Carnegie Mellon University

Introduction to Deep Learning for Engineers

Spring 2025, Deep Learning for Engineers
Feb 4, 2025, Seventh Session

Amir Barati Farimani
Associate Professor of Mechanical Engineering and Bio-Engineering
Carnegie Mellon University

Story so far on NN:

1. MLP (Perceptron, Non-linear separability, Capacity, Depth vs number of Neurons)

2. Universal Function Approximation

3. Empirical Risk Minimization (Changing Integral to Summation with samples)

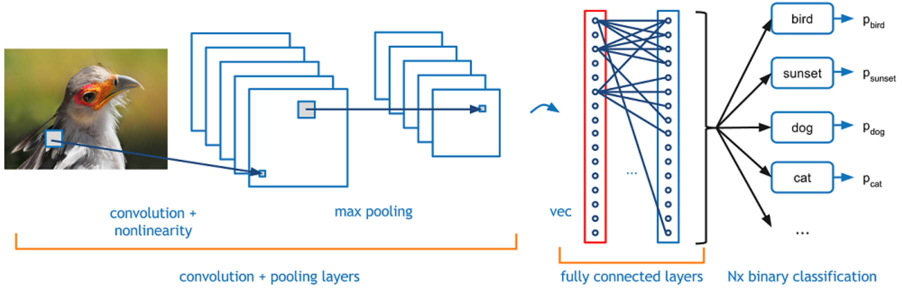
4. Neural Networks Ingredients (inputs, outputs, Loss functions, architectures)

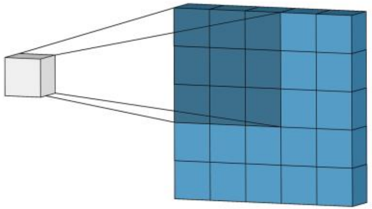
5. Optimization (Gradient Descent**) and Backpropogation** (Chain rules & automatic differentiation)

6. Design of F (x; w): Regularization (weight Initialization, Drop out, Data Augmentation, etc.)

Story so far on CNN:

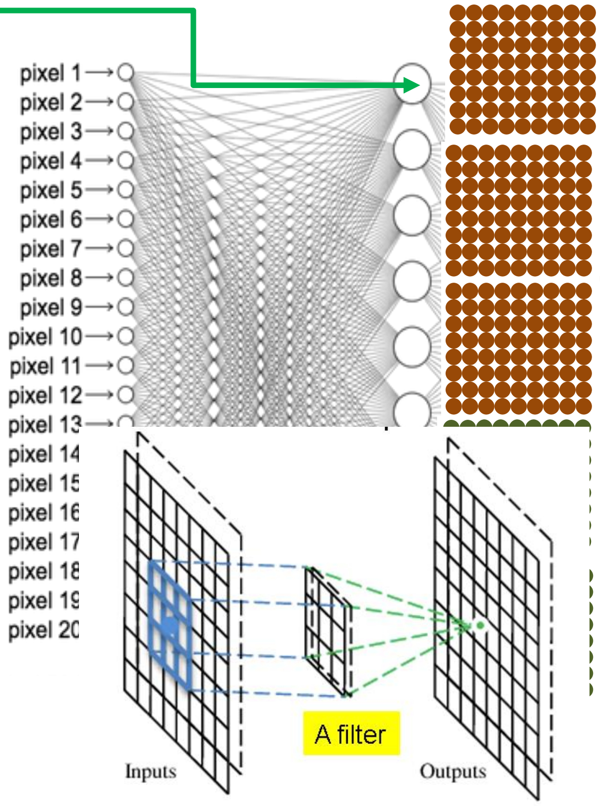
1. Concept of Representation + Learning
2. How to learn robust representation?
3. How to learn spatial, high level features (Swan example)
4. How to build a scanner for feature learning? what should be the properties of this scanner?
5. Can we design the scanners based on the learning tasks?
6. What is the mechanism of learning the scanners? (filters/Kernel)? What is the proof that they are learning representation?



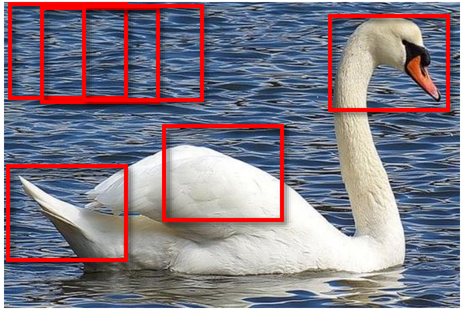


Carnegie Mellon University

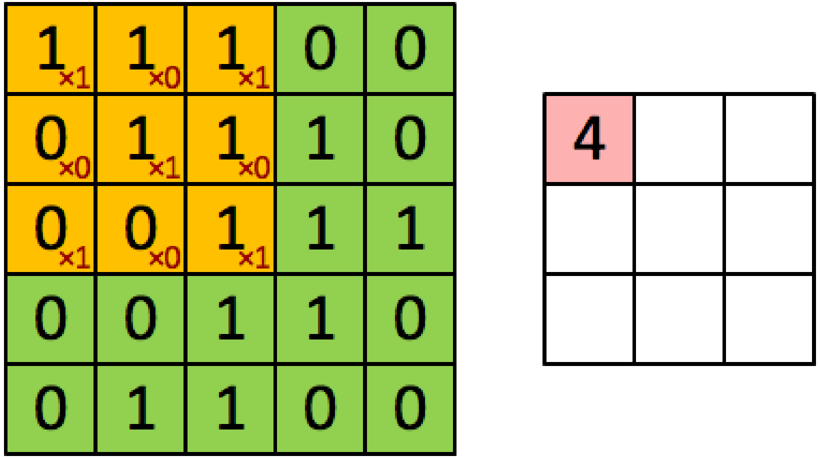
How can we do this?



Instead



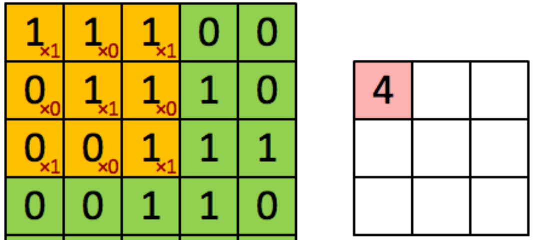
Scanner



Image

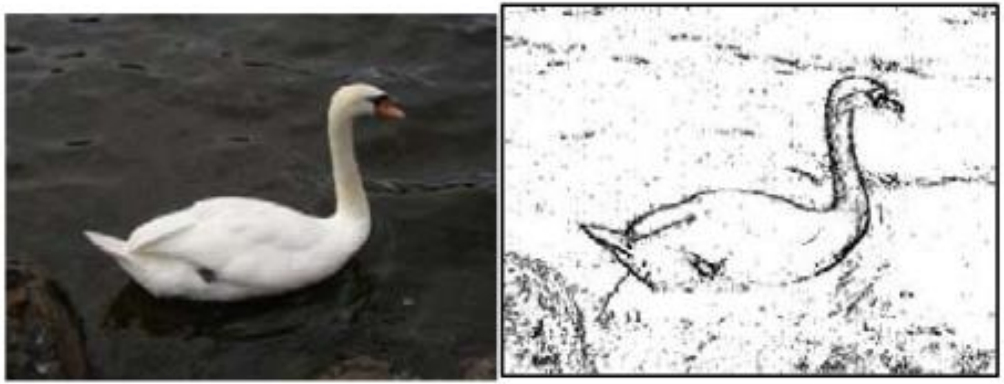
Convolved Feature

So what do we get as the output of convolved feature?



Image

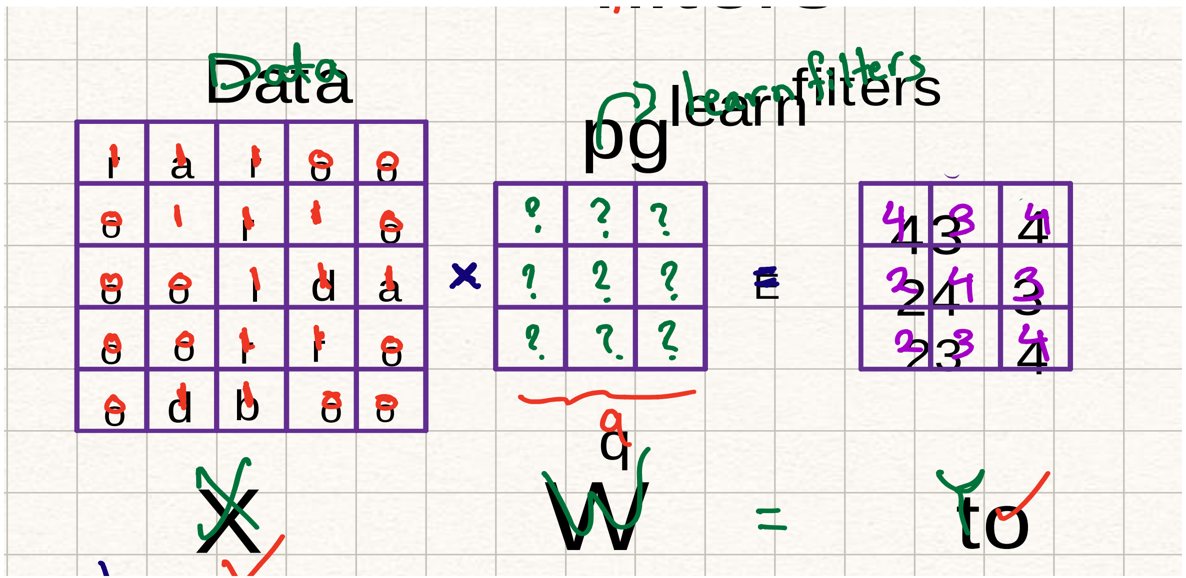
Convolved Feature



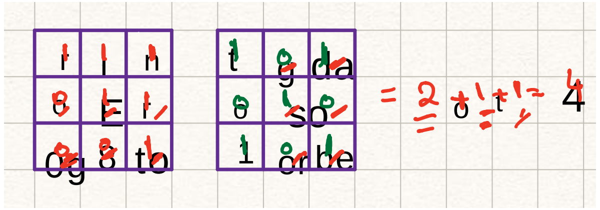
How Filters learn?

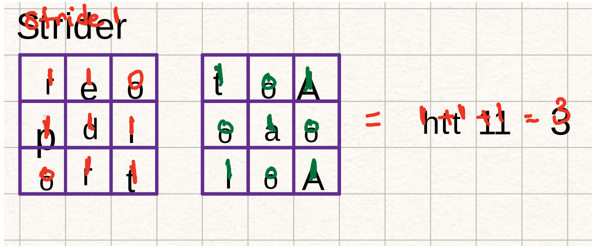
Input data

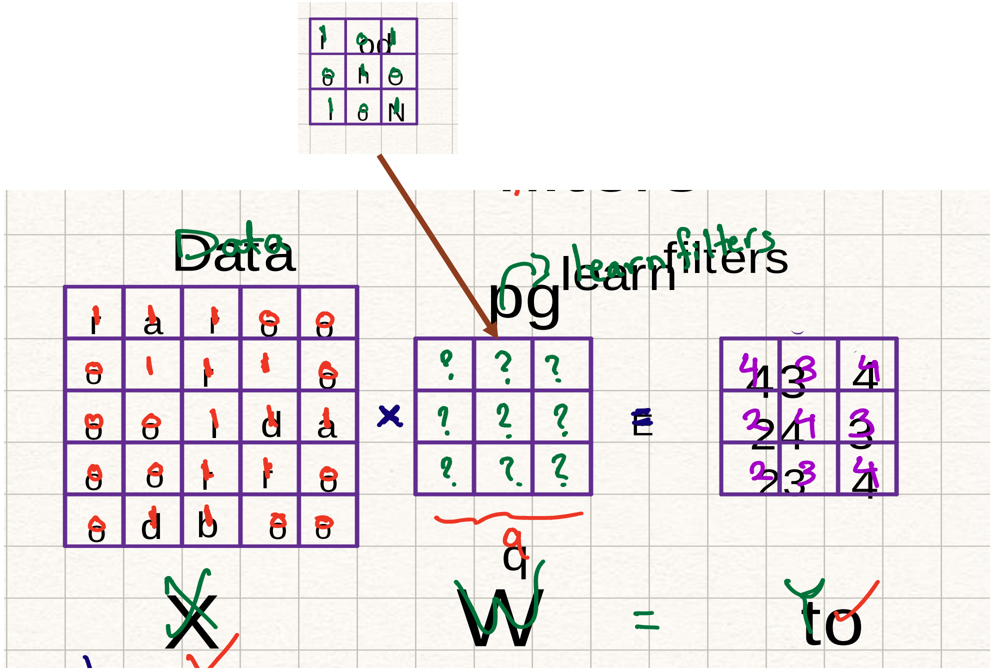
Desired pattern



How Filters work?







Some facts so far

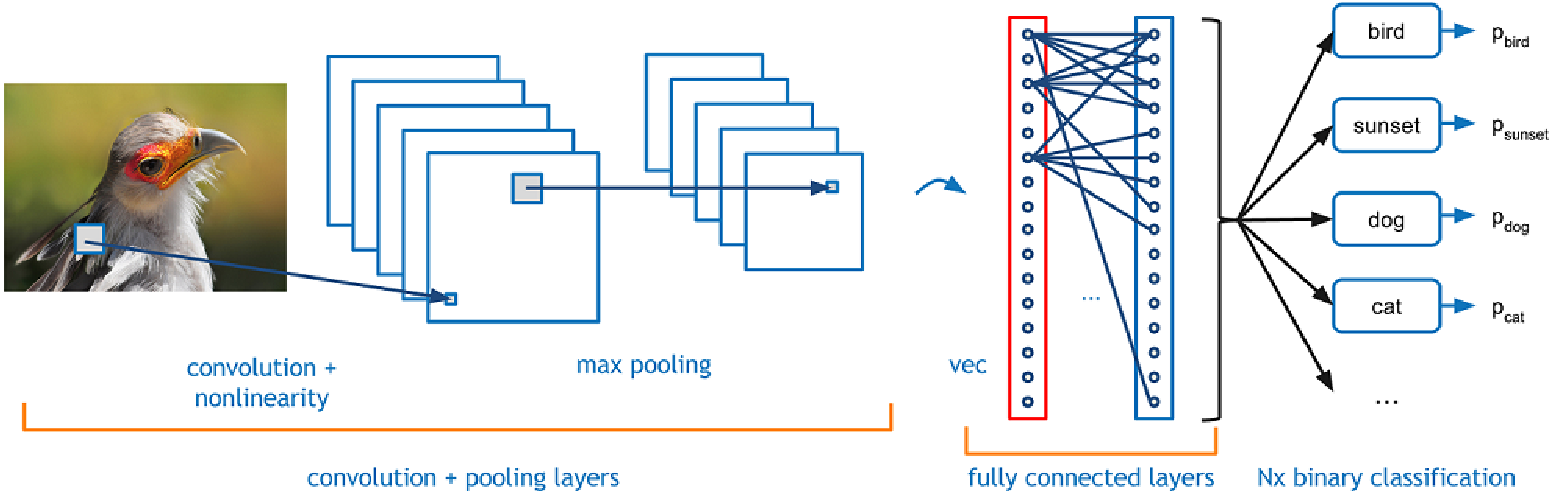
- Position-invariant pattern classification can be performed by scanning
  – 1-D scanning forsound
  – 2-D scanning for images
  – 3-D and higher-dimensional scans for higher dimensional data

- Scanning is equivalent to composing a large network with repeating subnets

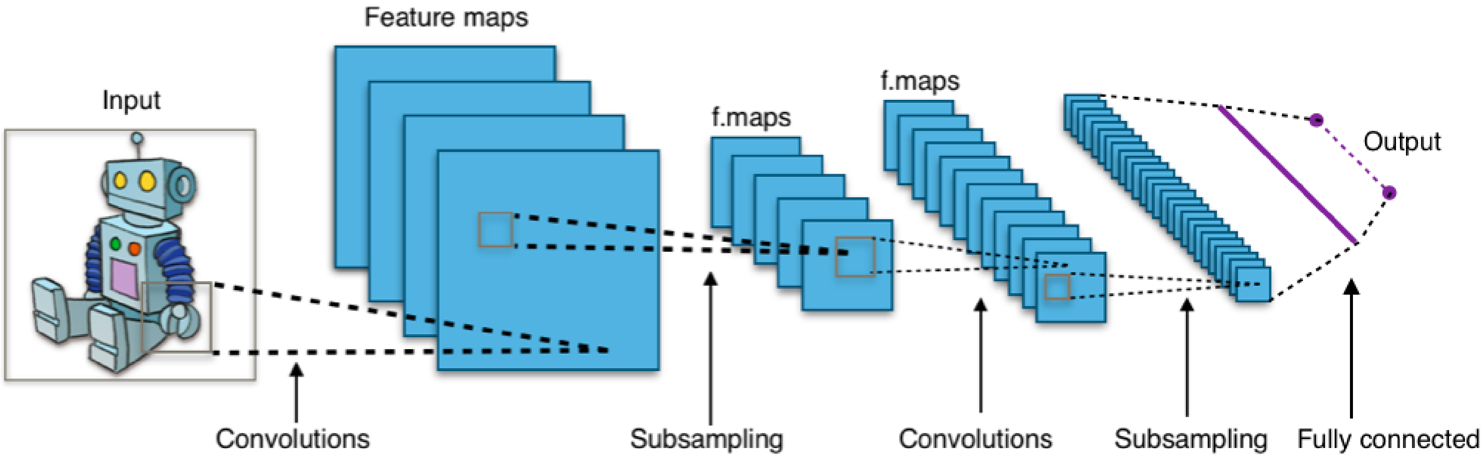
- The large network has shared subnets

- Learning in scanned networks: Backpropagation rules must be modified to combine gradients from parameters that share the same value
- – The principle applies in general for networks with shared parameters

CNN Overall Architecture



CNN Overall Architecture



CNNs are Automatic Feature Detectors

Sharing Weights

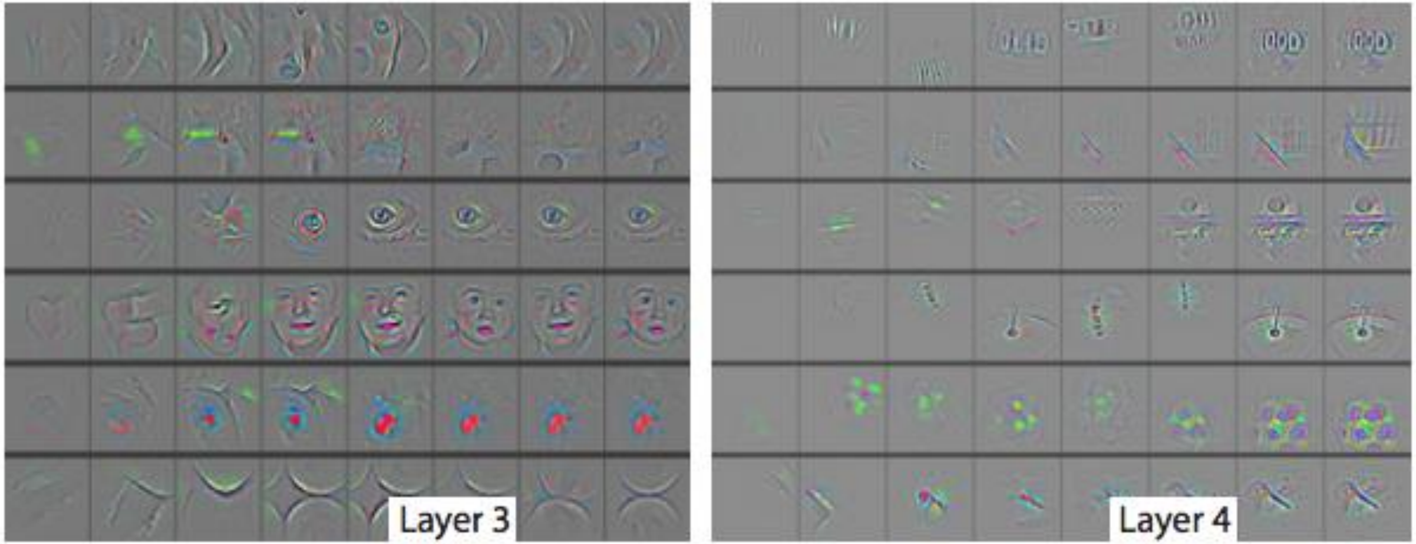
Feature Detection

Spatial Local Features

Translation In-variant

Who decides these features?

The network itself while **training** learns the filter weights and bias terms.

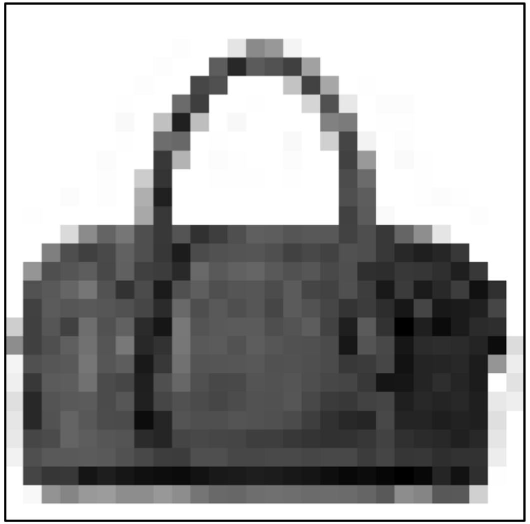


volution of randomly chosen subset of model features at **training epochs** 1,2,5,10,20,30,40,64.

Visualizing and Understanding Convolutional Networks,
Matthew D. Zeiler and Rob Fergus, ECCV 2014

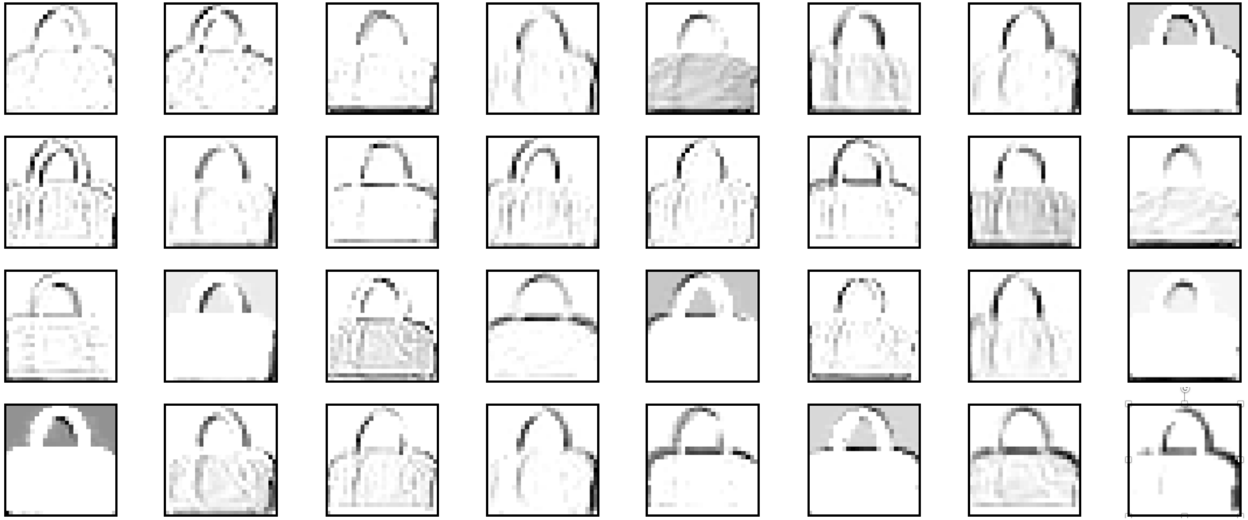
Activation map/layer=feature map

Input Image



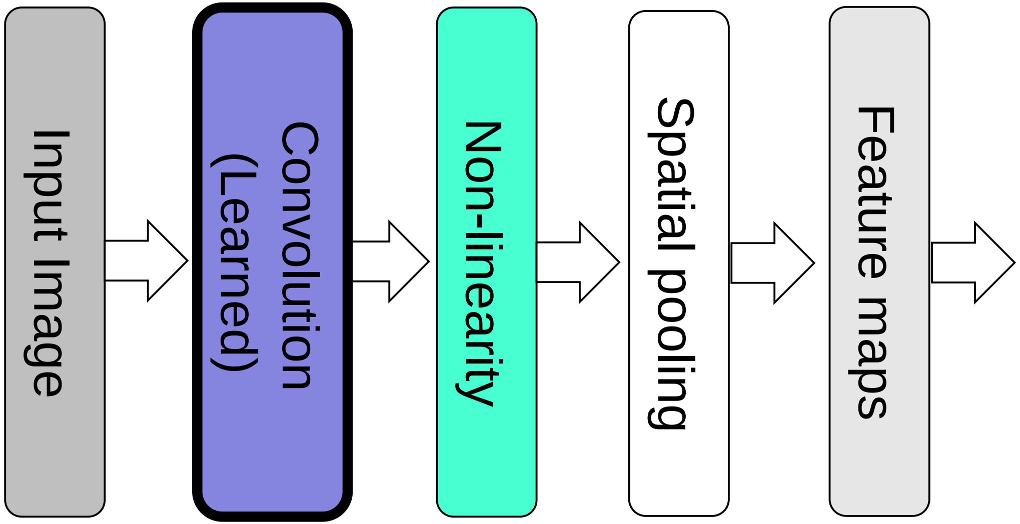
Output From Conv2D

(Feature Maps after ReLU Processing)



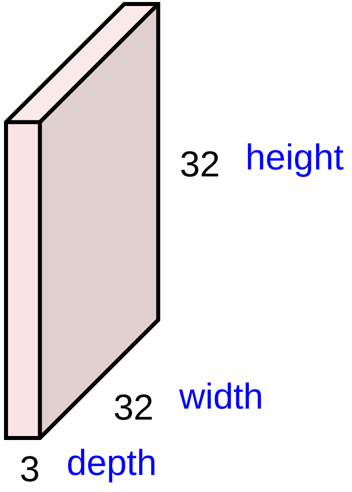
Carnegie Mellon University

CNN Components



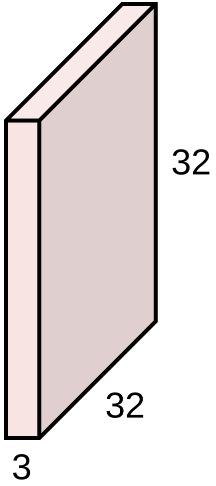
Convolution Layer

32x32x3 image -> preserve spatial structure

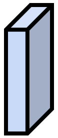


Convolution Layer

32x32x3 image



5x5x3 filter



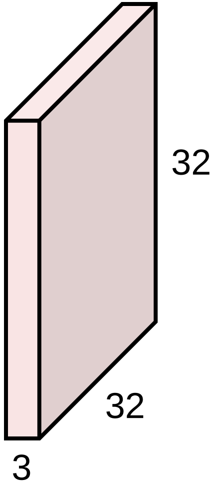
Convolve the filter with the image i.e. “slide over the image spatially, computing dot products”

Carnegie Mellon University

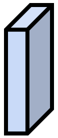
Convolution Layer

Filters always extend the full depth of the input volume

32x32x3 image

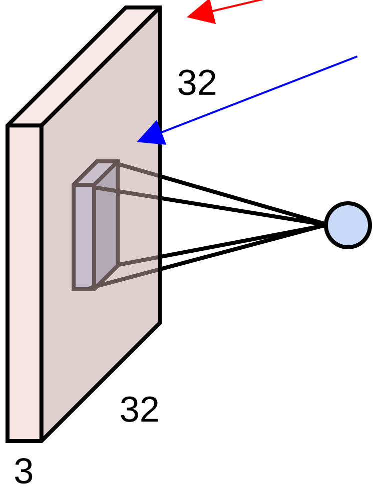


5x5x3 filter



Convolve the filter with the image i.e. “slide over the image spatially, computing dot products”

Convolution Layer

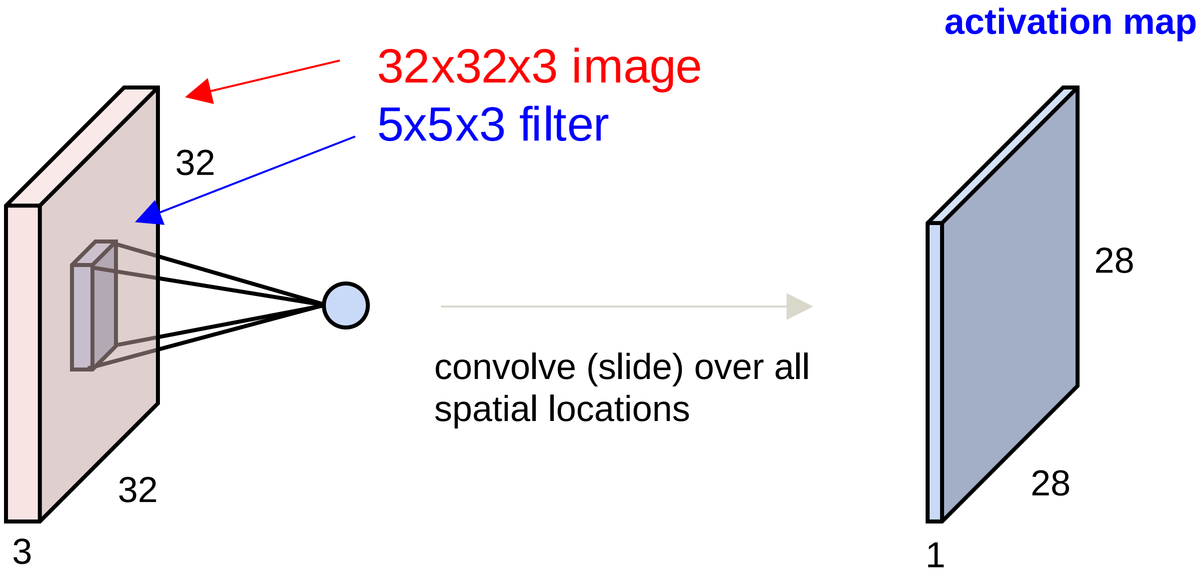


1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

Carnegie Mellon University

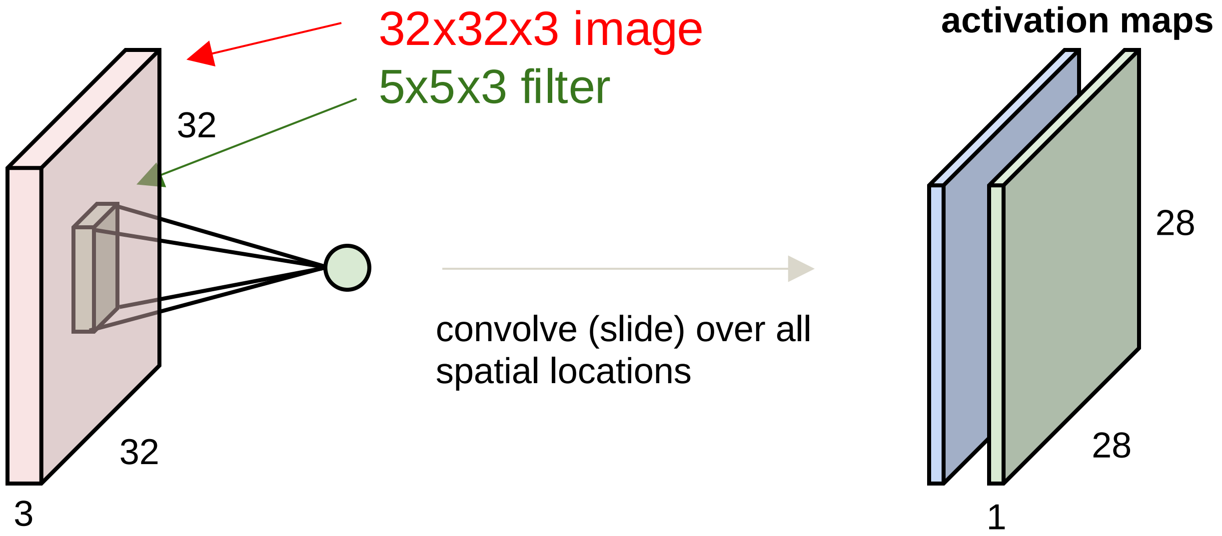
Convolution Layer



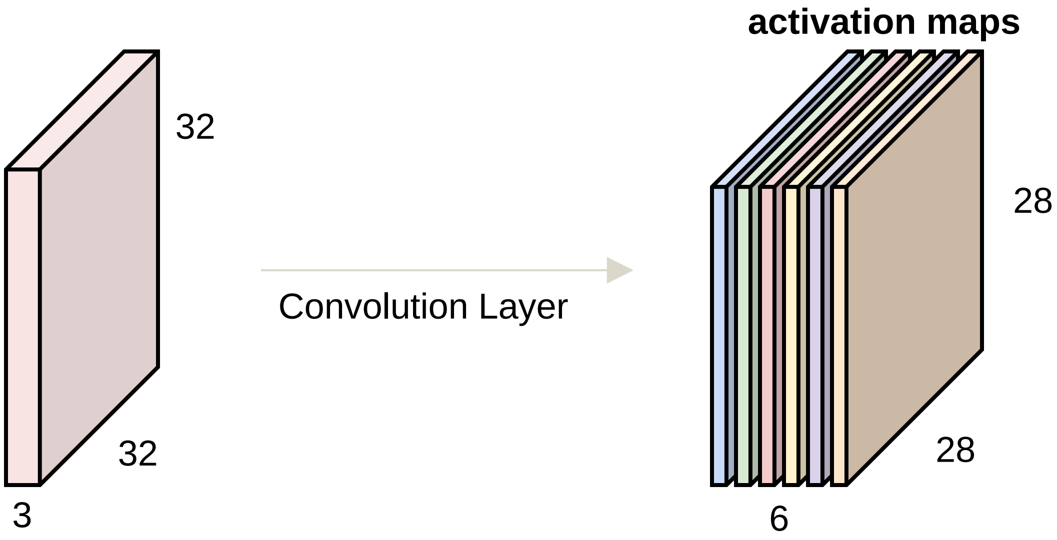
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Convolution Layer

consider a second, green filter

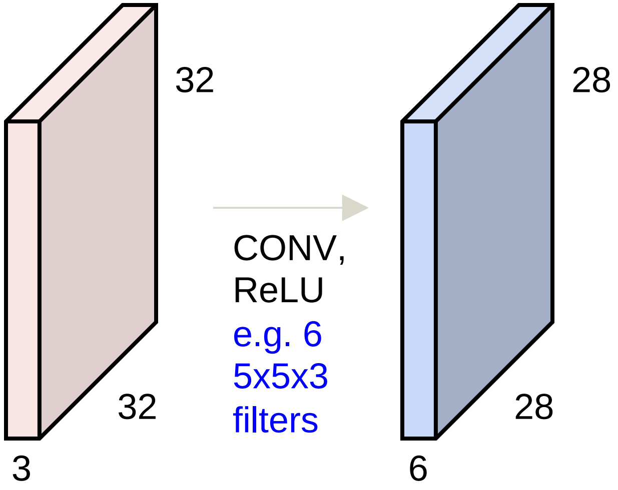


For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:

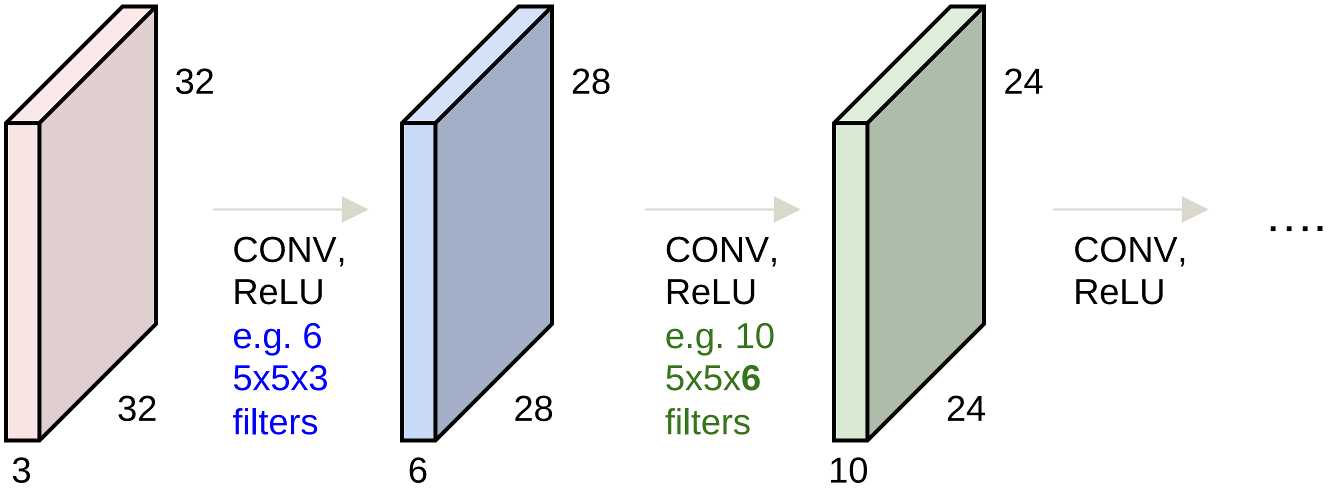


We stack these up to get a "new image" of size 28x28x6!

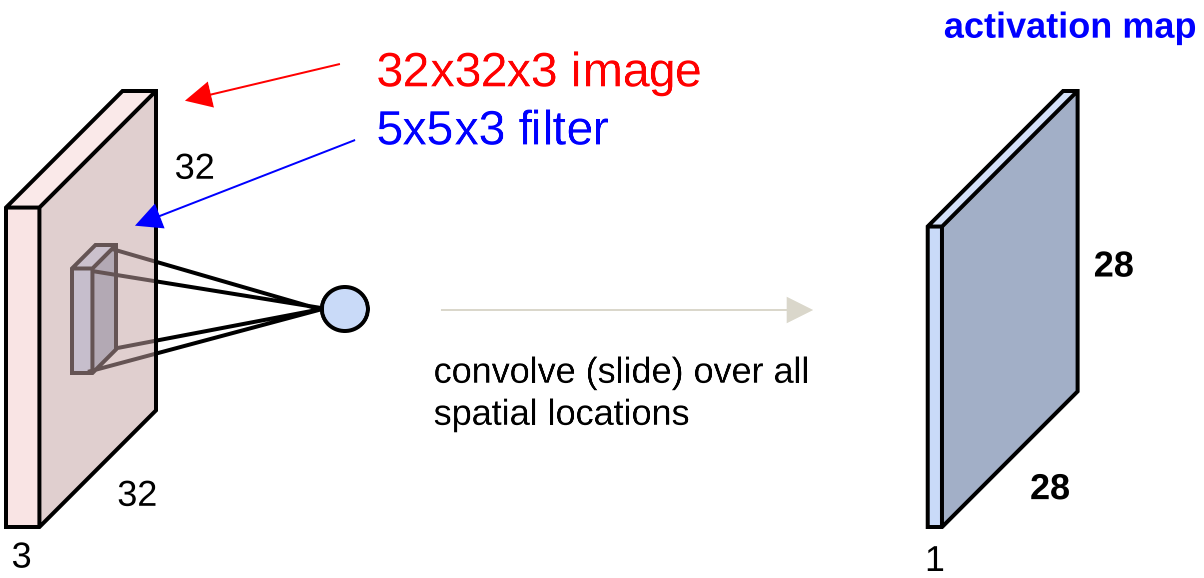
Preview: ConvNet is a sequence of Convolution Layers, interspersed with activation functions



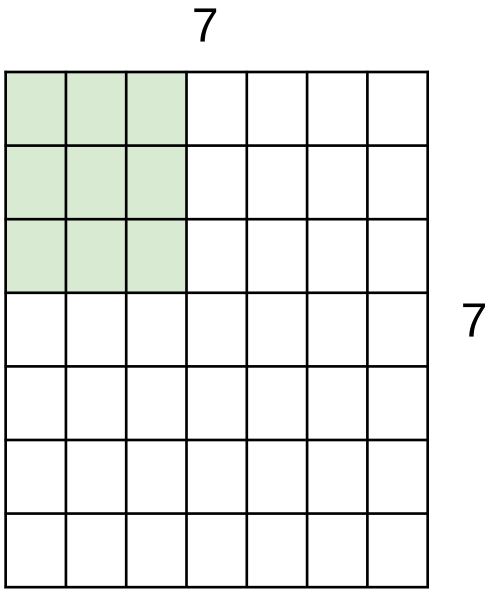
Preview: ConvNet is a sequence of Convolutional Layers, interspersed with activation functions



A closer look at spatial dimensions:

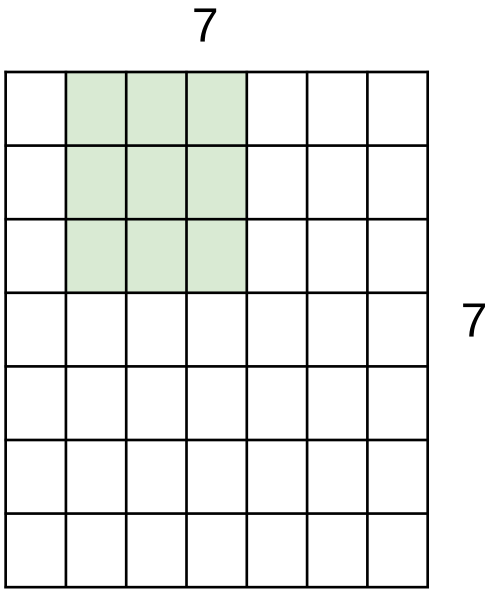


A closer look at spatial dimensions:



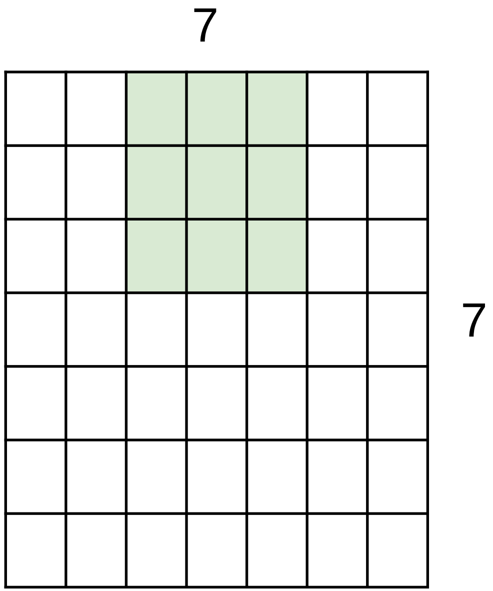
7x7 input (spatially)
assume 3x3 filter

A closer look at spatial dimensions:



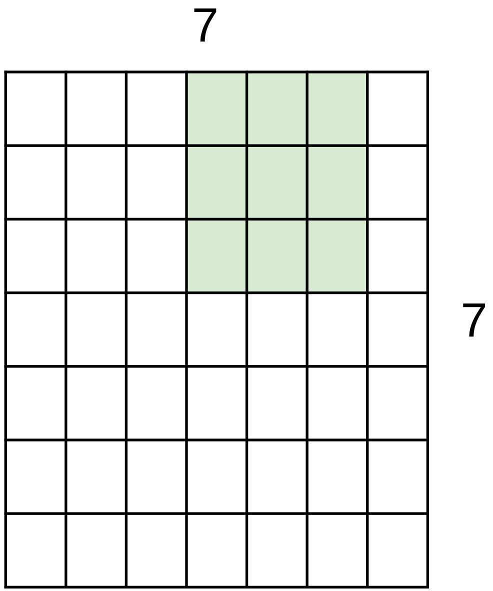
7x7 input (spatially)
assume 3x3 filter

A closer look at spatial dimensions:



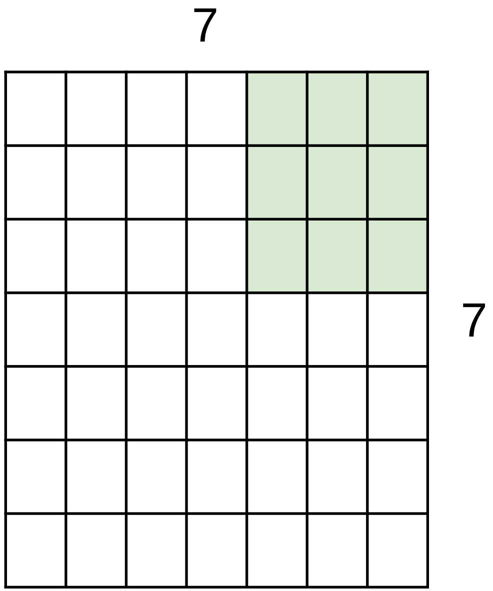
7x7 input (spatially)
assume 3x3 filter

A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

A closer look at spatial dimensions:

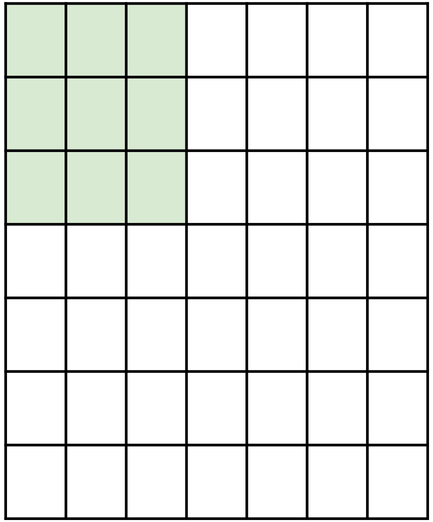


7x7 input (spatially)
assume 3x3 filter

=> 5x5 output

A closer look at spatial dimensions:

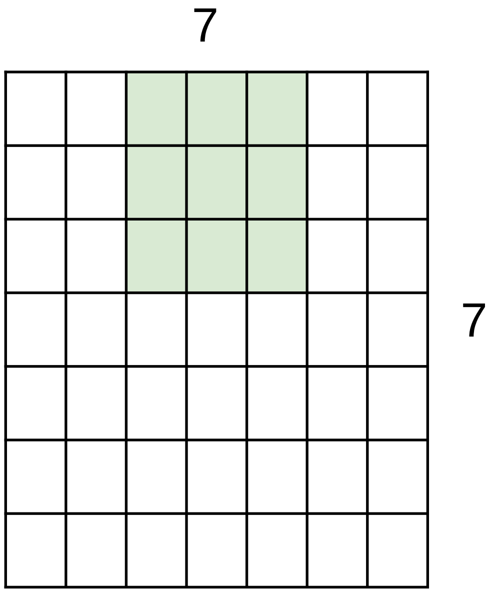
7



7

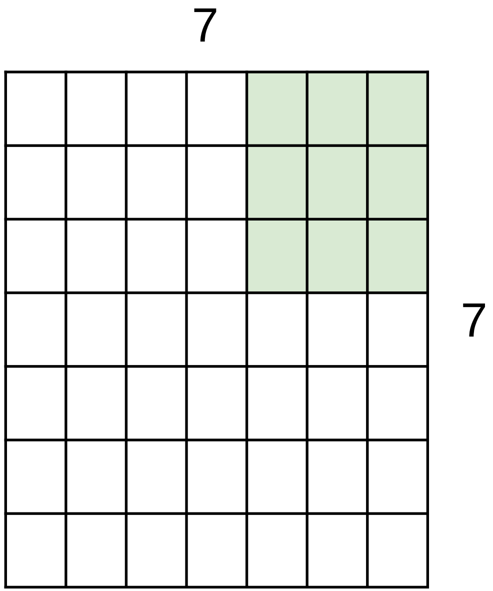
7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

A closer look at spatial dimensions:



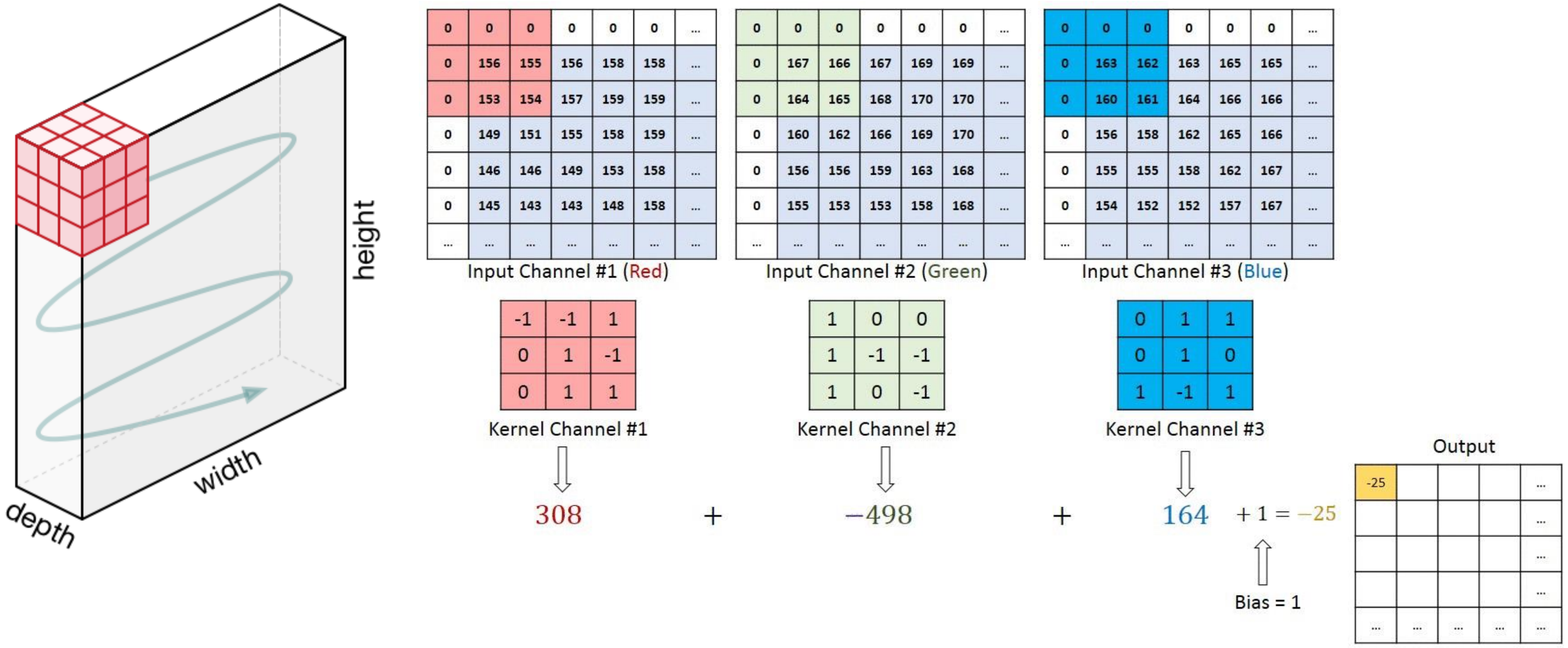
7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

A closer look at spatial dimensions:

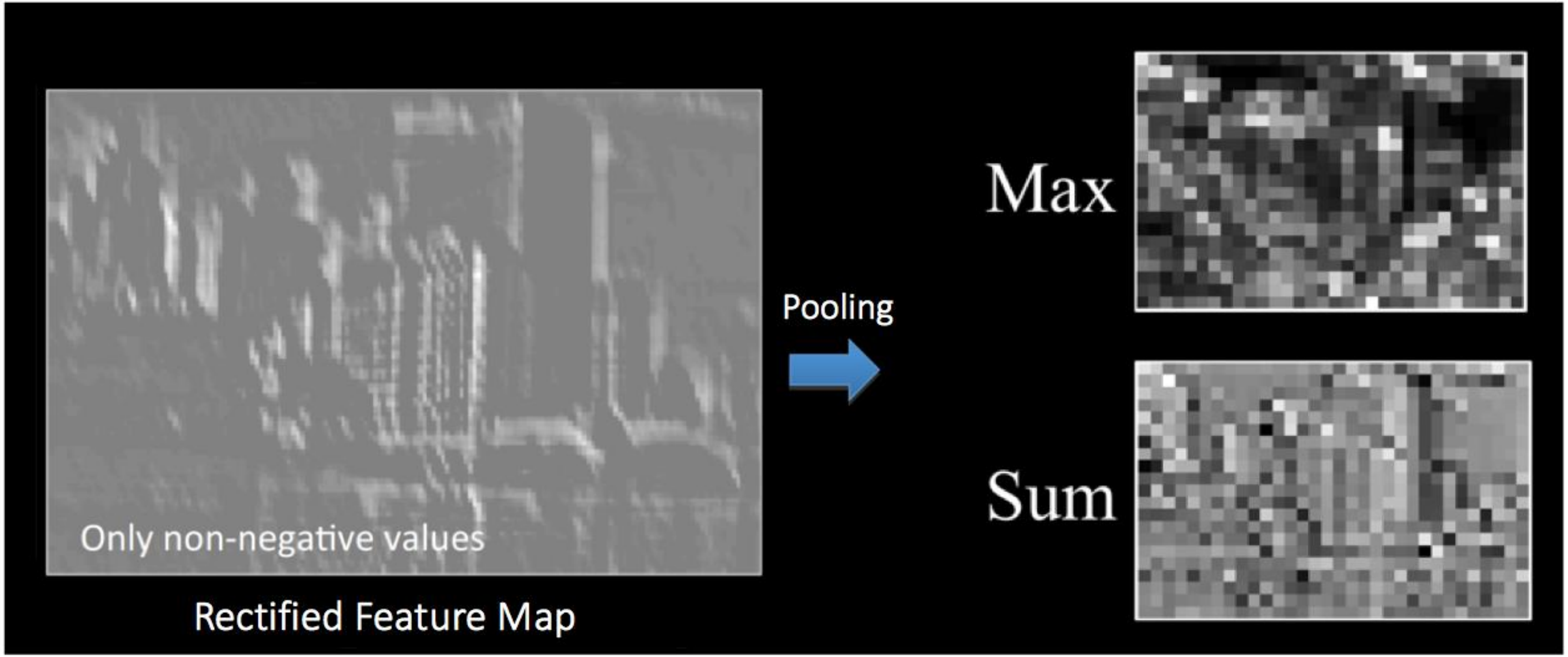


7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**
=> 3x3 output!

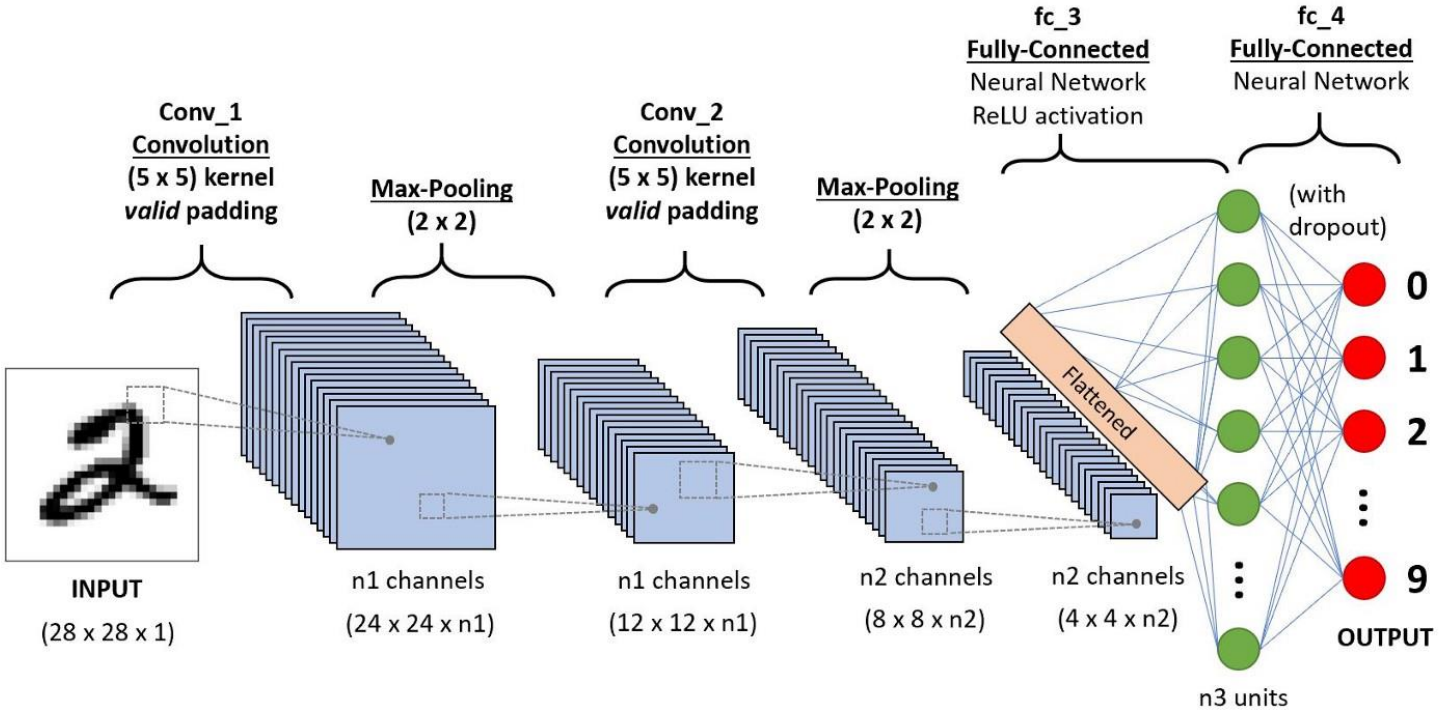
How it works for Images?



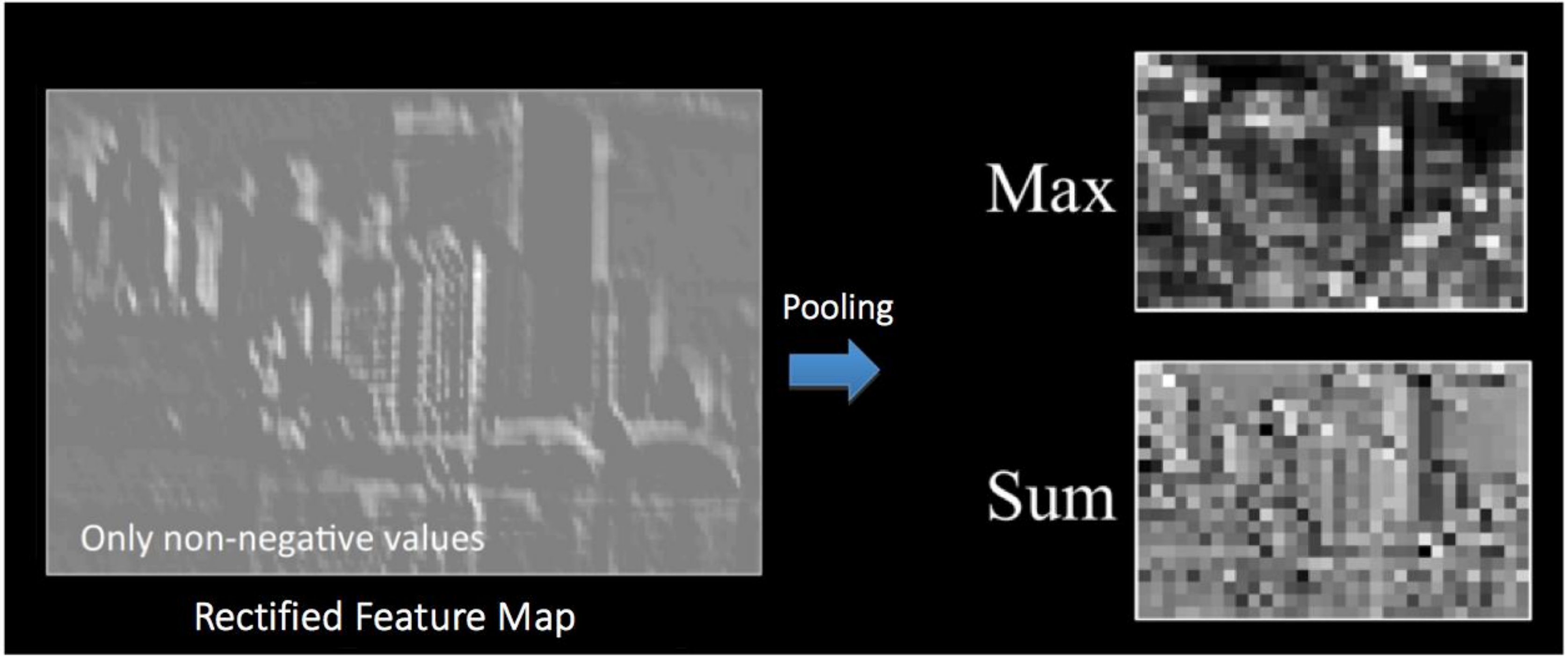
Pooling (Down sampling)



CNN Overall Architecture



Pooling (Down sampling)



Carnegie Mellon University

Pooling (Down sampling)

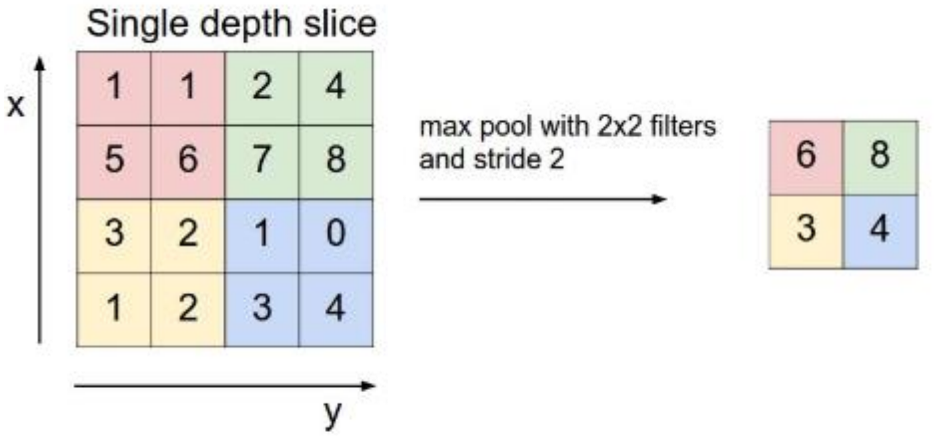


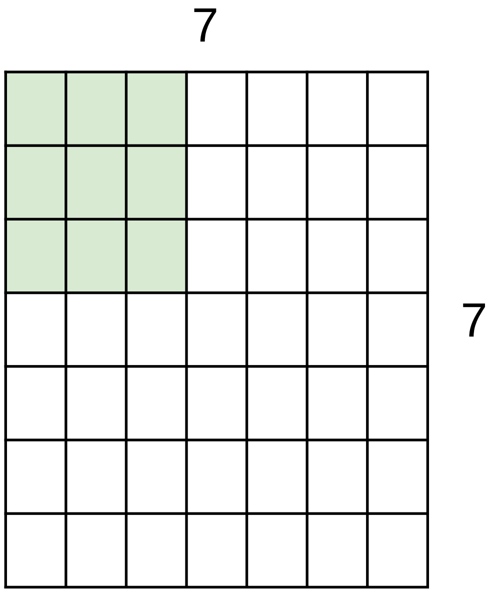
Fig. 2: F=2 S=2 Max Pooling

Max Pooling

Reduces dimensionality of each feature map, but retains the most important information

Reduced number of parameters reduces computation, memory reads, storage requirements and over-fitting to training data
Makes the network invariant to small transformations in input image, as max pooled value over local neighborhood won't change on small distortions

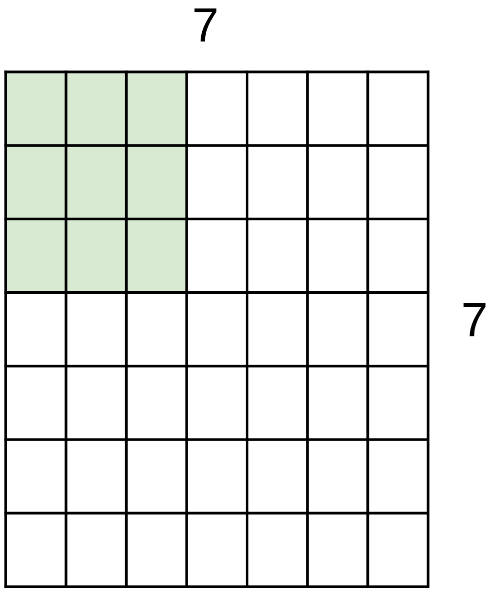
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter
applied **with stride 3?**

Carnegie Mellon University

A closer look at spatial dimensions:

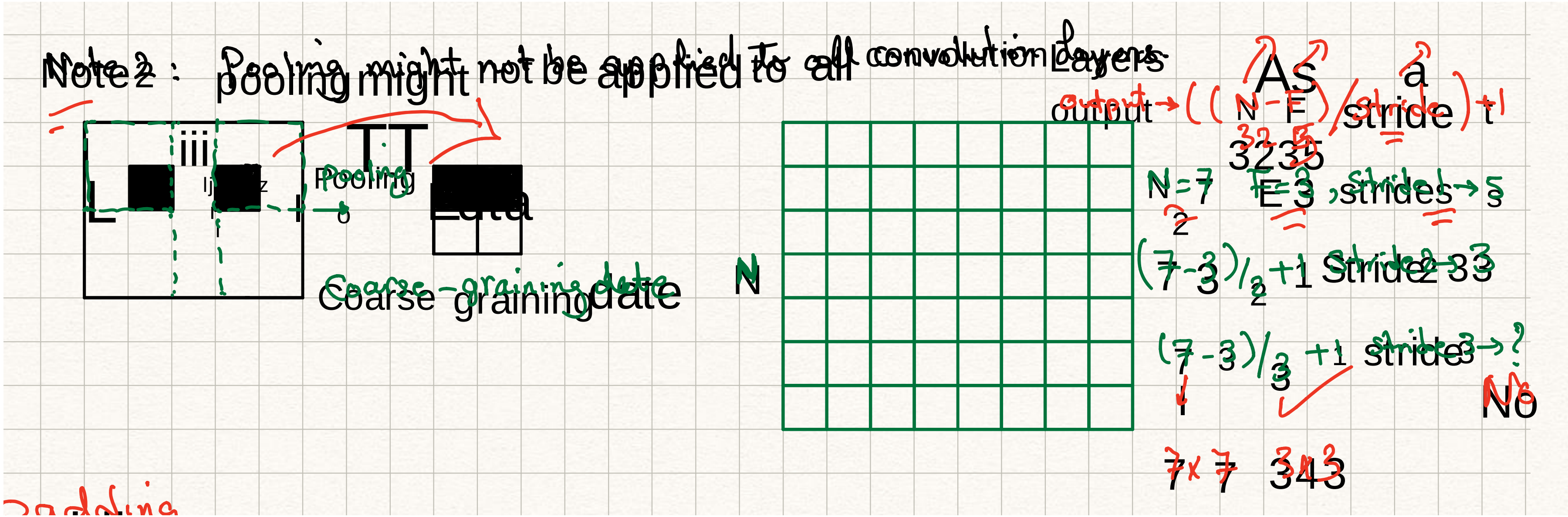


7x7 input (spatially)
assume 3x3 filter
applied **with stride 3?**

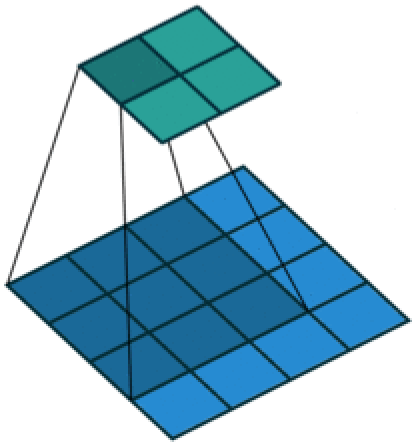
doesn't fit!
cannot apply 3x3 filter on
7x7 input with stride 3.

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

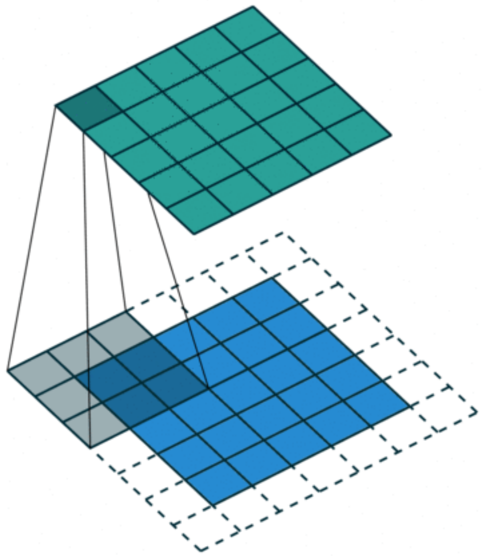
Padding Philosophy



Convolution with padding



4x4 input. 3x3 filter. Stride = 1.
2x2 output.



5x5 input. 3x3 filter. Stride = 1.
5x5 output.

Carnegie Mellon University

Animation source: https://github.com/vdumoulin/conv_arithmetic

Padding

PADDING

MOTIVATION

We have a dimensional constraint on CNN: the dimension of an output layer is $(N - F)/S + 1$, where N is the dimension of the input, F is the dimension of the filter, and S is the stride. Under this constraint, certain inputs cannot be performed CNN. For instance, if we have $N = 7$, $F = 3$, $S = 3$, the output dimension is non-integer. Therefore, we need a tool to fix dimension disagreement.

ZERO-PADDING

For spatial rearrangement, we add zeros outside the original data set. For instance, if our original data set is $\begin{bmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \\ 1 & 2 & 3 \end{bmatrix}$, after zero-padding once, the resulting matrix is $\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 0 \\ 0 & 1 & 2 & 3 & 0 \\ 0 & 1 & 2 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$.

If we apply zero padding once to the sample dimension mentioned in the motivation section, $N = 9$ after padding and CNN can be performed on the padded data.

Carnegie Mellon University

Padding

OUTPUT DIMENSION

The output dimension of a padding layer in which the dimension of the input being N, the dimension of the filter being F, the stride being S and padding being P:

$$\text{dim} = (2P + N - F)/S + 1$$

And we can analyze input and output on a Conv Layer:

1. Accepts a volume of size $W_1 \times H_1 \times D_1$
2. Takes two hyperparameters:
   (a) number of filters K
   (b) spatial extent F
   (c) stride S
   (d) amount of zero padding P
3. Produces a volume of size $W_2 \times H_2 \times D_2$:
   (a) $W_2 = (W_1 - F + 2P)/S + 1$
   (b) $H_2 = (H_1 - F + 2P)/S + 1$
   (c) $D_2 = K$
4. With parameter sharing, it introduces FFD1 weights per filter, for a total of $(F \cdot F \cdot D1) \cdot K$ weights and K biases

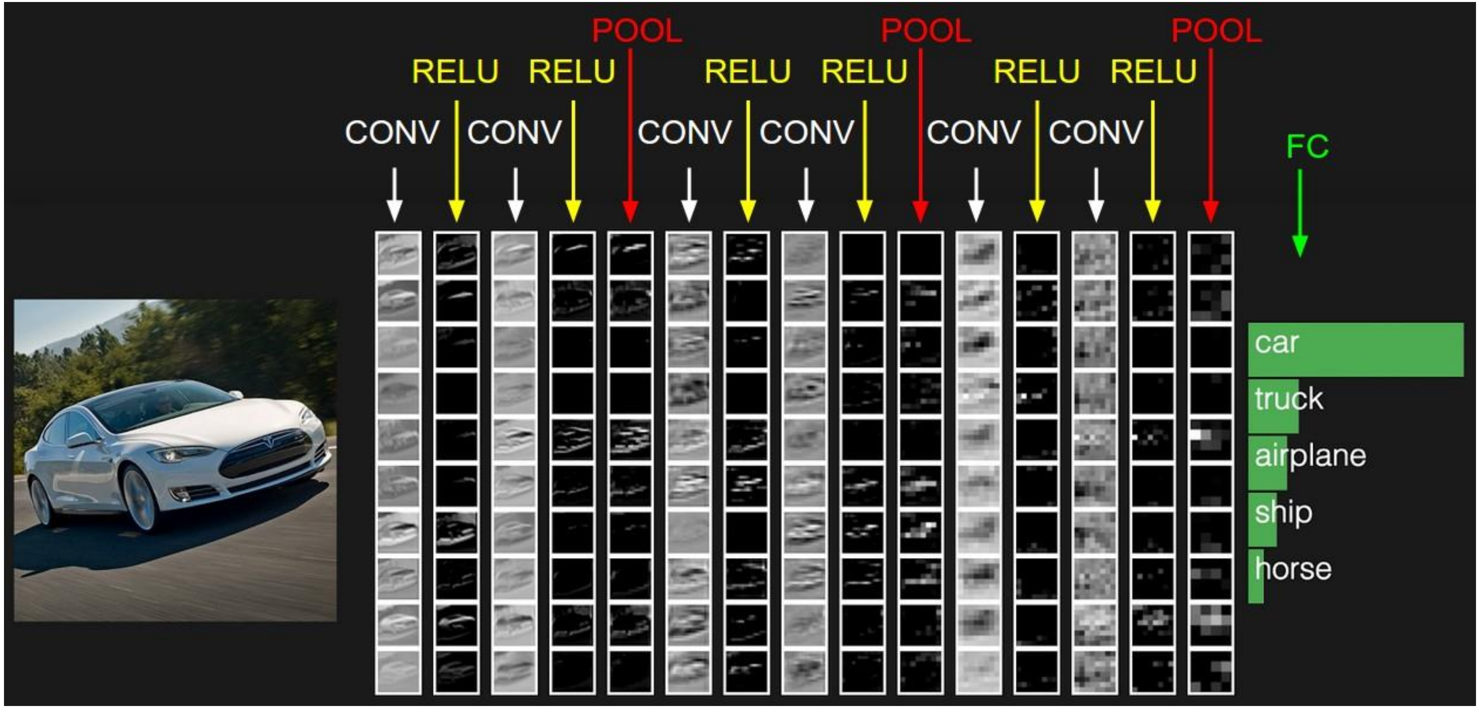
Note: A common setting of the hyperparameters is $\{F = 3, S = 1, P = 1\}$

Carnegie Mellon University

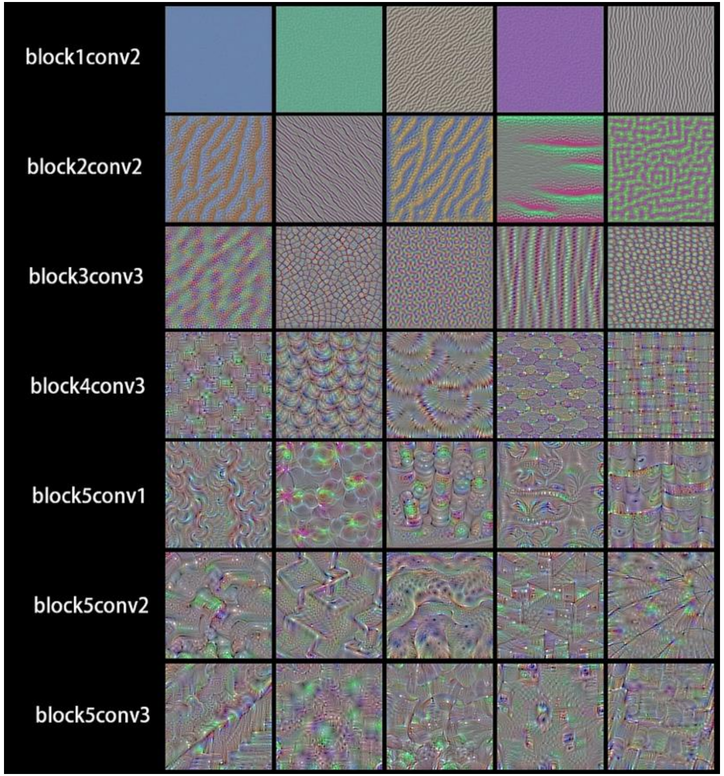
ReLU

Carnegie Mellon University

Feature Map (Convolution Layer)

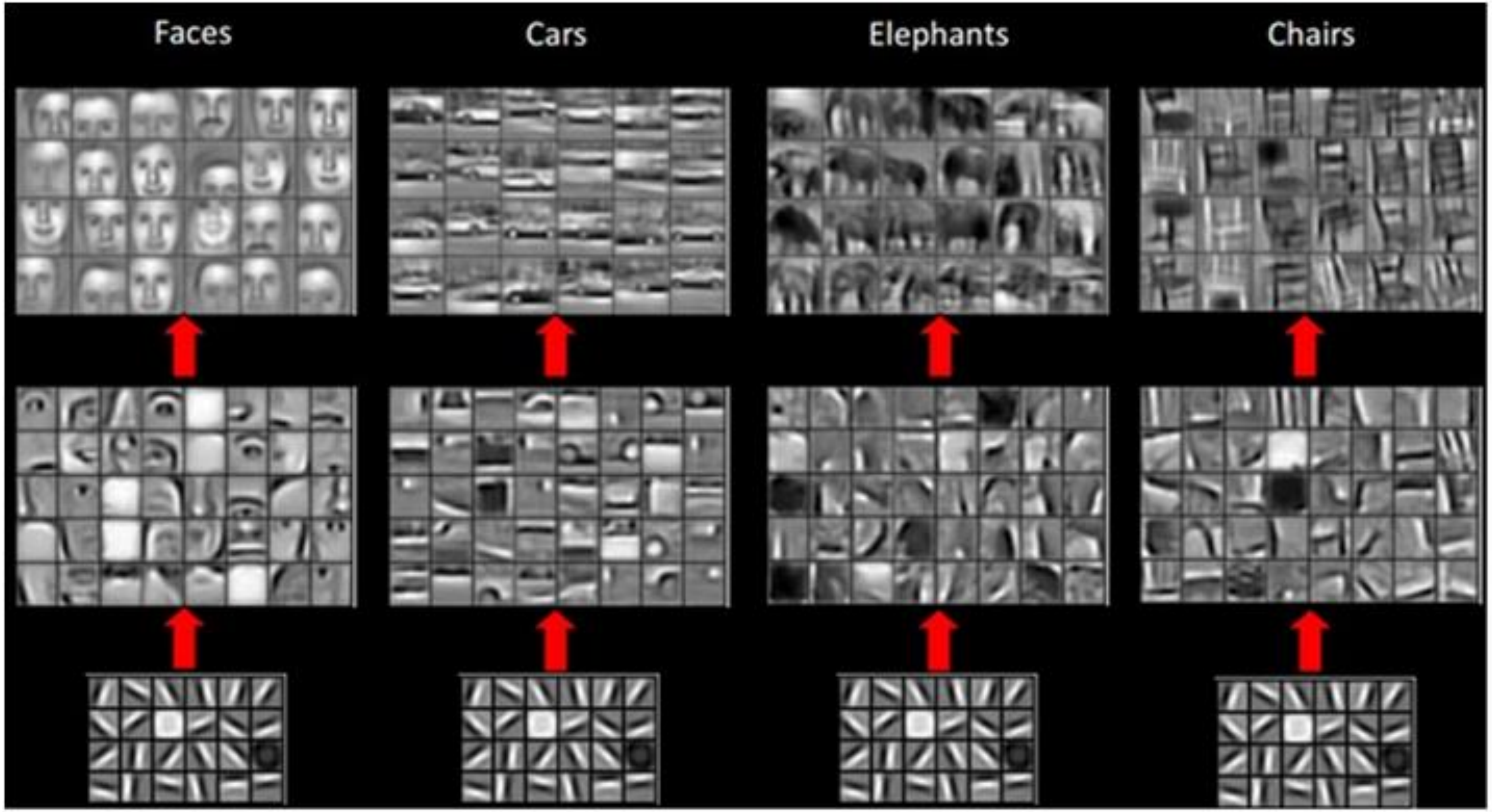


Feature Map

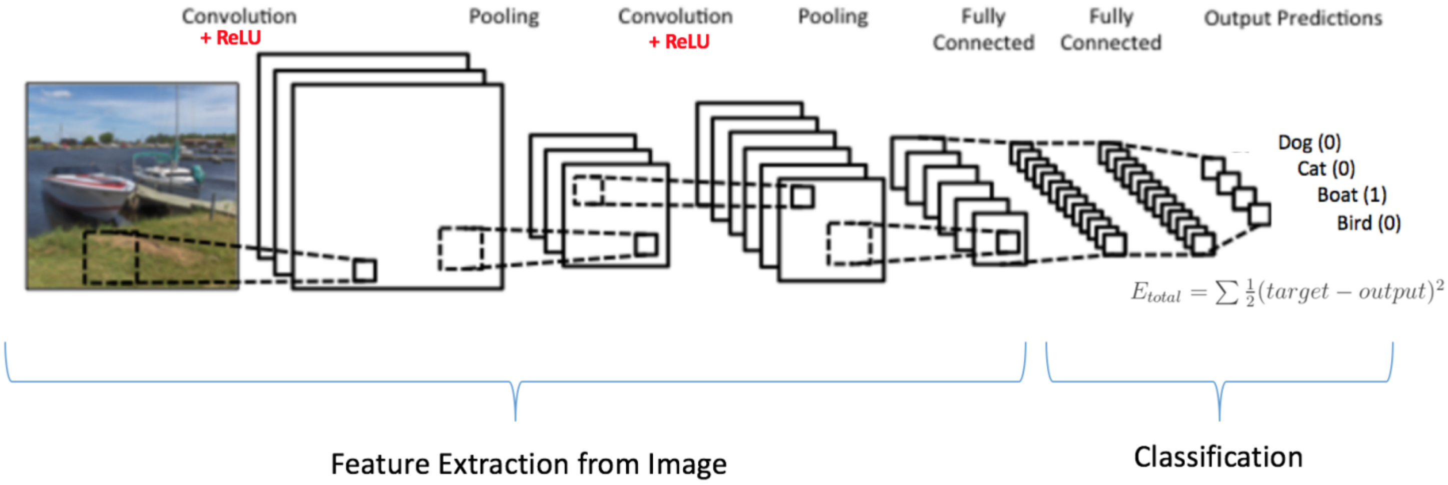

block1conv2
block2conv2
block3conv3
block4conv3
block5conv1
block5conv2
block5conv3

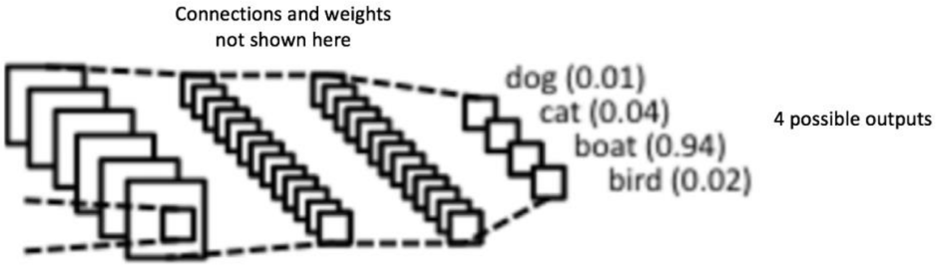
Each filter searches for a particular feature at different image locations (translation invariance)

Feature Map



Putting it together





Training Procedure

- **Step1:** We initialize all filters and parameters / weights with random values

- **Step2:** The network takes a training image as input, goes through the forward propagation step (convolution, ReLU and pooling operations along with forward propagation in the Fully Connected layer) and finds the output probabilities for each class.
  - Lets say the output probabilities for the boat image above are [0.2, 0.4, 0.1, 0.3]
  - Since weights are randomly assigned for the first training example, output probabilities are also random.

https://towardsdatascience.com/

Carnegie Mellon University

Training Procedure

- **Step3:** Calculate the total error at the output layer (summation over all 4 classes)
  - **Total Error = ∑ ½ (target probability – output probability) ²**
- **Step4:** Use Backpropagation to calculate the *gradients* of the error with respect to all weights in the network and use *gradient descent* to update all filter values / weights and parameter values to minimize the output error.
  - The weights are adjusted in proportion to their contribution to the total error.
  - When the same image is input again, output probabilities might now be [0.1, 0.1, 0.7, 0.1], which is closer to the target vector [0, 0, 1, 0].
  - This means that the network has *learnt* to classify this particular image correctly by adjusting its weights / filters such that the output error is reduced.
  - Parameters like number of filters, filter sizes, architecture of the network etc. have all been fixed before Step 1 and do not change during training process – only the values of the filter matrix and connection weights get updated.

https://towardsdatascience.com/

Training Procedure

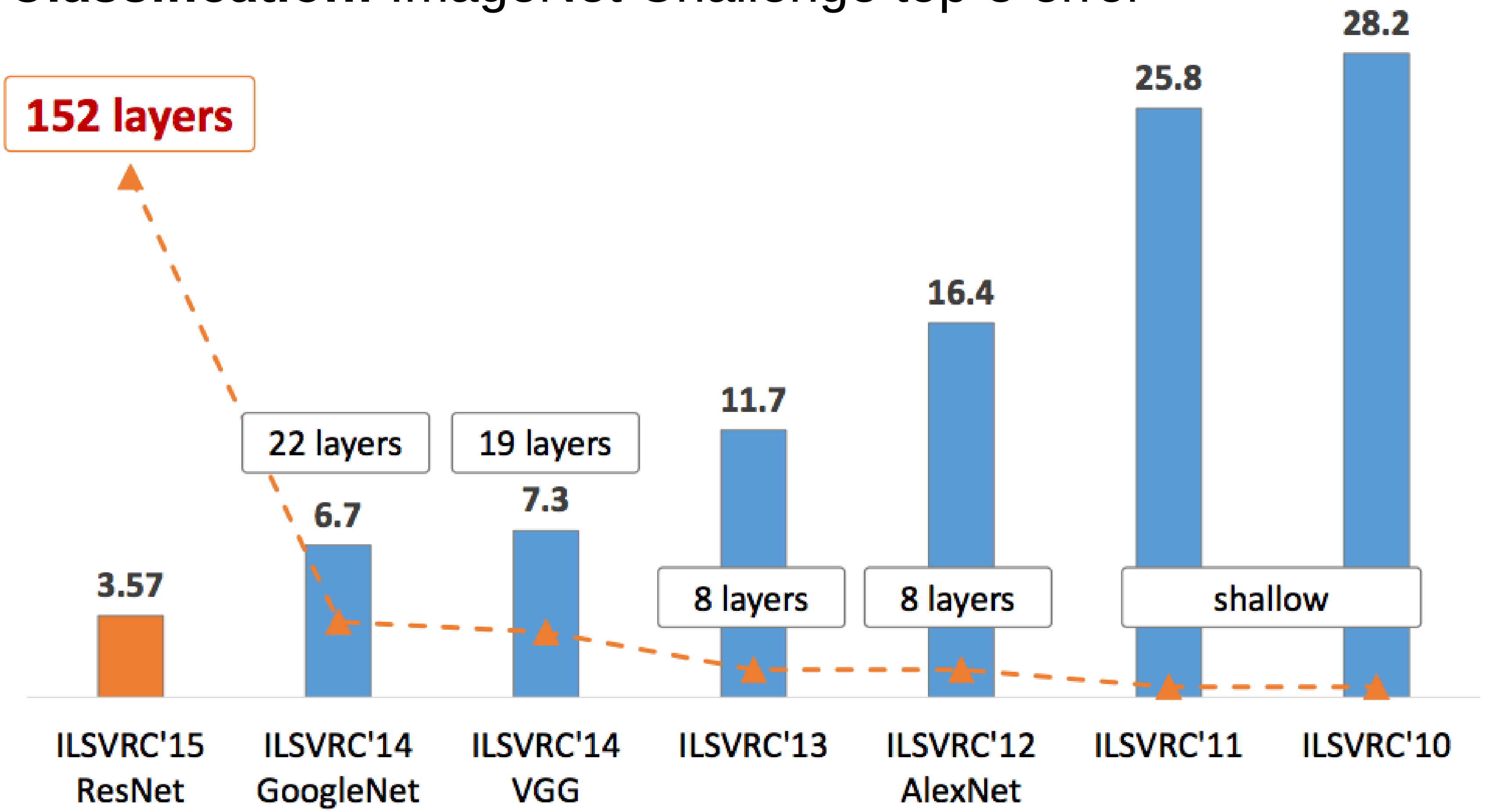
•**Step5:** Repeat steps 2-4 with all images in the training set.

The above steps *train* the ConvNet – this essentially means that all the weights and parameters of the ConvNet have now been optimized to correctly classify images from the training set.

When a new (unseen) image is input into the ConvNet, the network would go through the forward propagation step and output a probability for each class (for a new image, the output probabilities are calculated using the weights which have been optimized to correctly classify all the previous training examples). If our training set is large enough, the network will (hopefully) generalize well to new images and classify them into correct categories.

https://towardsdatascience.com/

Classification: ImageNet Challenge top-5 error



ie

ity