

AI and Deep Learning

CNN, Convolutional Neural Network

Jeju National University

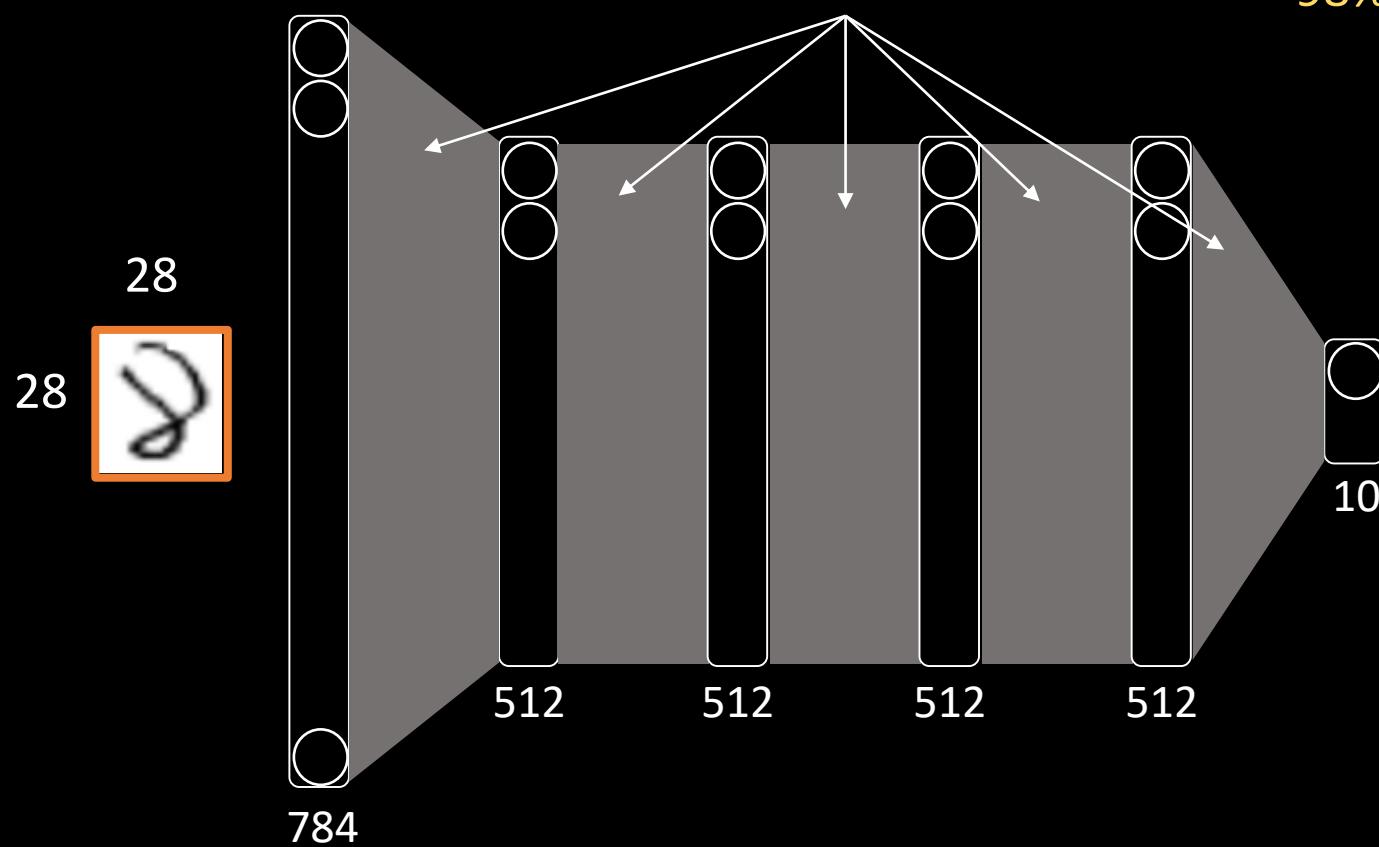
Yungcheol Byun

Agenda

- Drawbacks of FCNN
- Feature and Filter
- Activation Maps
- Pooling
- Theoretical Background
- Case Study
- RNN

Drawbacks of Fully Connected NN

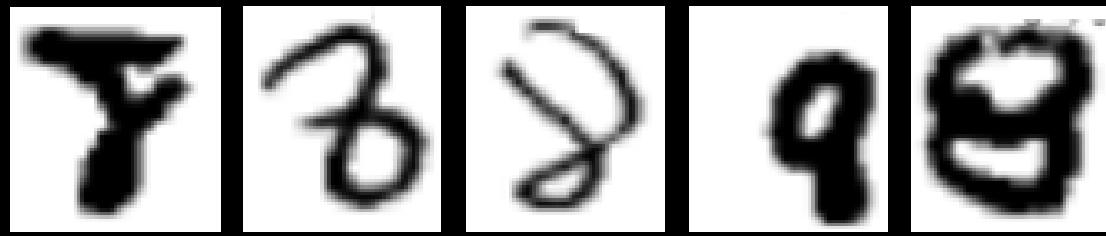
23_mnist_nn_deep.py
98%





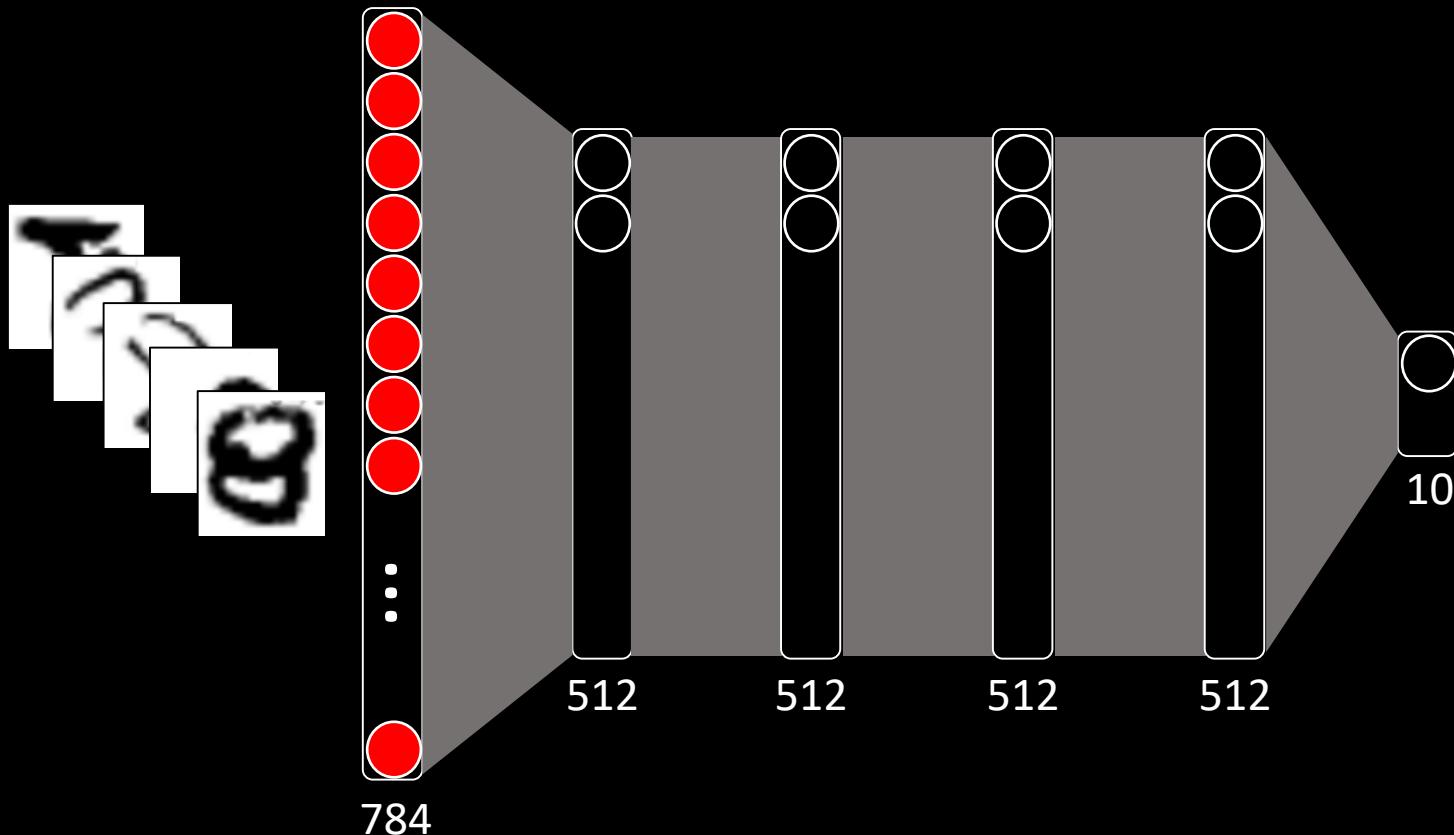
- 784(28x28) pixel values as inputs
- Various *shapes* - size, location, skewness, distortion





Many variations from writing styles

Many different Input values for the same digit

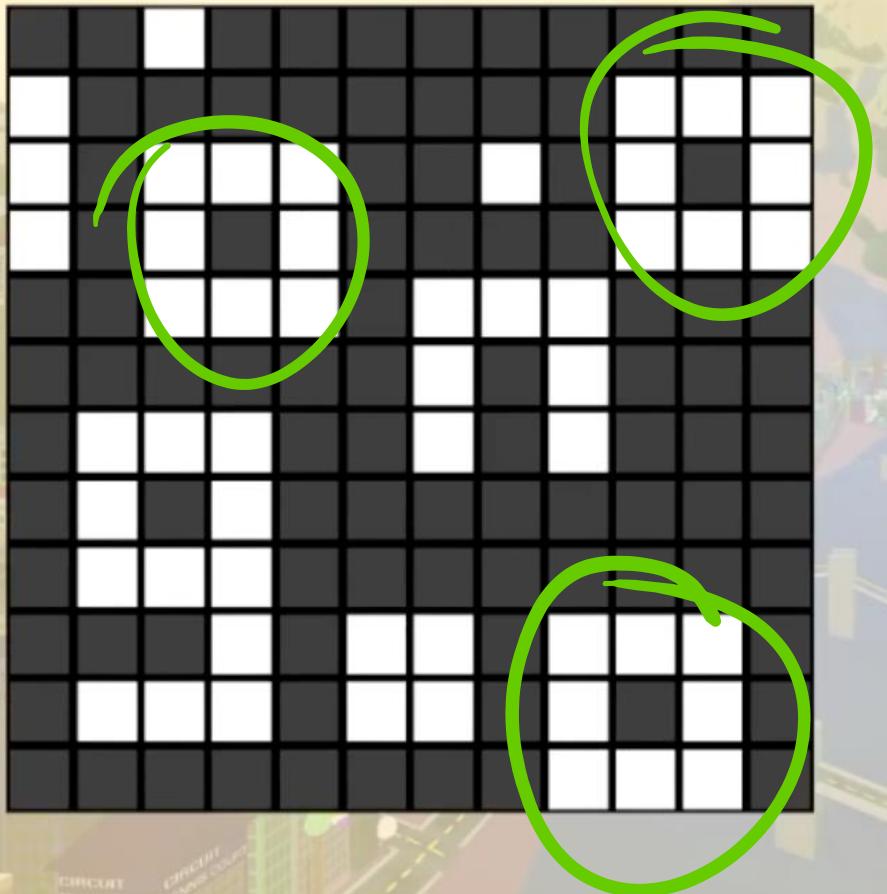


Any common features?



Therefore,
not the raw image but
feature extraction from it

How to extract doughnut features



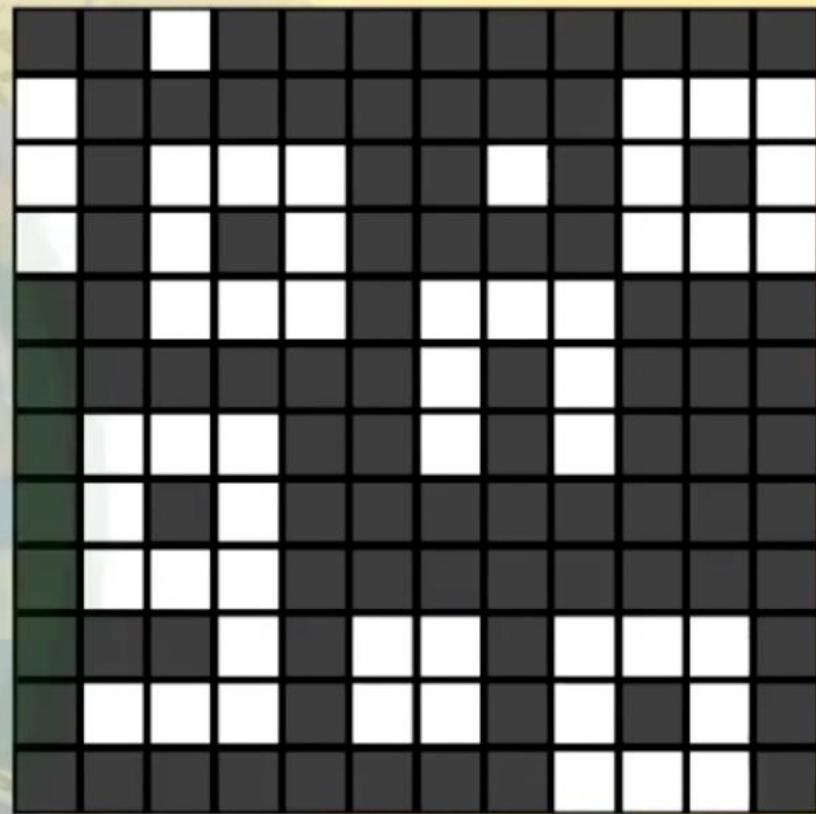
IMAGE



How to extract doughnut features

"DONUT FILTER"

| | | | | |
|----------------|---------------|---------------|---------------|----------------|
| DOESN'T MATTER | MUST BE BLACK | MUST BE BLACK | MUST BE BLACK | DOESN'T MATTER |
| MUST BE BLACK | MUST BE WHITE | MUST BE WHITE | MUST BE WHITE | MUST BE BLACK |
| MUST BE BLACK | MUST BE WHITE | MUST BE BLACK | MUST BE WHITE | MUST BE BLACK |
| MUST BE BLACK | MUST BE WHITE | MUST BE WHITE | MUST BE WHITE | MUST BE BLACK |
| DOESN'T MATTER | MUST BE BLACK | MUST BE BLACK | MUST BE BLACK | DOESN'T MATTER |



How to extract doughnut features

"DONUT FILTER"

upper-left pixel

| | | | | |
|----------------|---------------|---|---------------|----------------|
| DOESN'T MATTER | MUST BE BLACK | MUST BE BLACK | MUST BE BLACK | DOESN'T MATTER |
| MUST BE BLACK | MUST BE WHITE | MUST BE WHITE | MUST BE WHITE | MUST BE BLACK |
| MUST BE BLACK | MUST BE WHITE |  MUST BE WHITE | MUST BE WHITE | MUST BE BLACK |
| MUST BE BLACK | MUST BE WHITE | MUST BE WHITE | MUST BE WHITE | MUST BE BLACK |
| DOESN'T MATTER | MUST BE BLACK | MUST BE BLACK | MUST BE BLACK | DOESN'T MATTER |

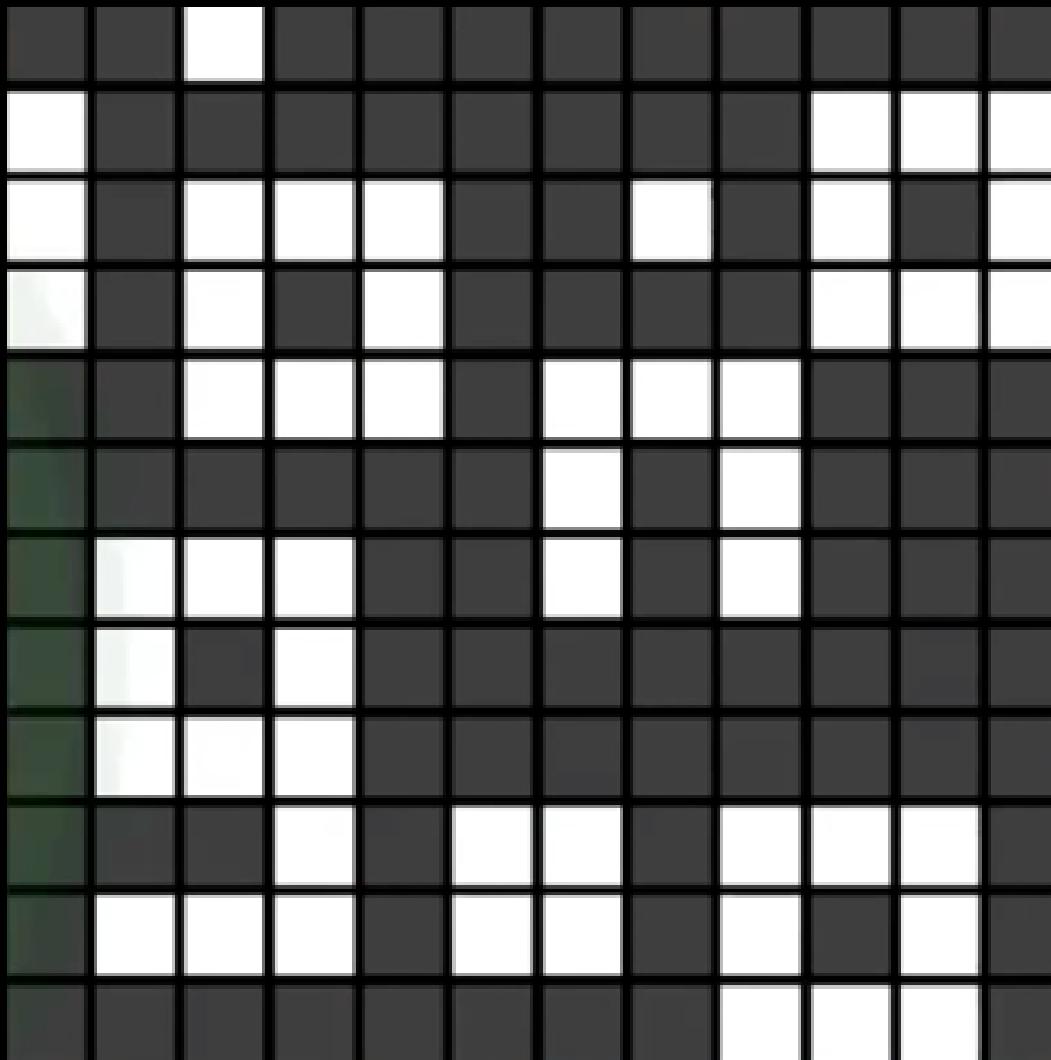
Out-of-bounds pixels are considered as black pixels.

"Is every condition of the filter satisfied?"

It's not a donut.

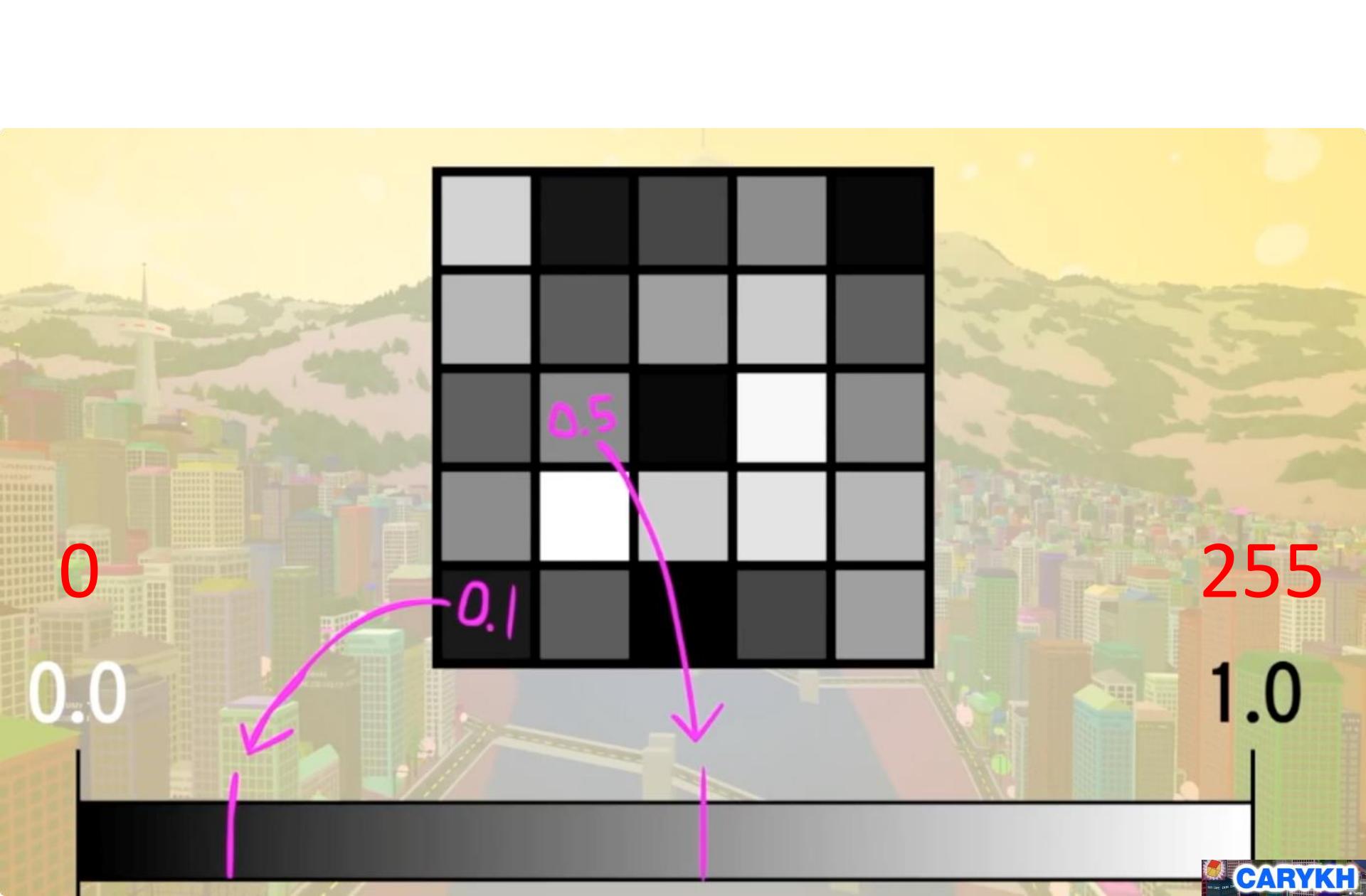


Black & White



Real images are color or gray





DONUT FILTER

| | | | | |
|----------------|---------------|---------------|---------------|----------------|
| DOESN'T MATTER | MUST BE BLACK | MUST BE BLACK | MUST BE BLACK | DOESN'T MATTER |
| MUST BE BLACK | MUST BE WHITE | MUST BE WHITE | MUST BE WHITE | MUST BE BLACK |
| MUST BE BLACK | MUST BE WHITE | MUST BE BLACK | MUST BE WHITE | MUST BE BLACK |
| MUST BE BLACK | MUST BE WHITE | MUST BE WHITE | MUST BE WHITE | MUST BE BLACK |
| DOESN'T MATTER | MUST BE BLACK | MUST BE BLACK | MUST BE BLACK | DOESN'T MATTER |

New improved filter

| | | | | |
|-------|-------|-------|-------|-------|
| x0.0 | x-0.2 | x-0.1 | x-0.2 | x0.0 |
| x-0.2 | x0.7 | x1.0 | x0.7 | x-0.2 |
| x-0.1 | x1.0 | x-2.5 | x1.0 | x-0.1 |
| x-0.2 | x0.7 | x1.0 | x0.7 | x-0.2 |
| x0.0 | x-0.1 | x-0.1 | x-0.1 | x0.0 |

"set of multipliers"

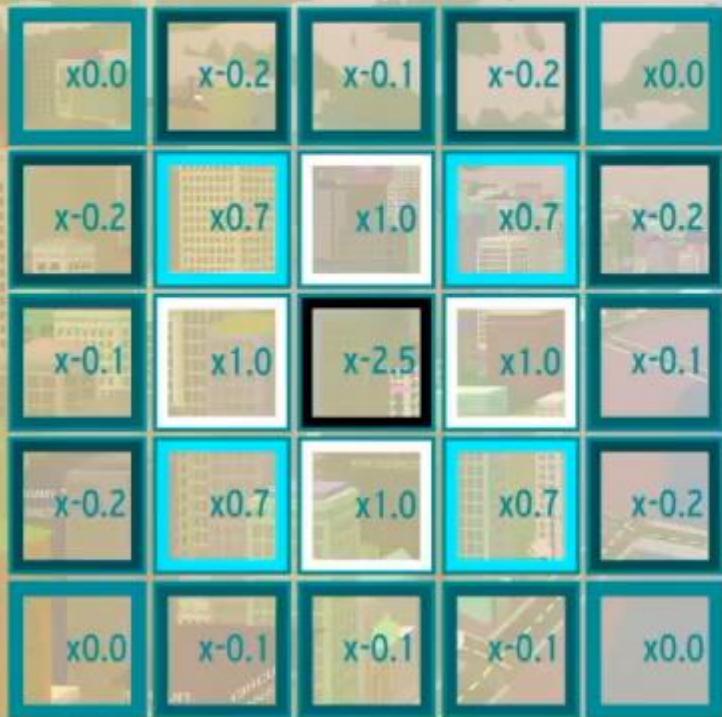
"If you want to
be considered
donutty..."



| | | | |
|-------|-------|-------|-------|
| x-0.2 | x-0.1 | x-0.2 | x0.0 |
| x0.7 | x1.0 | x0.7 | x-0.2 |
| x1.0 | x-2.5 | x1.0 | x-0.1 |
| x-0.2 | x0.7 | x1.0 | x0.7 |
| x0.0 | x-0.1 | x-0.1 | x0.0 |

...you'd better
have a high
value for this
pixel."

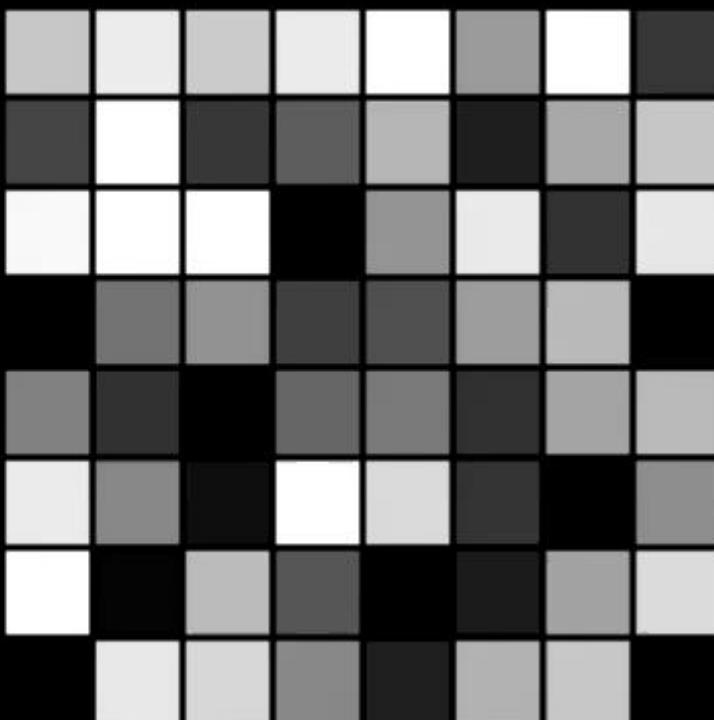
THE FILTER ↴



| | | | | | | | |
|------|------|------|------|------|------|------|------|
| 0.73 | 0.86 | 0.74 | 0.85 | 0.99 | 0.58 | 0.96 | 0.25 |
| 0.29 | 0.92 | 0.25 | 0.38 | 0.67 | 0.16 | 0.63 | 0.73 |
| 0.90 | 0.99 | 0.95 | 0.06 | 0.56 | 0.85 | 0.23 | 0.84 |
| 0.00 | 0.45 | 0.55 | 0.28 | 0.33 | 0.59 | 0.69 | 0.01 |
| 0.50 | 0.23 | 0.03 | 0.11 | 0.48 | 0.23 | 0.62 | 0.69 |
| 0.85 | 0.52 | 0.11 | 0.95 | 0.80 | 0.24 | 0.03 | 0.54 |
| 0.99 | 0.07 | 0.69 | 0.35 | 0.05 | 0.15 | 0.61 | 0.80 |
| 0.03 | 0.84 | 0.79 | 0.52 | 0.17 | 0.66 | 0.73 | 0.01 |

| | | | | | | | | | |
|------|--------|----------------|----------------|----------------|------|------|------|------|------|
| 0.00 | x-0.20 | x-0.10 | x-0.20 | x0.00 | | | | | |
| 0.20 | x0.70 | x1.00 | x0.70 | x-0.20 | | | | | |
| 0.10 | x1.00 | 0.73 x-2.50 | 0.86 x1.00 | 0.74 x-0.10 | 0.85 | 0.99 | 0.58 | 0.96 | 0.25 |
| 0.20 | x0.70 | 0.29 x1.00 | 0.92 x0.70 | 0.25 x-0.20 | 0.38 | 0.67 | 0.16 | 0.63 | 0.73 |
| 0.00 | x-0.20 | 0.90 x-0.10 | 0.99 x-0.20 | 0.95 x0.00 | 0.06 | 0.56 | 0.85 | 0.23 | 0.84 |
| | 0.00 | 0.45 | 0.55 | 0.28 | 0.33 | 0.59 | 0.69 | 0.01 | |
| | 0.50 | 0.23 | 0.03 | 0.41 | 0.48 | 0.23 | 0.62 | 0.69 | |
| | 0.85 | 0.52 | 0.11 | 0.95 | 0.80 | 0.24 | 0.03 | 0.54 | |
| | 0.99 | 0.07 | 0.69 | 0.35 | 0.05 | 0.15 | 0.61 | 0.80 | |
| | 0.03 | 0.84 | 0.79 | 0.52 | 0.17 | 0.66 | 0.73 | 0.01 | |

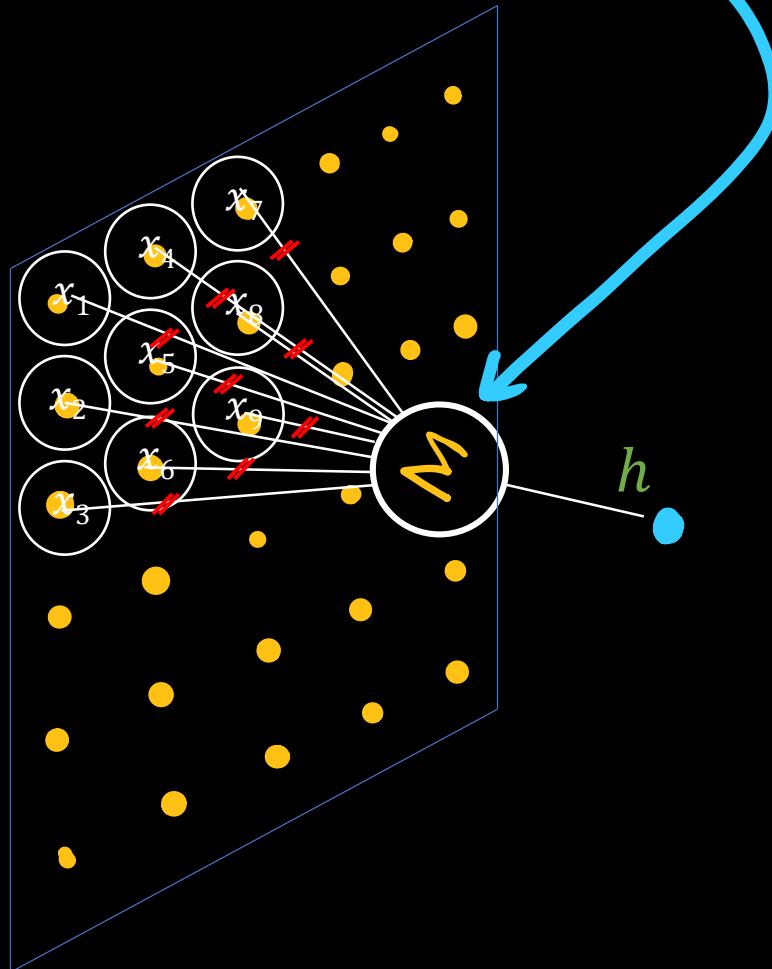
How donutty is each pixel?



Convolution?

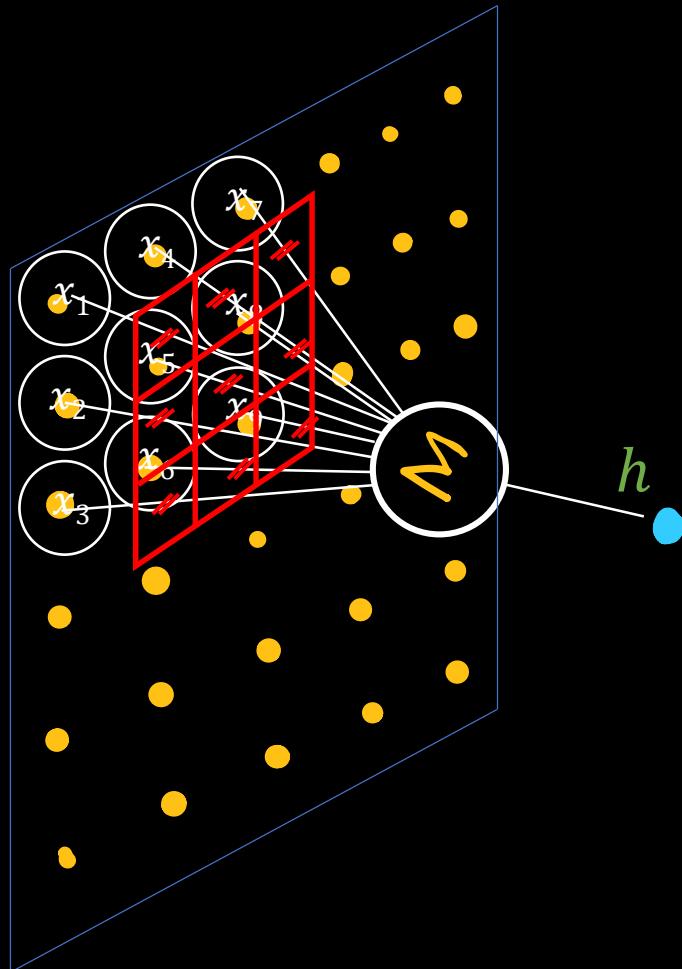
dot product over the image
and slide(move) filter spatially to the next pixel.

A neuron

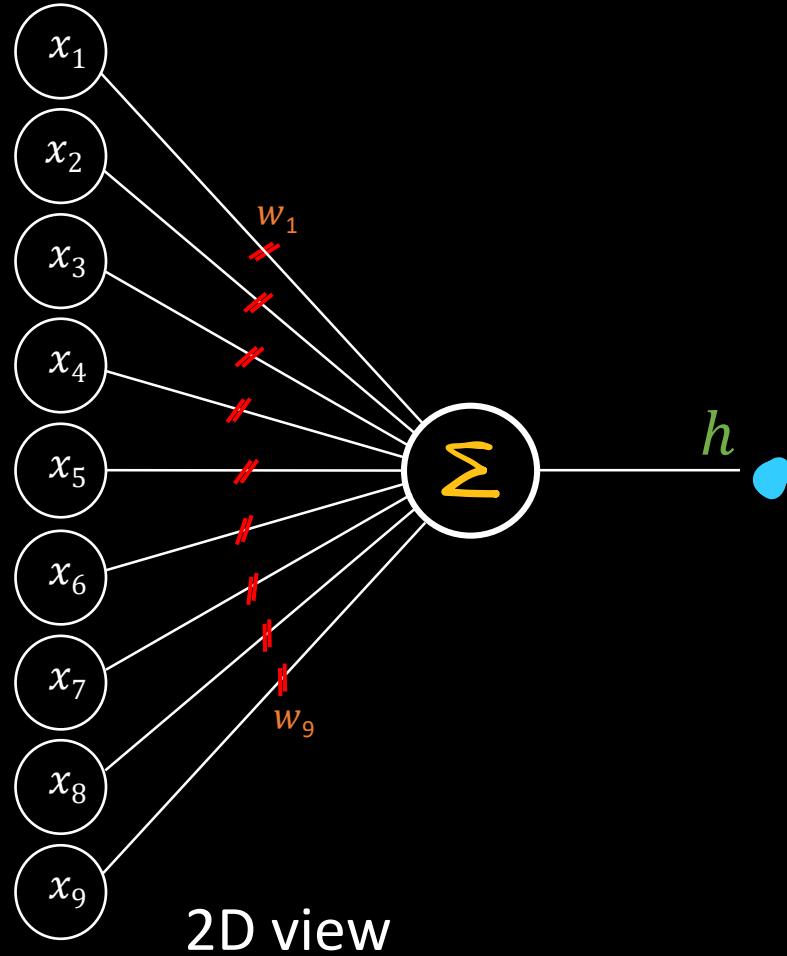


9 connections = 9 synaps (parameters)

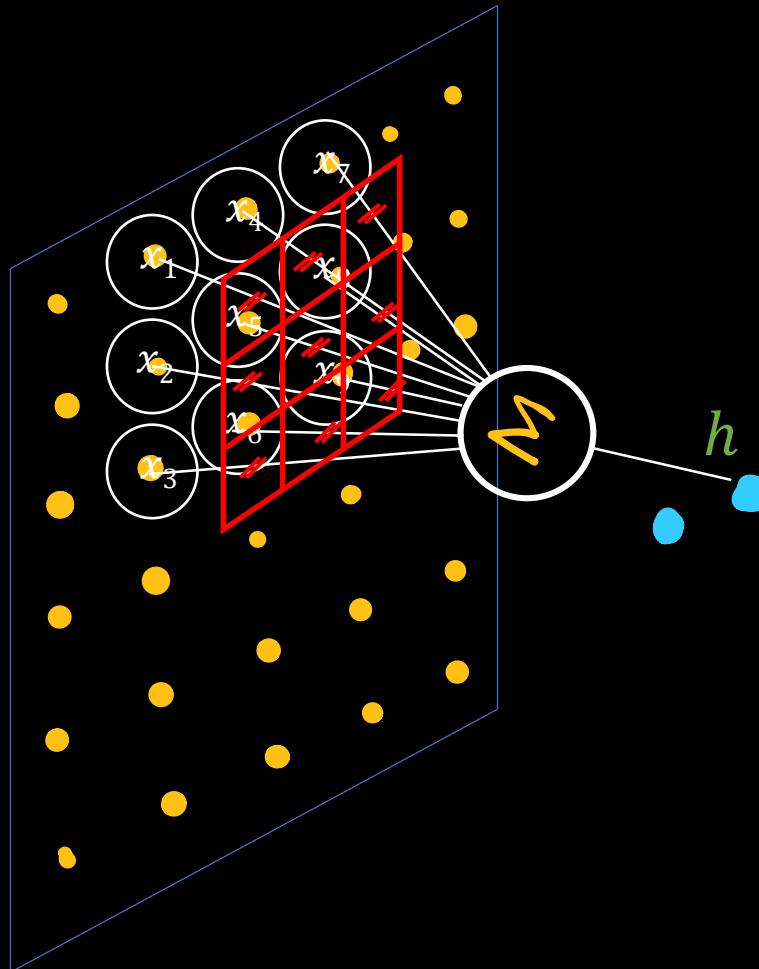
What's the filter?



9 connections = 9 synaps (parameters)

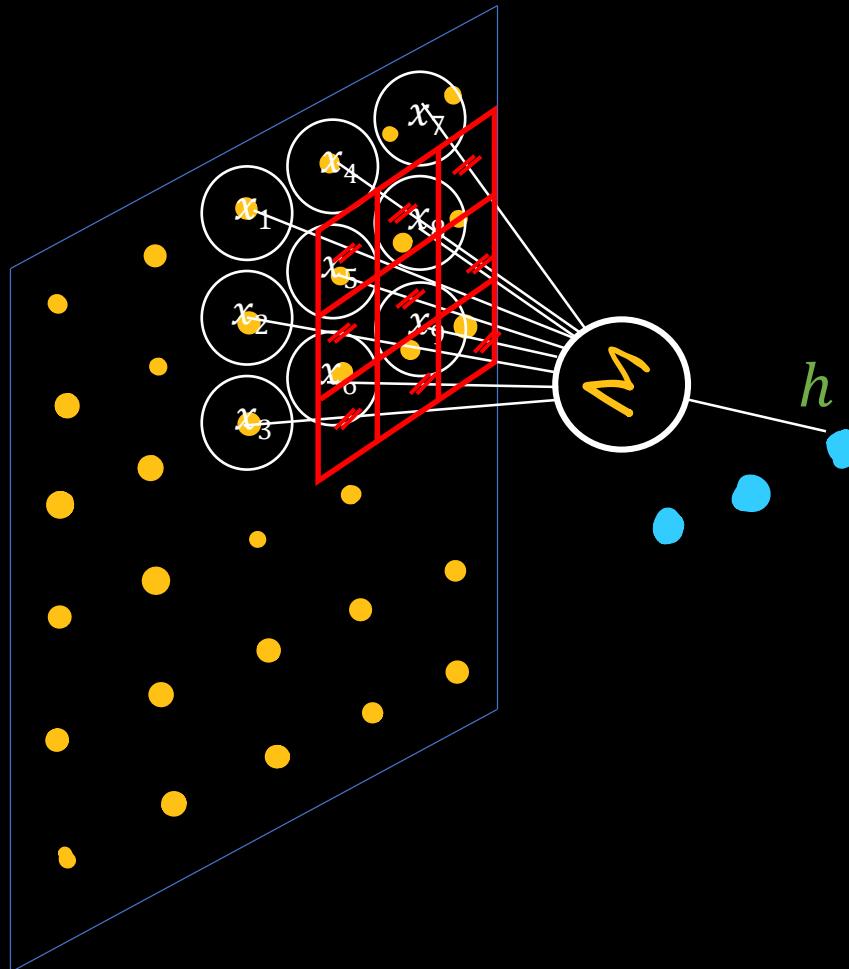


What's the filter?



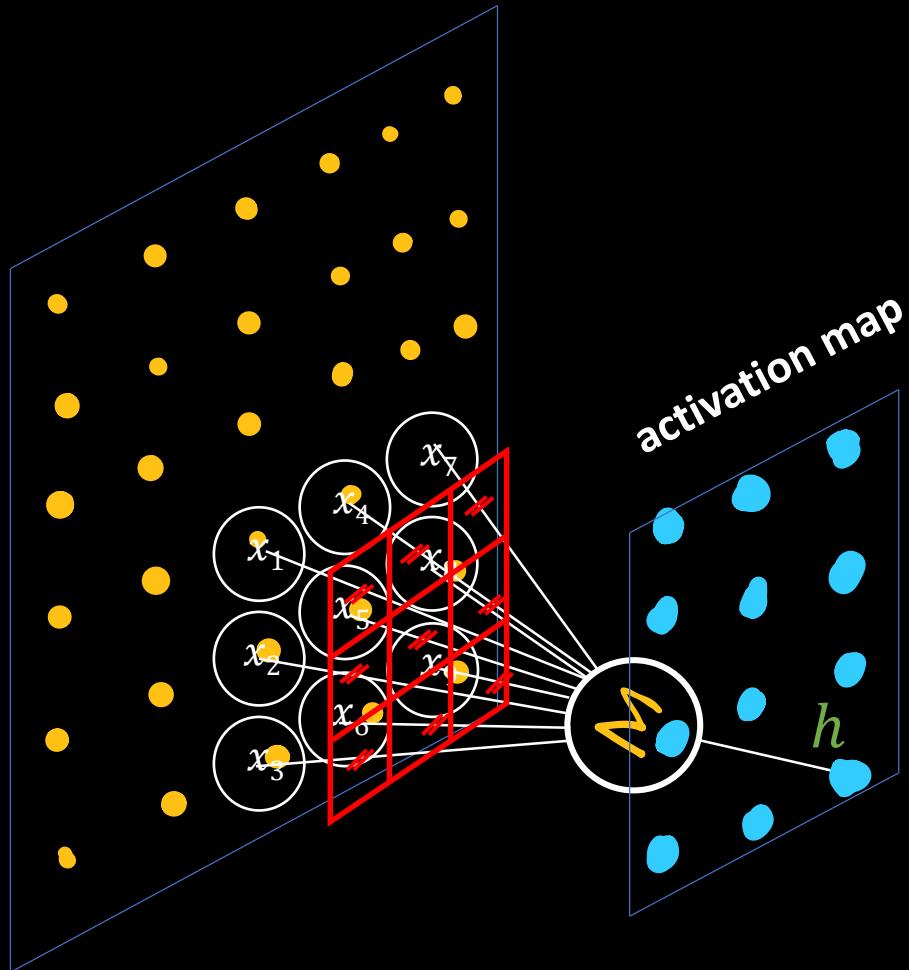
9 connections = 9 synaps (parameters)

What's the filter?



9 connections = 9 synaps (parameters)

What's the filter?



9 connections = 9 synaps (parameters)

A filter is a group of synapses
shared while shifting.

| | | | |
|-------|-------|-------|------|
| x1.0 | x1.5 | x1.0 | x0.4 |
| x0.7 | x1.0 | x0.7 | x0.6 |
| x-0.4 | x-0.4 | x-0.4 | x0.0 |

| | | | | | | | |
|------|------|------|------|------|------|------|------|
| 0.73 | 0.86 | 0.74 | 0.85 | 0.99 | 0.58 | 0.96 | 0.25 |
| 0.29 | 0.92 | 0.25 | 0.38 | 0.67 | 0.16 | 0.63 | 0.73 |
| 0.90 | 0.99 | 0.95 | 0.06 | 0.56 | 0.85 | 0.23 | 0.84 |
| 0.00 | 0.45 | 0.55 | 0.28 | 0.33 | 0.59 | 0.59 | 0.01 |
| 0.50 | 0.23 | 0.03 | 0.41 | 0.49 | 0.23 | 0.62 | 0.69 |
| 0.85 | 0.52 | 0.11 | 0.95 | 0.80 | 0.24 | 0.03 | 0.54 |
| 0.99 | 0.07 | 0.69 | 0.35 | 0.05 | 0.15 | 0.61 | 0.80 |
| 0.03 | 0.84 | 0.79 | 0.52 | 0.17 | 0.66 | 0.73 | 0.01 |

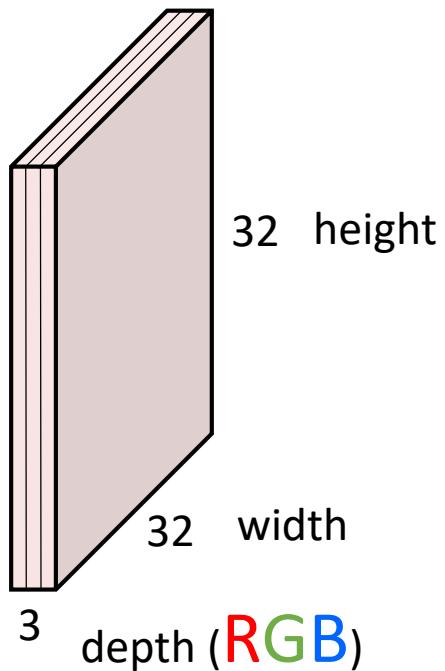
ALWAYS

COURT
TENNIS COURT



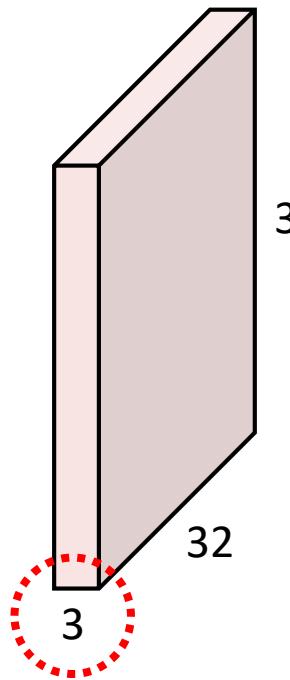
Convolution Layer

32x32x3 pixels image



Convolution Layer

32x32x3 image



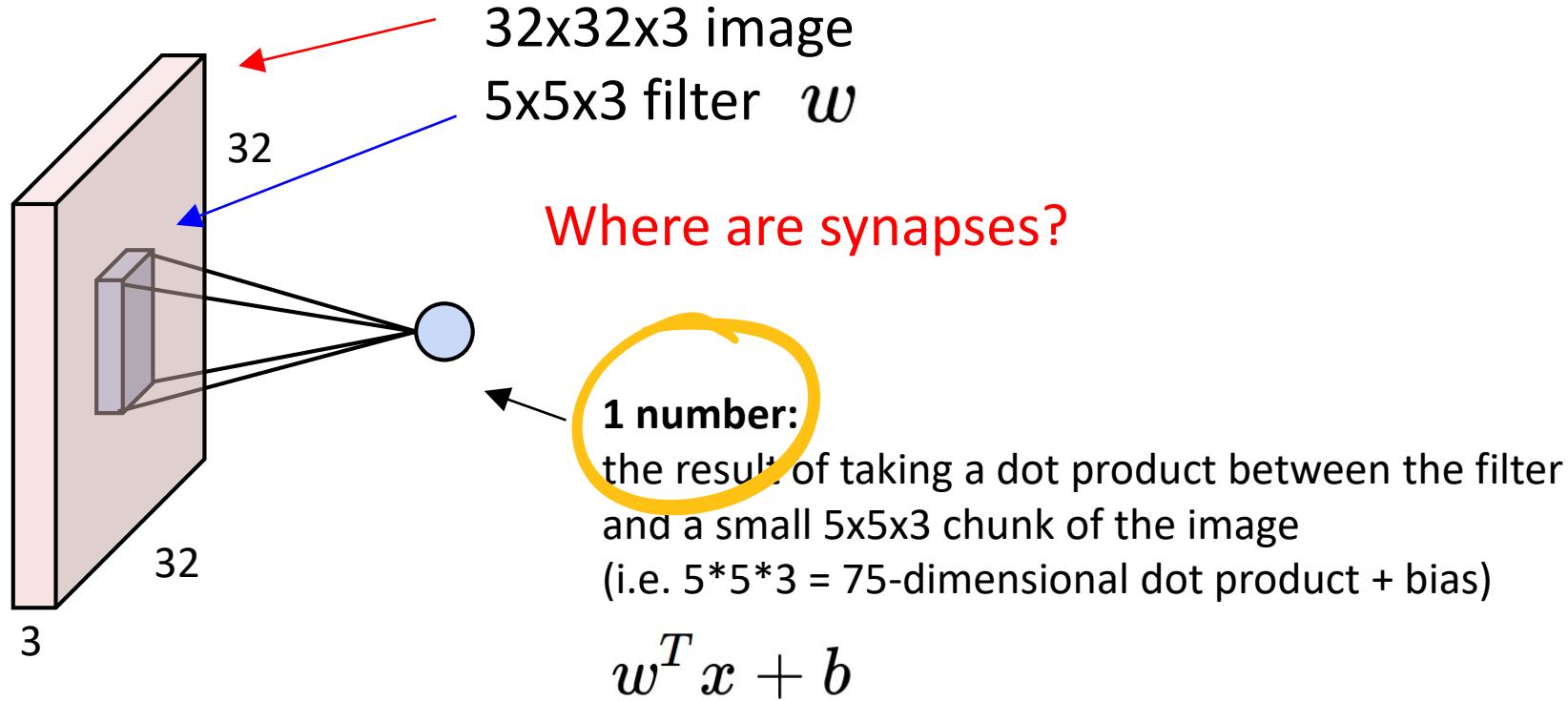
5x5x3 filter



Filters always extend the full depth of the input volume

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

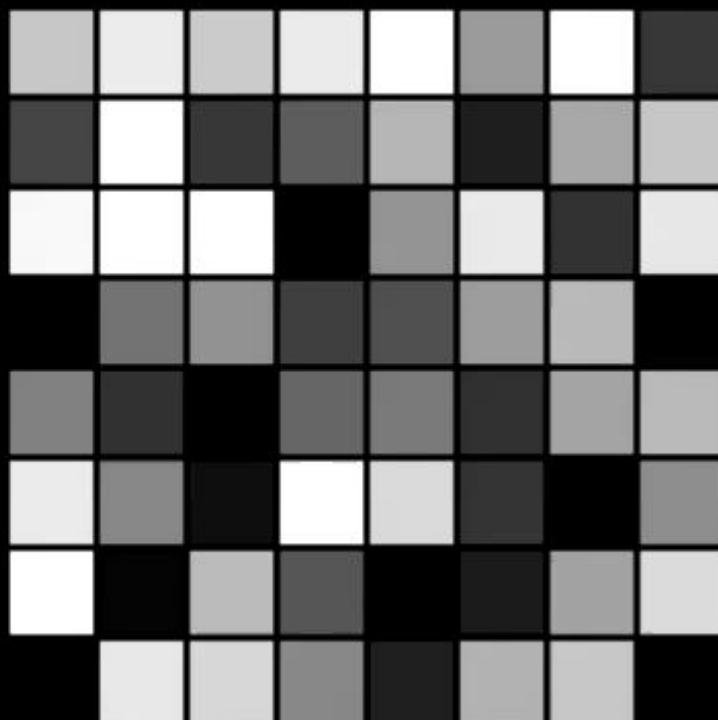
Convolution Layer



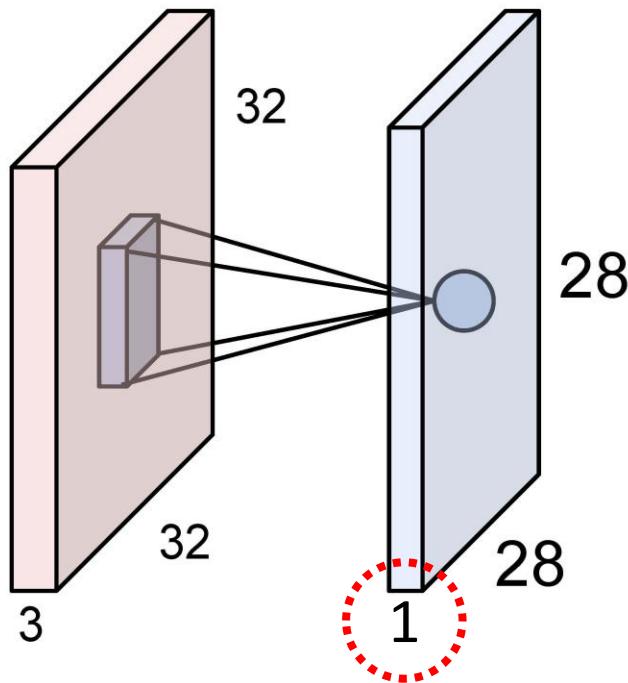
Where are synapses?

| | | | | | | | | | |
|------|--------|----------------|----------------|----------------|------|------|------|------|------|
| 0.00 | x-0.20 | x-0.10 | x-0.20 | x0.00 | | | | | |
| 0.20 | x0.70 | x1.00 | x0.70 | x-0.20 | | | | | |
| 0.10 | x1.00 | 0.73 x-2.50 | 0.86 x1.00 | 0.74 x-0.10 | 0.85 | 0.99 | 0.58 | 0.96 | 0.25 |
| 0.20 | x0.70 | 0.29 x1.00 | 0.92 x0.70 | 0.25 x-0.20 | 0.38 | 0.67 | 0.16 | 0.63 | 0.73 |
| 0.00 | x-0.20 | 0.90 x-0.10 | 0.99 x-0.20 | 0.95 x0.00 | 0.06 | 0.56 | 0.85 | 0.23 | 0.84 |
| | 0.00 | 0.45 | 0.55 | 0.28 | 0.33 | 0.59 | 0.69 | 0.01 | |
| | 0.50 | 0.23 | 0.03 | 0.41 | 0.48 | 0.23 | 0.62 | 0.69 | |
| | 0.85 | 0.52 | 0.11 | 0.95 | 0.80 | 0.24 | 0.03 | 0.54 | |
| | 0.99 | 0.07 | 0.69 | 0.35 | 0.05 | 0.15 | 0.61 | 0.80 | |
| | 0.03 | 0.84 | 0.79 | 0.52 | 0.17 | 0.66 | 0.73 | 0.01 | |

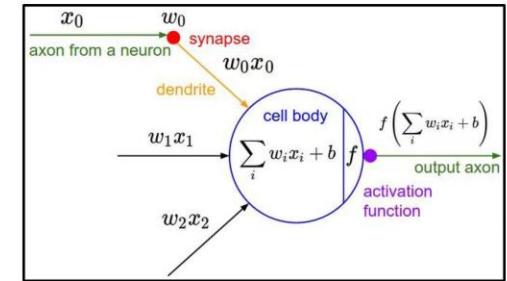
How donutty is each pixel?



The brain/neuron view of CONV Layer



Where are synapses?

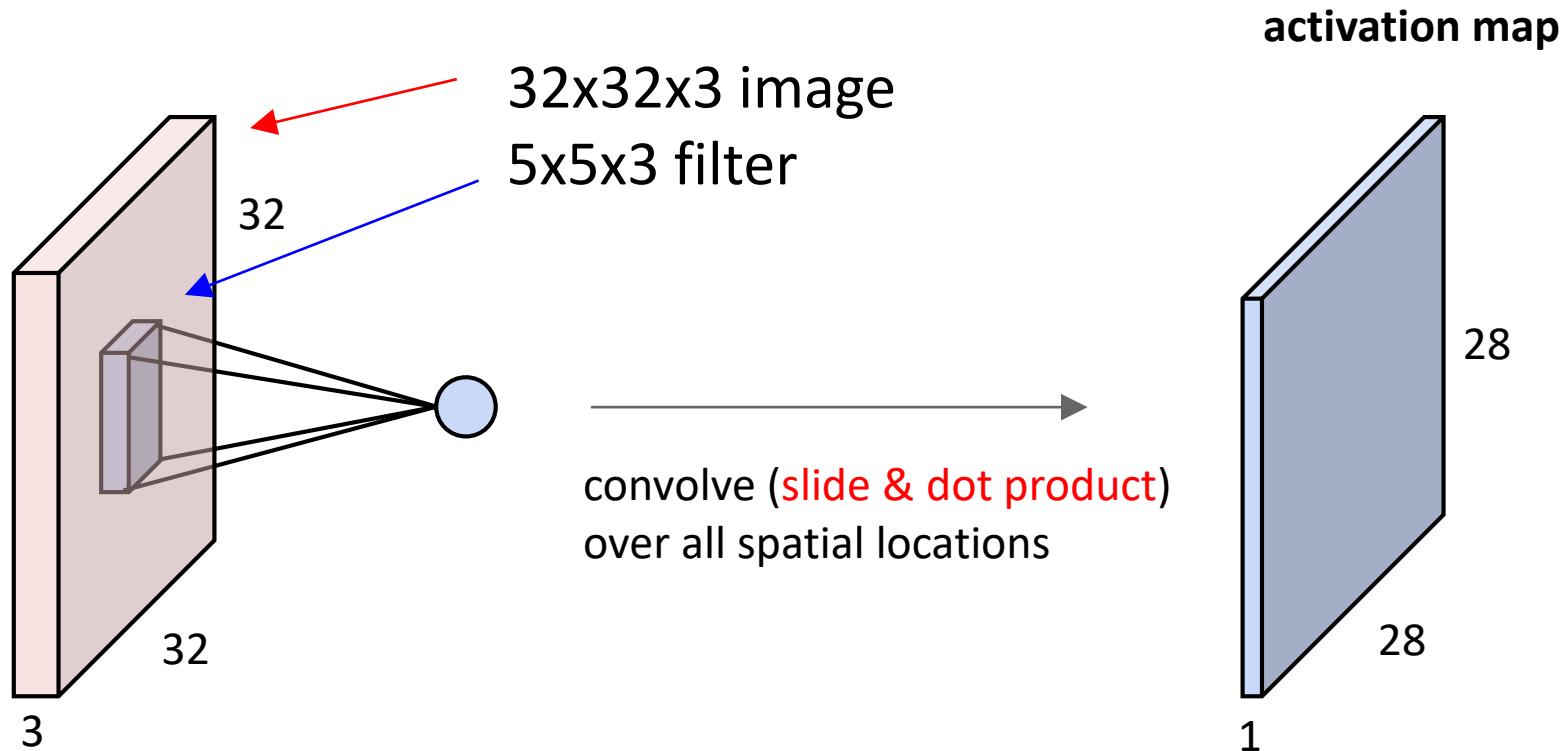


An activation map is a 28x28 sheet of neuron outputs:

1. Each is connected to a small region in the input
2. All of them share parameters

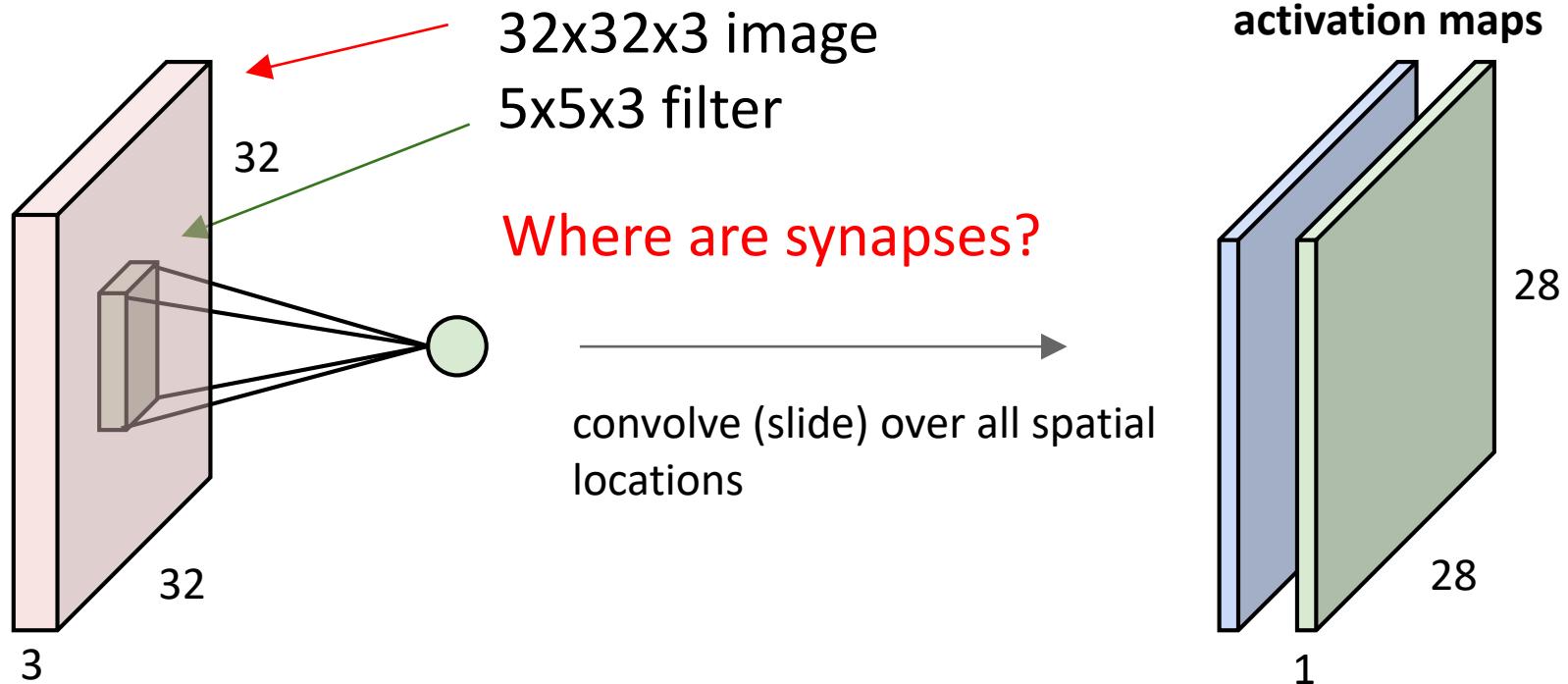
“5x5 filter” -> “5x5 receptive field for each neuron”

Convolution Layer

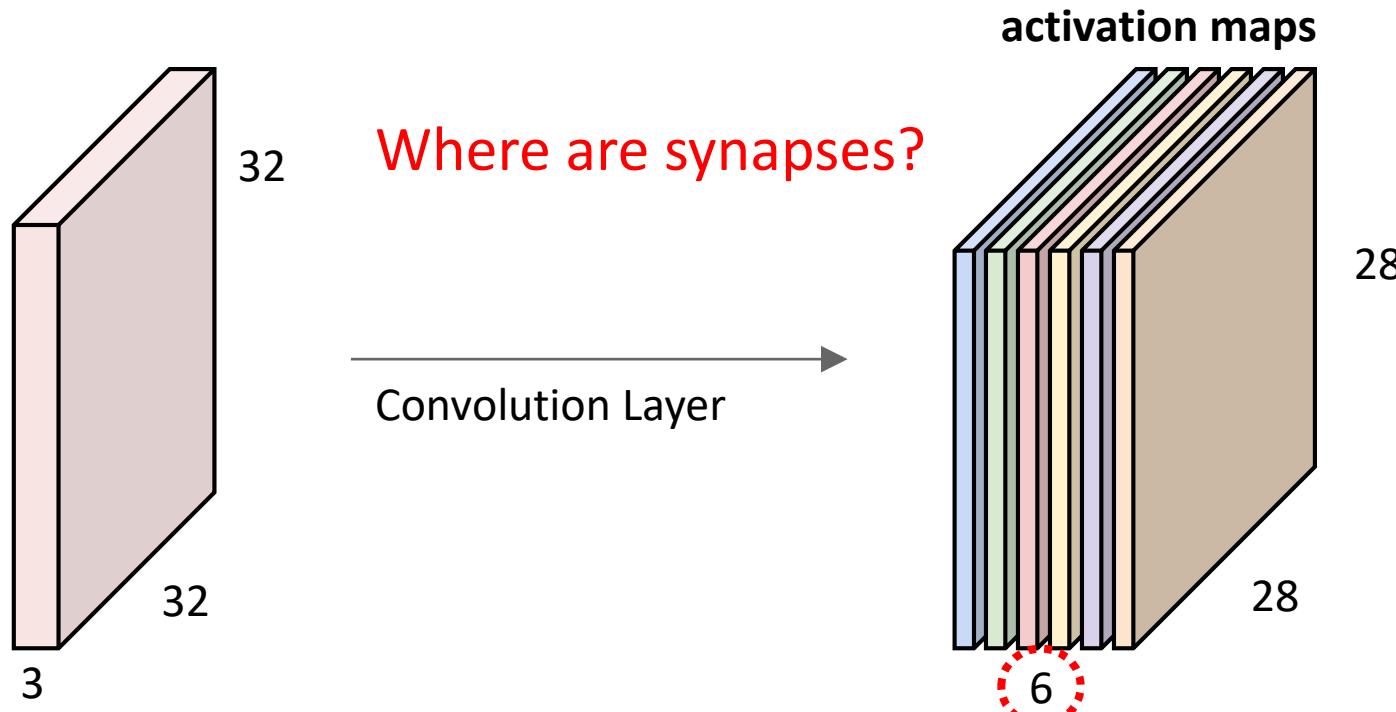


Convolution Layer

consider a second, green filter



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

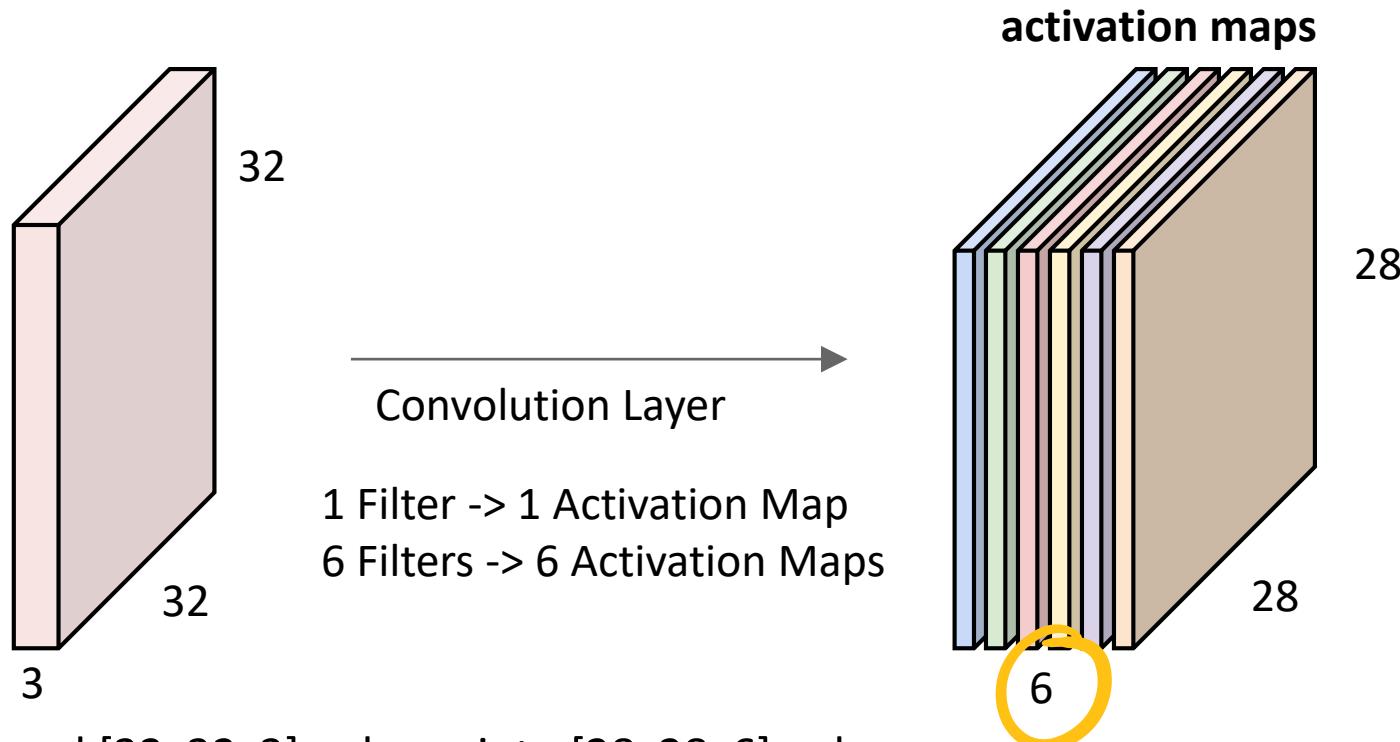


We processed [32x32x3] volume into [28x28x6] volume.

Q: how many parameters would this be if we used a fully connected layer instead?

A: $(32*32*3)*(28*28*6) = 14.5M$ parameters, ~14.5M multiplies

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

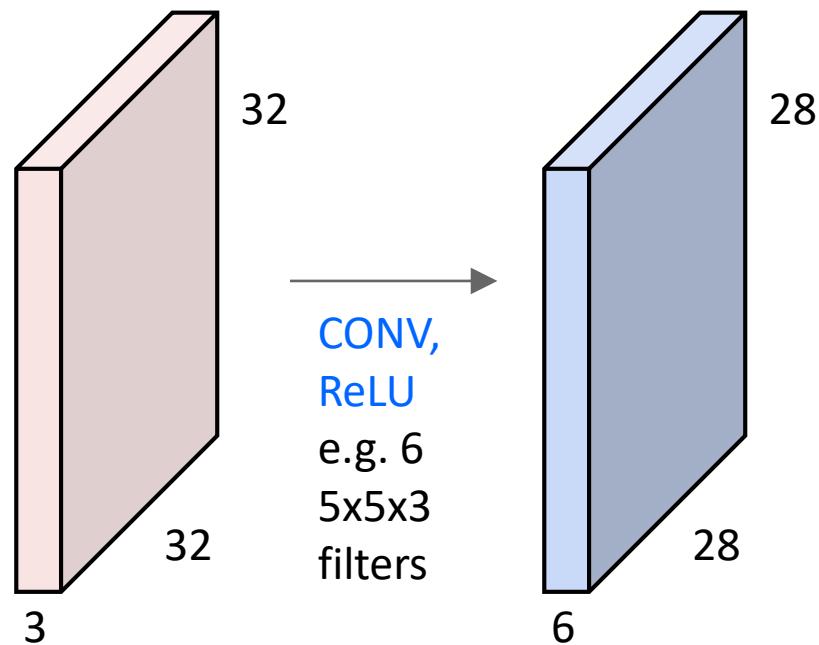


We processed [32x32x3] volume into [28x28x6] volume.

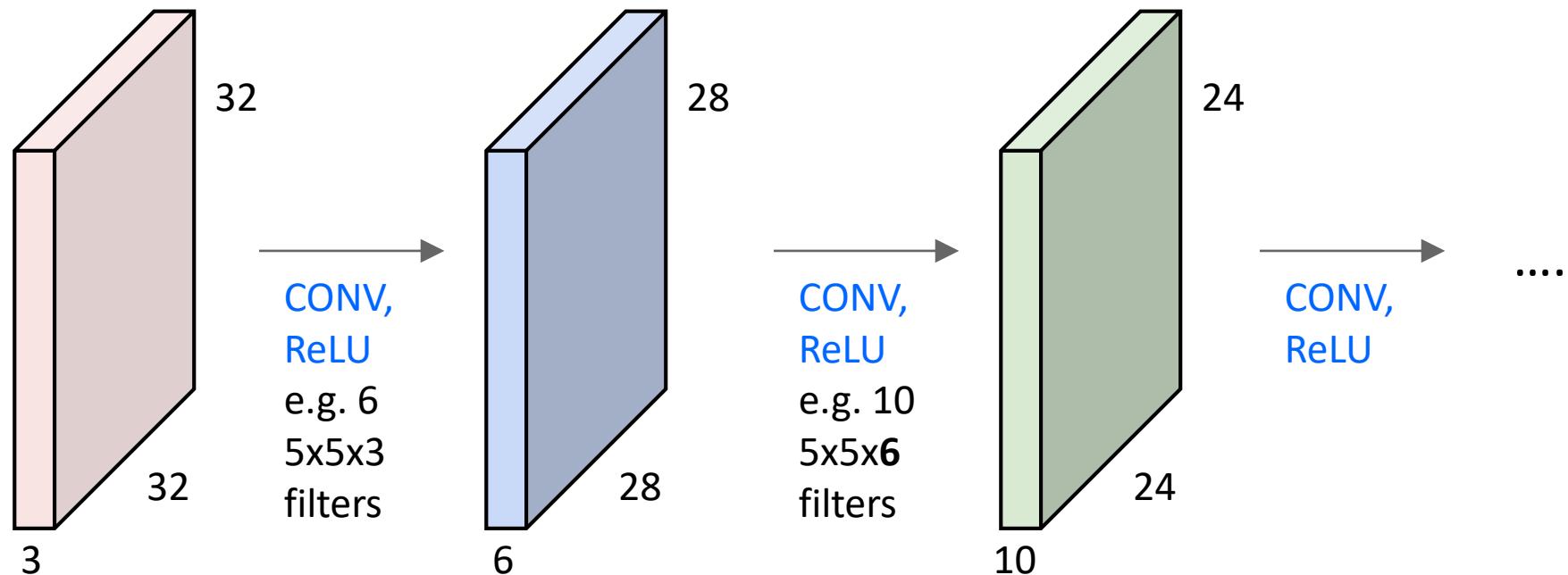
Q: how many parameters are used instead?

A: $(5*5*3)*6 = 450$ parameters, $(5*5*3)*(28*28*6) = \sim 350K$ multiplies

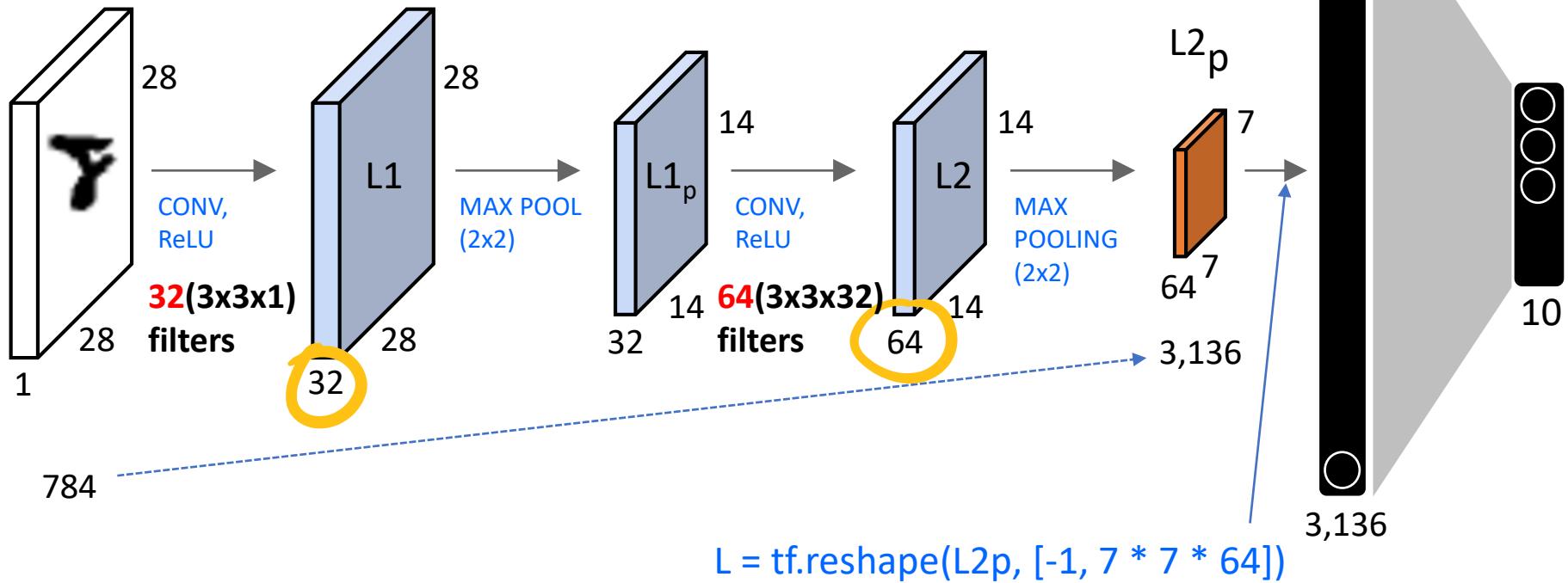
Preview: ConvNet is a sequence of Convolution Layers, interspersed with activation functions



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

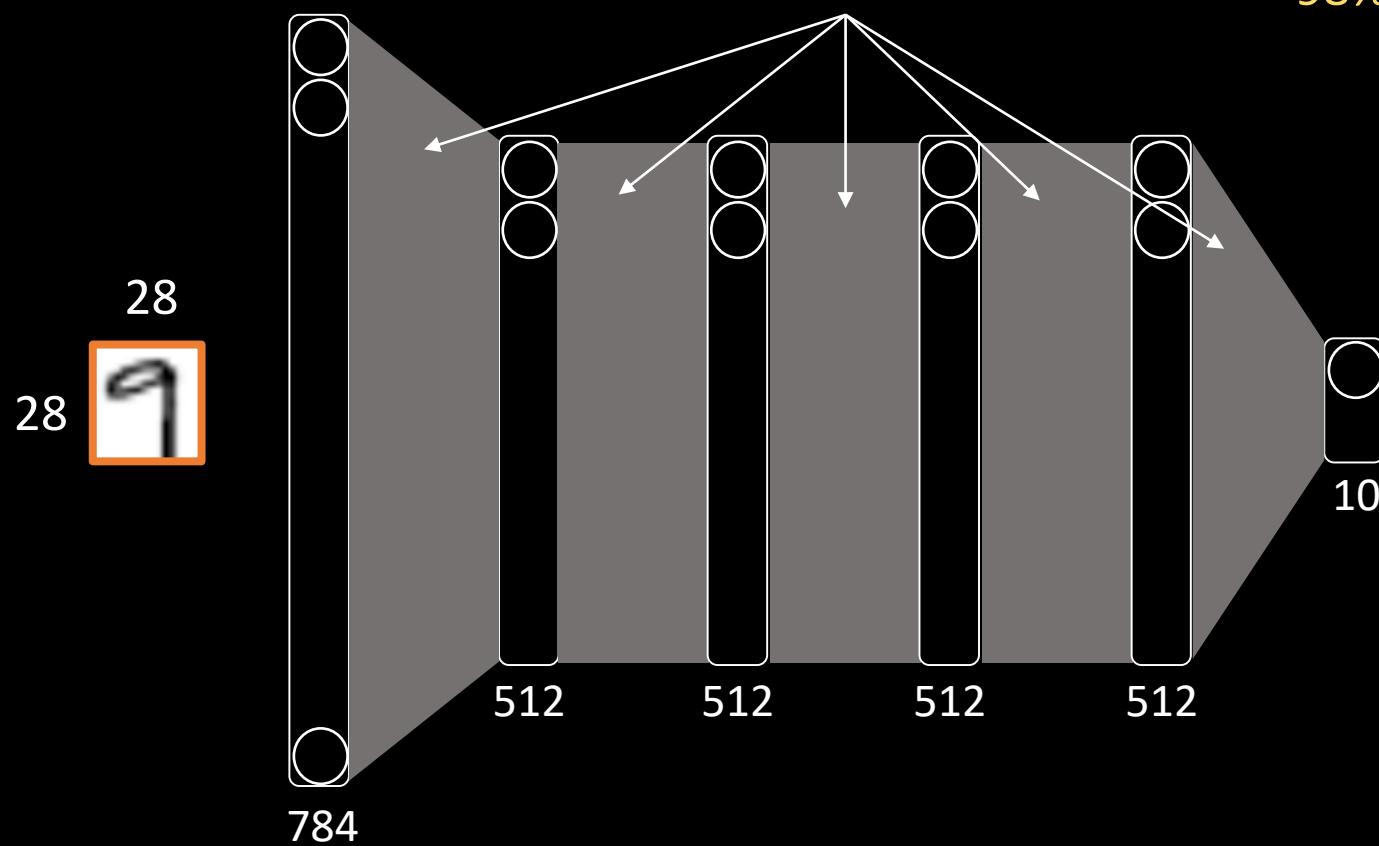


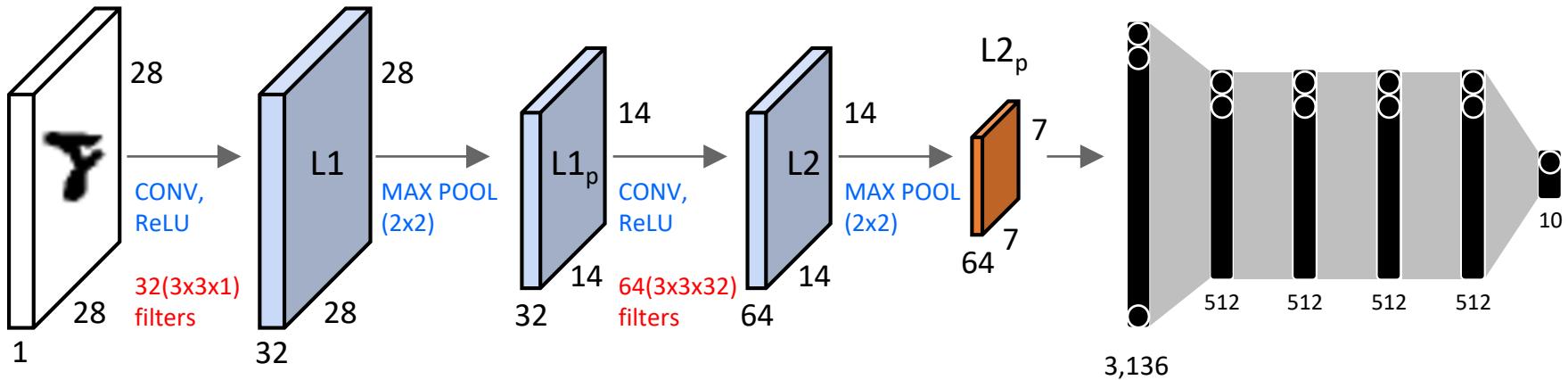
Accuracy: 99.38!!



Where are the features we try to extract?

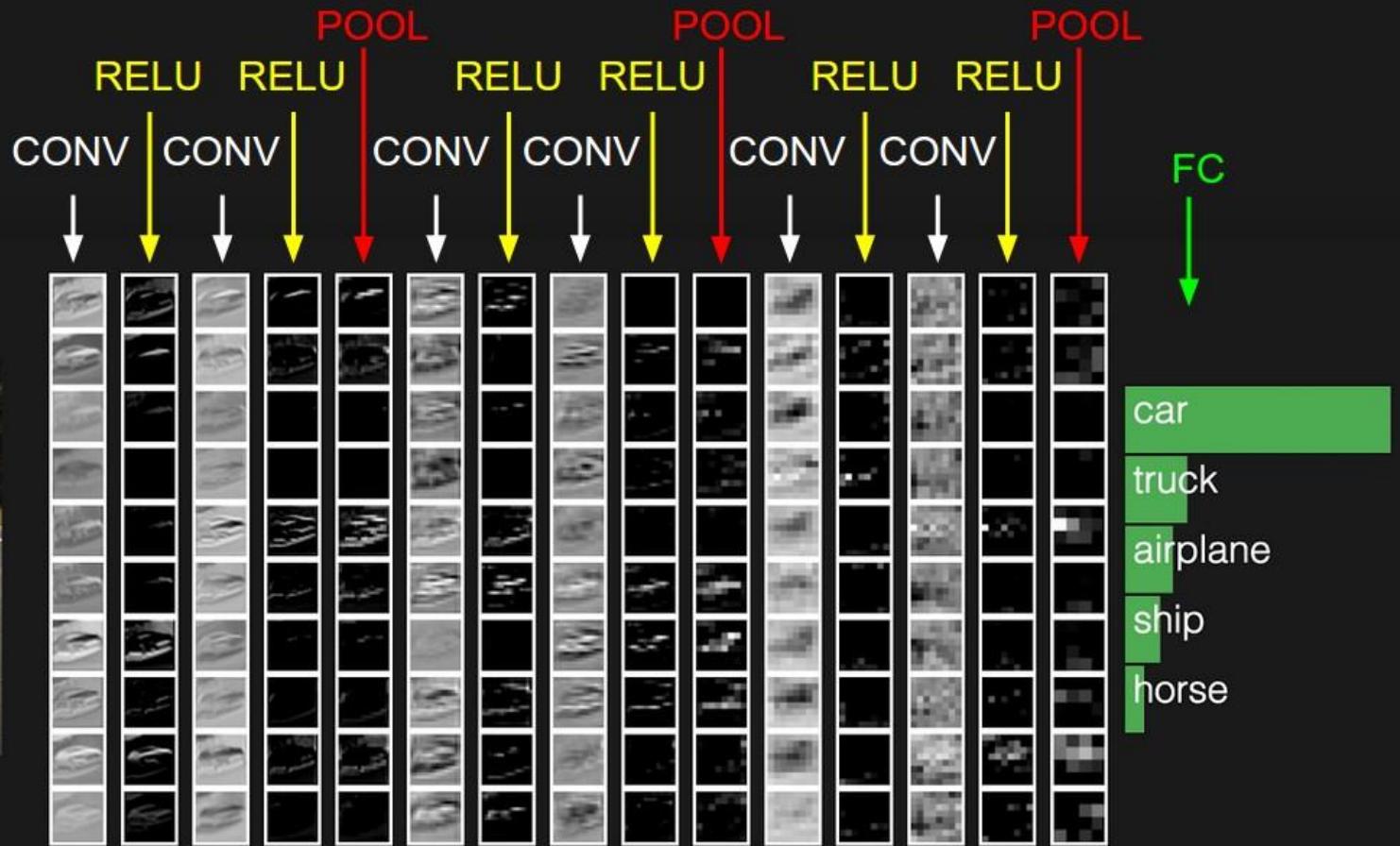
23_mnist_nn_deep.py
98%





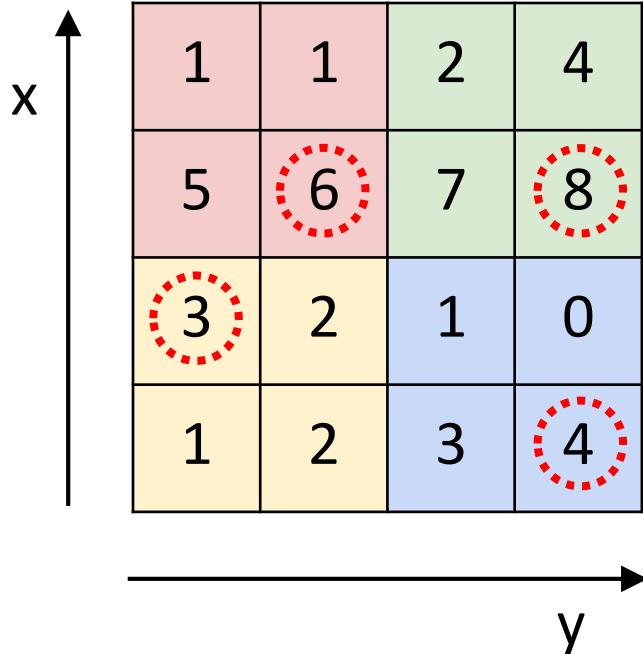
Where are the features we try to extract?

Two more layers to go: POOL/FC



MAX POOLING

Single depth slice

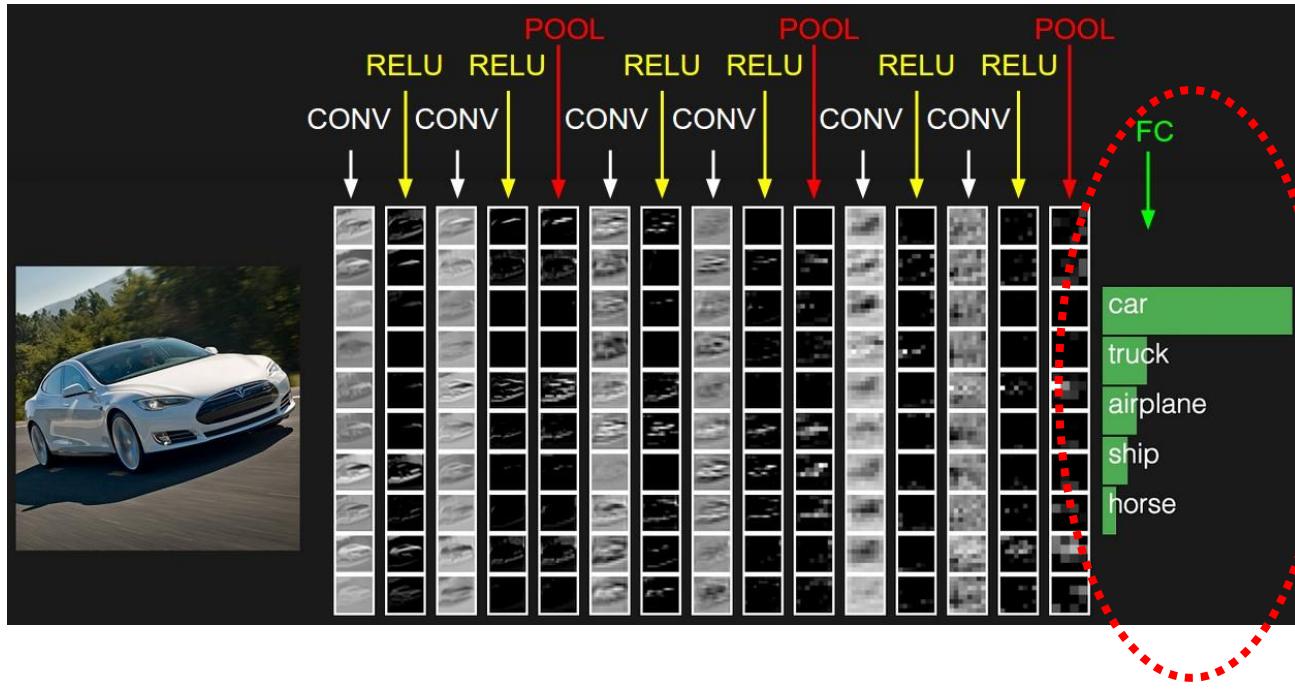


max pool with 2x2 filters
and stride 2

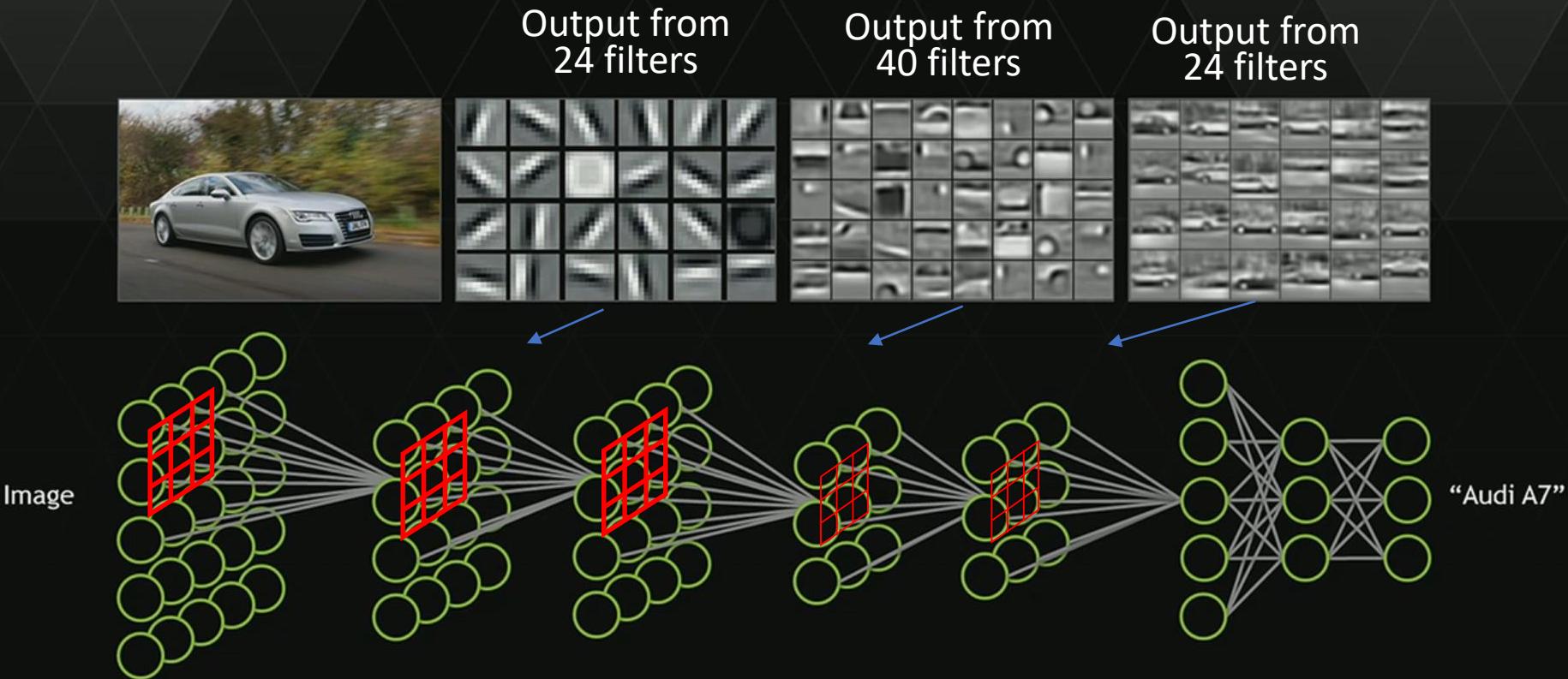
| | |
|---|---|
| 6 | 8 |
| 3 | 4 |

Fully Connected Layer (FC layer)

- Contains neurons that connect to the entire input volume, as in ordinary Neural Networks

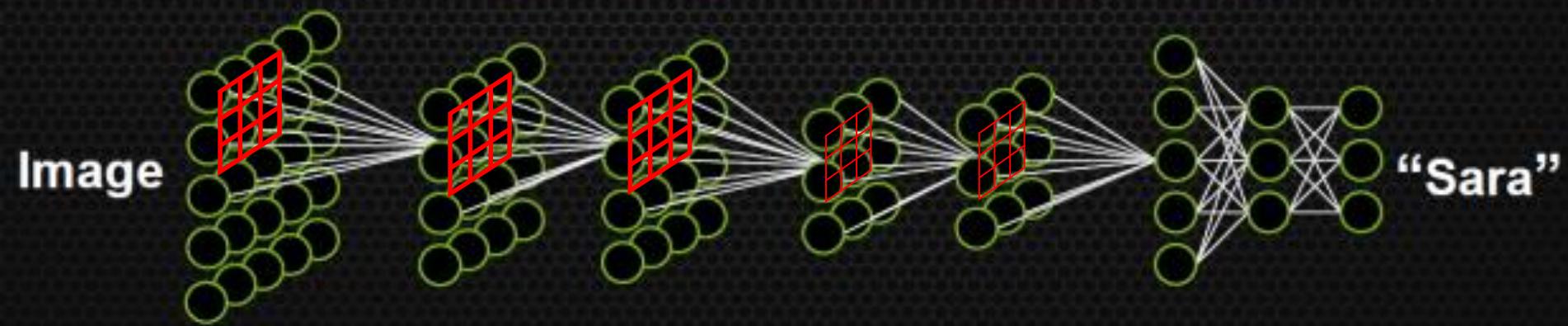
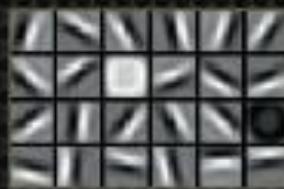


HOW A DEEP NEURAL NETWORK SEES



*Image source: "Unsupervised Learning of Hierarchical Representations with Convolutional Deep Belief Networks" ICML 2009 & Comm. ACM 2011.
Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Ng.*

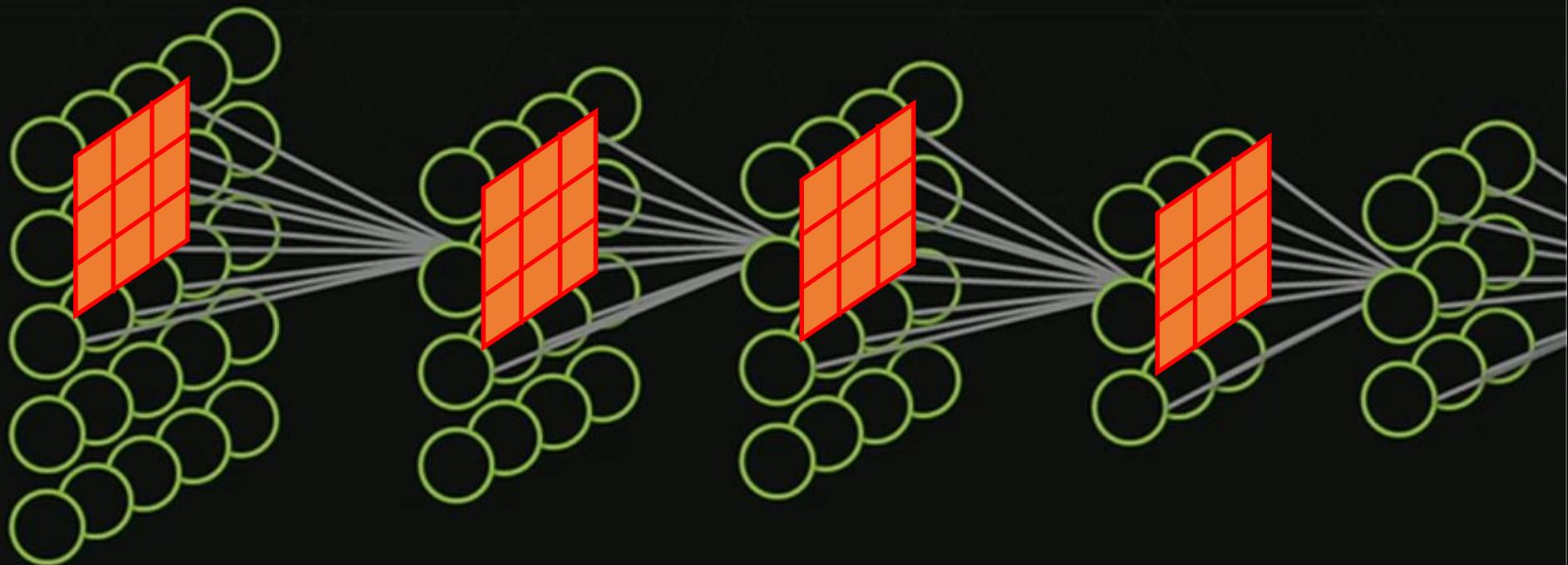
Where are the features we try to extract?



Where are the features we try to extract?

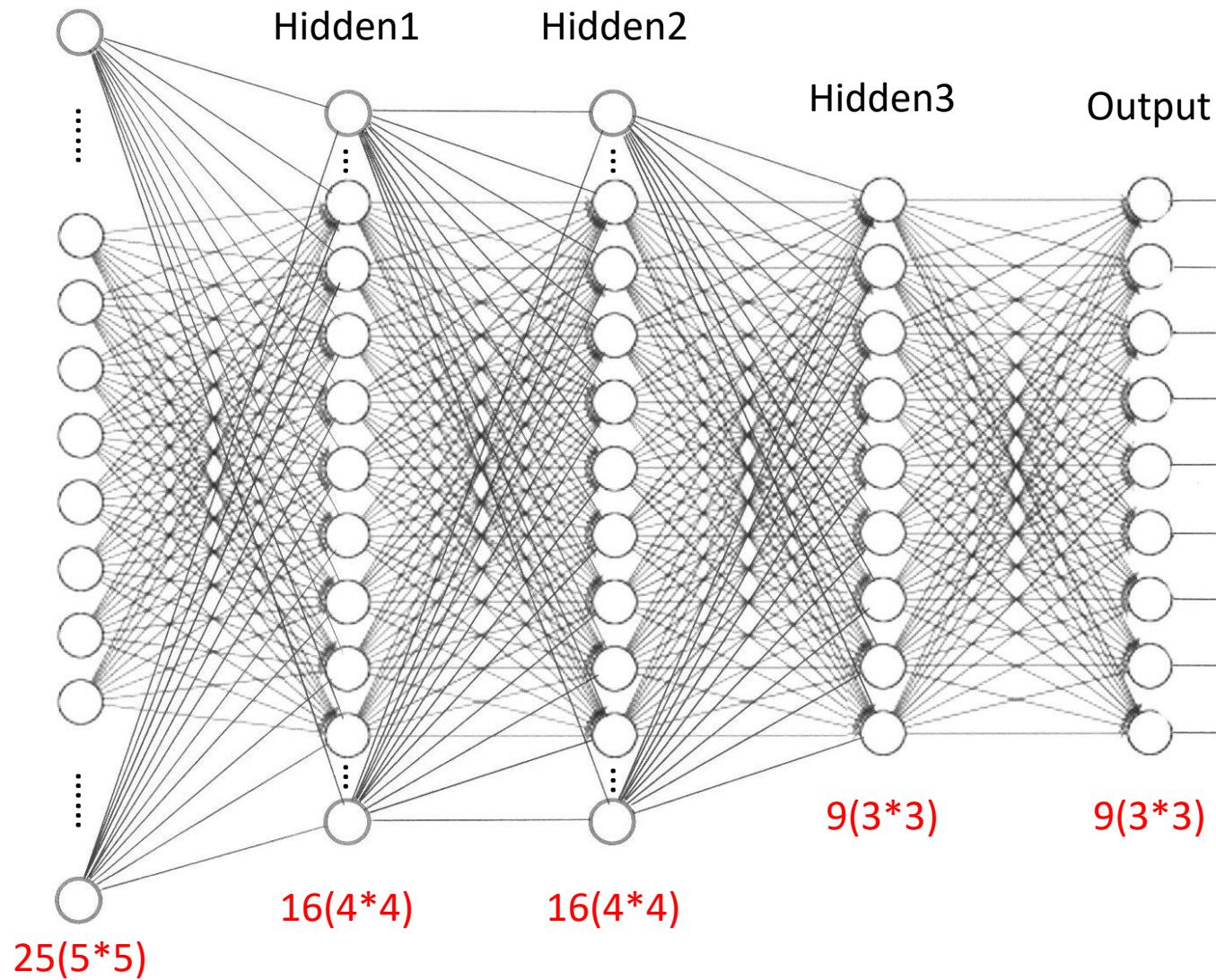
Filter size : 3x3, 1 filter for each convolution,
then how many parameters to be tuned?

$$9 * 4 = 36$$



If fully connected, $25*16+16*16+16*9+9*9 = 881$

Input



Some questions

- Fully connected? (Yes/No)
- Where are the parameters to be tuned?
- We have 2 filters, then?
- What happens if no padding(out-of-bound)?
- Convolution again with the resulting activation map

Learning in CNN

- Parameter tuning in filters
→ filter organizing
- Parameter tuning in fully connected layers

Theoretical backgrounds of CNN

A bit of history:

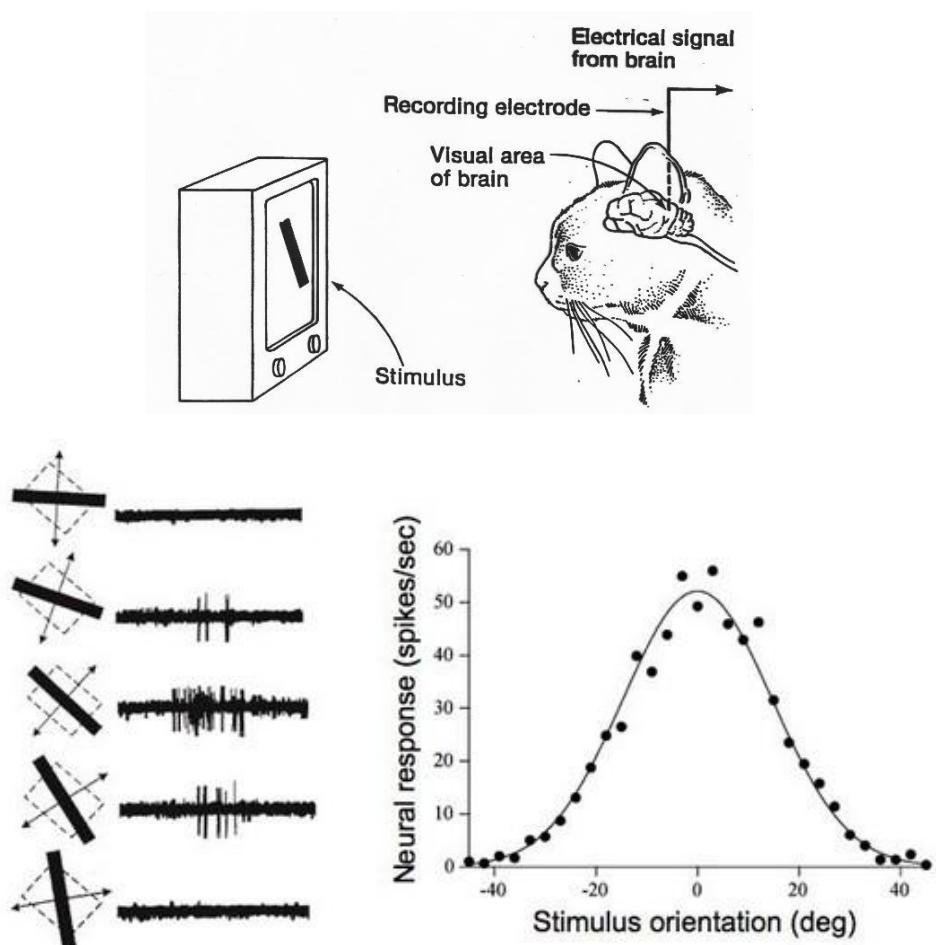
Hubel & Wiesel, 1959

RECEPTIVE FIELDS OF SINGLE
NEURONES IN
THE CAT'S STRIATE CORTEX

1962

RECEPTIVE FIELDS, BINOCULAR
INTERACTION
AND FUNCTIONAL ARCHITECTURE
IN
THE CAT'S VISUAL CORTEX

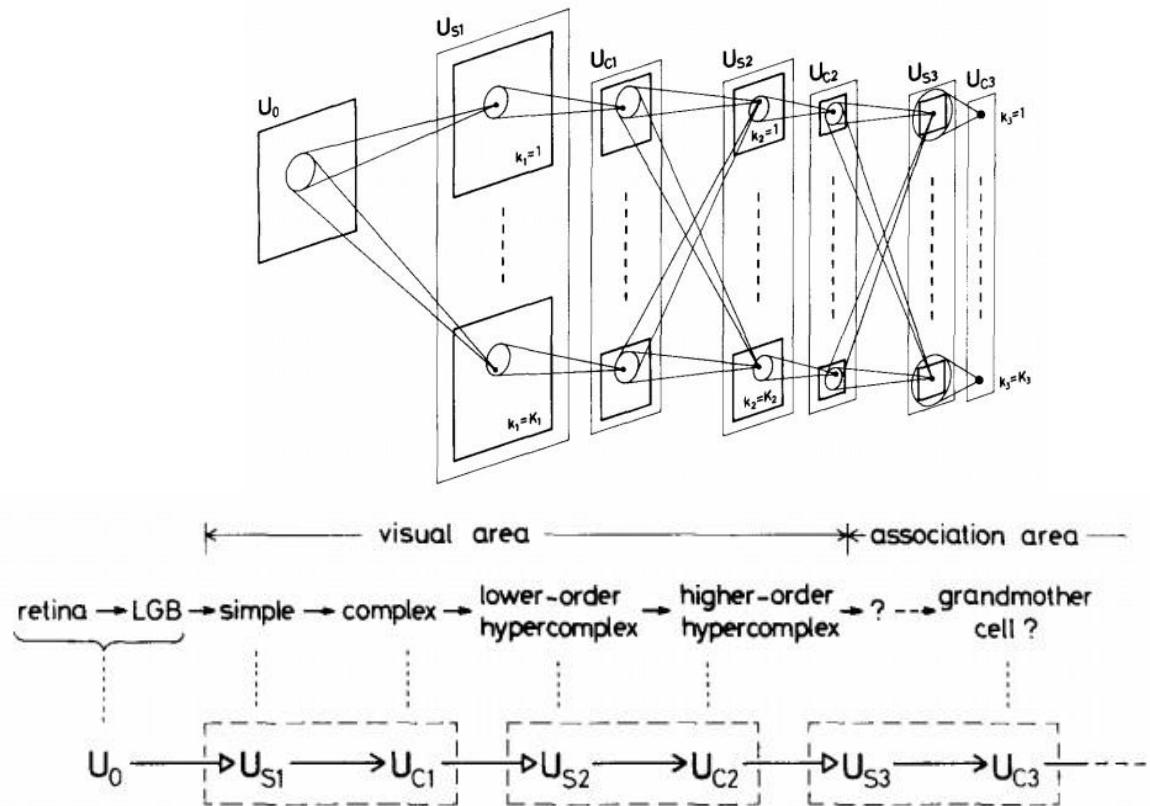
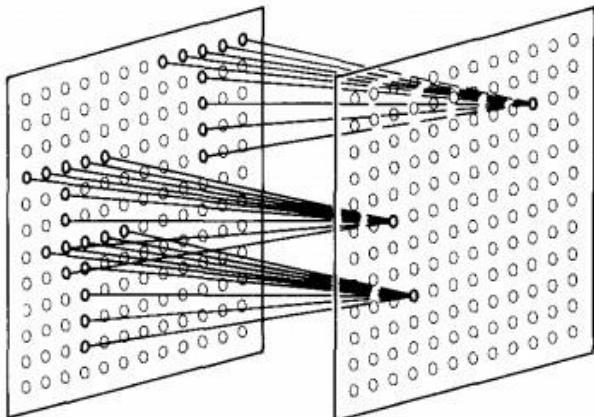
1968...



A bit of history:

“sandwich” architecture (SCSCSC...)
simple cells: modifiable parameters
complex cells: perform pooling

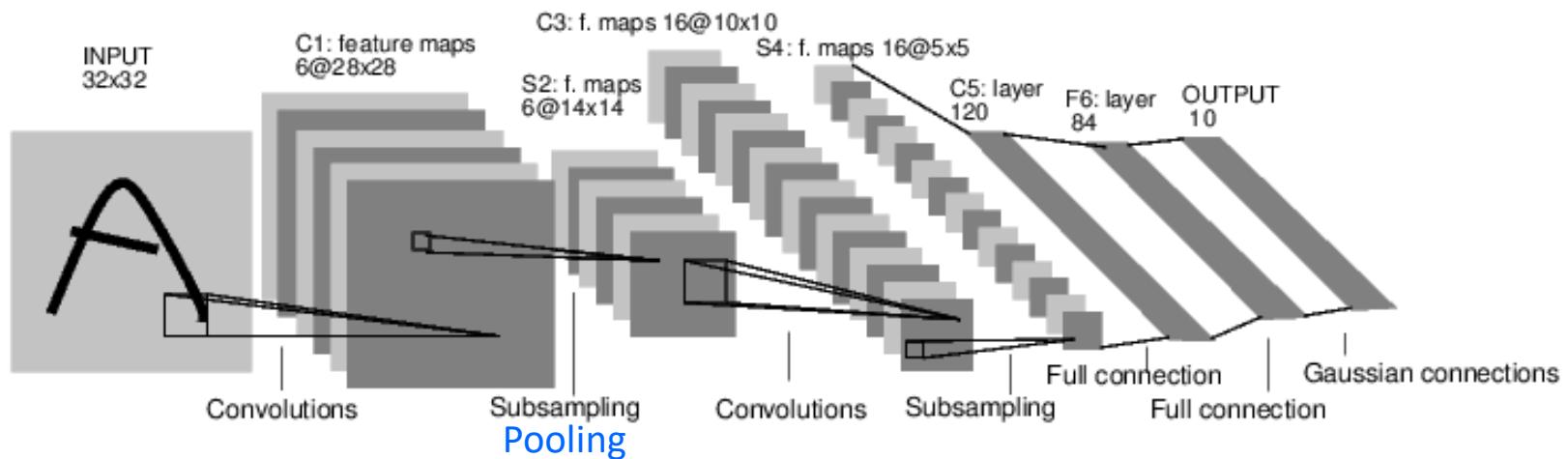
Neuro-cognitron [Fukushima 1980]



LeNet-5

Gradient-based learning applied to document recognition

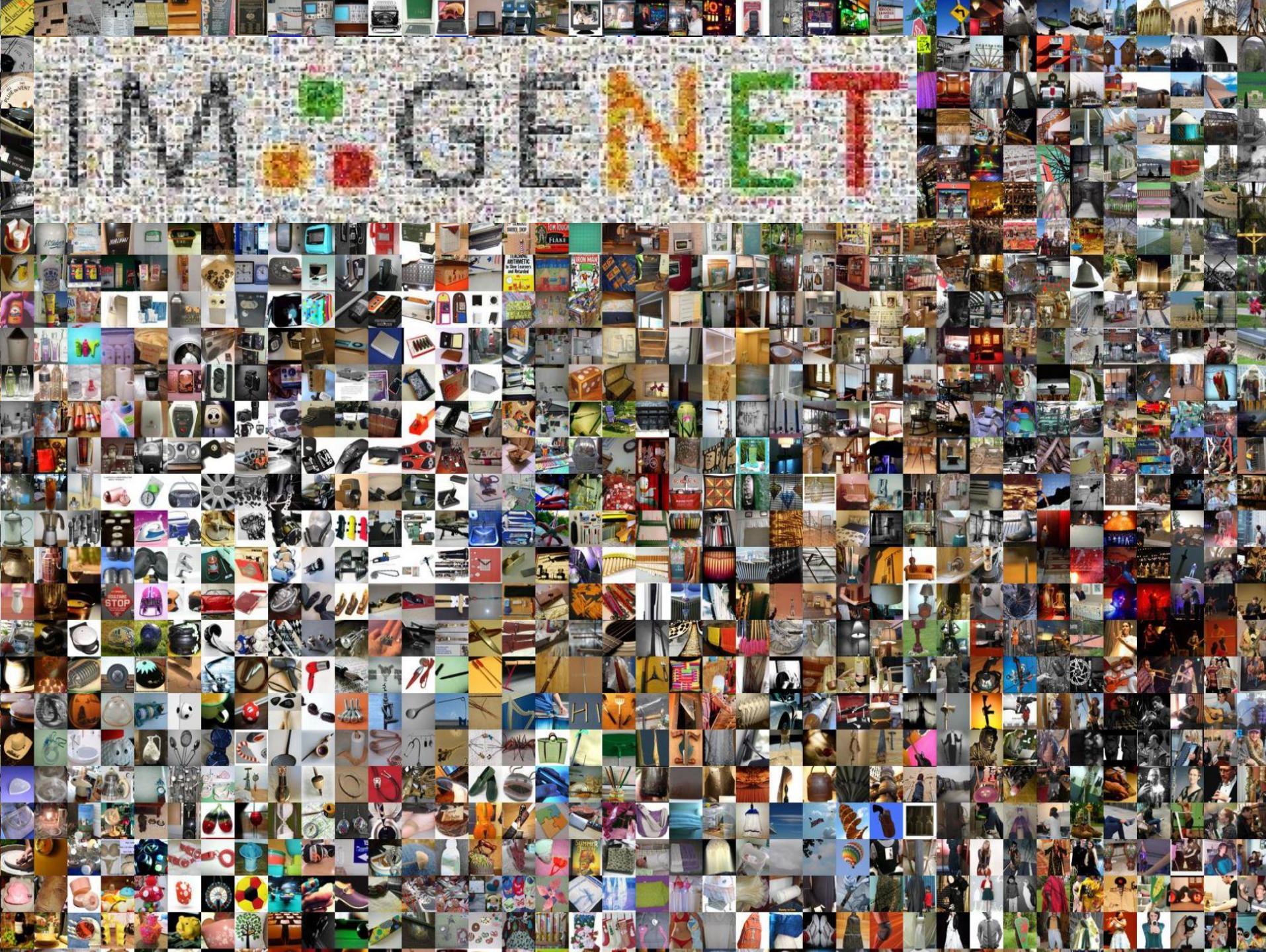
[LeCun, Bottou, Bengio, Haffner 1998]



Convolution-Pooling-Convolution-Pooling-FC

Where are the features we try to extract?

Case Studies

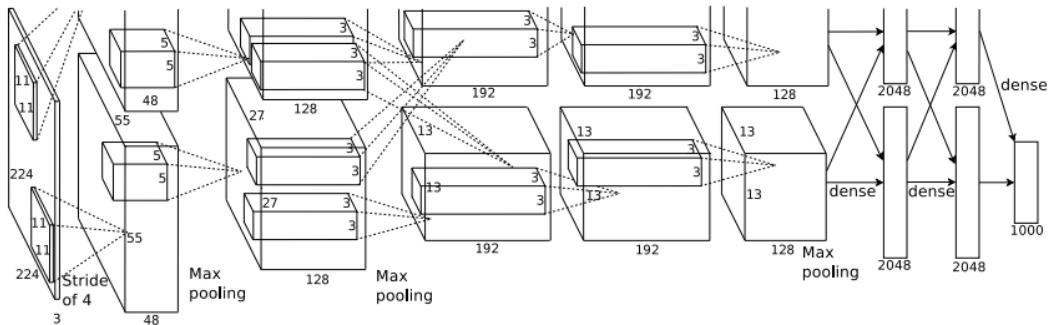


ImageNet and Competition

- **ImageNet**: categorization of huge amount of images according to WordNet schema
- database for ILSVRC (ImageNet Large Scale Visual Recognition Competition)
- **Largest Computer Vision Challenge** showing state-of-the-art performance on the dataset every year
- firstly collected by Deng et al. in 2009
- And it has been organized by Stanford University Vision Lab (prof. Fei-Fei Li) since 2010.

Case Study: AlexNet

[Krizhevsky et al. 2012]



Full (simplified) AlexNet architecture:

[227x227x3] INPUT

[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0

[27x27x96] MAX POOL1: 3x3 filters at stride 2

[27x27x96] NORM1: Normalization layer

[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2

[13x13x256] MAX POOL2: 3x3 filters at stride 2

[13x13x256] NORM2: Normalization layer

[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1

[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1

[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1

[6x6x256] MAX POOL3: 3x3 filters at stride 2

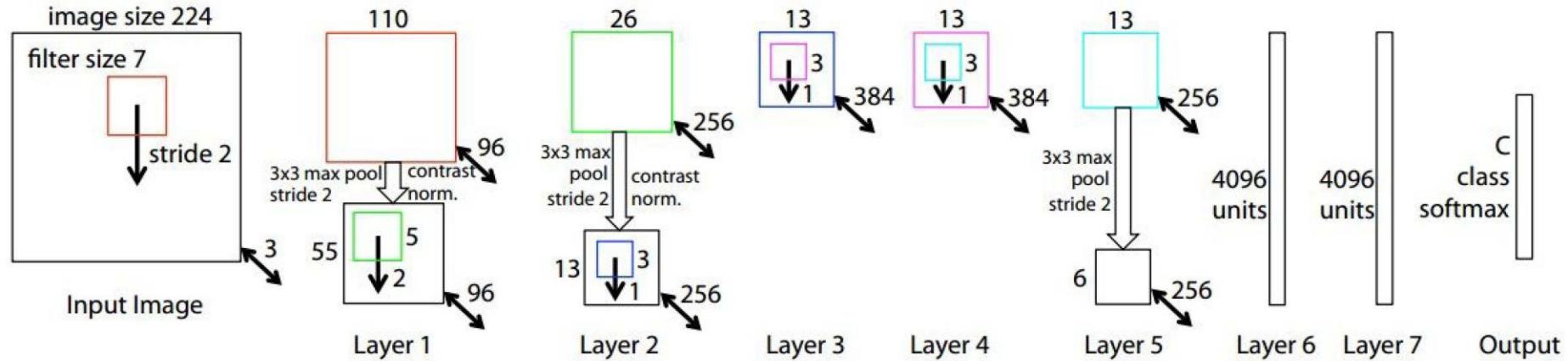
[4096] FC6: 4096 neurons

[4096] FC7: 4096 neurons

[1000] FC8: 1000 neurons (class scores)

Case Study: ZFNet

[Zeiler and Fergus, 2013]



AlexNet but:

CONV1: change from (11x11 stride 4) to (7x7 stride 2)

CONV3,4,5: instead of 384, 384, 256 filters use 512, 1024, 512

ImageNet top 5 error: 15.4% -> 14.8%

Case Study: VGGNet

[Simonyan and Zisserman, 2014]

Only 3x3 CONV stride 1, pad 1
and 2x2 MAX POOL stride 2

best model

11.2% top 5 error in ILSVRC 2013

->

7.3% top 5 error

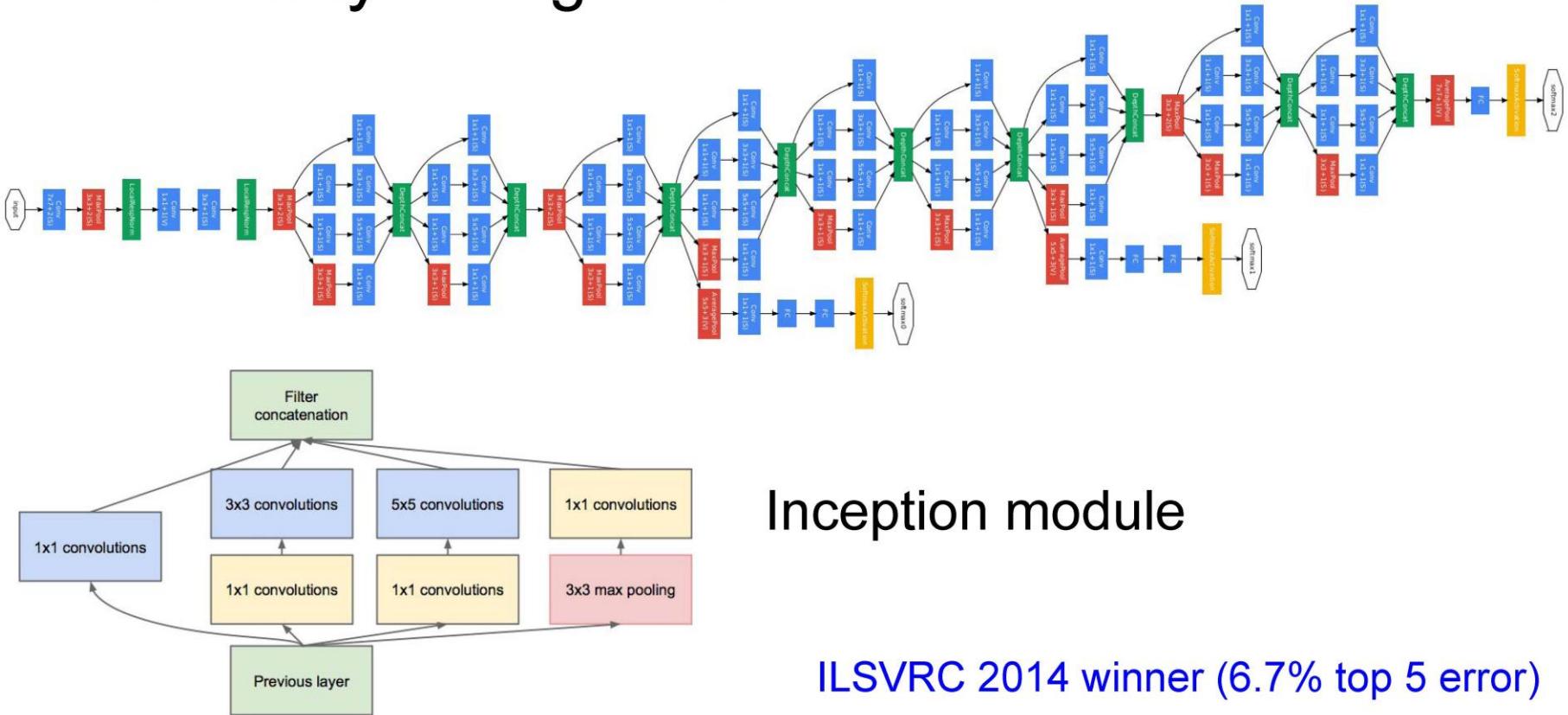
| ConvNet Configuration | | | | | |
|-----------------------------|------------------------|------------------------|------------------------|--|--|
| A | A-LRN | B | C | D | E |
| 11 weight layers | 11 weight layers | 13 weight layers | 16 weight layers | 16 weight layers | 19 weight layers |
| input (224 × 224 RGB image) | | | | | |
| conv3-64 | conv3-64 LRN | conv3-64 conv3-64 | conv3-64 conv3-64 | conv3-64 conv3-64 | conv3-64 conv3-64 |
| maxpool | | | | | |
| conv3-128 | conv3-128 | conv3-128 conv3-128 | conv3-128 conv3-128 | conv3-128 conv3-128 | conv3-128 conv3-128 |
| maxpool | | | | | |
| conv3-256 conv3-256 | conv3-256 conv3-256 | conv3-256 conv3-256 | conv3-256 conv3-256 | conv3-256 conv3-256 conv3-256 conv3-256 | conv3-256 conv3-256 conv3-256 conv3-256 |
| maxpool | | | | | |
| conv3-512 conv3-512 | conv3-512 conv3-512 | conv3-512 conv3-512 | conv3-512 conv3-512 | conv3-512 conv3-512 conv3-512 conv3-512 | conv3-512 conv3-512 conv3-512 conv3-512 |
| maxpool | | | | | |
| conv3-512 conv3-512 | conv3-512 conv3-512 | conv3-512 conv3-512 | conv3-512 conv3-512 | conv3-512 conv3-512 conv3-512 conv3-512 | conv3-512 conv3-512 conv3-512 conv3-512 |
| maxpool | | | | | |
| FC-4096 | | | | | |
| FC-4096 | | | | | |
| FC-1000 | | | | | |
| soft-max | | | | | |

Table 2: Number of parameters (in millions).

| Network | A,A-LRN | B | C | D | E |
|----------------------|---------|-----|-----|-----|-----|
| Number of parameters | 133 | 133 | 134 | 138 | 144 |

Case Study: GoogLeNet

[Szegedy et al., 2014]



Case Study: ResNet

[He et al., 2015]

ILSVRC 2015 winner (3.6% top 5 error)



MSRA @ ILSVRC & COCO 2015 Competitions

- **1st places in all five main tracks**

- ImageNet Classification: “Ultra-deep” (quote Yann) **152-layer nets**
- ImageNet Detection: **16%** better than 2nd
- ImageNet Localization: **27%** better than 2nd
- COCO Detection: **11%** better than 2nd
- COCO Segmentation: **12%** better than 2nd

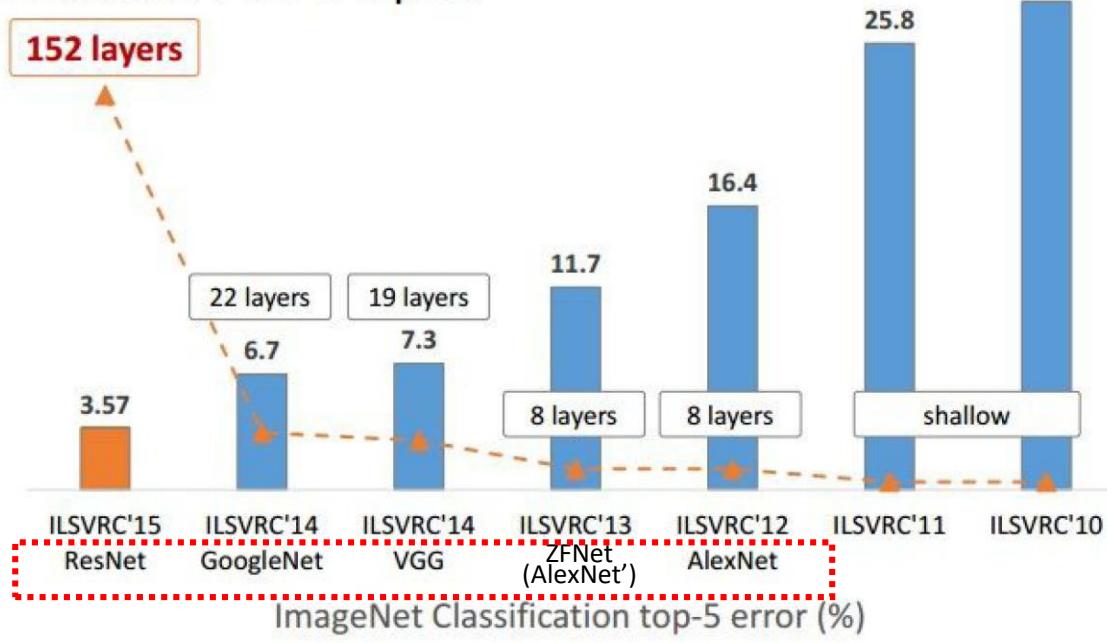
*improvements are relative numbers



Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". arXiv 2015.

Slide from Kaiming He's recent presentation <https://www.youtube.com/watch?v=1PGLj-uKT1w>

Revolution of Depth



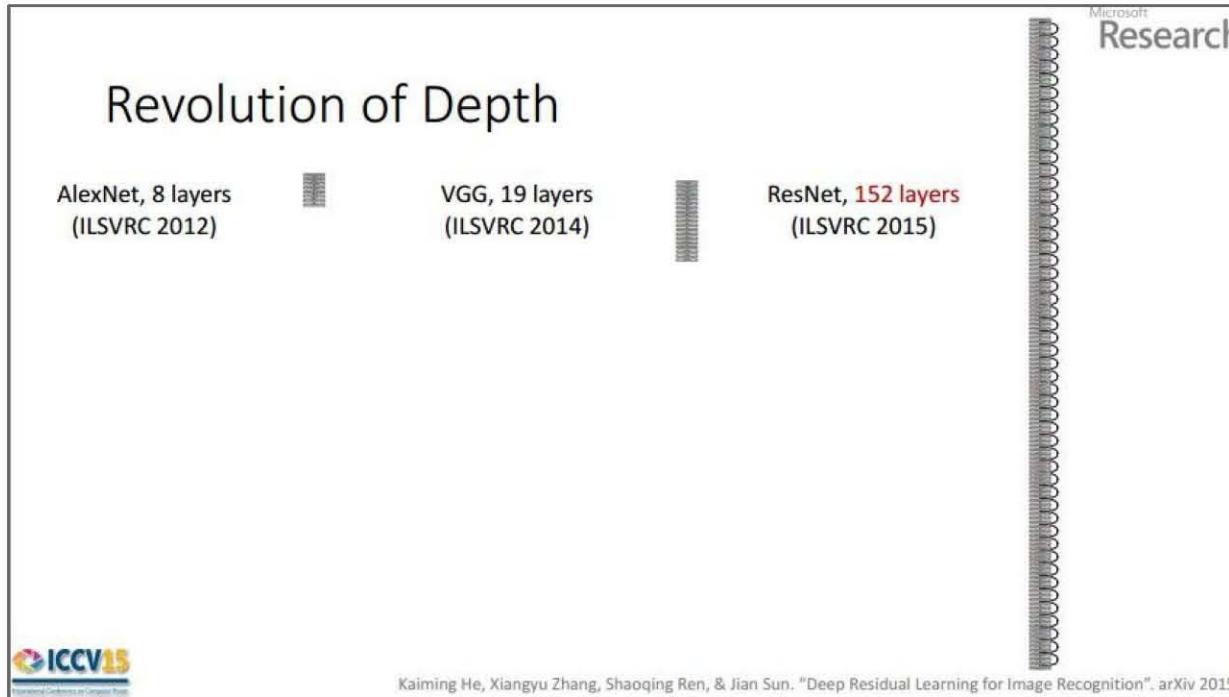
Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". arXiv 2015.

(slide from Kaiming He's recent presentation)

Case Study: ResNet

[He et al., 2015]

ILSVRC 2015 winner (3.6% top 5 error)

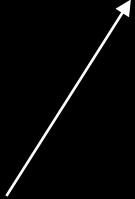


2-3 weeks of training on 8 GPU machine

at runtime: faster than a VGGNet!
(even though it has 8x more layers)

(slide from Kaiming He's recent presentation)

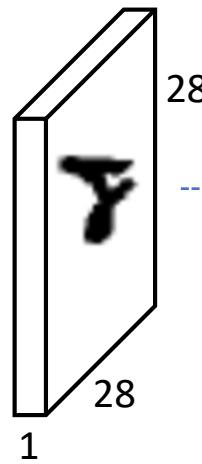
Dot product & shift for feature extraction



CNN, Convolutional Neural Network

20_mnist_softmax.py

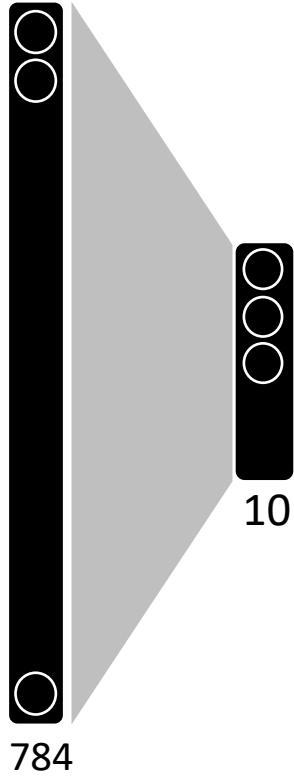
Accuracy: 90.23



28

28

1

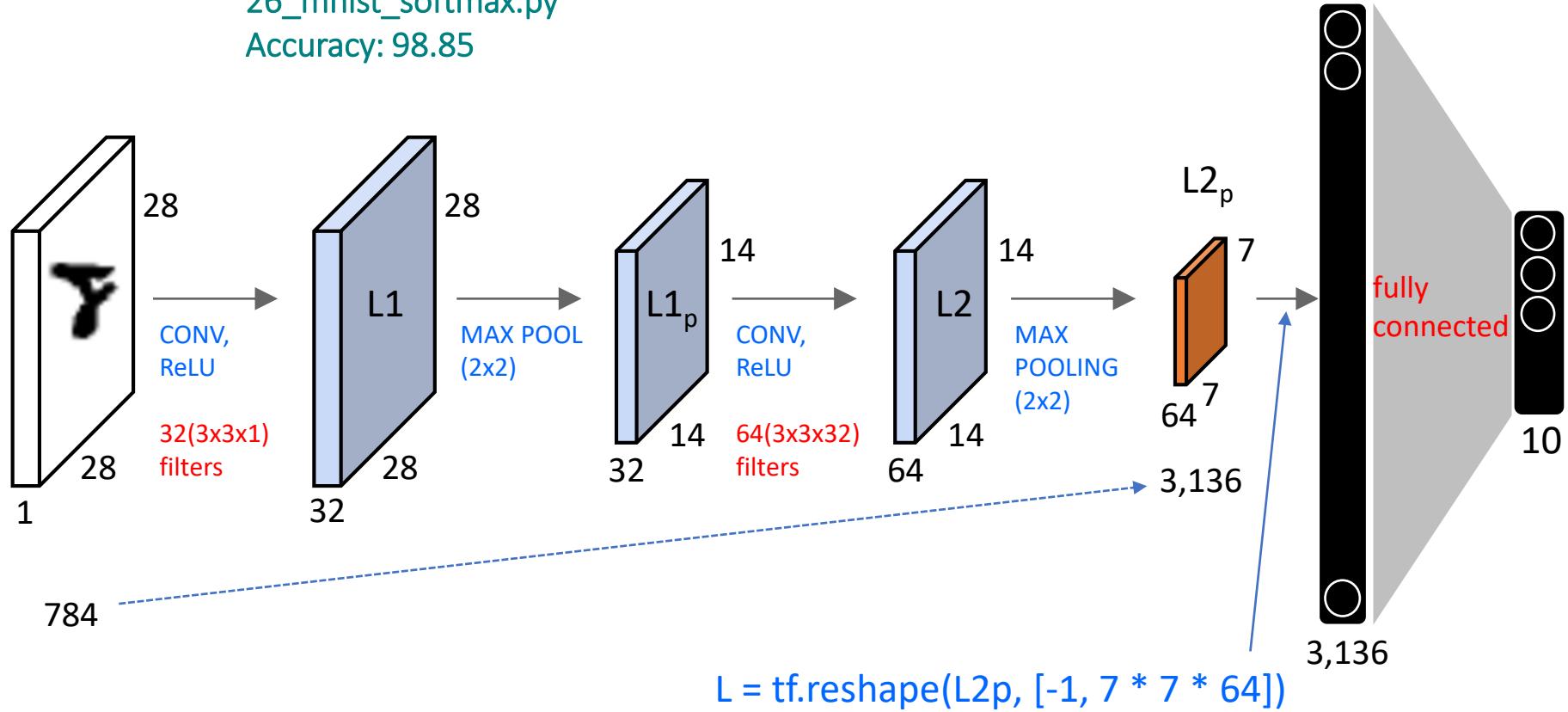


784

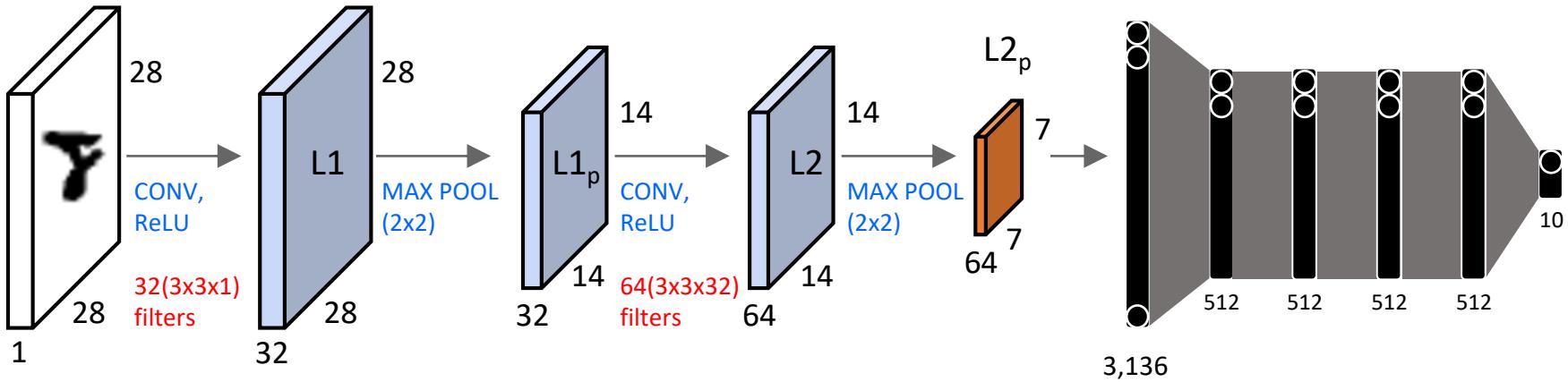
10

26_mnist_softmax.py

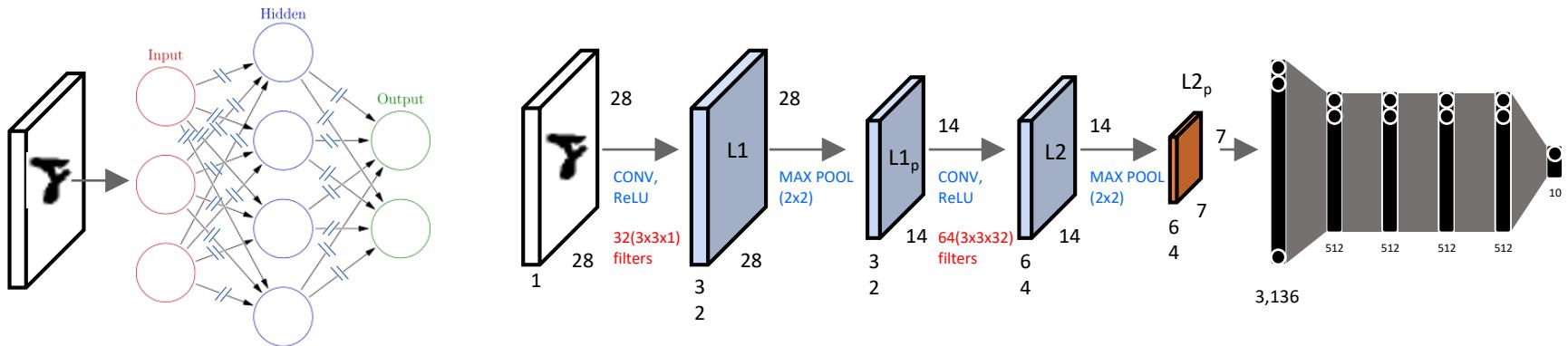
Accuracy: 98.85

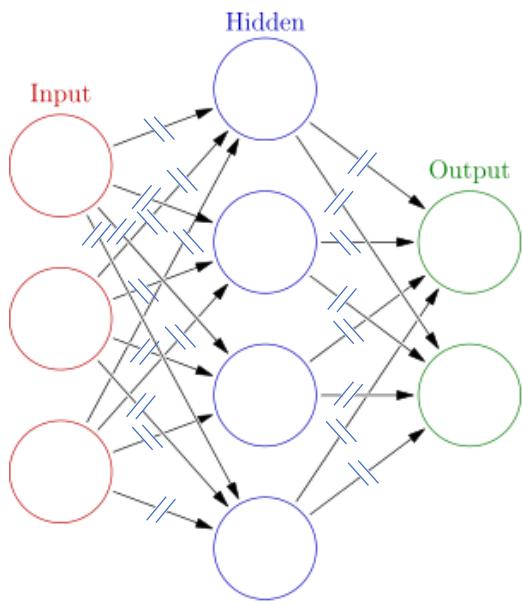


Accuracy: ?????

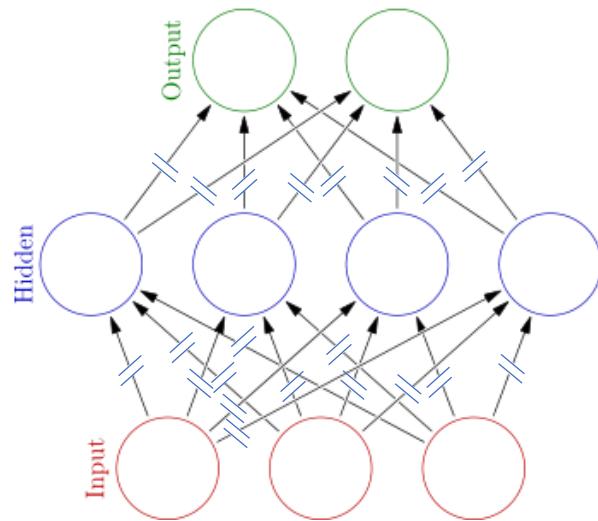


Feedforward NN

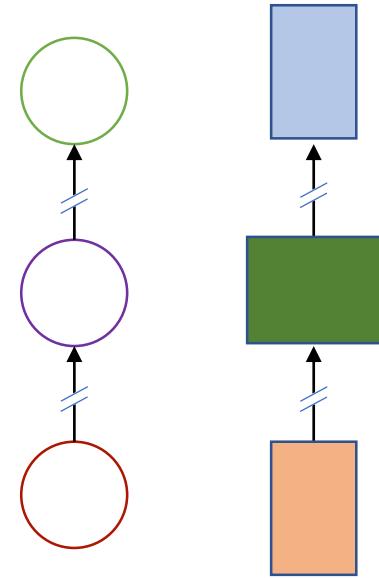




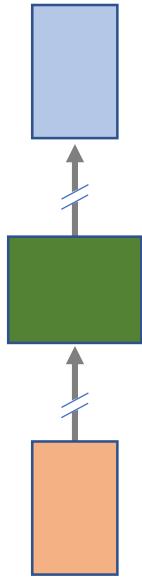
3 layered FCNN



Rotated 90°
clockwise

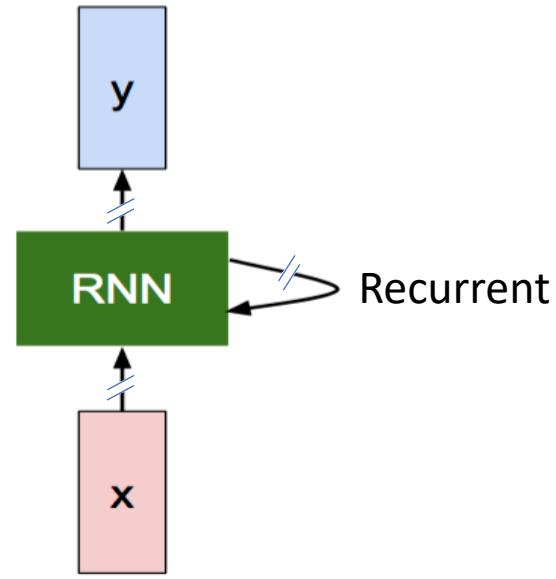


Simplified
Rectangular
representation



Simplified NN

What is the arrow?
Where are synapses?
What is learning?



Recurrent NN

Imagine the inside of RNN

Recurrent Neural Networks

RNN

Memorization

time series information

Song, poetry, novel
Stock prices
All kinds of time series information

Prediction & Creativity

Composition, Writing,
Stock Market Prediction
Natural language processing

“So, do you understand?”

The answer may be different
according to the context(state)
discussed at some previous time.

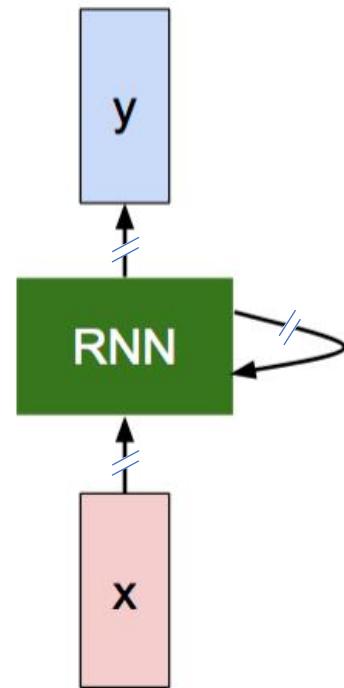
Same questions, but
different answers
with memory.

Recurrent Neural Network

We can process a sequence of vectors \mathbf{x} by applying a recurrence formula at every time step:

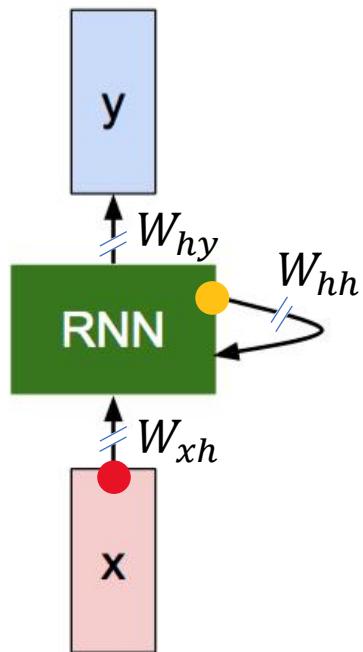
$$h_t = f_W(h_{t-1}, x_t)$$

new state old state input vector at
some function some time step
with parameters W



(Vanilla) Recurrent Neural Network

The state consists of a single “hidden” vector \mathbf{h} :



$$h_t = f_W(h_{t-1}, x_t)$$

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

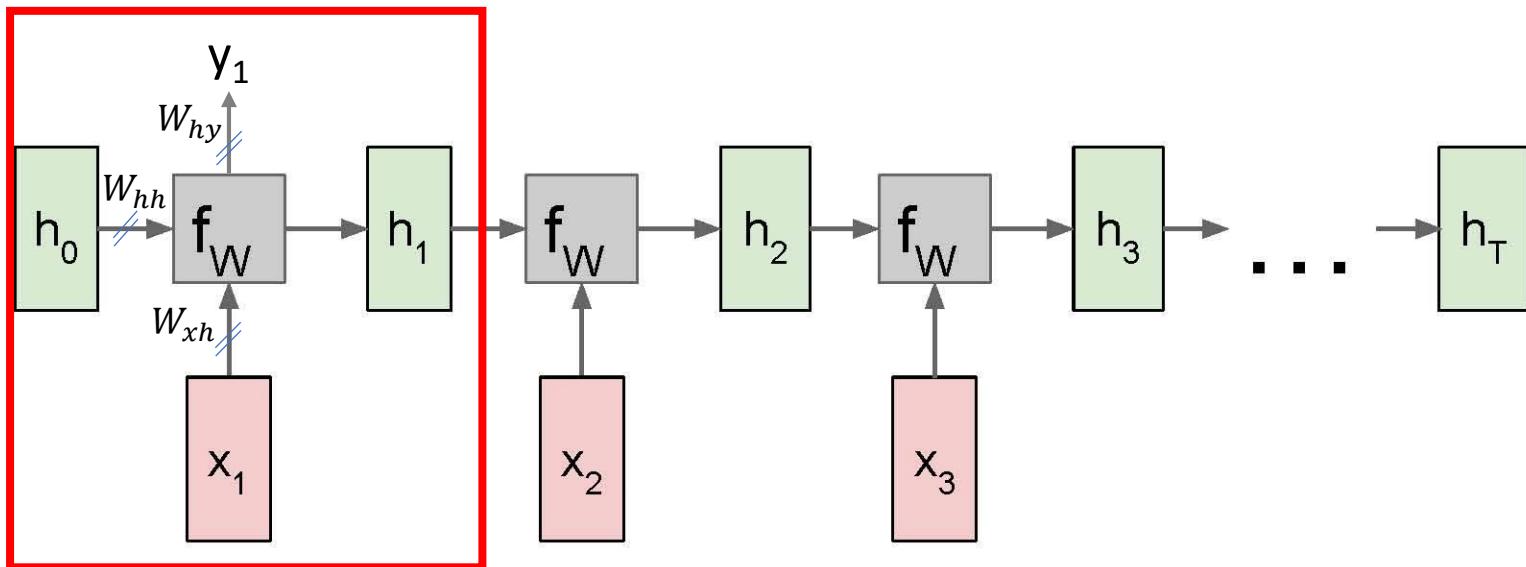
$$y_t = W_{hy}h_t$$



RNN: Computation Graph (Unfolding)

$$h_1 = \tanh(W_{hh}h_0 + W_{xh}x_1)$$

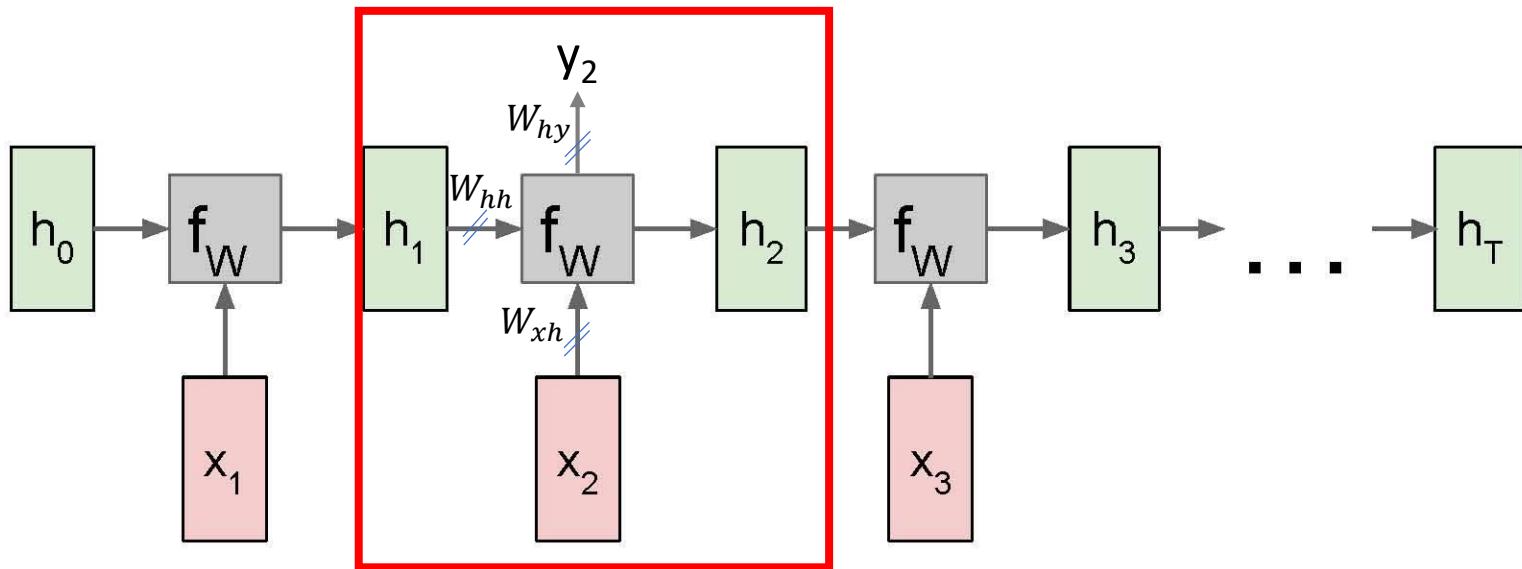
$$y_1 = W_{hy}h_1$$



RNN: Computational Graph (Unfolding)

$$h_2 = \tanh(W_{hh}h_1 + W_{xh}x_2)$$

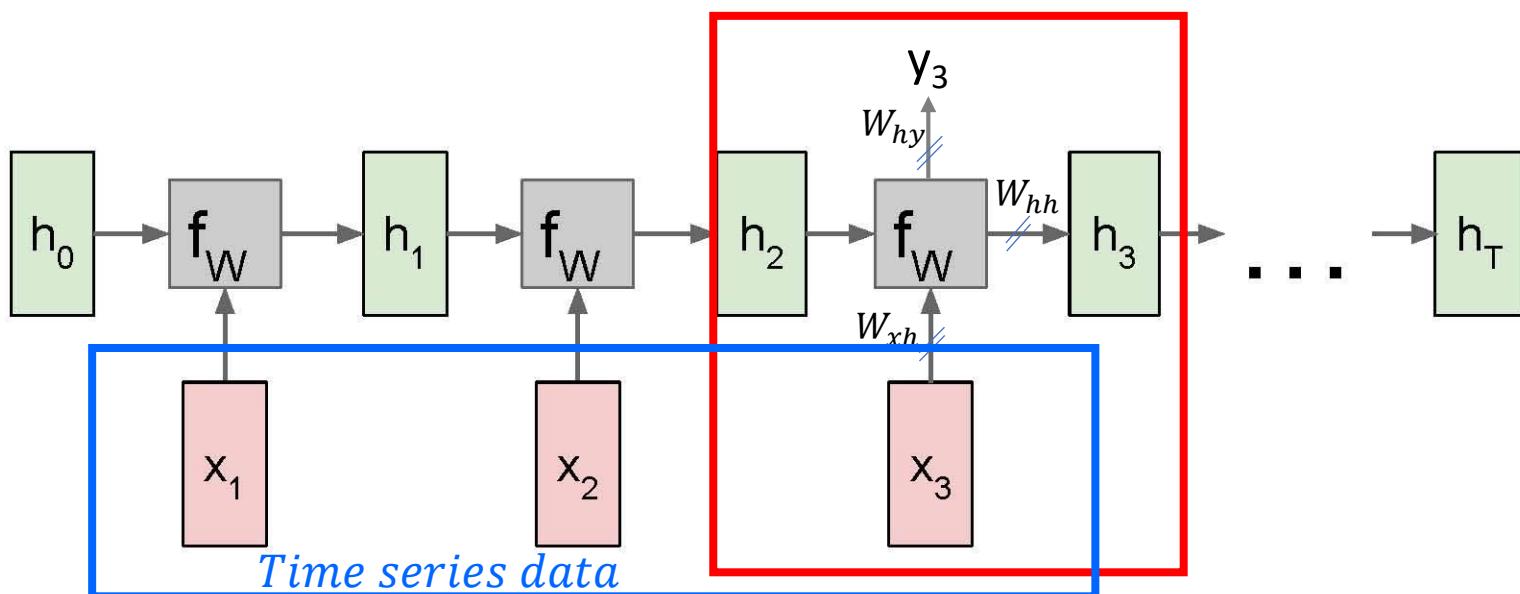
$$y_2 = W_{hy}h_2$$



RNN: Computational Graph (Unfolding)

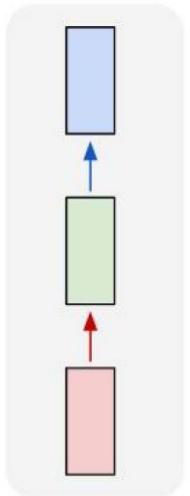
$$h_3 = \tanh(W_{hh}h_2 + W_{xh}x_3)$$

$$y_3 = W_{hy}h_3$$



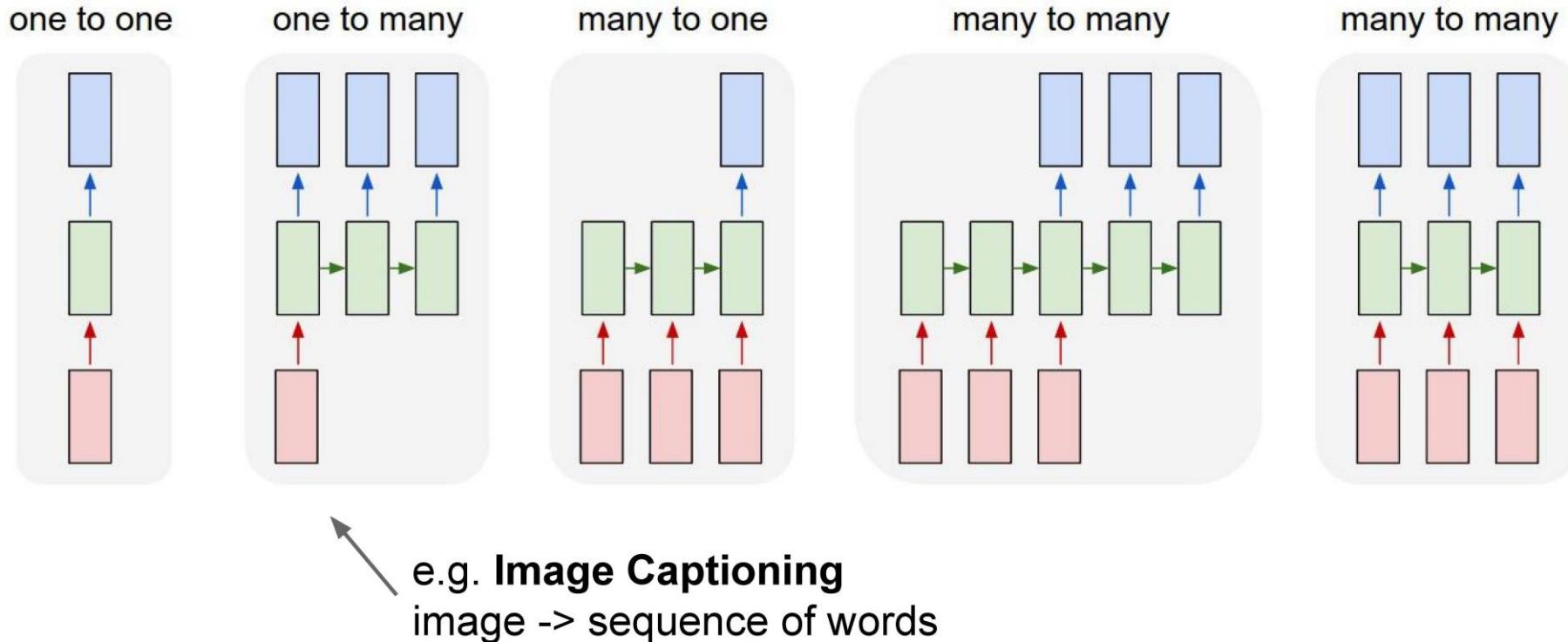
“Vanilla” Neural Network

one to one



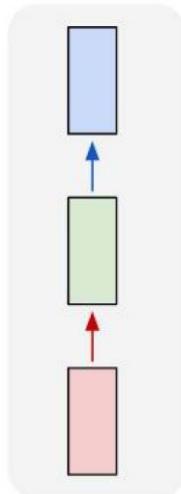
Vanilla Neural Networks

Recurrent Neural Networks: Process Sequences

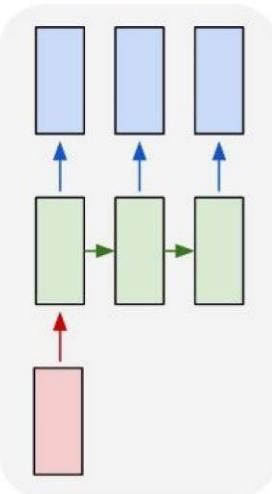


Recurrent Neural Networks: Process Sequences

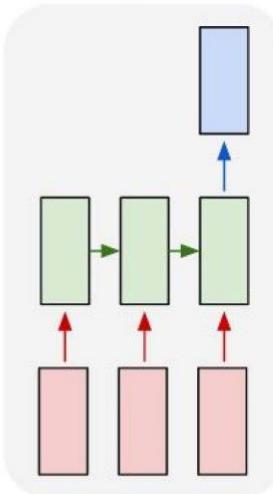
one to one



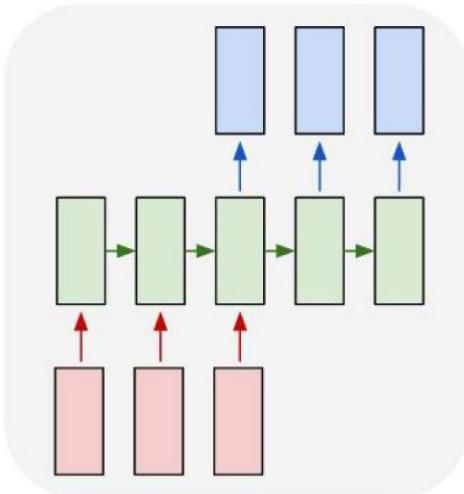
one to many



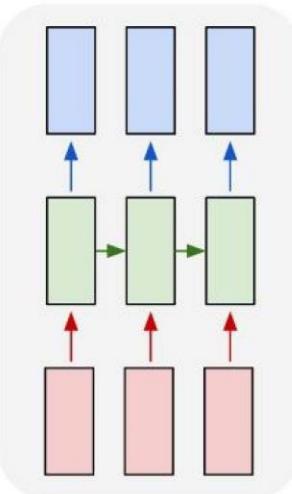
many to one



many to many



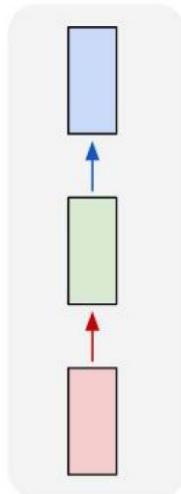
many to many



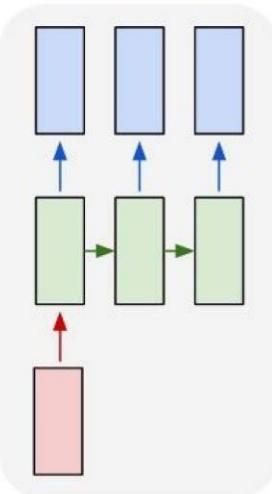
e.g. **Sentiment Classification**
sequence of words -> sentiment

Recurrent Neural Networks: Process Sequences

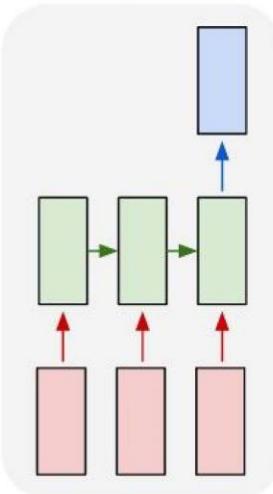
one to one



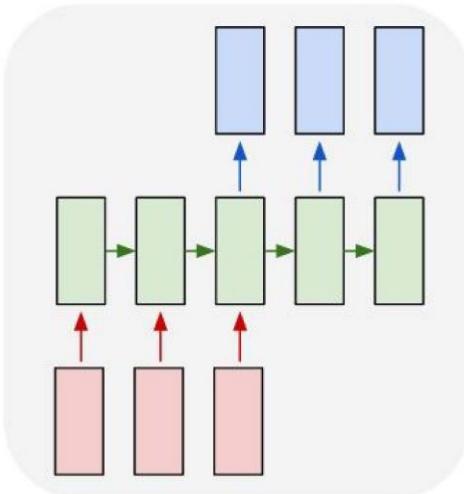
one to many



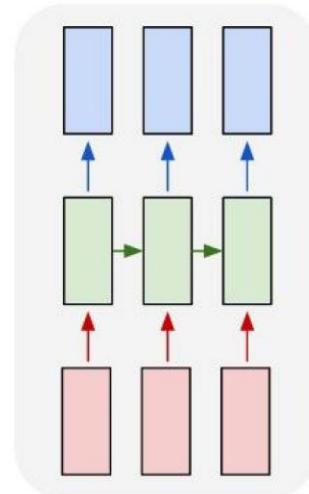
many to one



many to many

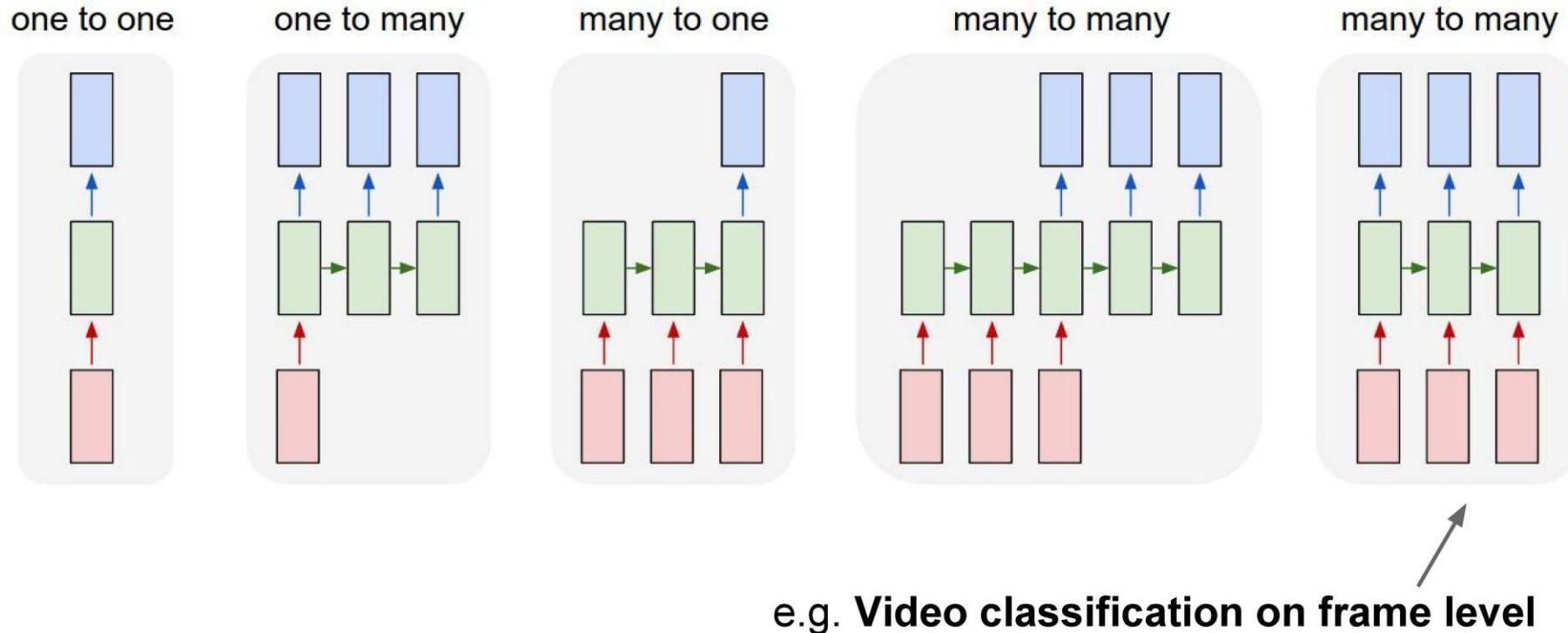


many to many



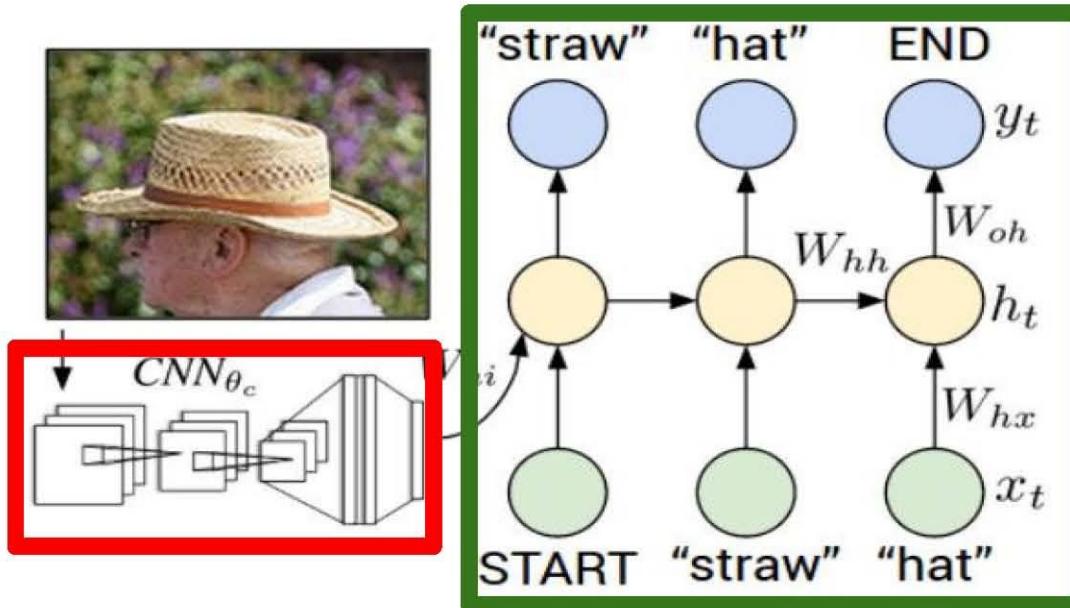
e.g. **Machine Translation**
seq of words -> seq of words

Recurrent Neural Networks: Process Sequences



CNN + RNN

Recurrent Neural Network



Convolutional Neural Network

CNN + RNN

Image Captioning: Example Results

Captions generated using neuraltalk2
All images are CC0 Public domain:
[cat suitcase](#), [cat tree](#), [dog bear](#),
[surfers](#), [tennis](#), [giraffe](#), [motorcycle](#)



A cat sitting on a suitcase on the floor



A cat is sitting on a tree branch



A dog is running in the grass with a frisbee



A white teddy bear sitting in the grass



Two people walking on the beach with surfboards



A tennis player in action on the court



Two giraffes standing in a grassy field



A man riding a dirt bike on a dirt track