Social Data Science: Text Data and Deep Learning

Week 5

Architectures and concepts

Overview

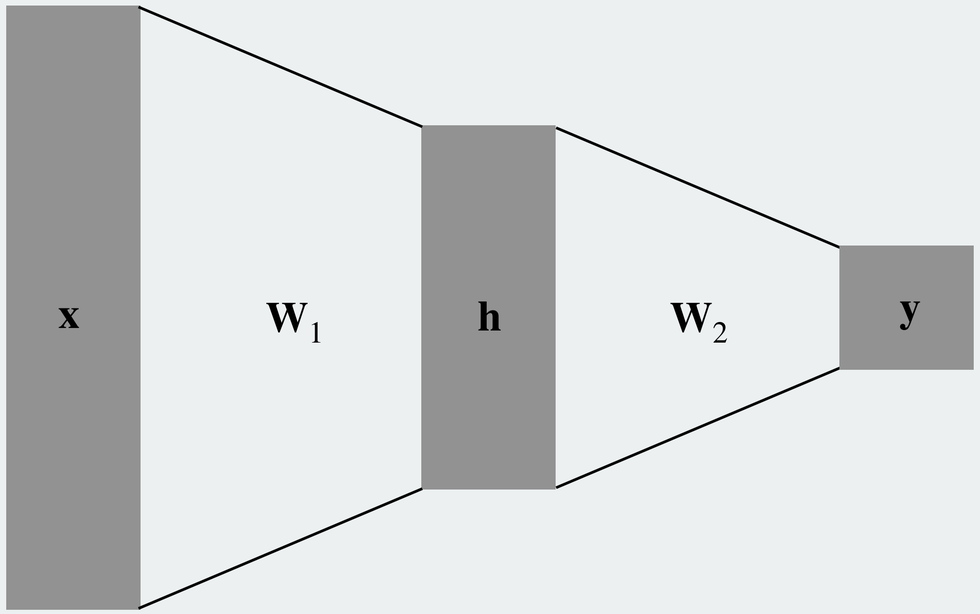
1. **Convolutional Neural Networks**
2. **Recurrent Neural Networks**
3. **Transfer learning**
4. **Generative Models**
    a. Generative Adversarial Networks (GANs)
    b. Variational Autoencoders (VAEs)

Convolutional Neural Networks

*Model architecture for **image data***

Convolutional Neural Networks

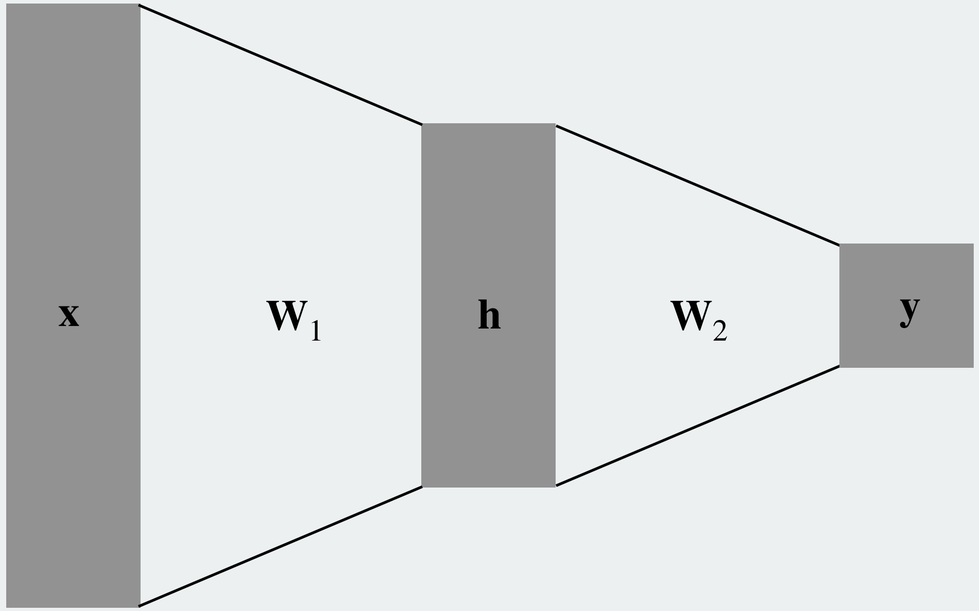
> The problem with vanilla neural networks



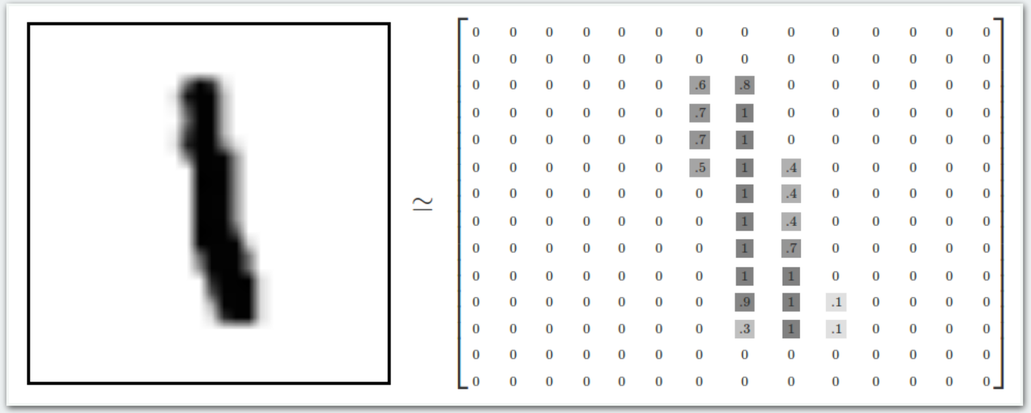
- Single operation on whole input
- Each neuron reacts to specific inputs

Convolutional Neural Networks

> The problem with vanilla neural networks



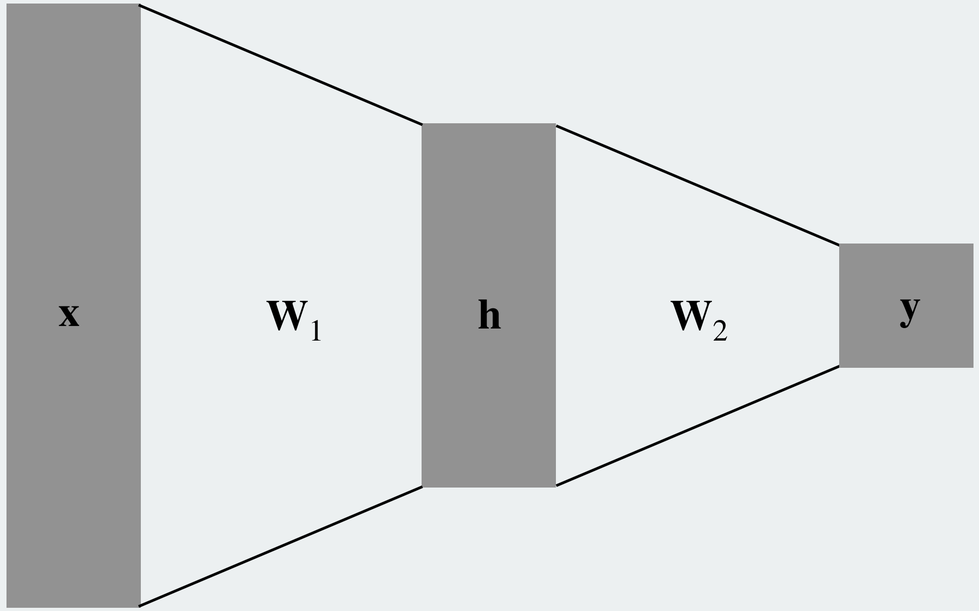
- Single operation on whole input
- Each neuron reacts to specific inputs
- Bad for images: objects move around
- No attention to spatial adjacency



Convolutional Neural Networks

> The problem with vanilla neural networks

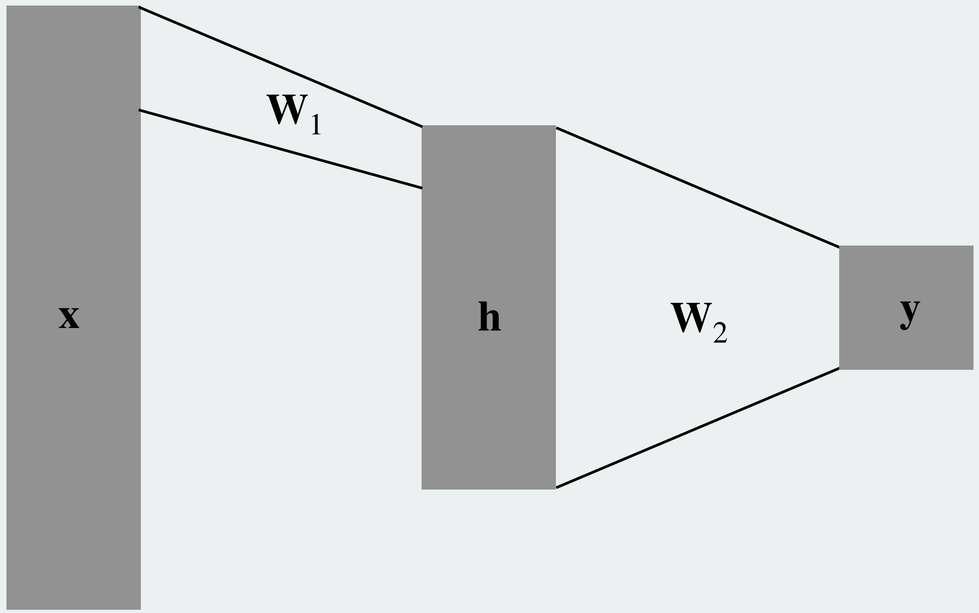
> Solution: “slide” the weight matrix over the input



Convolutional Neural Networks

> The problem with vanilla neural networks

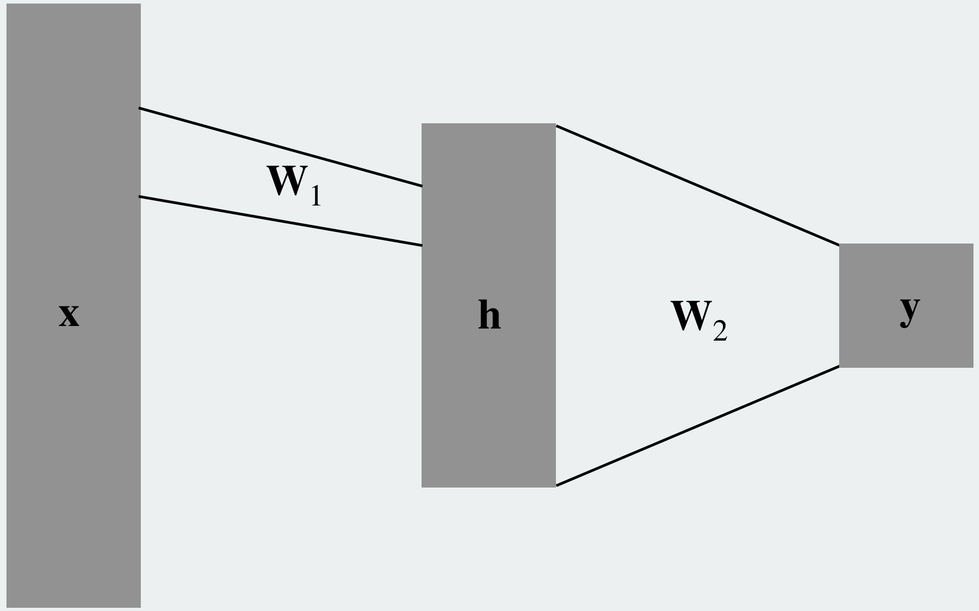
> Solution: “slide” the weight matrix over the input



Convolutional Neural Networks

> The problem with vanilla neural networks

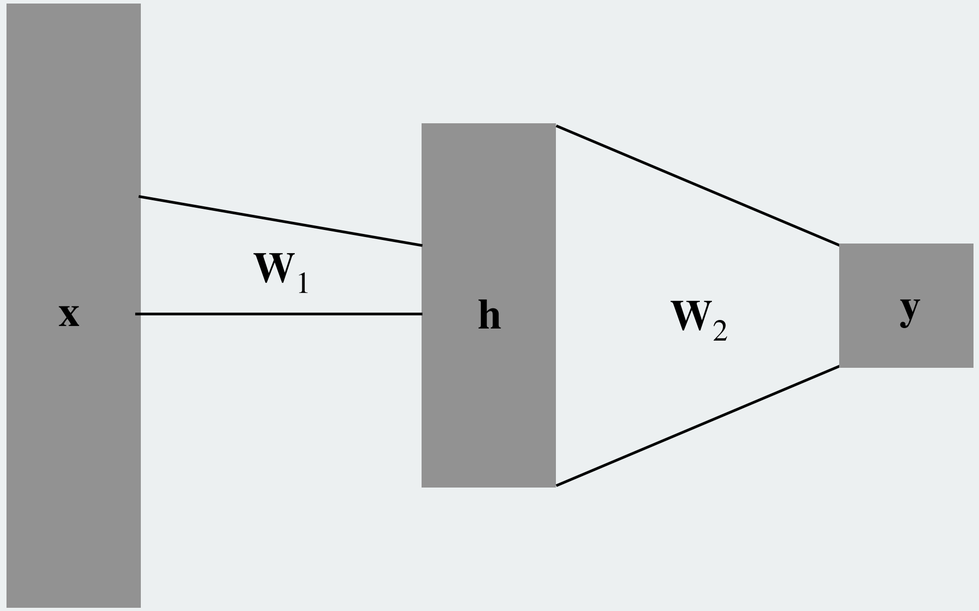
> Solution: “slide” the weight matrix over the input



Convolutional Neural Networks

> The problem with vanilla neural networks

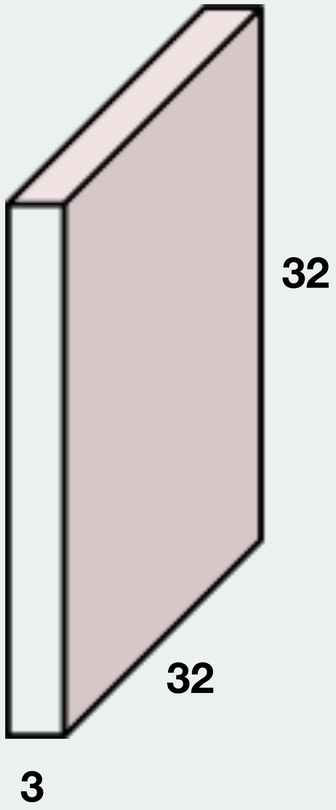
> Solution: “slide” the weight matrix over the input



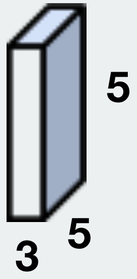
Convolutional Neural Networks

> Concept: *Convolution*

Input *image*
32 x 32 x 3



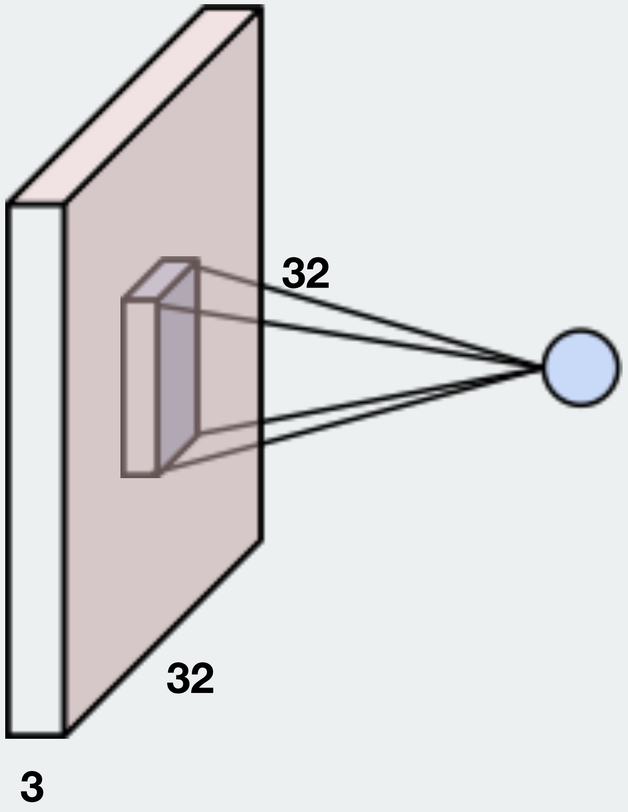
filter
5 x 5 x 3



Convolutional Neural Networks

> Convolve the filter across the input image to computing dot products

Input *image*
32 x 32 x 3

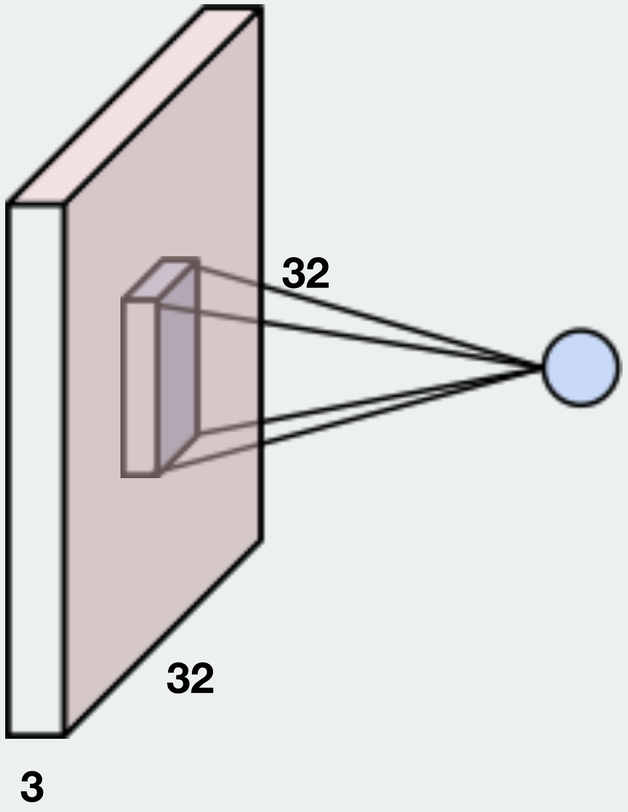


1 number: the result of taking the dot product between the filter and one chunk of the image (5*5*3 = 75-dimensional dot product + bias)

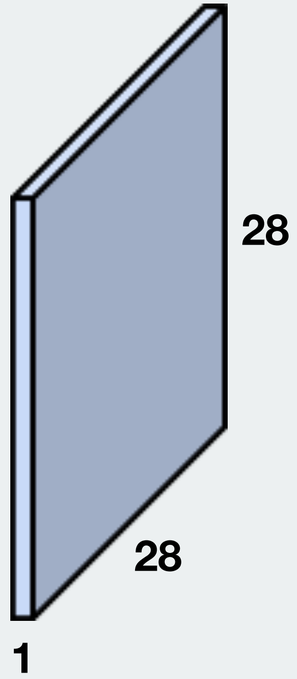
Convolutional Neural Networks

> Convolution by 1 filter produces new *activation map* of depth 1

Input *image*
32 x 32 x 3



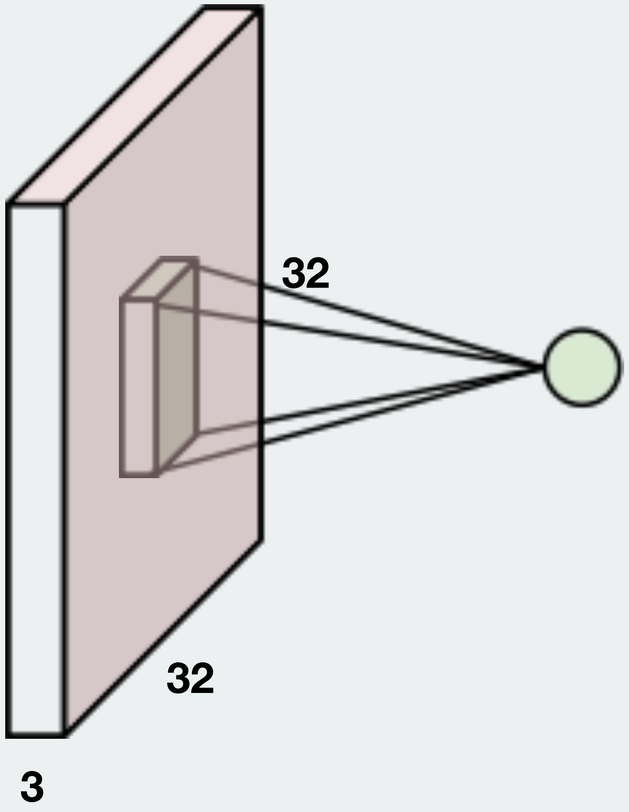
Convolve over all
spatial locations



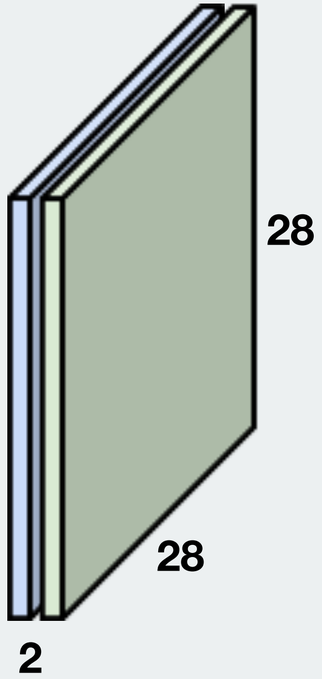
Convolutional Neural Networks

> Convolution by 2 filters produces new *activation map* of depth 2

Input *image*
32 x 32 x 3



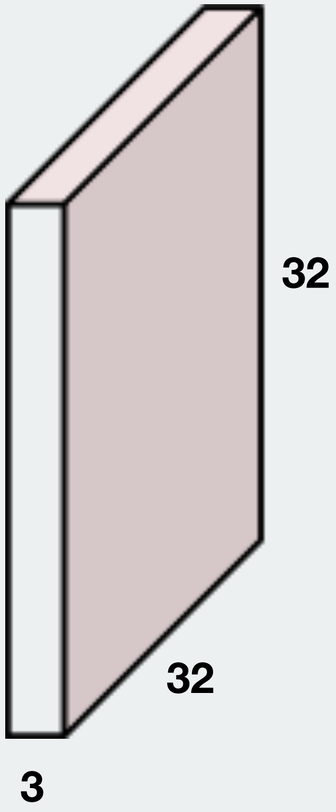
Convolve over all
spatial locations



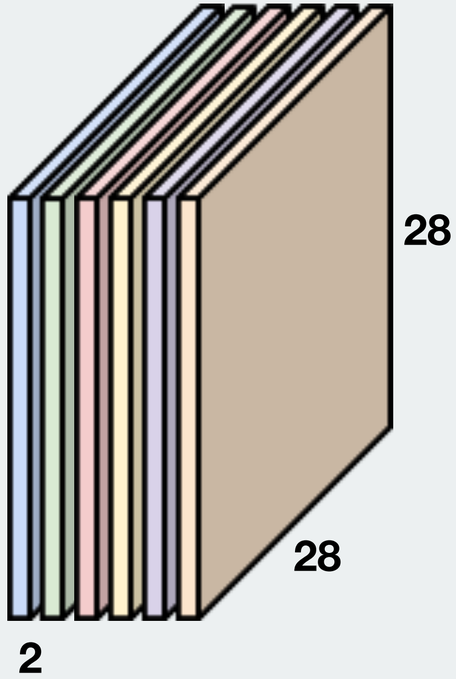
Convolutional Neural Networks

> Convolution by *n* filters produces new *activation map* of depth *n*

Input *image*
32 x 32 x 3

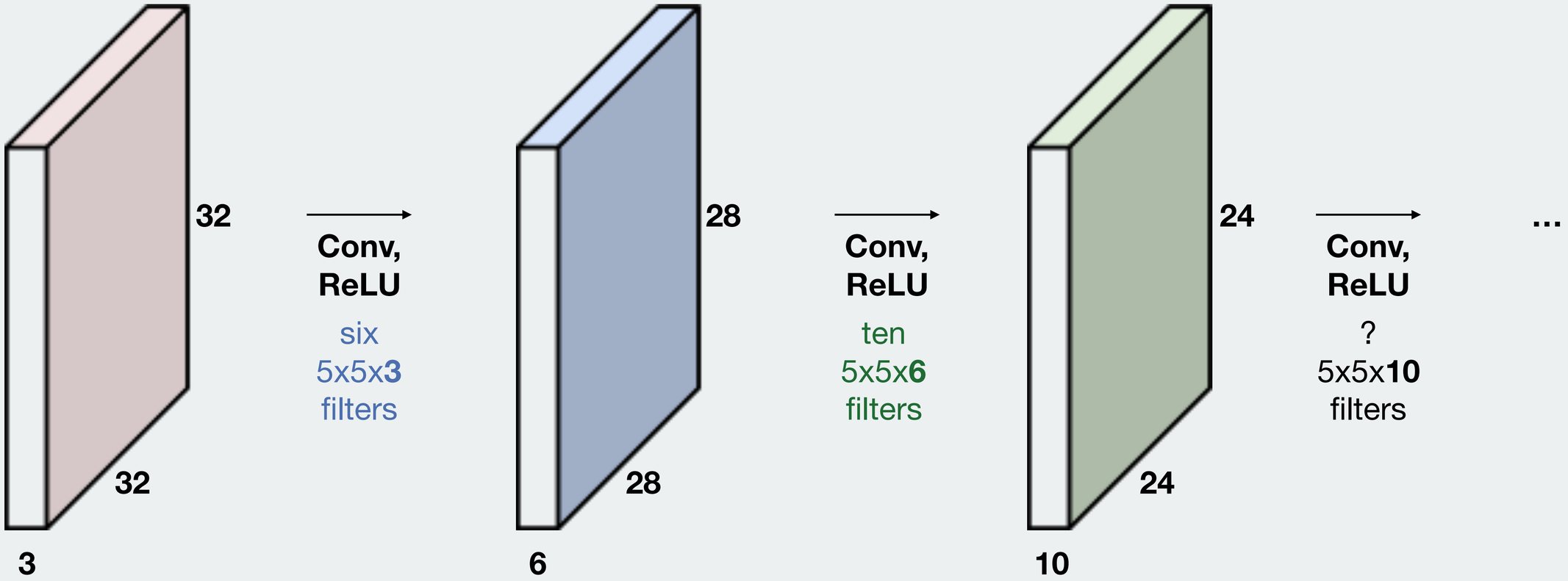


Convolution layer



Convolutional Neural Networks

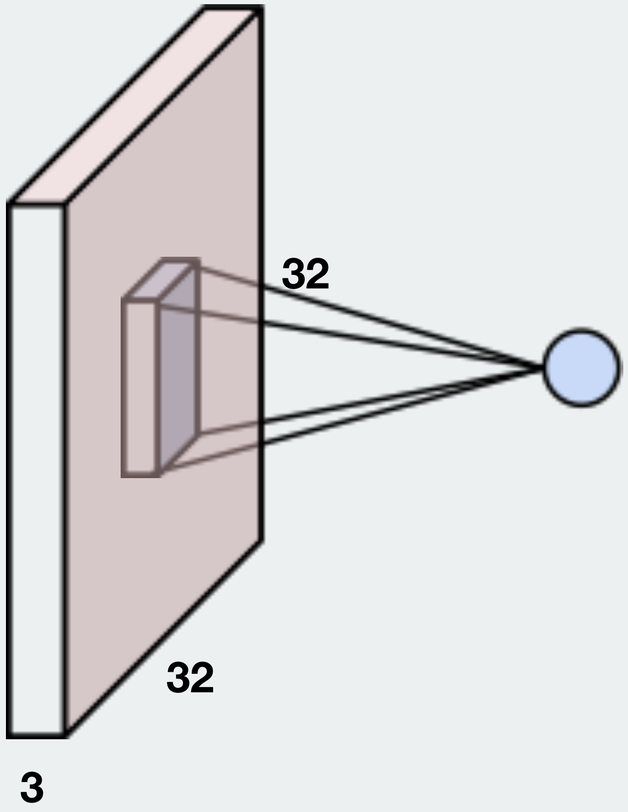
> Stack these operations



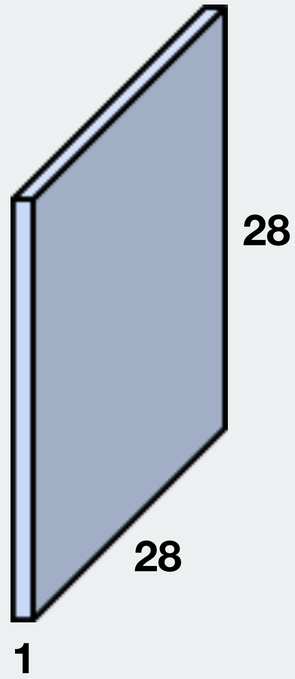
Convolutional Neural Networks

> Dimensions

Input *image*
32 x 32 x 3



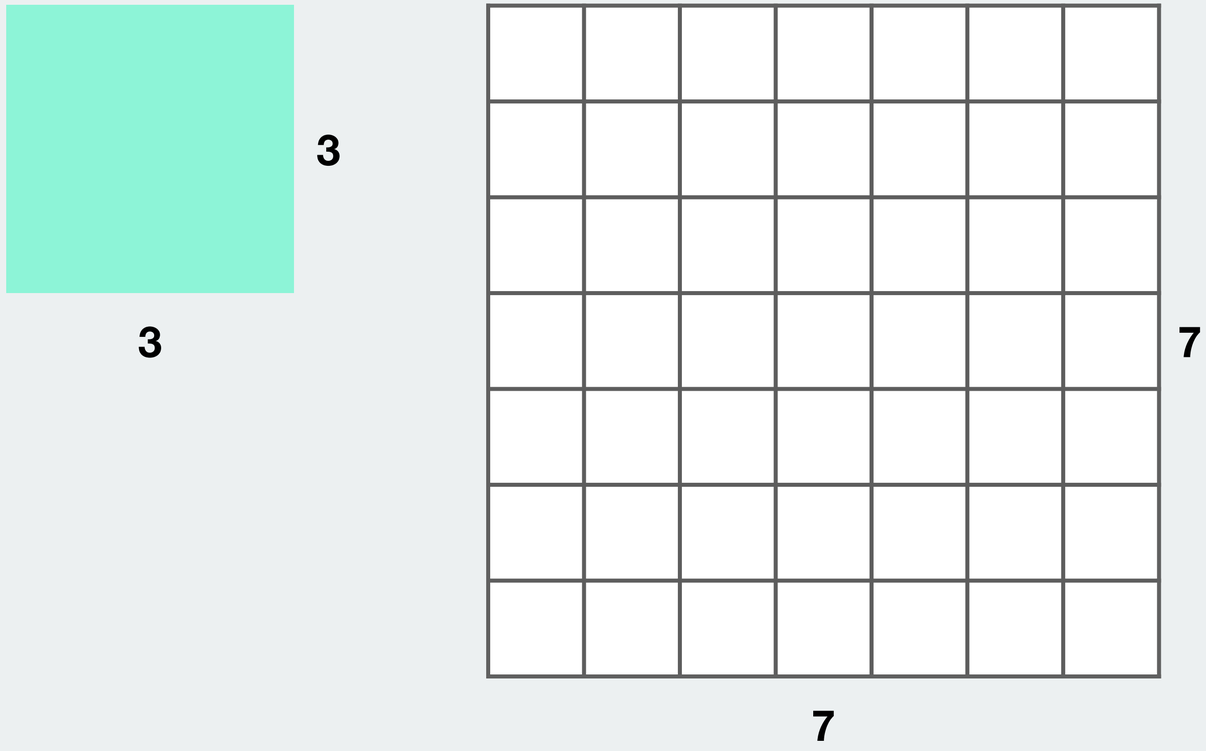
Convolve over all
spatial locations



Convolutional Neural Networks

> Dimensions

Example: 7 x 7 input
3 x 3 filter



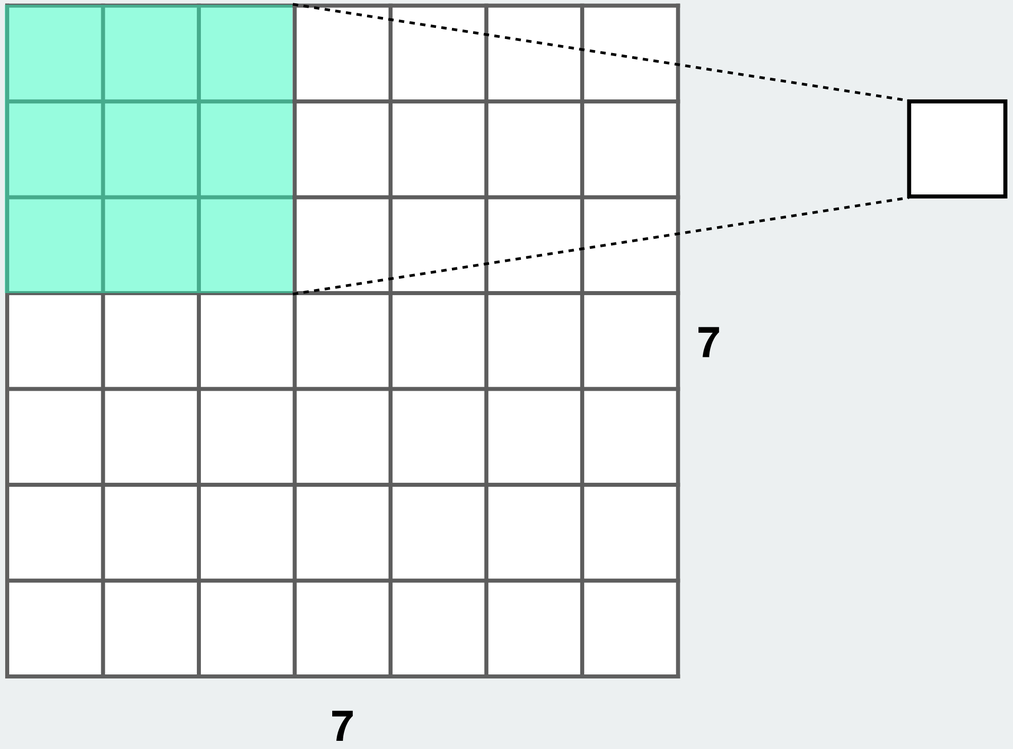
Question: What is the dimensions of the resulting activation map?

Convolutional Neural Networks

> Dimensions

Example: 7 x 7 input
3 x 3 filter

Question: What is the dimensions of the resulting activation map?

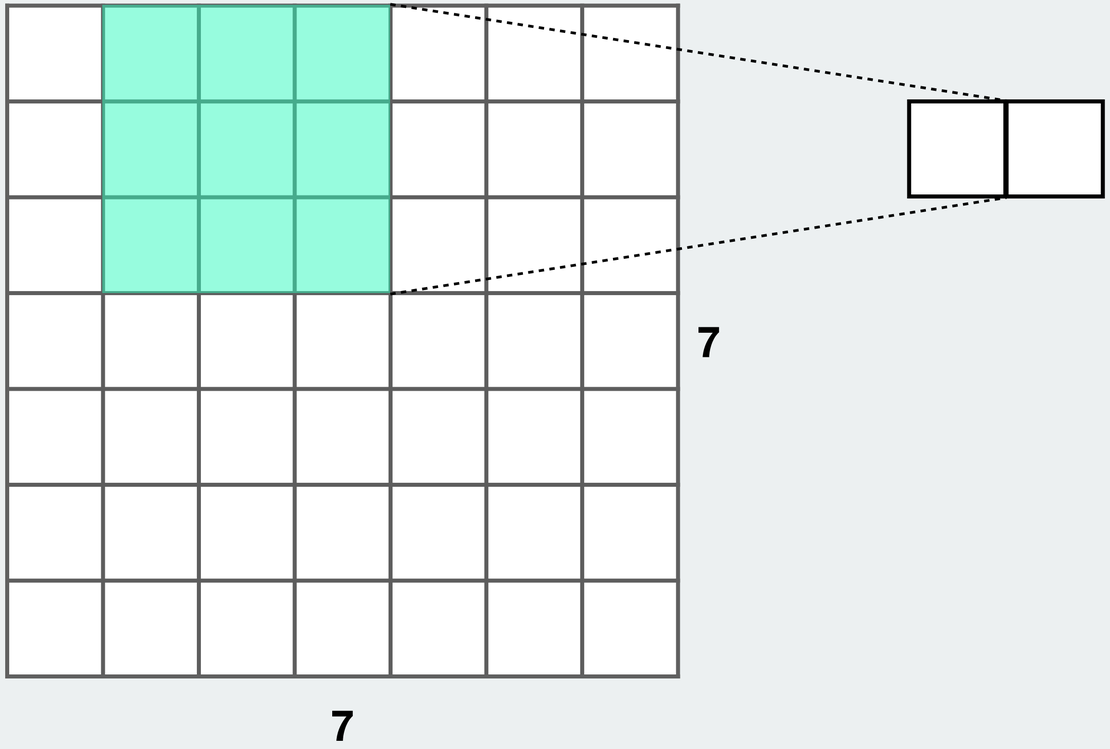


Convolutional Neural Networks

> Dimensions

Example: 7 x 7 input
3 x 3 filter

Question: What is the dimensions of the resulting activation map?

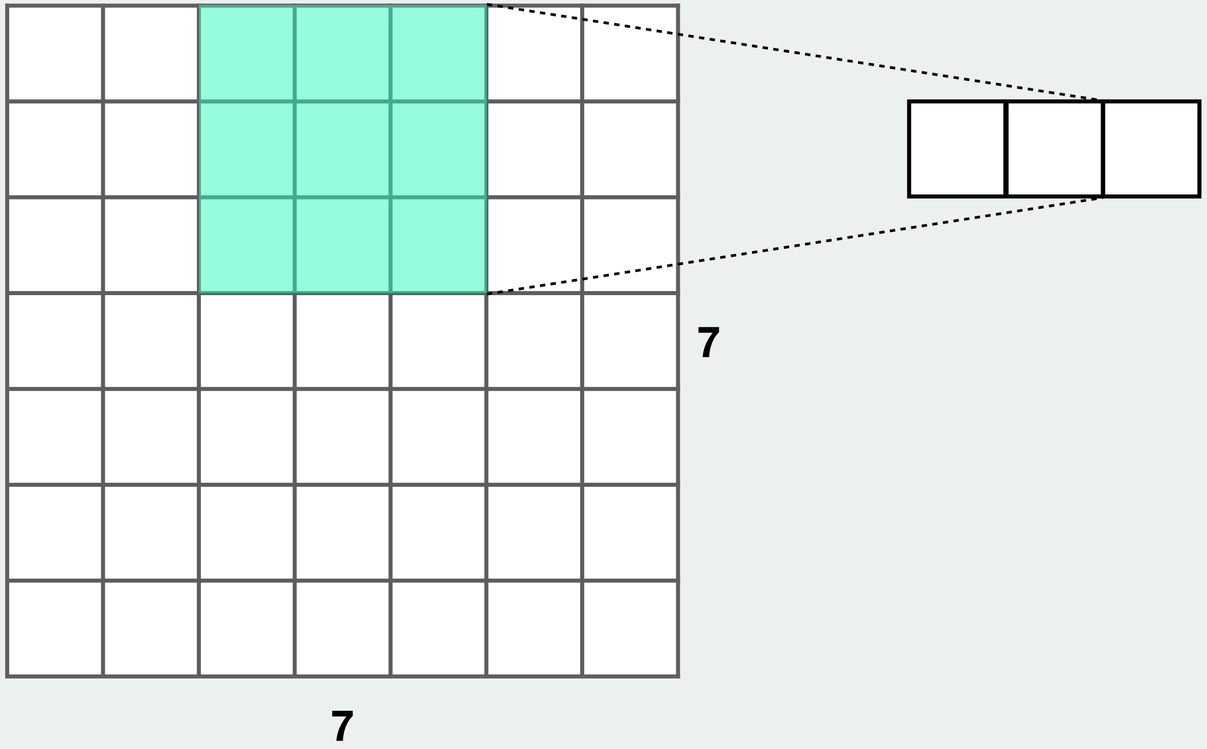


Convolutional Neural Networks

> Dimensions

Example: 7 x 7 input
3 x 3 filter

Question: What is the dimensions of the resulting activation map?

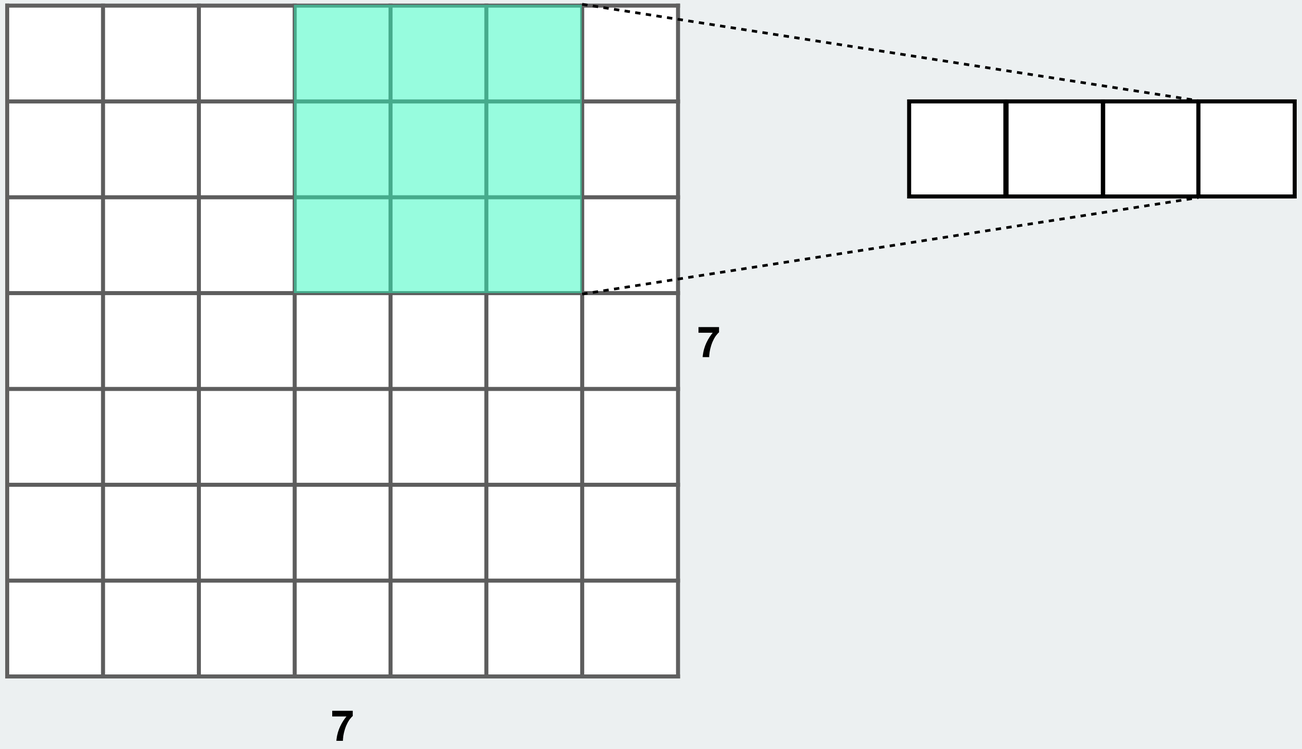


Convolutional Neural Networks

> Dimensions

Example: 7 x 7 input
3 x 3 filter

Question: What is the dimensions of the resulting activation map?

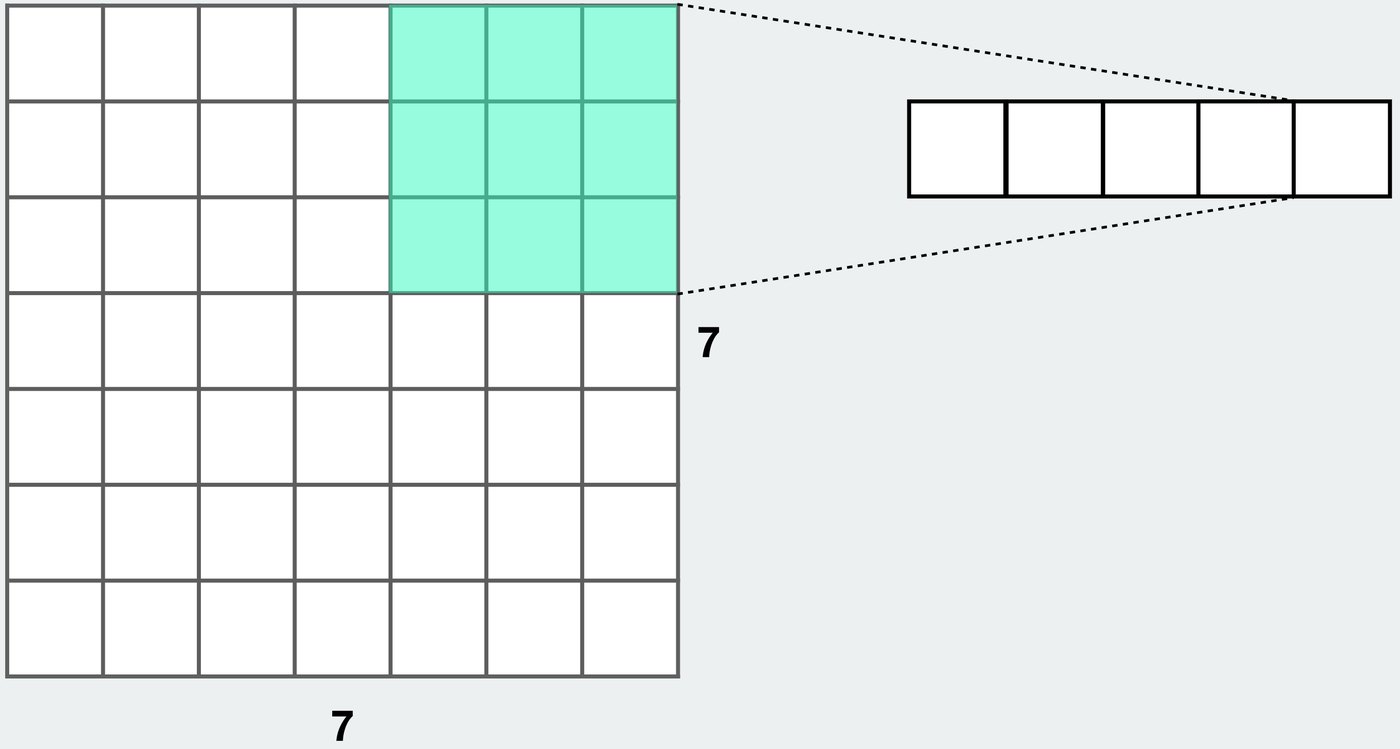


Convolutional Neural Networks

> Dimensions

Example: 7 x 7 input
3 x 3 filter

Question: What is the dimensions of the resulting activation map?

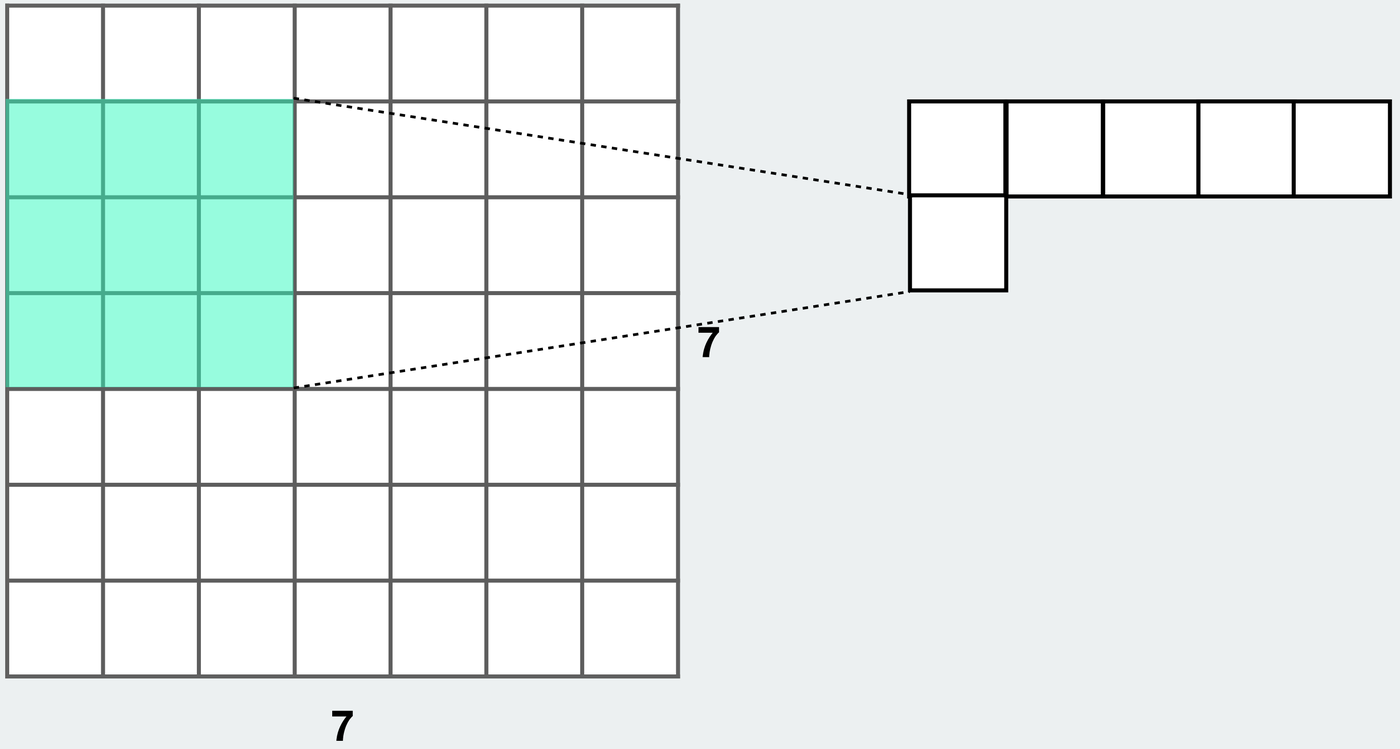


Convolutional Neural Networks

> Dimensions

Example: 7 x 7 input
3 x 3 filter

Question: What is the dimensions of the resulting activation map?

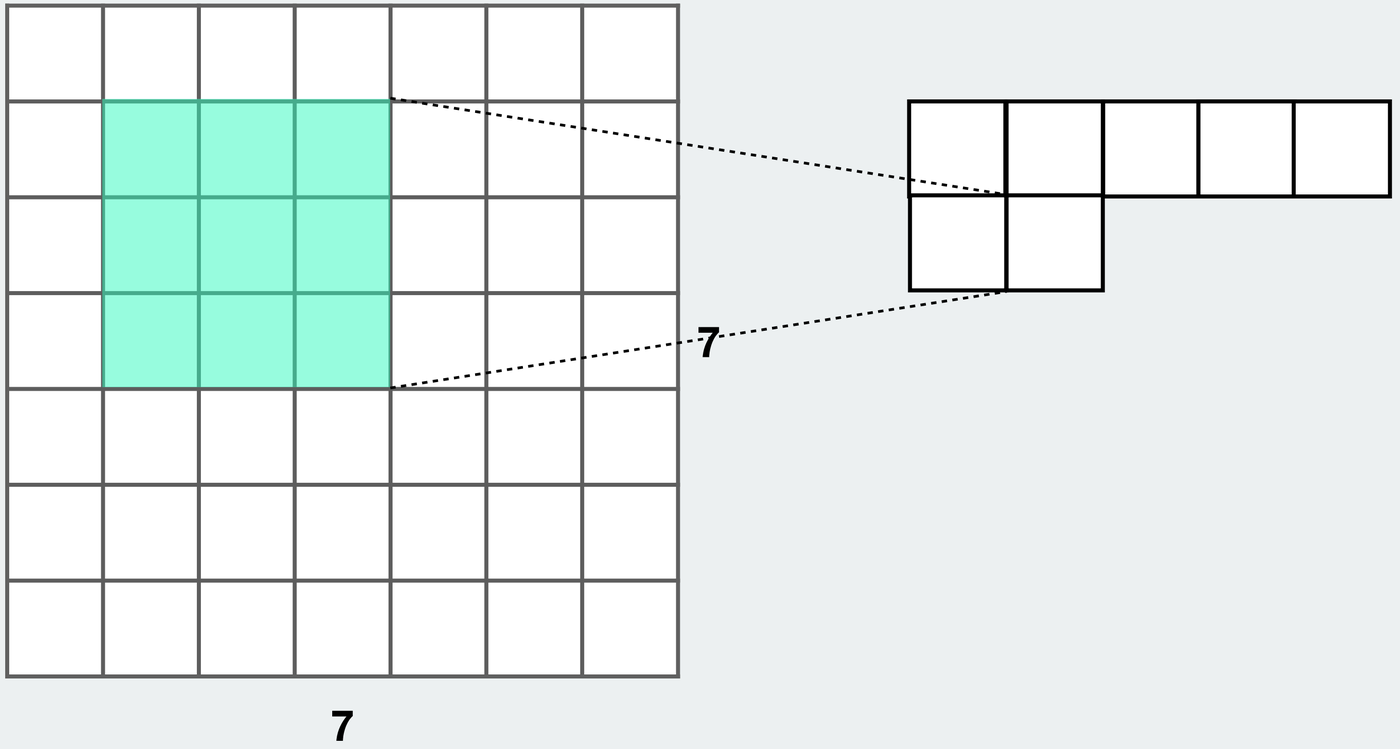


Convolutional Neural Networks

> Dimensions

Example: 7 x 7 input
3 x 3 filter

Question: What is the dimensions of the resulting activation map?



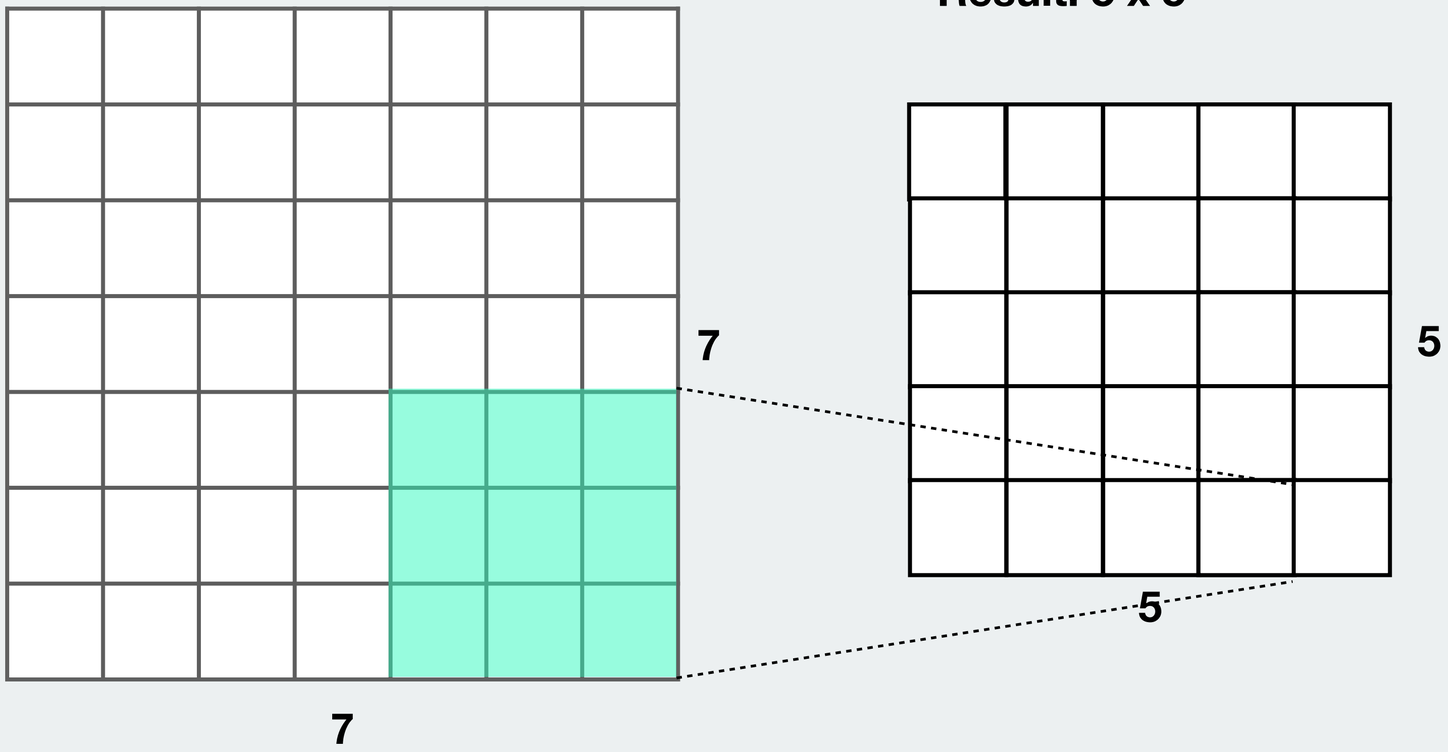
Convolutional Neural Networks

> Dimensions

Example: 7 x 7 input
3 x 3 filter

Question: What is the dimensions of the resulting activation map?

Result: 5 x 5

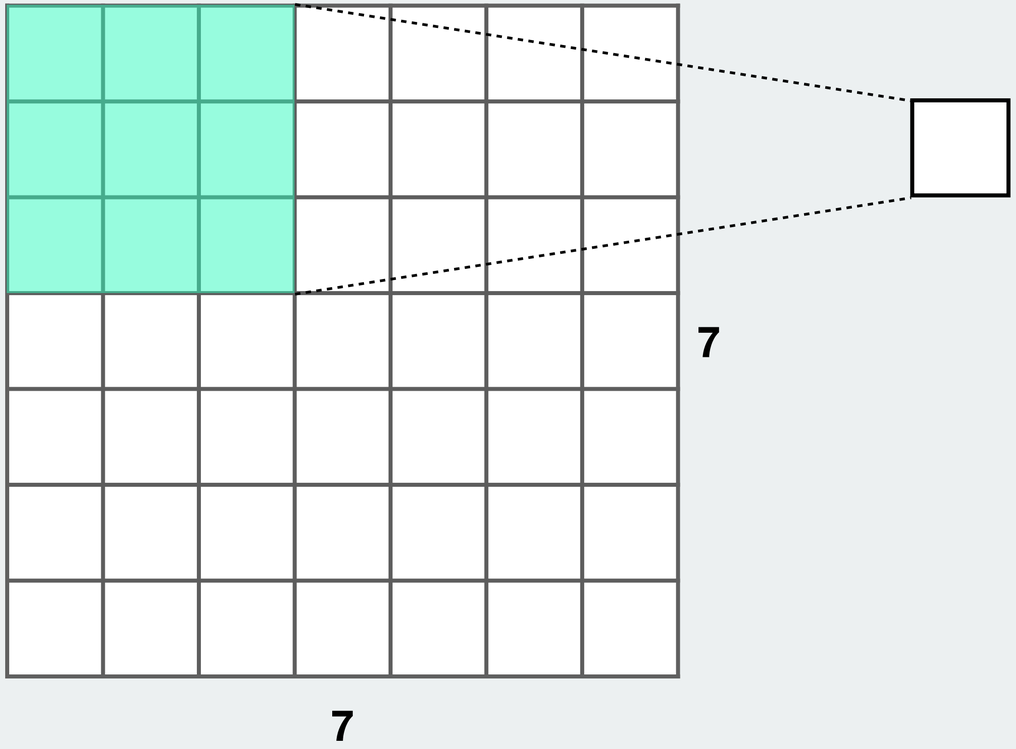


Convolutional Neural Networks

> Dimensions

Example: 7 x 7 input
3 x 3 filter

Question: What if we use stride 2?

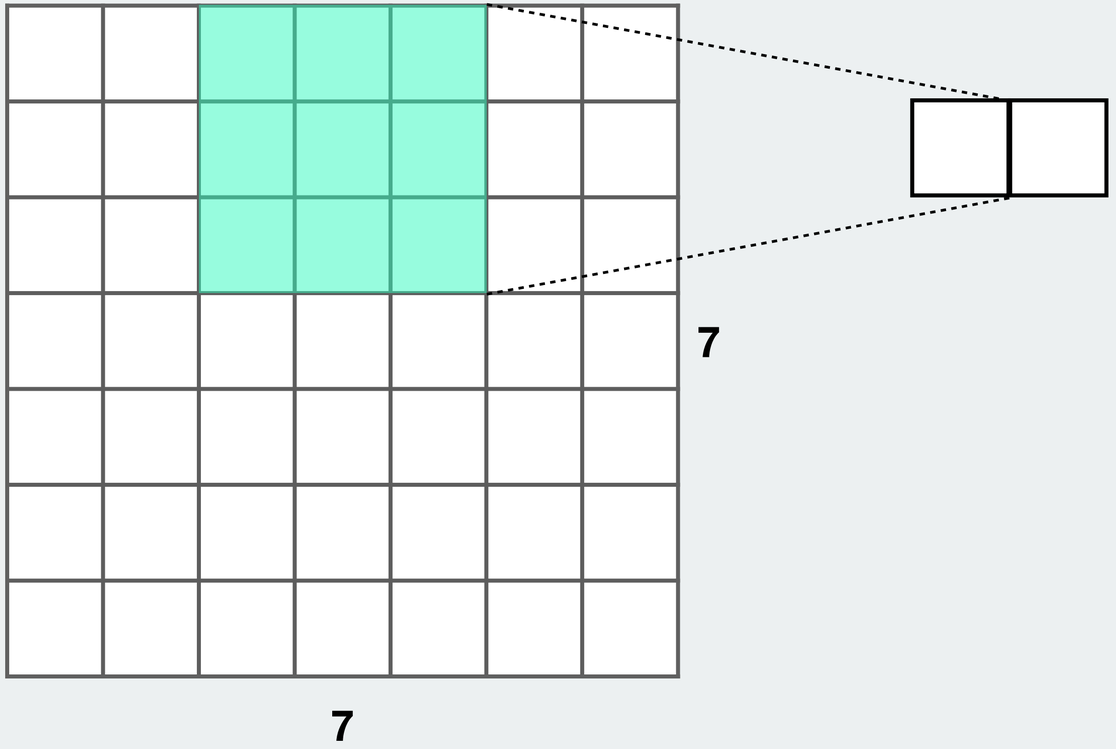


Convolutional Neural Networks

> Dimensions

Example: 7 x 7 input
3 x 3 filter

Question: What if we use stride 2?



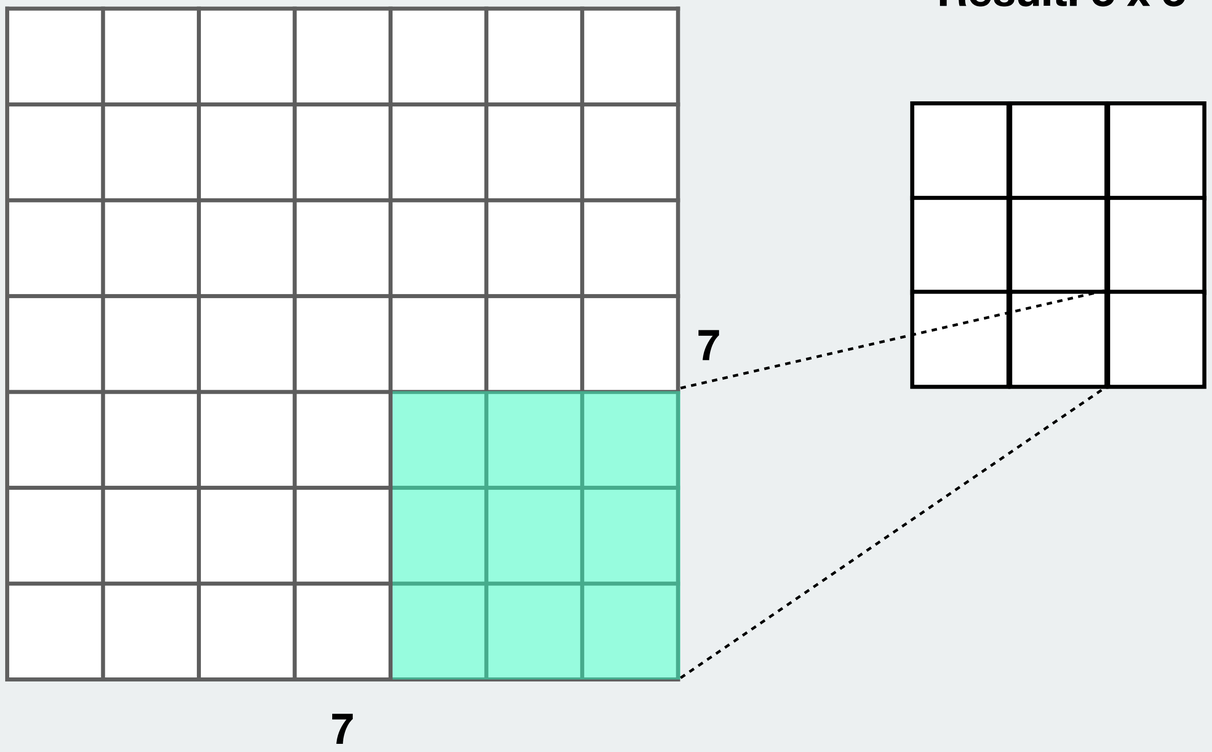
Convolutional Neural Networks

> Dimensions

Example: 7 x 7 input
3 x 3 filter

Question: What if we use stride 2?

Result: 3 x 3

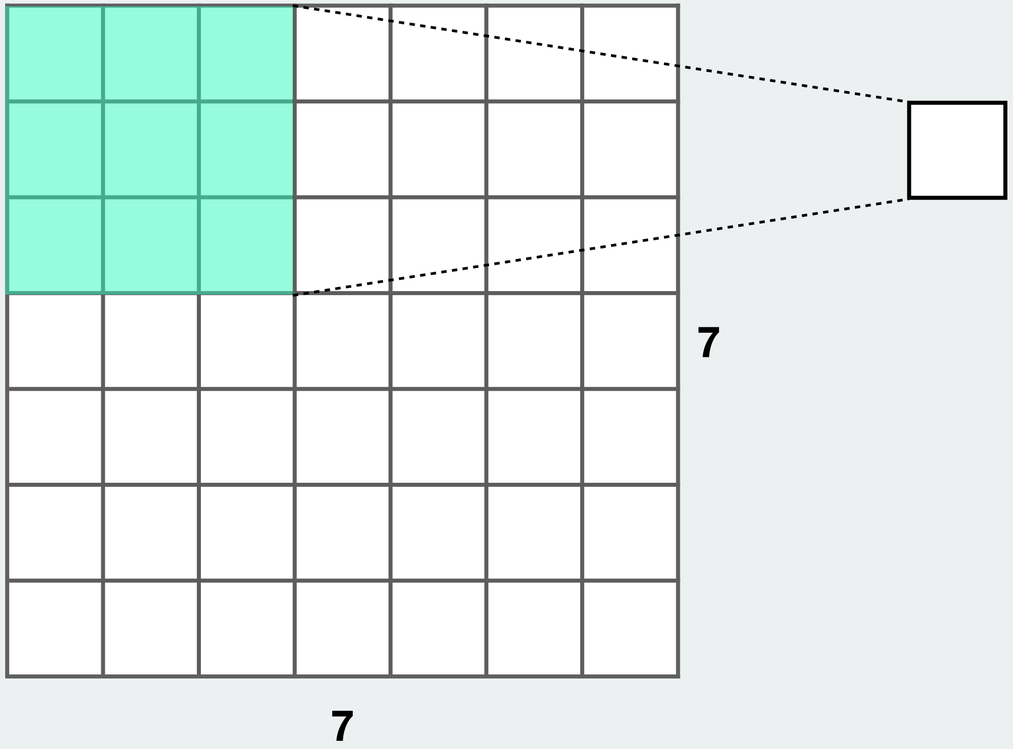


Convolutional Neural Networks

> Dimensions

Example: 7 x 7 input
3 x 3 filter

Question: Stride 3?

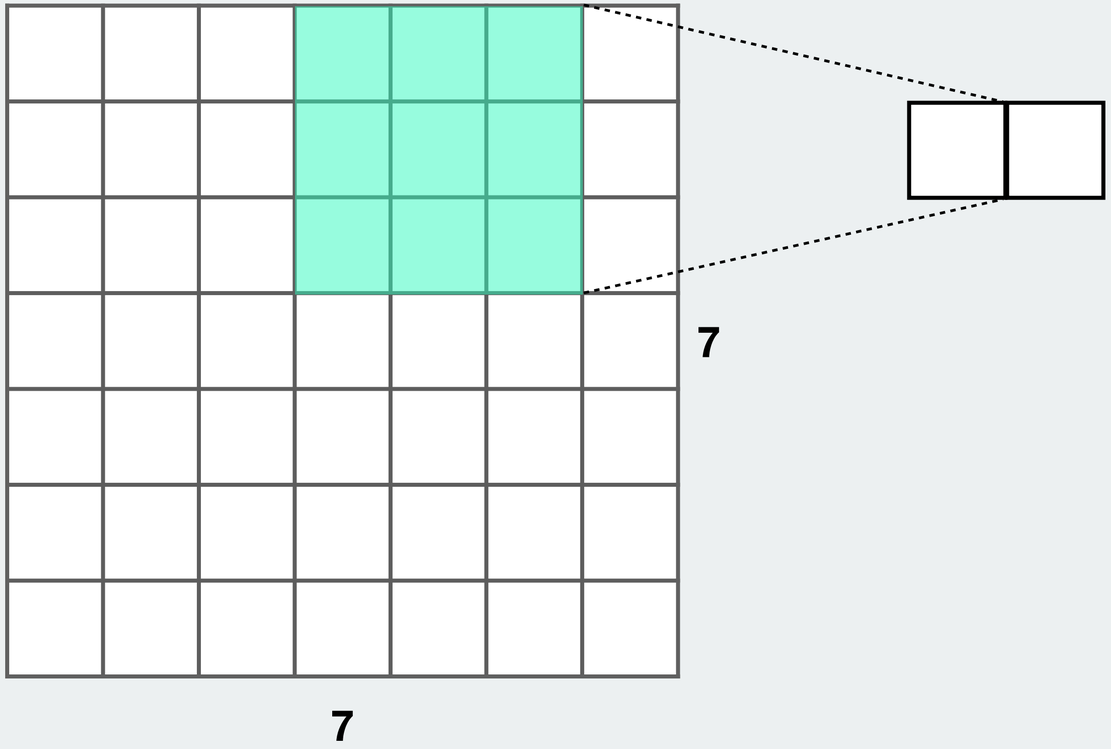


Convolutional Neural Networks

> Dimensions

Example: 7 x 7 input
3 x 3 filter

Question: Stride 3?

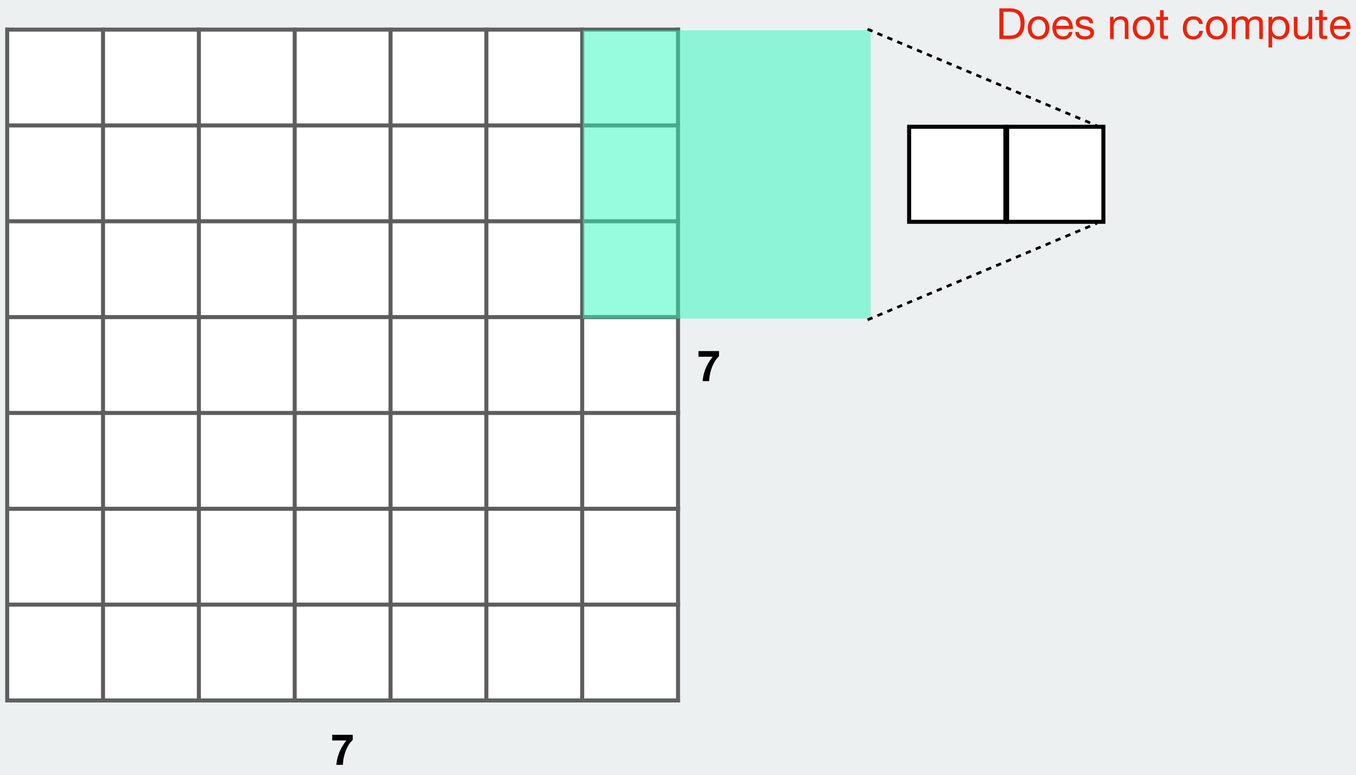


Convolutional Neural Networks

> Dimensions

Example: 7 x 7 input
3 x 3 filter

Question: Stride 3?

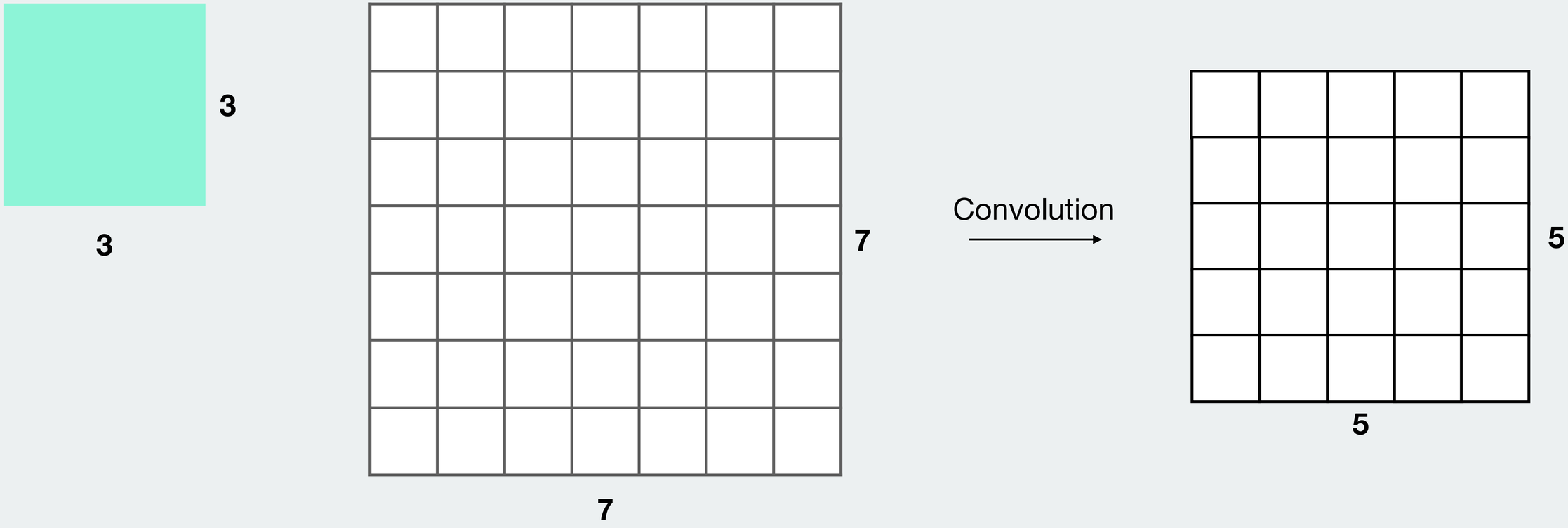


Convolutional Neural Networks

> Dimensions

Problem: The image *shrinks*

7 x 7 => 5 x 5



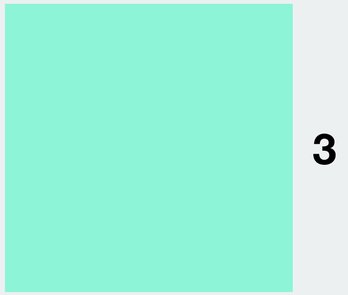
Convolutional Neural Networks

> Dimensions

Problem: The image *shrinks*

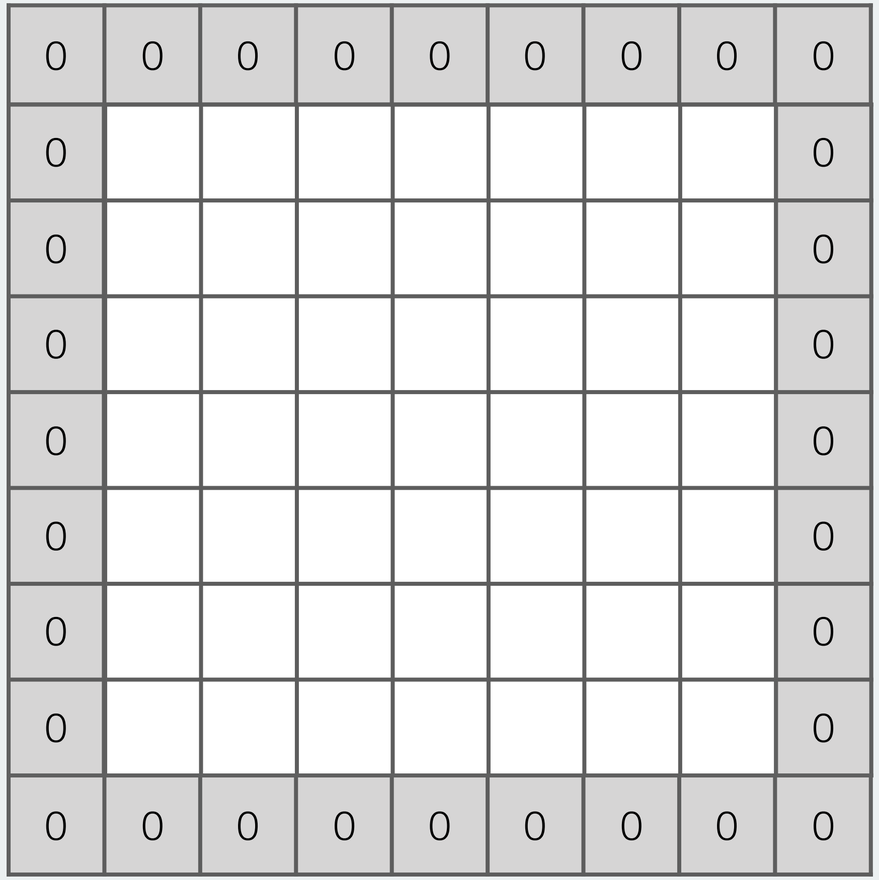
Solution: *Padding!*

7 x 7 => 5 x 5



3

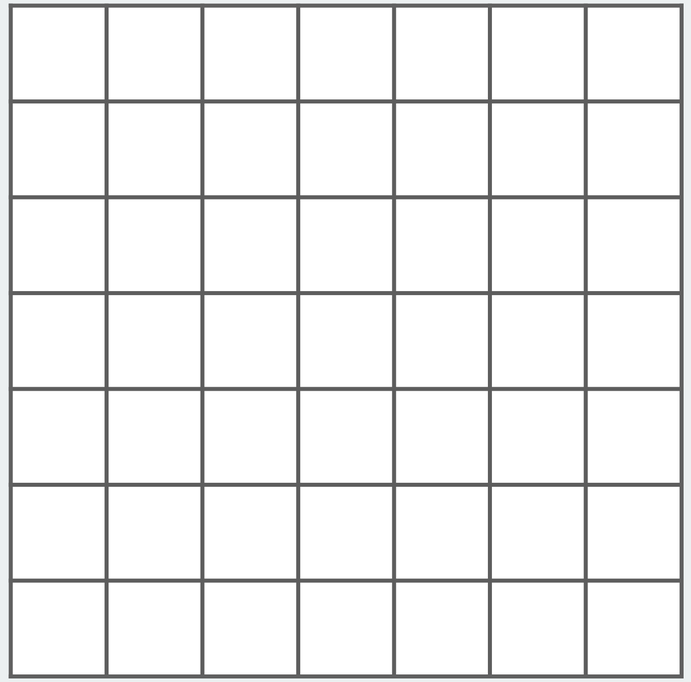
3



9

9

Convolution



7

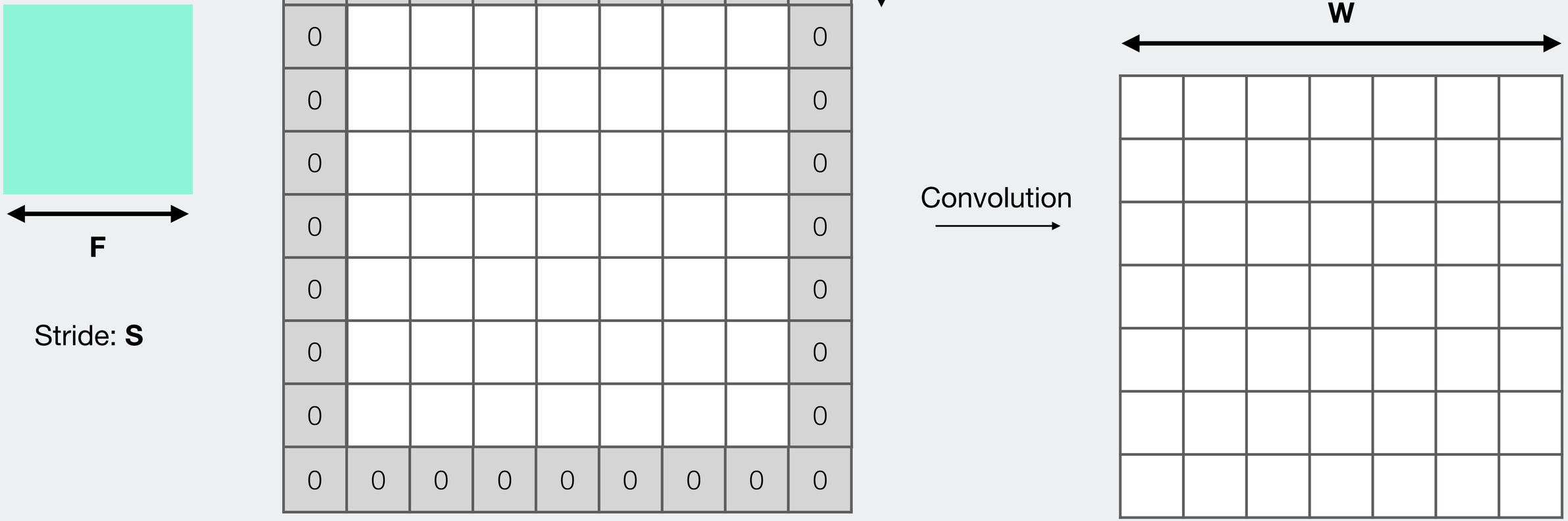
7

Convolutional Neural Networks

> Dimensions

General formula

$$W = \frac{N + 2P - F}{S} + 1$$



Animations

Convolutional Neural Networks

> Quiz time!

Given: 128 x 128 x 3 input

Ten 5 x 5 x 3 filters

Padding: 2

Stride: 1

Question 1: What are the output dimensions?

Question 2: What is the number of parameters?

Question 3: What if $F = N + 2P$?

Convolutional Neural Networks

> Quiz time!

Given: 128 x 128 x 3 input

Ten 5 x 5 x 3 filters

Padding: 2

Stride: 1

Question 1: What are the output dimensions?

Question 2: What is the number of parameters?

Question 3: What if $F = N + 2P$?

Answer 1:

$$W = \frac{128 + 2 \cdot 2 - 5}{1} + 1 = 128$$

Answer 2:

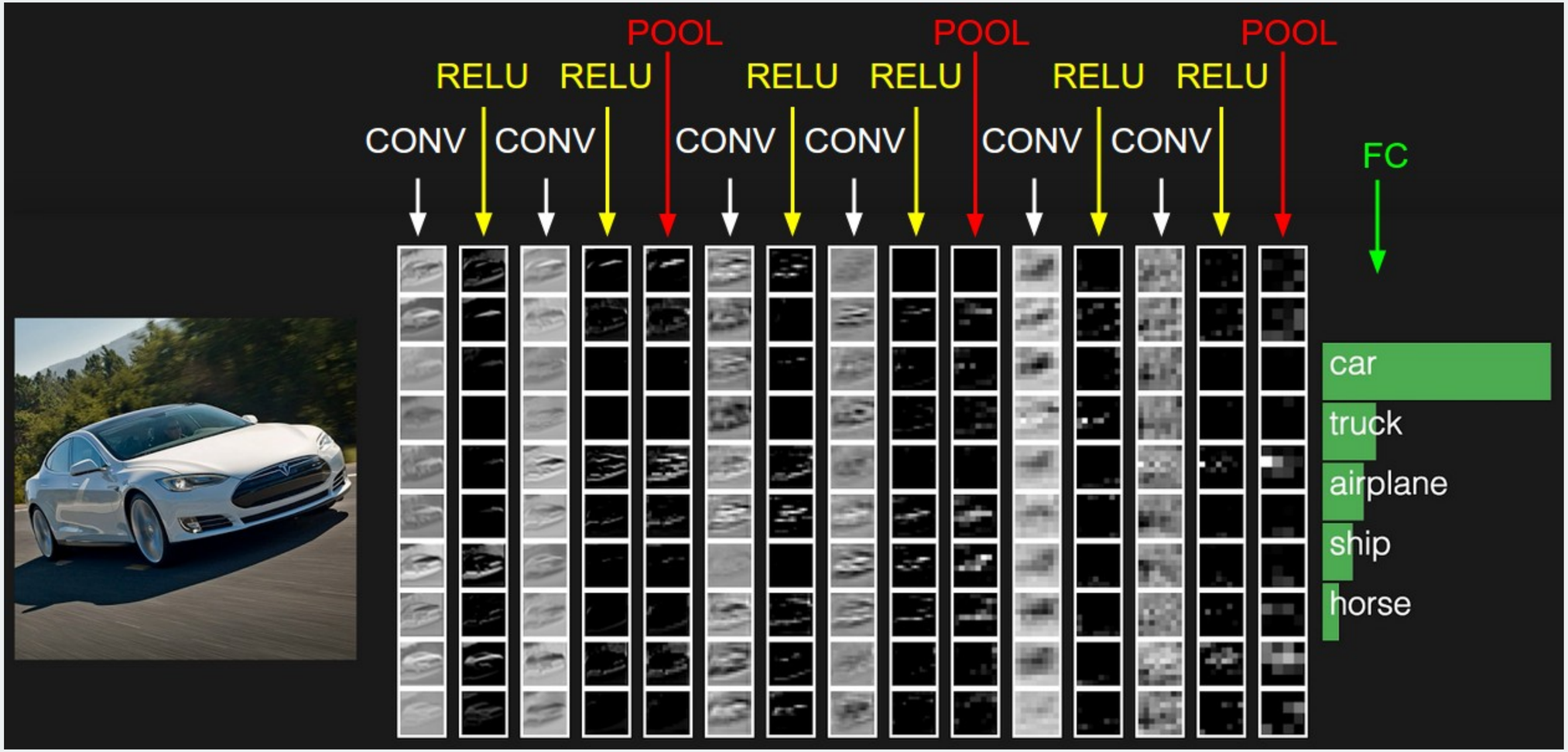
$$5 \cdot 5 \cdot 3 \cdot 10 + 10 = 760$$

Answer 3:

Then it's just a VNN!

Convolutional Neural Networks

> Real world example

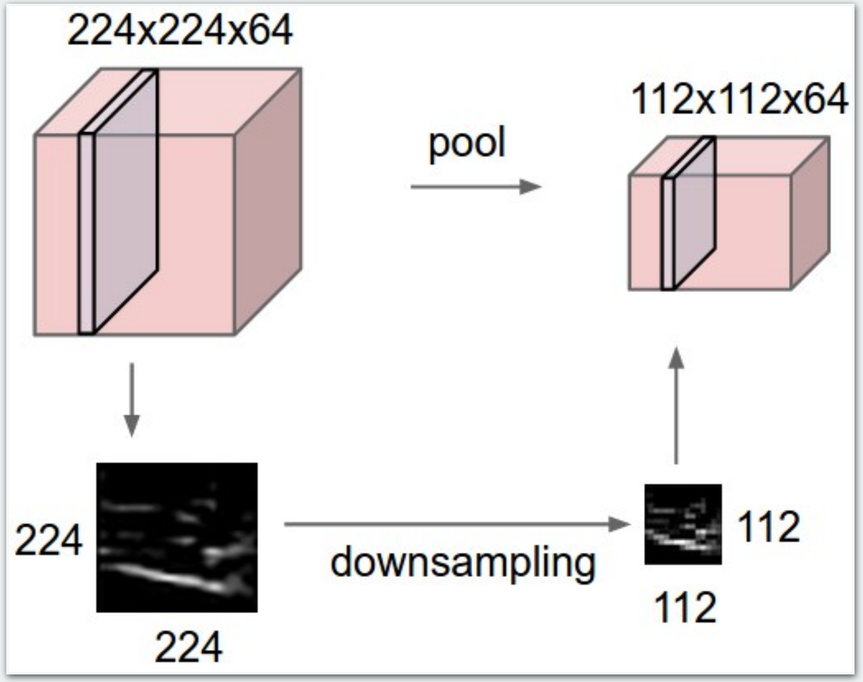


http://cs231n.github.io/convolutional-networks/

Convolutional Neural Networks

> Concept: *Pooling*

- Method used for downsampling
- Reduces number of parameters and computations
- Lowers width and height of volume by an integer factor
- Preserved depth

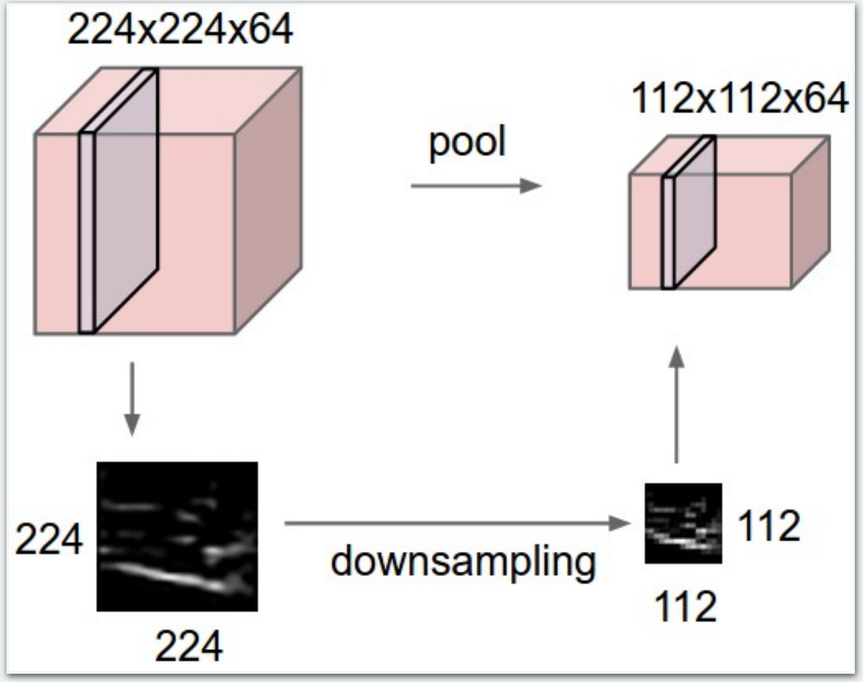


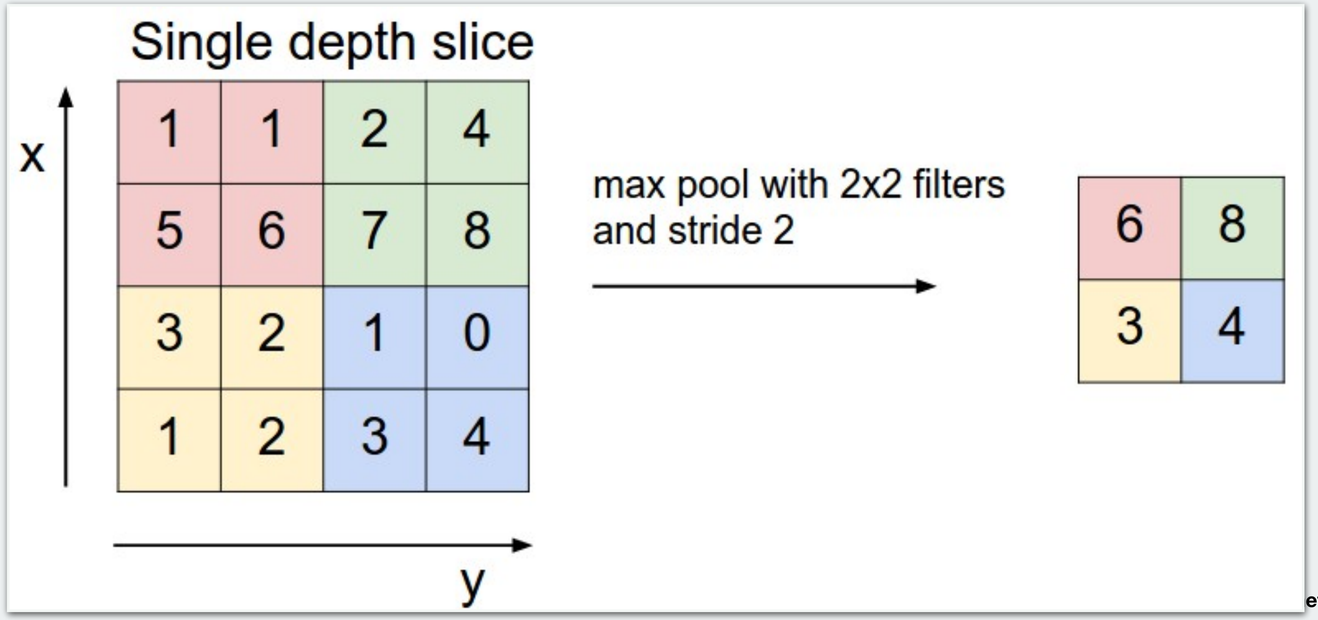
http://cs231n.github.io/convolutional-networks/

Convolutional Neural Networks

> Concept: *Pooling*

- Method used for downsampling
- Reduces number of parameters and computations
- Lowers width and height of volume by an integer factor
- Preserved depth





etworks/

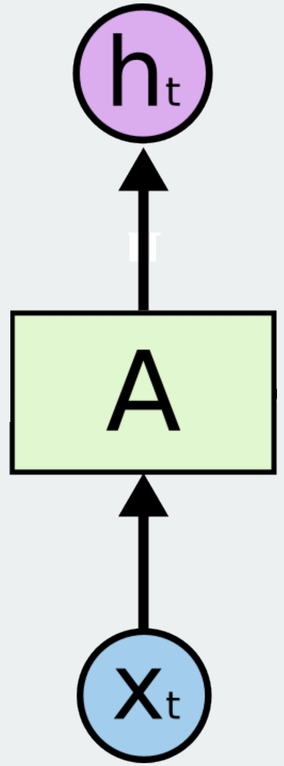
Recurrent Neural Networks

*Model architecture for **sequential data***

Recurrent Neural Networks

> In-built features of all feed forward neural networks

- Input and output must be of hard-assigned dimensions
- The network makes a fixed number of computations
- If input is sequence-like (video, sound, text, etc.) the network is ignorant to the order of samples

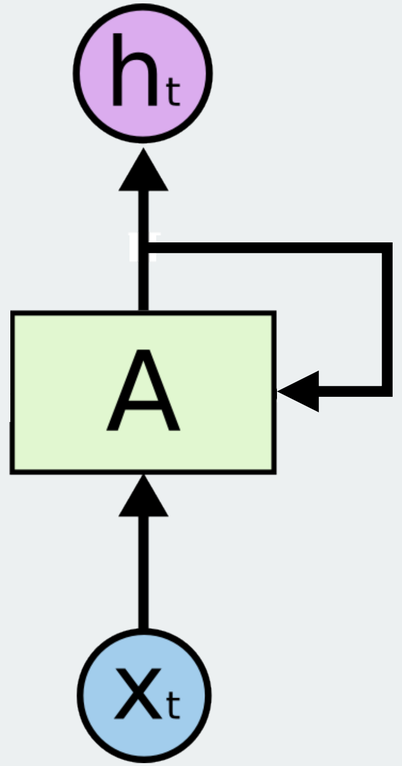


http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Recurrent Neural Networks

> Concept: *Recurrence*

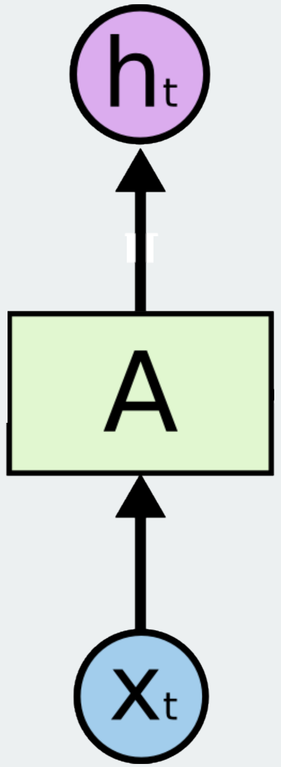
- Input and output must be of hard-assigned dimensions
- The network makes a fixed number of computations
- If input is sequence-like (video, sound, text, etc.) the network is ignorant to the order of samples



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Recurrent Neural Networks

> Fundamental idea of recurrence

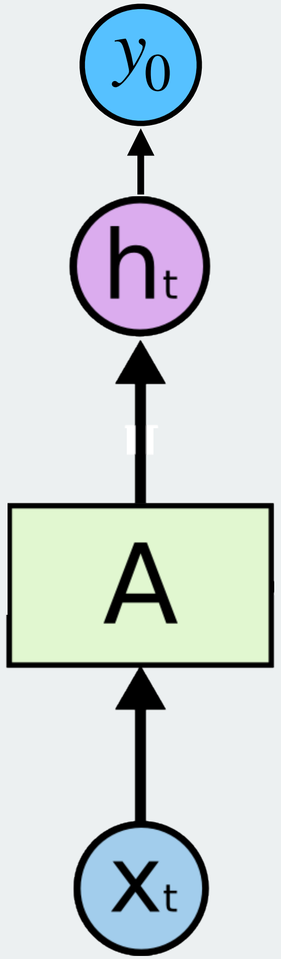


$$h_t = f_W(x_t)$$

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Recurrent Neural Networks

> Fundamental idea of recurrence



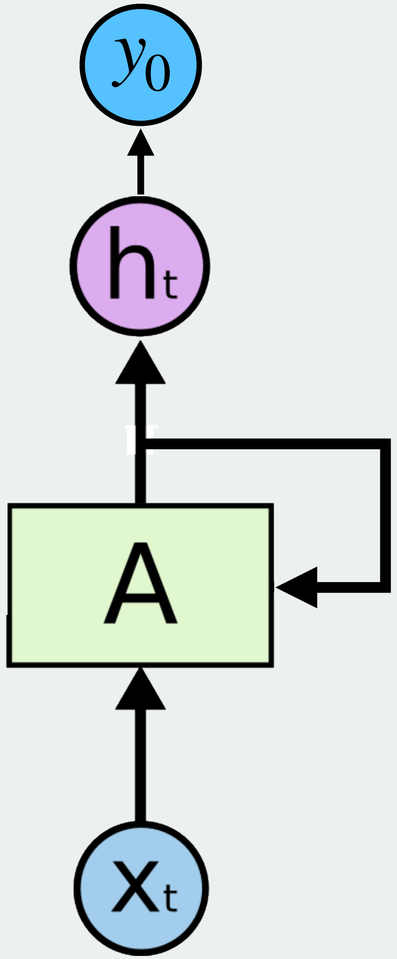
$$h_t = f_W(x_t)$$

$$y_t = W_{hy} h_t$$

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Recurrent Neural Networks

> Fundamental idea of recurrence



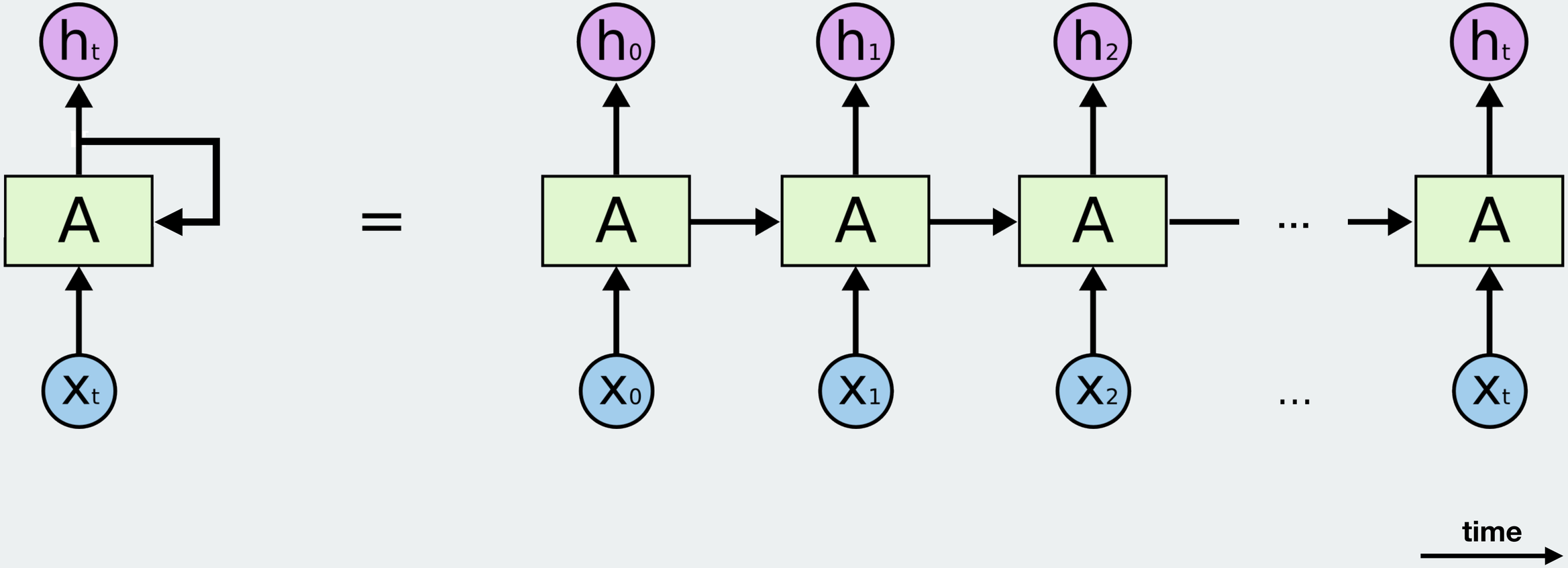
$$h_t = f_W(h_{t-1}, x_t)$$

$$y_t = W_{hy} h_t$$

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Recurrent Neural Networks

> Unrolled in time

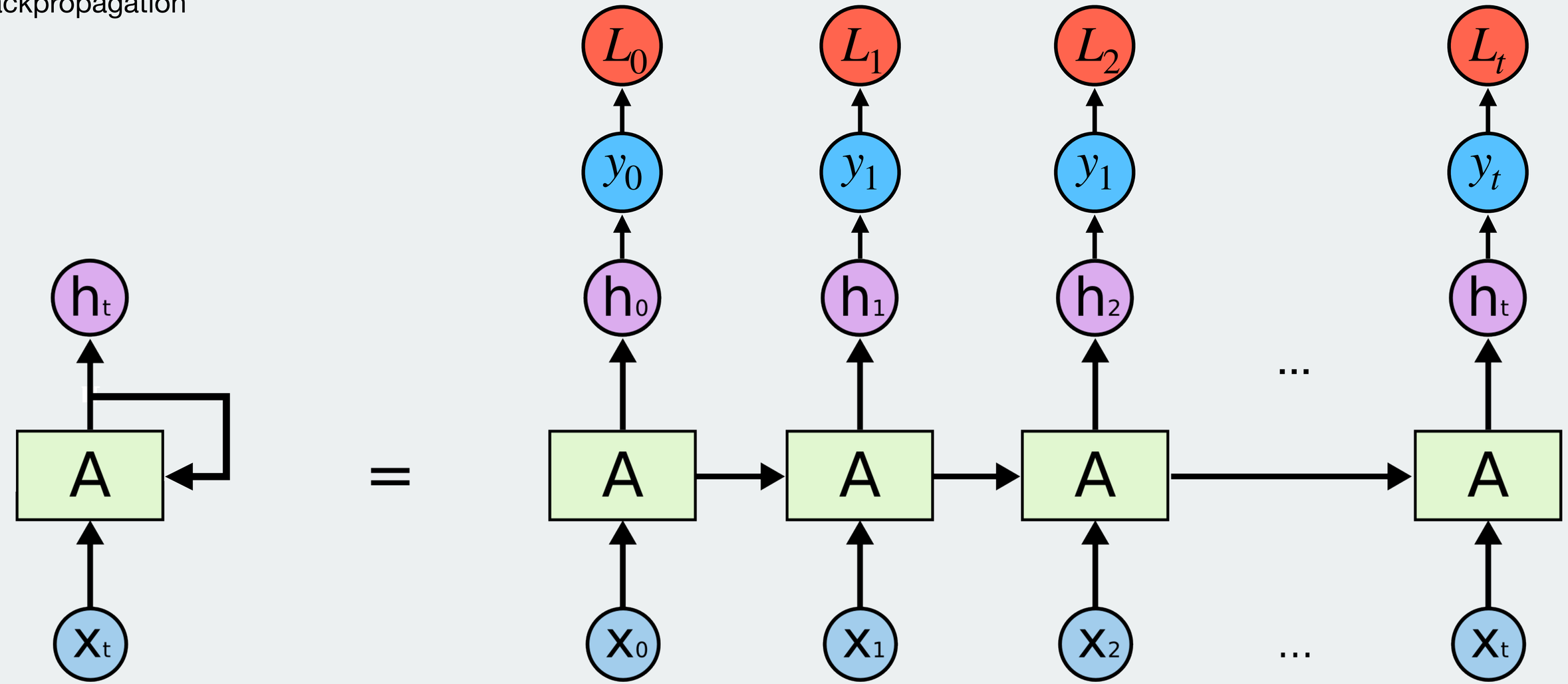


Notice: W is reused over time!

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Recurrent Neural Networks

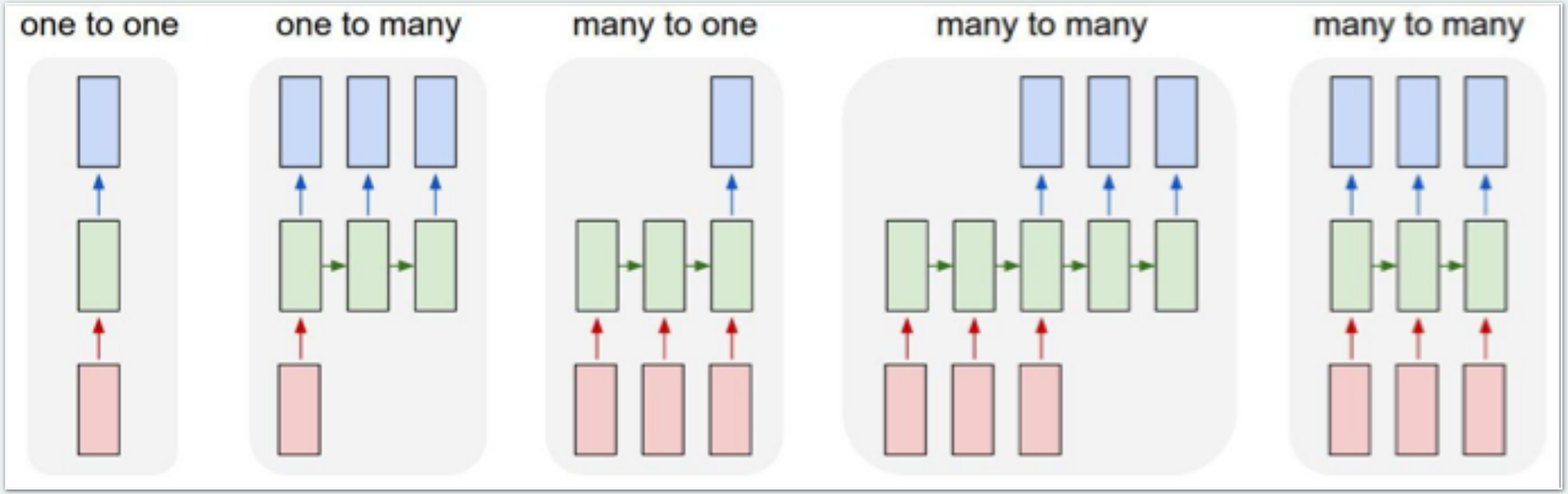
> Backpropagation



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Recurrent Neural Networks

> Ways to process sequential data

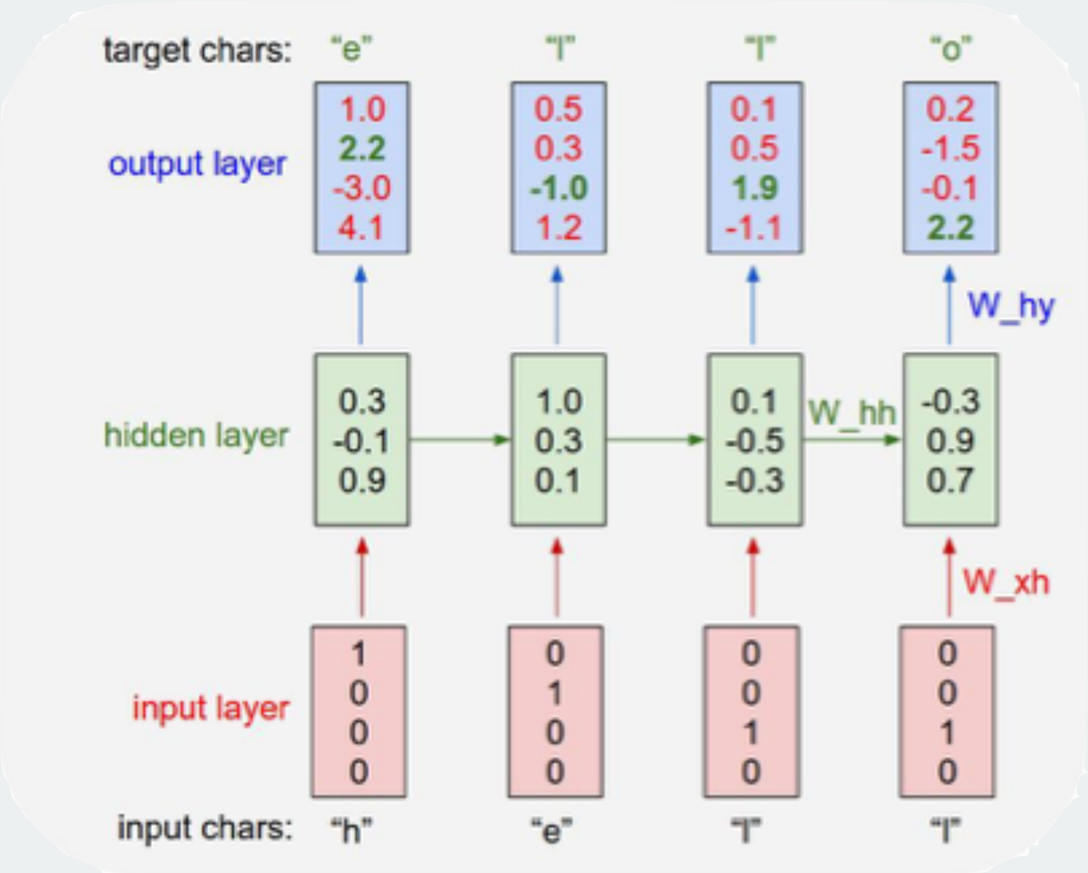


http://cs231n.stanford.edu/syllabus.html

Recurrent Neural Networks

> Example: predicting next character

training sequence**: “hello”**

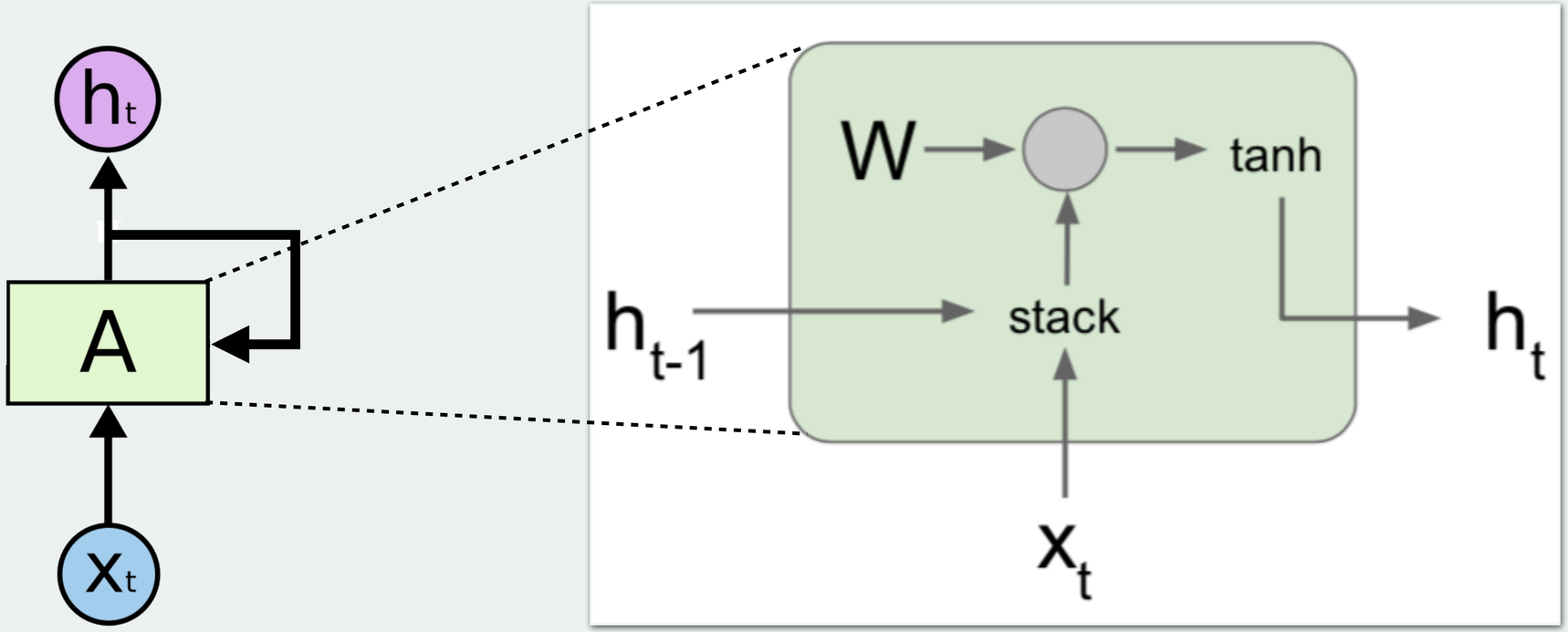


http://cs231n.stanford.edu/syllabus.html

Recurrent Neural Networks

> Vanilla RNN, architecture

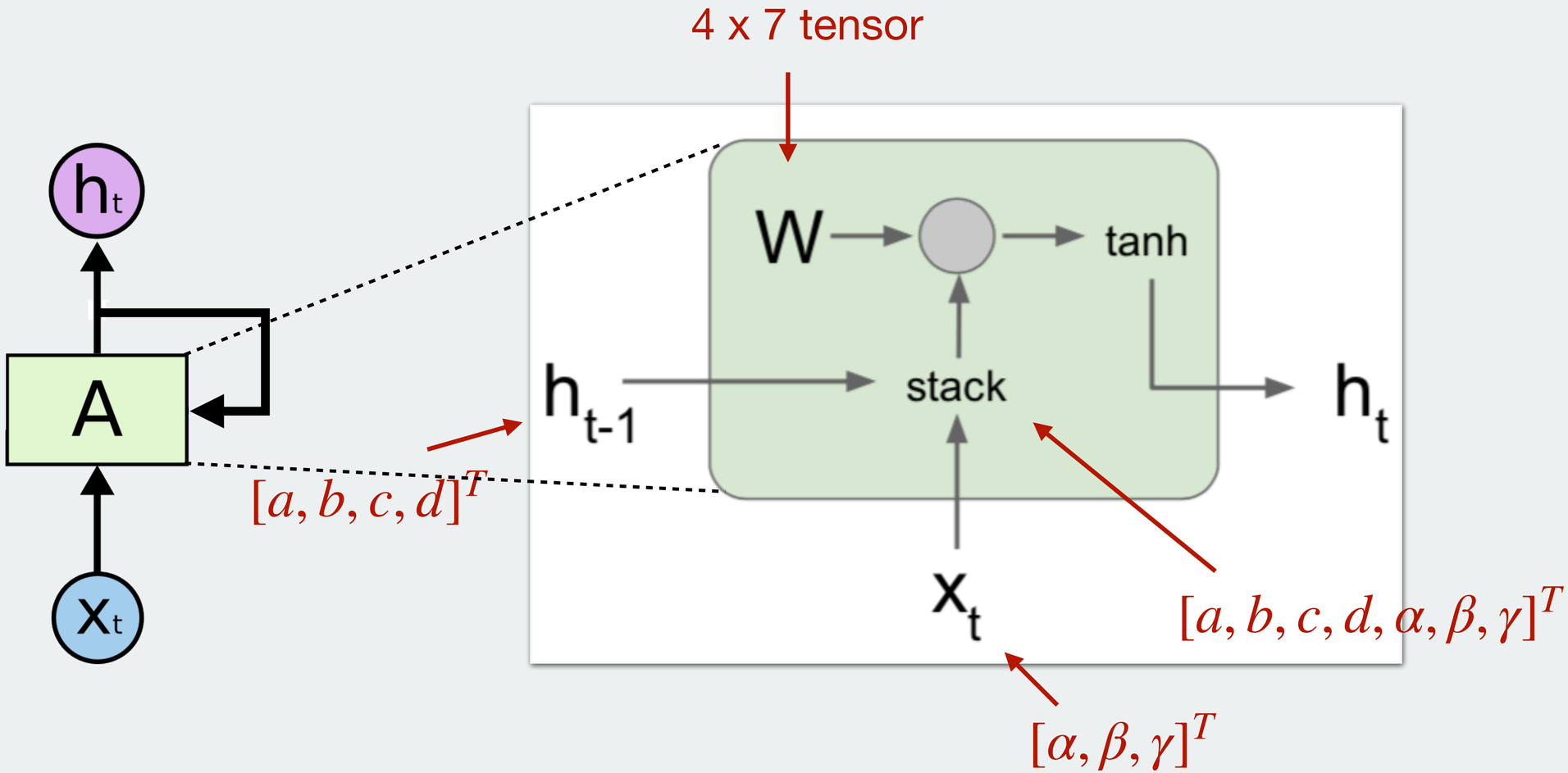
$$h_t = \tanh\left(W \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix} \right)$$



http://cs231n.stanford.edu/syllabus.html

Recurrent Neural Networks

> Vanilla RNN, architecture



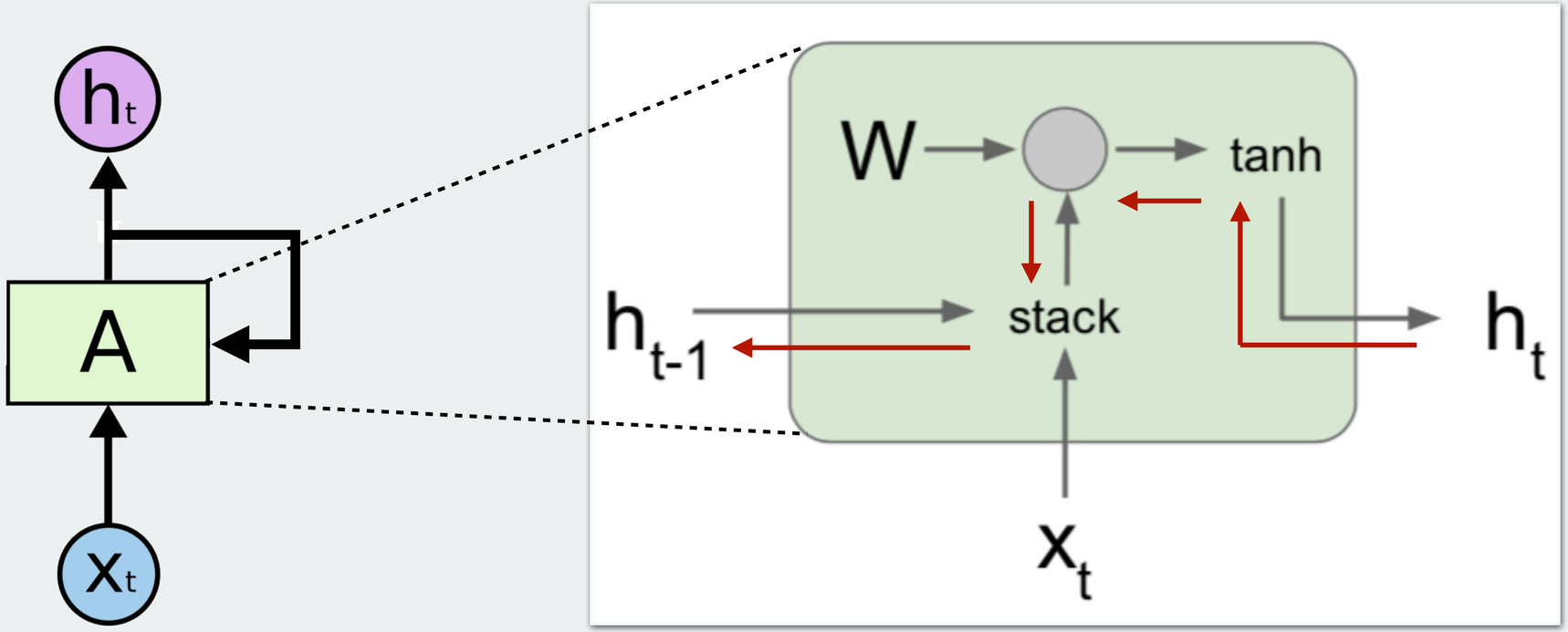
4: because dotting (h, t) onto it should result in a new vector with 4 elements

7: because the (h, t) vector we are dotting onto it has 7 elements in it

http://cs231n.stanford.edu/syllabus.html

Recurrent Neural Networks

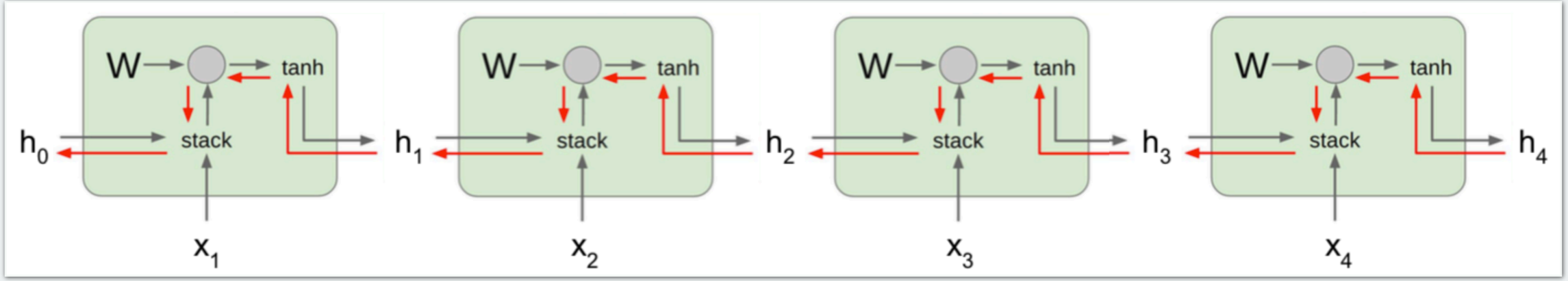
> Vanilla RNN, backpropagation



http://cs231n.stanford.edu/syllabus.html

Recurrent Neural Networks

> Vanilla RNN, backpropagation



Problem: Repeated multiplications by **W** during backpropagation

Leads to: Exploding/vanishing gradients

Solution: Gradient clipping (solves exploding gradients), or **change architecture**

http://cs231n.stanford.edu/syllabus.html

Recurrent Neural Networks

> Vanilla RNN vs. LSTM

Vanilla RNN

$$h_t = \tanh\left(W \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix} \right)$$

Long Short Term Memory (LSTM)

$$\begin{bmatrix} i \\ f \\ o \\ g \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} W \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix}$$

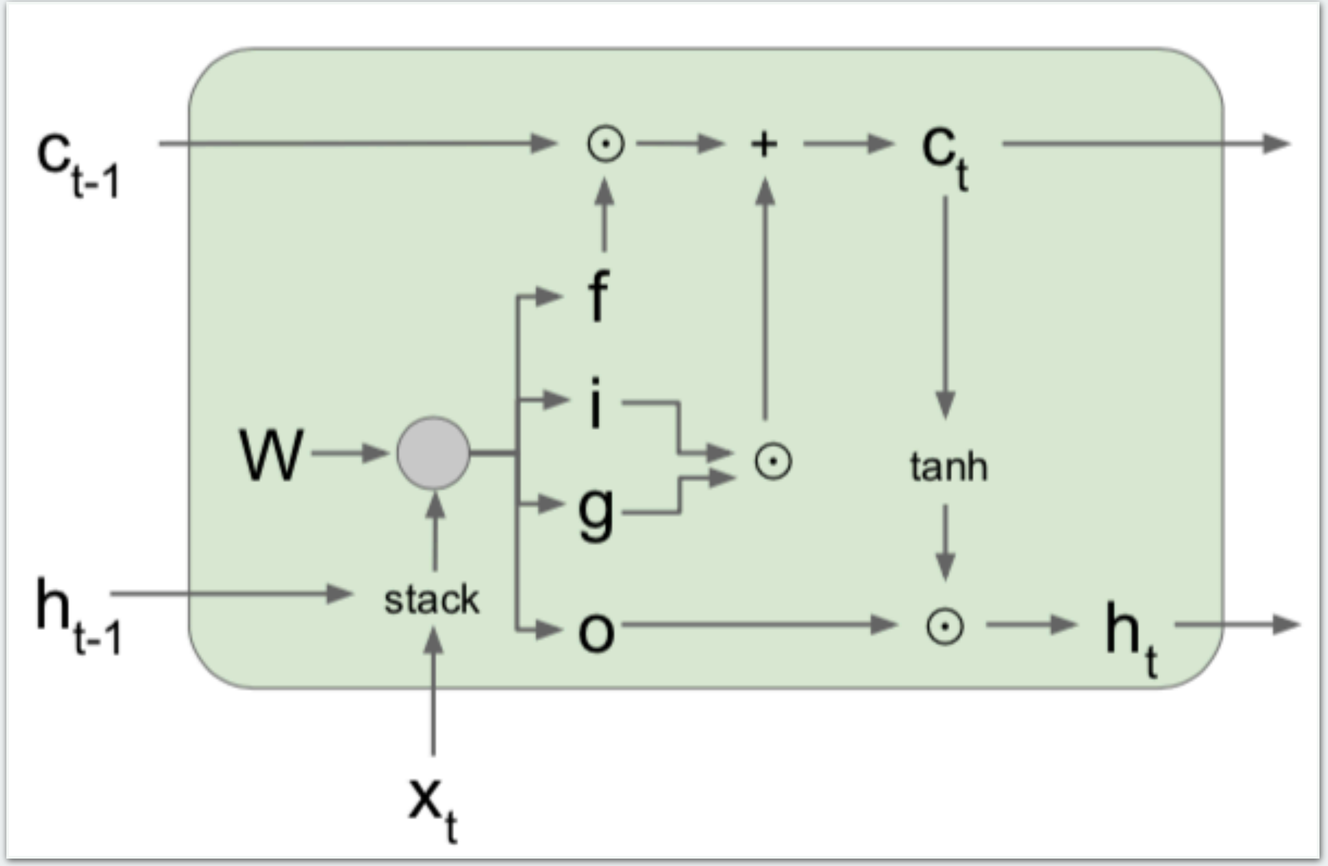
$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

http://cs231n.stanford.edu/syllabus.html

Recurrent Neural Networks

> Vanilla RNN vs. LSTM



Long Short Term Memory (LSTM)

$$\begin{bmatrix} i \\ f \\ o \\ g \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} W \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

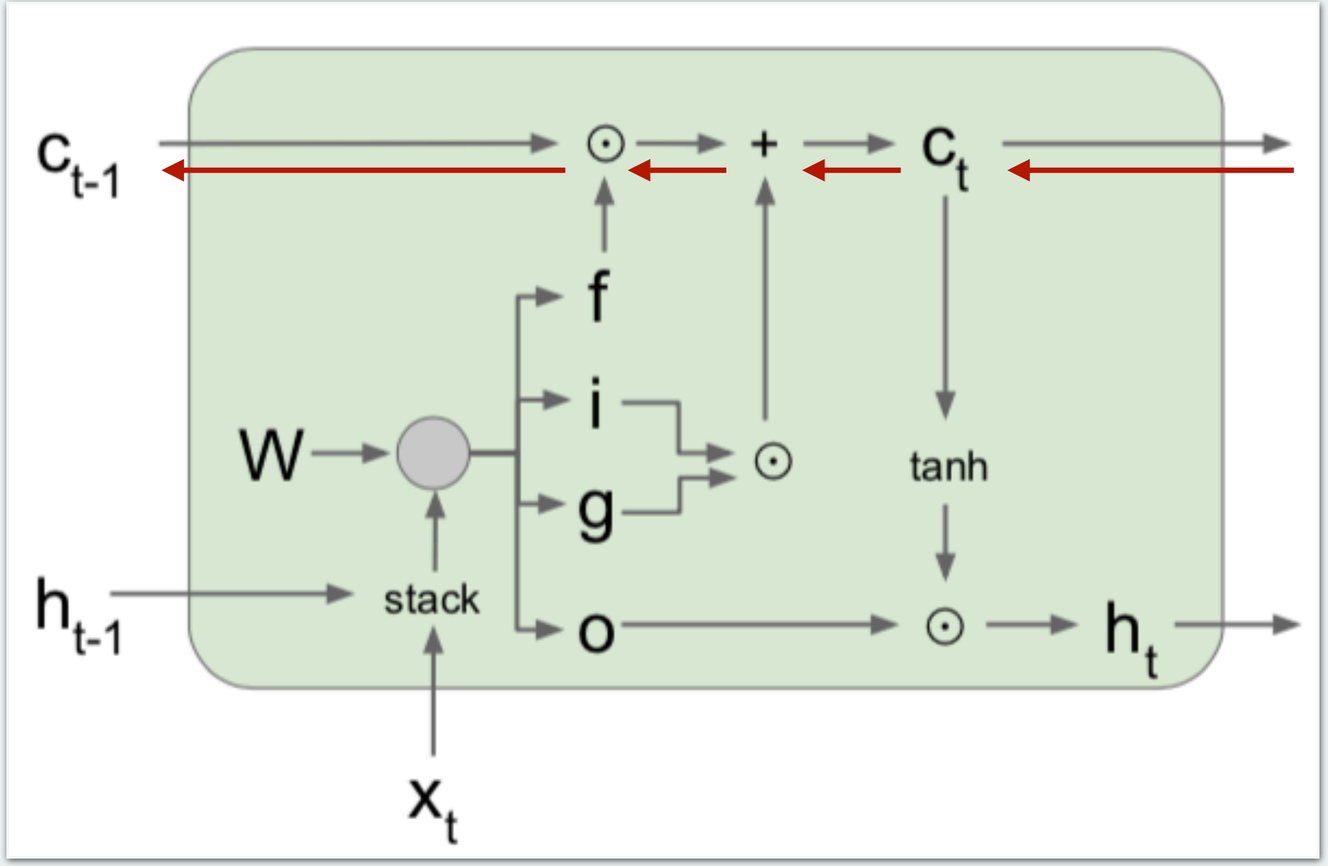
$$h_t = o \odot \tanh(c_t)$$

http://cs231n.stanford.edu/syllabus.html

Recurrent Neural Networks

> Vanilla RNN vs. LSTM

“Gradient highway”: solves vanishing/exploding gradient problems



Long Short Term Memory (LSTM)

$$\begin{bmatrix} i \\ f \\ o \\ g \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} W \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix}$$

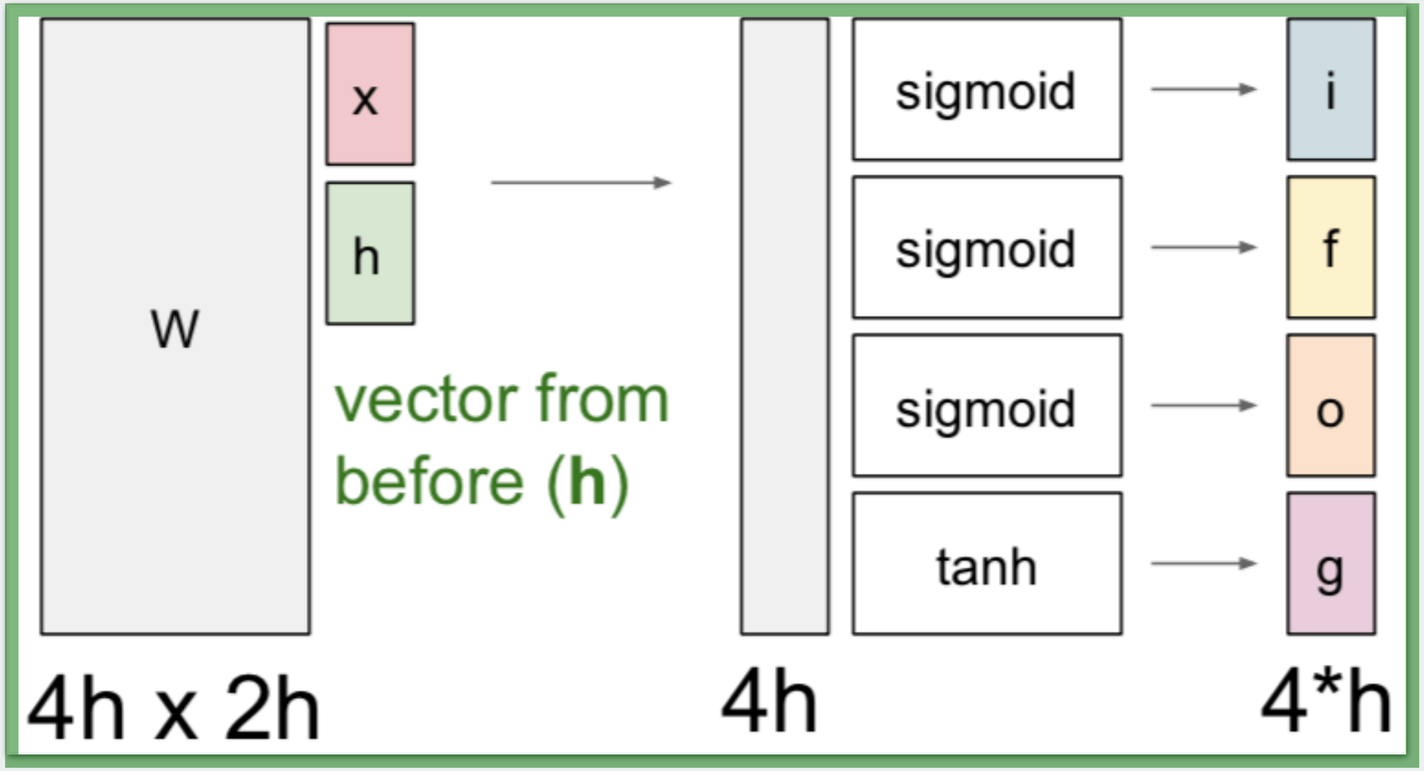
$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

http://cs231n.stanford.edu/syllabus.html

Recurrent Neural Networks

> Vanilla RNN vs. LSTM



Long Short Term Memory (LSTM)

$$\begin{bmatrix} i \\ f \\ o \\ g \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} W \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

http://cs231n.stanford.edu/syllabus.html

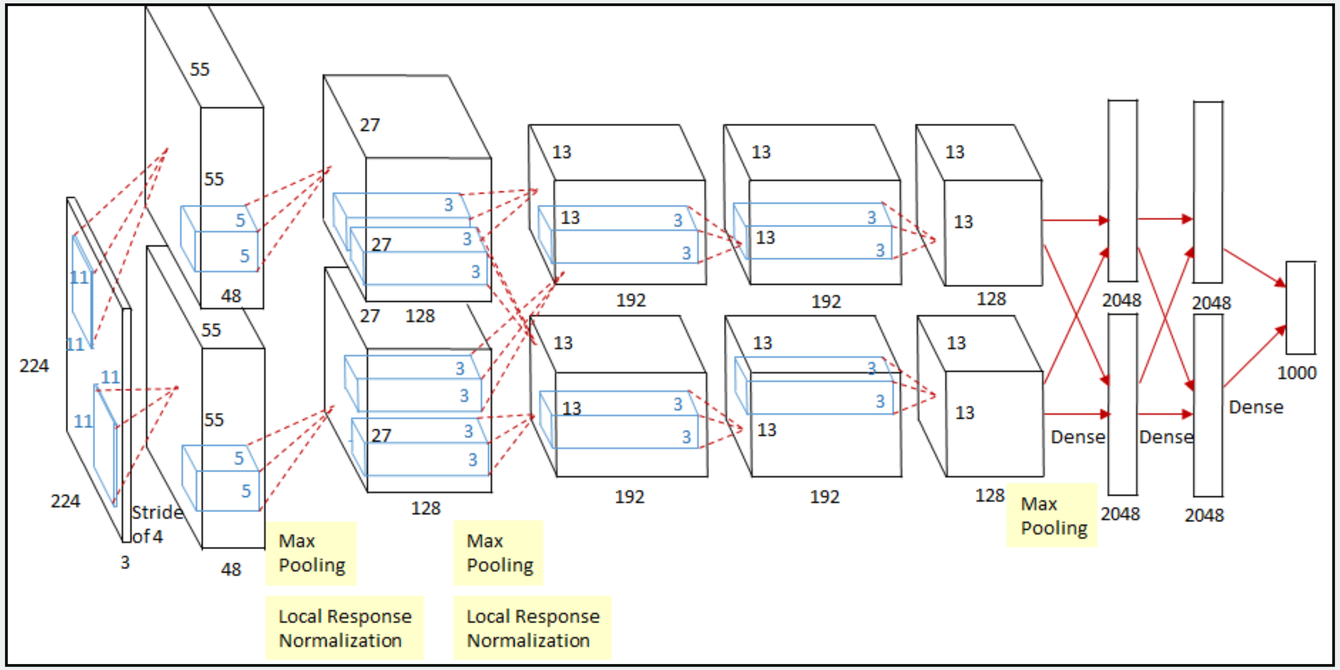
Transfer learning

Reusing trained models

Transfer learning

> The problem with training big deep learning models

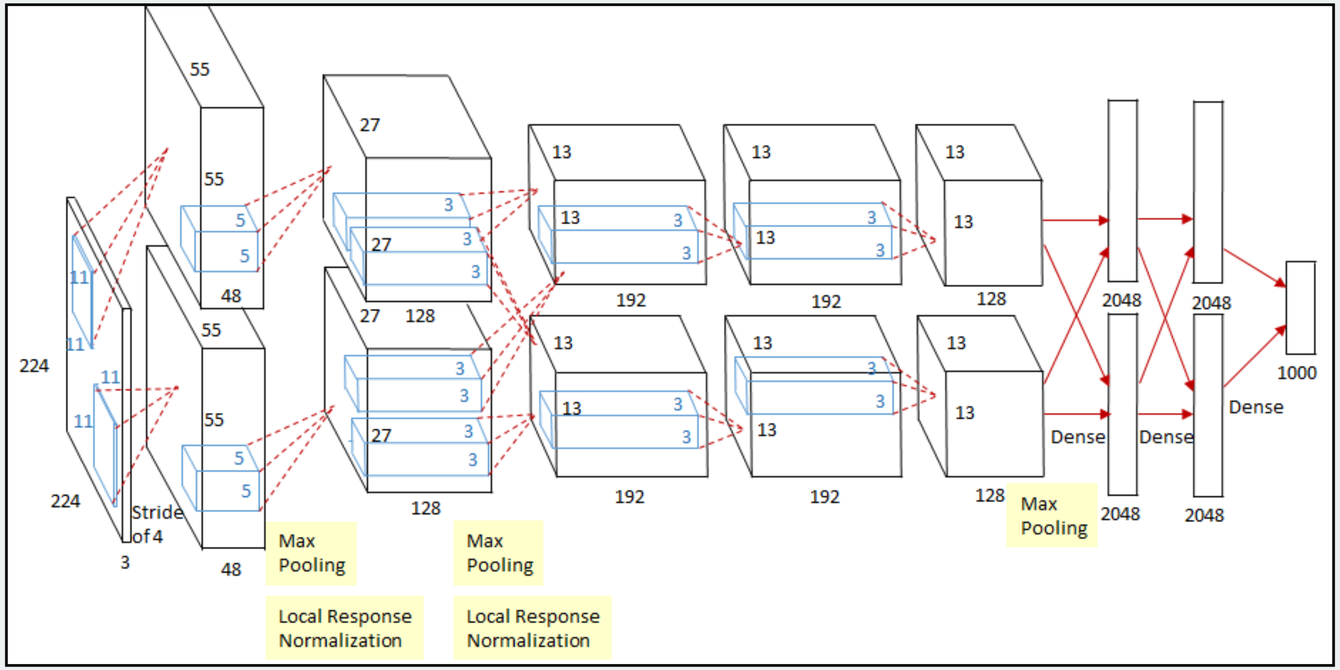
- Extremely long training times. Weeks for many big language models on big machines. OpenAI Five took 10 months to train.
- Expensive cloud computing fees, or GPU cost and electricity bills
- Massive CO_2 footprint



Transfer learning

> Solution: Reuse trained models!

- Extremely long training times. Weeks for many big language models on big machines. OpenAI Five took 10 months to train.
- Expensive cloud computing fees, or GPU cost and electricity bills
- Massive CO_2 footprint

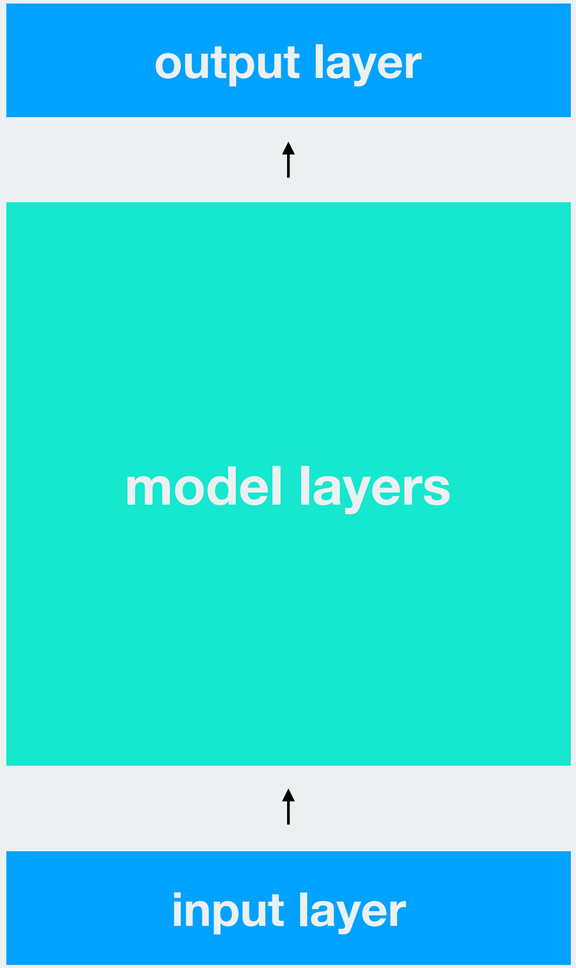


Transfer learning

> Fundamental idea

1. Train on one (huge) dataset

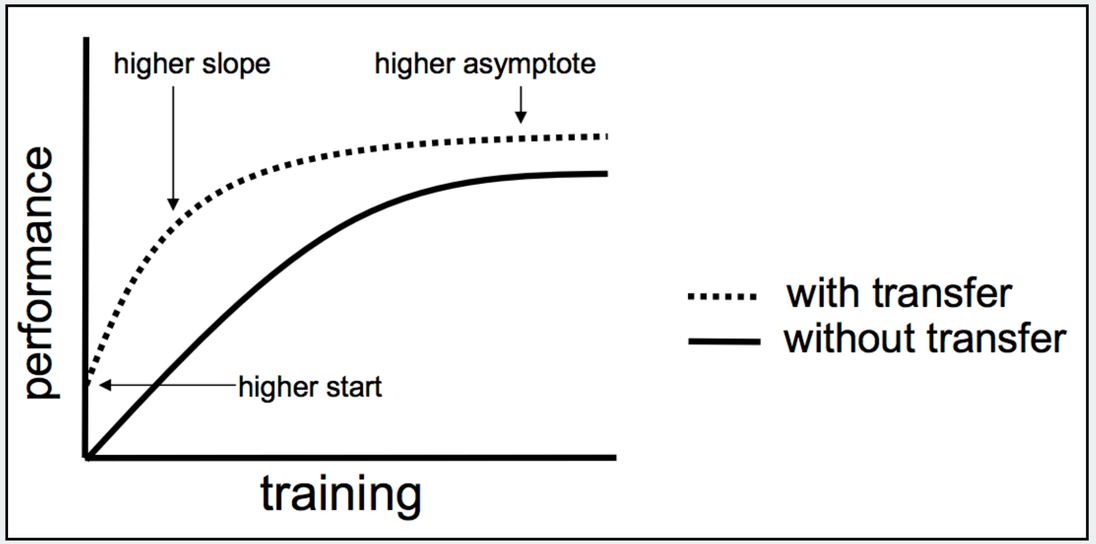
2. Reuse model to improve training on another dataset



Transfer learning

> Benefits

- Makes training on new data much faster
- Enables training on small datasets
- Helps avoid overfitting. Initial weights are usually better than random, helping avoid many local minima.



https://machinelearningmastery.com/transfer-learning-for-deep-learning/

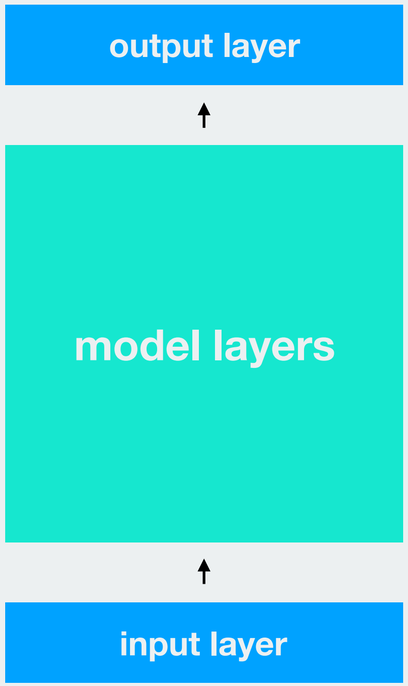
Transfer learning

> Fundamental idea (nuanced)

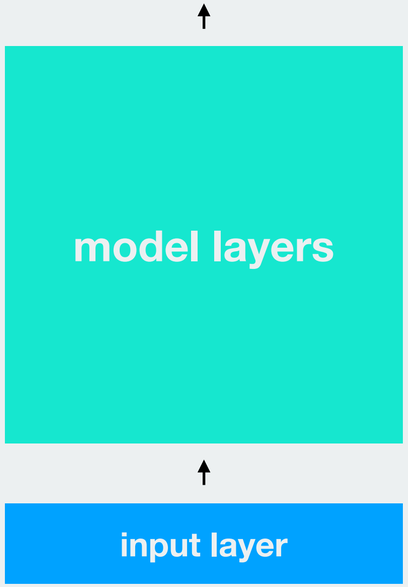
Pre-training dataset

Training dataset

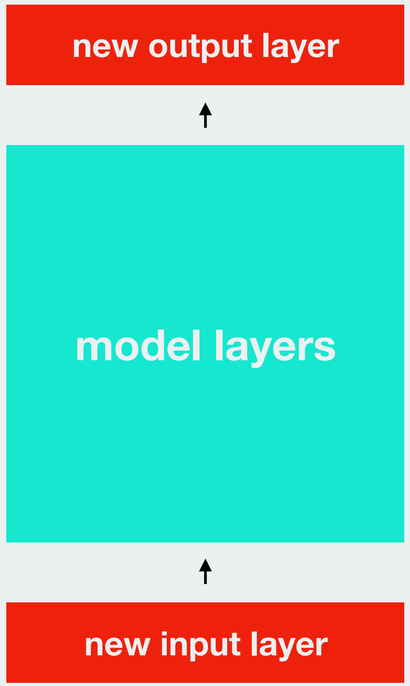
1. Train on huge dataset



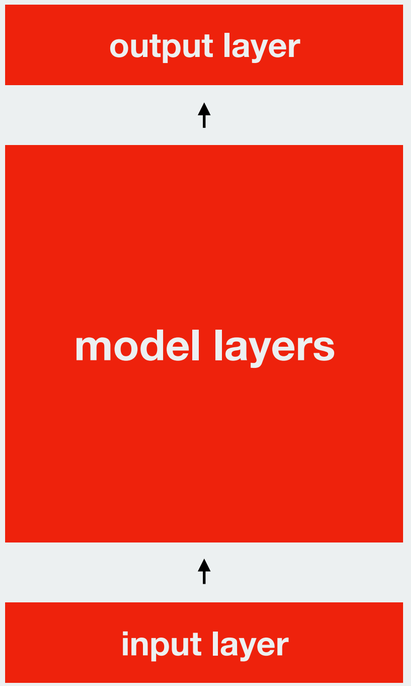
2. Use as feature extractor



3. Train new i/o layers



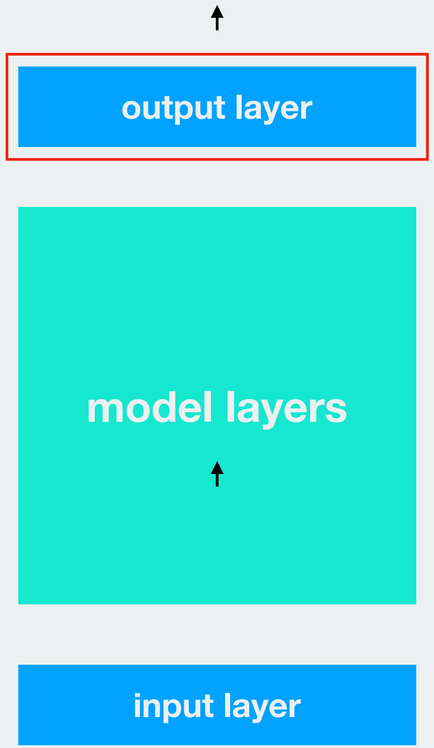
4. Continue training model



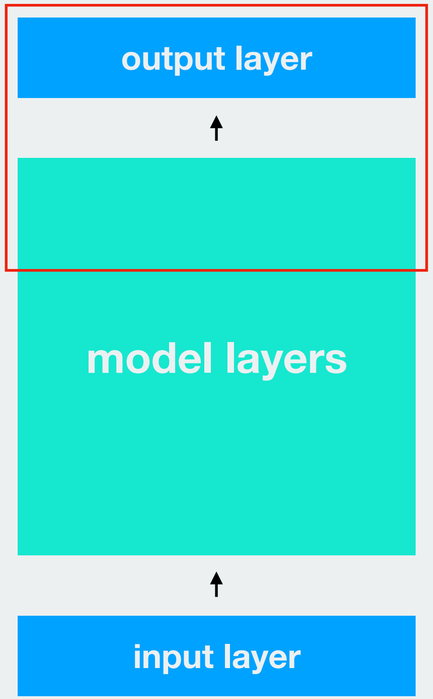
Transfer learning

> Common strategies

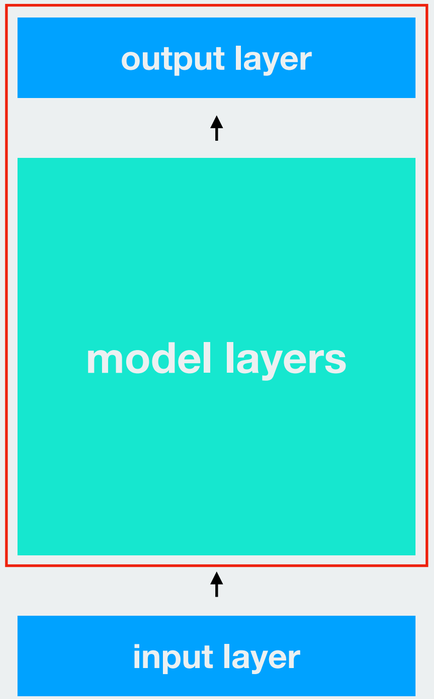
1. If training dataset is **small** only train **last layers**



2. If training dataset is **big** train more **last layers**



3. If training dataset is **huge** train all **layers** with reduced learning rate. 1/10th of orig. LR is good choice



Transfer learning

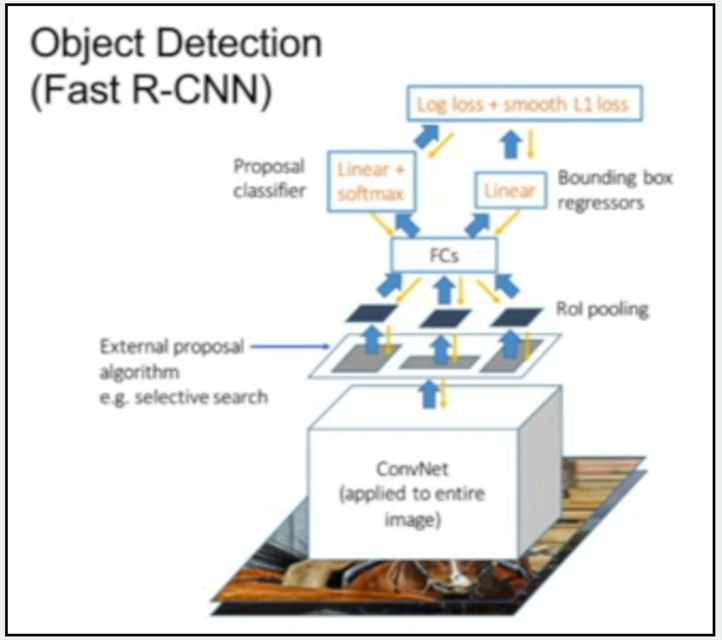
> Common strategies

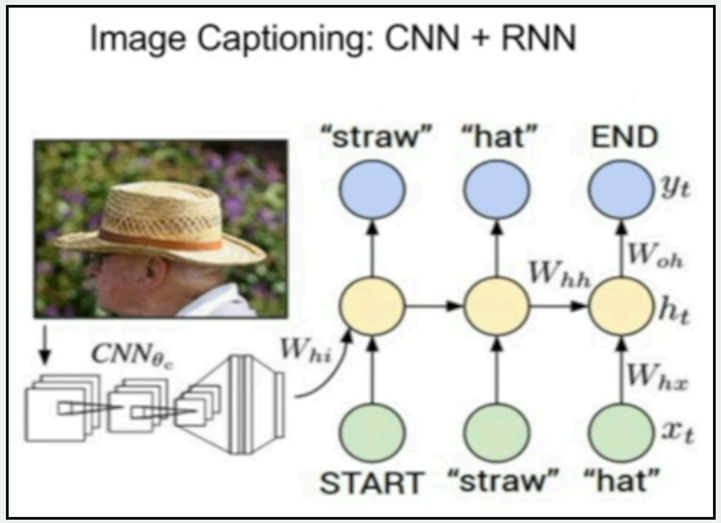
	Similar dataset	**Different dataset**
Little data	Use pre-trained model as feature extractor and do classification with new features and simpler model	Difficult. Maybe consider using a different pre-trained model or use different feature extractors
Much data	Finetune a few layers towards the end of the network, with lowered LR	Finetune a large number of layers, with lowered LR

Transfer learning

> Further

- Transfer learning is extremely pervasive, especially for image data
- Also used for language modeling. There exists publicly available *word embeddings* which encode words as vectors in an efficient way.
- Most research and industrial projects start with some pertained model and then builds something on top of that.





Generative models

Learn underlying patterns and generate new data

Generative models

> Two classes of machine learning

Supervised

When you have outcomes 

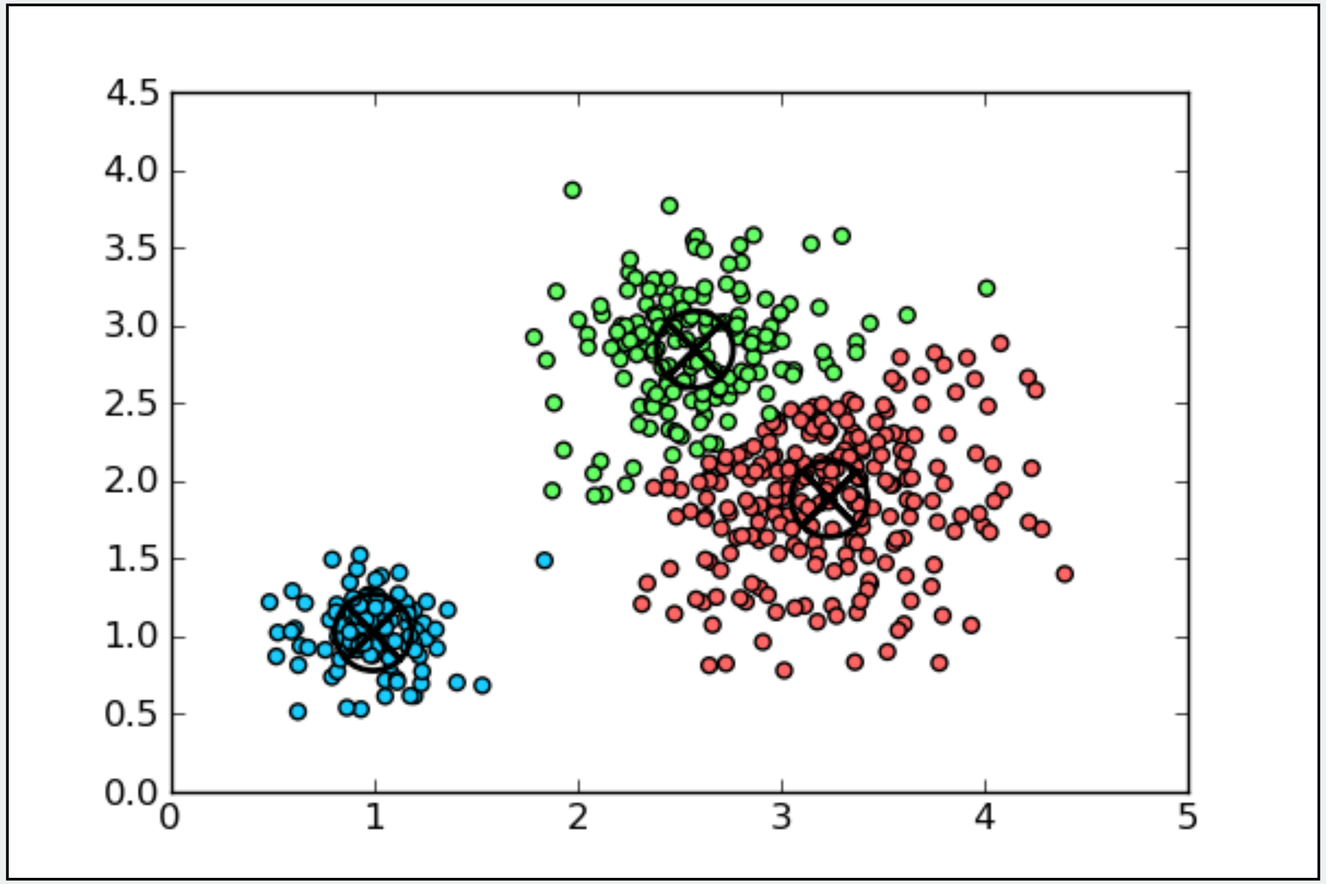
Unsupervised

When you don’t have outcomes 

Generative models

> Unsupervised learning: learning **underlying patterns** in data

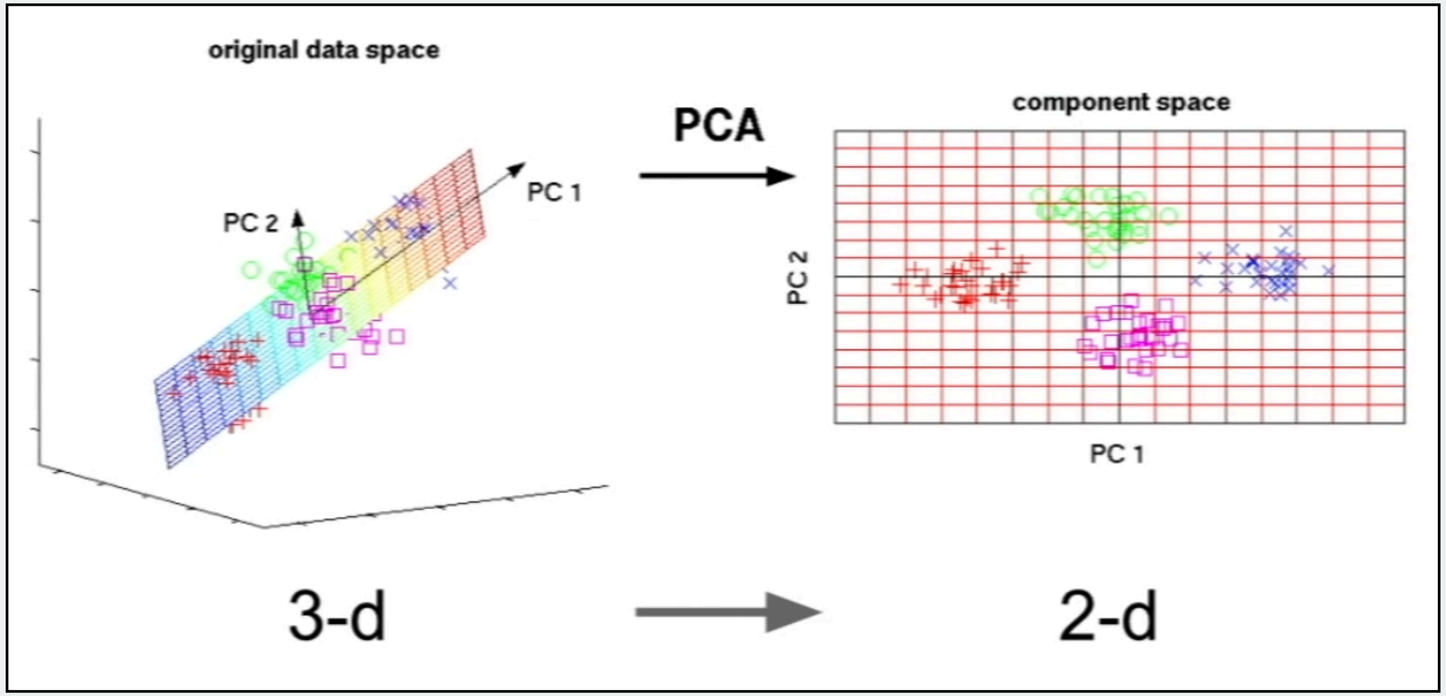
- Clustering



Generative models

> Unsupervised learning: learning **underlying patterns** in data

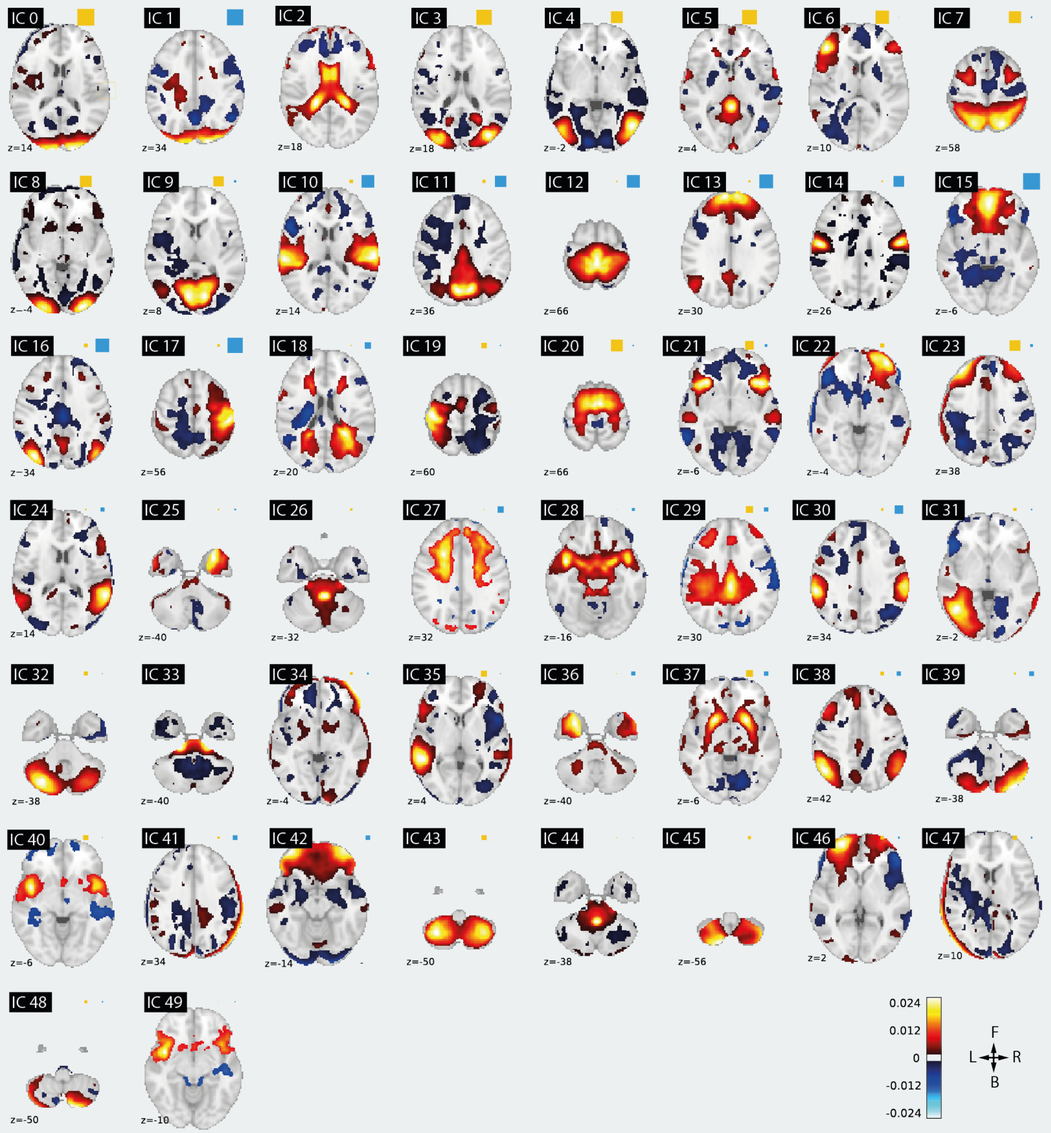
- Clustering
- Latent feature representation



Generative models

> Unsupervised learning: learning **underlying patterns** in data

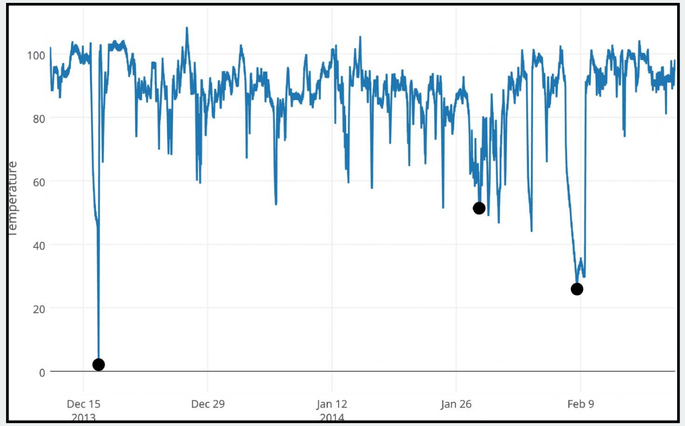
- Clustering
- Latent feature representation
- Signal separation

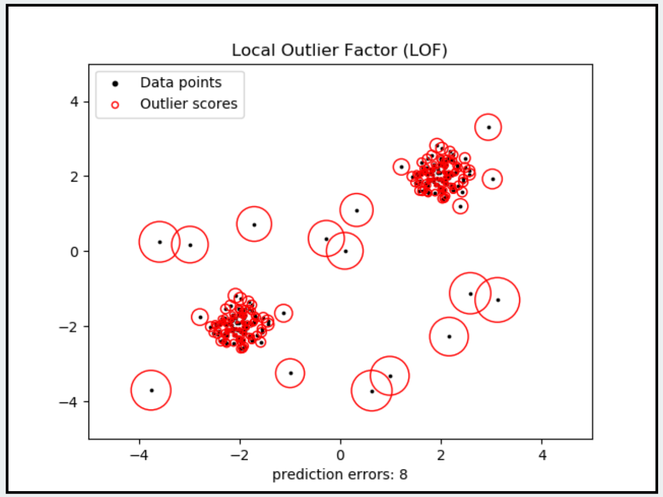


Generative models

> Unsupervised learning: learning **underlying patterns** in data

- Clustering
- Latent feature representation
- Signal separation
- Anomaly detection





Generative models

> Unsupervised learning: learning **underlying patterns** in data

- Clustering
- Latent feature representation
- Signal separation
- Anomaly detection

}

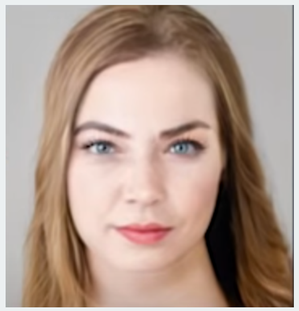
Lots of applications
with neural networks

Generative models

> Example: low-dimensional representation of faces

$$\longrightarrow \quad [-0.92,\ 0.97,\ -0.99,\ 0.42,\ \ldots,\ -0.46]$$

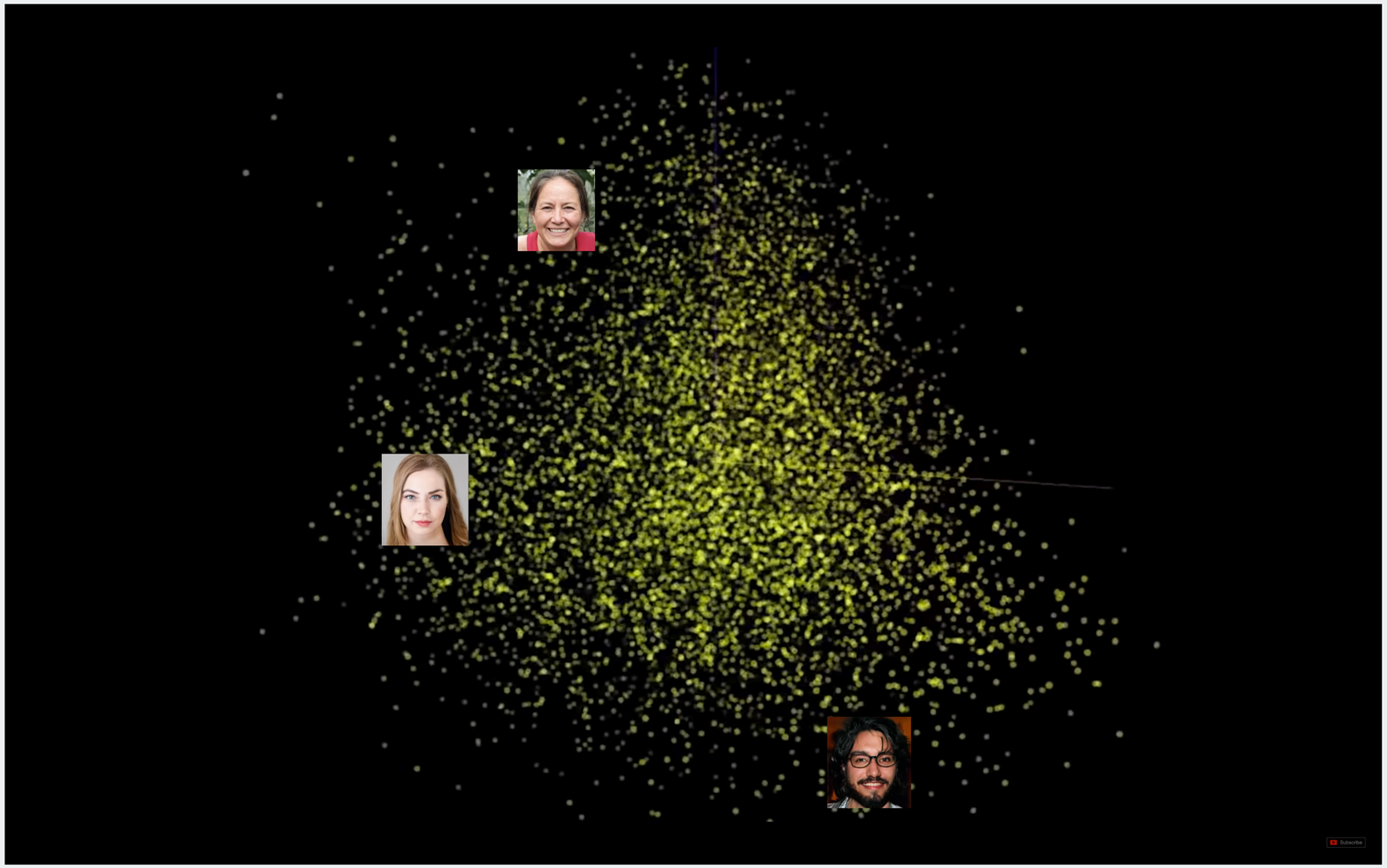
$$\longrightarrow \quad [0.56,\ -0.42,\ -0.03,\ -0.33,\ \ldots,\ 0.03]$$



$$\longrightarrow \quad [-0.18,\ 0.86,\ -0.96,\ 0.83,\ \ldots,\ -0.63]$$

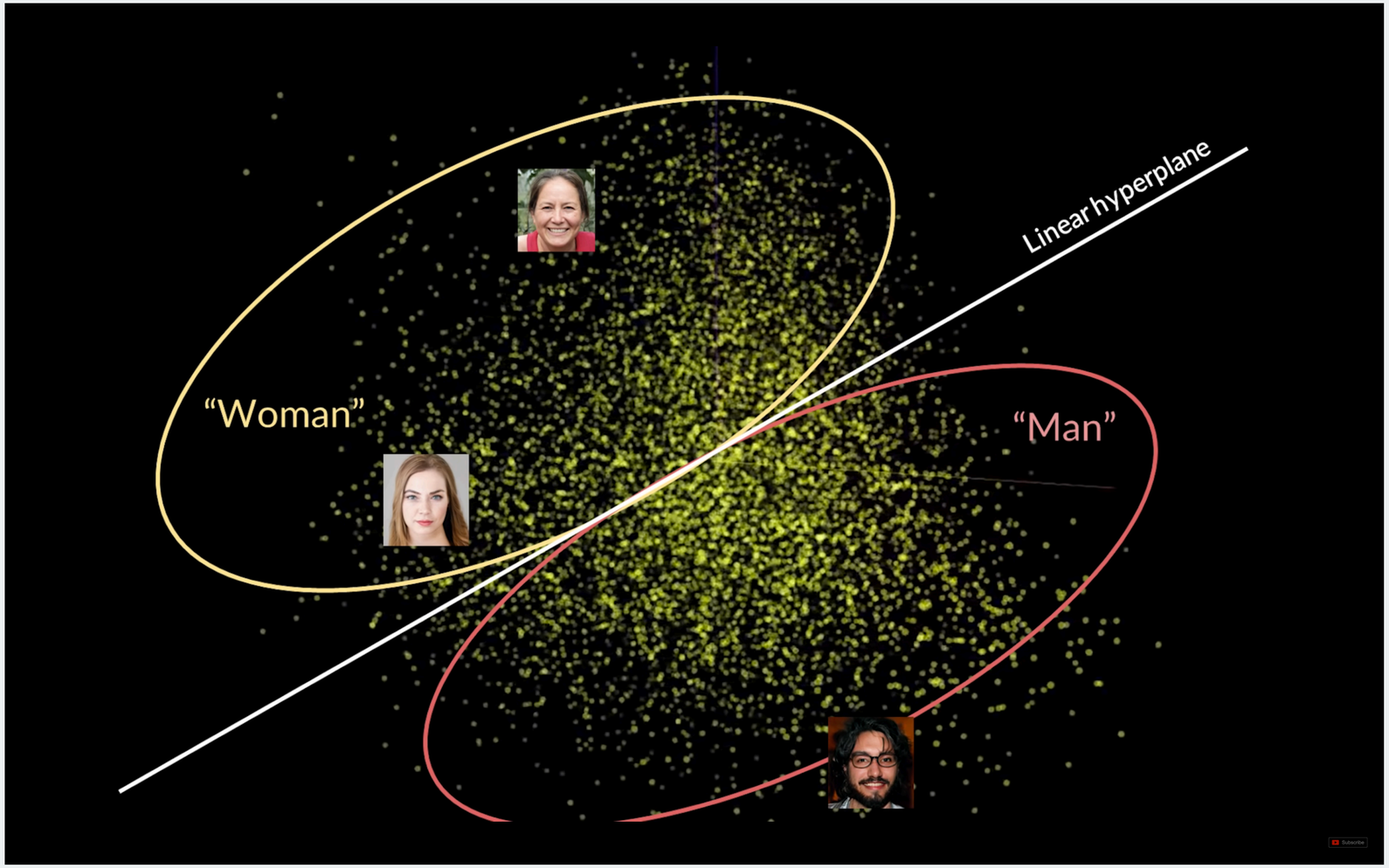
Generative models

> Example: low-dimensional representation of faces



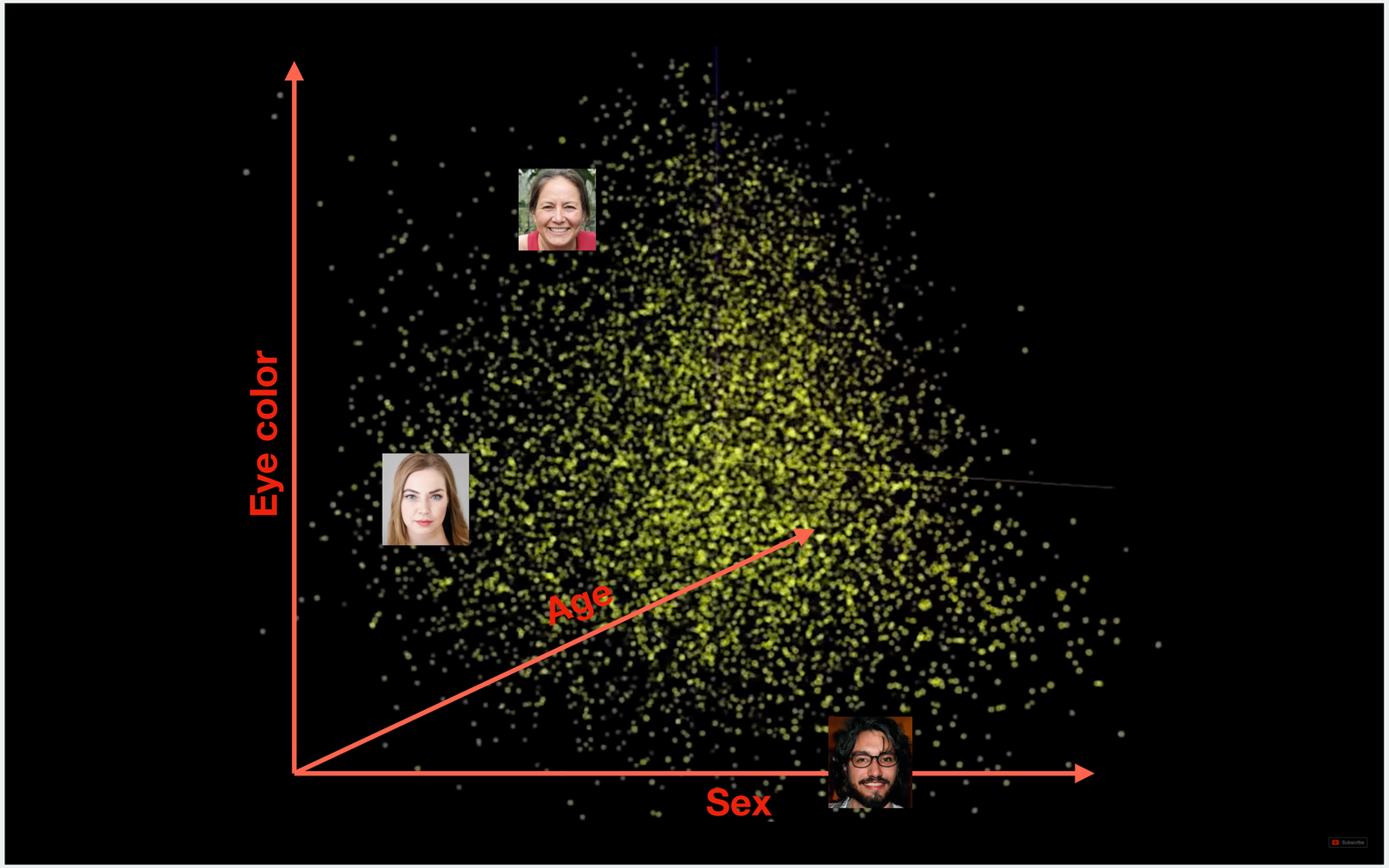
Generative models

> Example: low-dimensional representation of faces



Generative models

> Example: low-dimensional representation of faces



Generative models

> Learn about VAEs an GANs

"Variational Autoencoders"

"Learn how to morph faces with a Generative Adversarial Network!"

Further learning

- Convolutional RNN
- Graph CNN
- Bidirectional LSTM
- Object detection
- Image segmentation
- Embeddings (word, face, etc.)
- Reinforcement learning
- One shot learning
- Multitask-learning
- Attention
- Neural style transfer
- DeepDream
- Visualization
- …