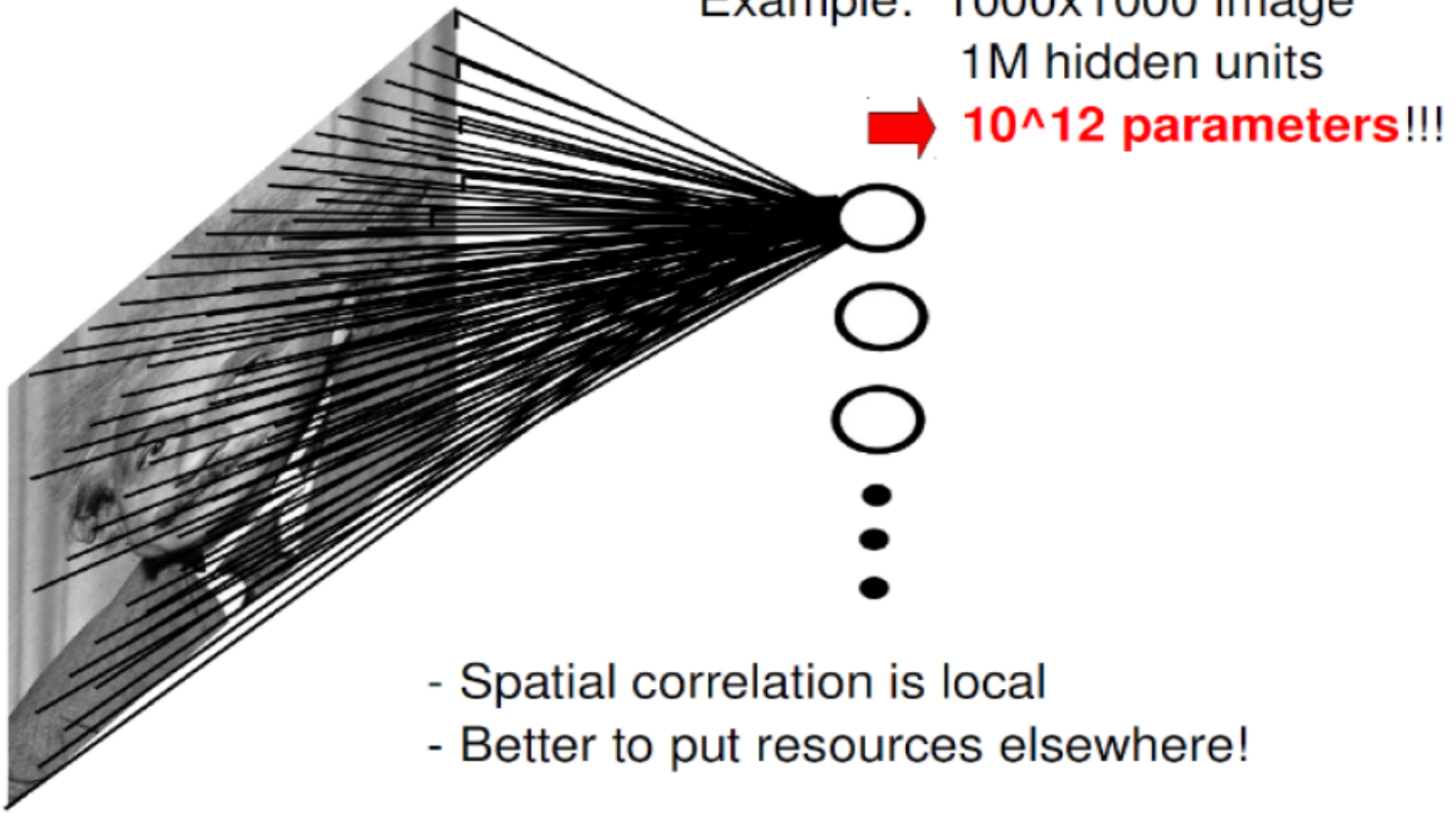
Convoluted Neural Network Internals

By Mohit Kumar

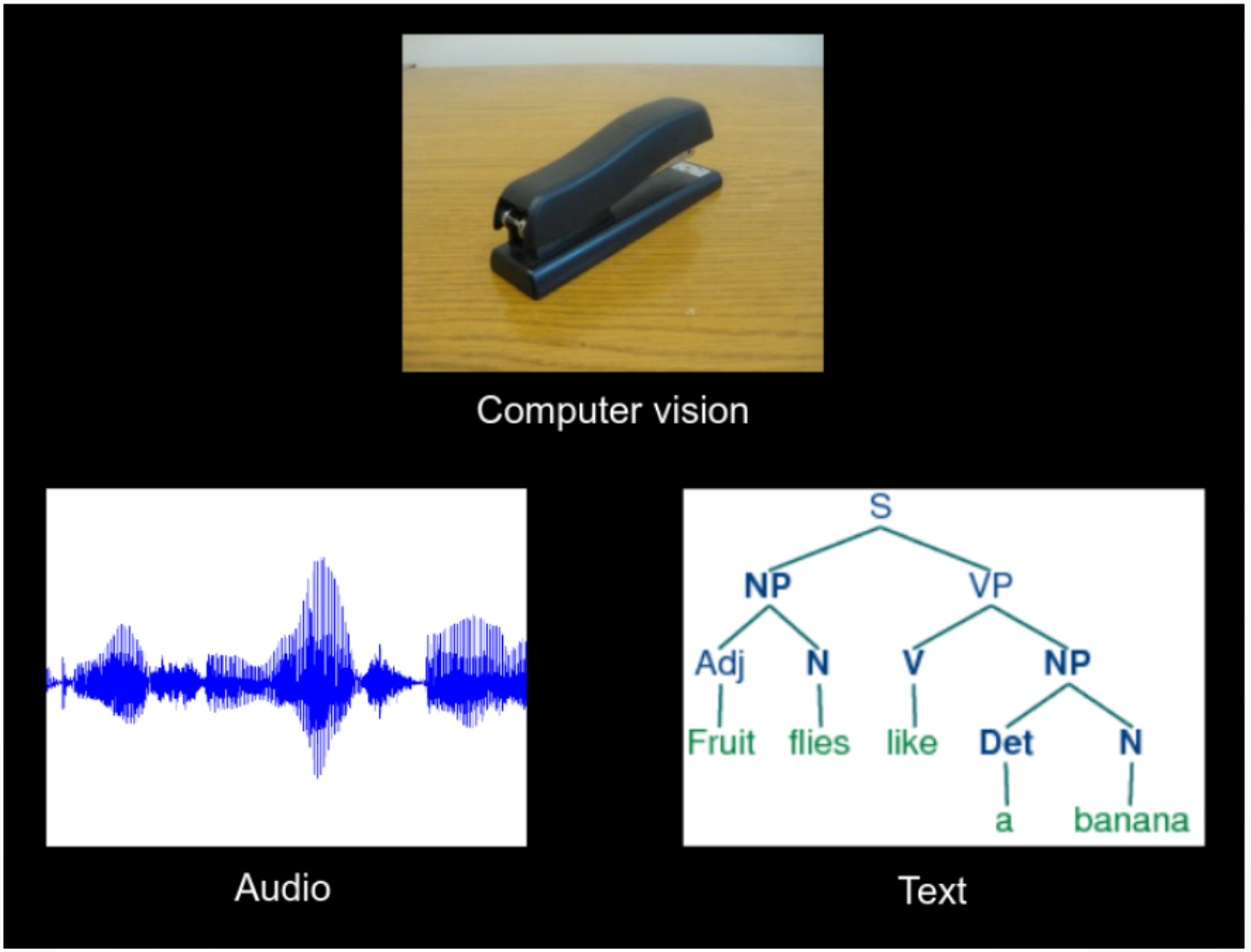
Convoluted Neural Network:Why?

Why Convolution?

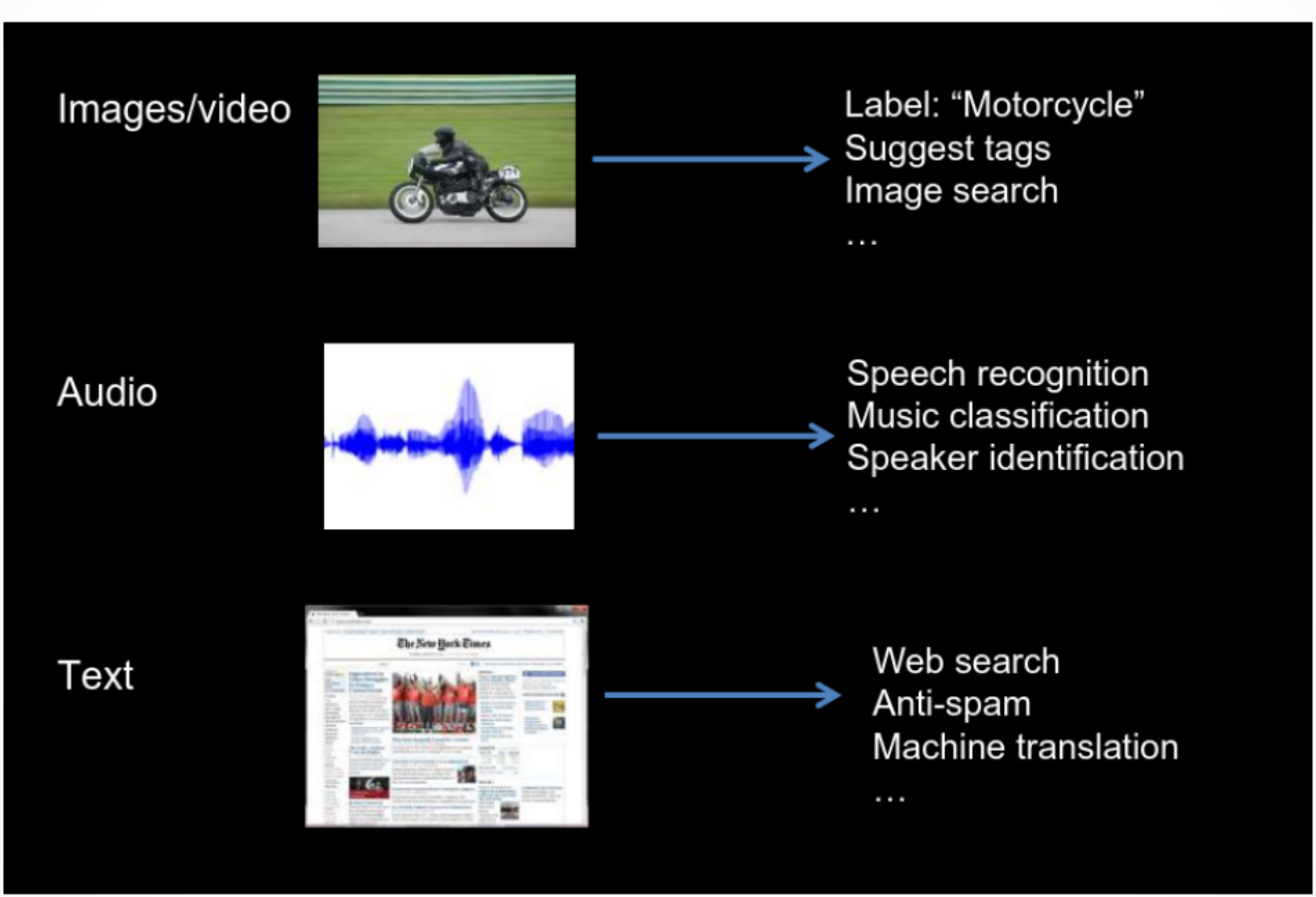
FULLY CONNECTED NEURAL NET



Convoluted Neural Network:Why?: Detection or classification

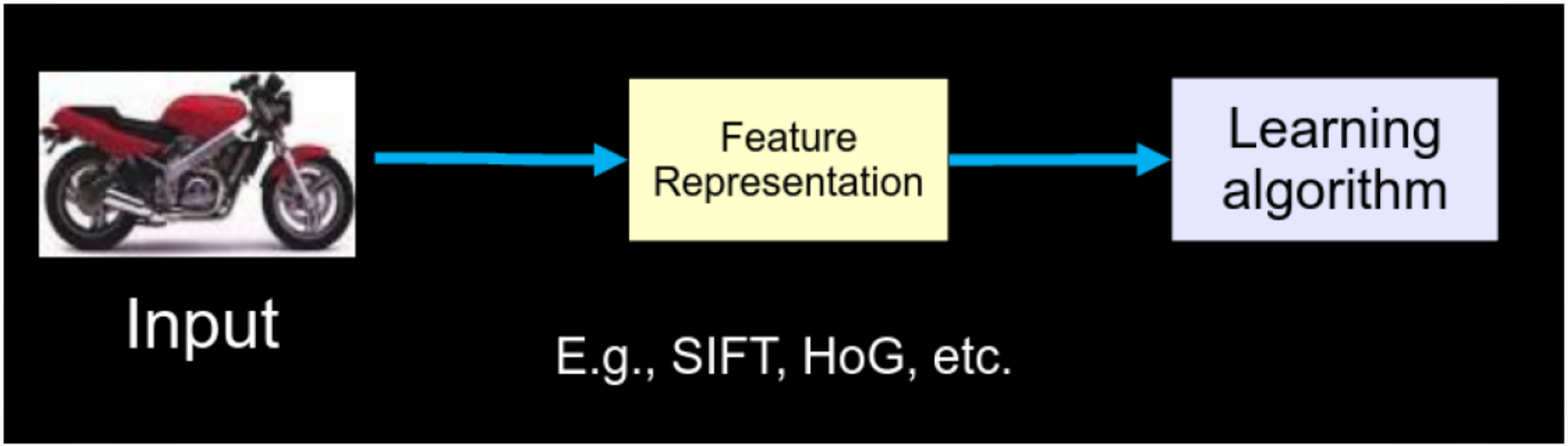


Convoluted Neural Network:Why?:What to do with this data?

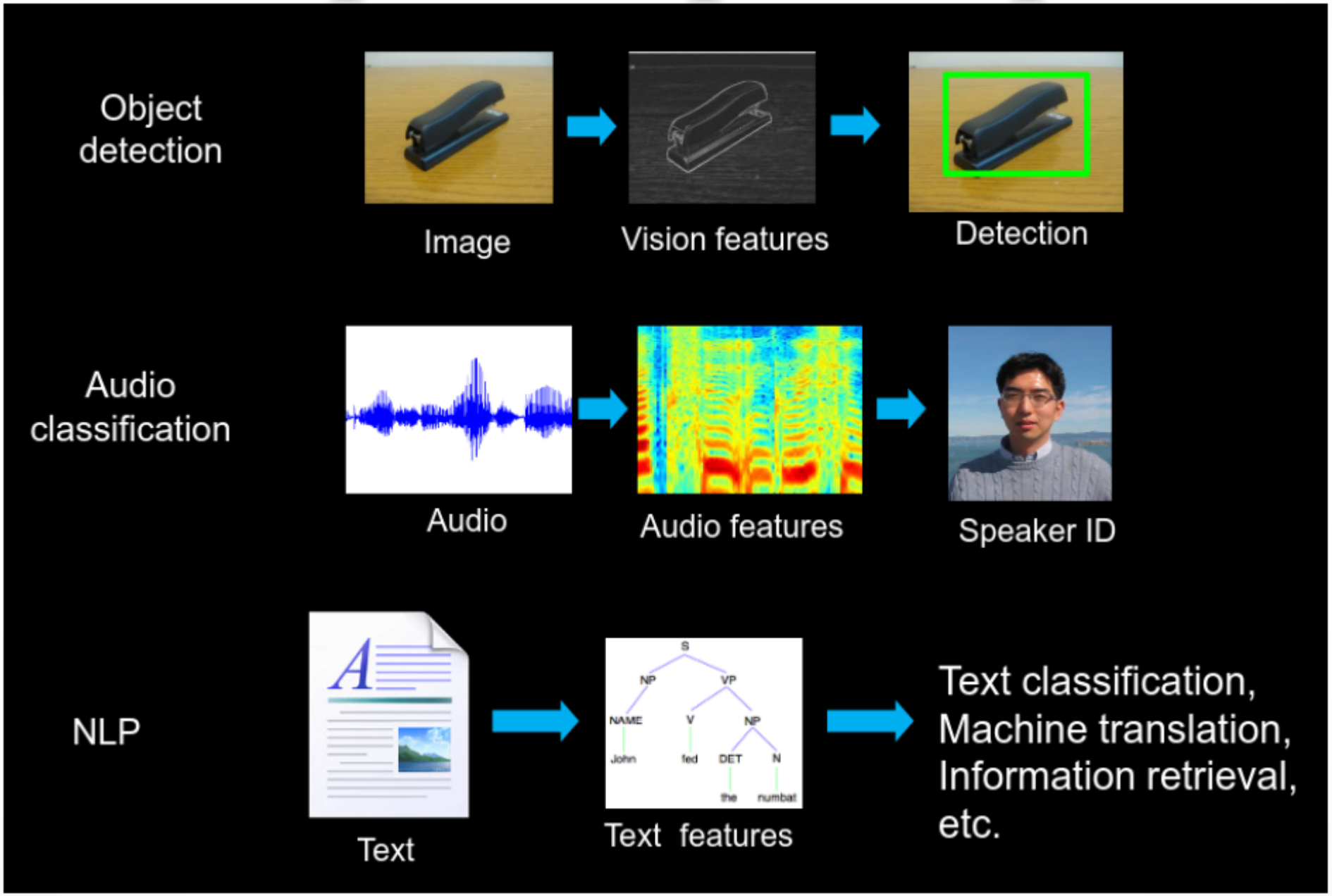


Convoluted Neural Network:Why?:Feature Selection

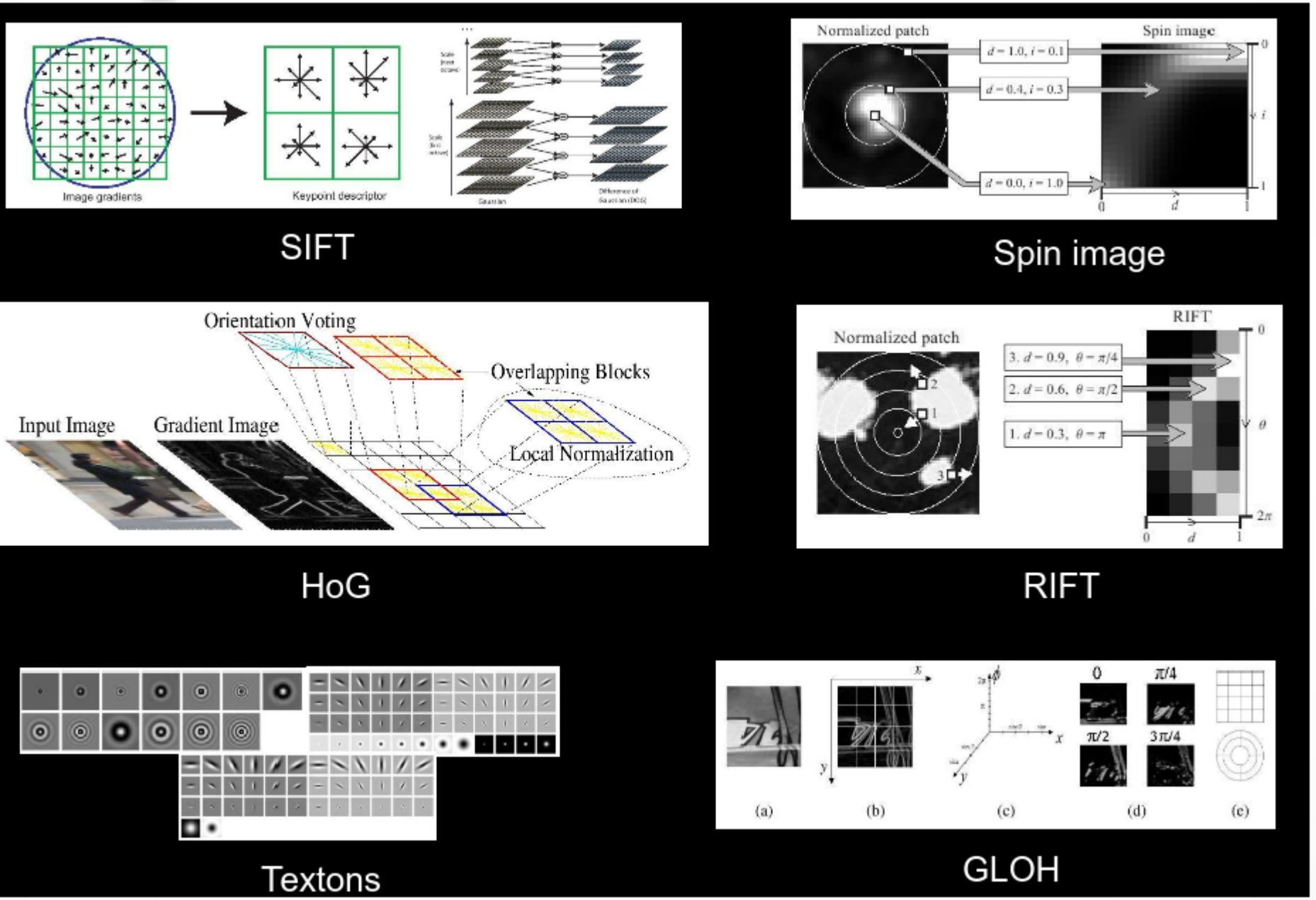
Convoluted Neural Network:Why?:Feature Selection



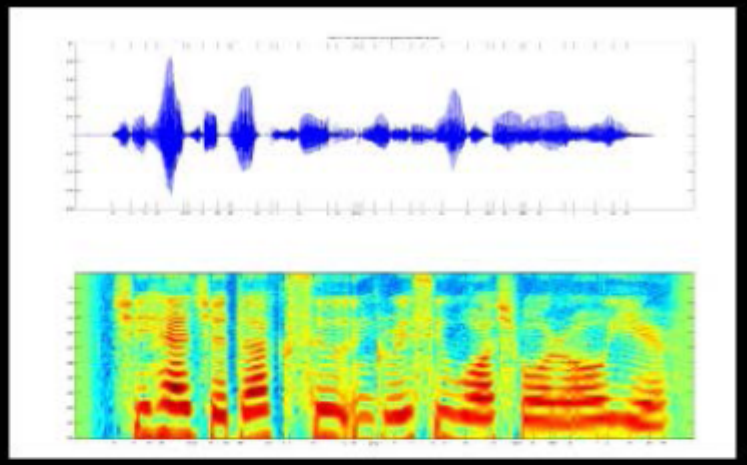
Convoluted Neural Network:Why?:How is computer Perception done?



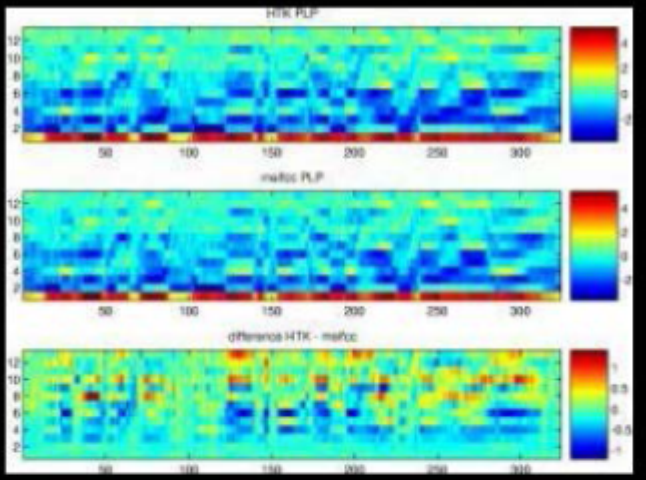
Convoluted Neural Network:Why?:Computer Vision features



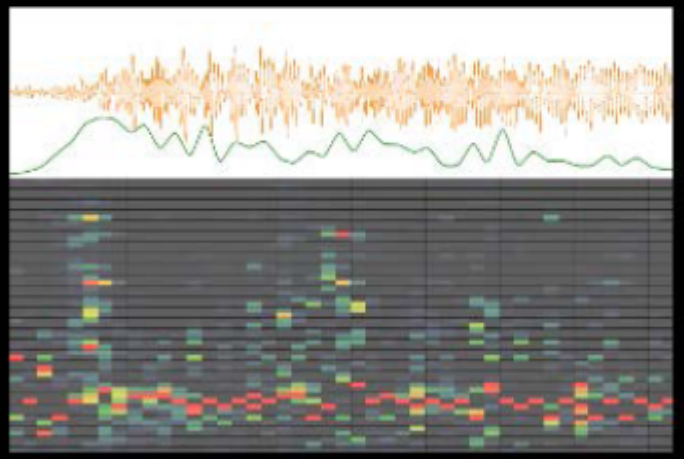
Convoluted Neural Network:Why?:Audio Features



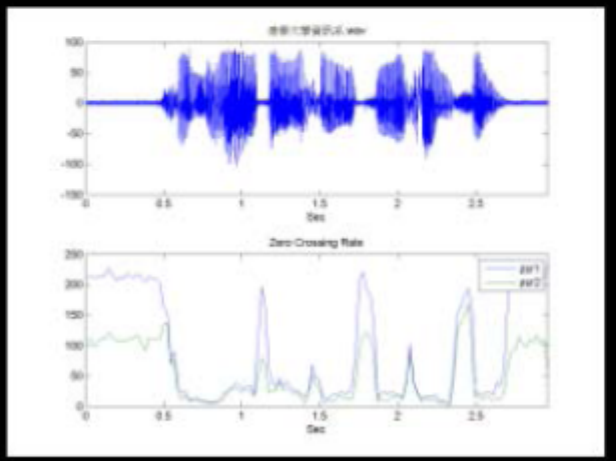
Spectrogram



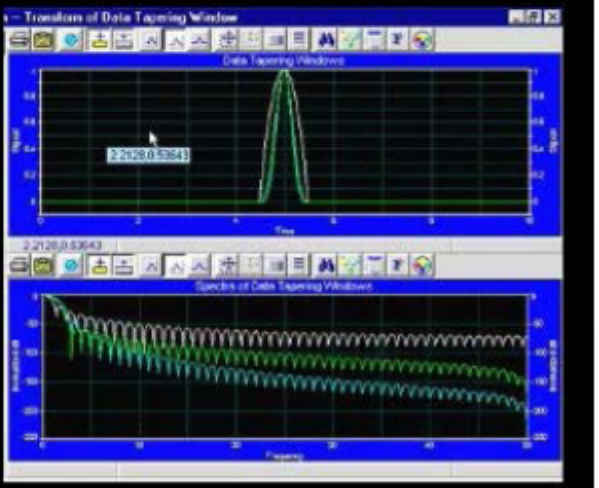
MFCC



Flux

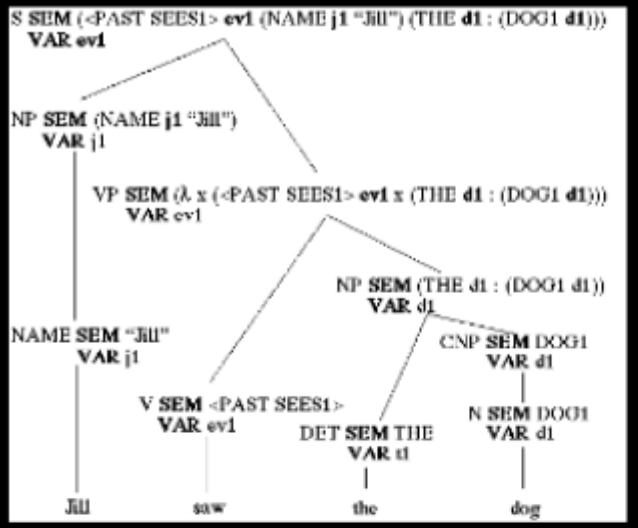


ZCR

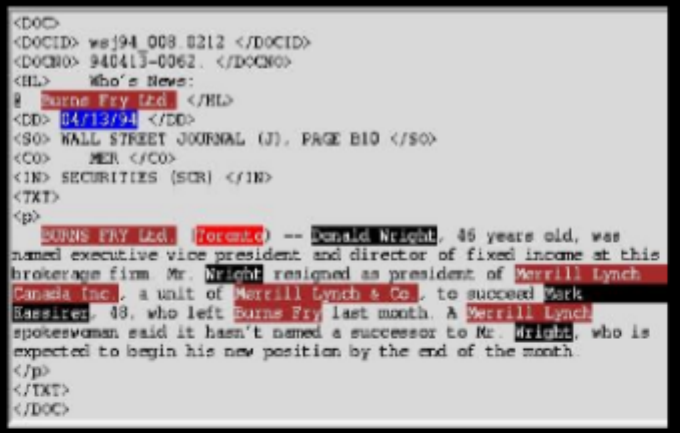


Rolloff

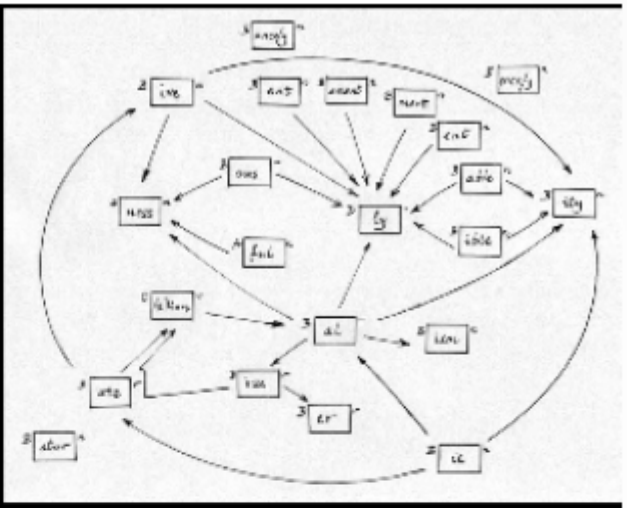
Convoluted Neural Network:Why?:NLP Features



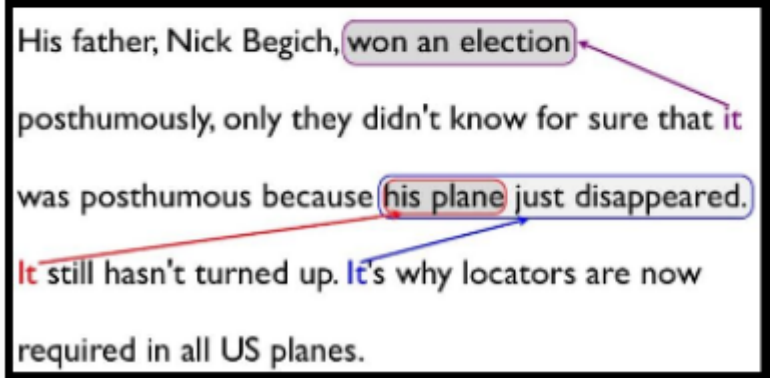
Parser features



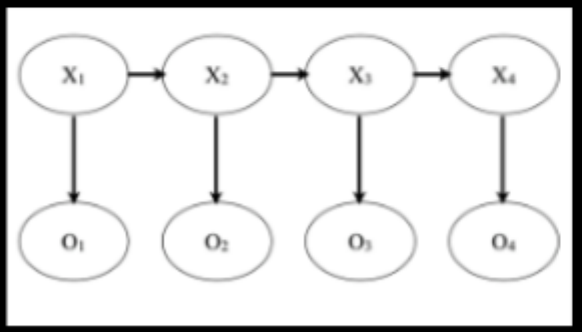
NER/SRL



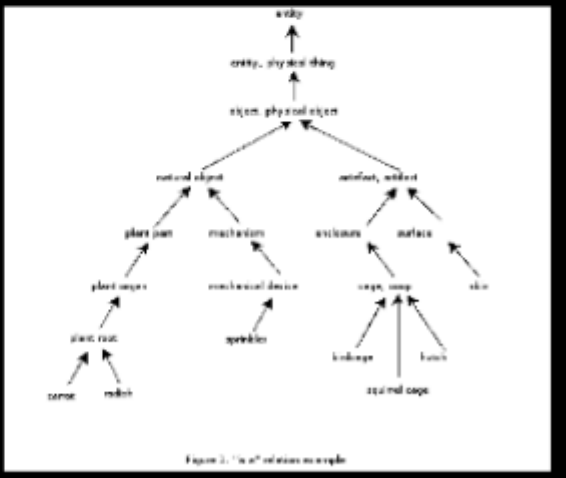
Stemming



Anaphora



POS tagging

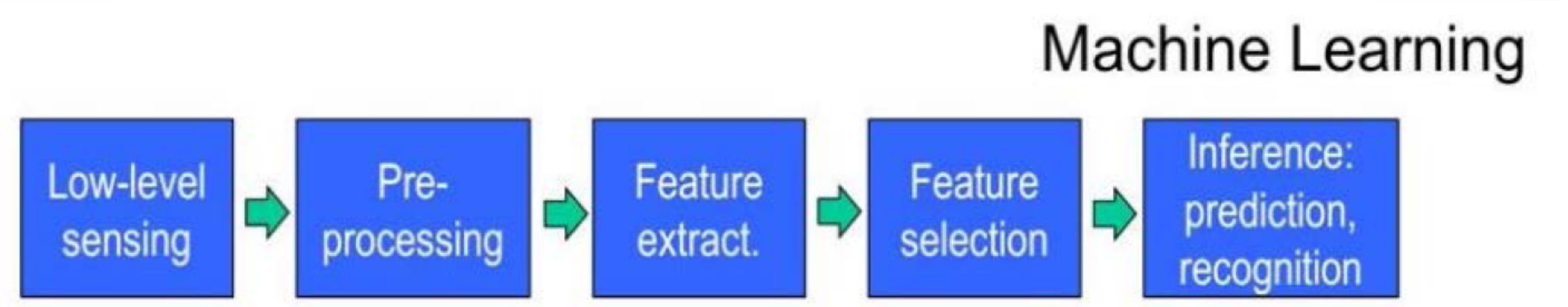


WordNet features

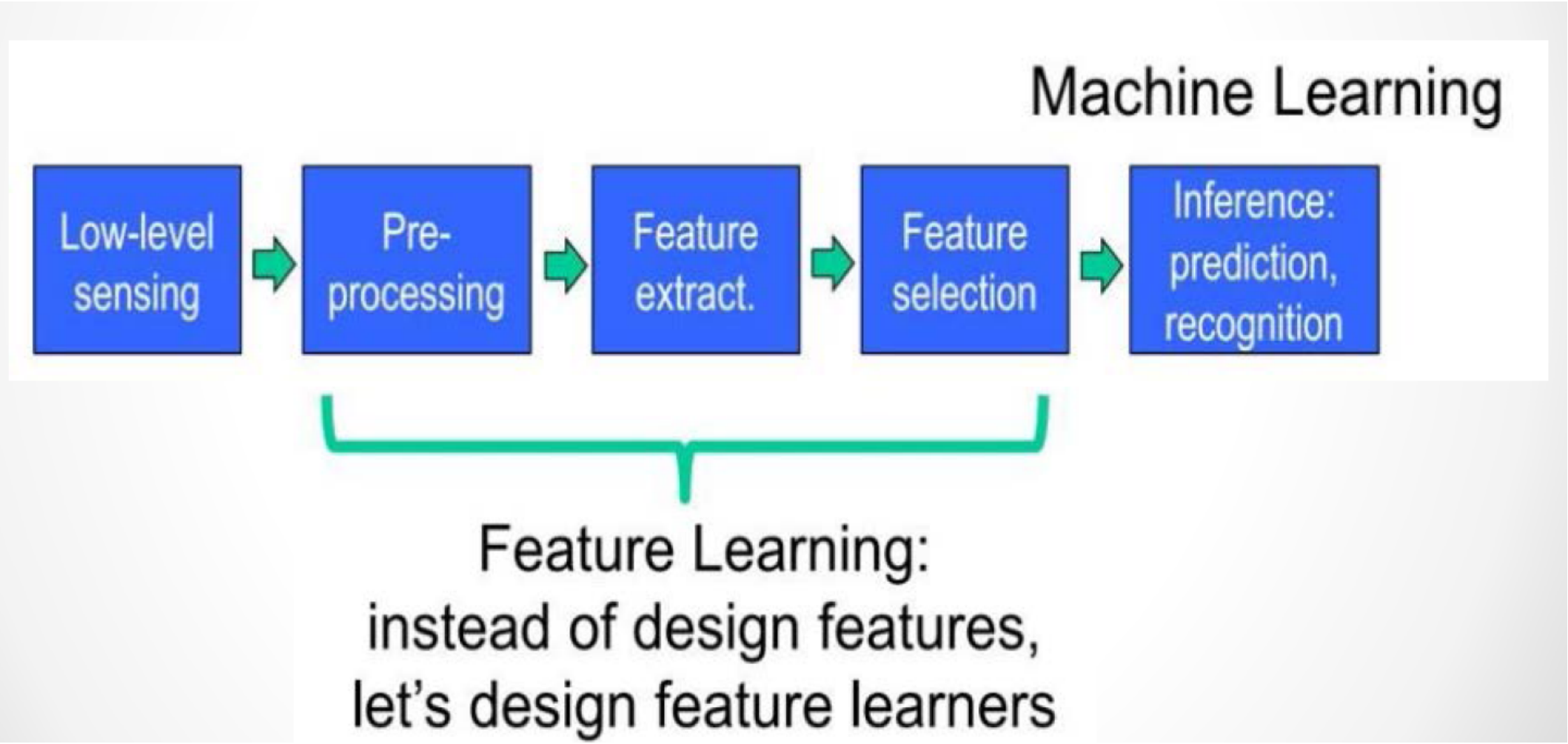
Convoluted Neural Network:Why?:Challenges

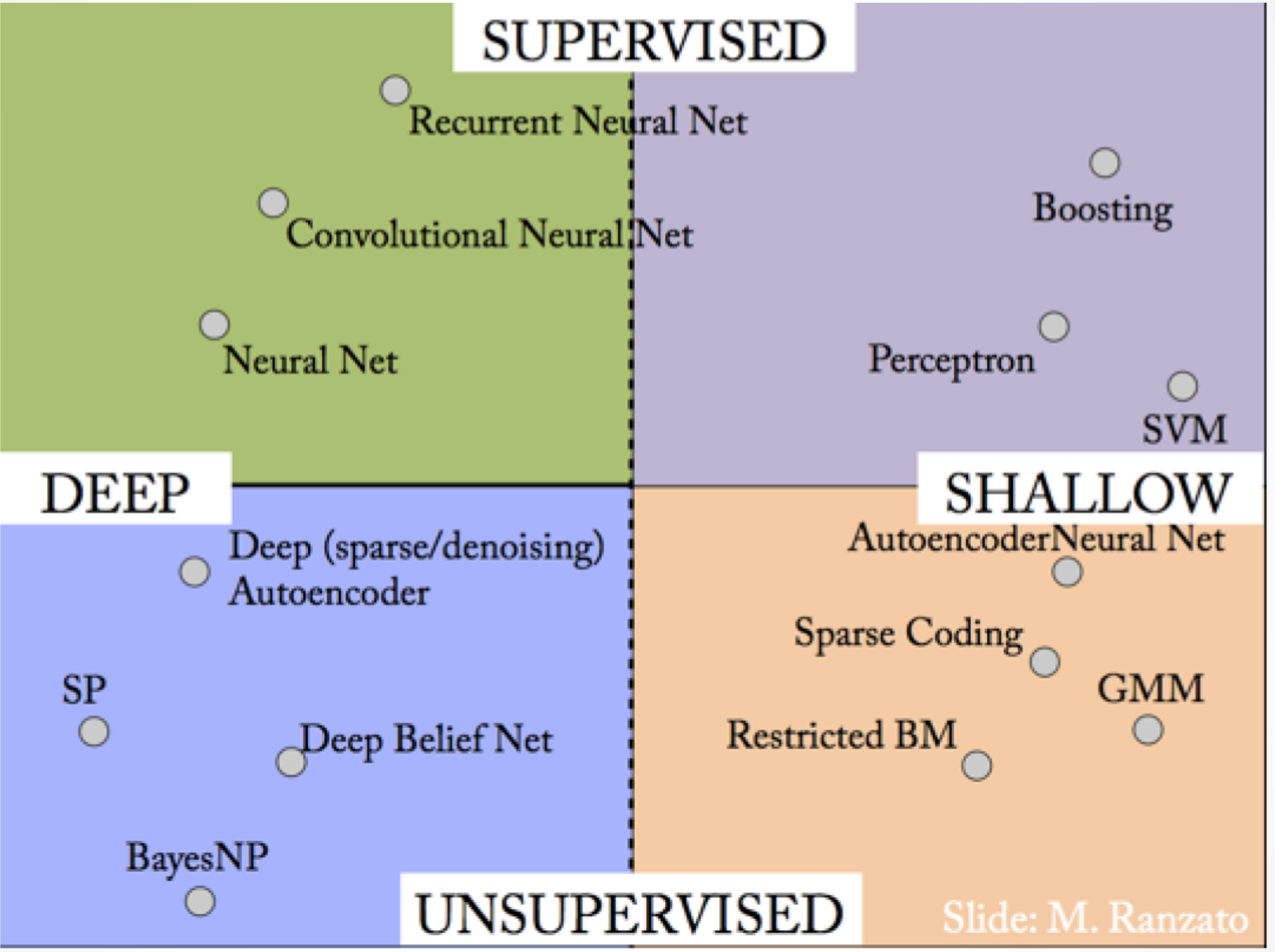
- Coming up with features is difficult,time consuming and requires expert knowledge in the domain.
- A lot of time is spend tuning features which are often handcrafted.

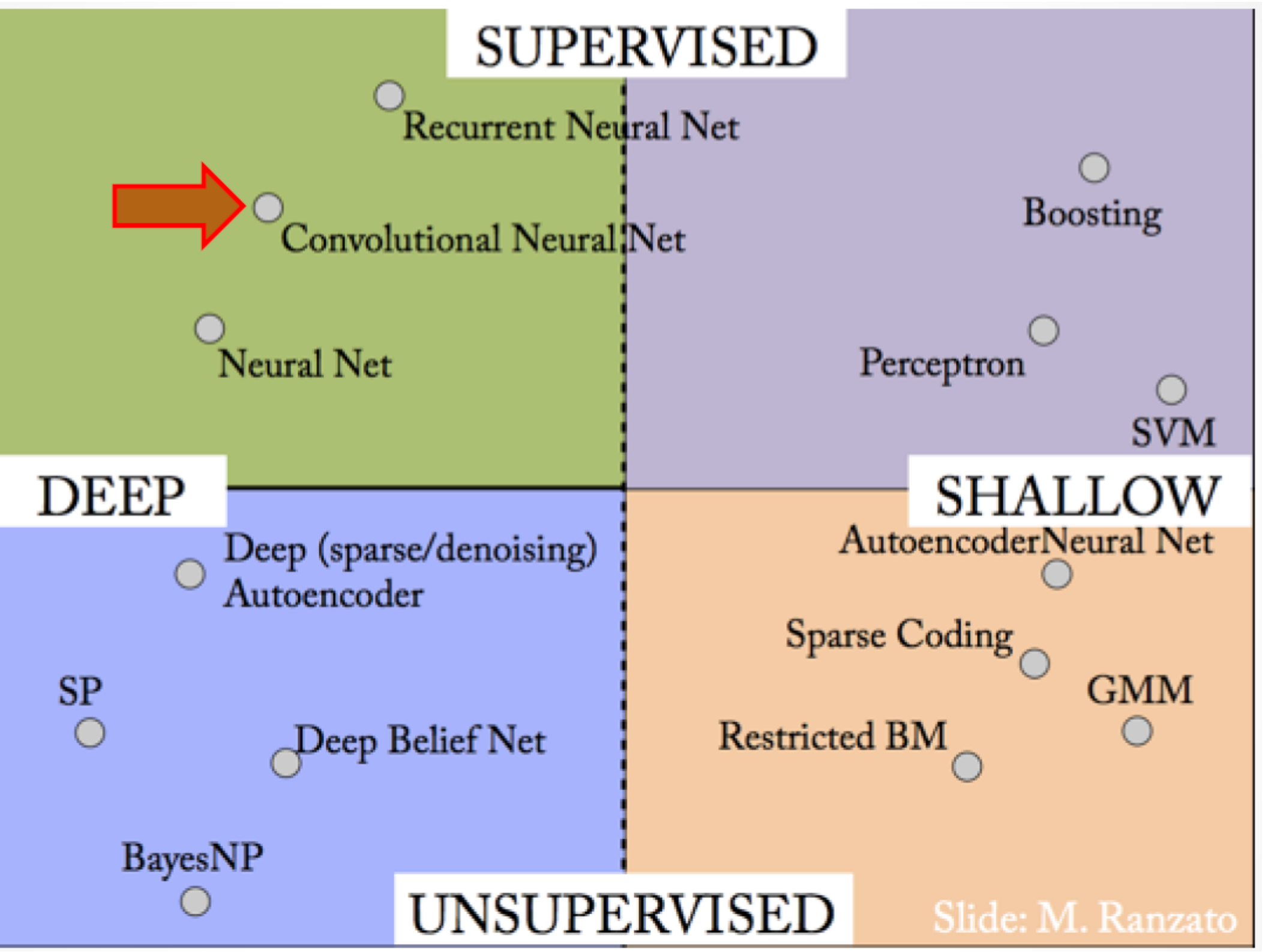
Convoluted Neural Network:Why?:Feature Representation



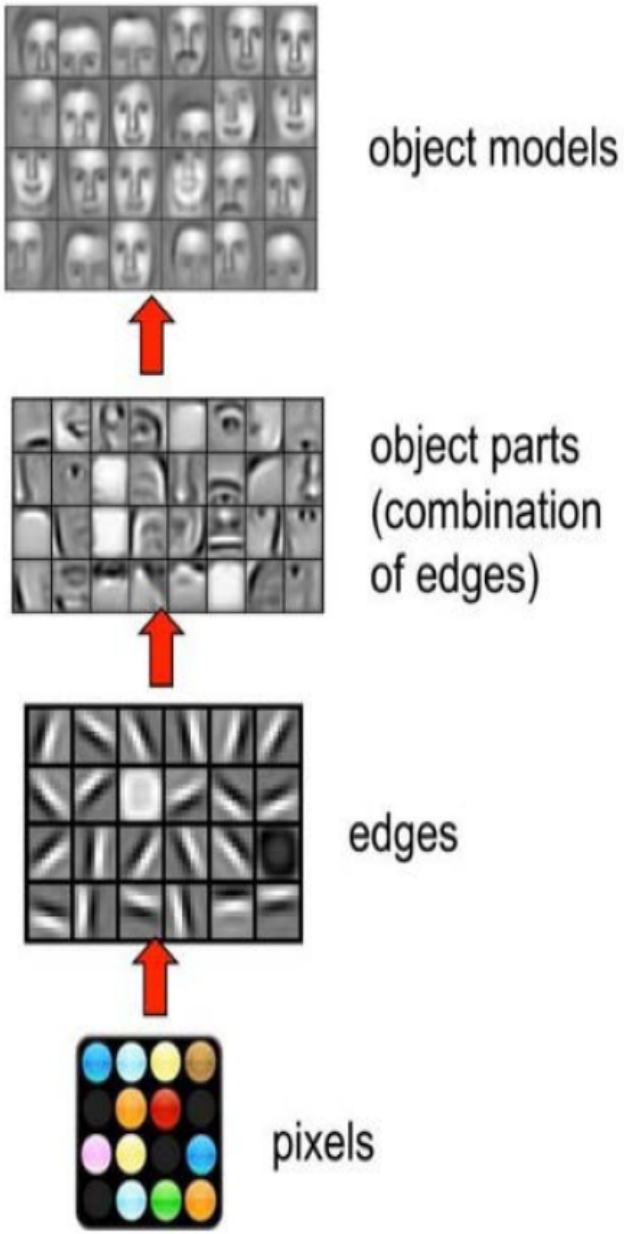
Convoluted Neural Network:Why?:Feature Representation

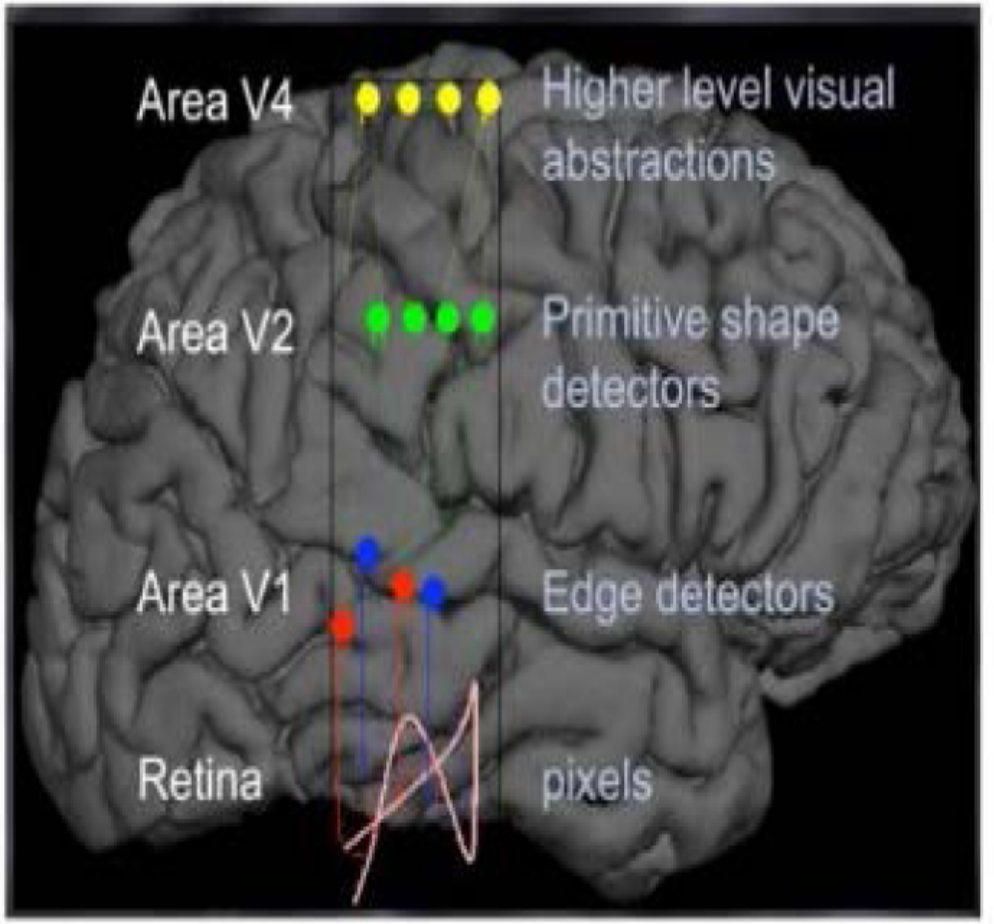



SUPERVISED
Recurrent Neural Net
Convolutional Neural Net
Neural Net
Boosting
Perceptron
SVM
DEEP
SHALLOW
Deep (sparse/denoising) Autoencoder
SP
Deep Belief Net
BayesNP
AutoencoderNeural Net
Sparse Coding
GMM
Restricted BM
UNSUPERVISED
Slide: M. Ranzato


SUPERVISED
Recurrent Neural Net
Convolutional Neural Net
Neural Net
Boosting
Perceptron
SVM
DEEP
SHALLOW
Deep (sparse/denoising) Autoencoder
AutoencoderNeural Net
Sparse Coding
GMM
SP
Deep Belief Net
Restricted BM
BayesNP
UNSUPERVISED
Slide: M. Ranzato

Convoluted Neural Network:Why?:Biological Inspiration

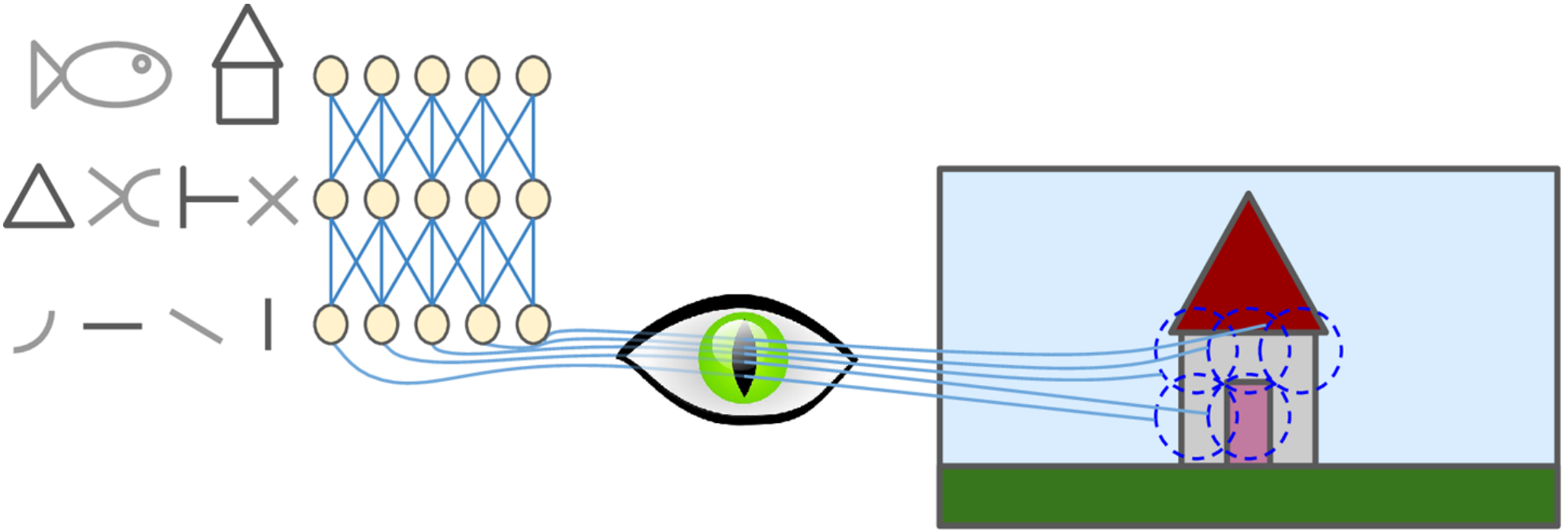




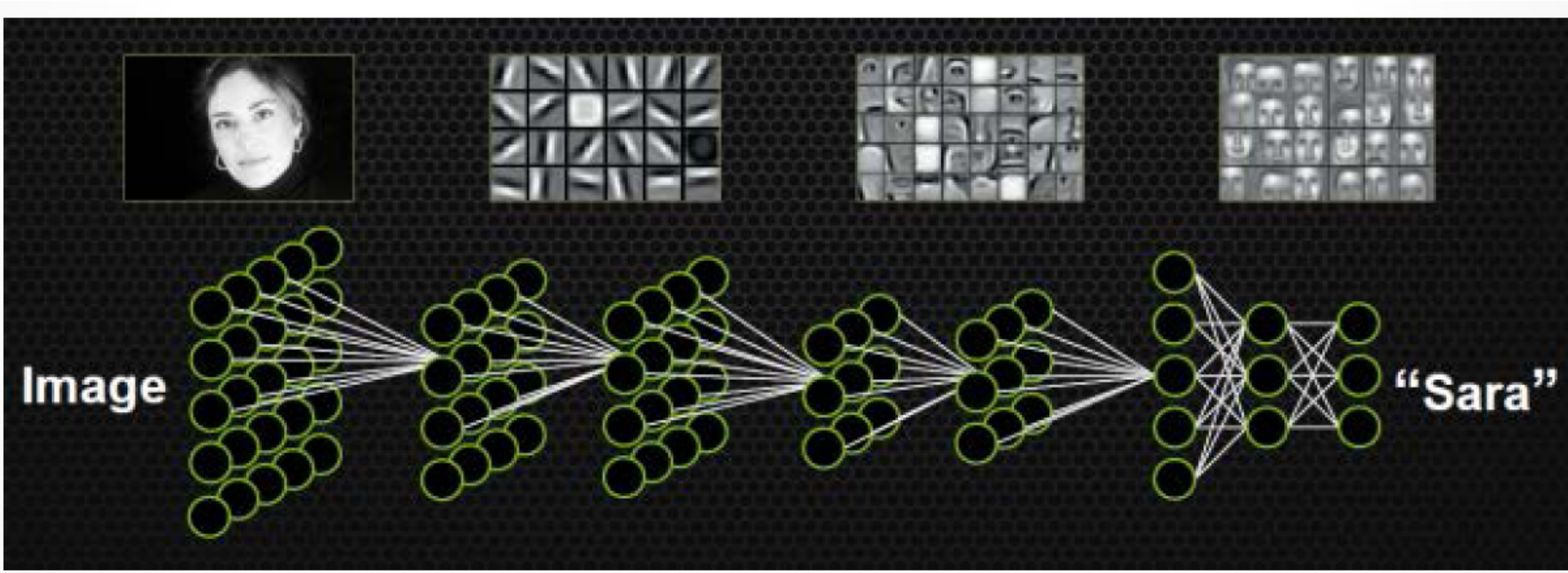
Convoluted Neural Network:Why?Biological Inspiration

- David H. Hubel and Torsten Wiesel performed a series of experiments on cats in 19581 and 19592 (and a few years later on monkeys3), giving crucial insights on the structure of the visual cortex (the authors received the Nobel Prize in Physiology or Medicine in 1981 for their work).
- In particular, they showed that many neurons in the visual cortex have a small local receptive field, meaning they react only to visual stimuli located in a limited region of the visual field .
- The receptive fields of different neurons may overlap, and together they tile the whole visual field. Moreover, the authors showed that some neurons react only to images of horizontal lines, while others react only to lines with different orientations (two neurons may have the same receptive field but react to different line orientations).
- They also noticed that some neurons have larger receptive fields, and they react to more complex patterns that are combinations of the lower-level patterns. These observations led to the idea that the higher-level neurons are based on the outputs of neighboring lower-level neurons . This powerful architecture is able to detect all sorts of complex patterns in any area of the visual field.

Convoluted Neural Network:Why?Biological Inspiration



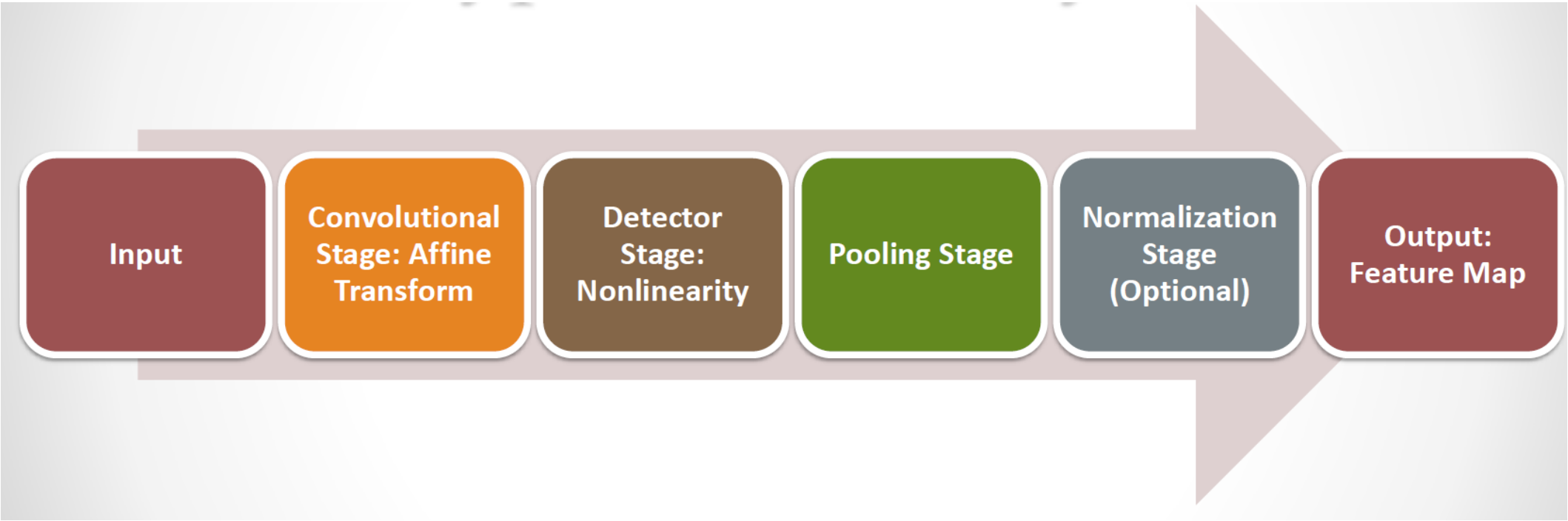
Convoluted Neural Network:Why?Features learnt by Deep training



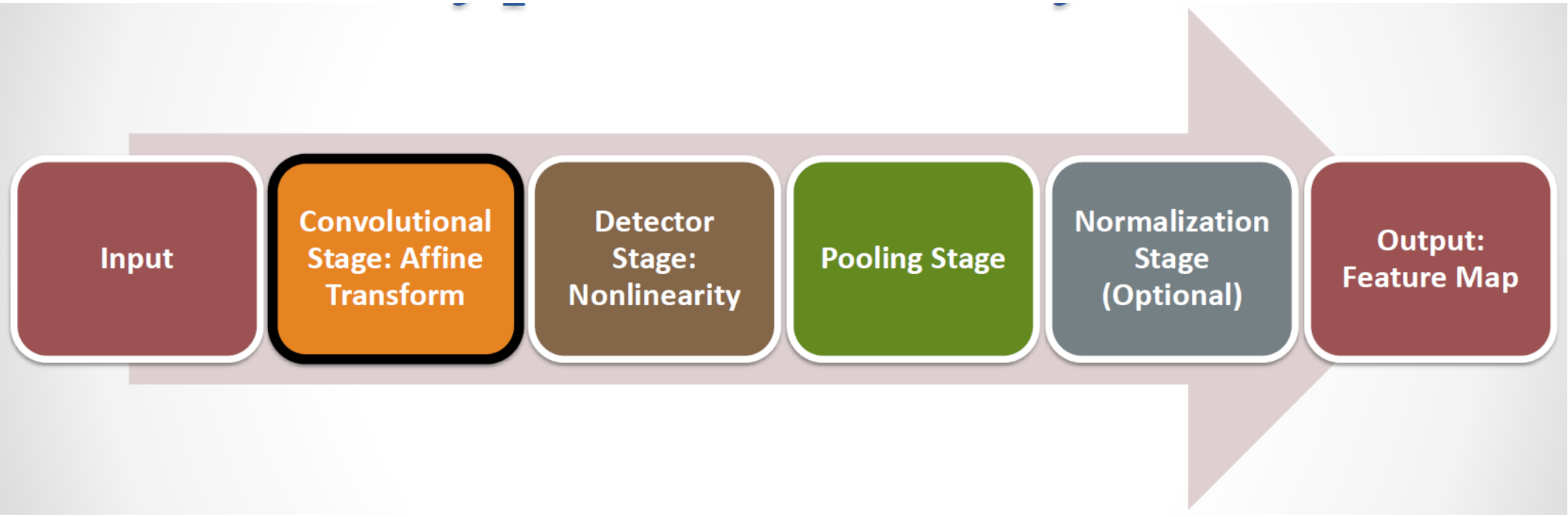
Convoluted Neural Network:What is it?

- Essentially neural networks that use convolution in place of general matrix multiplication in at least one of their layers.
  - Locally Receptive Fields
  - Shared Weights
  - Spatial or Temporal Sub-sampling

Convoluted Neural Network:Typical Architecture



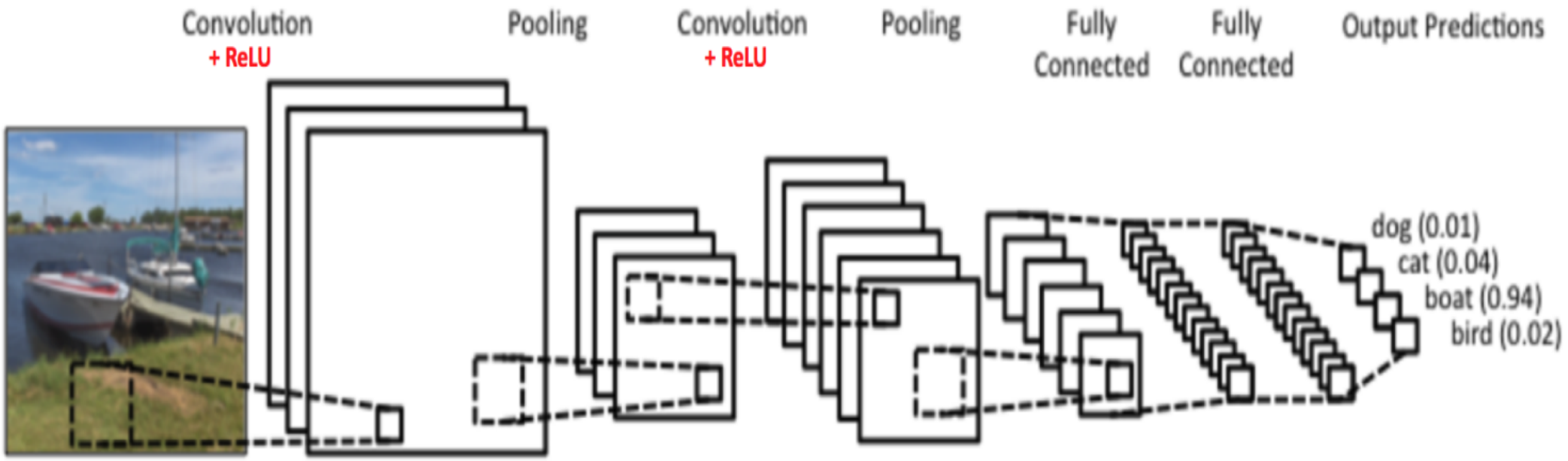
Convoluted Neural Network:Typical Architecture



Convoluted Neural Network:Typical Architecture

- LeNet was one of the very first convolutional neural networks which helped propel the field of Deep Learning.
- This pioneering work by Yann LeCun was named **LeNet5** after many previous successful iterations since the year 1988 [3].
- At that time the LeNet architecture was used mainly for character recognition tasks such as reading zip codes, digits, etc.

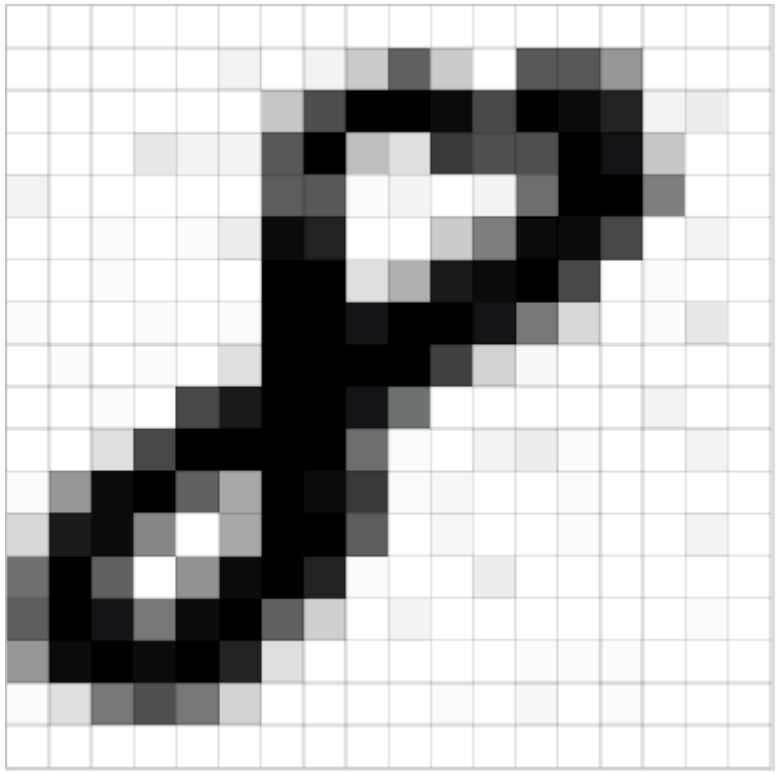
Convoluted Neural Network:Lenet



Convoluted Neural Network:Typical Architecture

- There are four main operations in the ConvNet:
  - Convolution
  - Non Linearity (ReLU)
  - Pooling or Sub Sampling
  - Classification (Fully Connected Layer)

Convoluted Neural Network:Lenet



0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	12	0	11	39	137	37	0	152	147	84	0	0	0
0	0	1	0	0	0	41	160	250	255	235	162	255	238	206	11	13	0
0	0	0	16	9	9	150	251	45	21	184	159	154	255	233	40	0	0
10	0	0	0	0	0	145	146	3	10	0	11	124	253	255	107	0	0
0	0	3	0	4	15	236	216	0	0	38	109	247	240	169	0	11	0
1	0	2	0	0	0	253	253	23	62	224	241	255	164	0	5	0	0
6	0	0	4	0	3	252	250	228	255	255	234	112	28	0	2	17	0
0	2	1	4	0	21	255	253	251	255	172	31	8	0	1	0	0	0
0	0	4	0	163	225	251	255	229	120	0	0	0	0	0	11	0	0
0	0	21	162	255	255	254	255	126	6	0	10	14	6	0	0	9	0
3	79	242	255	141	66	255	245	189	7	8	0	0	5	0	0	0	0
26	221	237	98	0	67	251	255	144	0	8	0	0	7	0	0	11	0
125	255	141	0	87	244	255	208	3	0	0	13	0	1	0	1	0	0
145	248	228	116	235	255	141	34	0	11	0	1	0	0	0	1	3	0
85	237	253	246	255	210	21	1	0	1	0	0	6	2	4	0	0	0
6	23	112	157	114	32	0	0	0	0	2	0	8	0	7	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Convoluted Neural Network:Lenet:Convolutions

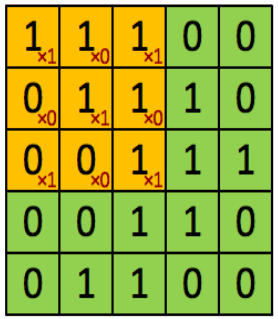
1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

. Consider a 5X5 image whose pixel value are only 1 or 0 as a simplification.

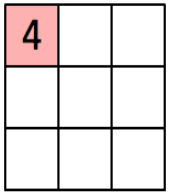
1	0	1
0	1	0
1	0	1

. Also consider 3X3 matrix, called a filter, or kernel, or feature detector.

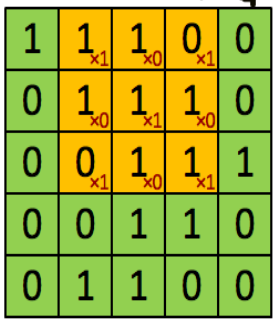
Convoluted Neural Network:Lenet:Convolutions



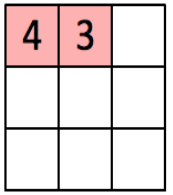
Image



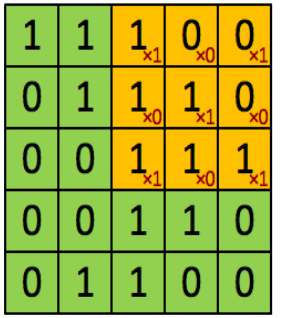
Convolved Feature



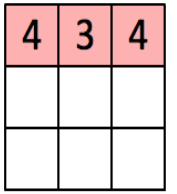
Image



Convolved Feature

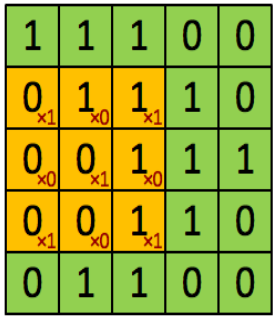


Image

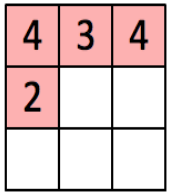


Convolved Feature

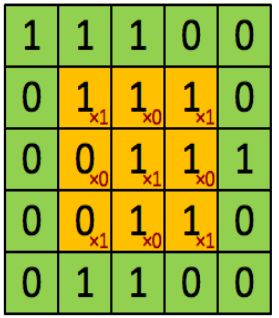
- Called a stride, 1 pixel in this case.



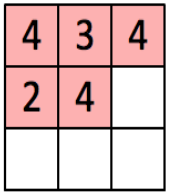
Image



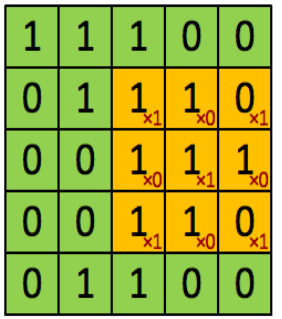
Convolved Feature



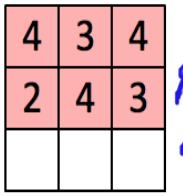
Image



Convolved Feature

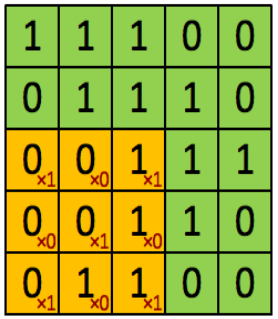


Image

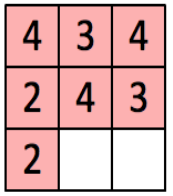


Convolved Feature

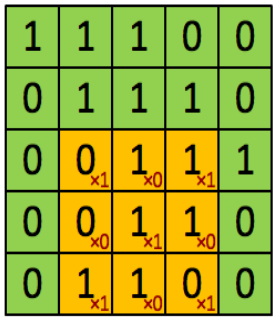
- Also called Activation map or feature map



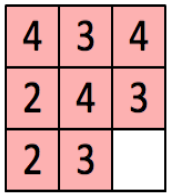
Image



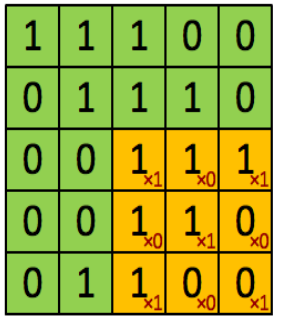
Convolved Feature



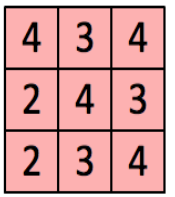
Image



Convolved Feature



Image

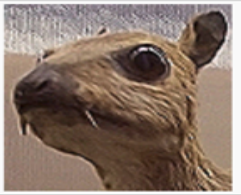


Convolved Feature

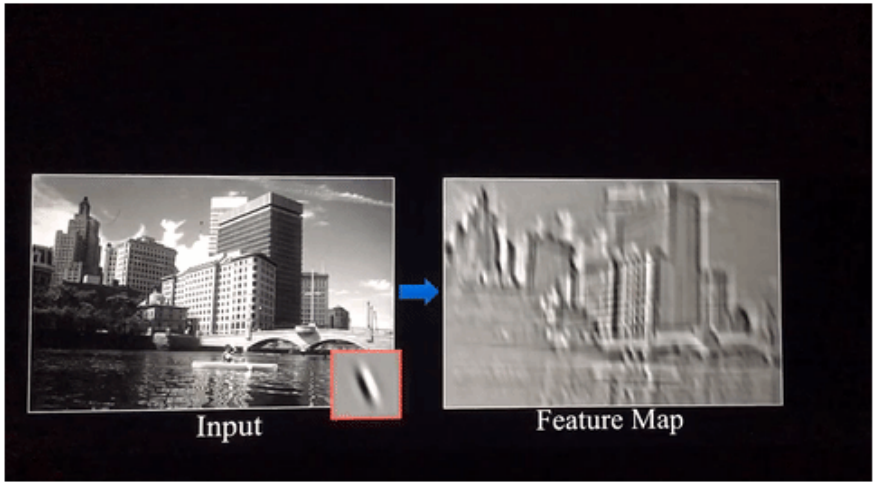
- Different filters obviously produce different features.

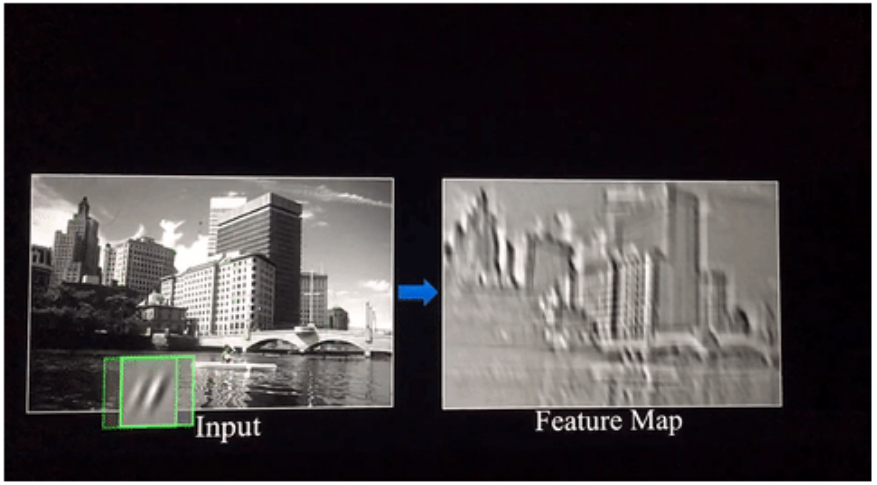
Convoluted Neural Network:Lenet:Convolutions

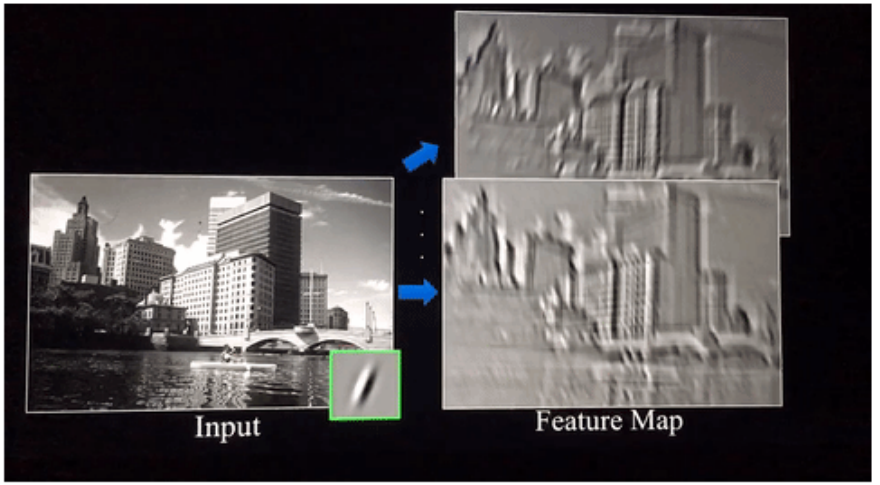
original image

Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Convoluted Neural Network:Lenet:Convolutions

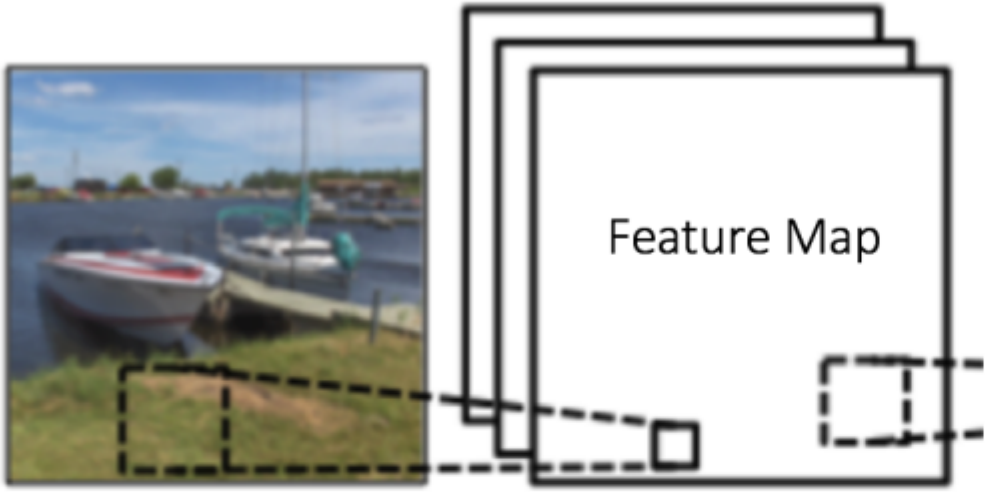






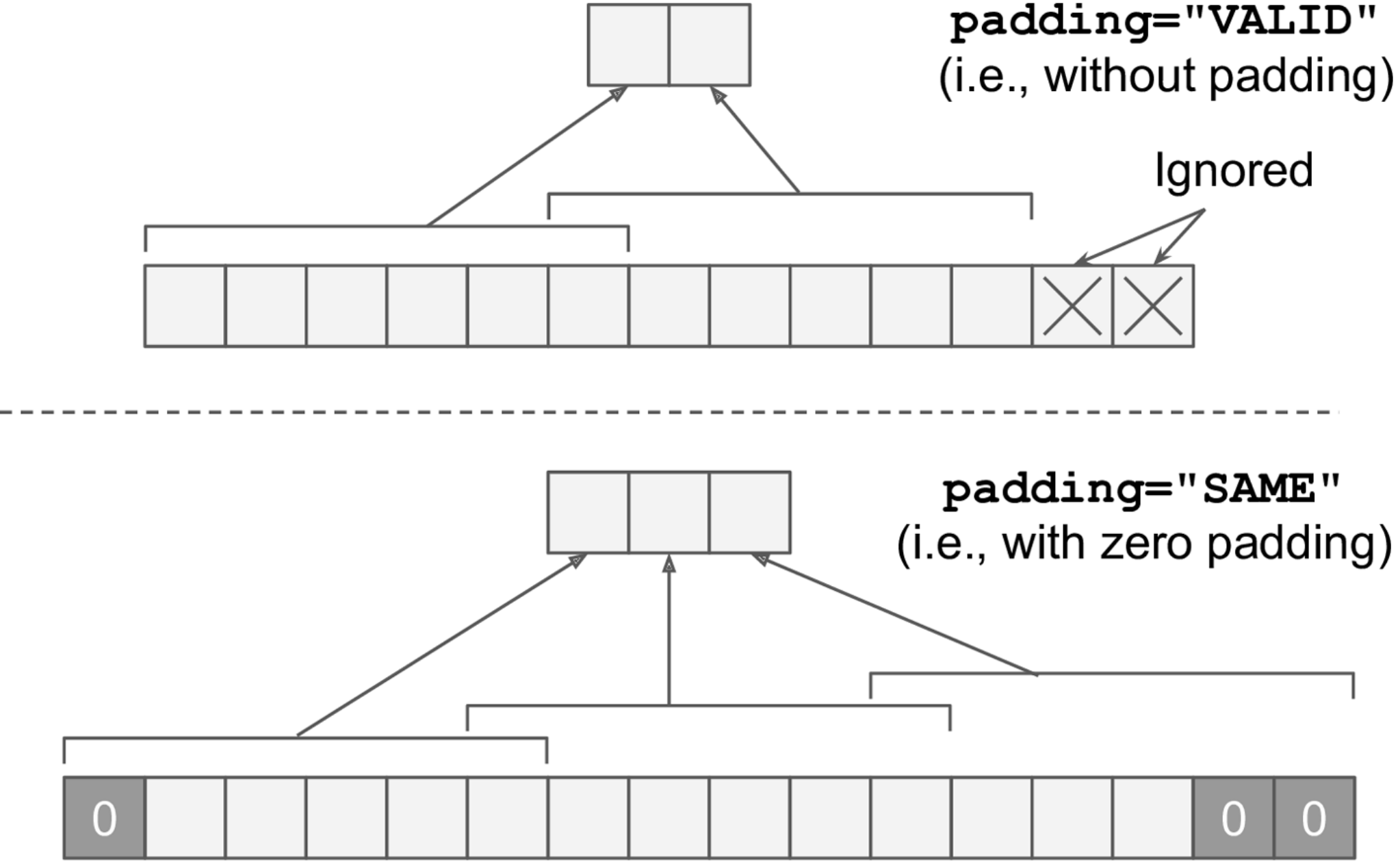
- More filters we have, more features maps we have.

Convoluted Neural Network:Lenet:Convolutions

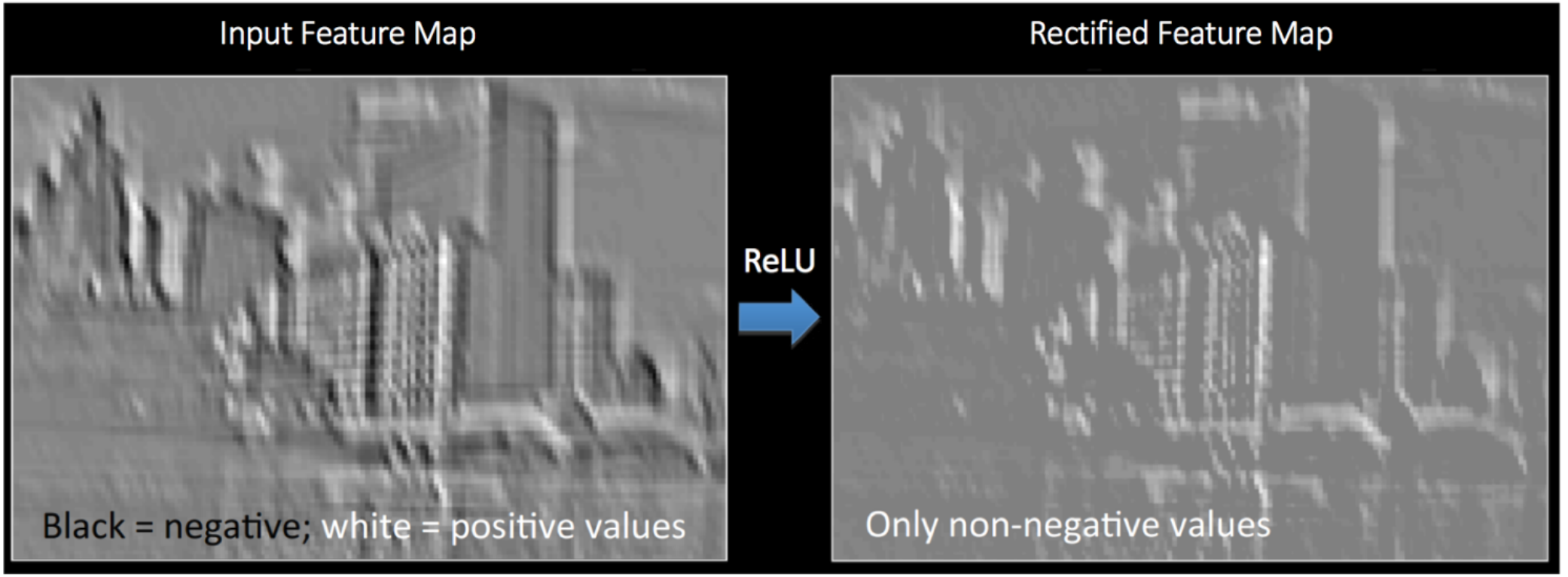


- Depth corresponds to the number of features.

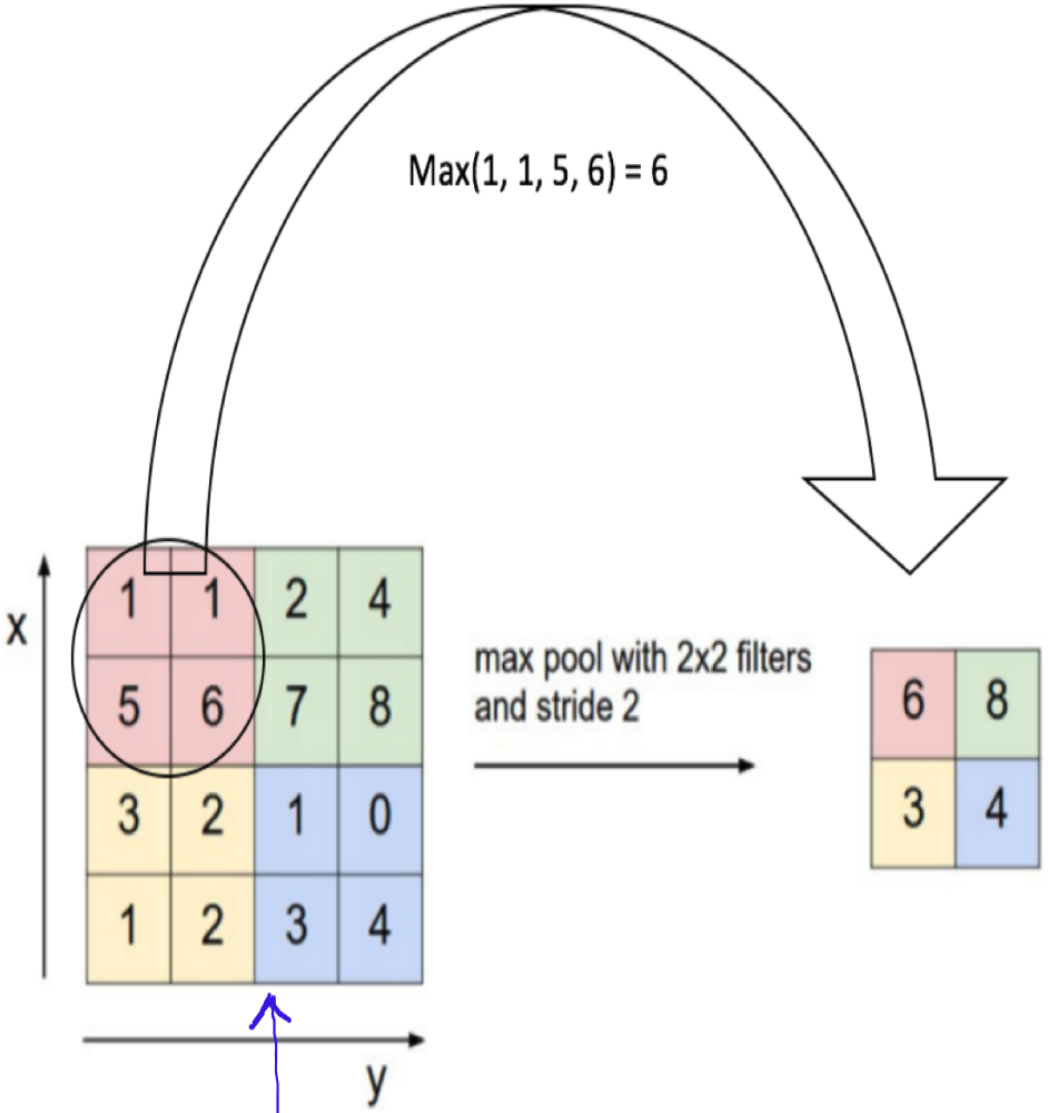
Convoluted Neural Network:Lenet:Convolutions



Convoluted Neural Network:Lenet:Convolutions:Relu

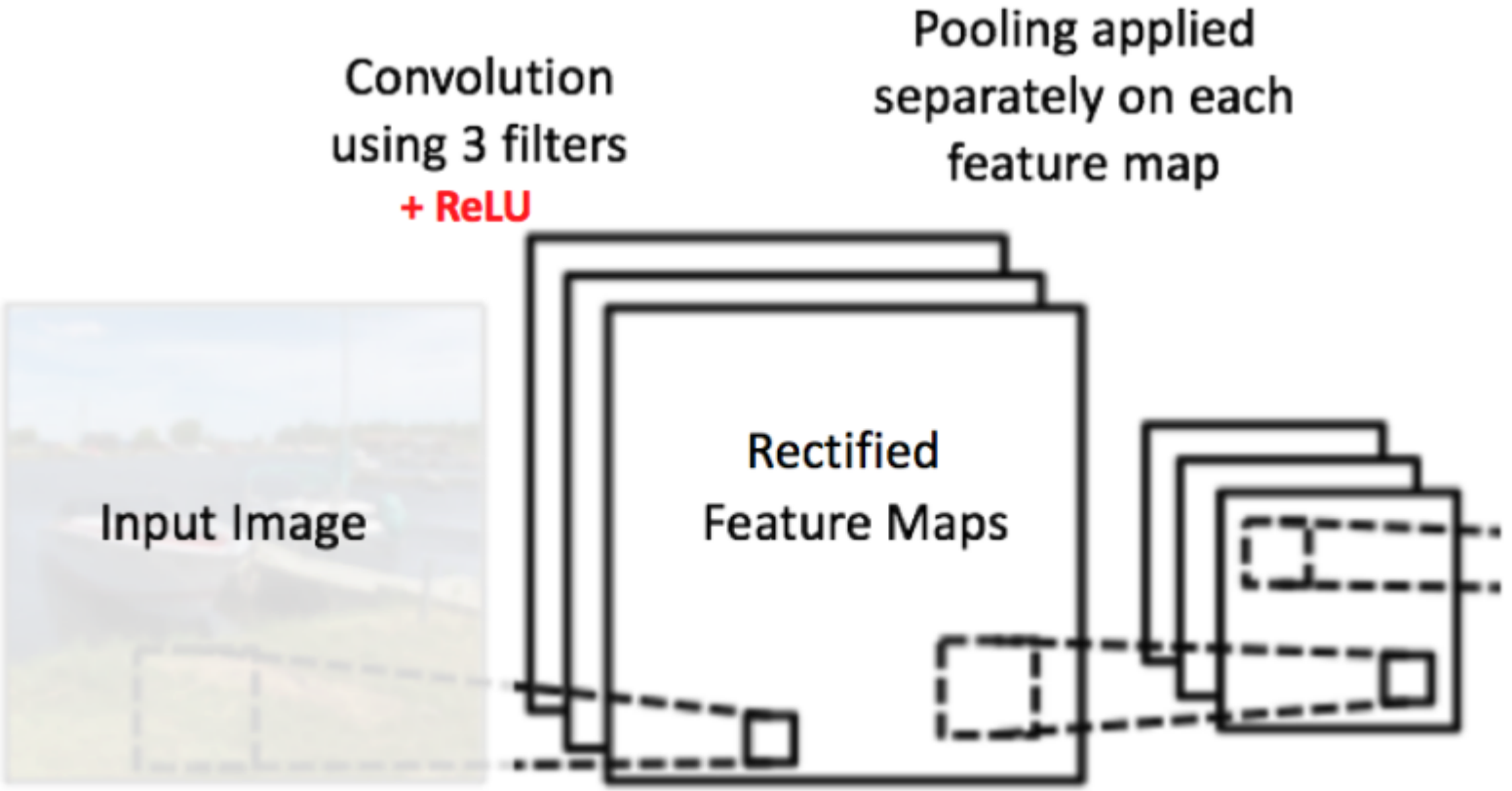


Convoluted Neural Network:Lenet:Pooling

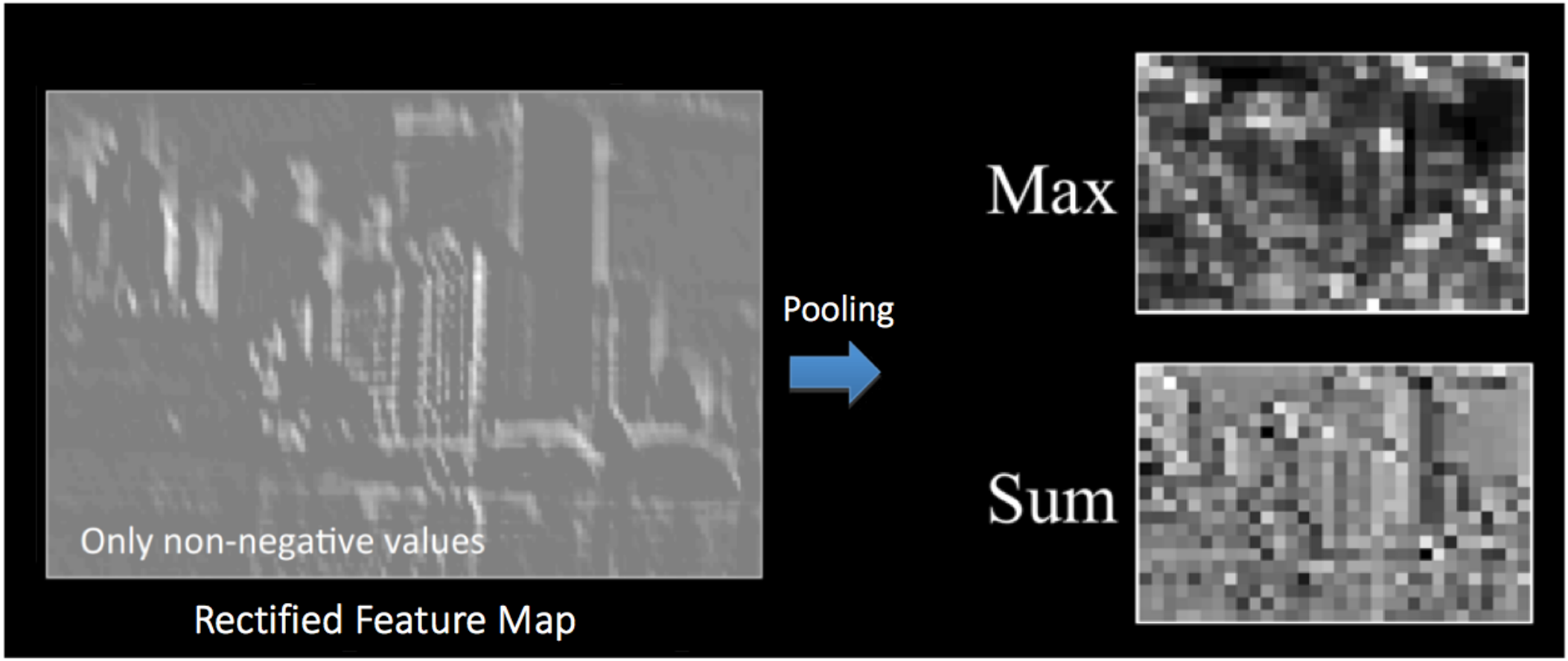


- Spatial pooling also called subsampling or down sampling.
- Different kinds of pooling can be done, Max, Average, sum.
- In practice max pooling works better.

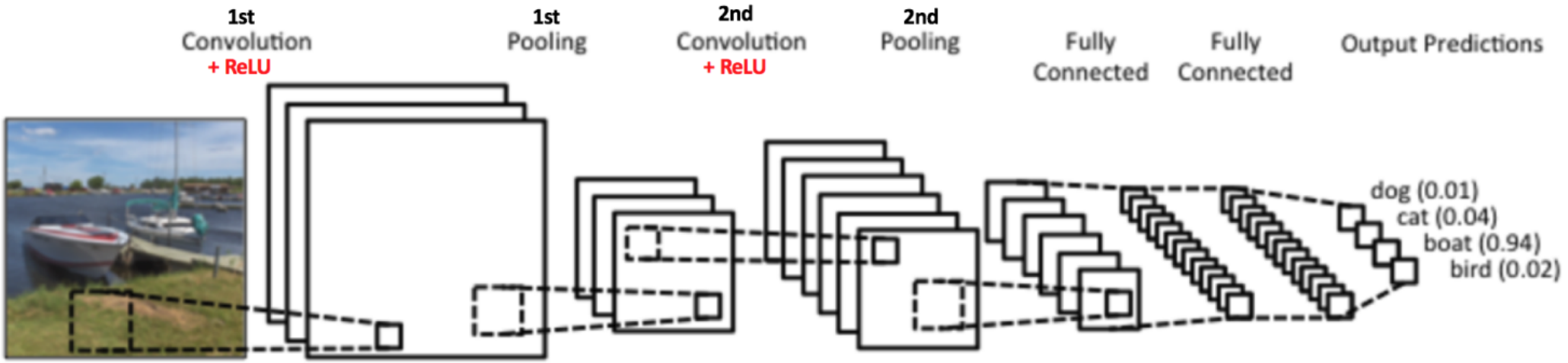
Convoluted Neural Network:Lenet:Pooling



Convoluted Neural Network:Lenet:Pooling

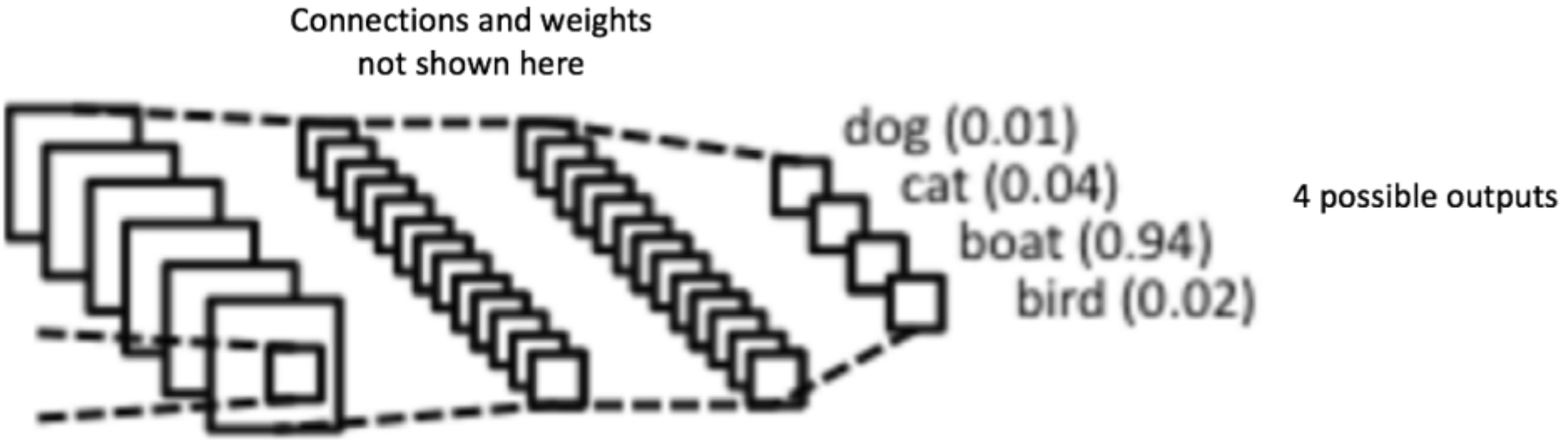


Convoluted Neural Network:Lenet

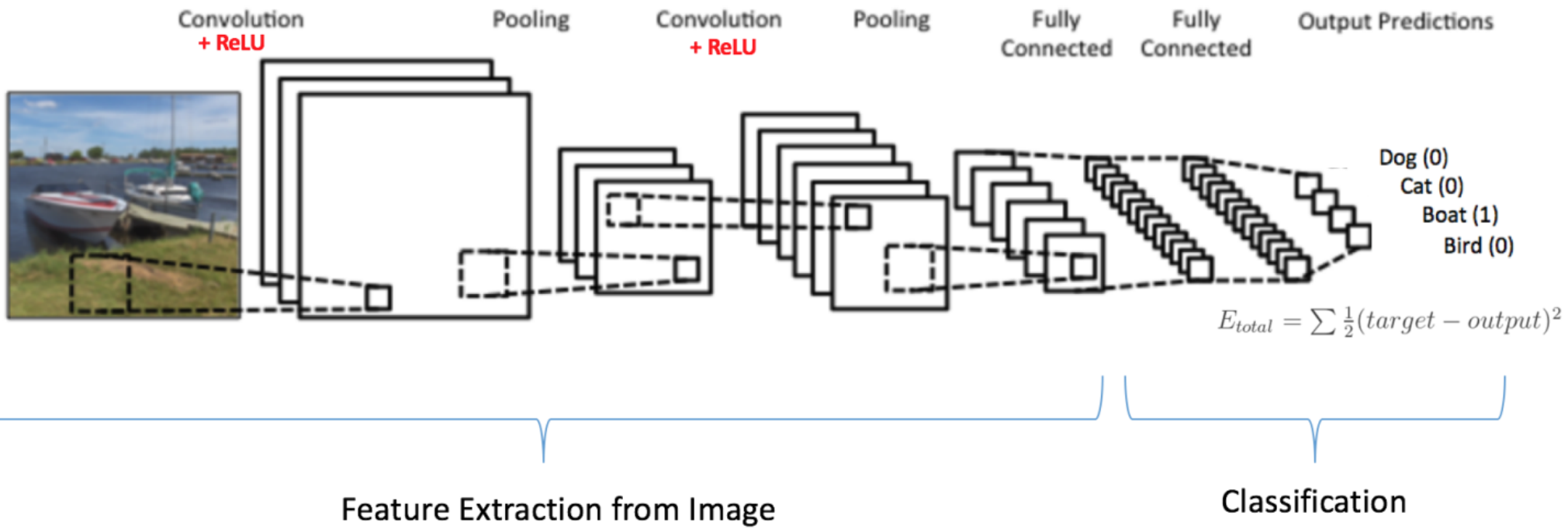


- 1st Convolution creates 3 feature maps.
- pooling
- 2nd Convolution creates 6 feature map
- pooling
- Finally FC layer and a softmax for classification.

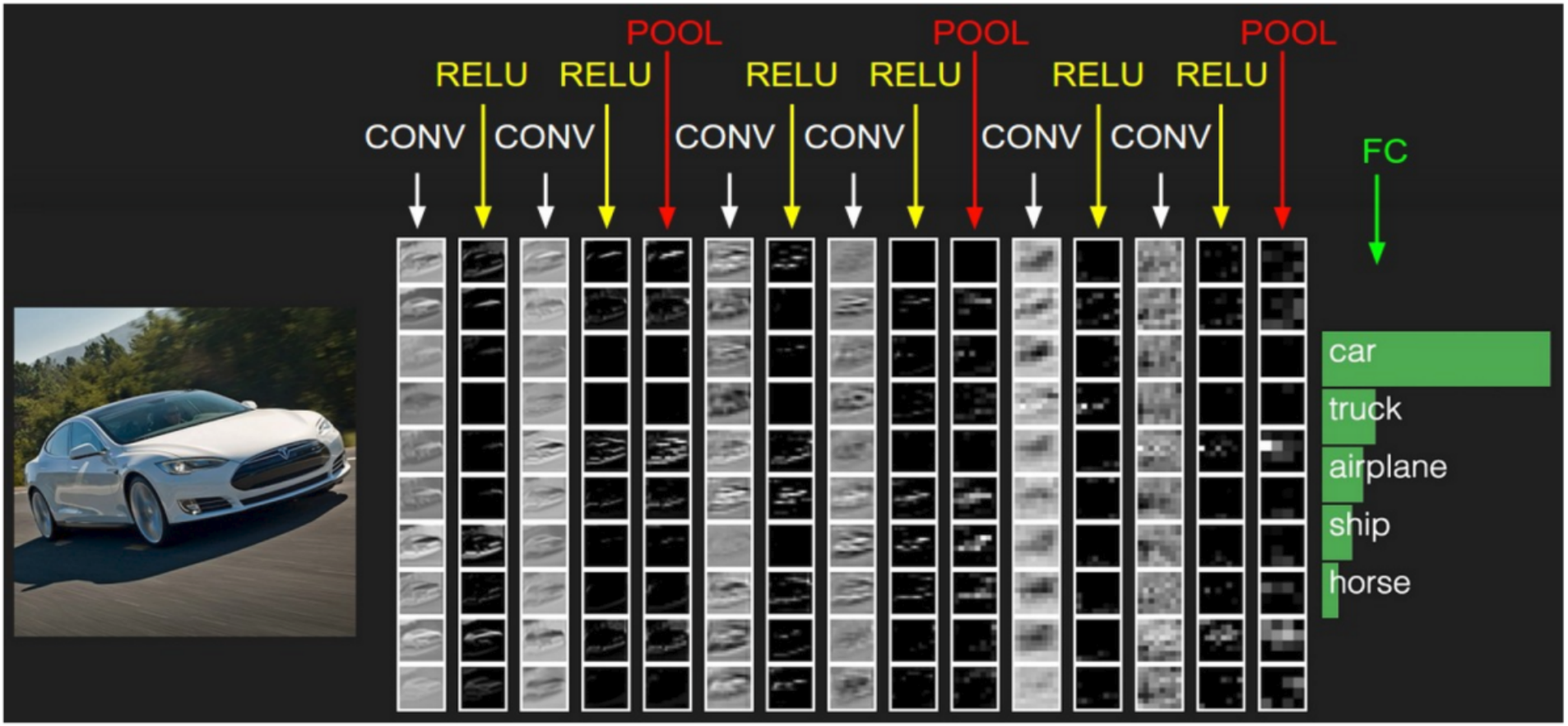
Convoluted Neural Network:Lenet:Fully Connected



Convoluted Neural Network:Lenet

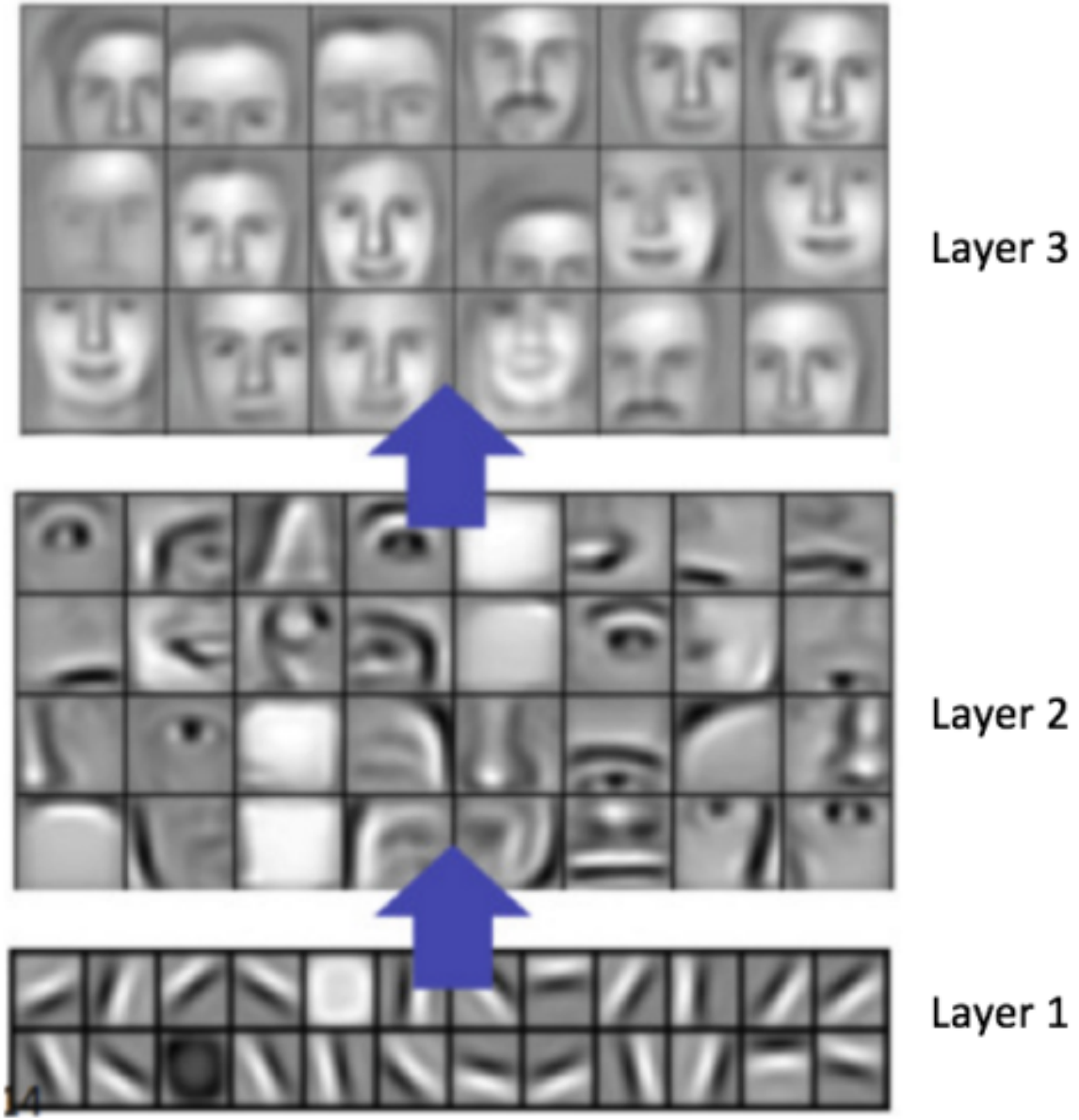


Convoluted Neural Network:Visualization

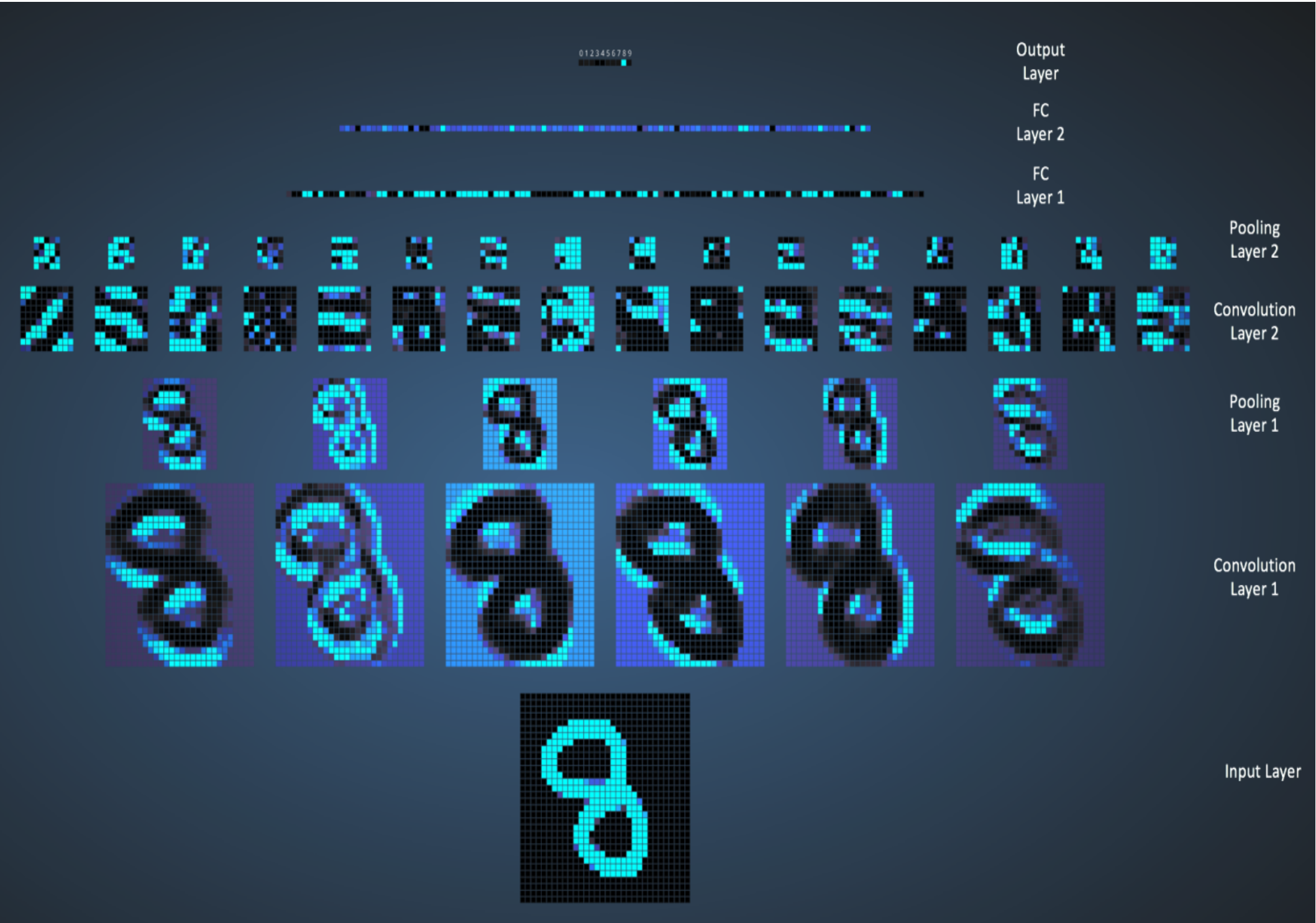


- These operations can be repeated any number of times
- Best Performing CNN, have 10. of Convolution and pooling layer.

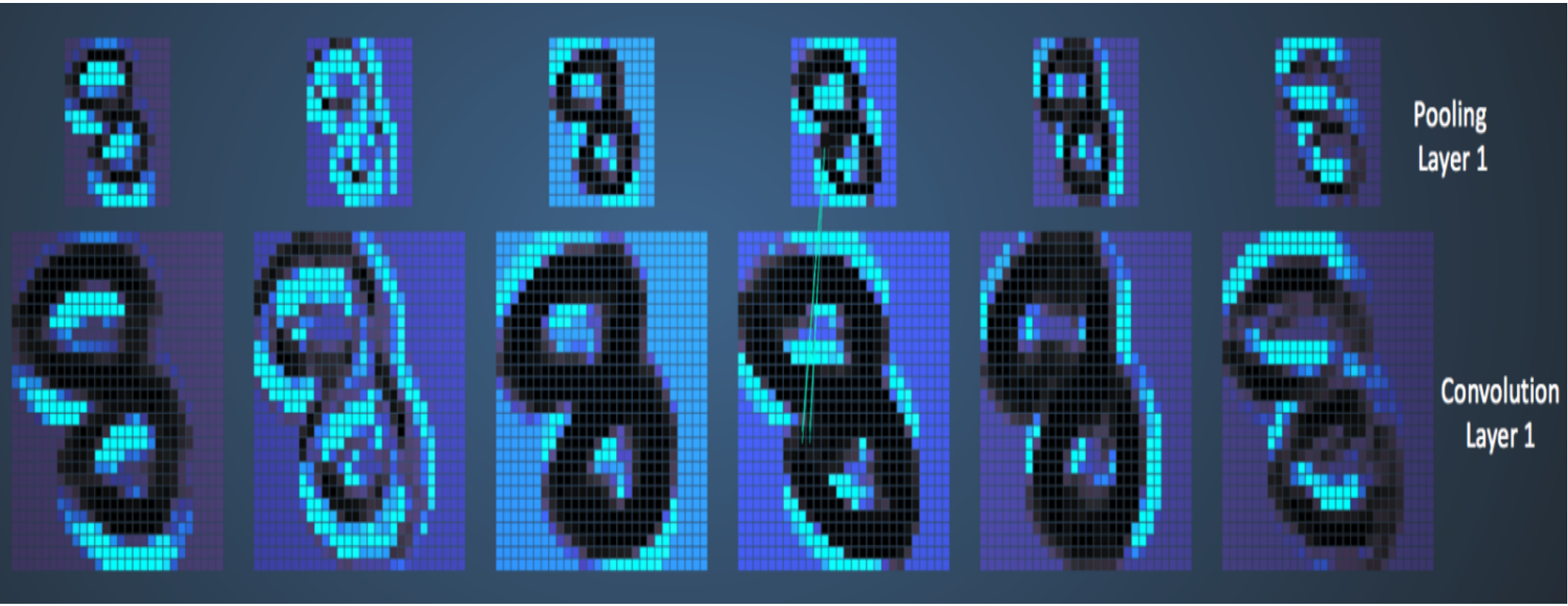
Convoluted Neural Network:Visualization



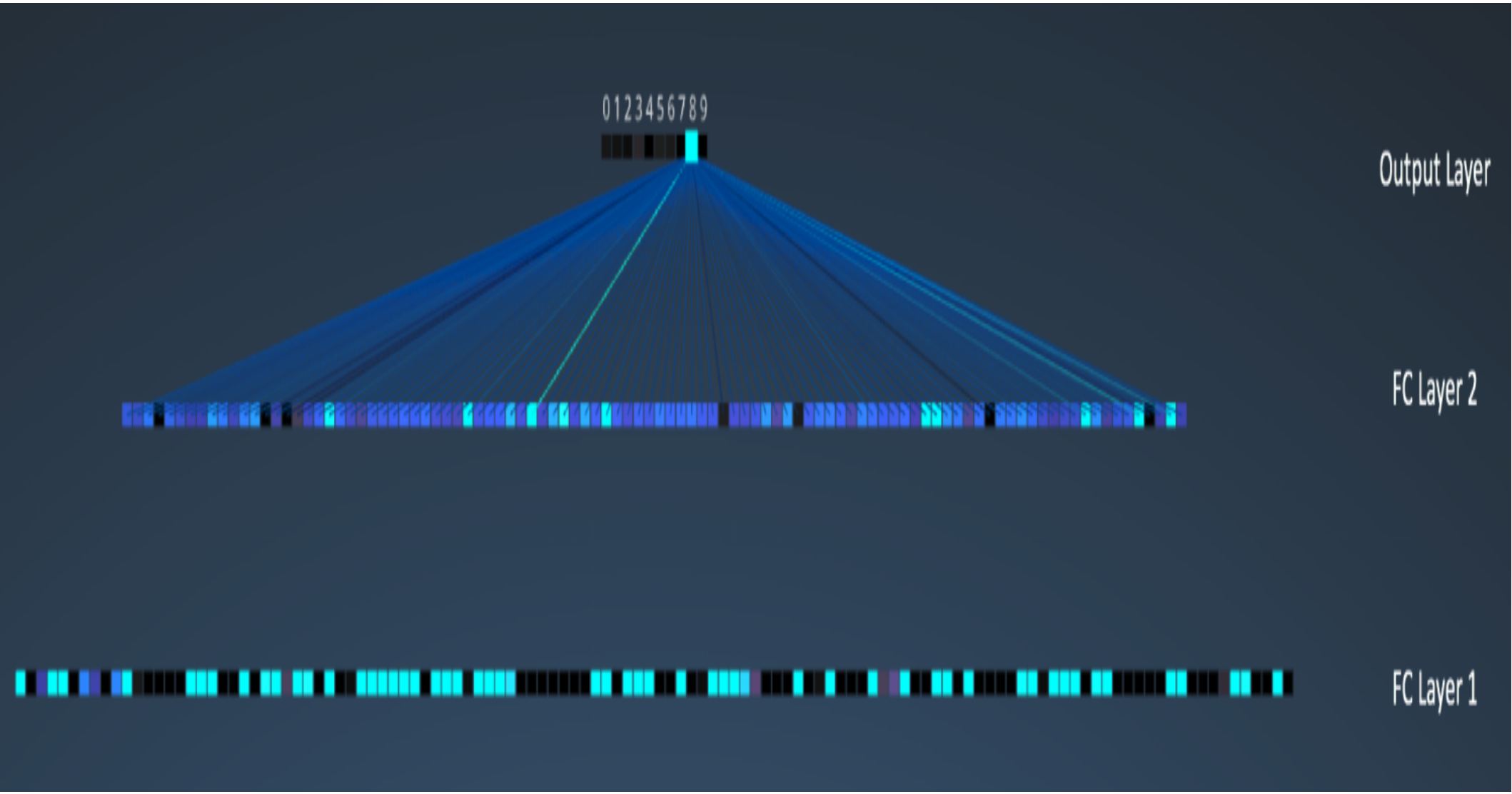
Convoluted Neural Network:Visualization



Convoluted Neural Network:Visualization

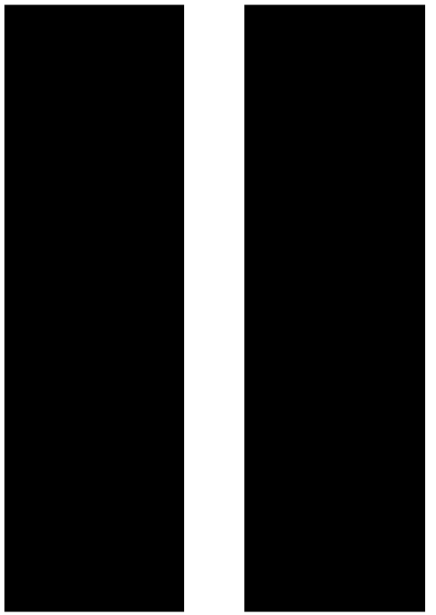


Convoluted Neural Network:Visualization



Convoluted Neural Network:Filter

```
from sklearn.datasets import load_sample_image

def plot_image(image):
    plt.imshow(image, cmap="gray", interpolation="nearest")
    plt.axis("off")

def plot_color_image(image):
    plt.imshow(image.astype(np.uint8),interpolation="nearest")
    plt.axis("off")

fmap = np.zeros(shape=(7, 7, 1, 2), dtype=np.float32)
fmap[:, 3, 0, 0] = 1
fmap[3, :, 0, 1] = 1
fmap[:, :, 0, 0]
plot_image(fmap[:, :, 0, 0])
plt.show()
plot_image(fmap[:, :, 0, 1])
plt.show()
```



- Filter to detect vertical edges

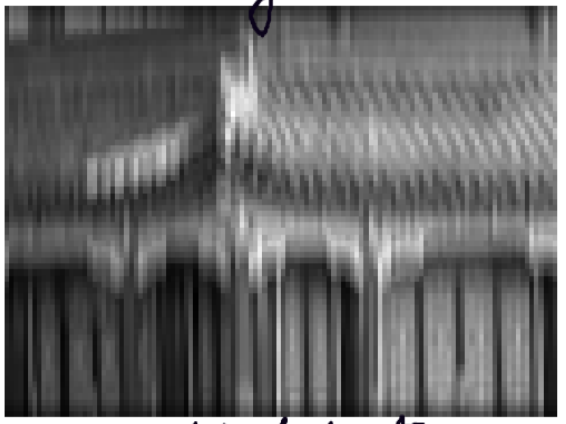
- Filter to detect horizontal edges.

Convoluted Neural Network:Convolution

```
china = load_sample_image("china.jpg")
flower = load_sample_image("flower.jpg")
image = china[150:220, 130:250]
height, width, channels = image.shape
image_grayscale = image.mean(axis=2).astype(np.float32)
images = image_grayscale.reshape(1, height, width, 1)

fmap = np.zeros(shape=(7, 7, 1, 2), dtype=np.float32)
fmap[:, 3, 0, 0] = 1
fmap[3, :, 0, 1] = 1
fmap[:, :, 0, 0]

X = tf.placeholder(tf.float32, shape=(None, height, width, 1))
feature_maps = tf.constant(fmap)
convolution = tf.nn.conv2d(X, feature_maps, strides=[1,1,1,1], padding="SAME", use_cudnn_on_gpu=False)

with tf.Session() as sess:
    output = convolution.eval(feed_dict={X: images})


plot_image(images[0, :, :, 0])
fl.save_fig("china_original", tight_layout=False)
plt.show()

plot_image(output[0, :, :, 0])
fl.save_fig("china_vertical", tight_layout=False)
plt.show()

plot_image(output[0, :, :, 1])
fl.save_fig("china_horizontal", tight_layout=False)
plt.show()
```

- Convolutional layer
- Filters as params.

original



vertical feature map

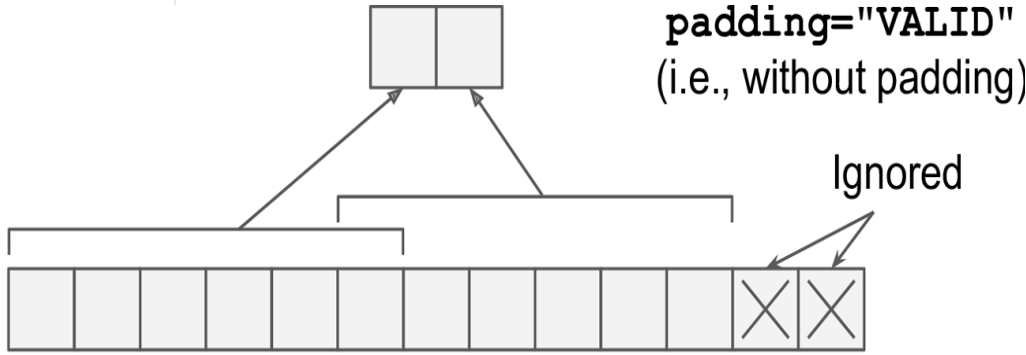
horizontal feature map

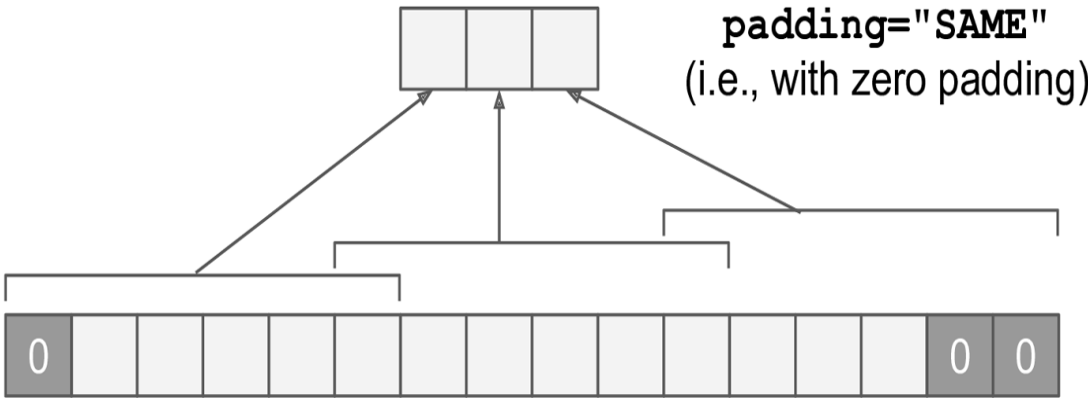
Convoluted Neural Network:Convolution

```
from sklearn.datasets import load_sample_image

def plot_image(image):
    plt.imshow(image, cmap="gray", interpolation="nearest")
    plt.axis("off")

def plot_color_image(image):
    plt.imshow(image.astype(np.uint8),interpolation="nearest")
    plt.axis("off")

china = load_sample_image("china.jpg")
flower = load_sample_image("flower.jpg")
dataset = np.array([china, flower], dtype=np.float32)
batch_size, height, width, channels = dataset.shape

filters = np.zeros(shape=(7, 7, channels, 2), dtype=np.float32)
filters[:, 3, :, 0] = 1  # vertical line
filters[3, :, :, 1] = 1  # horizontal line

X = tf.placeholder(tf.float32, shape=(None, height, width, channels))
convolution = tf.nn.conv2d(X, filters, strides=[1,2,2,1], padding="SAME")

with tf.Session() as sess:
    output = sess.run(convolution, feed_dict={X: dataset})

for image_index in (0, 1):
    for feature_map_index in (0, 1):#every image second feature map
        plot_image(output[image_index, :, :, feature_map_index])
        plt.show()
```

Convoluted Neural Network:Convolution:Padding

```
def plot_image(image):
    plt.imshow(image, cmap="gray", interpolation="nearest")
    plt.axis("off")

def plot_color_image(image):
    plt.imshow(image.astype(np.uint8),interpolation="nearest")
    plt.axis("off")

tf.reset_default_graph()

filter_primes = np.array([2., 3., 5., 7., 11., 13.], dtype=np.float32)
x = tf.constant(np.arange(1, 13+1, dtype=np.float32).reshape([1, 1, 13, 1]))
filters = tf.constant(filter_primes.reshape(1, 6, 1, 1))

valid_conv = tf.nn.conv2d(x, filters, strides=[1, 1, 5, 1], padding='VALID')
same_conv = tf.nn.conv2d(x, filters, strides=[1, 1, 5, 1], padding='SAME')

with tf.Session() as sess:
    print("VALID:\n", valid_conv.eval())
    print("SAME:\n", same_conv.eval())

print(filter_primes)
print("VALID:")
print(np.array([1,2,3,4,5,6]).T.dot(filter_primes))
print(np.array([6,7,8,9,10,11]).T.dot(filter_primes))
print("SAME:")
print(np.array([0,1,2,3,4,5]).T.dot(filter_primes))
print(np.array([5,6,7,8,9,10]).T.dot(filter_primes))
print(np.array([10,11,12,13,0,0]).T.dot(filter_primes))
```





Convoluted Neural Network:Convolution:Padding

```
def plot_image(image):
    plt.imshow(image, cmap="gray", interpolation="nearest")
    plt.axis("off")

def plot_color_image(image):
    plt.imshow(image.astype(np.uint8),interpolation="nearest")
    plt.axis("off")

tf.reset_default_graph()

china = load_sample_image("china.jpg")
flower = load_sample_image("flower.jpg")
dataset = np.array([china, flower], dtype=np.float32)
batch_size, height, width, channels = dataset.shape

batch_size, height, width, channels = dataset.shape

filters = np.zeros(shape=(7, 7, channels, 2), dtype=np.float32)
filters[:, 3, :, 0] = 1  # vertical line
filters[3, :, :, 1] = 1  # horizontal line

X = tf.placeholder(tf.float32, shape=(None, height, width, channels))
max_pool = tf.nn.max_pool(X, ksize=[1, 2, 2, 1], strides=[1,2,2,1], padding="VALID")

with tf.Session() as sess:
    output = sess.run(max_pool, feed_dict={X: dataset})

plot_color_image(dataset[0])
fl.save_fig("china_original")
plt.show()

plot_color_image(output[0])
fl.save_fig("china_max_pool")
plt.show()
```

original

pooled.

Convoluted Neural Network:Tensorflow:CNN-1

```
graph = tf.Graph()
with graph.as_default():
    with tf.name_scope("inputs"):
        X = tf.placeholder(tf.float32, shape=[None, n_inputs], name="X")
        X_reshaped = tf.reshape(X, shape=[-1, height, width, channels])
        y = tf.placeholder(tf.int32, shape=[None], name="y")
        is_training = tf.placeholder_with_default(False, shape=[], name='is_training')
    conv1 = tf.layers.conv2d(X_reshaped, filters=conv1_fmaps, kernel_size=conv1_ksize, strides=conv1_stride,
                                                padding=conv1_pad, activation=tf.nn.relu, name="conv1")
    conv2 = tf.layers.conv2d(conv1, filters=conv2_fmaps, kernel_size=conv2_ksize, strides=conv2_stride,
                                                padding=conv2_pad, activation=tf.nn.relu, name="conv2")
    with tf.name_scope("pool3"):
        pool3 = tf.nn.max_pool(conv2, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding="VALID")
        pool3_flat = tf.reshape(pool3, shape=[-1, pool3_fmaps * 14 * 14])
        pool3_flat_drop = tf.layers.dropout(pool3_flat, conv2_dropout_rate, training=is_training)
    with tf.name_scope("fc1"):
        fc1 = tf.layers.dense(pool3_flat_drop, n_fc1, activation=tf.nn.relu, name="fc1")
        fc1_drop = tf.layers.dropout(fc1, fc1_dropout_rate, training=is_training)

    with tf.name_scope("output"):
        logits = tf.layers.dense(fc1, n_outputs, name="output")
        Y_proba = tf.nn.softmax(logits, name="Y_proba")

    with tf.name_scope("train"):
        xentropy = tf.nn.sparse_softmax_cross_entropy_with_logits(logits=logits, labels=y)
        loss = tf.reduce_mean(xentropy)
        optimizer = tf.train.AdamOptimizer()
        training_op = optimizer.minimize(loss)

    with tf.name_scope("eval"):
        correct = tf.nn.in_top_k(logits, y, 1)
        accuracy = tf.reduce_mean(tf.cast(correct, tf.float32))

    with tf.name_scope("init_and_save"):
        init = tf.global_variables_initializer()
        saver = tf.train.Saver()

mse_summary = tf.summary.scalar('ACCURACY', accuracy)
```

Convoluted Neural Network:Tensorflow:CNN-2

```
mse_summary = tf.summary.scalar('ACCURACY', accuracy)
summary_writer = tf.summary.FileWriter(logdir, tf.get_default_graph())

from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("/tmp/data/")

n_epochs = 25
batch_size = 50

best_acc_val = 0
check_interval = 100
checks_since_last_progress = 0
max_checks_without_progress = 100
best_model_params = None

with tf.Session(graph=graph) as sess:
    init.run()
    for epoch in range(n_epochs):
        for iteration in range(mnist.train.num_examples // batch_size):
            X_batch, y_batch = mnist.train.next_batch(batch_size)
            sess.run(training_op, feed_dict={X: X_batch, y: y_batch, is_training: True})
            if iteration % check_interval == 0:
                acc_val = accuracy.eval(feed_dict={X: mnist.test.images[:2000], y: mnist.test.labels[:2000]})
                if acc_val > best_acc_val:
                    best_acc_val = acc_val
                    checks_since_last_progress = 0
                    best_model_params = get_model_params()
                else:
                    checks_since_last_progress += 1
        acc_train = accuracy.eval(feed_dict={X: X_batch, y: y_batch})
        acc_test = accuracy.eval(feed_dict={X: mnist.test.images[2000:], y: mnist.test.labels[2000:]})
        print(epoch, "Train accuracy:", acc_train, "Test accuracy:", acc_test, "Best validation accuracy:", best_acc_val)
        if checks_since_last_progress > max_checks_without_progress:
            print("Early stopping!")
            break

    if best_model_params:
        restore_model_params(best_model_params)
```

Convoluted Neural Network:Tensorflow:CNN-3

```
(ml_home) mohit@nomind:~/Work/ArtificialIntelligence$ ./DL/CNN/cnnmnist.py
Extracting /tmp/data/train-images-idx3-ubyte.gz
Extracting /tmp/data/train-labels-idx1-ubyte.gz
Extracting /tmp/data/t10k-images-idx3-ubyte.gz
Extracting /tmp/data/t10k-labels-idx1-ubyte.gz
2017-07-02 06:11:10.925714: W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFl
p CPU computations.
2017-07-02 06:11:10.925742: W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFl
p CPU computations.
2017-07-02 06:11:10.925757: W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFl
PU computations.
2017-07-02 06:11:10.925763: W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFl
CPU computations.
2017-07-02 06:11:10.925778: W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFl
PU computations.
0 Train accuracy: 0.98 Test accuracy: 0.9875 Best validation accuracy: 0.978
1 Train accuracy: 1.0 Test accuracy: 0.989 Best validation accuracy: 0.9835
2 Train accuracy: 1.0 Test accuracy: 0.98925 Best validation accuracy: 0.985
3 Train accuracy: 1.0 Test accuracy: 0.9925 Best validation accuracy: 0.987
4 Train accuracy: 1.0 Test accuracy: 0.991 Best validation accuracy: 0.987
5 Train accuracy: 1.0 Test accuracy: 0.992125 Best validation accuracy: 0.987
6 Train accuracy: 1.0 Test accuracy: 0.992 Best validation accuracy: 0.989
7 Train accuracy: 1.0 Test accuracy: 0.99 Best validation accuracy: 0.989
8 Train accuracy: 1.0 Test accuracy: 0.99225 Best validation accuracy: 0.989
9 Train accuracy: 1.0 Test accuracy: 0.990375 Best validation accuracy: 0.989
10 Train accuracy: 1.0 Test accuracy: 0.991375 Best validation accuracy: 0.989
11 Train accuracy: 1.0 Test accuracy: 0.991625 Best validation accuracy: 0.989
12 Train accuracy: 1.0 Test accuracy: 0.99025 Best validation accuracy: 0.989
13 Train accuracy: 1.0 Test accuracy: 0.99175 Best validation accuracy: 0.989
14 Train accuracy: 1.0 Test accuracy: 0.991375 Best validation accuracy: 0.989
15 Train accuracy: 1.0 Test accuracy: 0.99175 Best validation accuracy: 0.989
16 Train accuracy: 1.0 Test accuracy: 0.991875 Best validation accuracy: 0.989
Early stopping!
Final accuracy on test set: 0.991625
```

Convoluted Neural Network:Tensorflow:Pretrained Models

```
import matplotlib.image as mpimg
test_image = mpimg.imread(os.path.join("input/images","cnn","test_image.png"))[:, :, :channels]
plt.imshow(test_image)
plt.axis("off")
plt.show()

from tensorflow.contrib.slim.nets import inception
import tensorflow.contrib.slim as slim

tf.reset_default_graph()

X = tf.placeholder(tf.float32, shape=[None, height, width, channels], name="X")
with slim.arg_scope(inception.inception_v3_arg_scope()):
    logits, end_points = inception.inception_v3(X, num_classes=1001, is_training=False)
predictions = end_points["Predictions"]
saver = tf.train.Saver()

X_test = test_image.reshape(-1, height, width, channels)

with tf.Session() as sess:
    saver.restore(sess, INCEPTION_V3_CHECKPOINT_PATH)
    predictions_val = predictions.eval(feed_dict={X: X_test})

print("Predicted Class:",class_names[np.argmax(predictions_val[0])])

print(np.argmax(predictions_val, axis=1))

top_5 = np.argpartition(predictions_val[0], -5)[-5:]
top_5 = top_5[np.argsort(predictions_val[0][top_5])]
for i in top_5:
    print("{0}: {1:.2f}%".format(class_names[i], 100*predictions_val[0][i]))
```

```
swing: 0.04%
beer bottle: 0.05%
common newt, Triturus vulgaris: 0.05%
red fox, Vulpes vulpes: 2.36%
hyena, hyaena: 93.85%
```