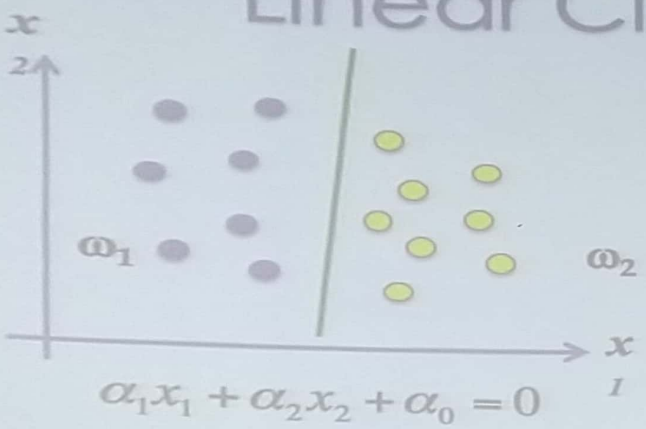
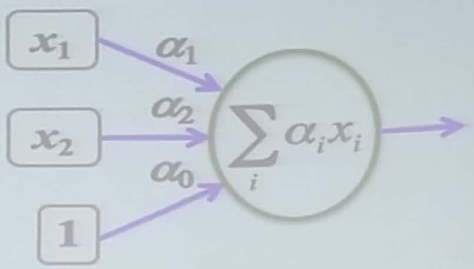
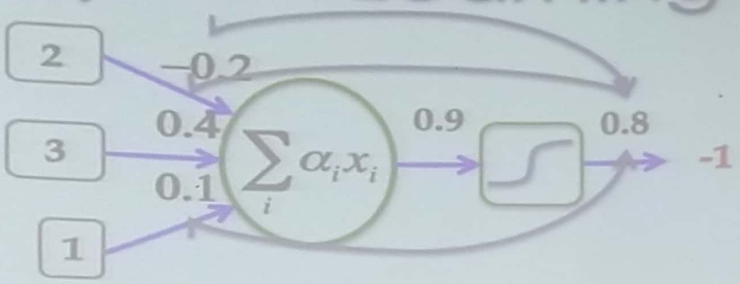
Linear Classifier





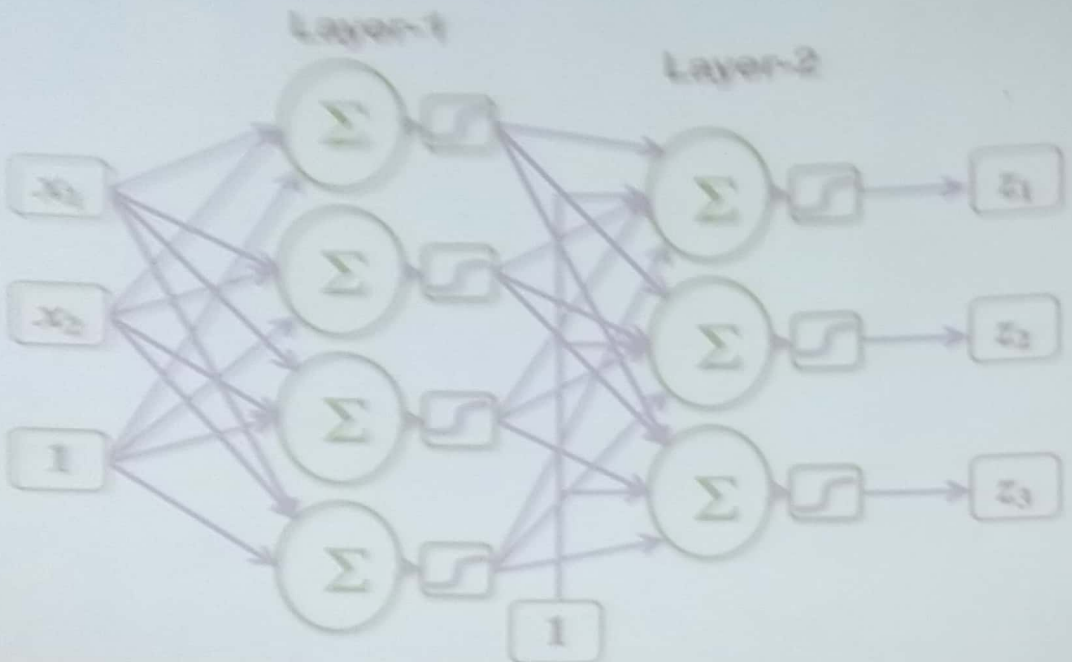
- A linear boundary separates two classes.

Perceptron Learning

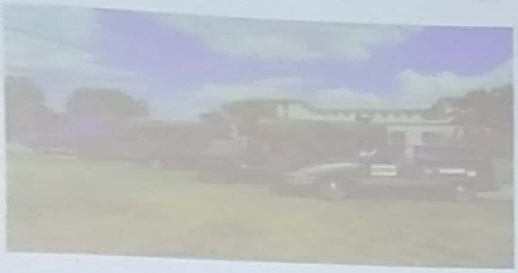


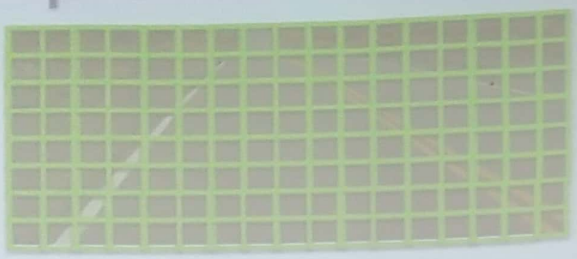
- Randomly Initialize the weights
- For each sample:
  - Feed a sample and find the output (forward pass)
  - Find the difference between actual and desired outputs (cost function)
  - Find the effect of each weight on the cost (derivative)
  - Update the weights with a learning rate (GD)

Multi-layer Perceptron



MLP in Computer Vision

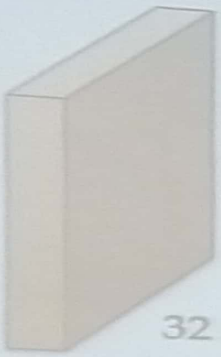




- 30x32 "Input Retina"
- 5 hidden units
- 30 output units
  - Sharp Left to Sharp Right

Convolution Layer

- 32 x 32 x 3 image

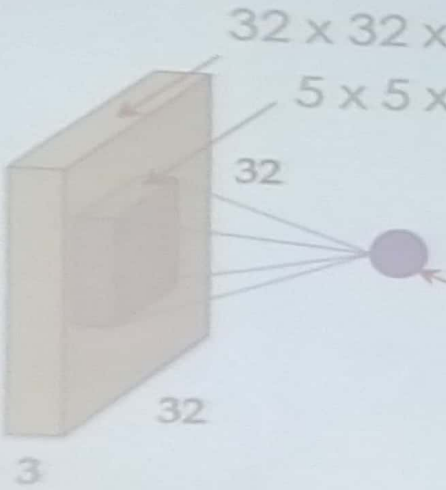


Filters always extend the full depth of the input volume

5 x 5 x 3 filter

Convolve the filter with the image. i.e. "Slide over the image spatially, computing dot products"

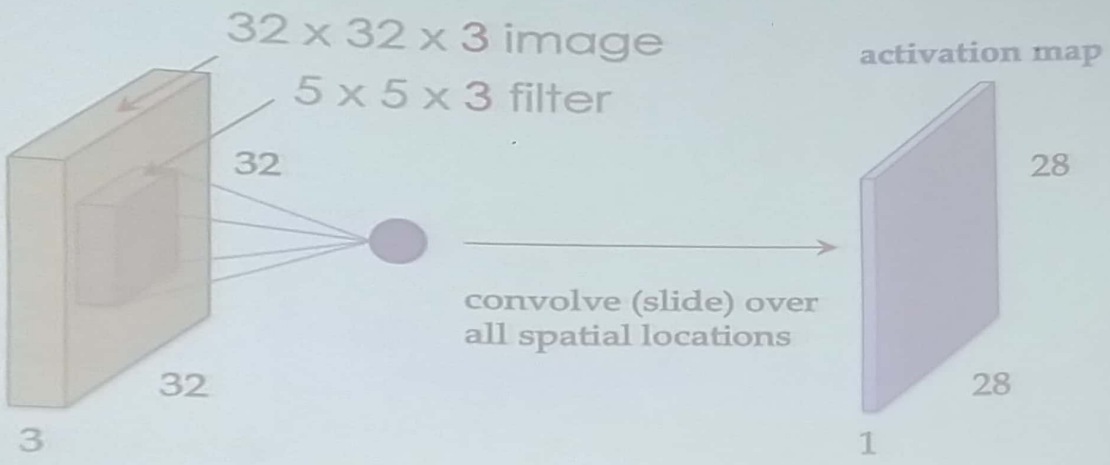
Convolution Layer



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

$$w^T x + b$$

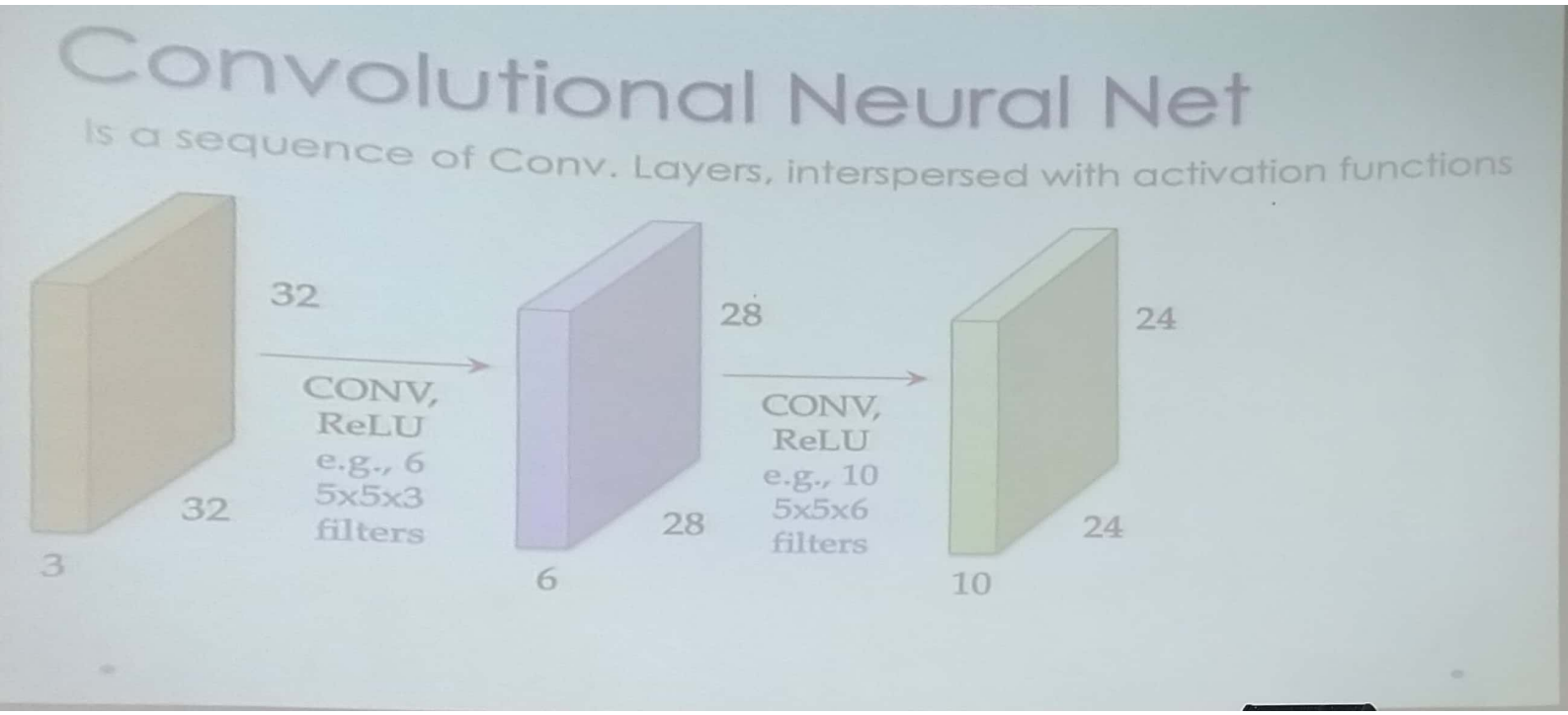
Convolution Layer

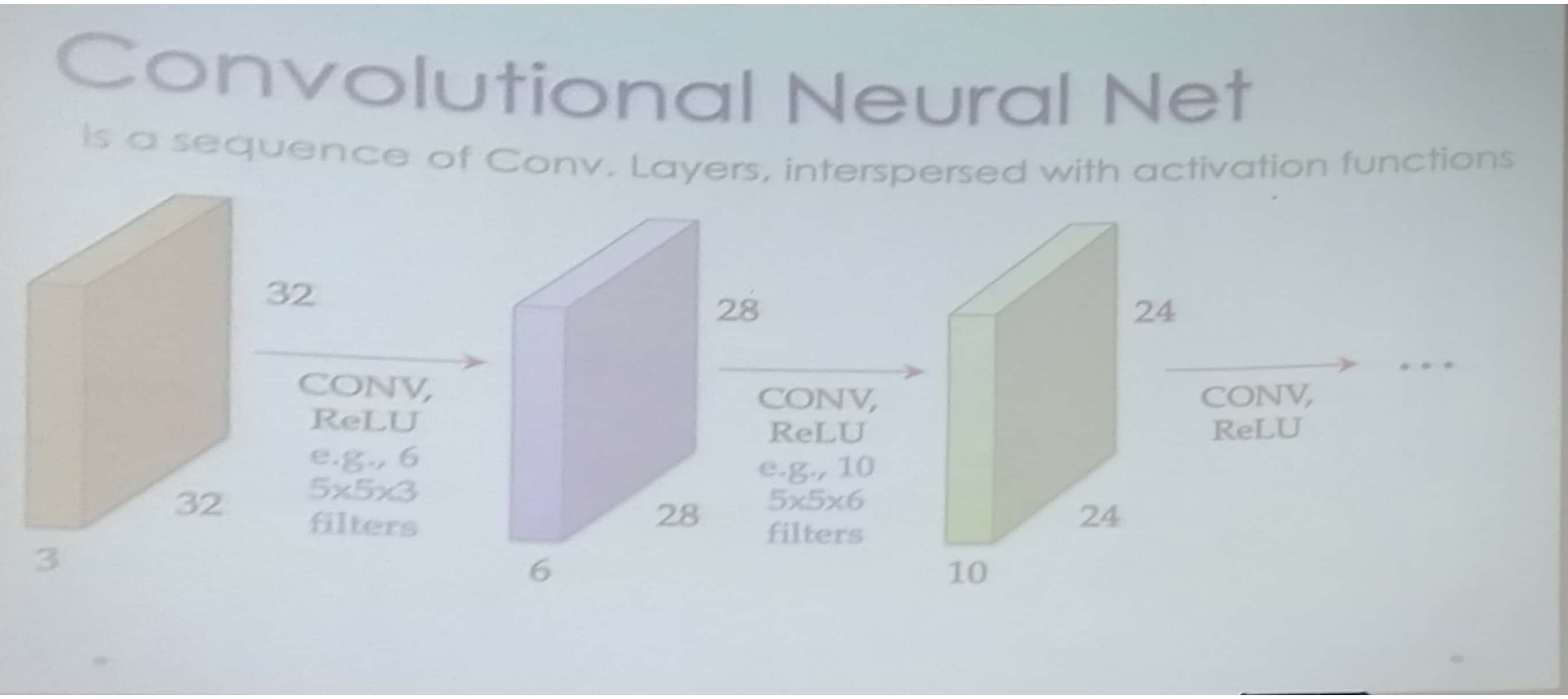


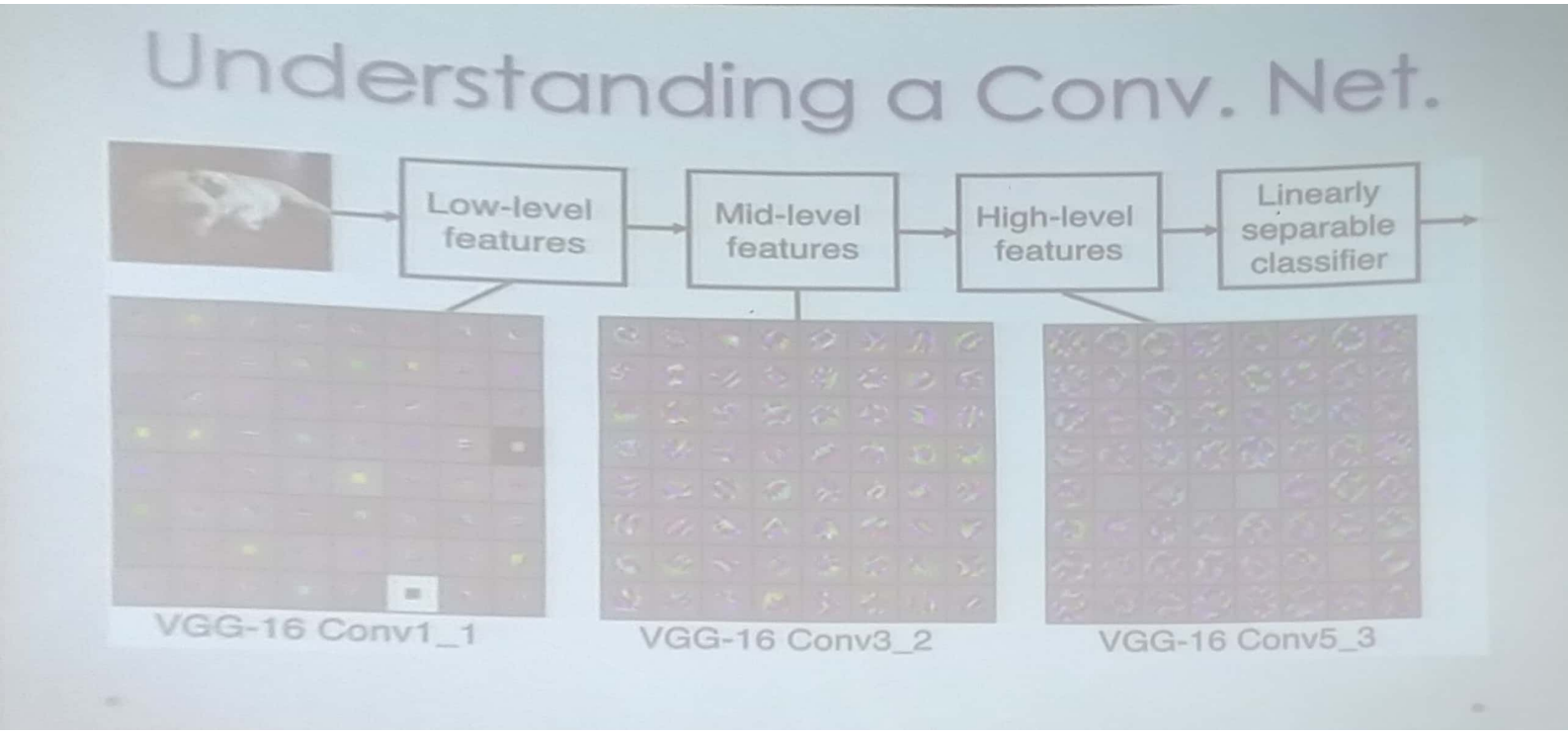
$$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1,n_2] \cdot g[x-n_1, y-n_2]$$

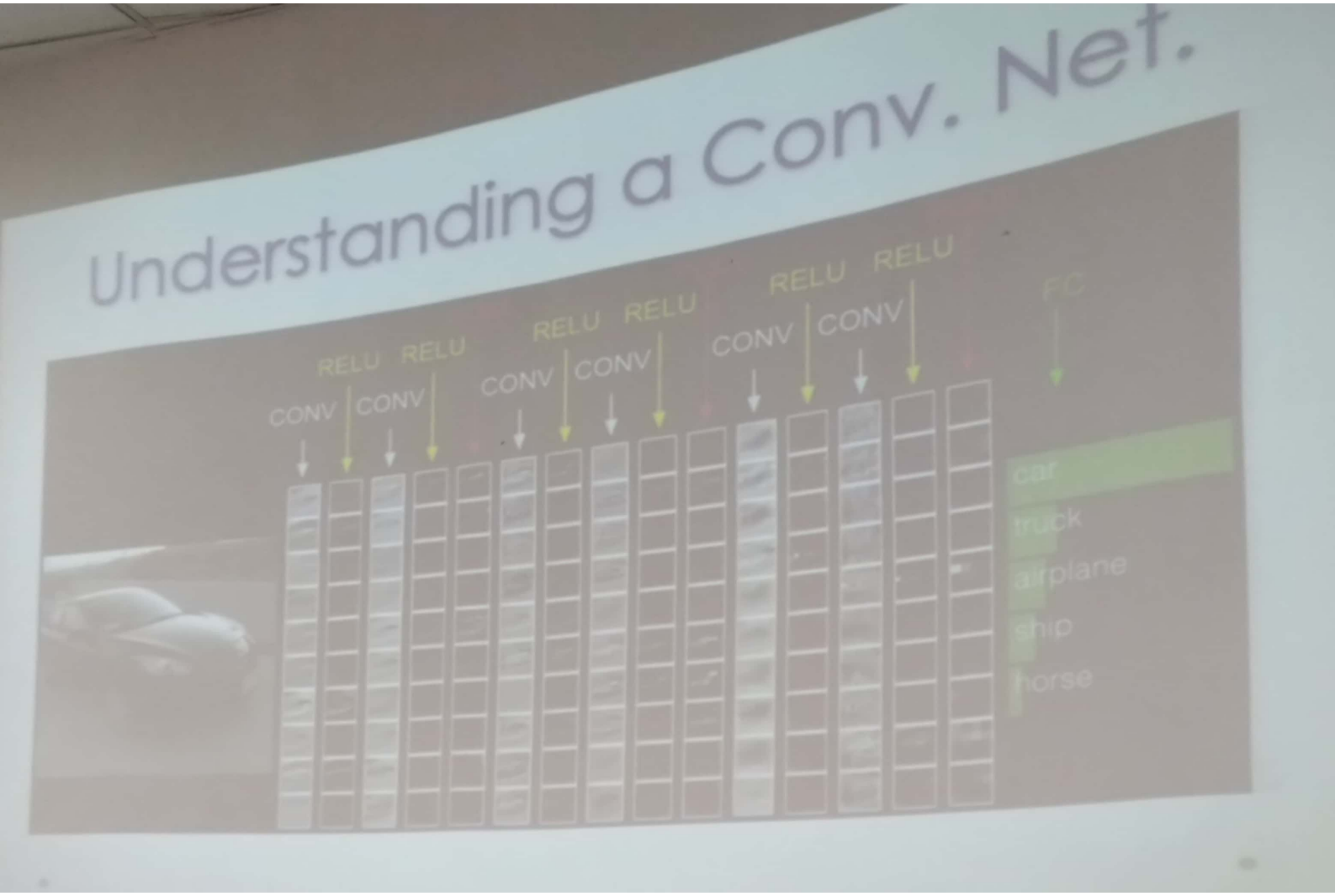
Convolutional Neural Net

Is a sequence of Conv. Layers, interspersed with activation functions

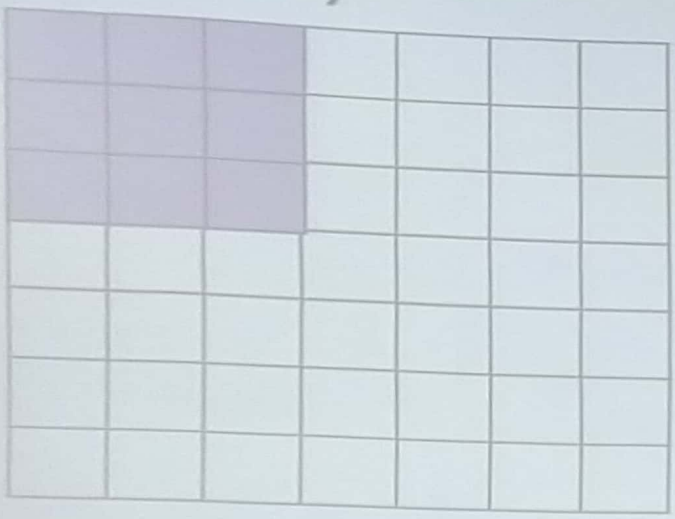








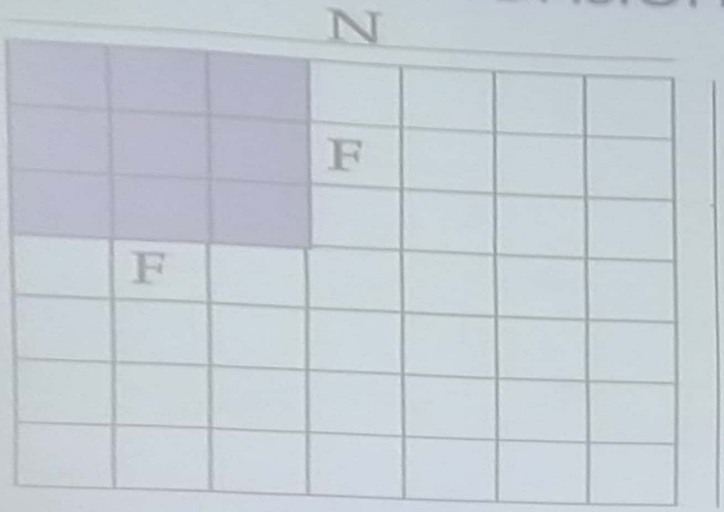
A closer look at spatial dimensions:



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

Doesn't fit !
Cannot apply 3x3 filter on a 7x7 input with stride 3

Output Dimensions:

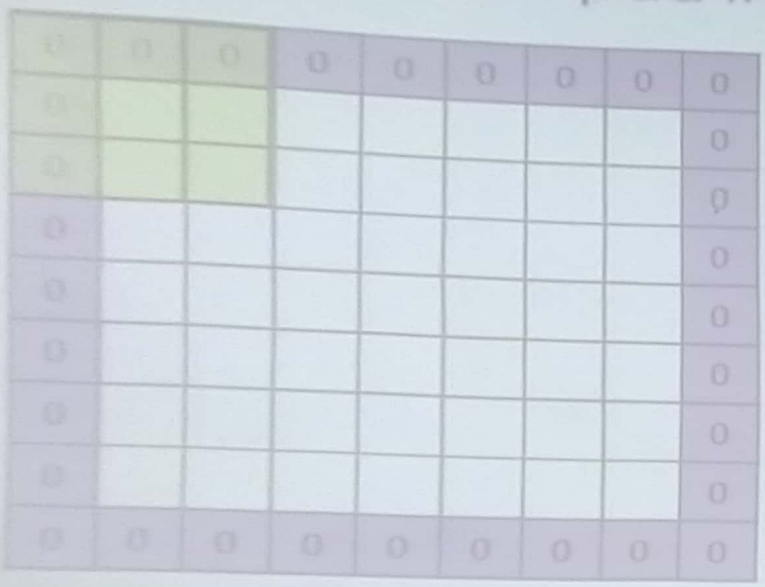


Output Size:
(N-F)/stride + 1

e.g., N = 7, F = 3

- stride 1: (7-3)/1 + 1 = 5
- stride 2: (7-3)/2 + 1 = 3
- stride 3: (7-3)/3 + 1 = 2.33 !

Common to Zero-pad the border in practice



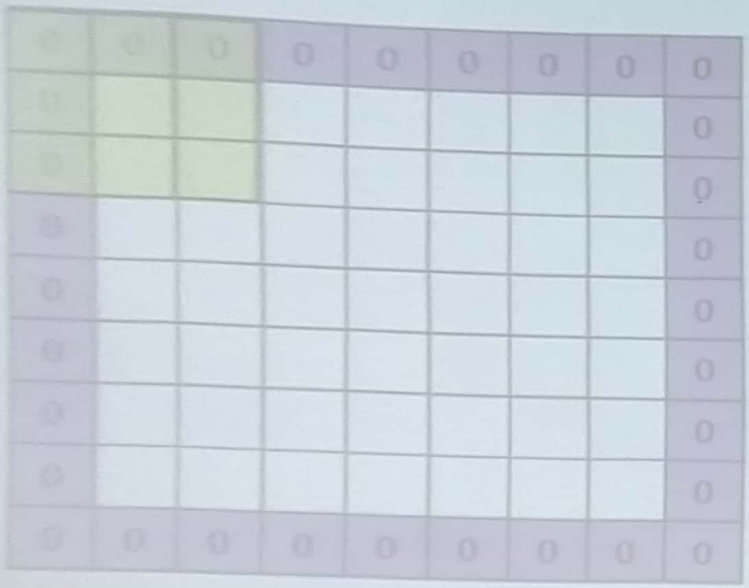
e.g., input 7x7
3x3 filter applied **with stride 1**
pad with 1 pixel border

What is the output size?

Size = 7x7

Note: output Size:
(N-F)/stride + 1

Common to Zero-pad the border in practice

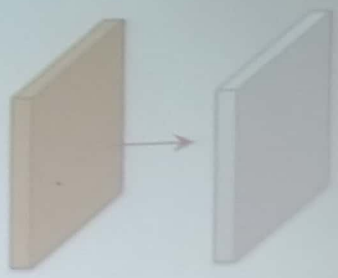


Output Size = 7x7

In general, it is common to have conv layers with **stride 1**, **filter size FxF**, and **zero padding (F-1)/2**, preserving spatial size

- F=3 → zero pad with 1
- F=5 → zero pad with 2
- F=7 → zero pad with 3

Example:



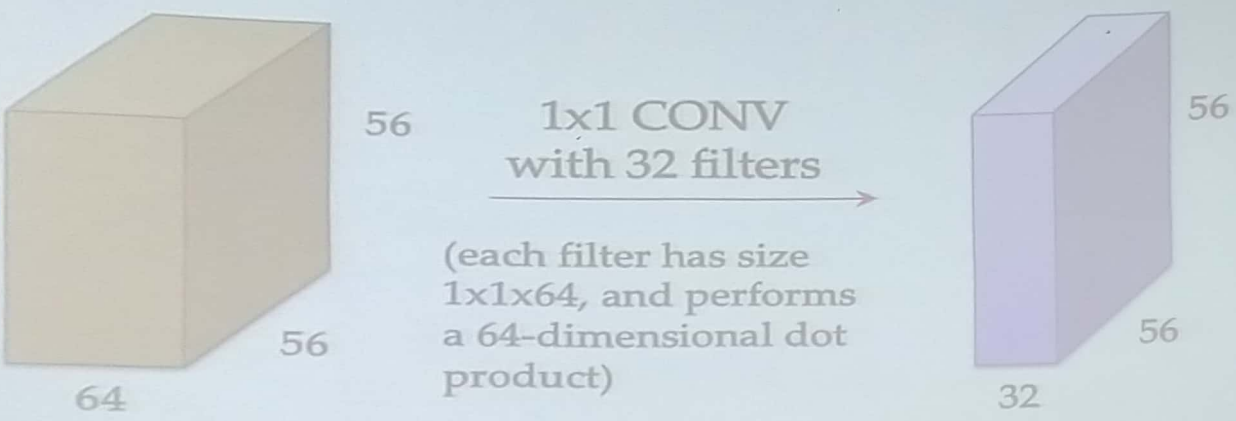
- Input Volume: **32 x 32 x 3**
- 10 **5x5** filters with stride 1, pad 2

- Number of parameters in this layer?

each filter has 5*5*3 + 1 = **76 params** (1 for bias)

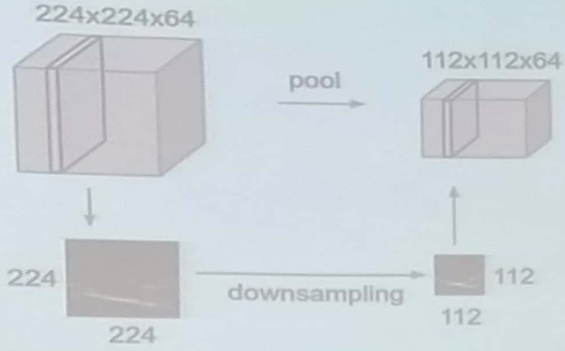
→ **76*10 = 760 parameters in the layer**

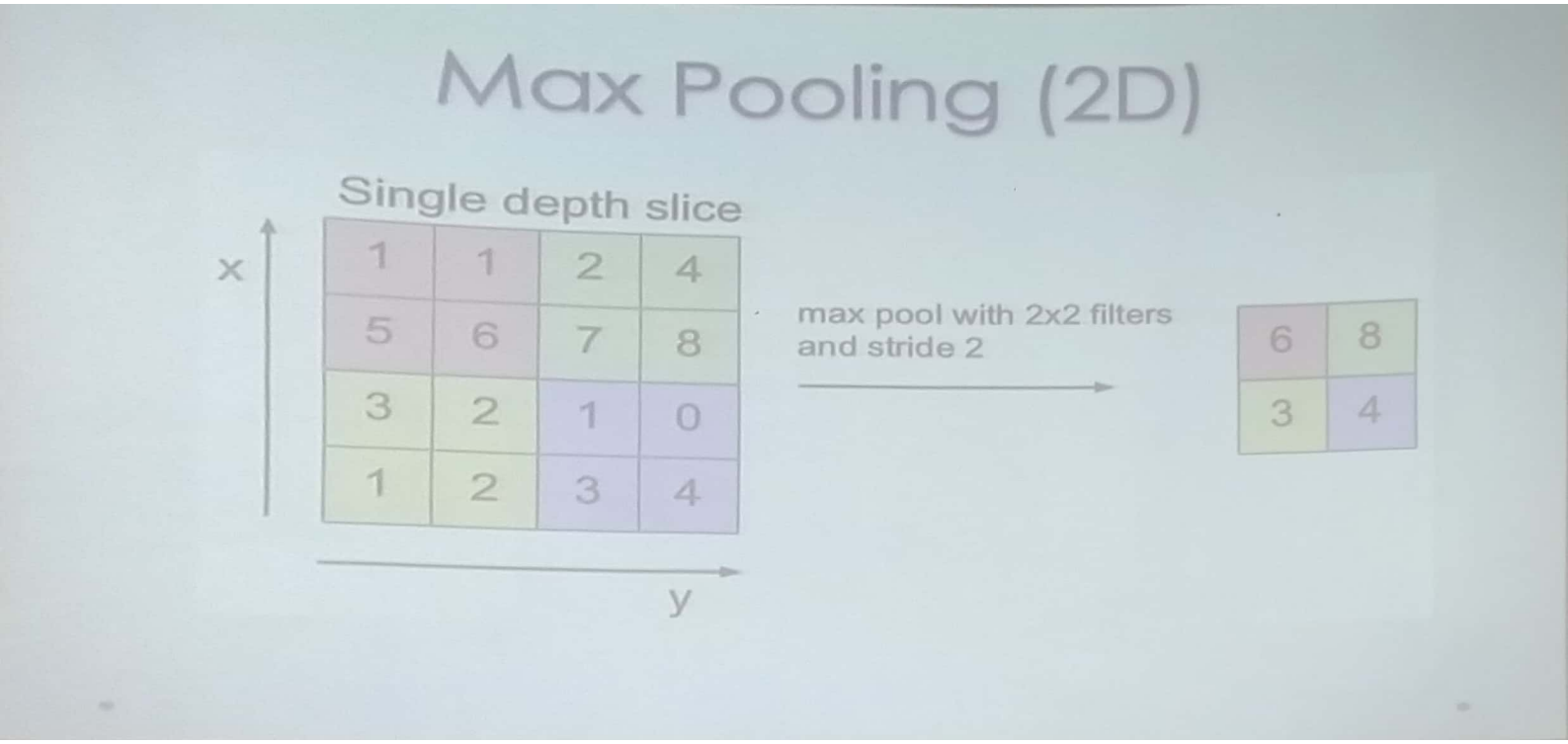
Note: 1x1 convolutions are perfectly fine



Pooling Layer

- Makes representations smaller and manageable
- Later filters have larger support
- Operates over each activation map independently





Summary

- CNNs are a series of CONV, ReLU, Pool, FC layers
- CNNs are computationally efficient and compact
- Parallels to human/animal visual system.
- Learnt features can be used for classification
- Recent Trends:
  - Stick with 3x3 filters, make the network deeper
  - Improve connectivity
  - Several innovations for specific applications