Machine Learning Applications using Python

Neda Hantehzadeh

Week 4

Outline

- Adaptive learning rates with example code
- An Example of CNN with Visualization (Caffe Example)
- History related to architecture of CNNs
- Convolutional Neural Networks
    1. Activation maps
    2. Max Pooling
    3. ReLu
    4. Filter size
    5. Stride and Padding
- Transfer Learning
- Types of Transfer Learning

Coding:

- Example of CNN using MNIST data (visualize training and testing)
- Example of CNN using CIFAR-10 dataset
- Checkpoints and callbacks

Adaptive Learning Rate

SGD is one of the most stablished and popular optimizers used in Deep Neural Networks

Adopting learning rate can increase accuracy and reduce training time

Two methods for adoptive learning rate are:

1. Time-based learning rate
2. Drop-based learning rate

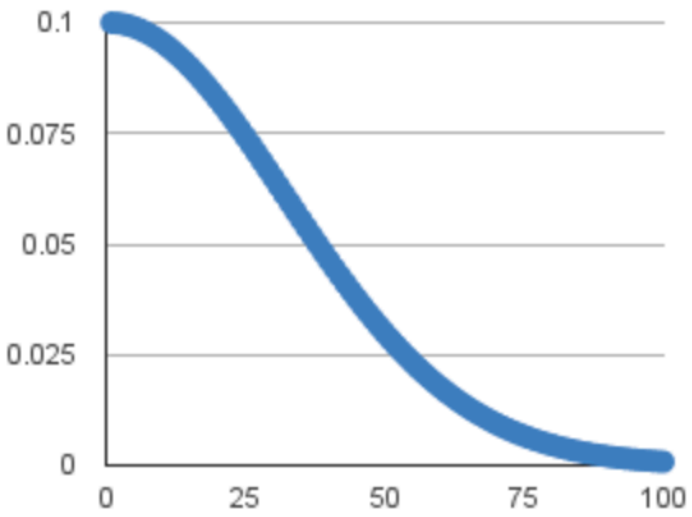
Adaptive Learning Rate

Time-based learning rate decrease

LearningRate = LearningRate * 1/(1 + decay * epoch)

If decay =0:
LearningRate = 0.1 * 1/(1 + 0.0 * 1)
LearningRate = 0.1

Decay = LearningRate / Epochs
Decay = 0.1 / 100
Decay = 0.001



Adaptive Learning Rate

Current epoch number

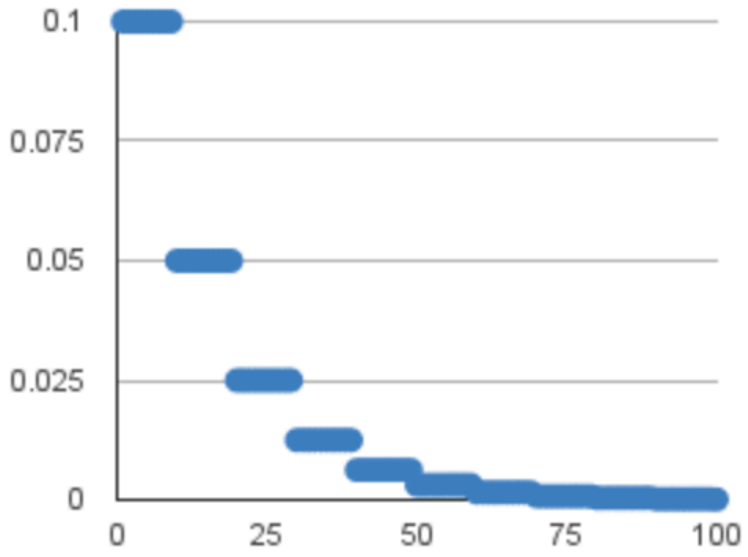
Drop-based Learning rate decrease

LearningRate = InitialLearningRate * DropRate^floor(Epoch / EpochDrop)

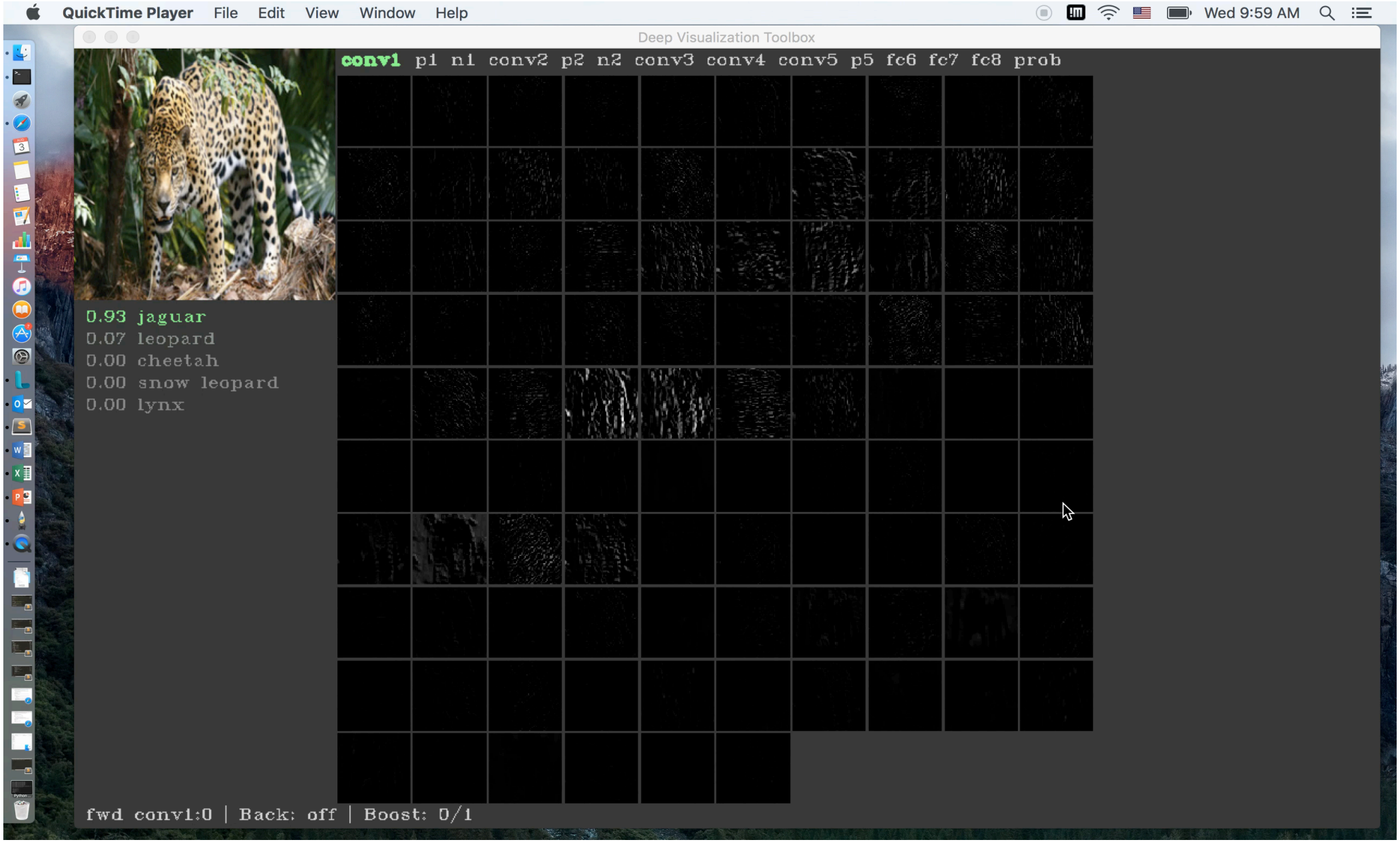
Initial learning rate such as 0.01

How often to change the learning rate such as 10

Amount the learning rate is modified each time it is changed



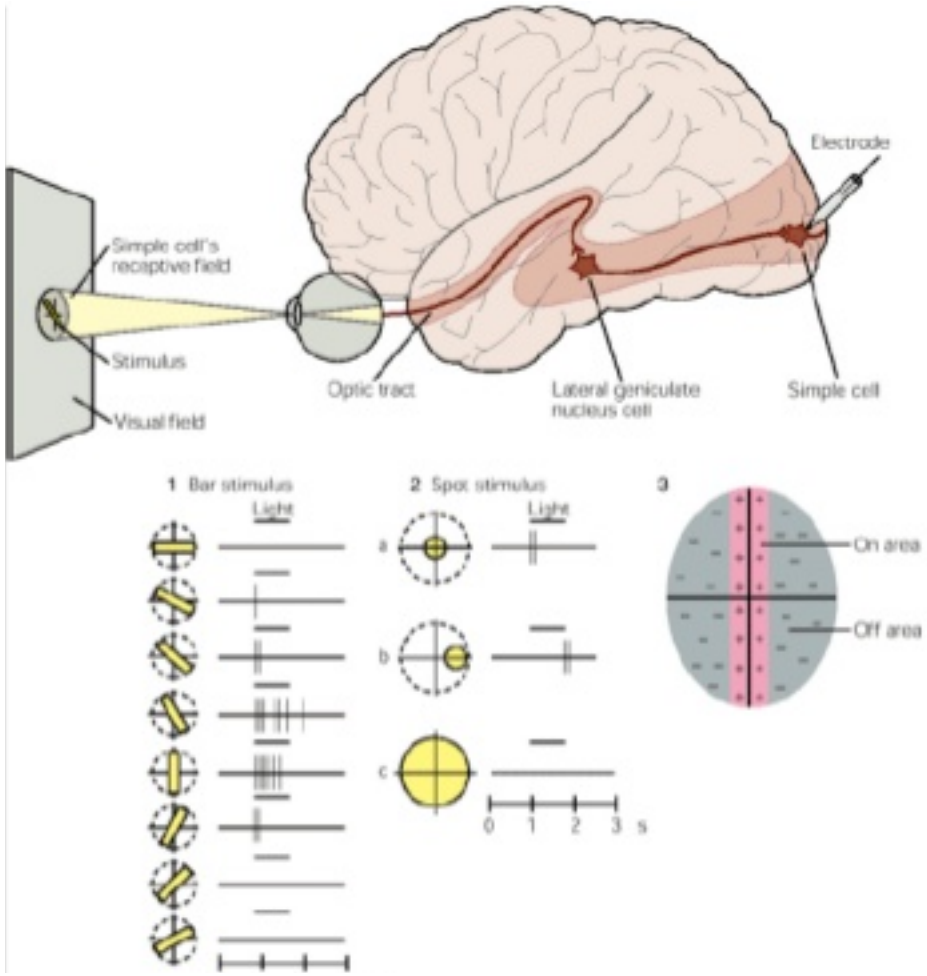
Lets look at example codes for both techniques!



https://github.com/yosinski/deep-visualization-toolbox

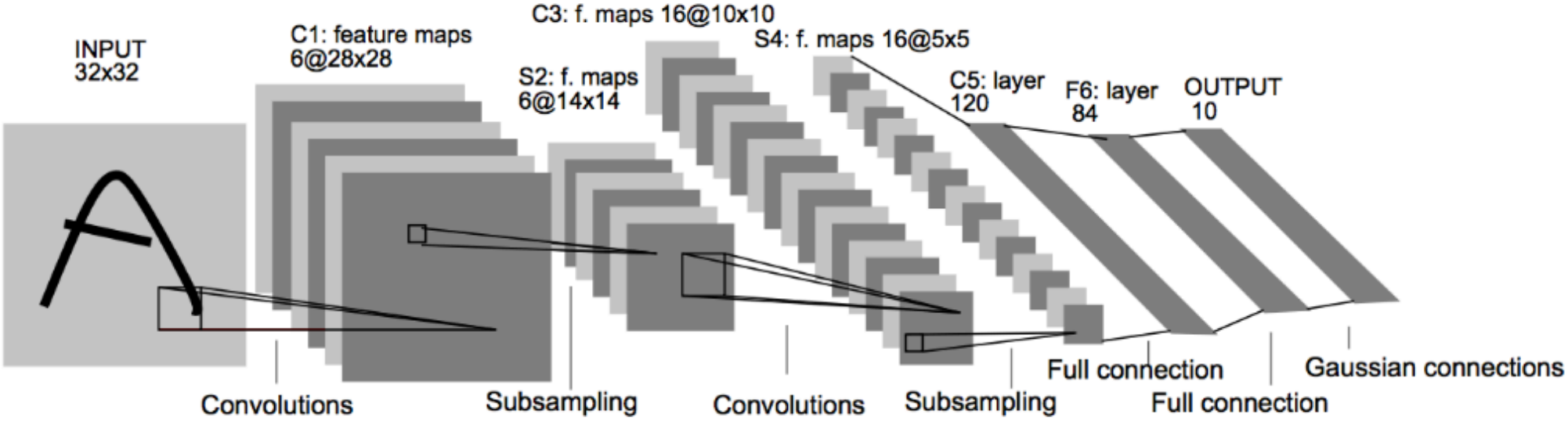
History of CNNs

Hubel and Wiesel Experiment



History of CNNs

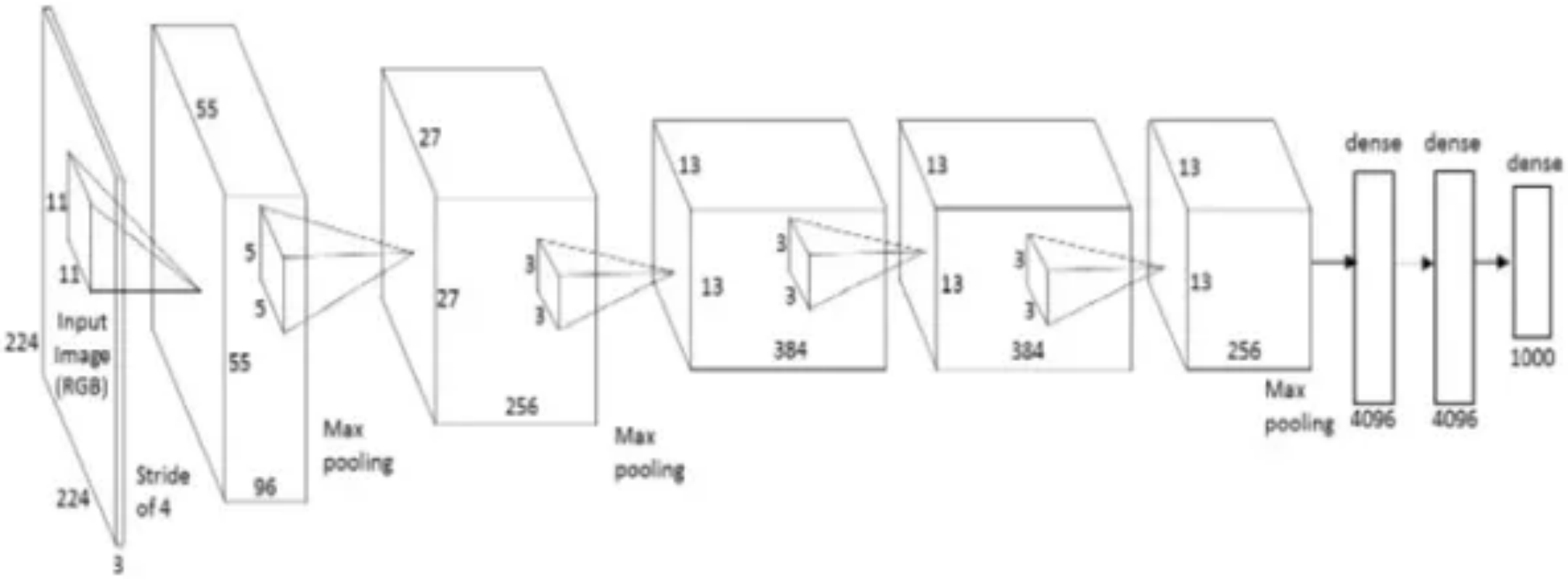
LeNet by Yann LeCune 1998



CNN called LeNet by Yann LeCun (1998)

History of CNNs

AlexNet by Alex Krizhevsy and Ilya Sutskever and Hinton, 2012



CNNs Today

Video Sequence Prediction

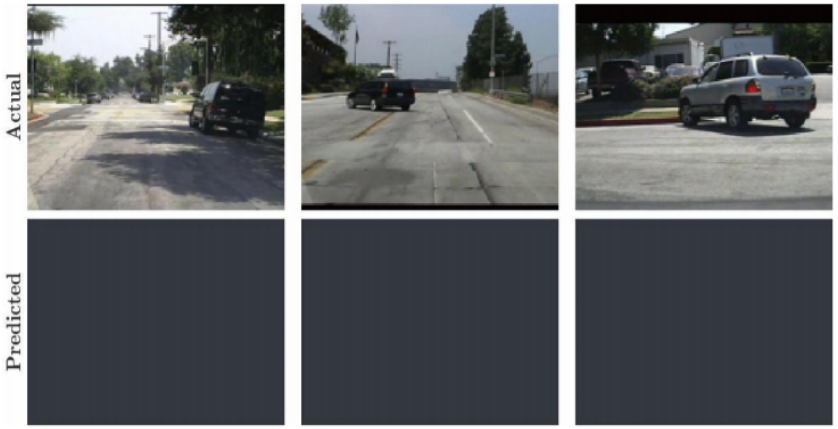


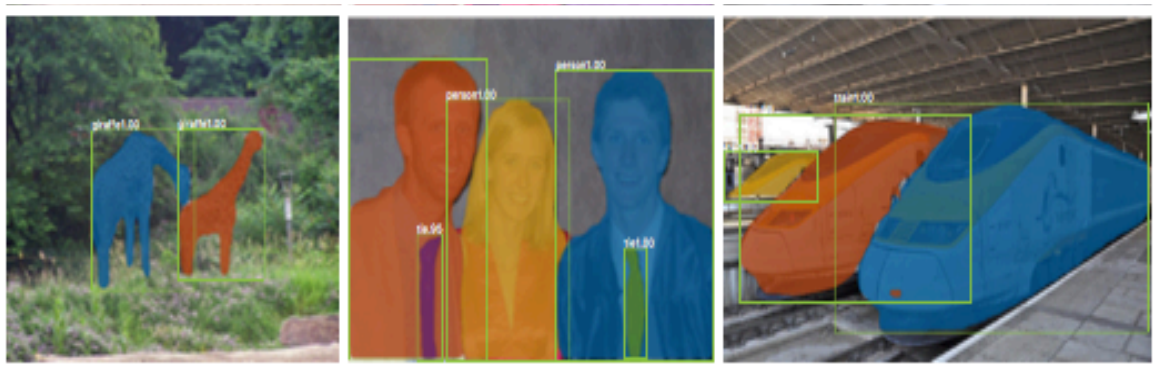
Image Captioning

A close up of a child holding a stuffed animal

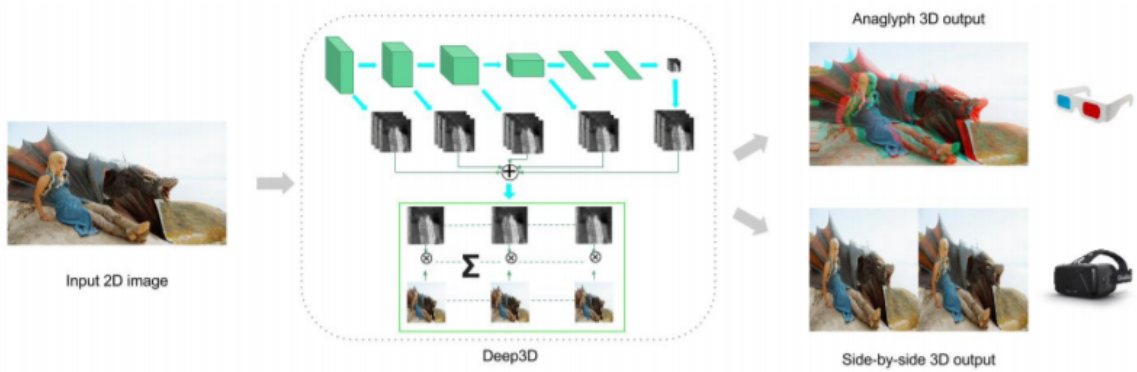
Two pizzas sitting on top of a stove top oven

A man flying through the air while riding a skateboard

Semantic Segmentation

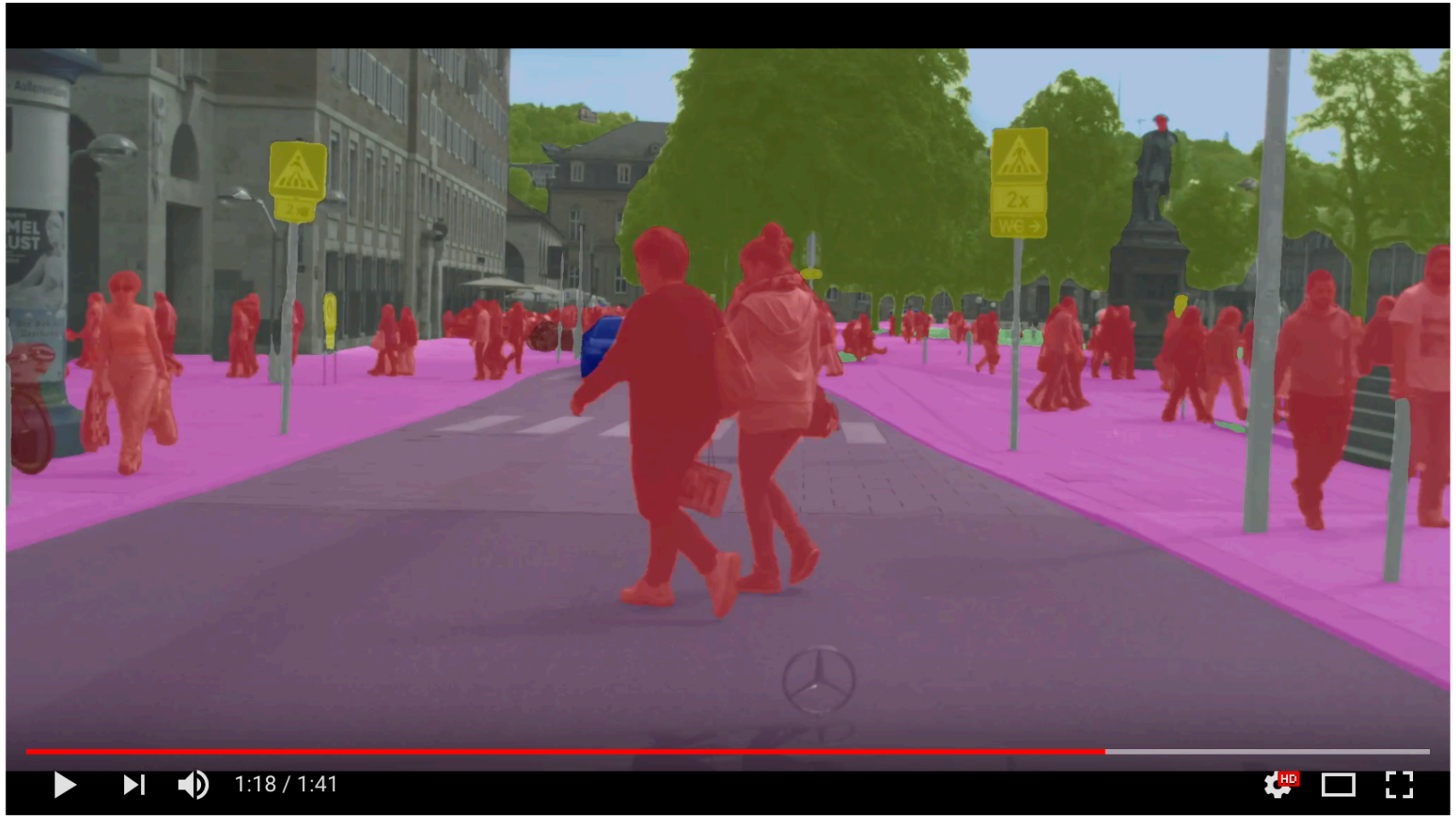


2D to 3D Conversion



CNNs Today

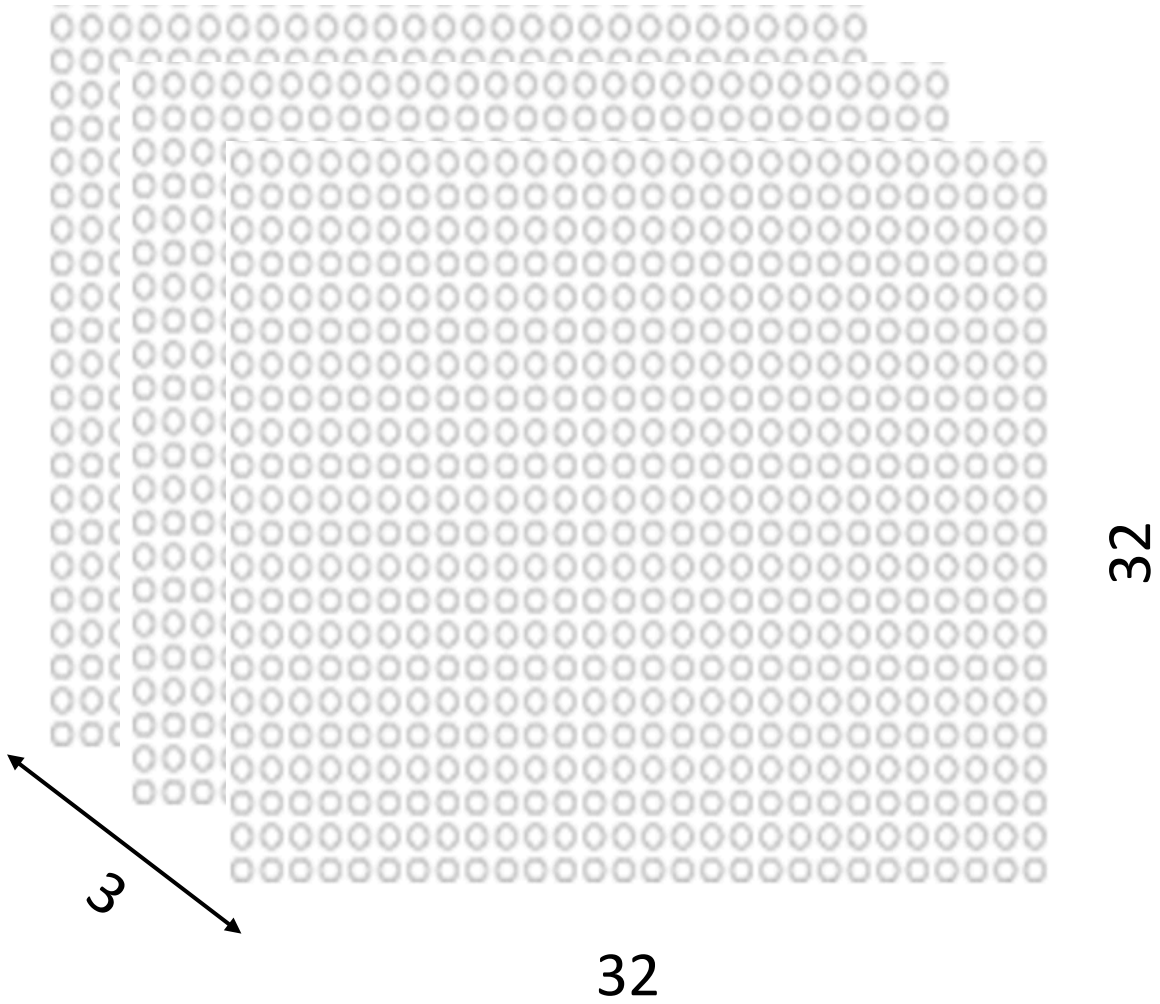
Example of Scene Parsing by PSPNet (Best work in 2017!)



https://www.youtube.com/watch?v=HYghTzmbv6Q

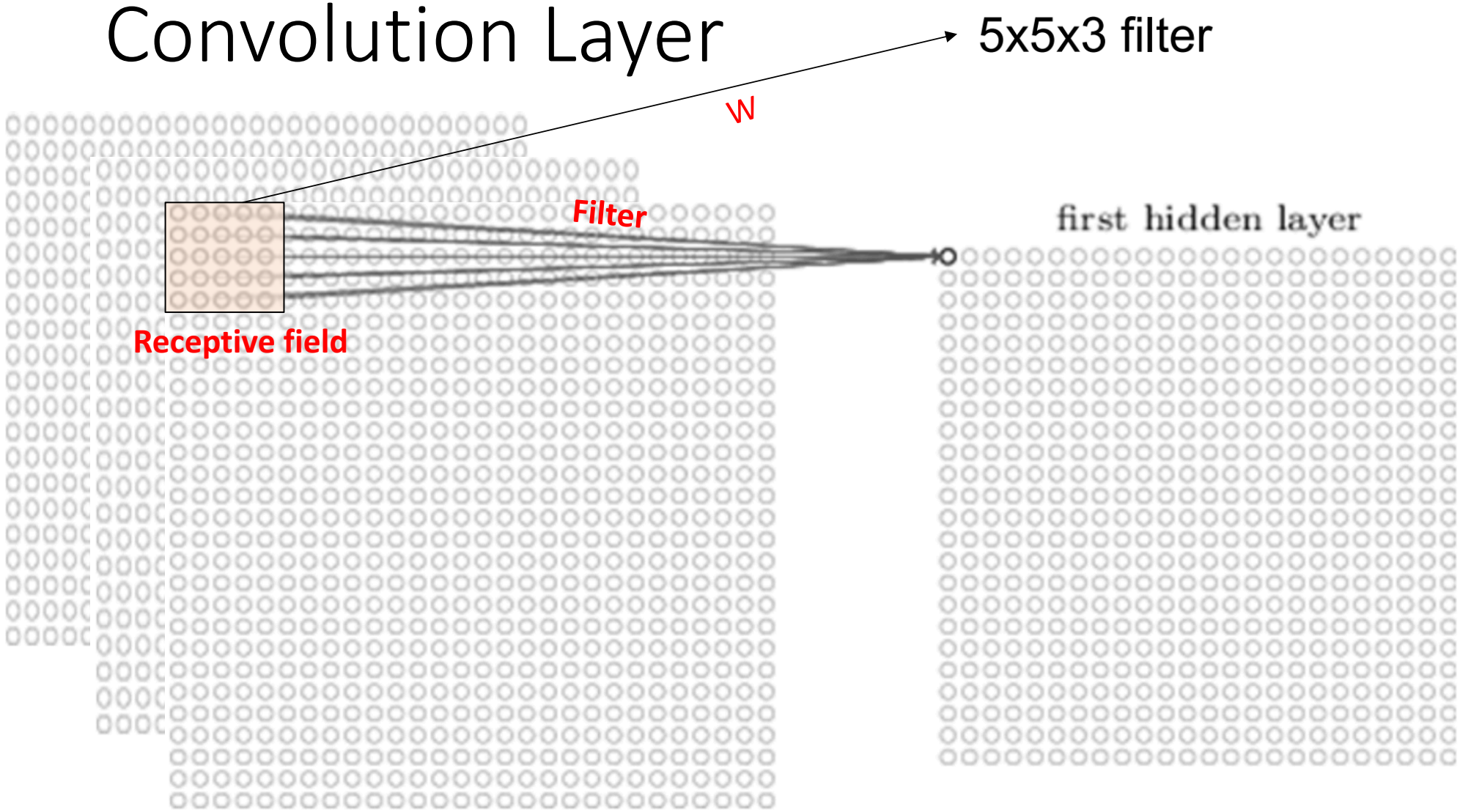
Convolutional Neural Networks

Convolution Layer



32x32x3 image

Convolution Layer

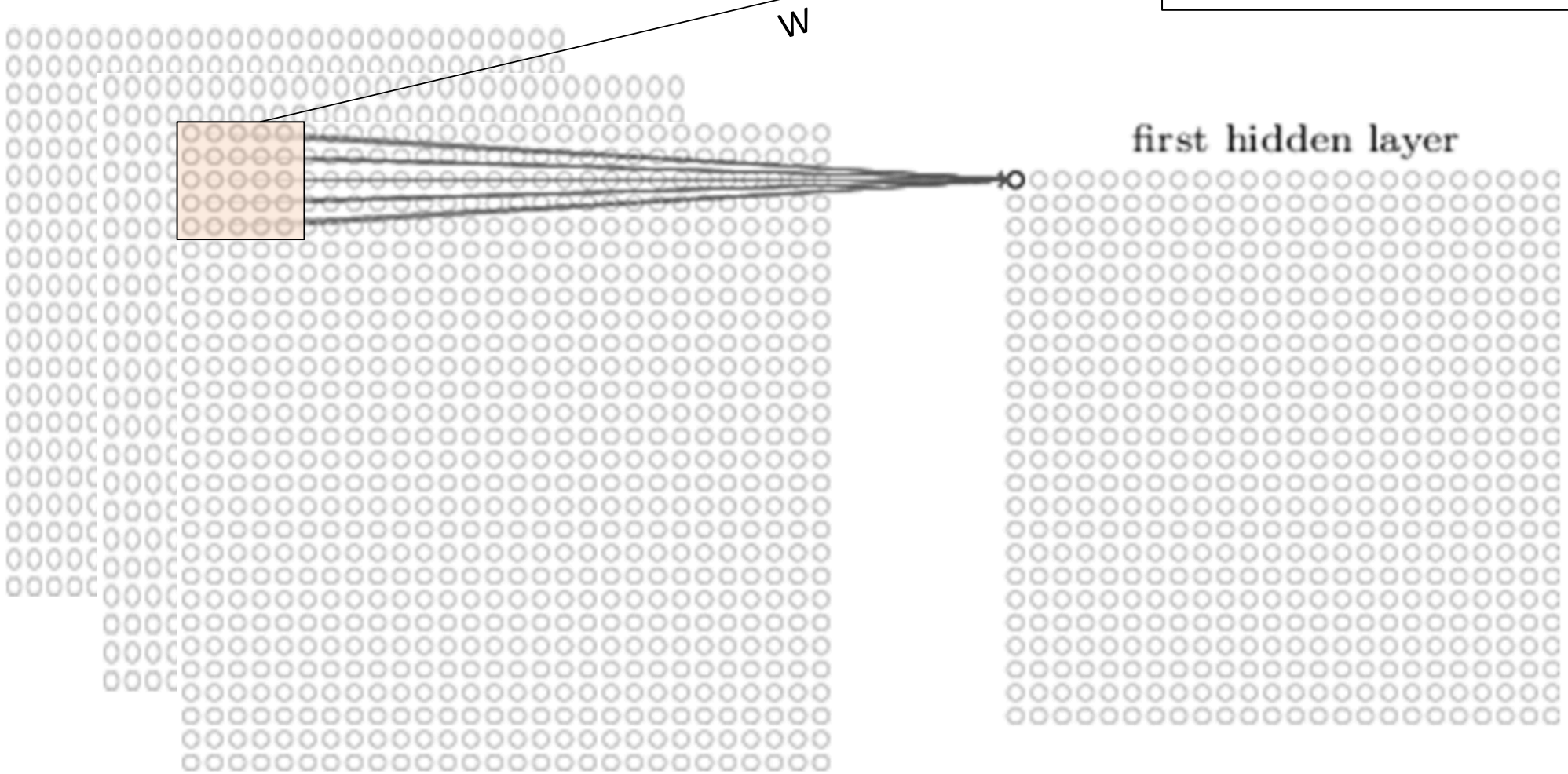


32x32x3 image

Convolution Layer

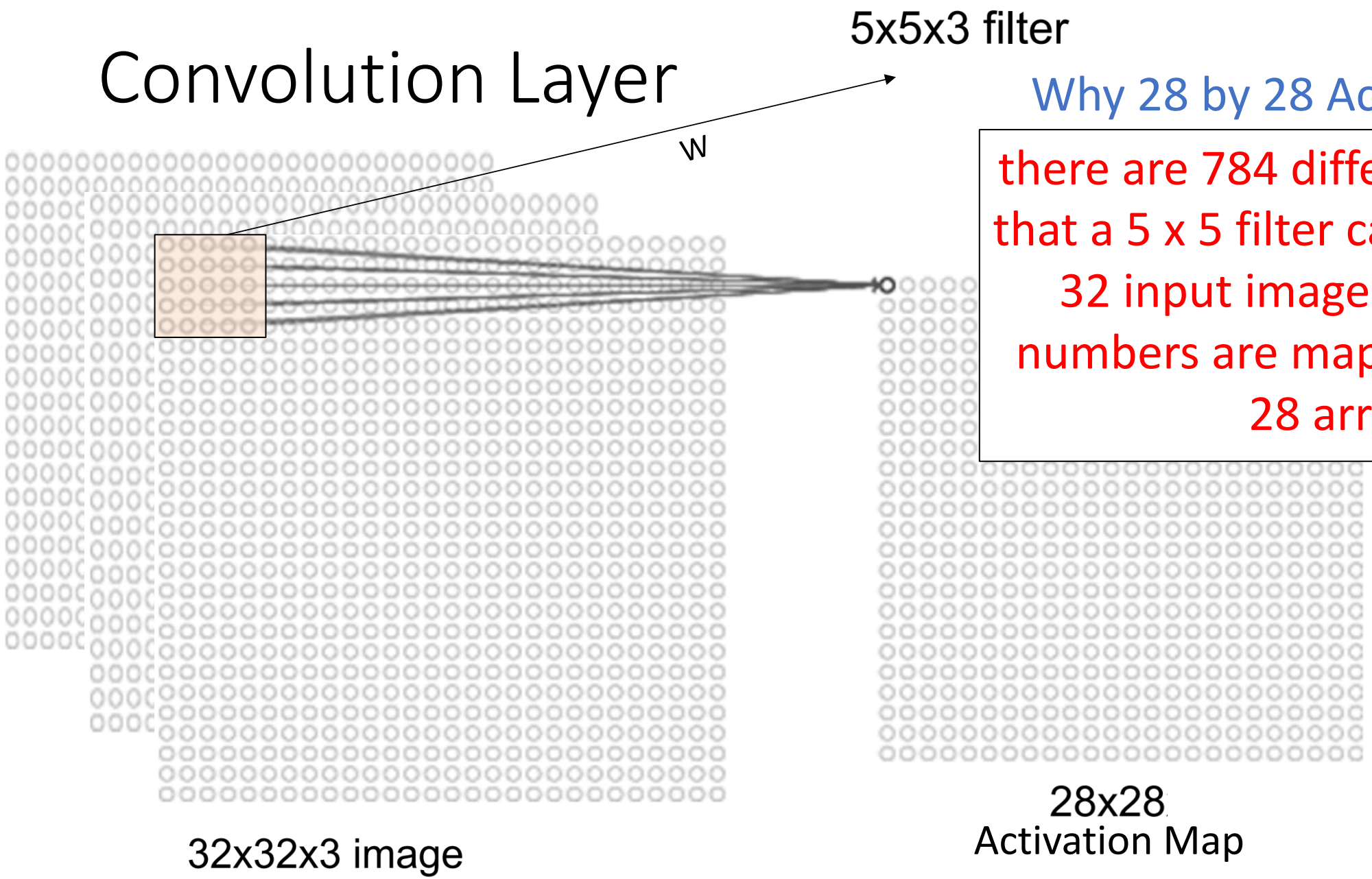
5x5x3 filter

the result of taking a dot product between the filter and a small 5x5x3 chunk of the image (i.e. 5*5*3 = 75-dimensional dot product



32x32x3 image

Convolution Layer



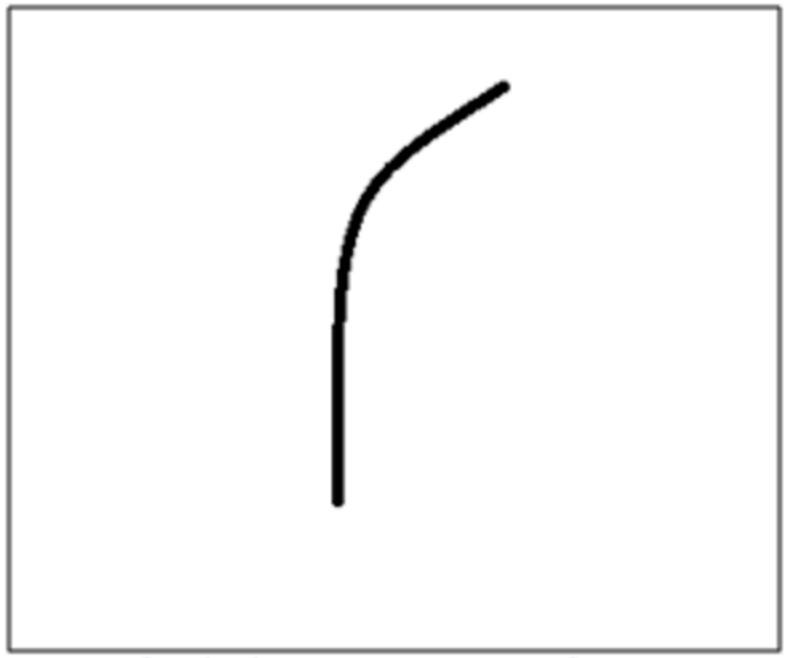
Why 28 by 28 Activation map?

there are 784 different locations that a 5 x 5 filter can fit on a 32 x 32 input image. These 784 numbers are mapped to a 28 x 28 array

Simple Example of The Mathematics

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

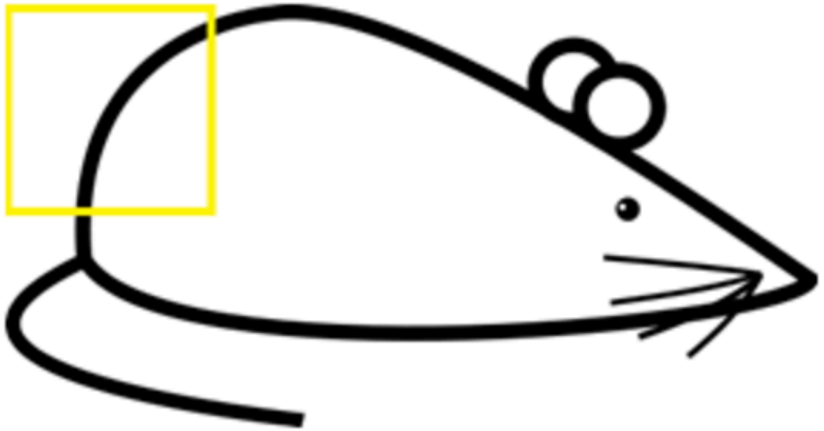
Pixel representation of filter



Visualization of a curve detector filter

https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/

Original image



Visualization of the filter on the image

Visualization of the receptive field

0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

Pixel representation of the receptive field

*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = (50*30)+(50*30)+(50*30)+(20*30)+(50*30) = 6600 (A large number!)

https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/

Visualization of the filter on the image

0	0	0	0	0	0	0
0	40	0	0	0	0	0
40	0	40	0	0	0	0
40	20	0	0	0	0	0
0	50	0	0	0	0	0
0	0	50	0	0	0	0
25	25	0	50	0	0	0

Pixel representation of receptive field

*

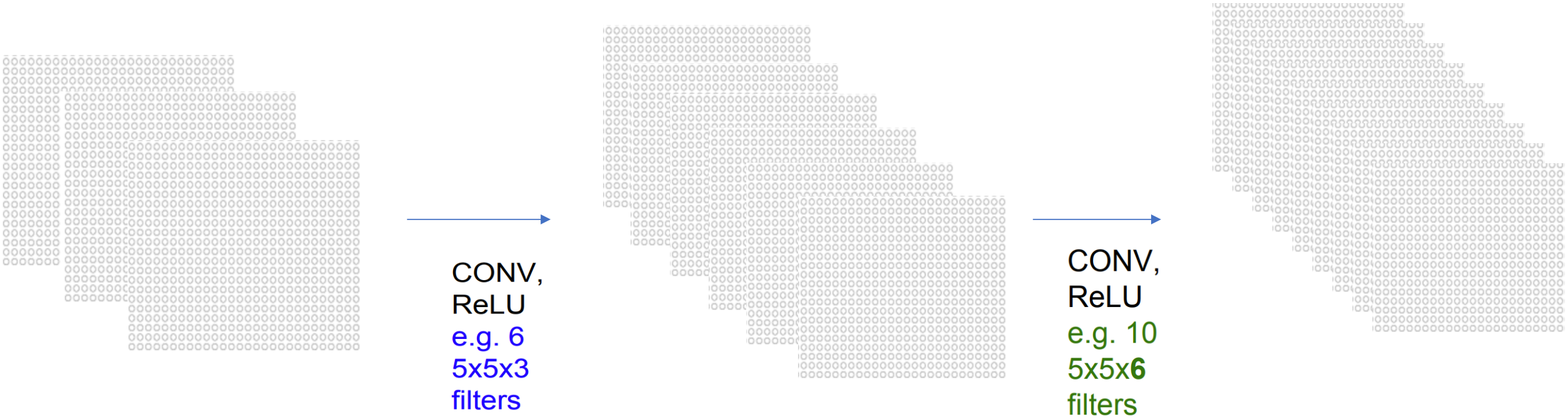
0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = 0

Convolution Layer

Simple question: Why this type of networks are called convolutional neural networks?

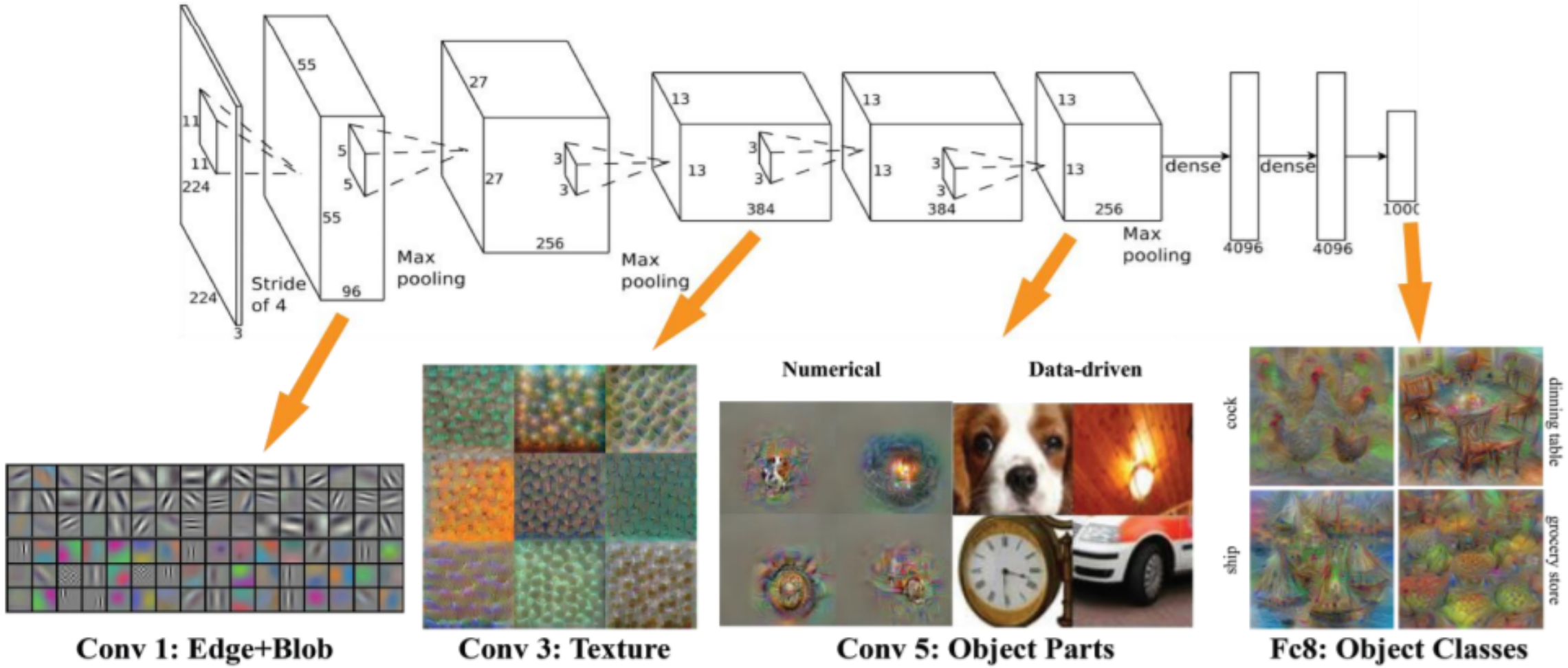


32x32x3 image

28x28x6

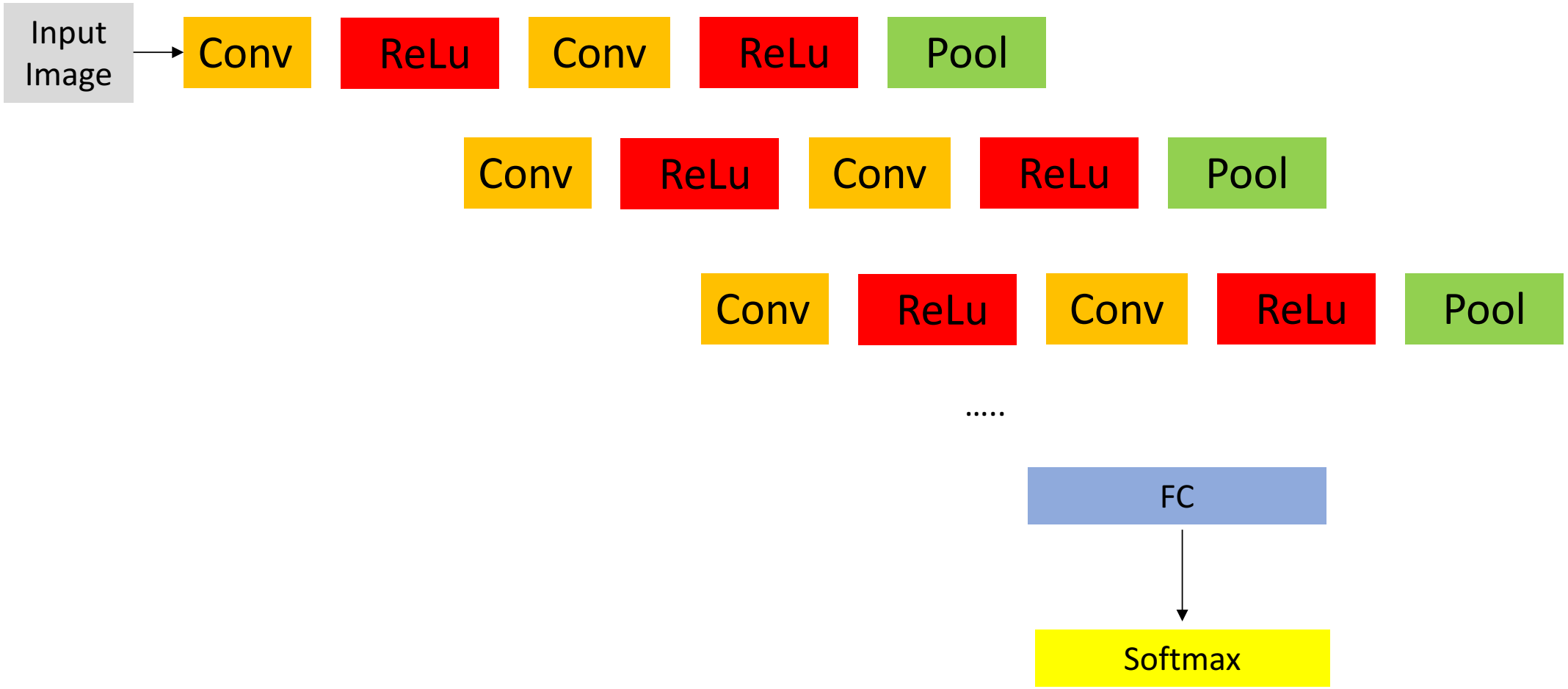
$24 \times 24 \times 10$

Overall Representation CNNs



http://vision03.csail.mit.edu/cnn_art/index.html

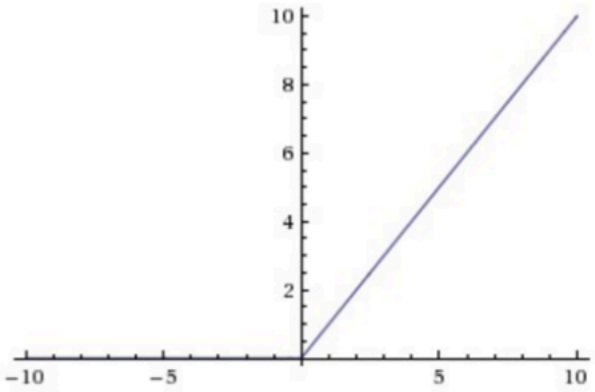
General Architecture of CNNs



Keras example of adding CNN layers

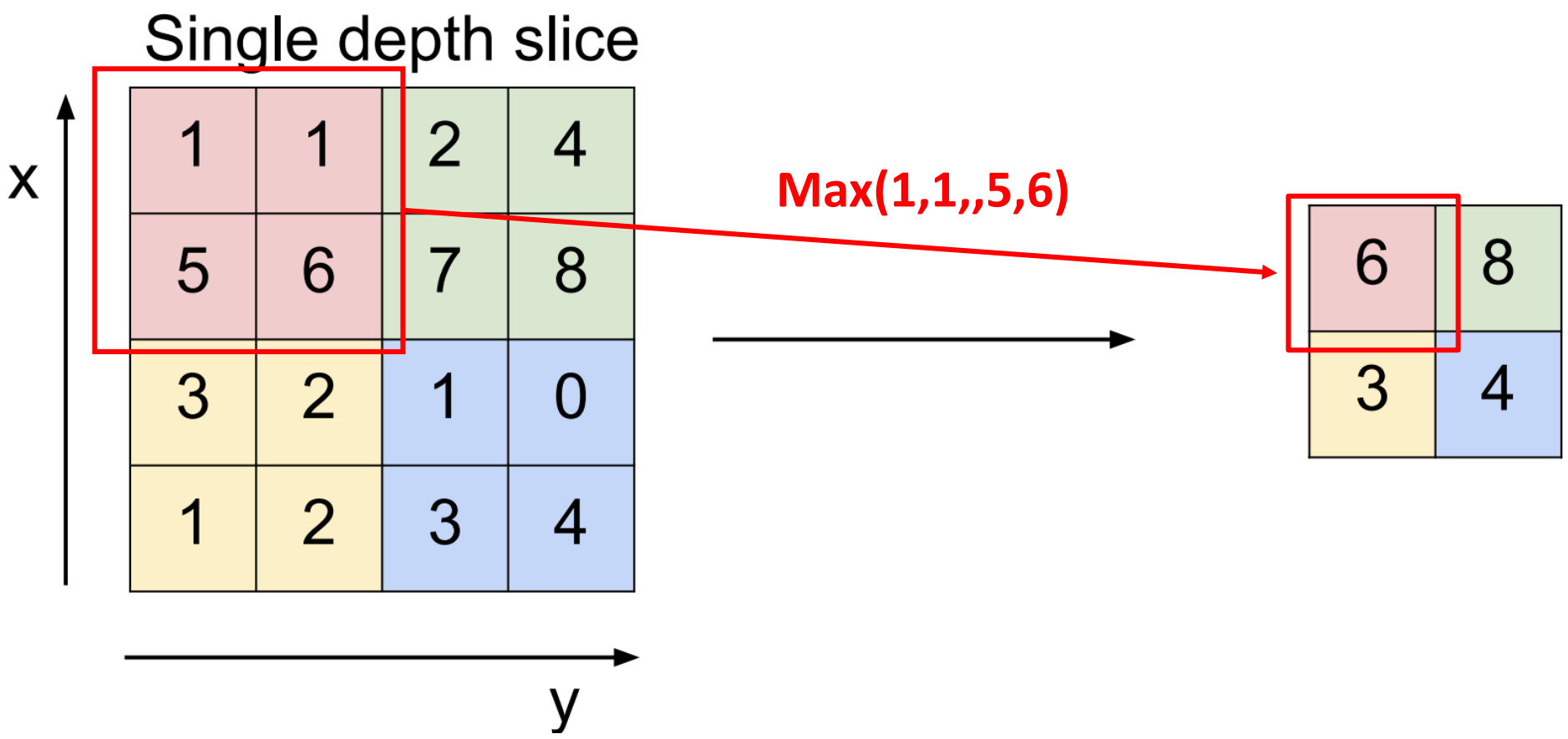
```
model = Sequential()
model.add(Conv2D(32, kernel_size=(5, 5), strides=(1, 1),
                 activation='relu',
                 input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Conv2D(64, (5, 5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(1000, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))
```

ReLu Operation

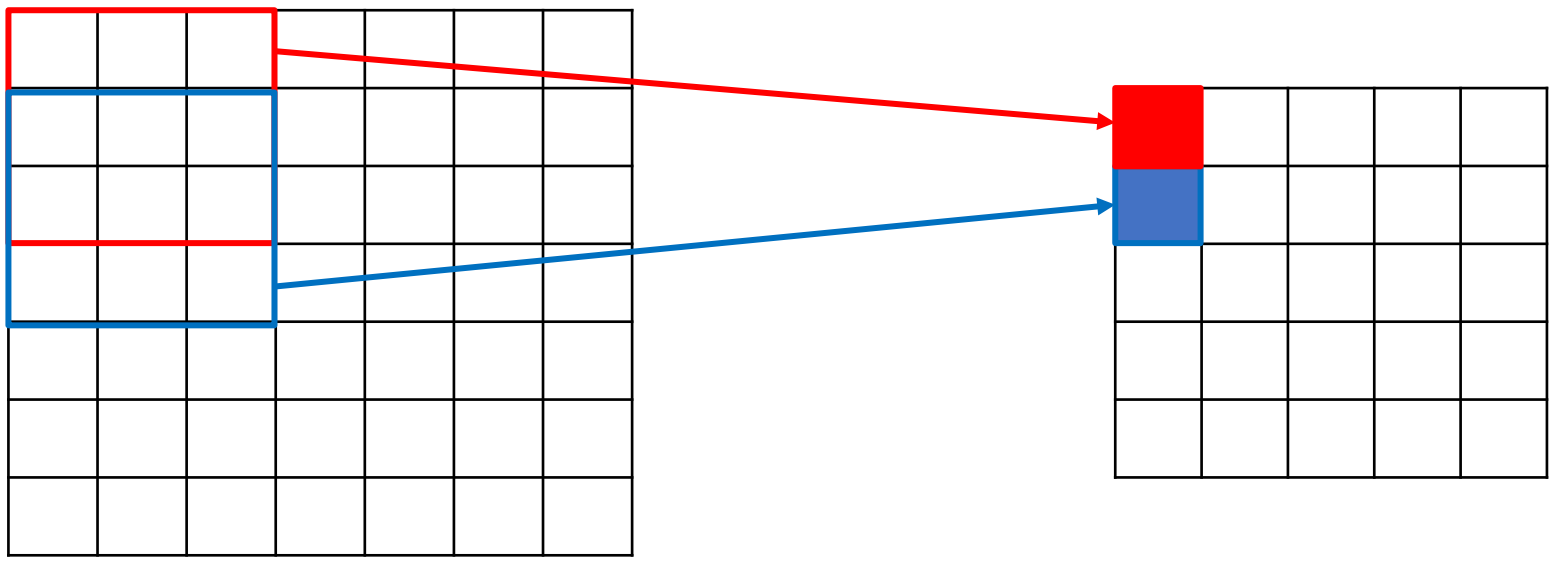


Output = Max(zero, Input)

Max Pooling



Stride and Padding

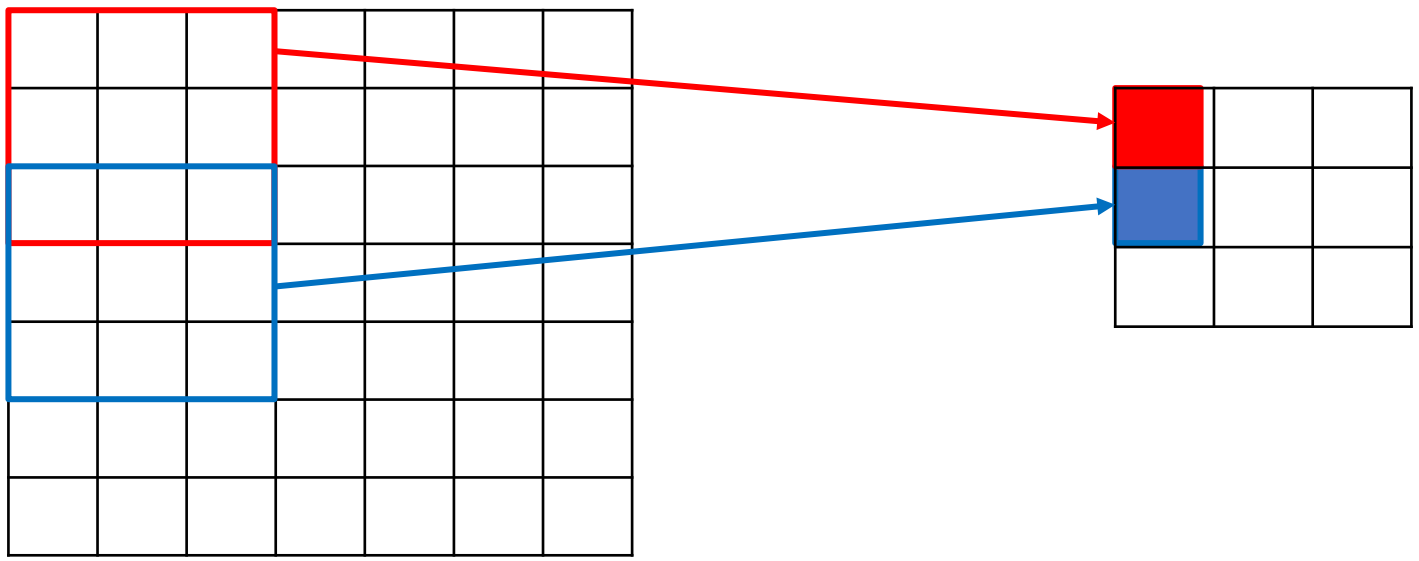


7 × 7 Input

5 × 5 output

Stride = 1

Stride and Padding

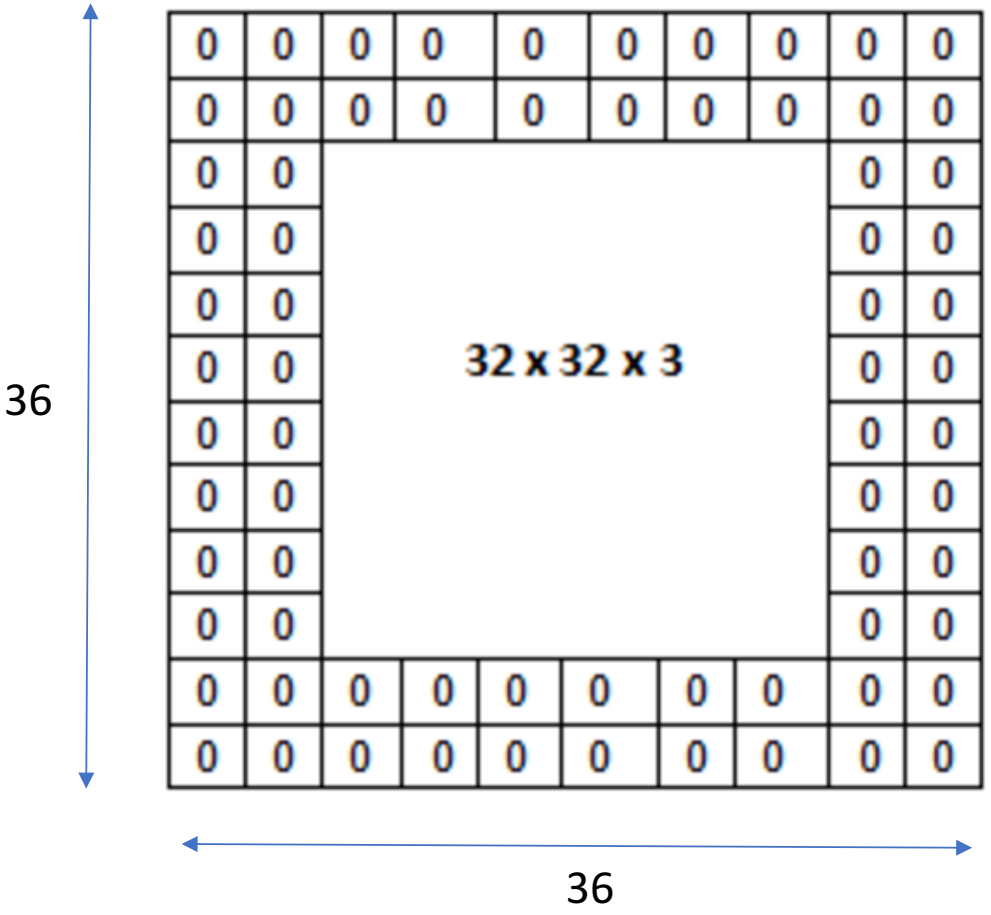


7 × 7 Input

3 × 3 output

Stride = 2

Stride and Padding



$$Zero\ Padding = \frac{(K-1)}{2}$$

K is the filter size

$$O = \frac{(W - K + 2P)}{S} + 1$$

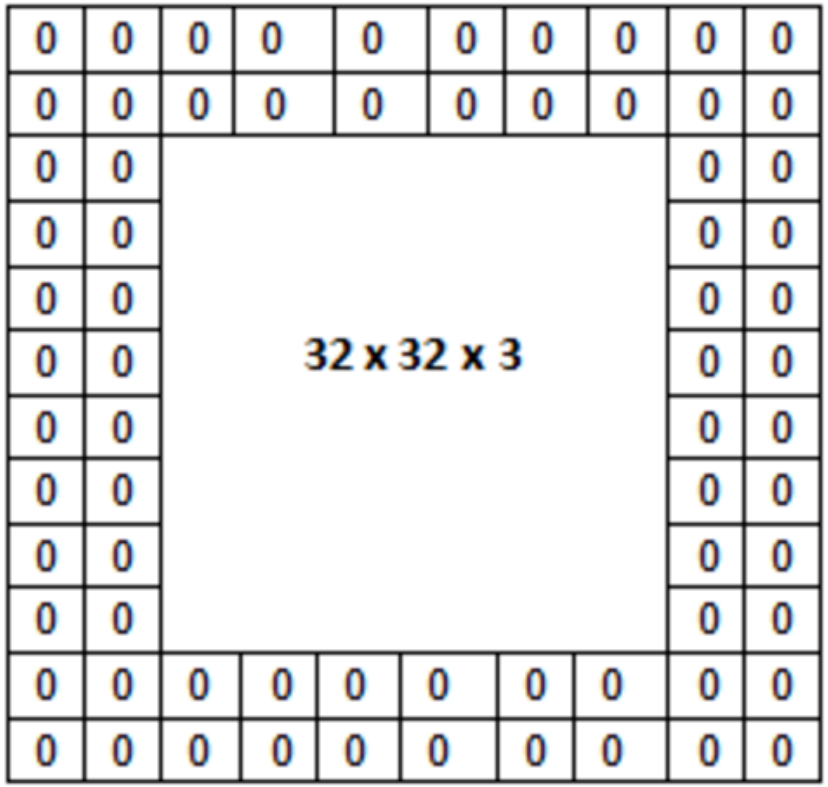
Output size

Input size

Stride

Padding

Stride and Padding

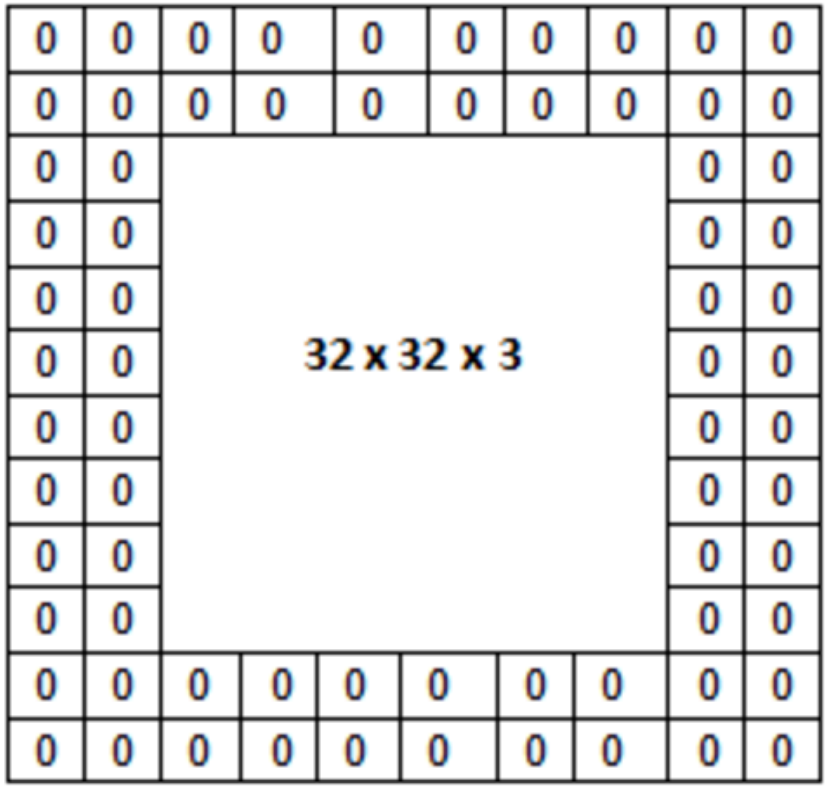


$$Zero\ Padding = \frac{(K-1)}{2}$$

$$O = \frac{(W-K+2P)}{S} + 1$$

What is the Zero Padding with filter size of 3×3?

Stride and Padding



$$Zero\ Padding = \frac{(K-1)}{2}$$

$$O = \frac{(W-K+2P)}{S} + 1$$

What is the Output size with filter size of 3×3, Input of 32 ×32 , padding of 1 and Stride of 1?

Transfer Learning

What is transfer learning?

Transfer learning is a topic in machine learning that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem.

Why transfer learning?

1. Most researchers can't find a dataset with decent examples
2. Very Deep Networks are expensive to train.
3. Finding the hyper parameters for deep learning is a black art with not much theory to guide you

Scenarios of transfer learning

- Use trained networks as feature extractors
- Fine-Tune a Deep Model
- Use Pre-trained models

When and how to do transfer learning?

1. Your dataset is small and similar: Best option- Feature extraction using pre-trained weights from last layers of the ne
2. Your dataset is small and different: Best option- Feature extraction using pre-trained weights from beginning layers of the net
3. Your dataset is large and similar: Best option- Fine-tune the net but freeze the beginning layers
4. Your dataset is large and different: Best option- Fine-tune the whole net

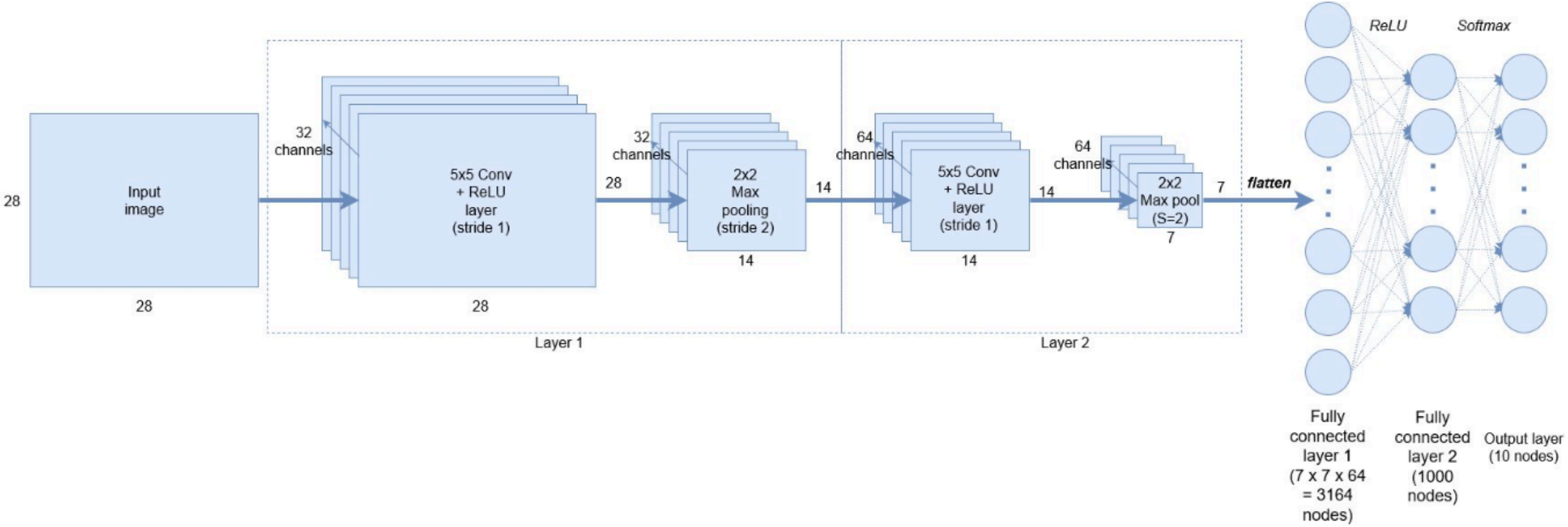
Important concepts for mid-term exam!

MNIST Dataset

- The MNIST problem is a dataset developed by Yann LeCun, Corinna Cortes and Christopher Burges for evaluating machine learning models on the handwritten digit classification problem
- This makes it an excellent dataset for evaluating models, allowing the developer to focus on the machine learning with very little data cleaning or preparation required.
- Each image is a 28 by 28 pixel square (784 pixels total). 60,000 images are used to train a model and a separate set of 10,000 images are used for testing
- 10 digits (0 to 9) or 10 classes to predict

Lets Look at the performance of a MLP model before building a CNN model!

Building a Simple 2 Layer CNN with Keras



Step by step guide to build a simple 2 Layer CNN

1. Load the data into train and test sets

```
# load the MNIST data set, which already splits into train and test sets fo
(x_train, y_train), (x_test, y_test) = mnist.load_data()
print (x_train.shape)
print (y_train.shape)
```

```
# reshape the data into a 4D tensor - (sample_number, x_img_size, y_img_si
# because the MNIST is greyscale, we only have a single channel - RGB colou
x_train = x_train.reshape(x_train.shape[0], 28, 28, 1)
x_test = x_test.reshape(x_test.shape[0], 28, 28, 1)
```

```
# convert the data to the right type
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
```

Step by step guide to build a simple 2 Layer CNN

2. Define the Layers

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(5, 5), strides=(1, 1),
                 activation='relu',
                 input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Conv2D(64, (5, 5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(1000, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adam(),
              metrics=['accuracy'])
```

Step by step guide to build a simple 2 Layer CNN

3. Fit the model

```
model.fit(x_train, y_train,
          batch_size=128,
          epochs=10,
          verbose=1,
          validation_data=(x_test, y_test))
```

Step by step guide to build a simple 2 Layer CNN

4. Evaluate on test data

```
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

Visualize Training History in Keras

Use ‘history callback’ function https://keras.io/callbacks

```
history=model.fit(x_train, y_train,
          batch_size=128,
          epochs=10,
          verbose=1,
          validation_data=(x_test, y_test))
```

```
# summarize history for accuracy
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

Lets run some example code in Python!

How to save your model as checkpoints during training

- You need to use ‘ModelCheckpoint’ and ‘callbacks’

```
from sklearn.pipeline import Pipeline
from keras.callbacks import ModelCheckpoint
```

Lets Look at an example code!