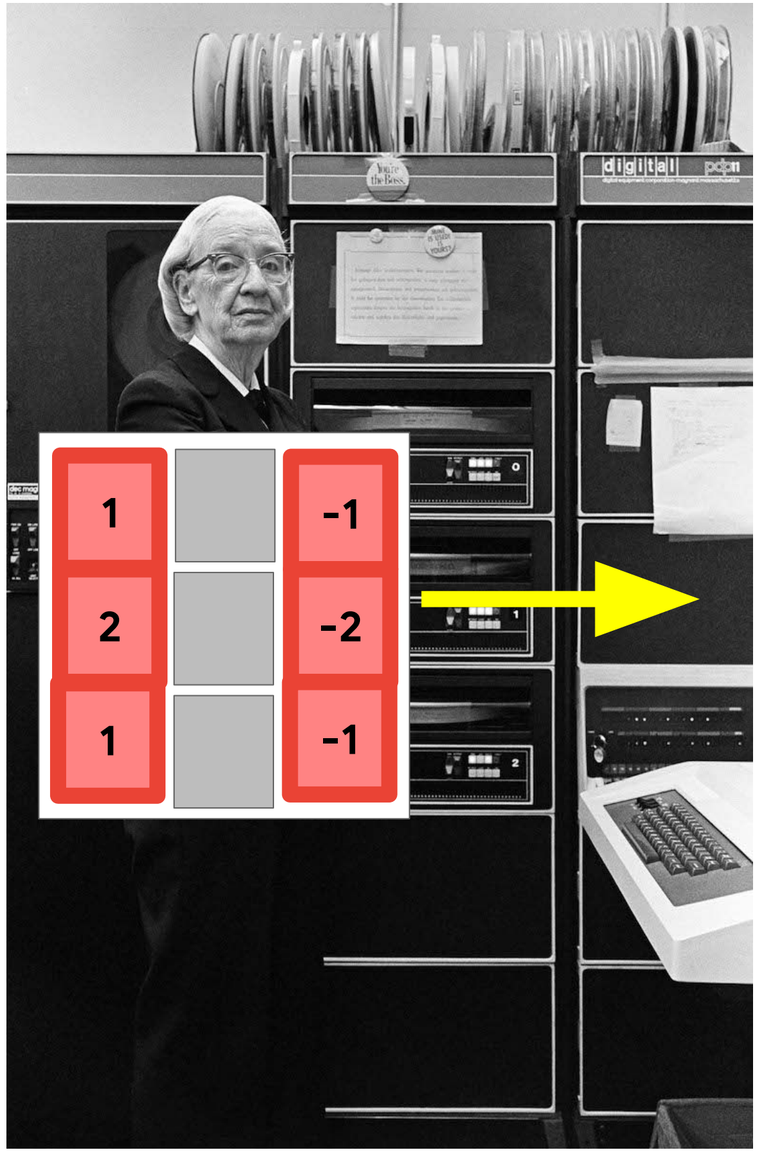
Different weights
detect different features

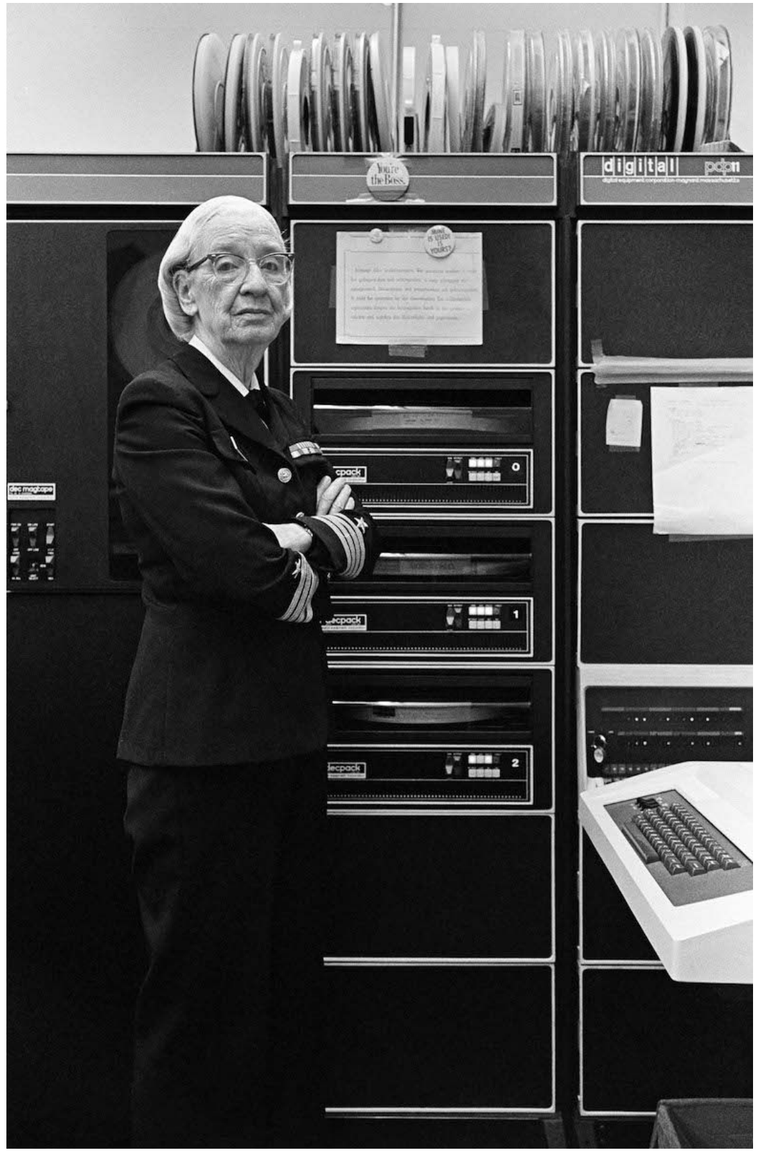


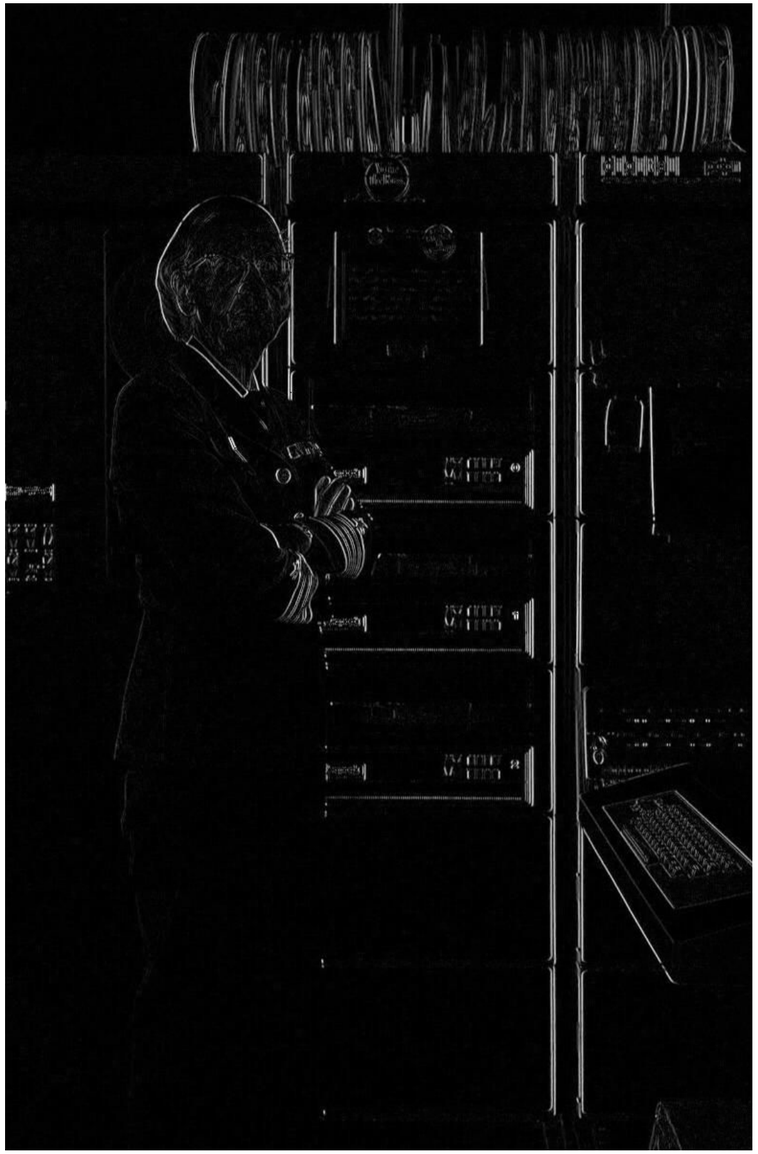
Detects
horizontal edges



Detects
vertical edges

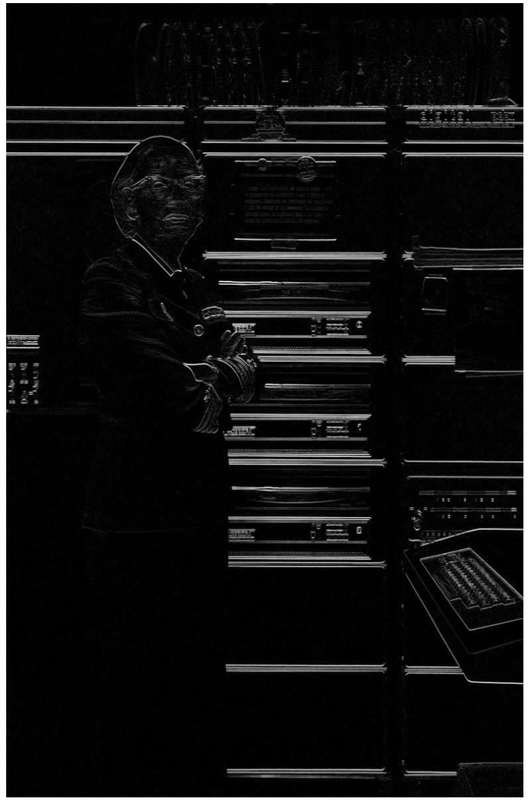


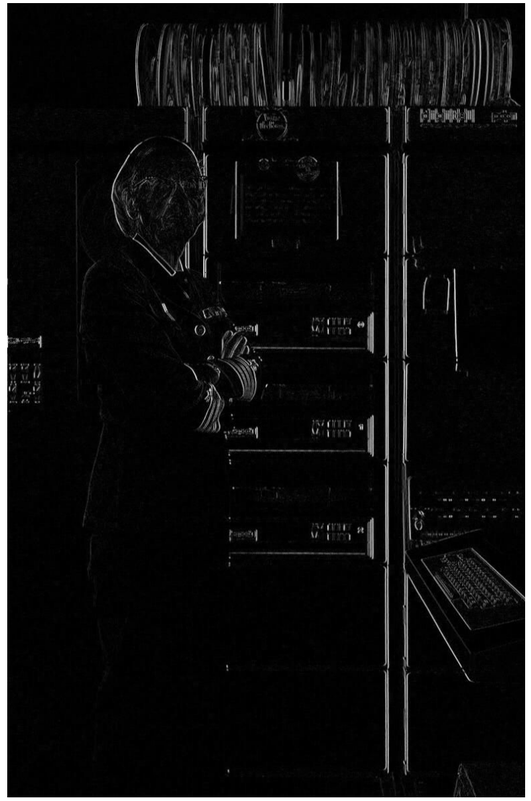

digital
pdp-11
You're the Boss
dec magtape
tecpack
0
1
2



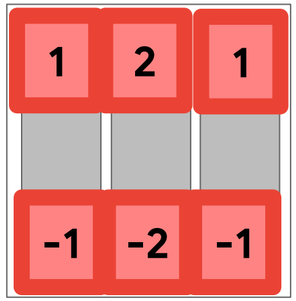
1	2	1
-1	-2	-1

1		-1
2		-2
1		-1

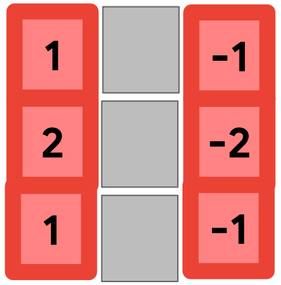




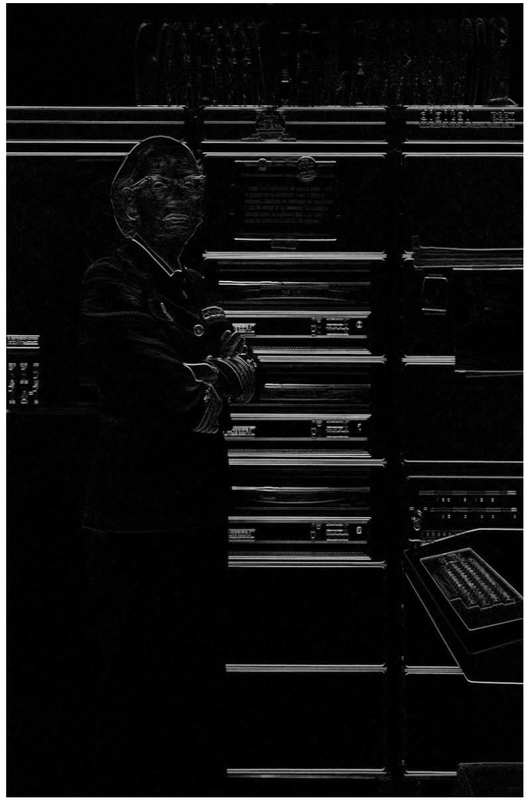
Kernel 1

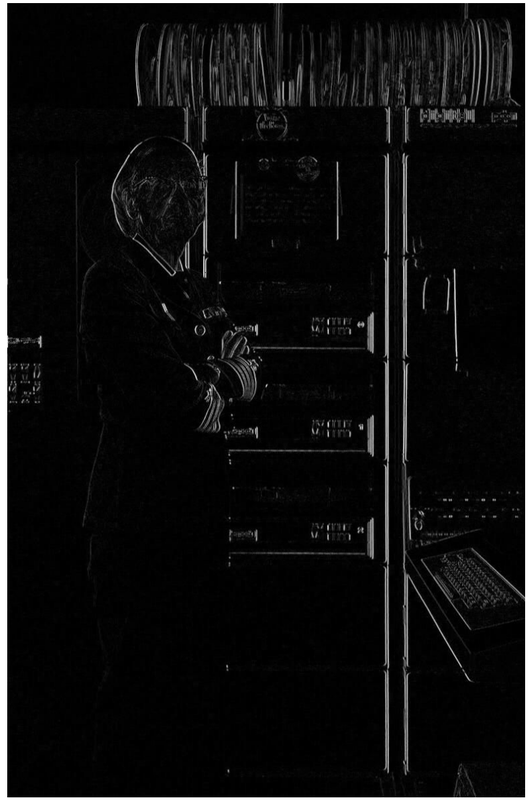


Kernel 2

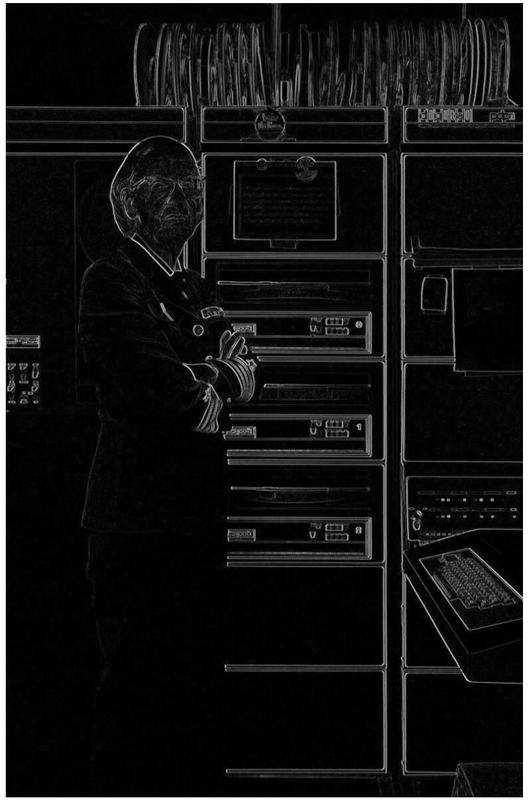


Sum of the two filter outputs

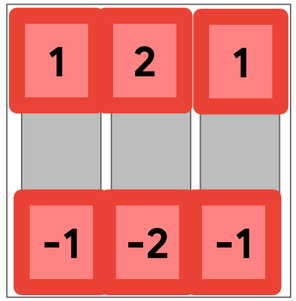




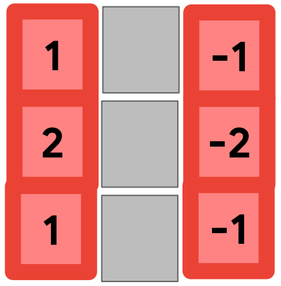
=



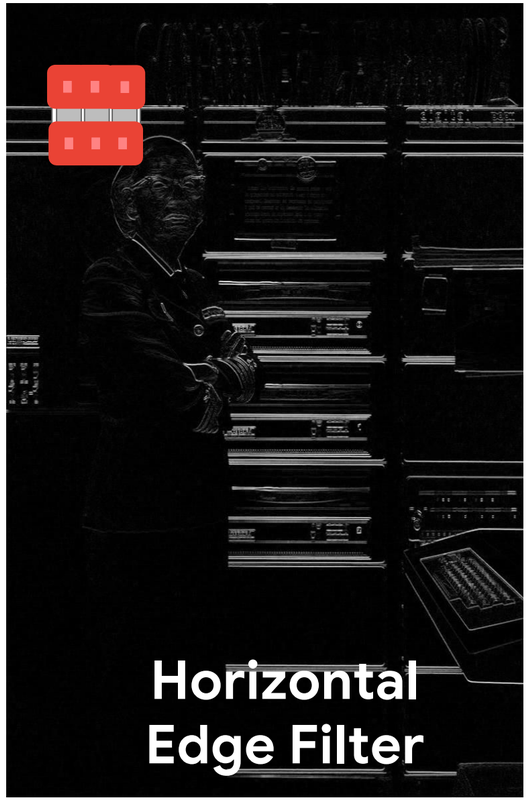
Kernel 1

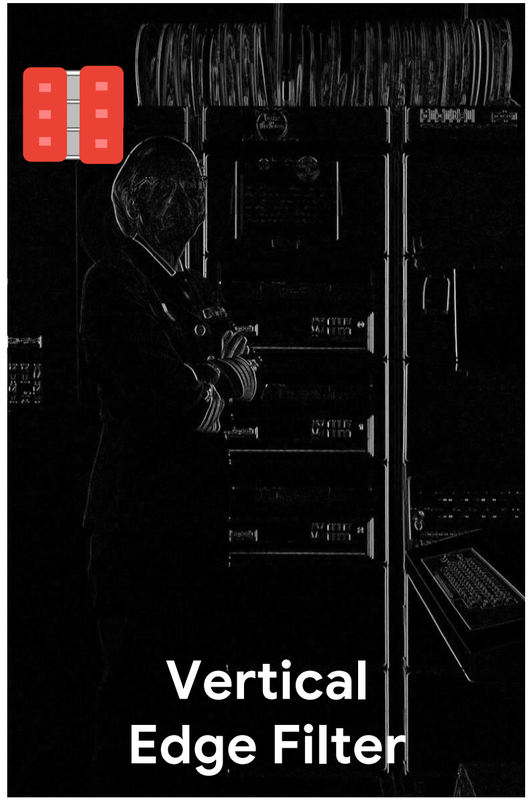


Kernel 2

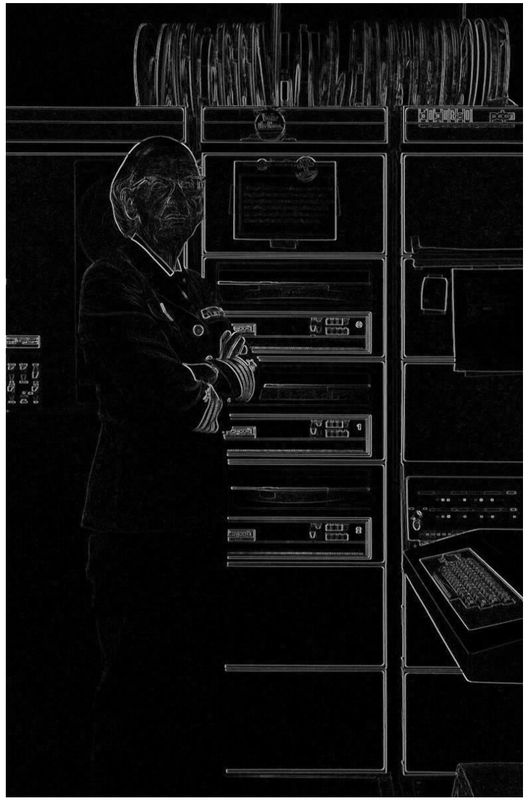


Sum of the two filter outputs

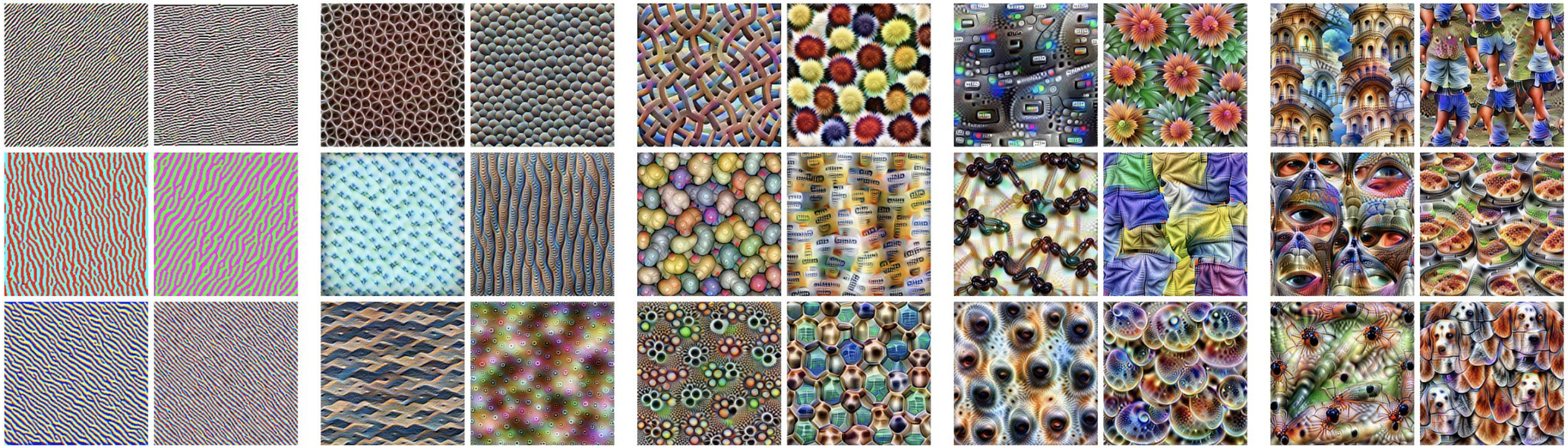




=



CNNs can learn a hierarchy of features

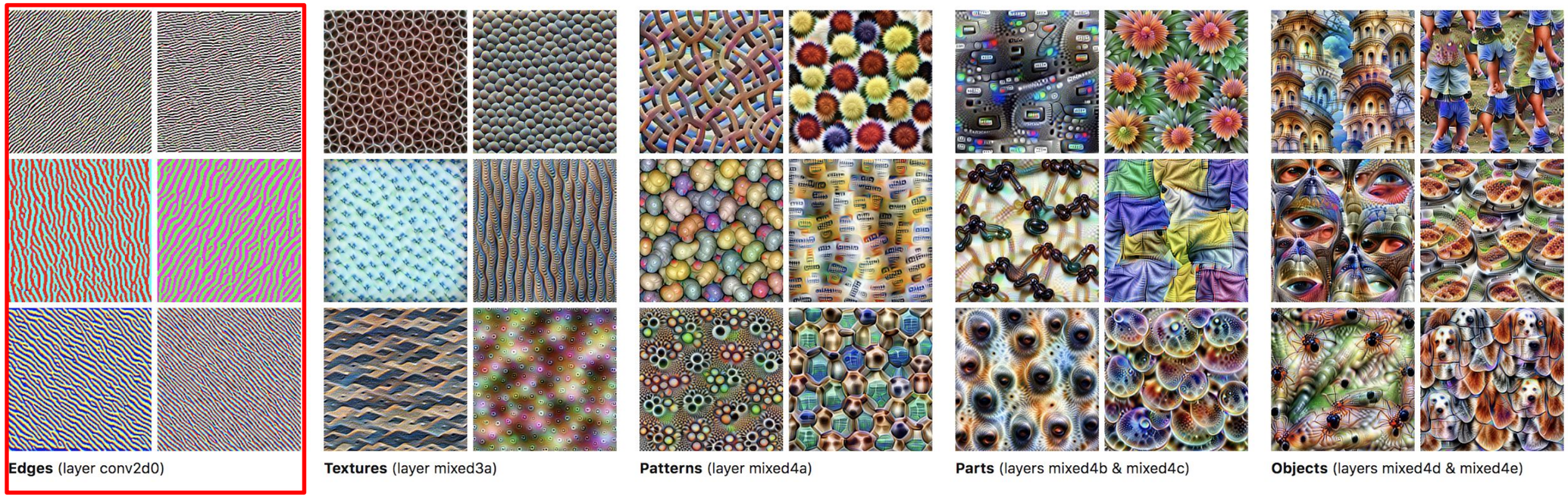


Edges (layer conv2d0) **Textures** (layer mixed3a) **Patterns** (layer mixed4a) **Parts** (layers mixed4b & mixed4c) **Objects** (layers mixed4d & mixed4e)

Flow of data in the network

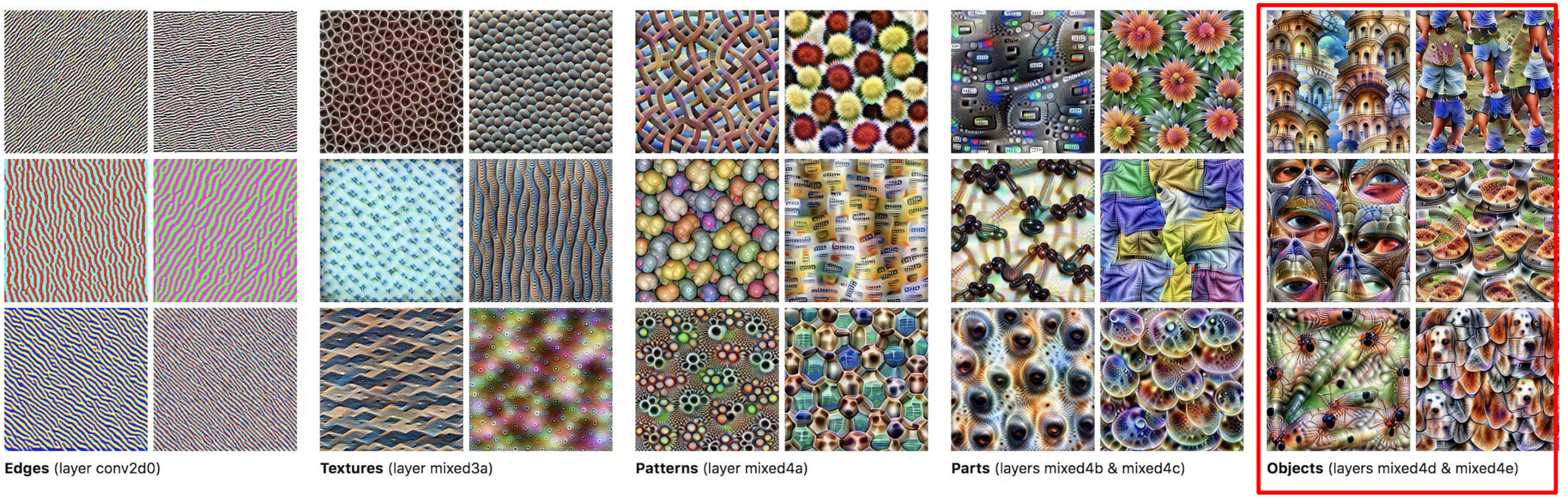
https://distill.pub/2017/feature-visualization/

CNNs can learn a hierarchy of features



https://distill.pub/2017/feature-visualization/

CNNs can learn a hierarchy of features



https://distill.pub/2017/feature-visualization/

Quiz:

What is the process of "sliding" a kernel across an image called?

1. Convolution
2. Confusion matrix
3. Contortion
4. Curved detection

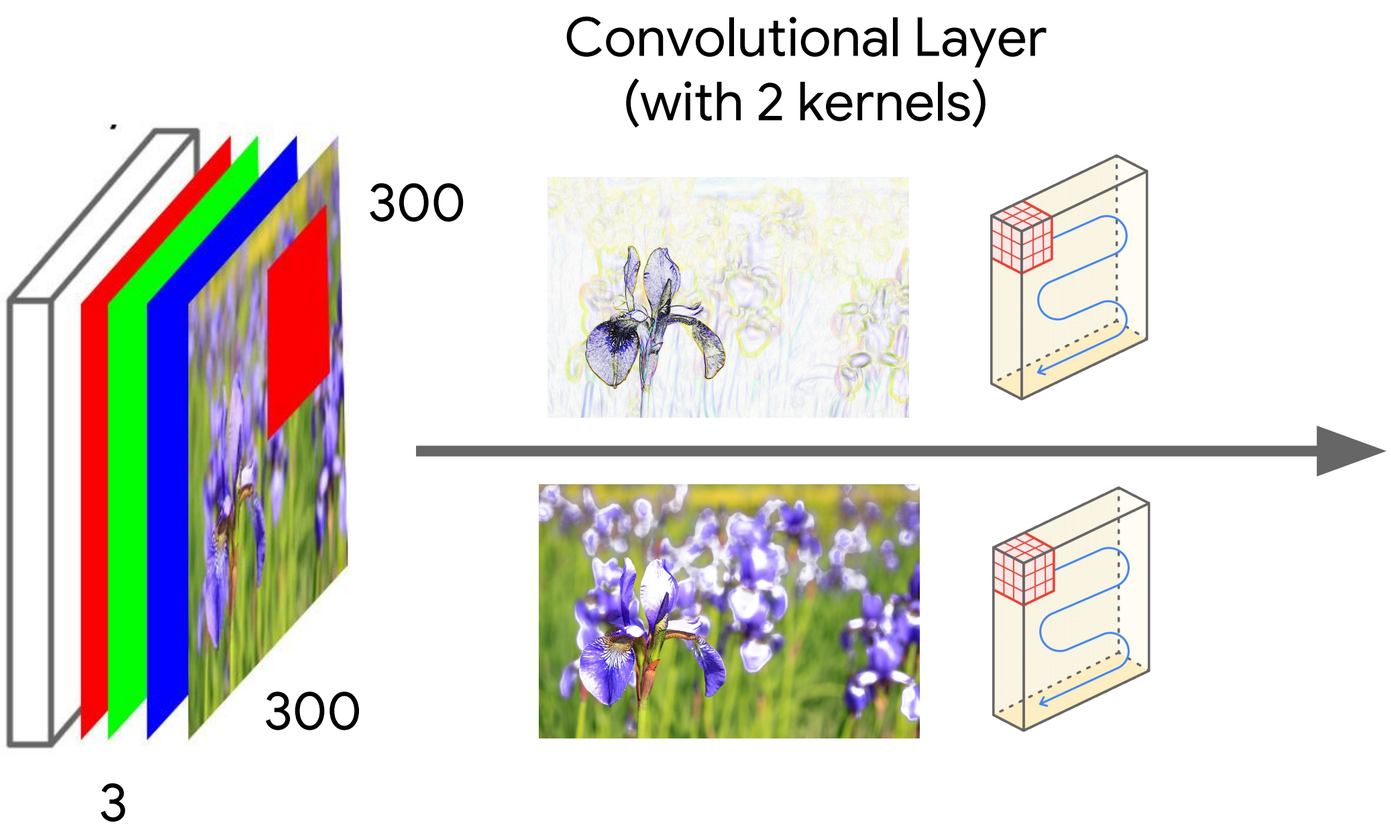
Quiz:

What is the process of "sliding" a kernel across an image called?

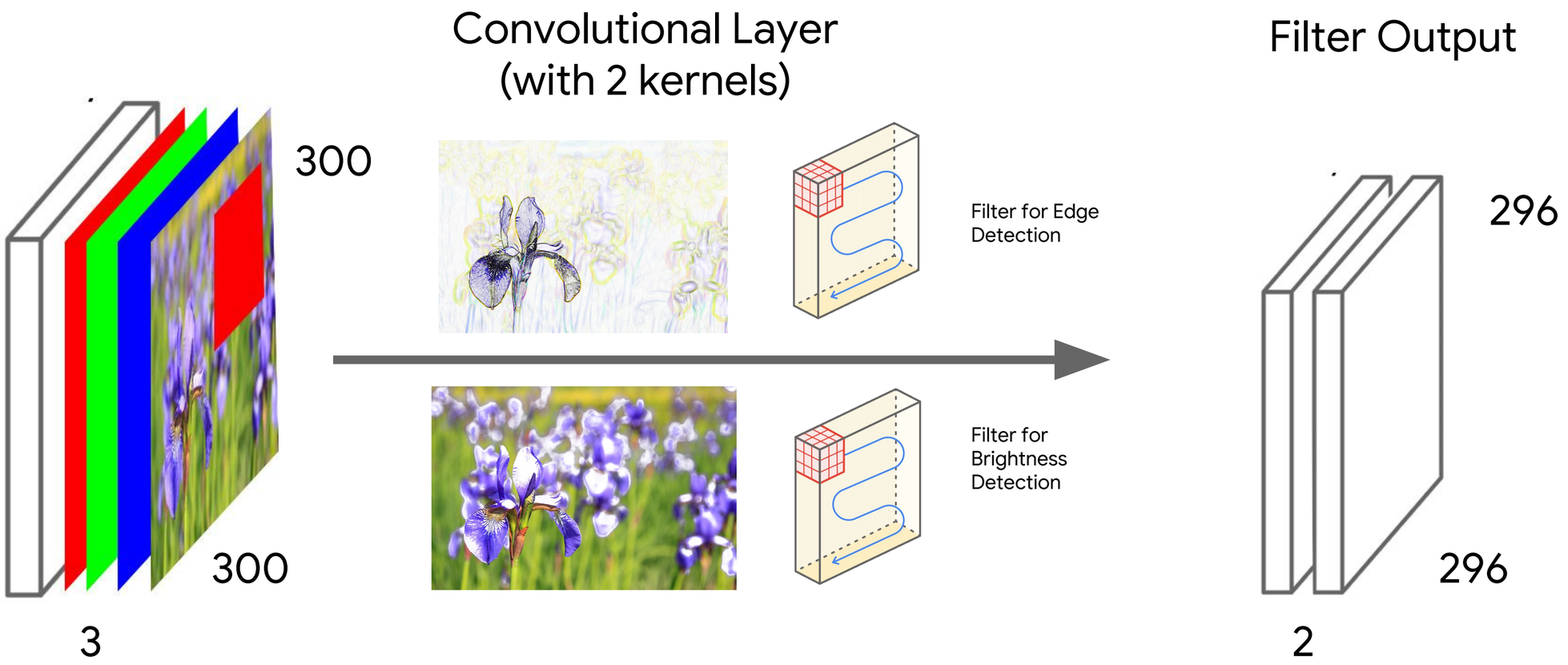
1. **Convolution**
2. Confusion matrix
3. Contortion
4. Curved detection

Convolutional Layers are collections of filters

Convolutional Layers are collections of filters



Convolutional Layers are collections of filters

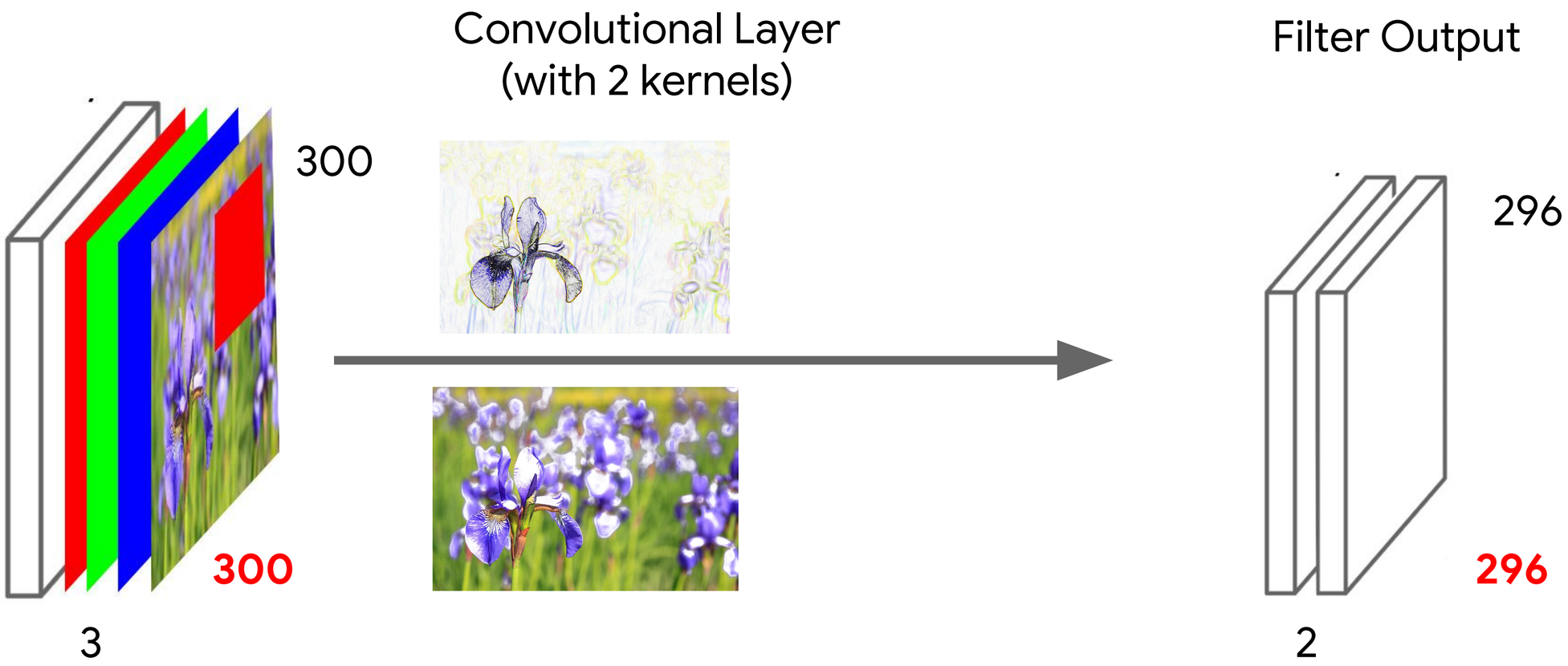


File Name: T-ICML-O_2_l3_cnn_parameters

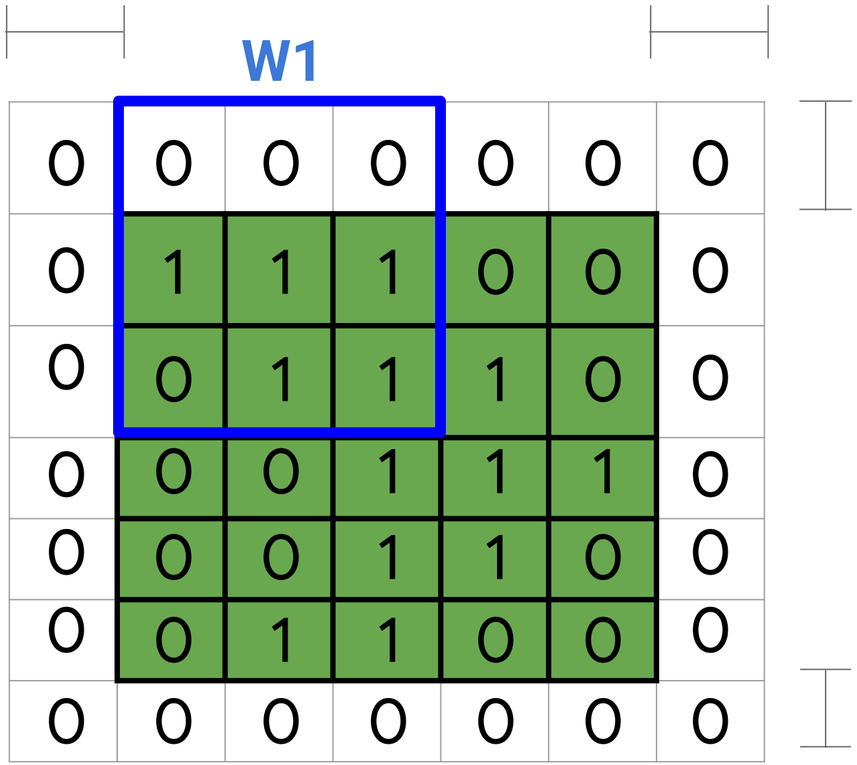
Format: Presenter in Studio

Presenter: Carl Osipov

Convolutional Layers are collections of filters



Padding preserves the shape of the input after the convolution



"Same" padding for 3x3 kernel size

TensorFlow provides a high level API to set up a convolutional layer

```
tf.layers.conv2d(
     inputs=...,              # [batch_size, filter_height, filter_width, in_channels]
     filters=...,             # number of filters, i.e. out_channels
     kernel_size=3,           # size of the kernel, e.g. 3 for a 3x3 kernel
     padding='same'           # maintain the same shape across the input and output
    ...
 )
```

TensorFlow provides a high level API to set up a convolutional layer

```
tf.layers.conv2d(
     inputs=...,              # [batch_size, filter_height, filter_width, in_channels]
     filters=...,             # number of filters, i.e. out_channels
     kernel_size=3,           # size of the kernel, e.g. 3 for a 3x3 kernel
     padding='same'           # maintain the same shape across the input and output
    ...
 )
```

TensorFlow provides a high level API to set up a convolutional layer

```
tf.layers.conv2d(
     inputs=...,              # [batch_size, filter_height, filter_width, in_channels]
     filters=...,             # number of filters, i.e. out_channels
     kernel_size=3,           # size of the kernel, e.g. 3 for a 3x3 kernel
     padding='same'           # maintain the same shape across the input and output
    ...
 )
```

TensorFlow provides a high level API to set up a convolutional layer

```
tf.layers.conv2d(
     inputs=...,              # [batch_size, filter_height, filter_width, in_channels]
     filters=...,             # number of filters, i.e. out_channels
     kernel_size=3,           # size of the kernel, e.g. 3 for a 3x3 kernel
     padding='same'           # maintain the same shape across the input and output
    ...
 )
```

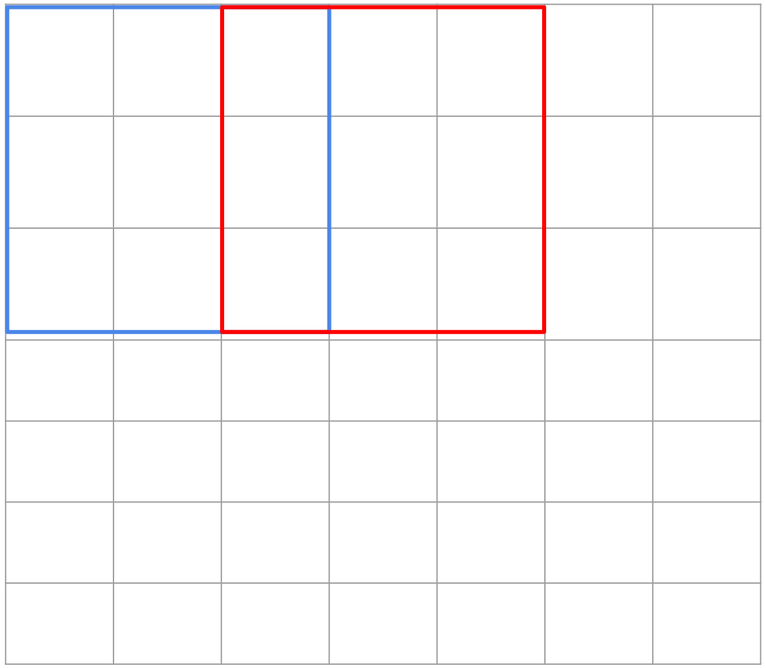
TensorFlow provides a high level API to set up a convolutional layer

```
tf.layers.conv2d(
     inputs=...,              # [batch_size, filter_height, filter_width, in_channels]
     filters=...,             # number of filters, i.e. out_channels
     kernel_size=3,           # size of the kernel, e.g. 3 for a 3x3 kernel
     padding='same'           # maintain the same shape across the input and output
    ...
 )
```

TensorFlow provides a high level API to set up a convolutional layer

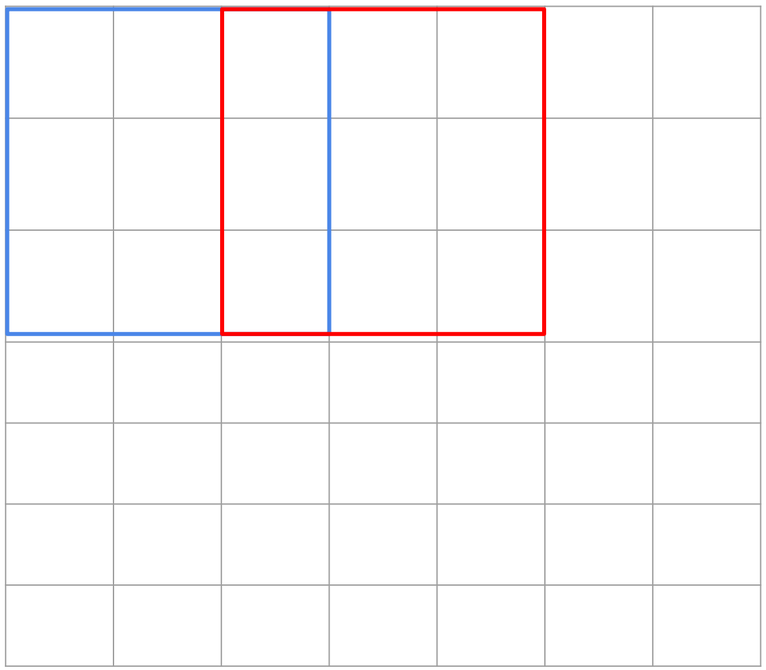
```
tf.layers.conv2d(
     inputs=...,              # [batch_size, filter_height, filter_width, in_channels]
     filters=...,             # number of filters, i.e. out_channels
     kernel_size=3,           # size of the kernel, e.g. 3 for a 3x3 kernel
     padding='same'           # maintain the same shape across the input and output
    ...
 )
```

Dimensionality reduction decreases the total # of parameters and computation time



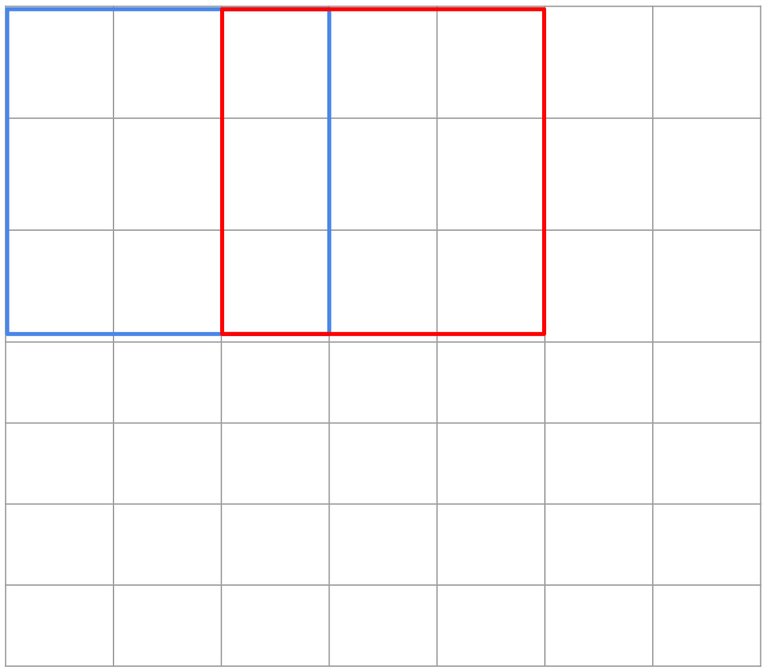
Dimensionality reduction decreases the total # of parameters and computation time

stride=2

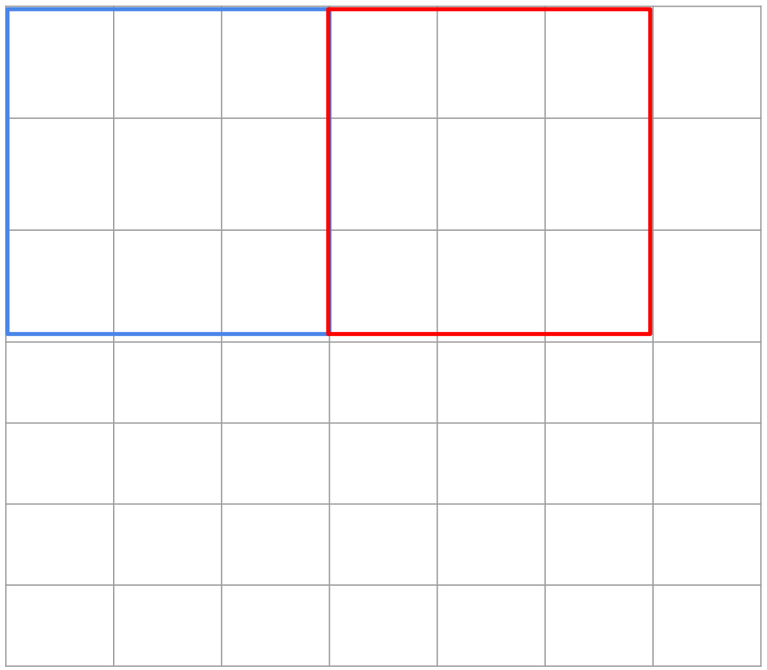


Dimensionality reduction decreases the total # of parameters and computation time

stride=2



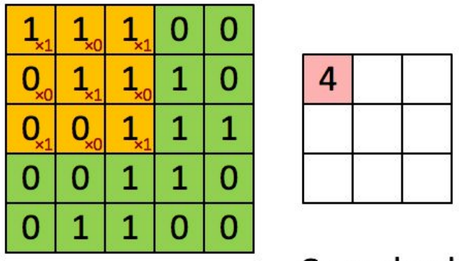
stride=3



Quiz:

What does the green part of the below diagram represent?

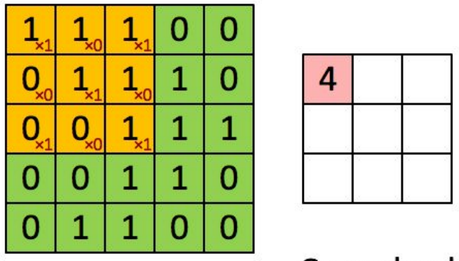
1. The kernel
2. The filter output
3. The original image



Quiz:

What does the green part of the below diagram represent?

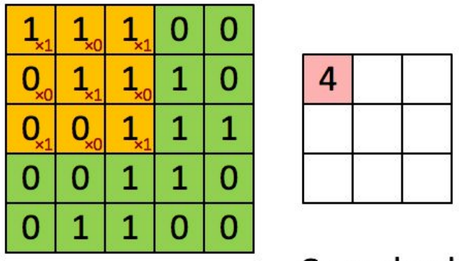
1. The kernel
2. The filter output
3. **The original image**



Quiz:

What does the yellow part of the below diagram represent?

1. The kernel convolving
2. The filter output
3. The original image



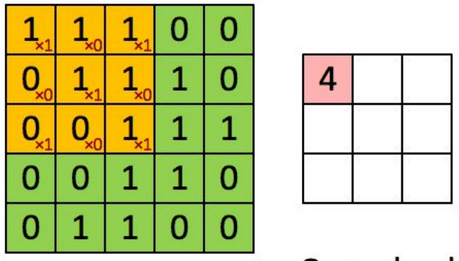
Image

Convolved Feature

Quiz:

What does the yellow part of the below diagram represent?

1. **The kernel convolving**
2. The filter output
3. The original image



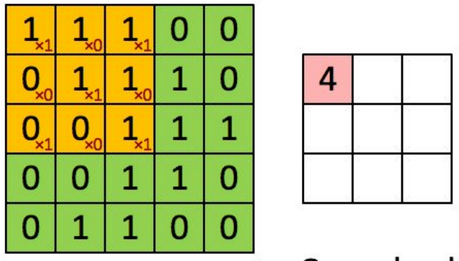
Image

Convolved Feature

Quiz:

What is the size of the stride step in the below diagram?

1. Stride of 3
2. Stride of 2
3. Stride of 1
4. Unknown



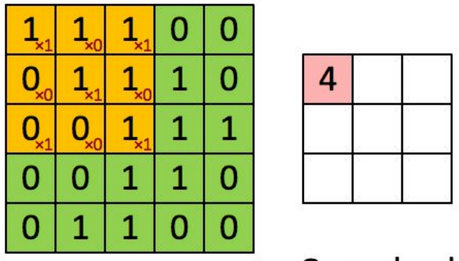
Image

Convolved Feature

Quiz:

What is the size of the stride step in the below diagram?

1. Stride of 3
2. Stride of 2
3. **Stride of 1**
4. Unknown



Image

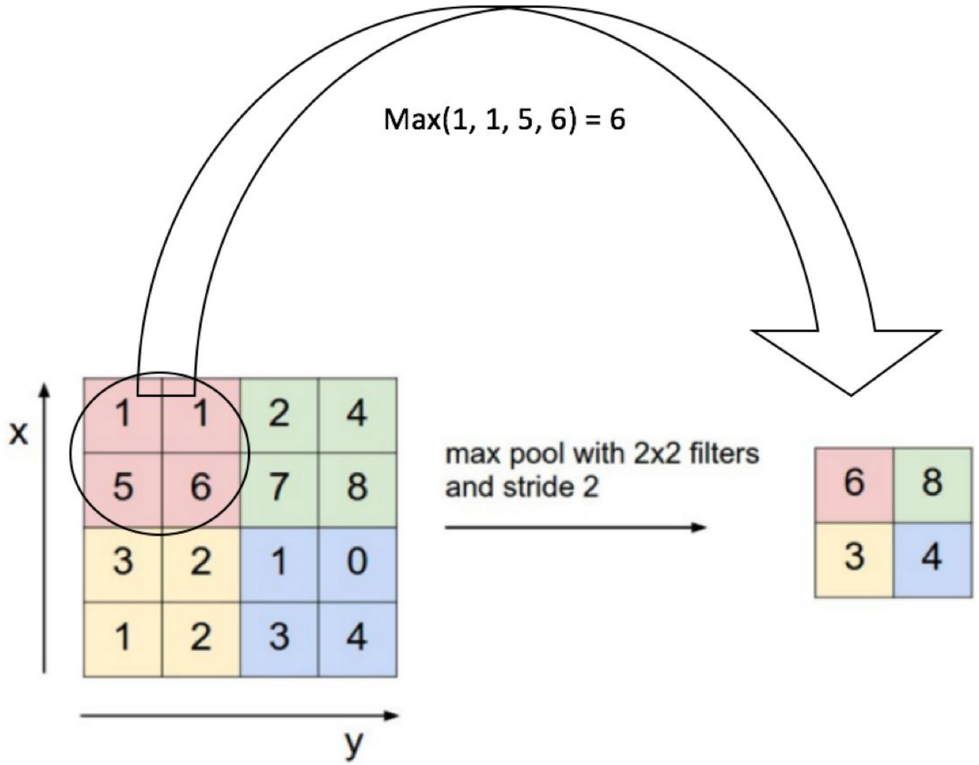
Convolved Feature

File Name: T-ICML-O_2_l4_pooling_layers

Format: Presenter in Studio

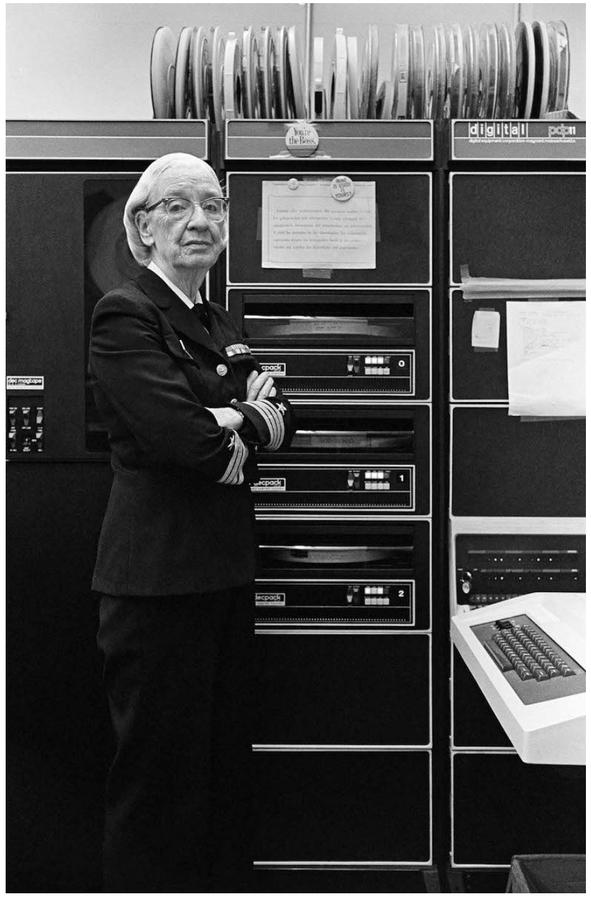
Presenter: Carl Osipov

Maxpooling also reduces dimensionality



Rectified Feature Map

Maxpooling example



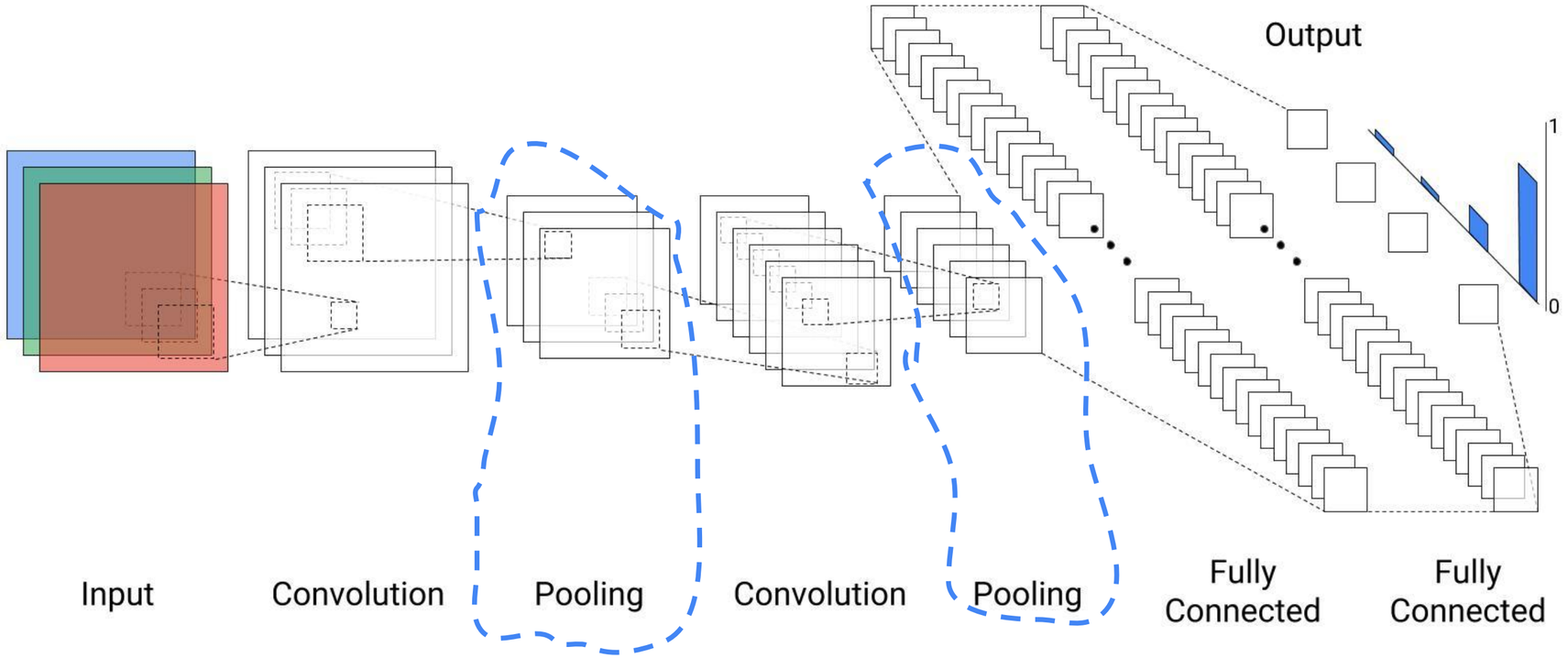
Brighter values correspond to larger pixel values

Maxpool

Kernel: 2x2
Stride: 2

Dimensionality is decreased by a factor of 2 both horizontal and vertically

Maxpooling operations are just additional layers in our network



TensorFlow provides the tf.layers.max_pooling2d API

```
tf.layers.max_pooling2d(

        inputs,           # [batch_size, filter_height, filter_width, in_channels]
        pool_size=2,      # size of the maxpooling kernel
        strides=2         # size of the stride step

)
```

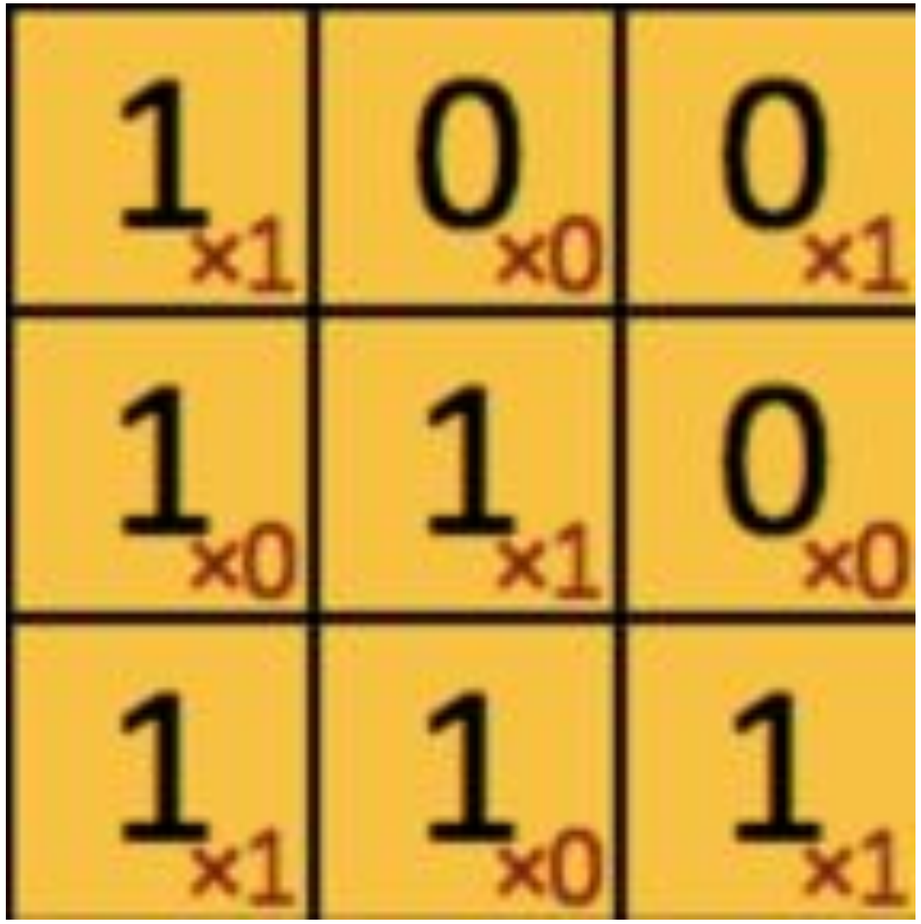
Quiz:

What do the smaller red numbers represent in this image you've seen before?

1. The weights of the particular kernel we are applying (i.e. what feature we are detecting)

2. The intensity of the pixel for that area of the original image

3. The channel depth of the image



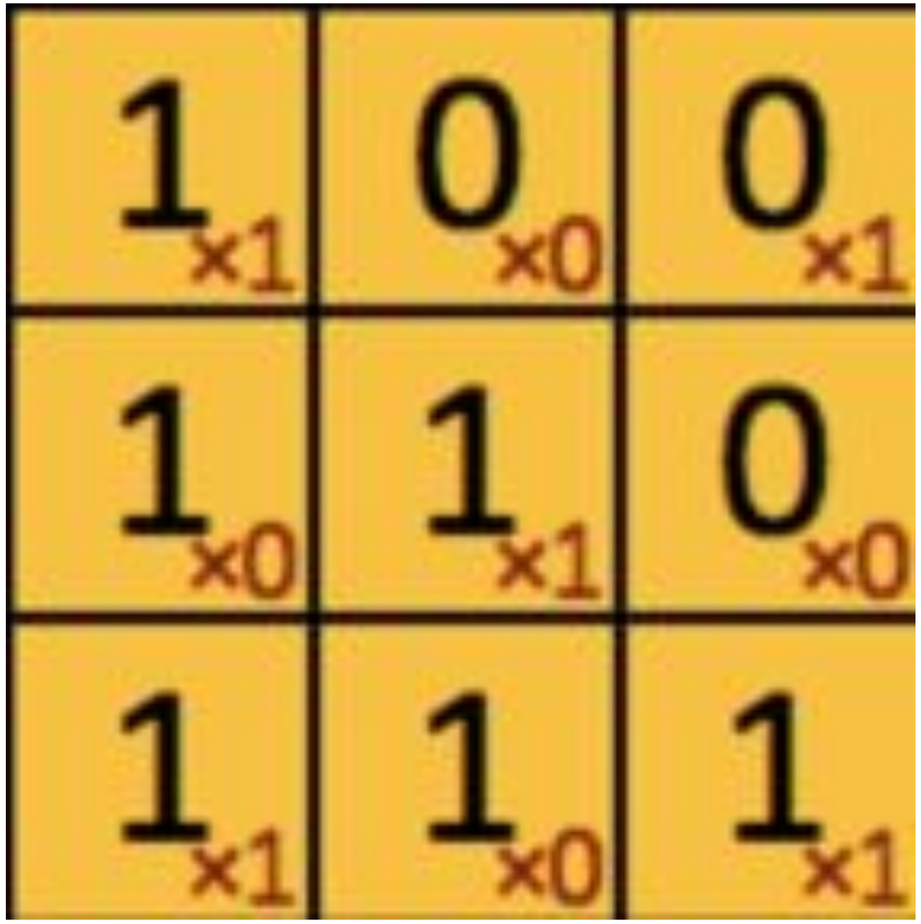
Quiz:

What do the smaller red numbers represent in this image you've seen before?

1. **The weights of the particular kernel we are applying (i.e. what feature we are detecting)**

2. The intensity of the pixel for that area of the original image

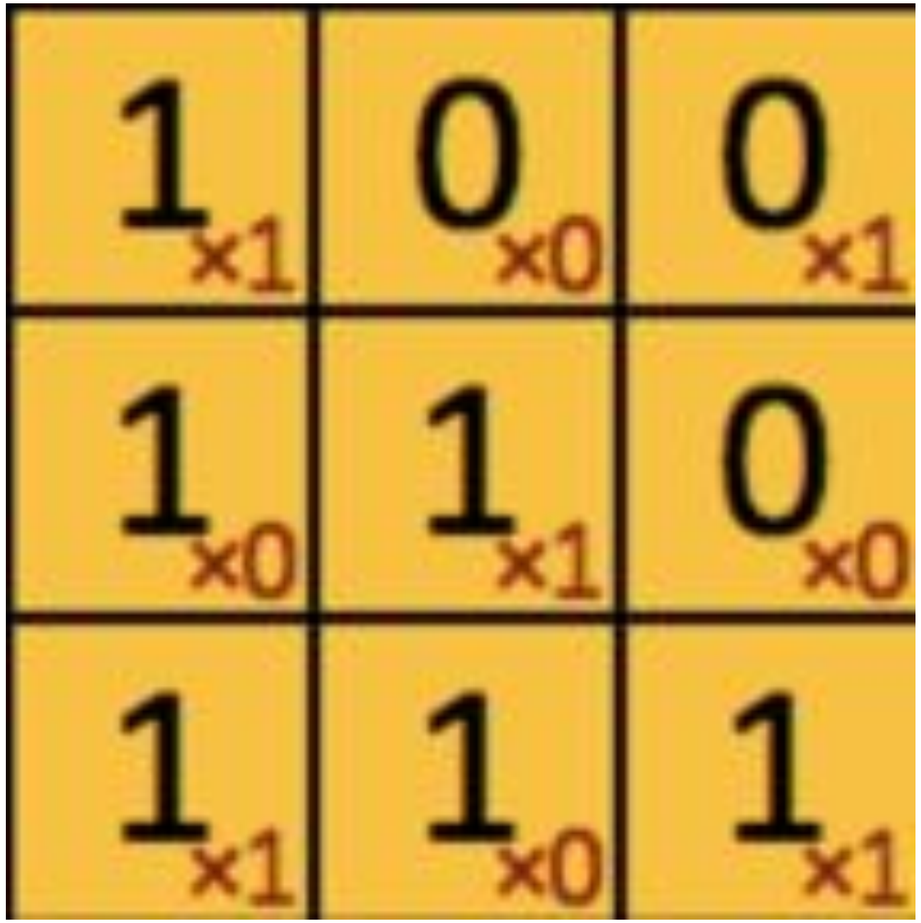
3. The channel depth of the image



Quiz:

Given the above, what the value of the output that the kernel will generate for this part of the image?

- 0
- 1
- 2
- 3
- 4
- 5



Quiz:

Given the above, what the value of the output that the kernel will generate for this part of the image?

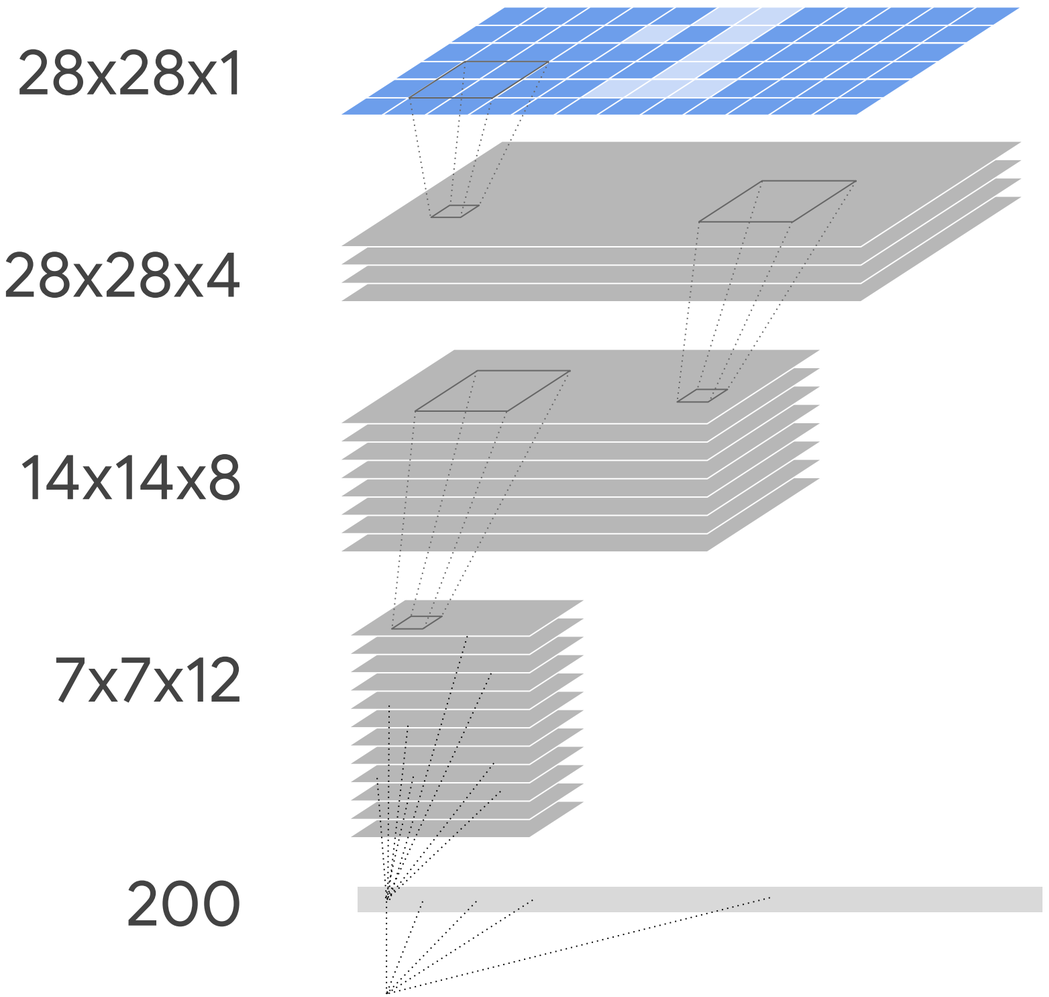
- 0
- 1
- 2
- 3
- **4**
- 5

File Name: T-ICML-O_2_l5_implementing_cnns

Format: Presenter in Studio

Presenter: Carl Osipov

Successive convolution layers apply filters to increasing scales



convolutional layer, 4 channels deep
W1[5, 5, 1, 4] stride 1

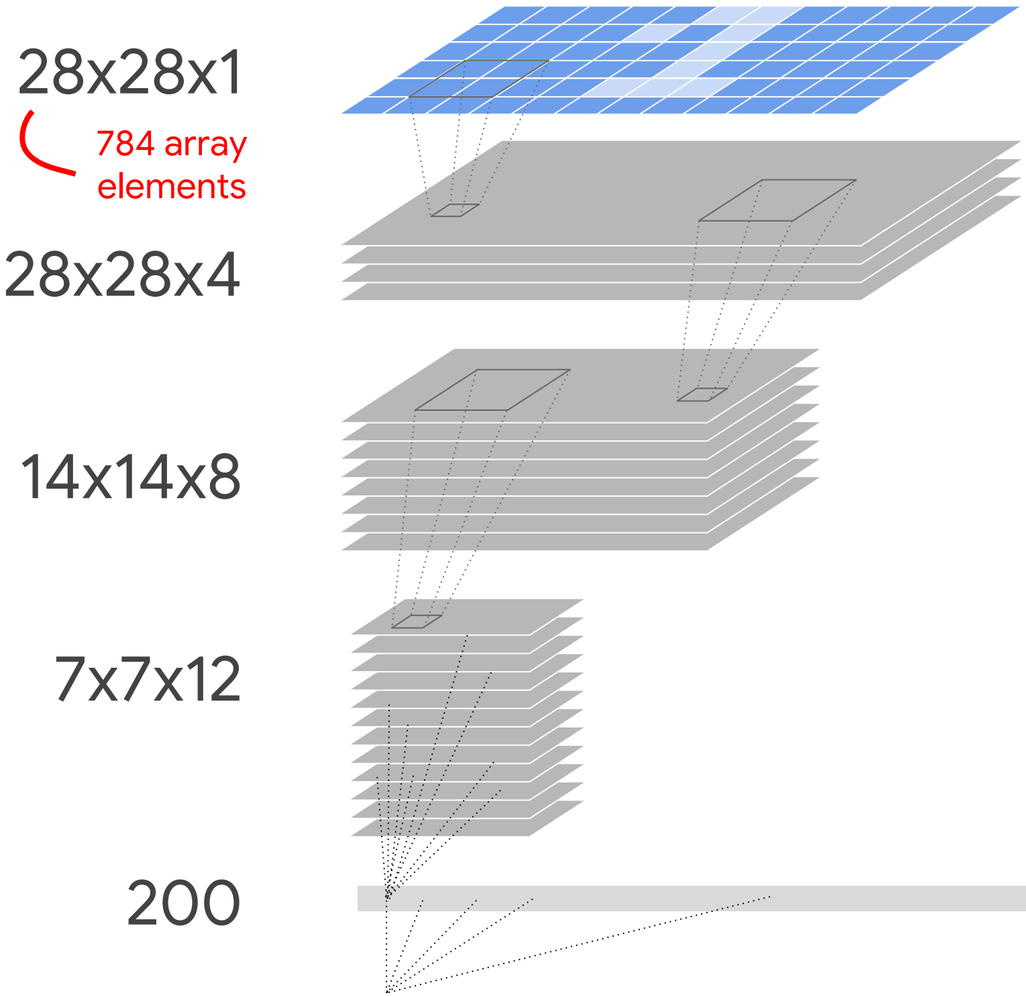
convolutional layer, 8 channels deep
W2[3, 3, 4, 8] stride 2

convolutional layer, 12 channels deep
W3[3, 3, 8, 12] stride 2

fully connected layer W4[7x7x12, 200]

softmax readout layer W5[200, 10]

Successive convolution layers apply filters to increasing scales



convolutional layer, 4 channels deep
W1[5, 5, 1, 4] stride 1

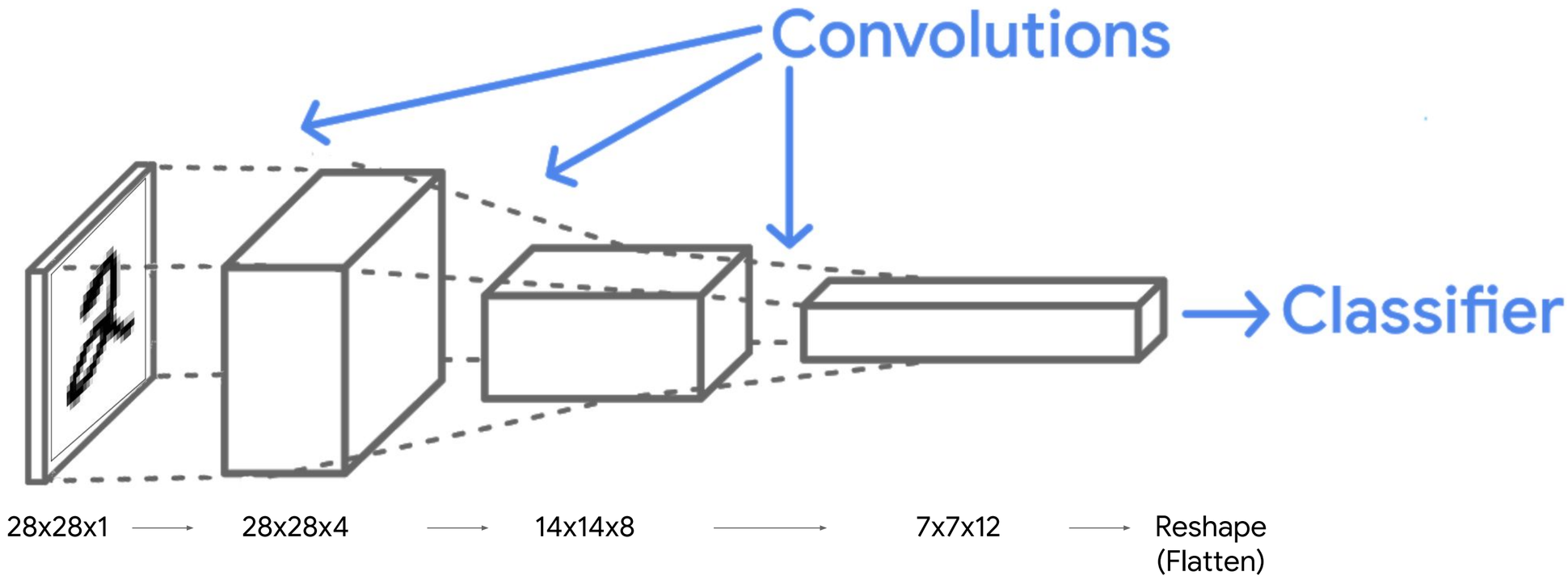
convolutional layer, 8 channels deep
W2[3, 3, 4, 8] stride 2

convolutional layer, 12 channels deep
W3[3, 3, 8, 12] stride 2

fully connected layer W4[7x7x12, 200]

softmax readout layer W5[200, 10]

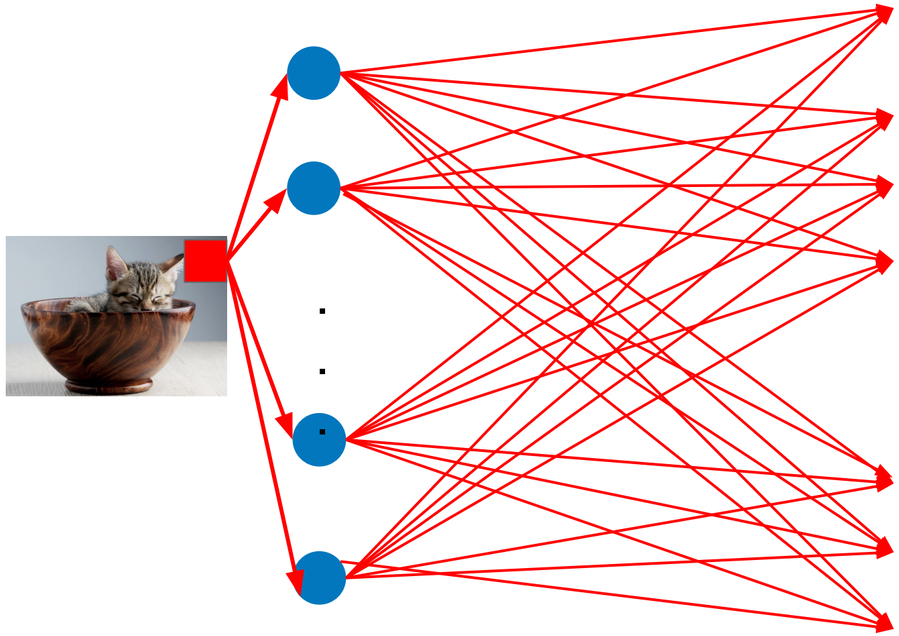
CNN model architecture for MNIST image



Setting up a model with Convolutional layers

```
def cnn_model(img, mode):
  X = tf.reshape(img, [-1, HEIGHT, WIDTH, 1]) # as a 2D image with one grayscale
channel

  c1 = tf.layers.conv2d(X, filters=4, kernel_size=5, strides=1,
                        padding='same', activation=tf.nn.relu) # ?x28x28x4

  c2 = tf.layers.conv2d(c1, filters=8, kernel_size=3, strides=2,
                        padding='same', activation=tf.nn.relu)# ?x14x14x8

  c3 = tf.layers.conv2d(c2, filters=12, kernel_size=3, strides=2,
                        padding='same', activation=tf.nn.relu)# ?x7x7x12

  c3flat = tf.reshape(c3, [-1, 7*7*12]) # flattened
  h3 = tf.layers.dense(c3flat, 200, activation=tf.nn.relu)
  ylogits = tf.layers.dense(h3, NCLASSES, activation=None)
  return ylogits, NCLASSES
```

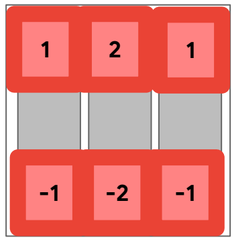
Using only a DNN for image classification could have billions of weights



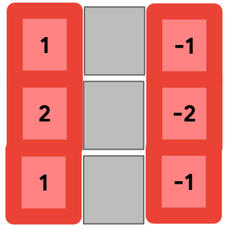
(Note: Many neurons not shown)

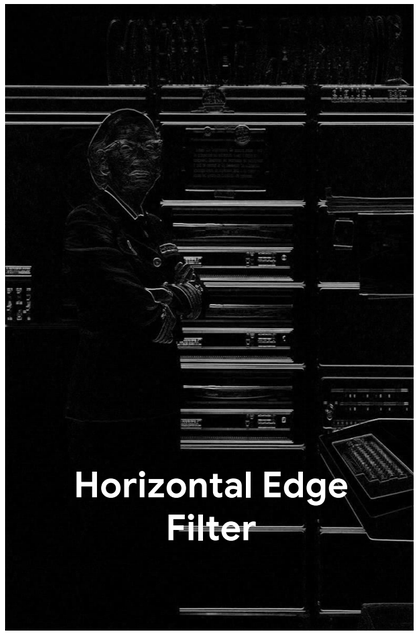
Kernels detect patterns and output filters

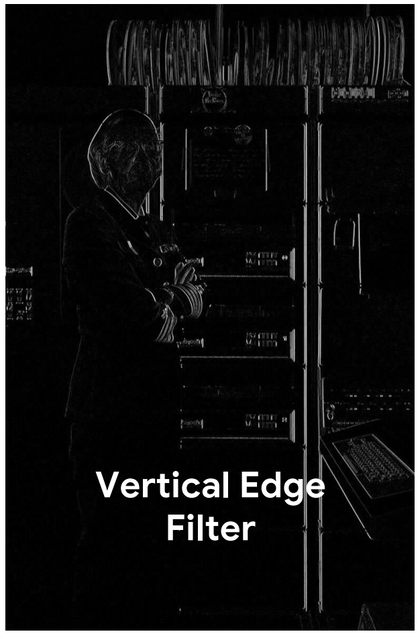
Kernel 1



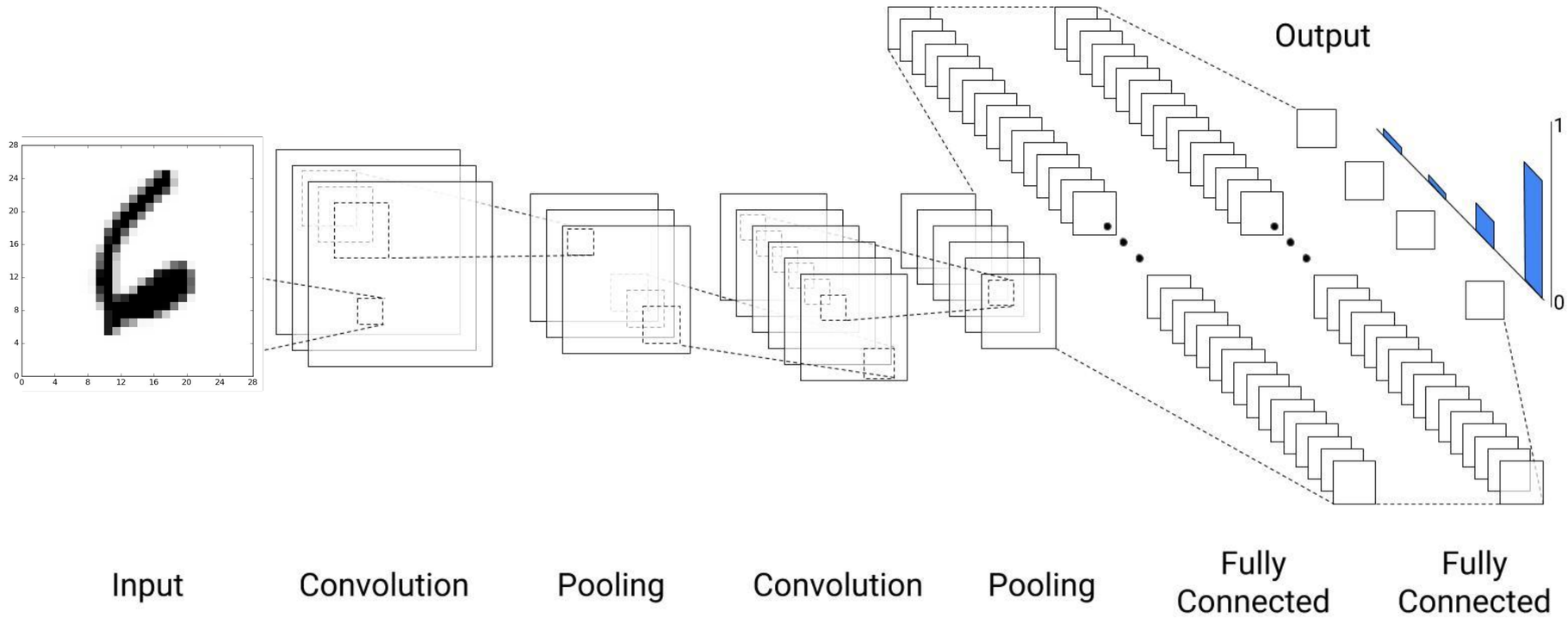
Kernel 2







Convolutional neural networks for image classification



Quiz:

Choose the advantage(s) of processing a small patch of the image at a time using a convolution kernel instead of using a dense layer for the entire image

1. Fewer weights
2. Faster processing
3. Weights are shared across the input
4. All of the above

Quiz:

Choose the advantage(s) of processing a small patch of the image at a time using a convolution kernel instead of using a dense layer for the entire image

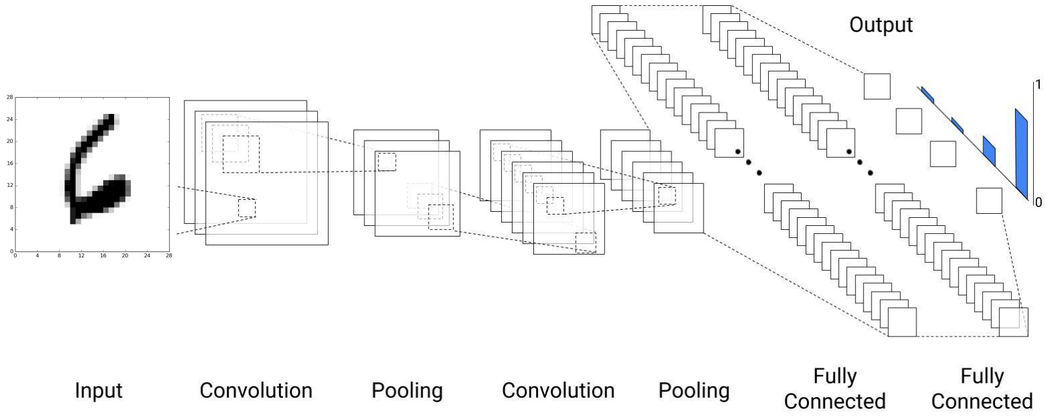
1. Fewer weights
2. Faster processing
3. Weights are shared across the input
4. **All of the above**

File Name: T-ICML-O_2_l6_lab_intro_cnn

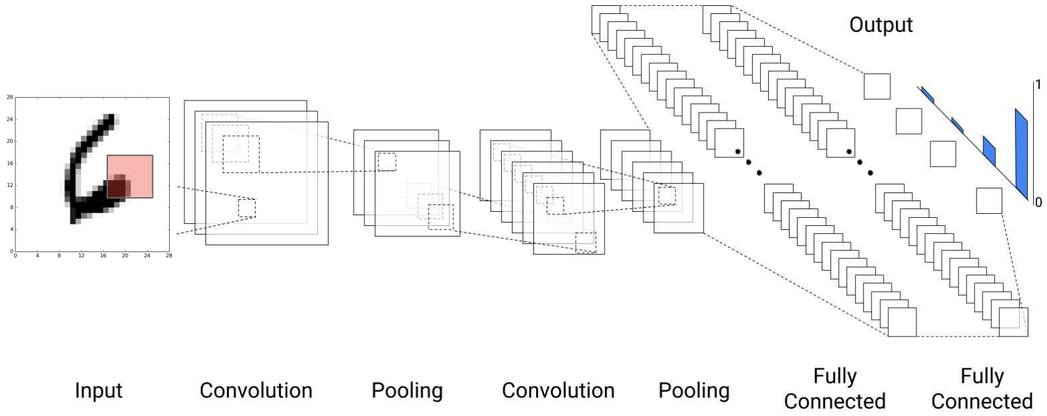
Format: Presenter in Studio

Presenter: Carl Osipov

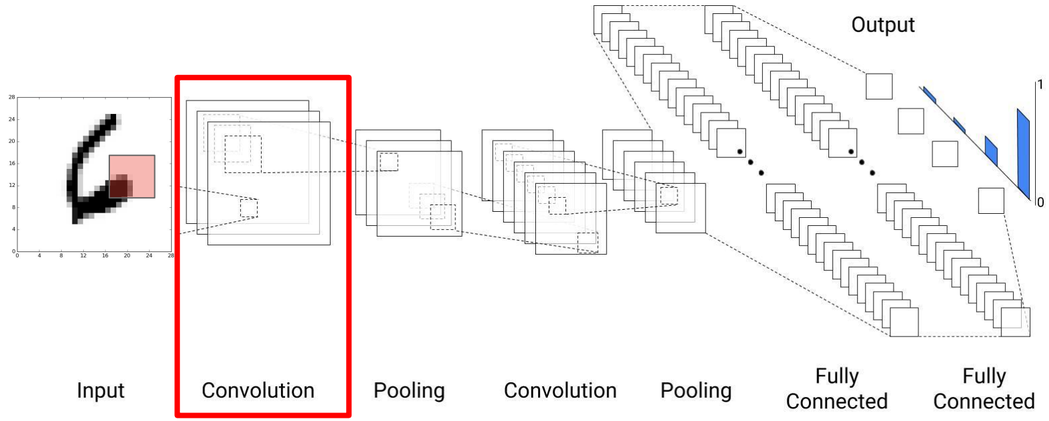
CNN Architecture Review



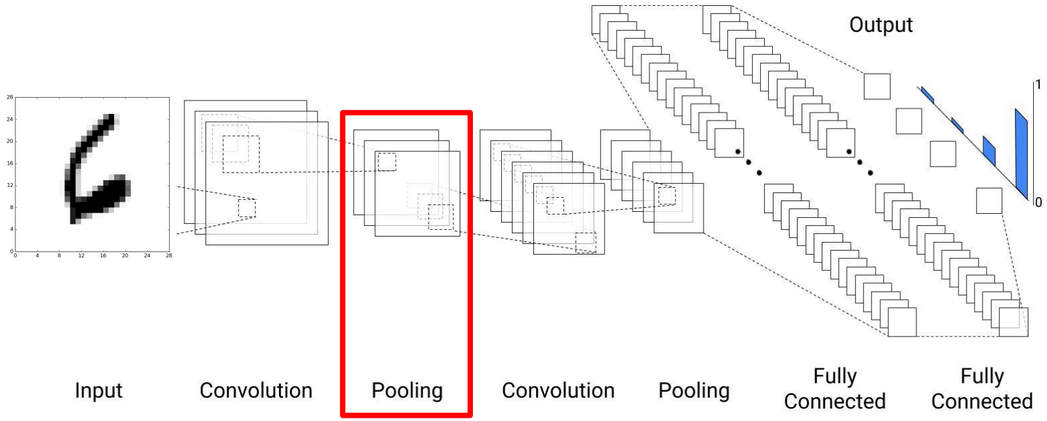
CNN Architecture Review



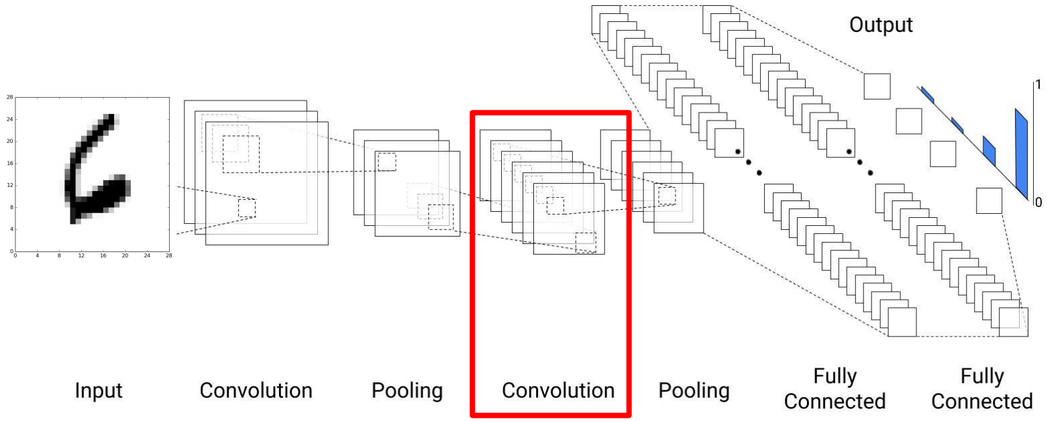
CNN Architecture Review



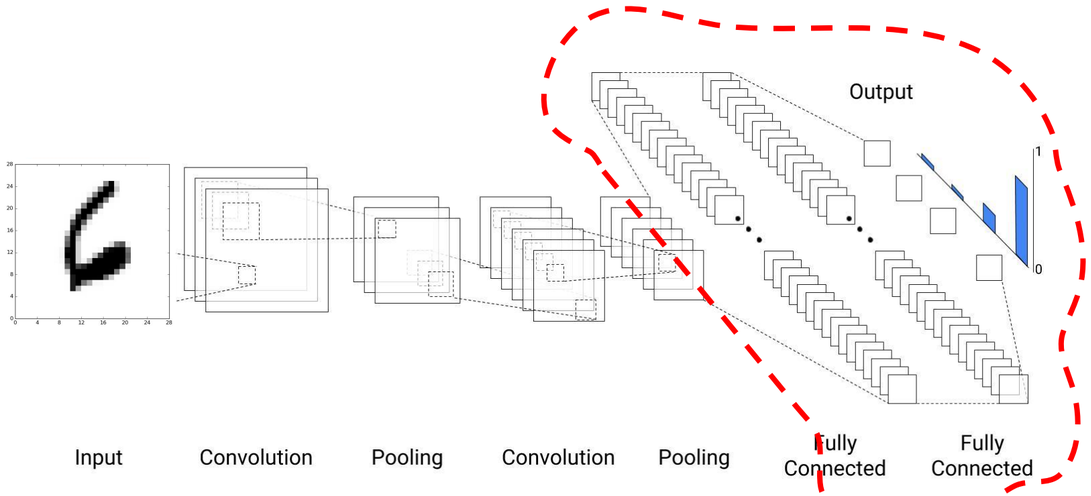
CNN Architecture Review



CNN Architecture Review



CNN Architecture Review



Fully-Connected DNN for classification

Lab

MNIST Image Classification with TensorFlow on CMLE (CNN)

Carl Osipov

Lab Steps

- Import the training dataset of MNIST handwritten images
- Reshape and preprocess the image data
- Setup your CNN with 10 classes
- Create convolutional and pooling layers + softmax function
- Define and create your EstimatorSpec in tensorflow to create your custom estimator
- Define and run your train_and_evaluate

File Name: T-ICML-O_2_l7_lab_solution_cnn

Format: Presenter in Studio

Presenter: Carl Osipov