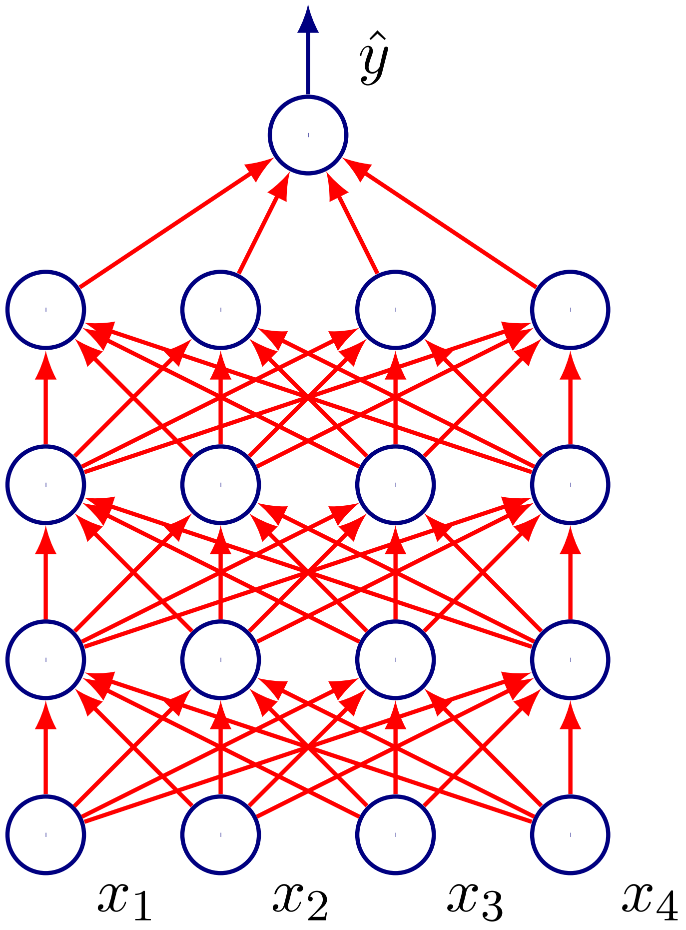
UNIVERSITÀ DELLA CALABRIA

ConvNets

Fully Connected Networks



$$a_i = \sum_{j \prec i} w_{i,j} z_j$$

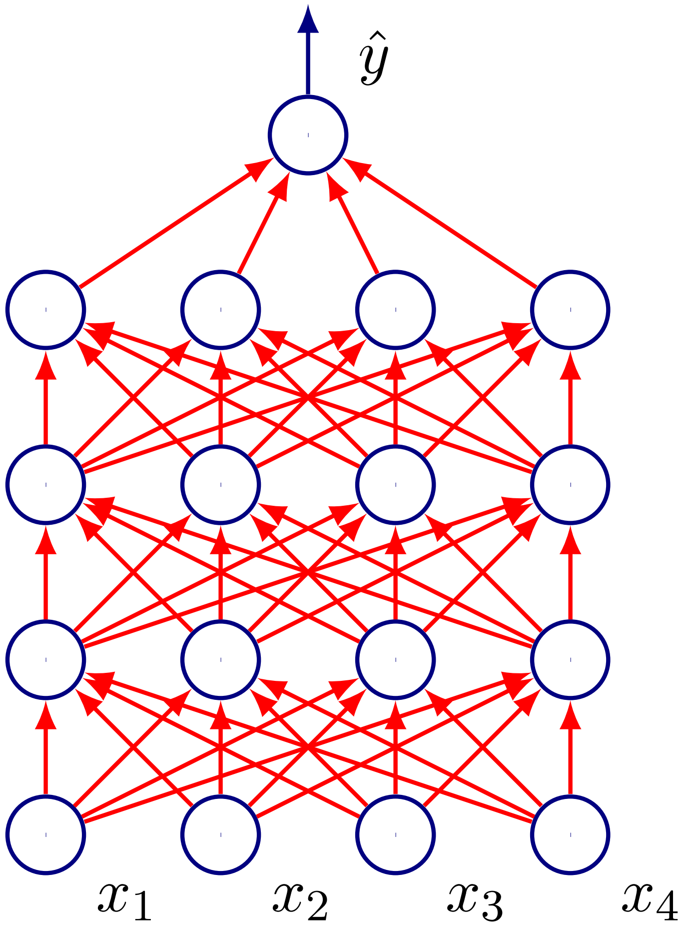
$$z_i = f(a_i)$$

$$\mathbf{a}^{(h+1)} = \mathbf{W}^{(h)} \mathbf{z}^{(h)}$$

$$\mathbf{z}^{(h+1)} = f\left(\mathbf{a}^{(h+1)}\right)$$

$$\mathbf{z}^{(0)} = \mathbf{x}$$

Fully Connected Networks



$$a_i = \sum_{j \prec i} w_{i,j} z_j$$

$$z_i = f(a_i)$$

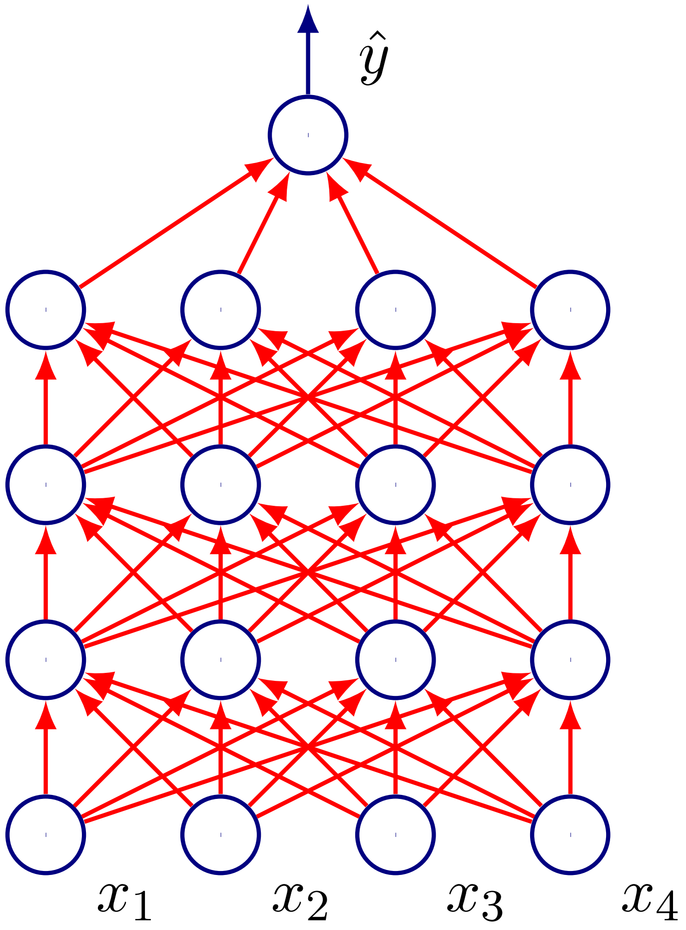
$$\mathbf{a}^{(h+1)} = \mathbf{W}^{(h)} \mathbf{z}^{(h)}$$

$$\mathbf{z}^{(h+1)} = f\left(\mathbf{a}^{(h+1)}\right)$$

$$\mathbf{z}^{(0)} = \mathbf{x}$$

- Each element is connected to the other, so we have
  - (5*4) + (5*4) + (5*4) + (5*4) + 5 connections

Fully Connected Networks



$$a_i = \sum_{j \prec i} w_{i,j} z_j$$

$$z_i = f(a_i)$$

$$\mathbf{a}^{(h+1)} = \mathbf{W}^{(h)} \mathbf{z}^{(h)}$$

$$\mathbf{z}^{(h+1)} = f\left(\mathbf{a}^{(h+1)}\right)$$

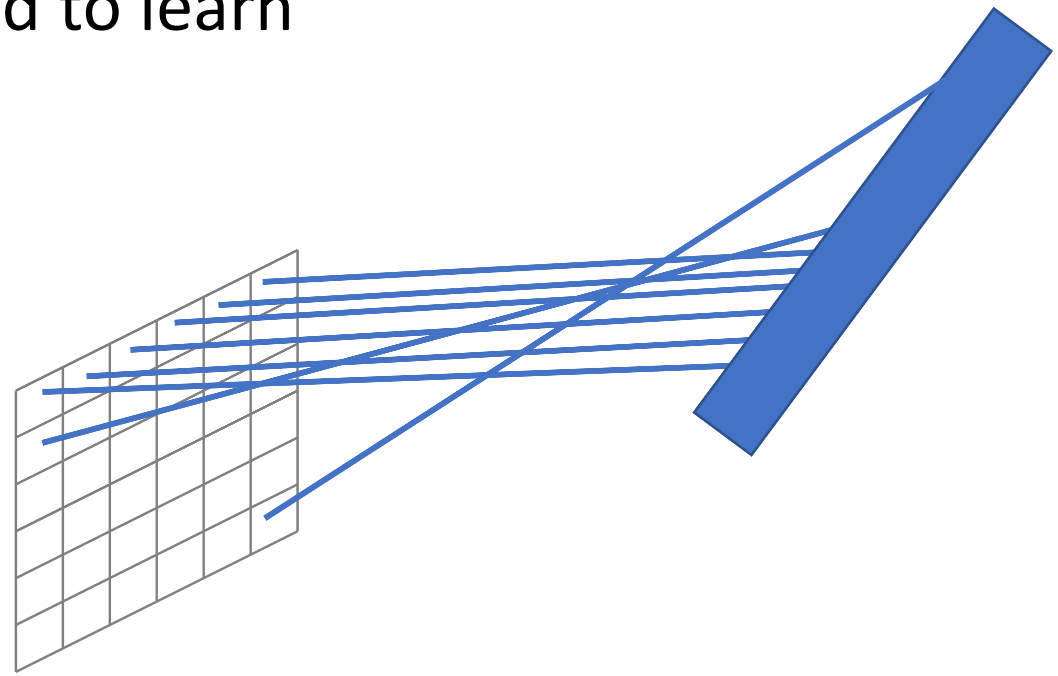
$$\mathbf{z}^{(0)} = \mathbf{x}$$

- Each element is connected to the other, so we have
  - (5*4) + (5*4) + (5*4) + (5*4) + 5 connections
- For a generic network with *k* layers and size d_h for each layer *h*, we have

$$\sum_{h=1}^{k} d_h \cdot d_{h-1}$$

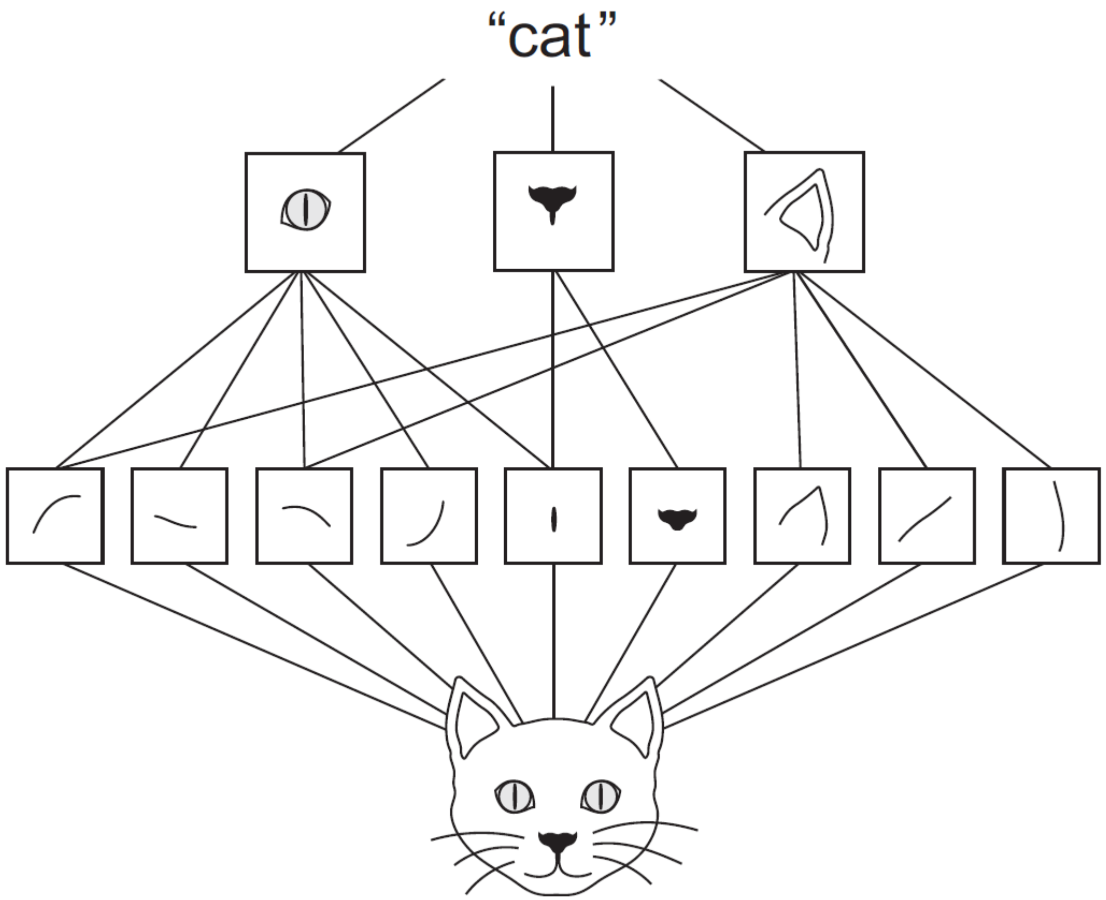
Further Issues with Images

- The «flat» approach is not suited to learn
  - Spatial patterns
  - Spatial hierarchies

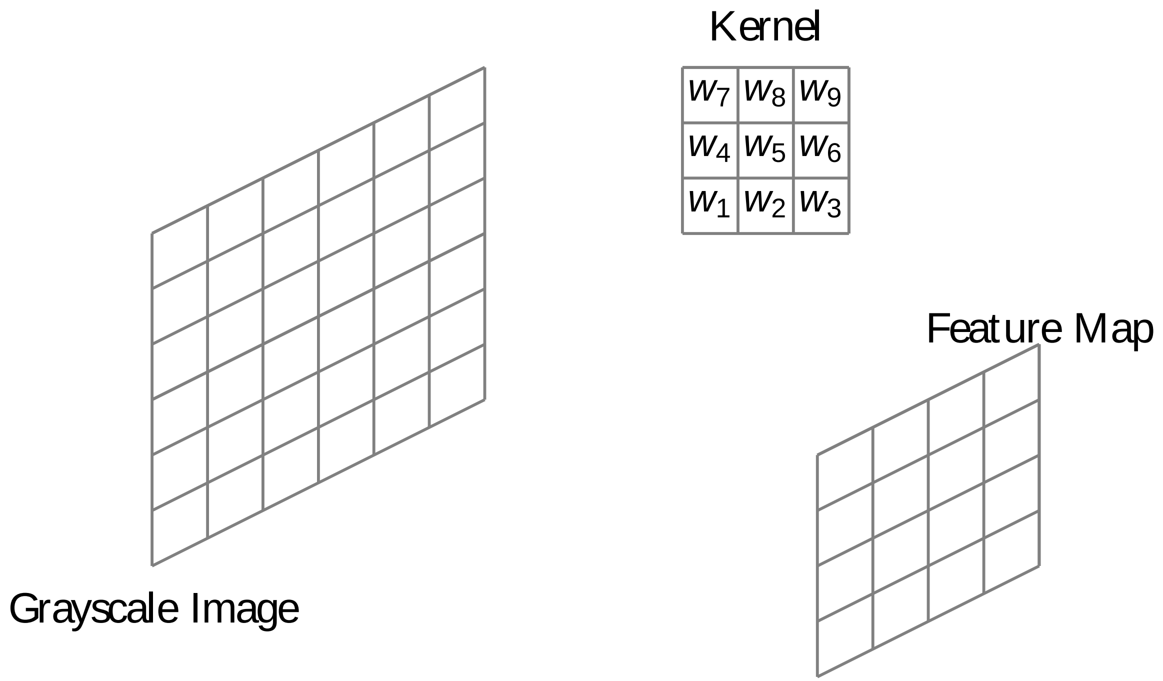


Example of patterns and hierarchies

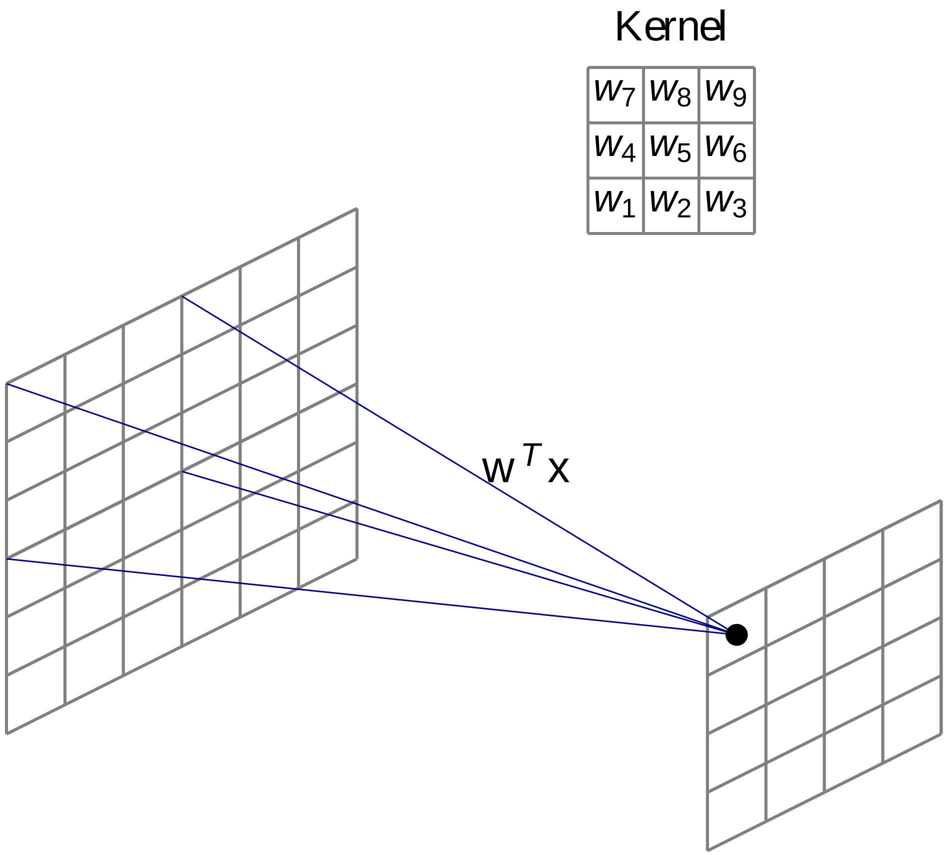
- How to identify such patterns?
- How to generalize them?



Convolution



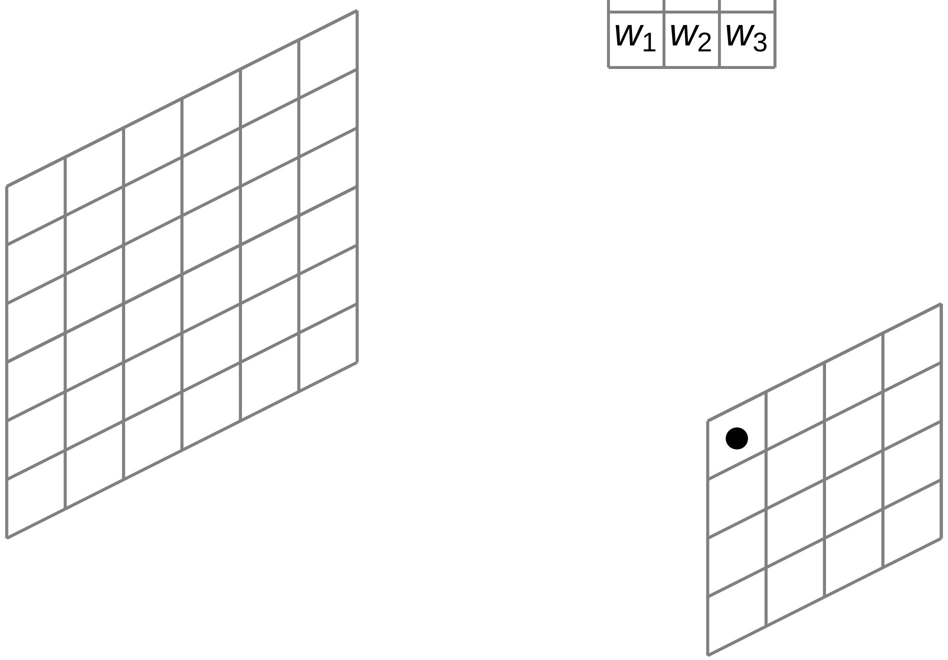
Convolution



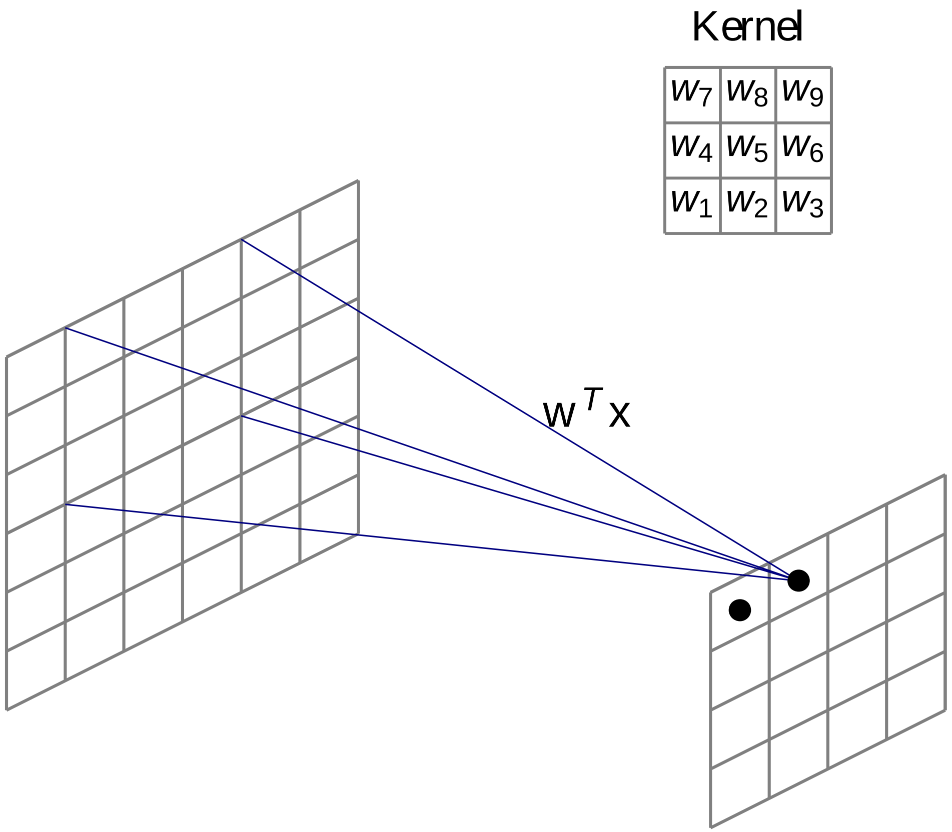
Convolution

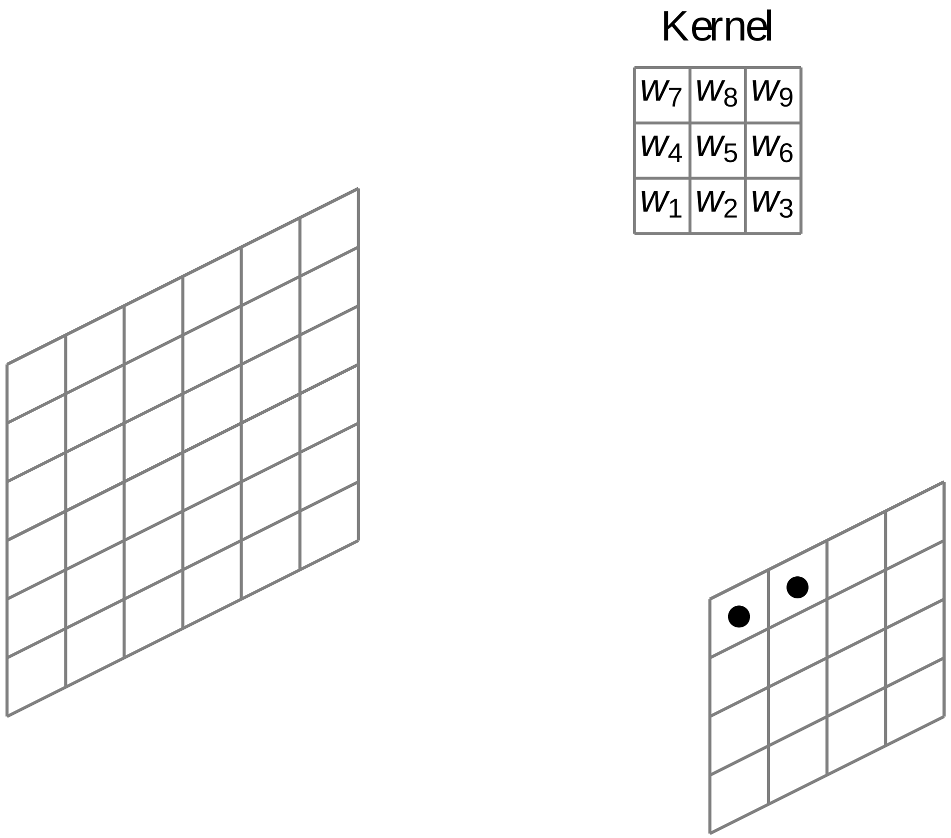
Kernel

w_7	w_8	w_9
w_4	w_5	w_6
w_1	w_2	w_3

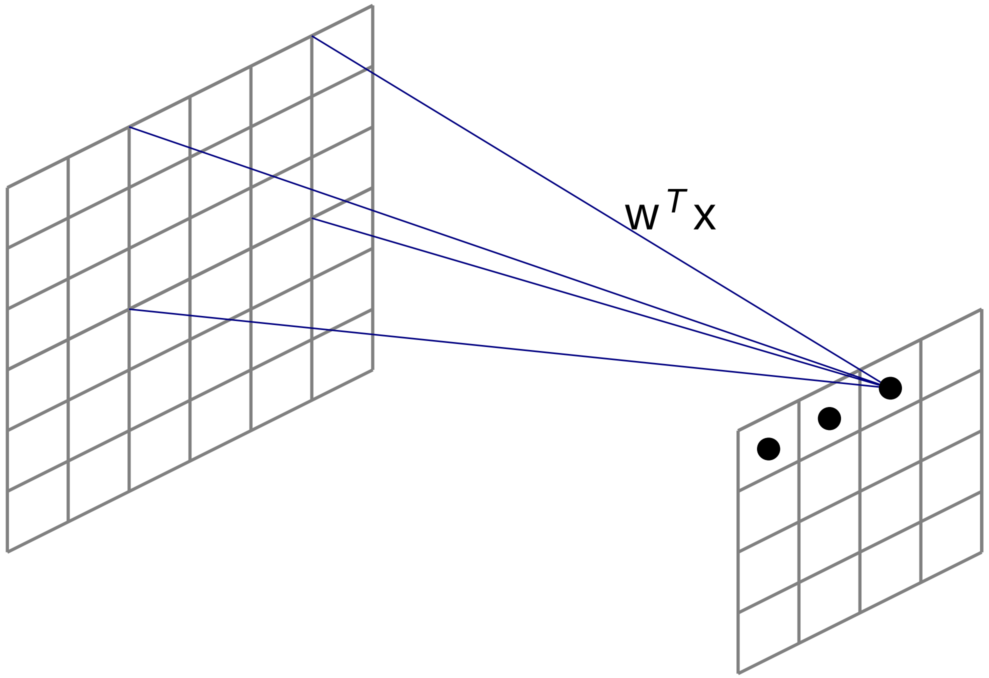


Convolution

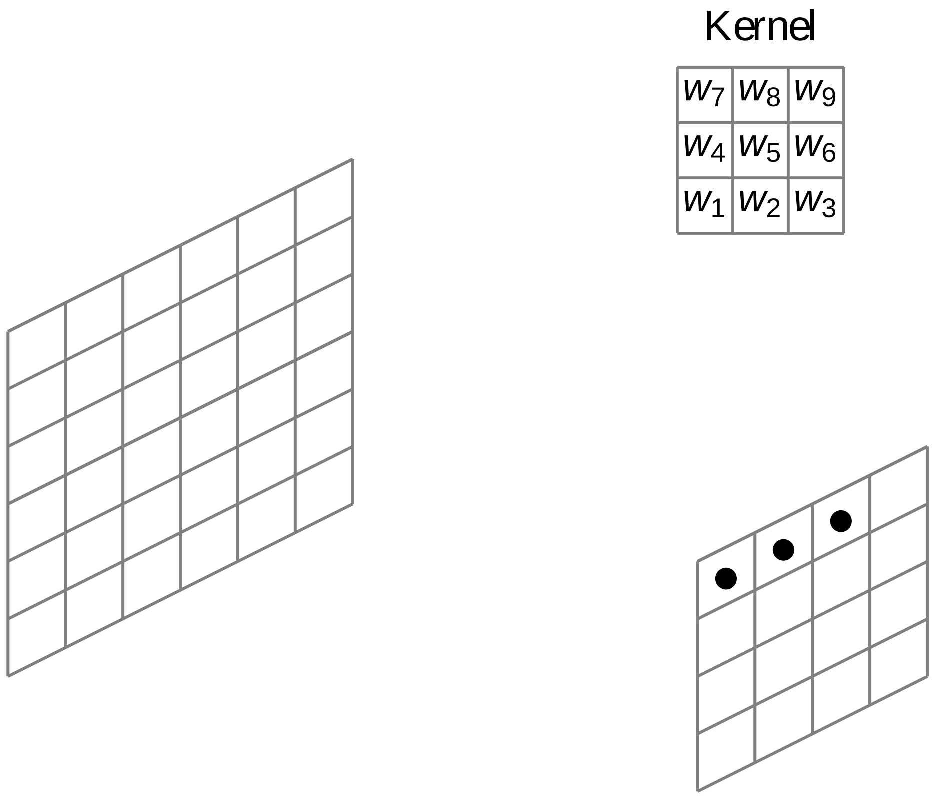



Kernel
w_7 w_8 w_9
w_4 w_5 w_6
w_1 w_2 w_3

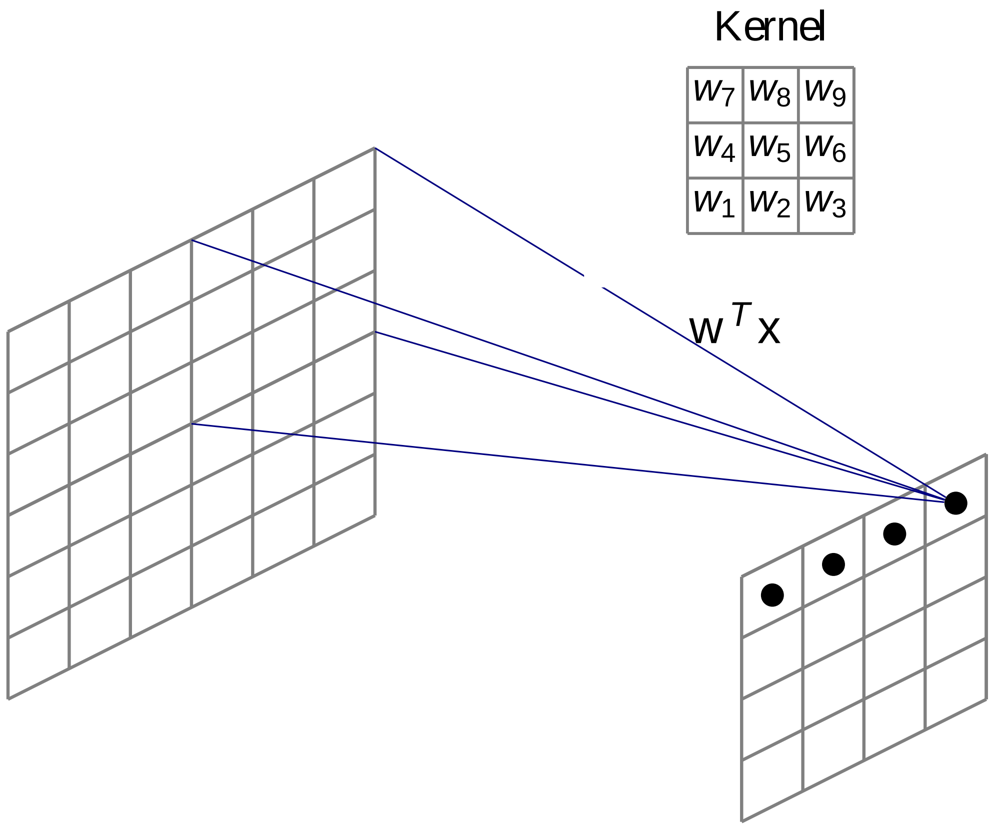
Convolution



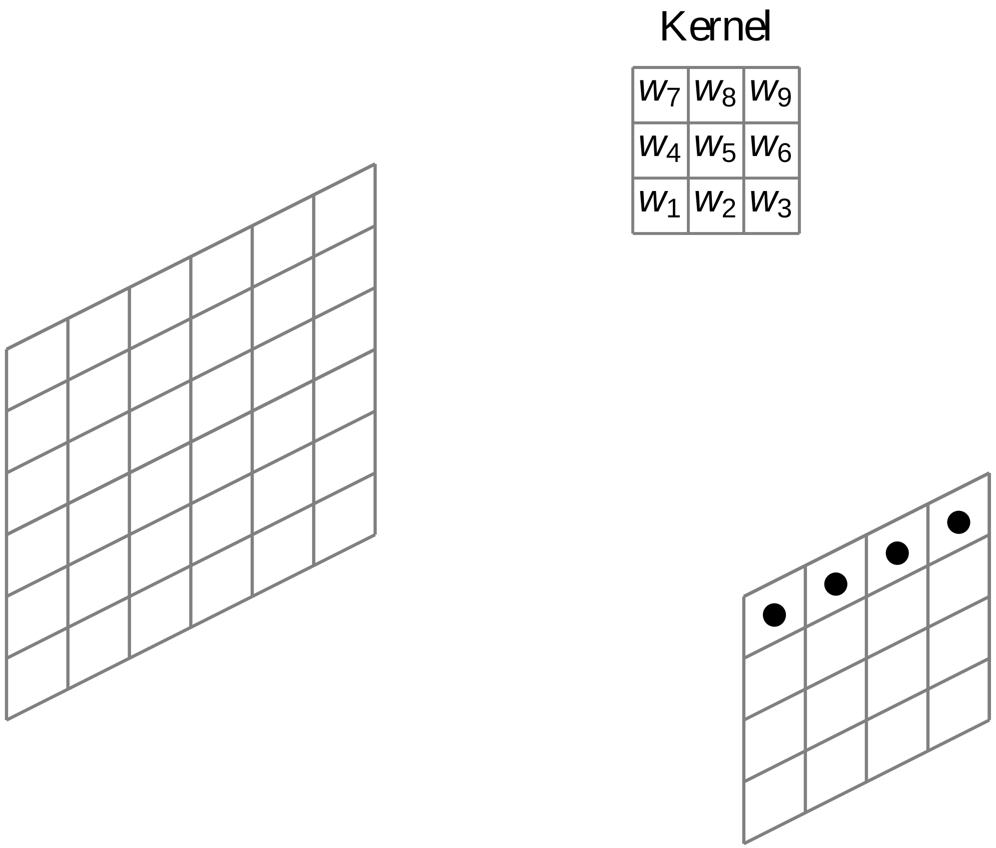
Convolution



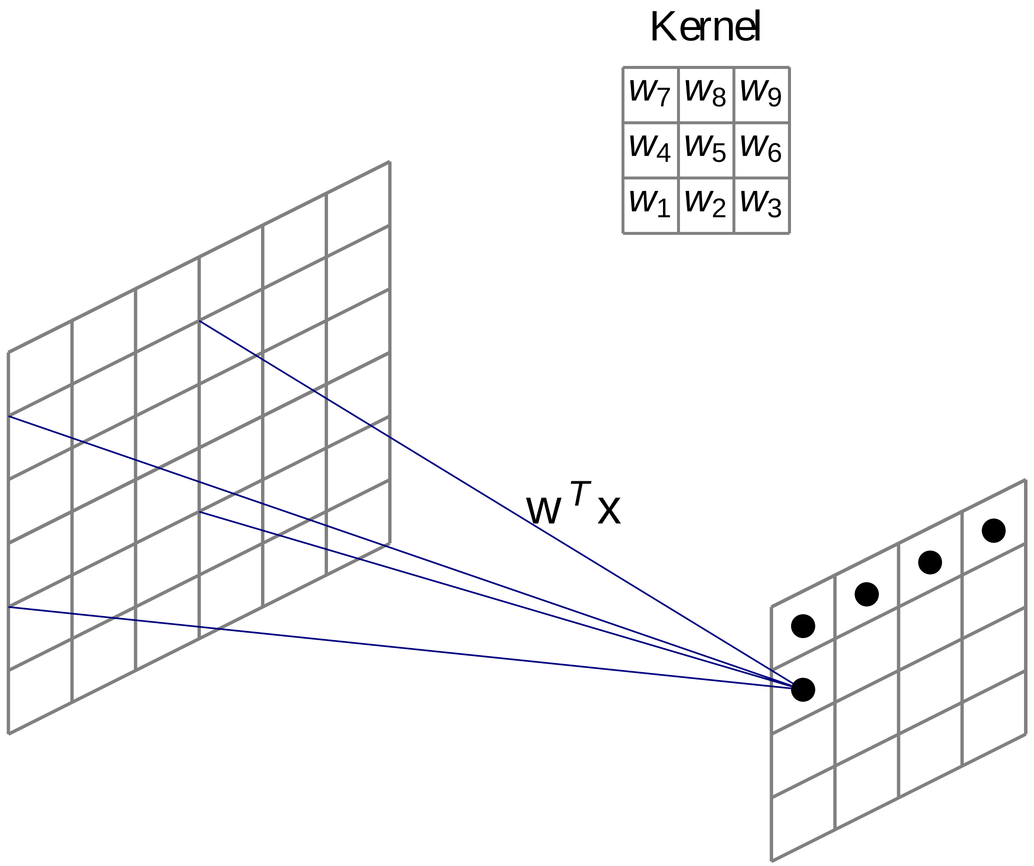
Convolution



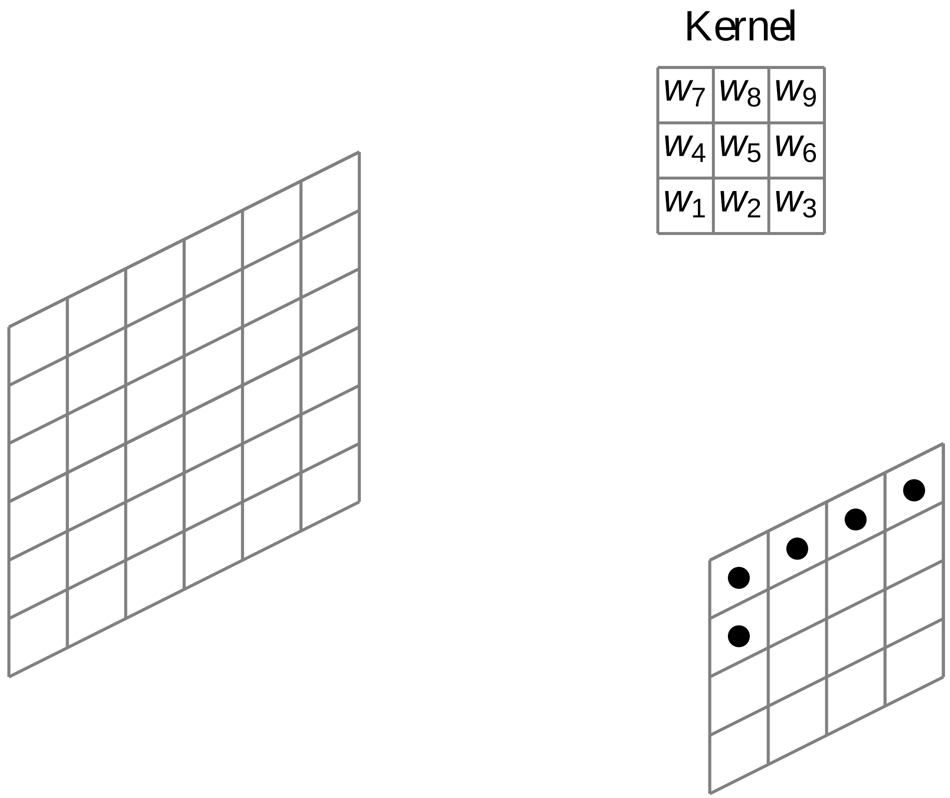
Convolution



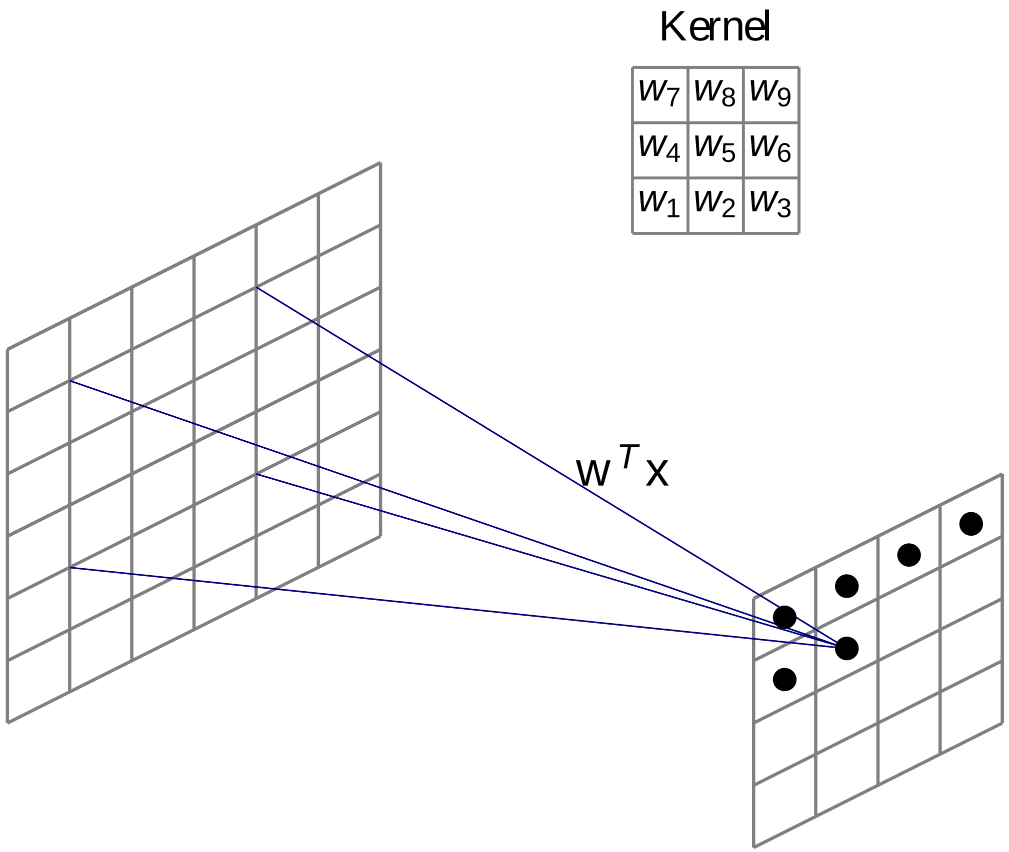
Convolution



Convolution



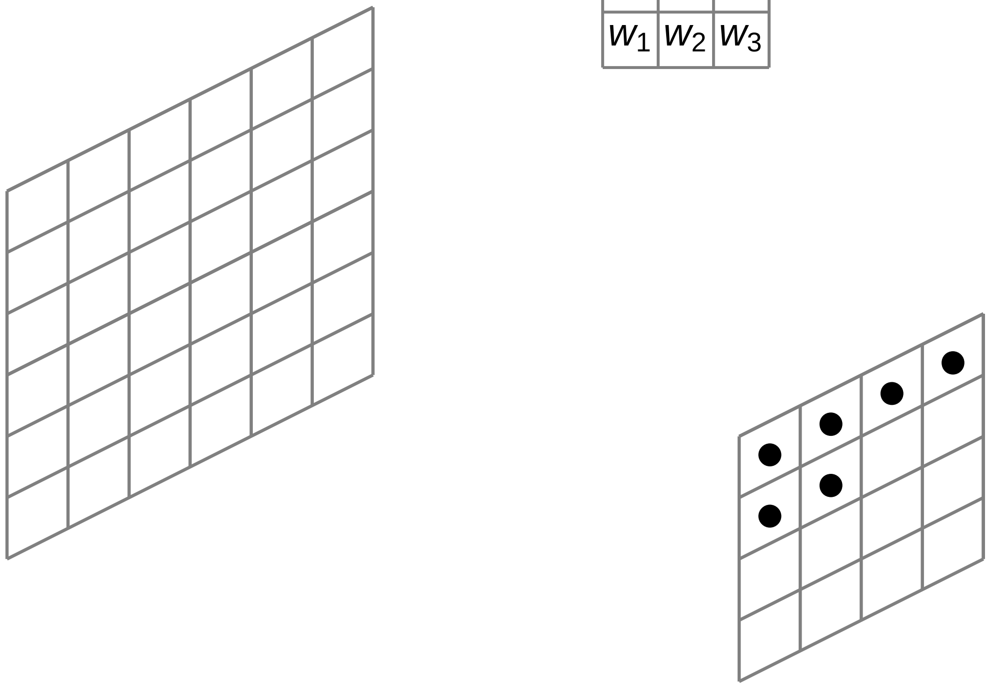
Convolution



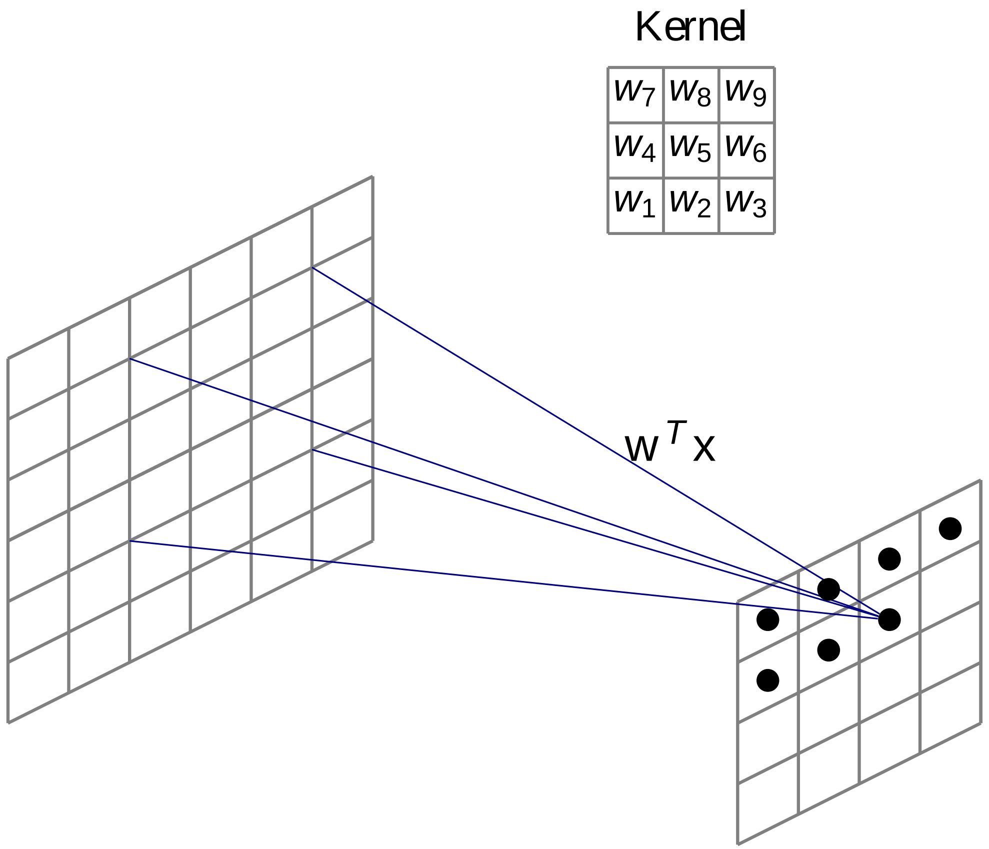
Convolution

Kernel

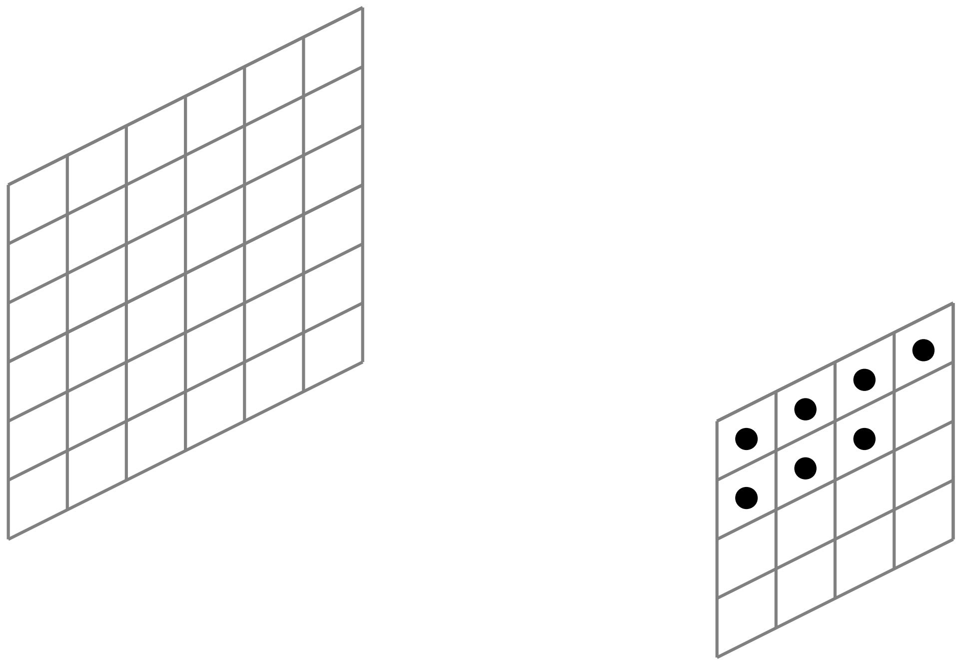
w_7	w_8	w_9
w_4	w_5	w_6
w_1	w_2	w_3



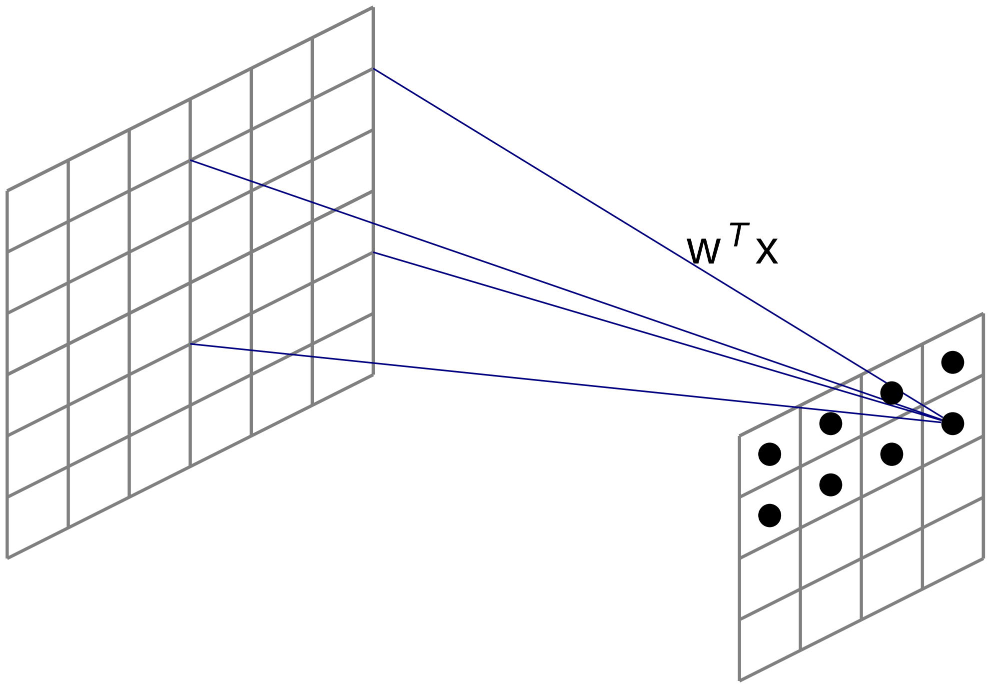
Convolution



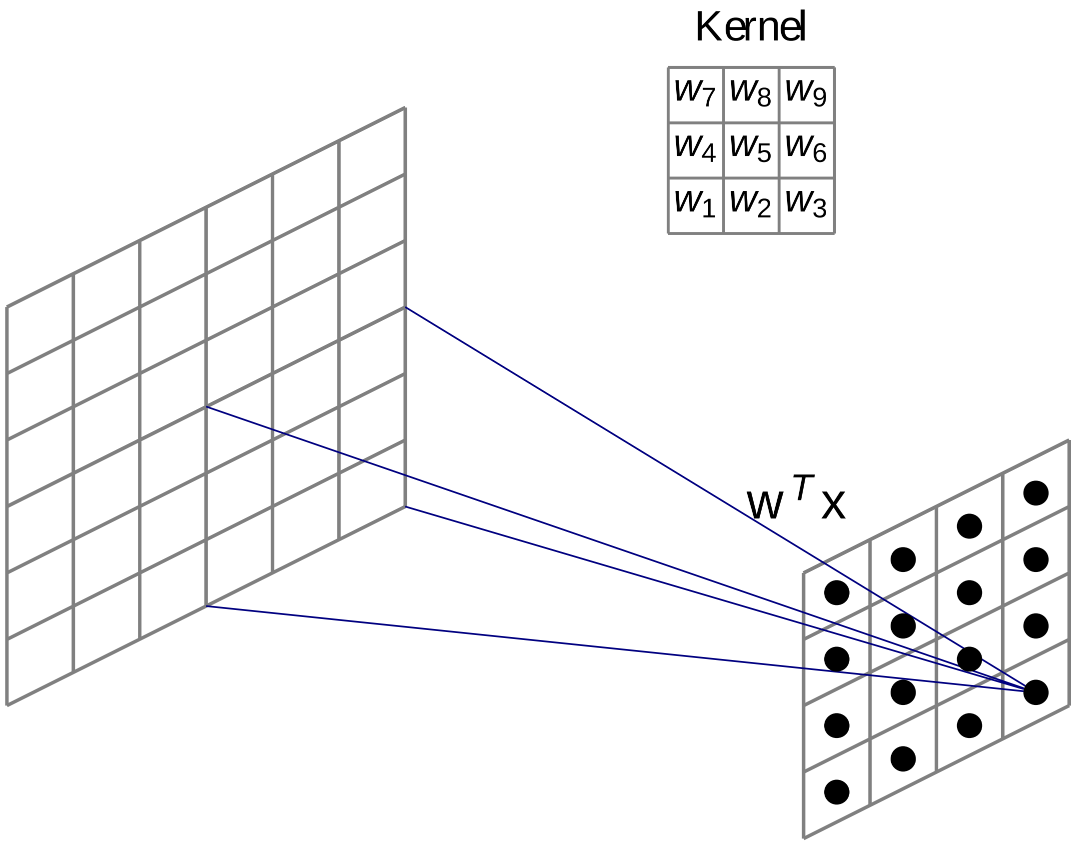
Convolution



Convolution

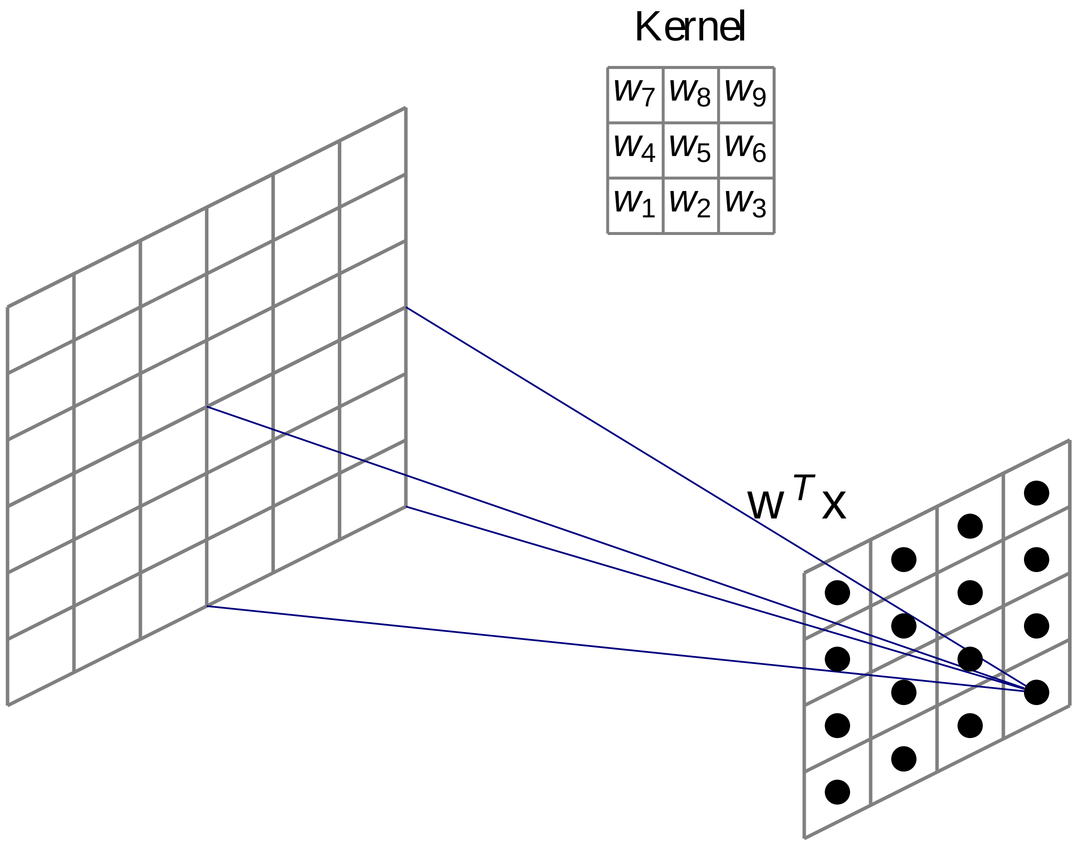


Convolution



Convolution

- What is the number of parameters?

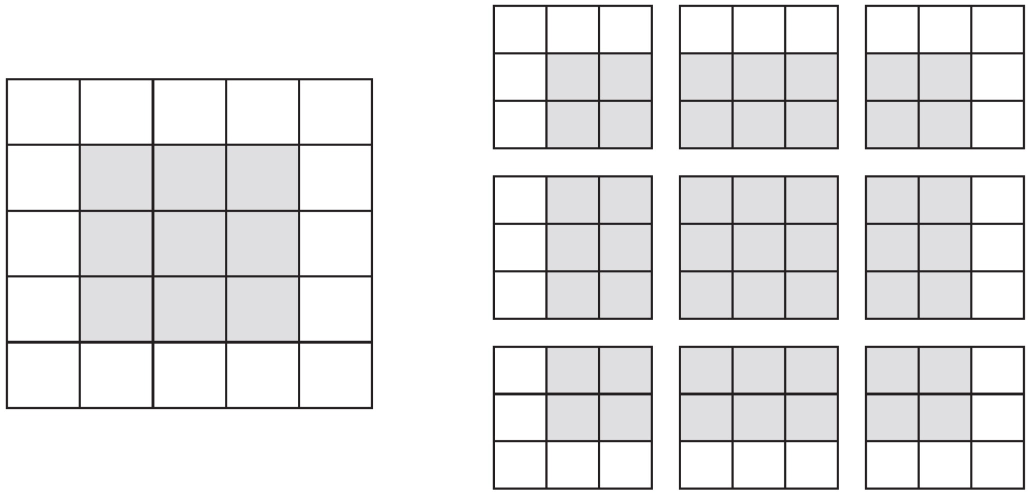


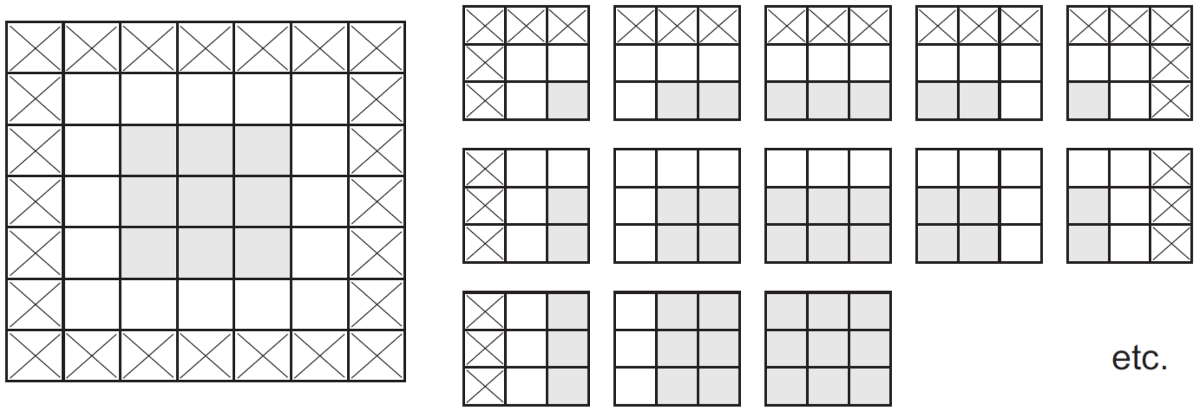
Why CNNs?

- Convolution leverages four ideas:
  - Sparse interactions
    - need to store fewer parameters, computing output needs fewer operations ($O(m \times n)$ versus $O(k \times n)$)
  - Parameter sharing
    - Same kernel is used throughout the input, so instead learning a parameter for each location, only a set of parameters is learnt
  - Equivariant representations
  - Ability to work with inputs of variable size

Padding

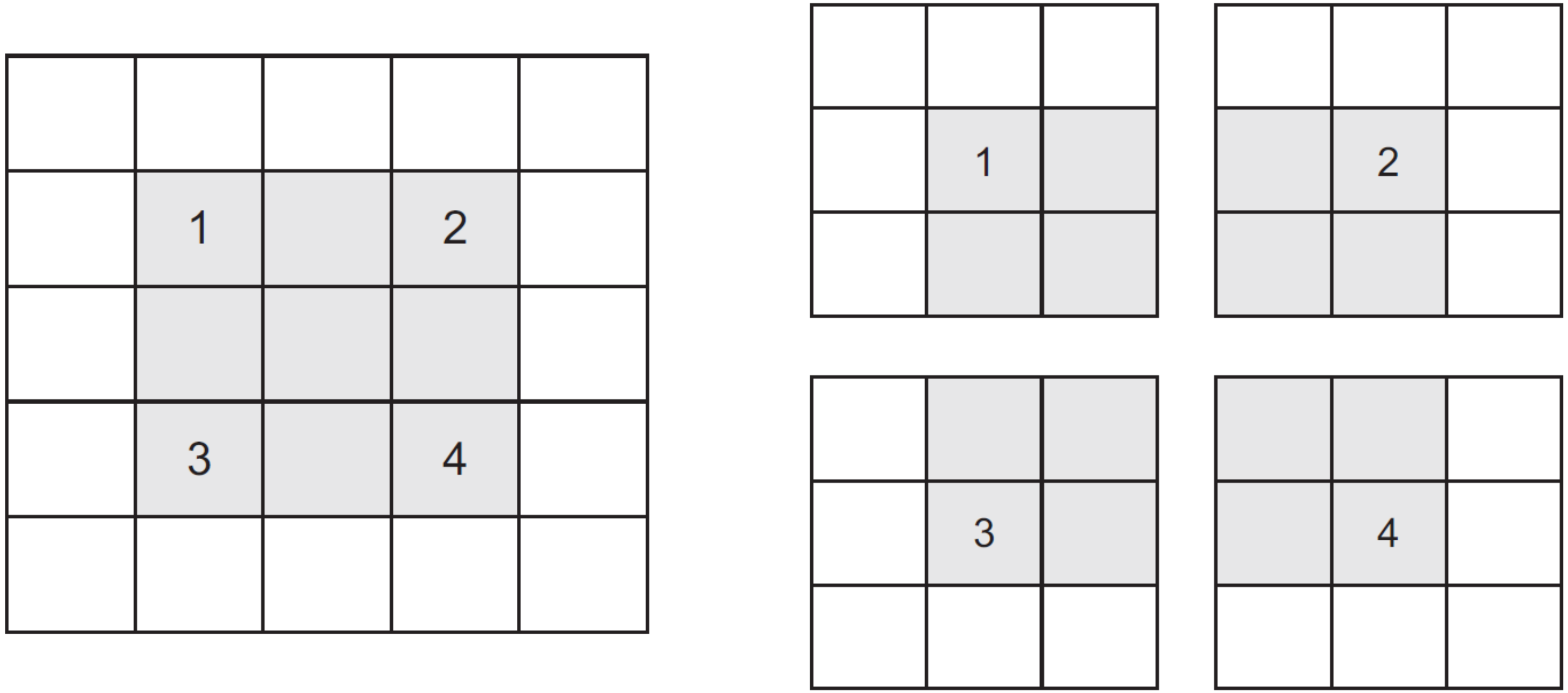
- The result of the convolution with a 3x3 feature maps shriks the images by 2 pixels along each dimension
- To avoid this effect, one might use padding, that is, add one external box of appropriate width and height.





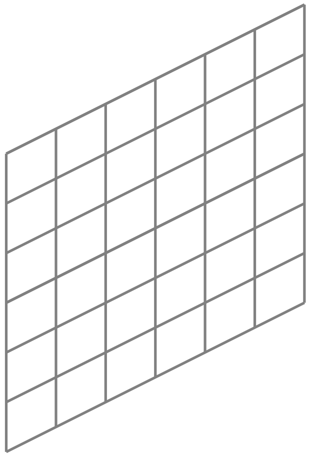
Strides

- The center of the convolution are not necessarily contiguous
- The distance between two consecutive windows is the stride



Example of 2x2 stride

Multiple Filters

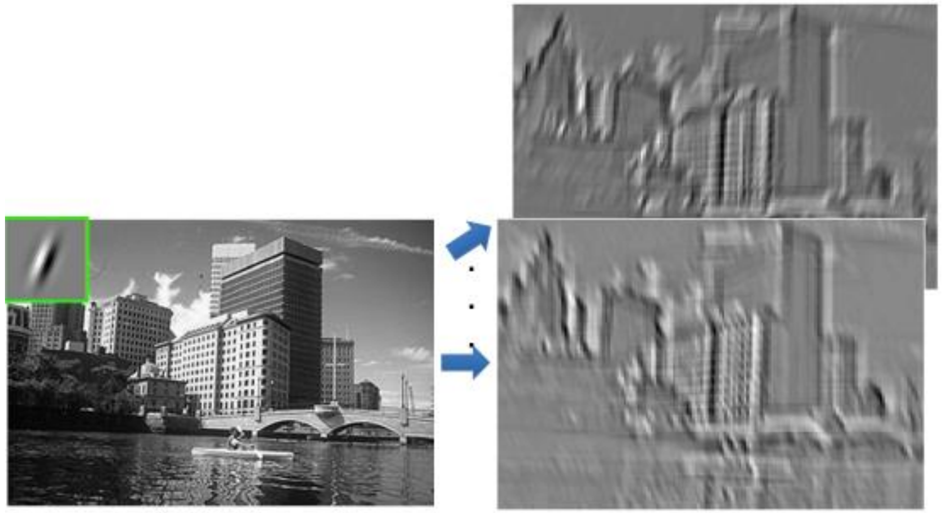


Kernel

7	8	9
4	5	6
1	2	3

Kernel

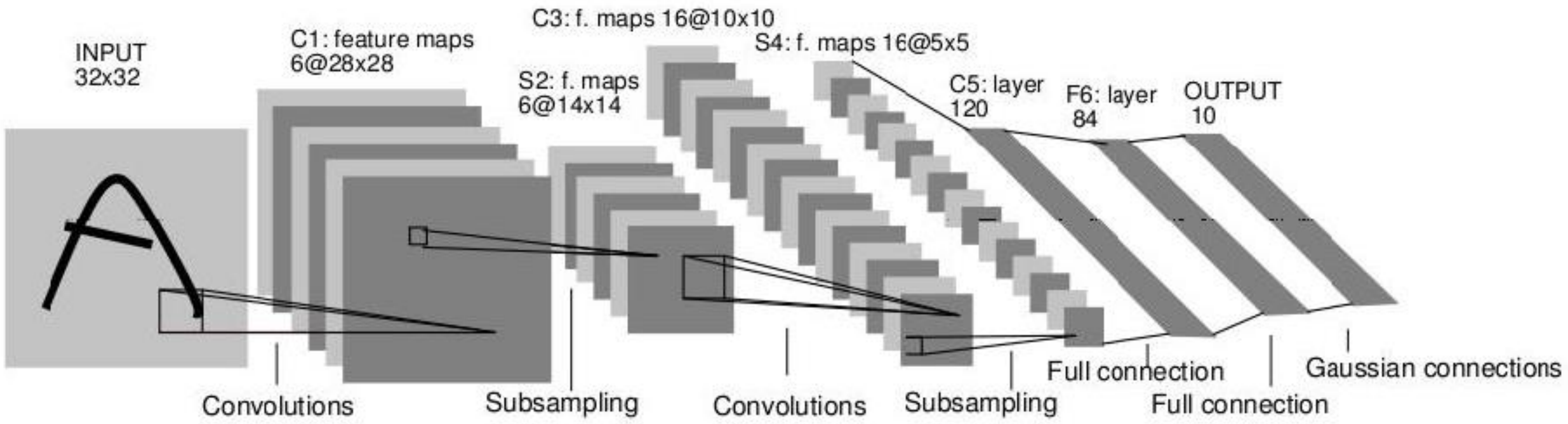
w_7	w_8	w_9
w_4	w_5	w_6
w_1	w_2	w_3



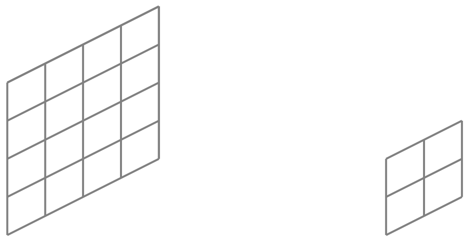
- We can use multiple kernels…
- … and each kernel identifies a feature map

Convolutional Networks

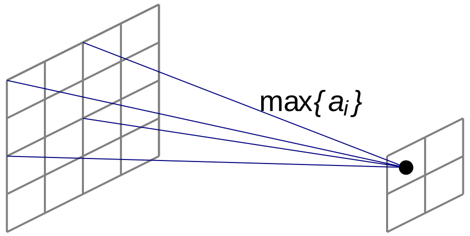
- Neural Networks that use convolution in place of general matrix multiplication in at least one layer



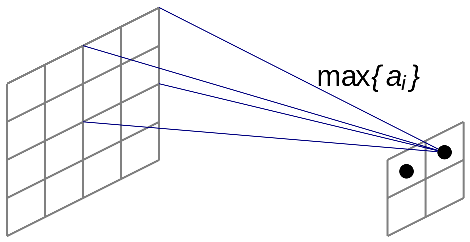
Pooling



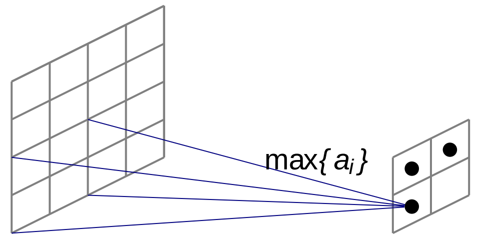
Pooling



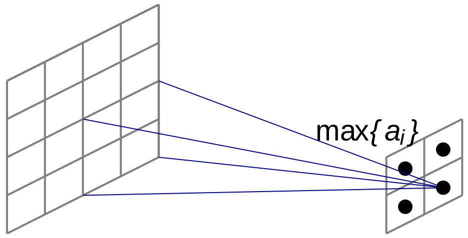
Pooling



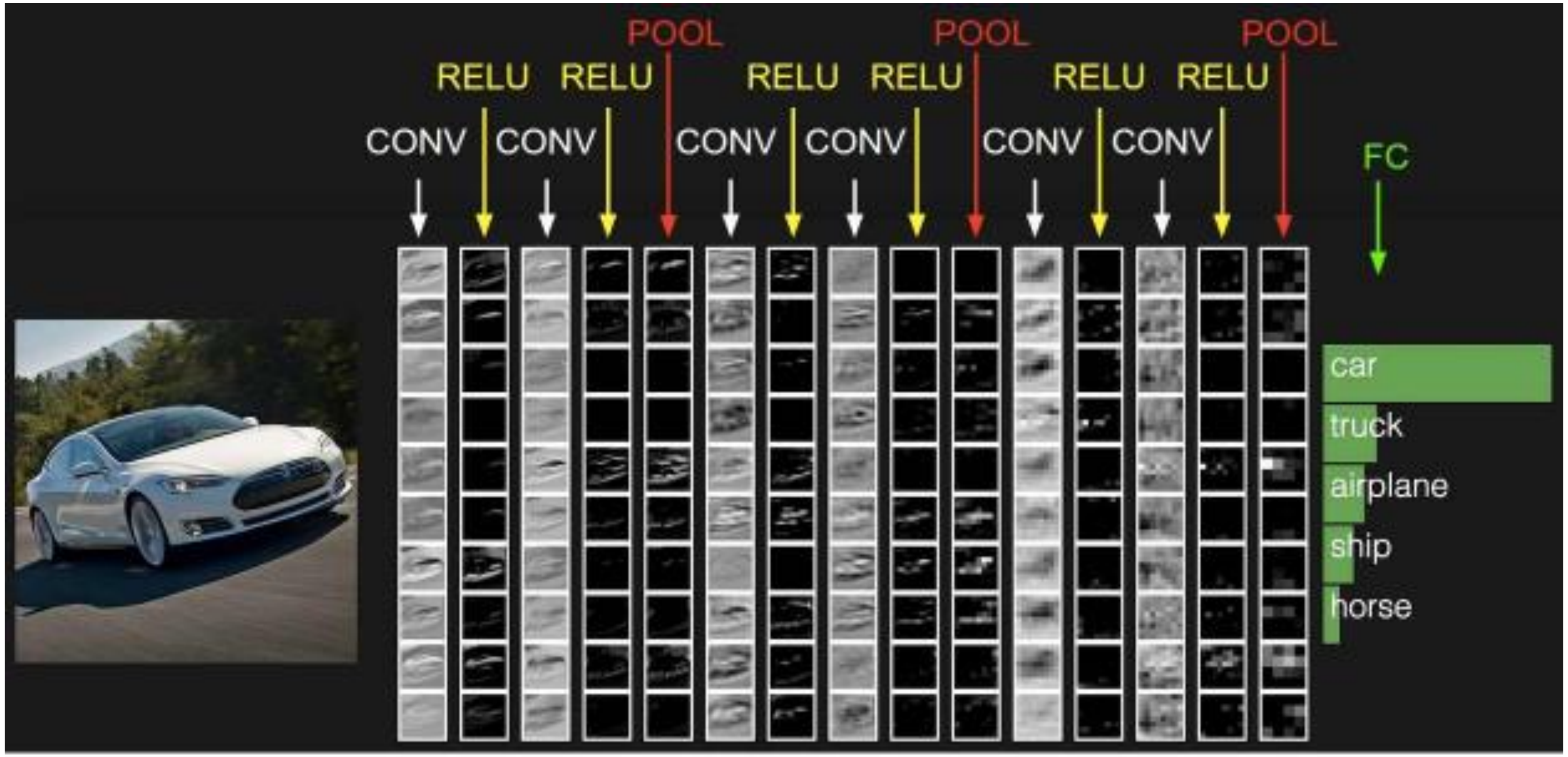
Pooling



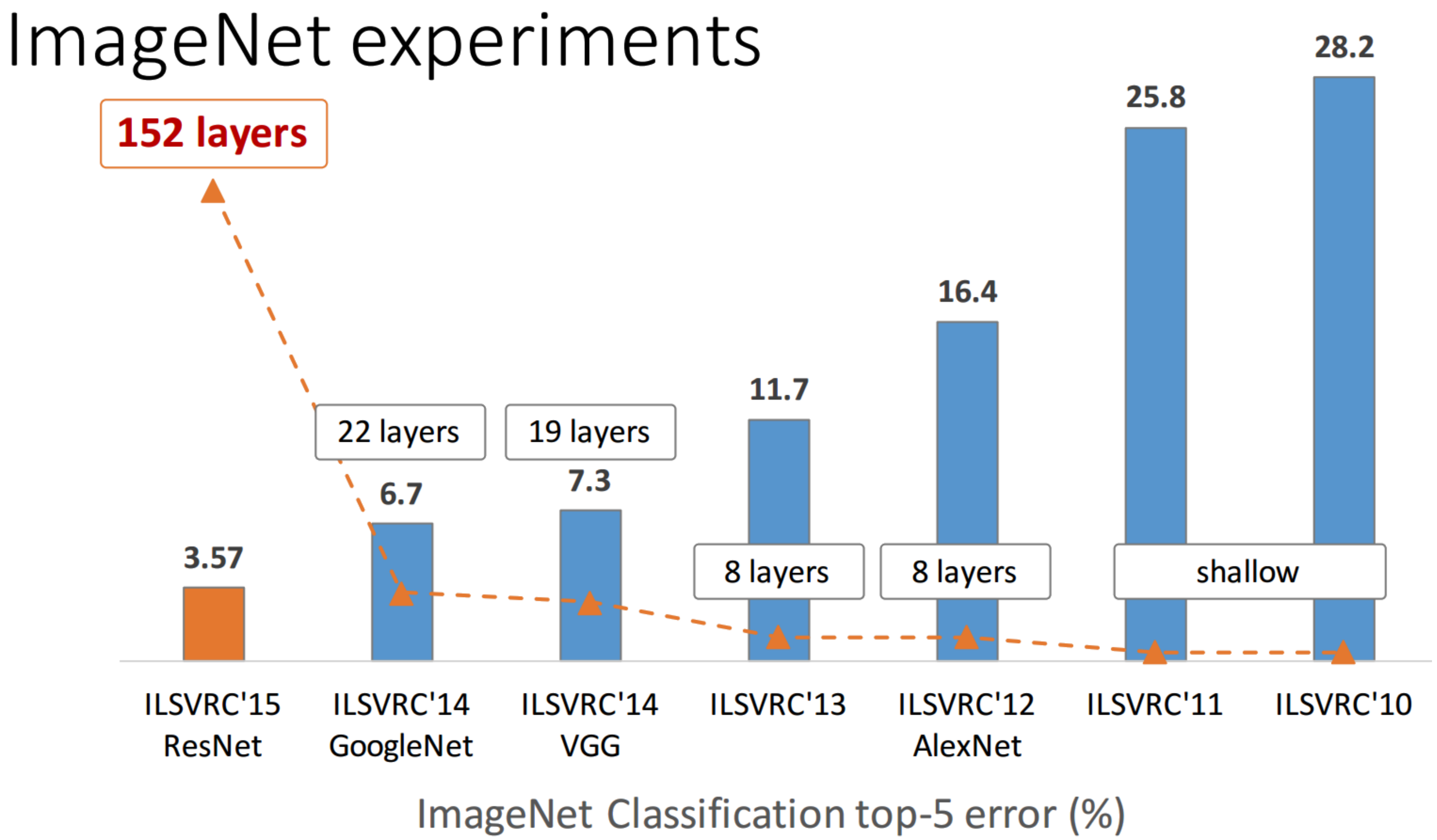
Pooling



Convolutional neural networks

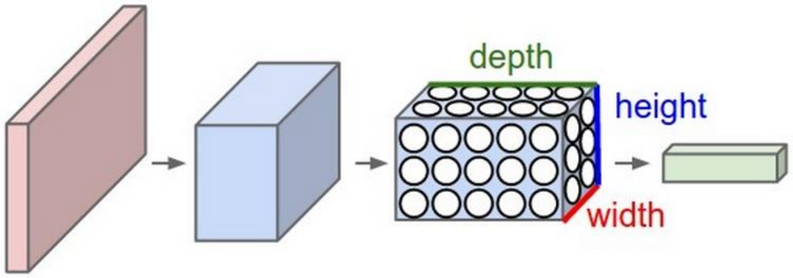


Deep Learning and Convolution



Convolution in general

- Not just 2-D image as a running example
  - Operates on volumes
    - E.g., RGB Images would be depth 3 input
  - Operates on 1-D vect



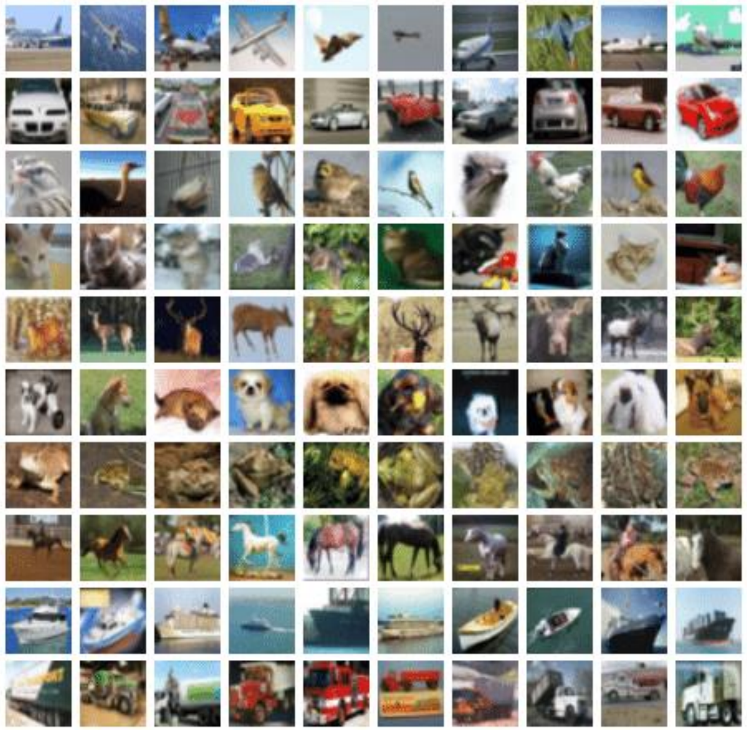
Example with RGB Images (1/4)

```
import tensorflow as tf
from tensorflow.keras import datasets, layers, models, optimizers

# CIFAR_10 is a set of 60K images 32x32 pixels on 3 channels
IMG_CHANNELS = 3
IMG_ROWS = 32
IMG_COLS = 32

#constant
BATCH_SIZE = 128
EPOCHS = 20
CLASSES = 10
VALIDATION_SPLIT = 0.2
OPTIM = tf.keras.optimizers.RMSprop()
```

Multiclass classification



Example with RGB Images (2/4)

Number of filters

Kernel size

No padding (otherwise, use 'same')

```
#define the convnet
def build(input_shape, classes):
  model = models.Sequential()
  model.add(layers.Convolution2D(32, (3, 3), activation='relu', padding='valid'
                        input_shape=input_shape))
  model.add(layers.MaxPooling2D(pool_size=(2, 2)))
  model.add(layers.Dropout(0.25))

  model.add(layers.Flatten())
  model.add(layers.Dense(512, activation='relu'))
  model.add(layers.Dropout(0.5))
  model.add(layers.Dense(classes, activation='softmax'))
  return model
```

Example with RGB Images (3/4)

```
# data: shuffled and split between train and test sets
(X_train, y_train), (X_test, y_test) = datasets.cifar10.load_data()
# normalize
X_train, X_test = X_train / 255.0, X_test / 255.0
# convert to categorical
# convert class vectors to binary class matrices
y_train = tf.keras.utils.to_categorical(y_train, CLASSES)
y_test = tf.keras.utils.to_categorical(y_test, CLASSES)

model=build((IMG_ROWS, IMG_COLS, IMG_CHANNELS), CLASSES)
model.summary()
```

Example with RGB Images (4/4)

```
# use TensorBoard,
callbacks = [
  # Write TensorBoard logs to `./logs` directory
  tf.keras.callbacks.TensorBoard(log_dir='./logs')
]

# train
model.compile(loss='categorical_crossentropy', optimizer=OPTIM, metrics=['accuracy'])

model.fit(X_train, y_train, batch_size=BATCH_SIZE,
  epochs=EPOCHS, validation_split=VALIDATION_SPLIT,
  verbose=VERBOSE, callbacks=callbacks)
score = model.evaluate(X_test, y_test,
                     batch_size=BATCH_SIZE, verbose=VERBOSE)
print("\nTest score:", score[0])
print('Test accuracy:', score[1])
```