

# Robot Learning

Basics of computer vision for robotics  
Representation learning

# Logistics that I forgot to tell last time

- Project has multiple components.

Component	Contribution to Grade	
Project Proposal Report	5%	October 8 <sup>th</sup>
Project Milestone Report	5%	November 5 <sup>rd</sup>
Project Presentation (Possibly with Demo)	10%	December 1 <sup>st</sup>
Final Project Report	15%	December 3 <sup>rd</sup>
Peer Review	5%	December 10 <sup>th</sup>
Total	40%	

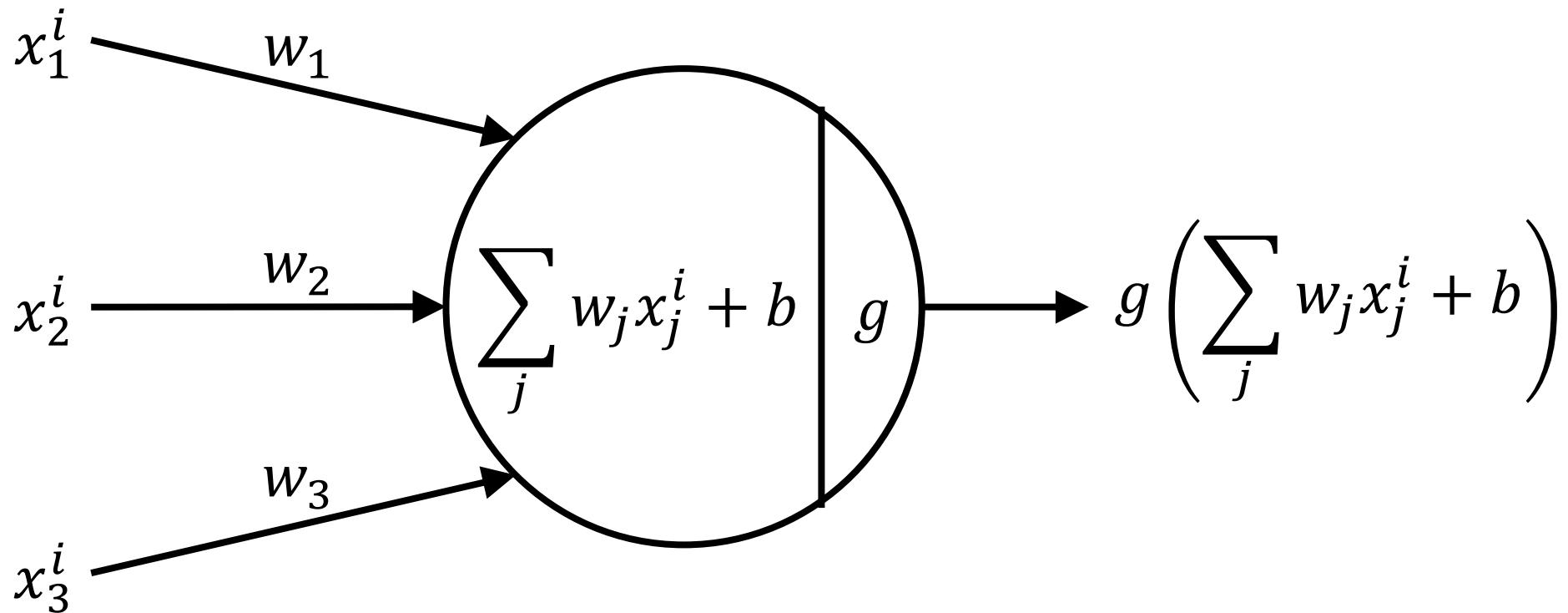
# Announcements

- Homework 1 is out, due September 17<sup>th</sup>.
- Class presentations:

Mao, Jiageng	Unsupervised Learning of Depth and Ego-Motion from Video
Banayeeanzade, Amin	Representation Learning with Contrastive Predictive Coding
Kulkarni, Yogesh	A Simple Framework for Contrastive Learning of Visual Representations
Zhang, Hejia	SIRL: Similarity-based Implicit Representation Learning

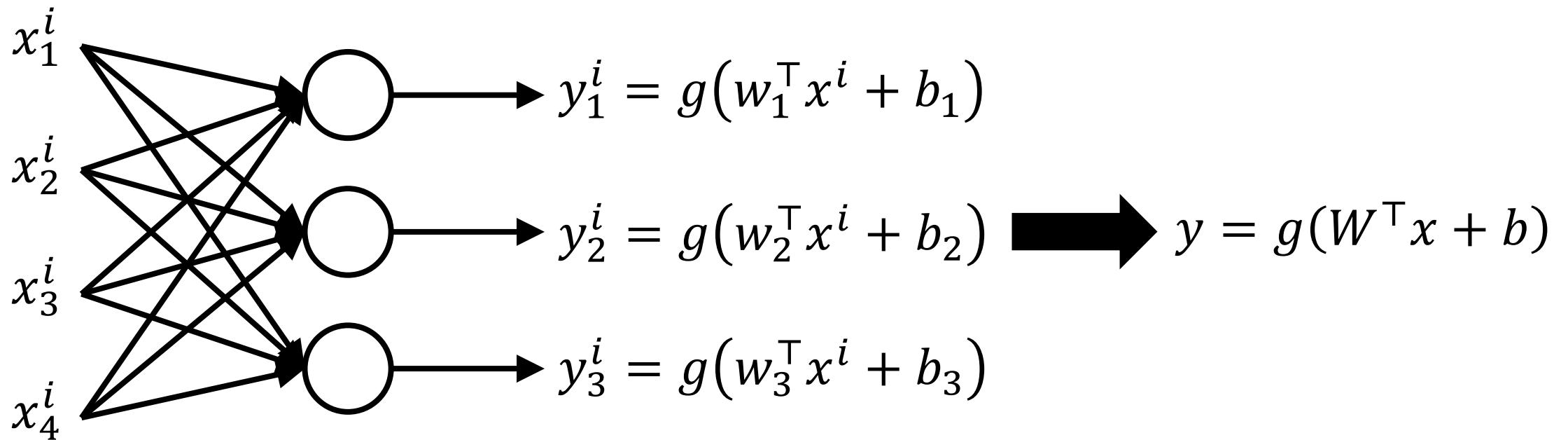
# Last time...

## 1. A perceptron



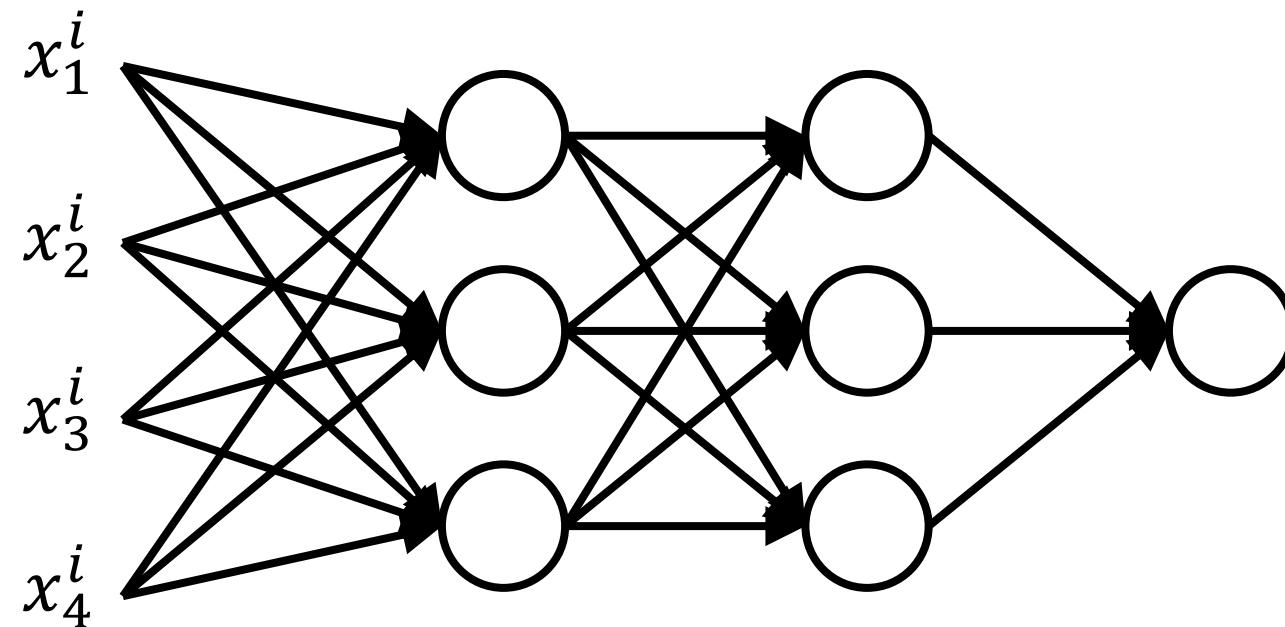
# Last time...

## 2. A single layer neural network

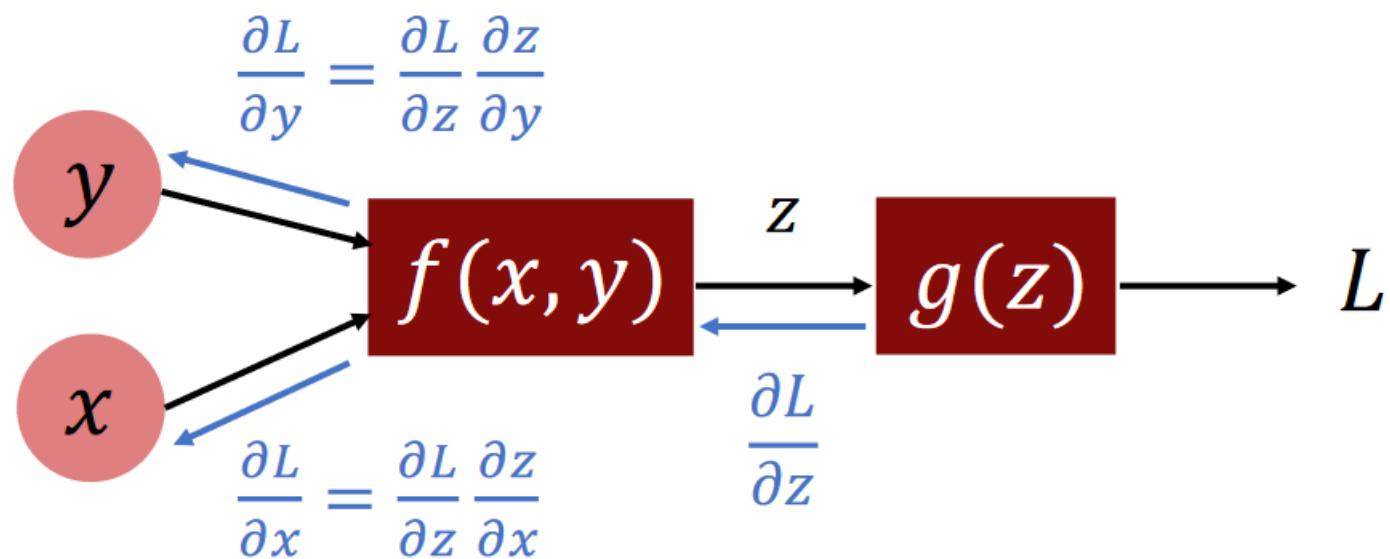


# Last time...

## 3. A deep neural network



# Last time...

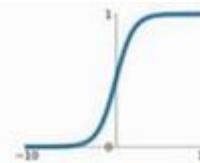


# Last time...

$g$  should not be a linear function.

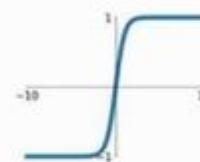
**Sigmoid**

$$\frac{1}{1+e^{-x}}$$



**tanh**

$$\tanh(x)$$



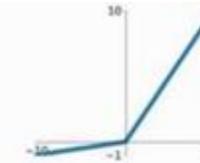
**ReLU**

$$\max(0, x)$$



**Leaky ReLU**

$$\max(0.1x, x)$$



**Maxout**

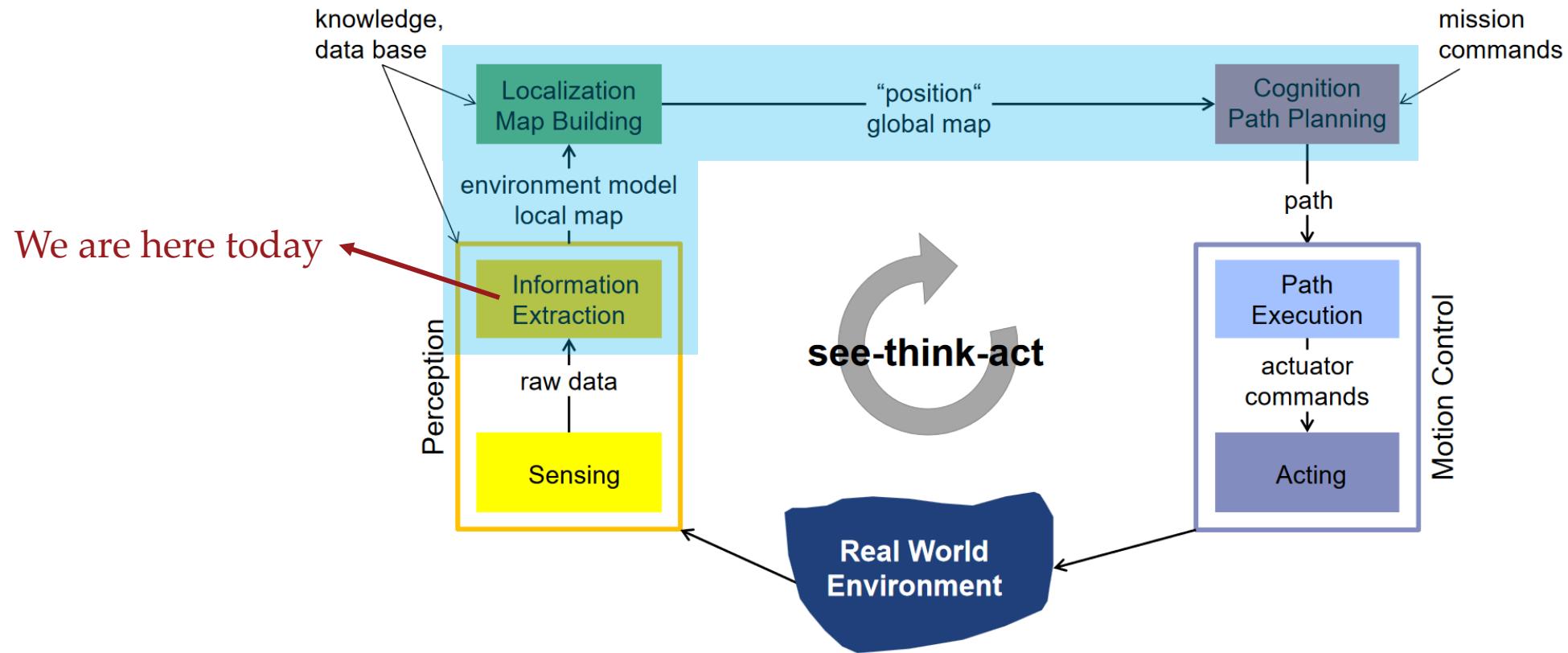
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

**ELU**

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

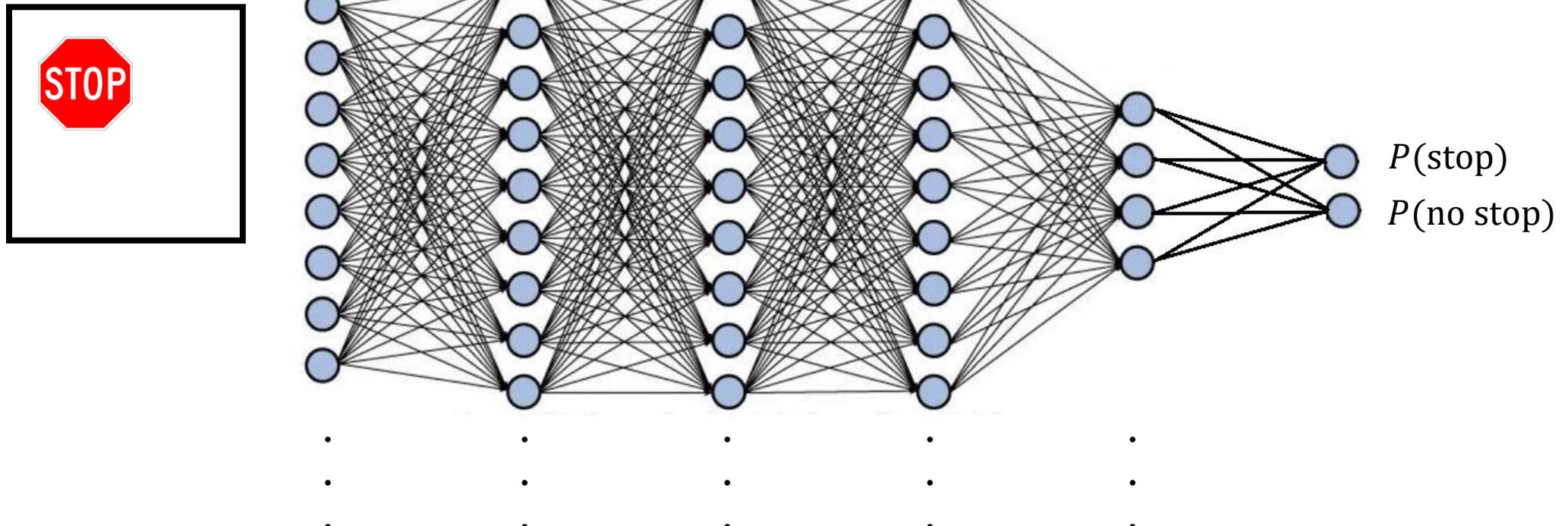


# Robot learning



# Neural networks for images

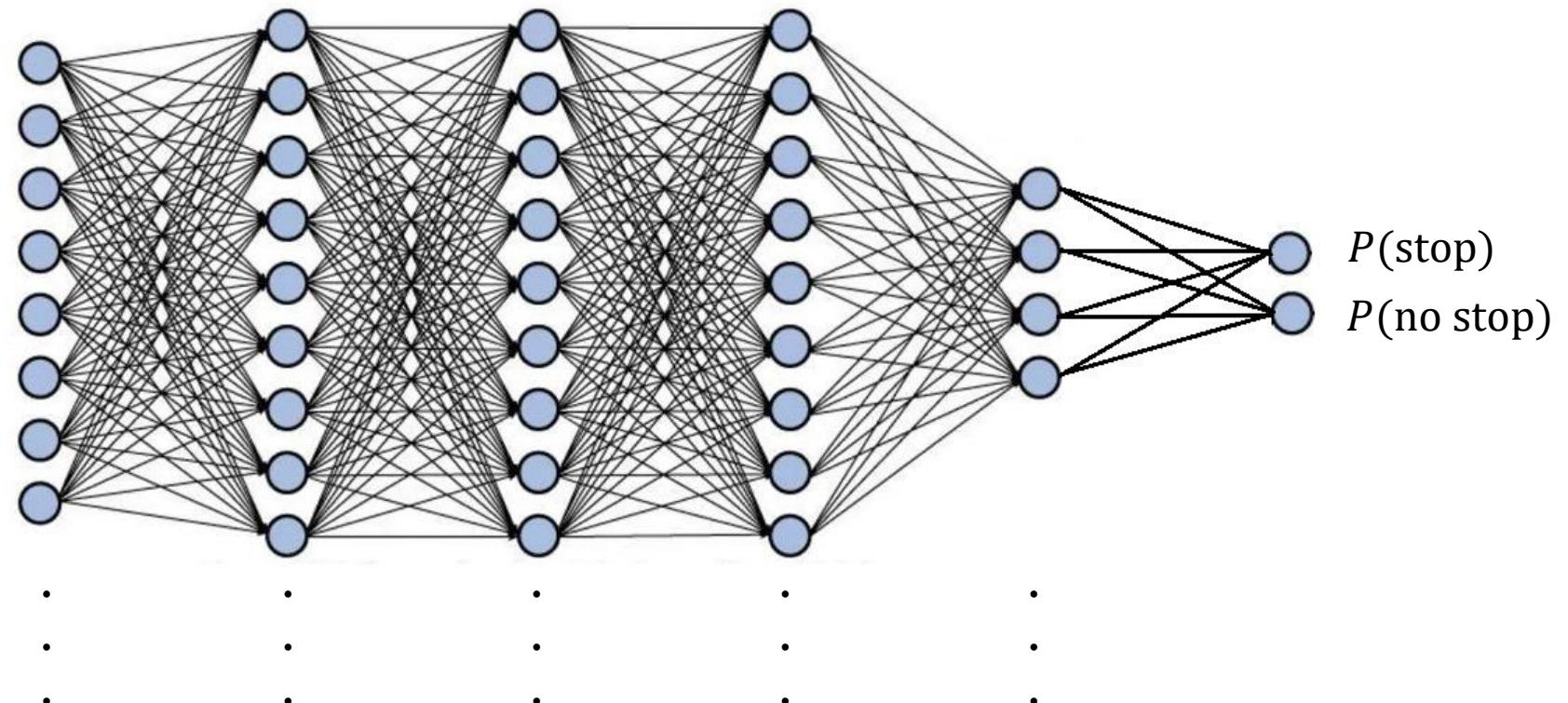
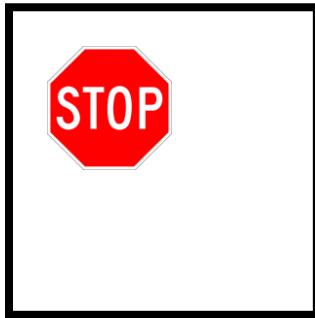
Easy “solution”: flatten the image and make it a vector.



# Neural networks for images

Easy “solution”: flatten the image and make it a vector.

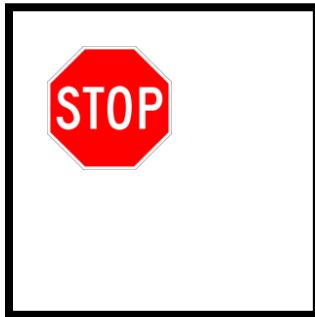
This is a  
stop sign.



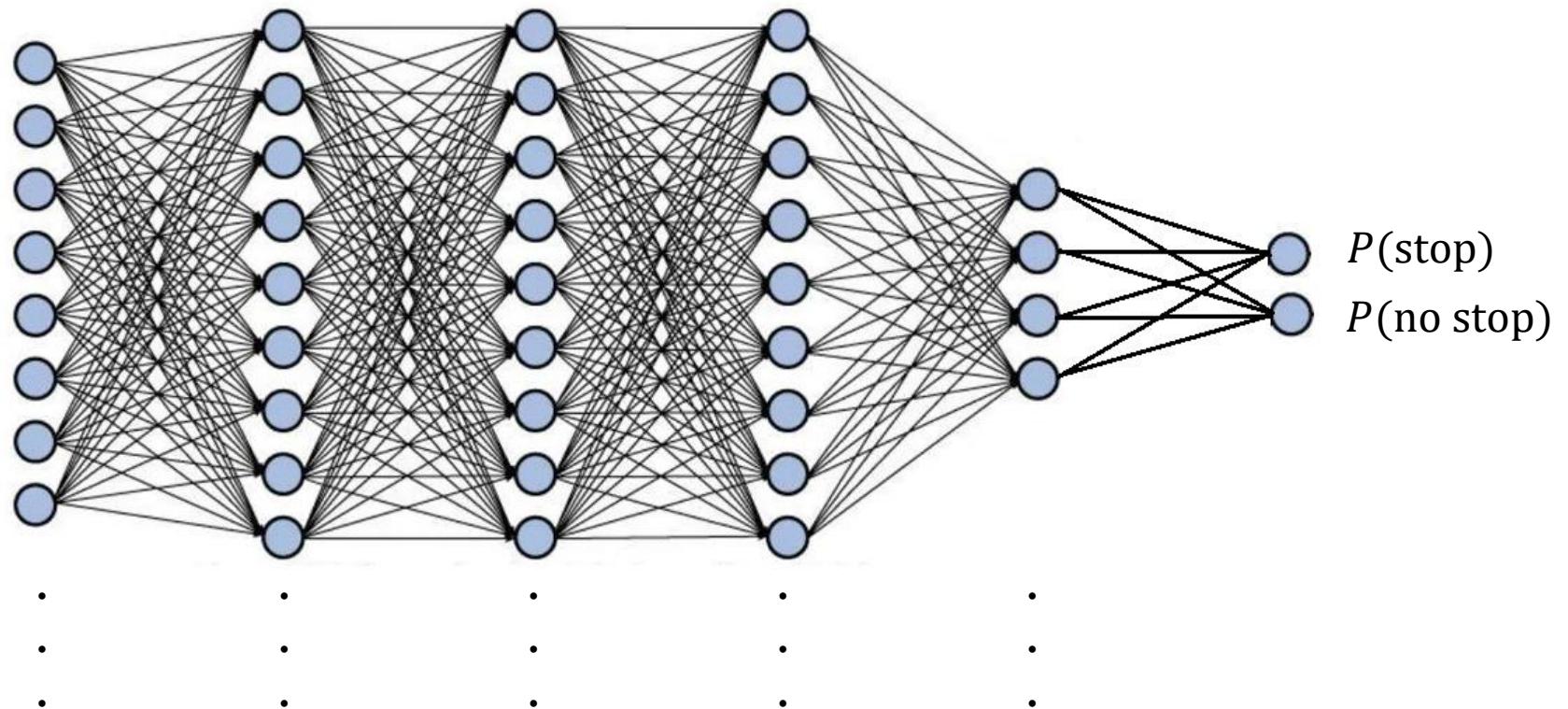
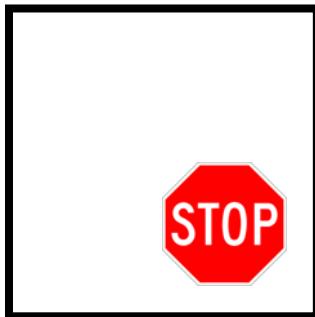
# Neural networks for images

Easy “solution”: flatten the image and make it a vector.

This is a  
stop sign.

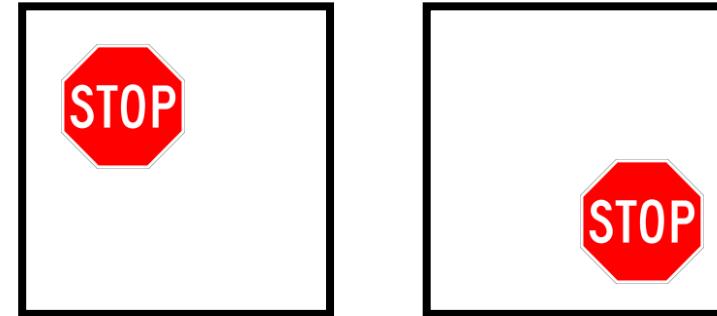


This is also  
a stop sign!



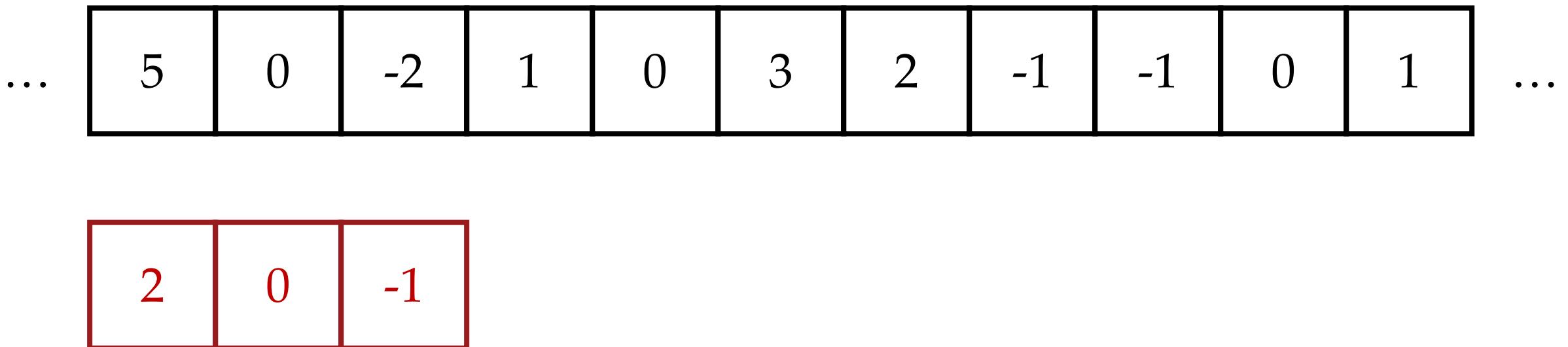
# Neural networks for images

Regardless of its location in the image,  is a stop sign.

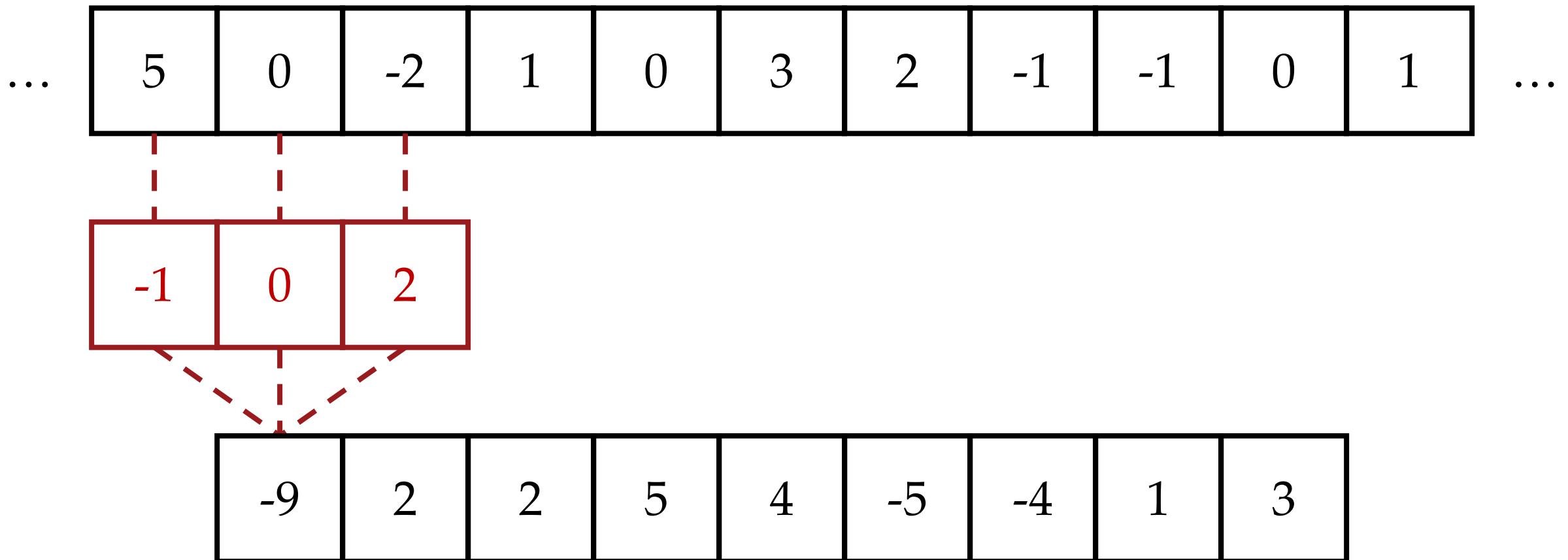


Spatial locality → weight sharing

# Convolution



# Convolution



# Why is convolution important?

Many major technological developments in the past 50 years somehow relates to convolution.

- Used in: image processing, machine learning, telecommunications, audio processing, medical imaging, radar/sonar systems, seismology, quantum computing, optics, data compression, 3D graphics...

# Why is convolution important?

Many major technological developments in the past 50 years somehow relates to convolution.

- Used in: image processing, machine learning, telecommunications, audio processing, medical imaging, radar/sonar systems, seismology, quantum computing, optics, data compression, 3D graphics...

Became especially popular after James Cooley and John Tukey developed Fast Fourier Transform (FFT) in 1965.



Reduces the computational complexity of convolution from  $O(NM)$  to  $O(N \log N)$

# What's more?

FFT is “*the most important numerical algorithm of our lifetime.*”

Gilbert Strang, 1994

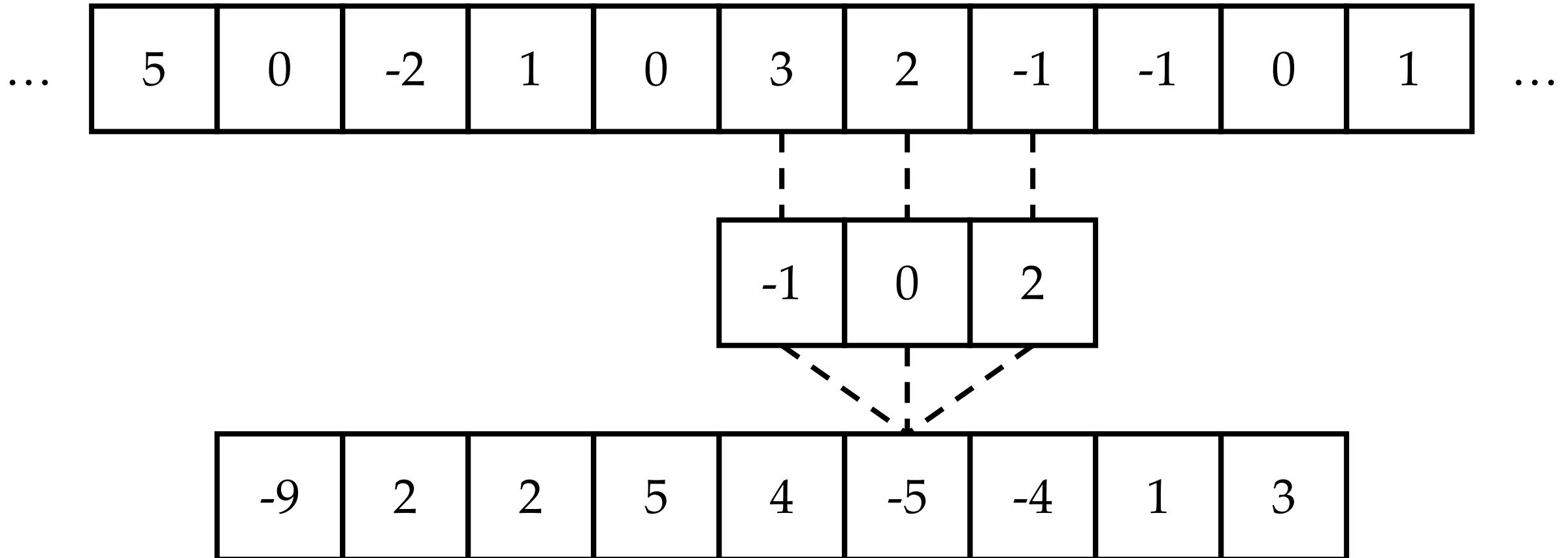
Top 10 algorithms of the 20<sup>th</sup> century, according to IEEE’s Computing in Science & Engineering magazine:

- Metropolis Algorithm for Monte Carlo
- Simplex Method for Linear Programming
- Krylov Subspace Iteration Methods
- The Decompositional Approach to Matrix Computations
- The Fortran Optimizing Compiler
- QR Algorithm for Computing Eigenvalues
- Quicksort Algorithm for Sorting
- Fast Fourier Transform
- Integer Relation Detection
- Fast Multipole Method

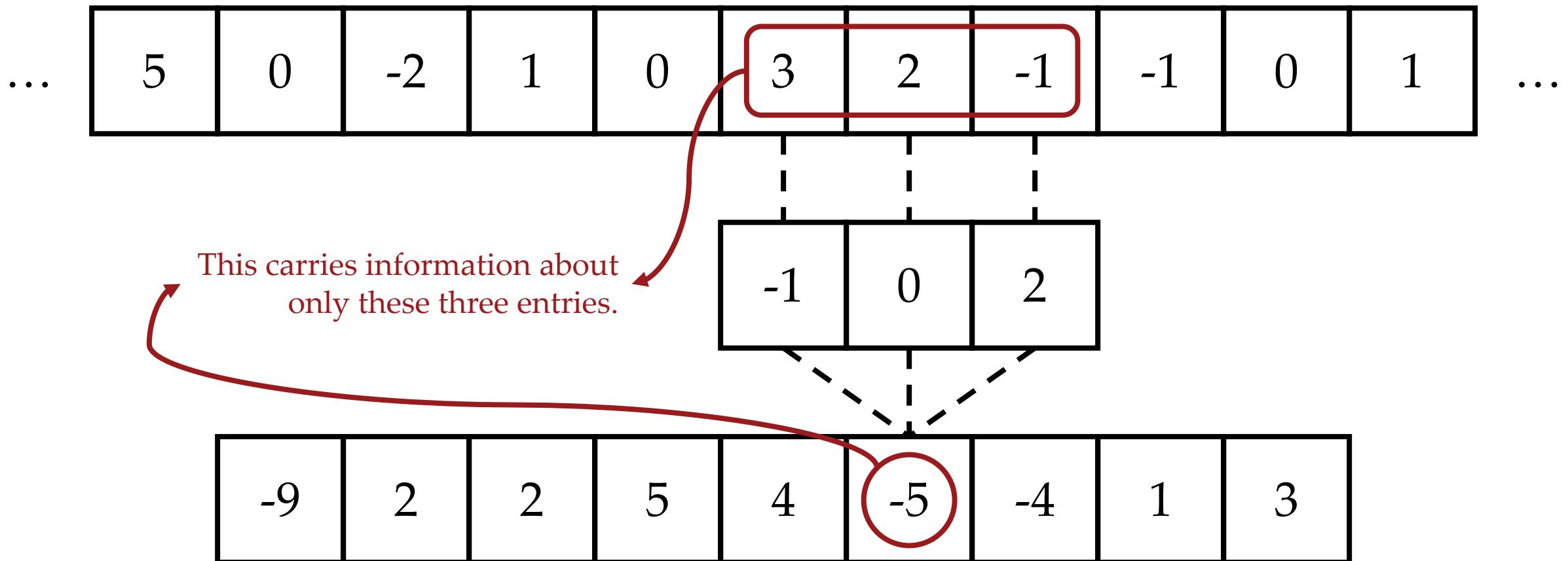
# More on convolution and FFT

- EE 301L: Linear Systems
- EE 483: Introduction to Digital Signal Processing
- EE 434Lx: Digital Signal Processing Design Laboratory
- BME 413: Bioengineering Signals and Systems

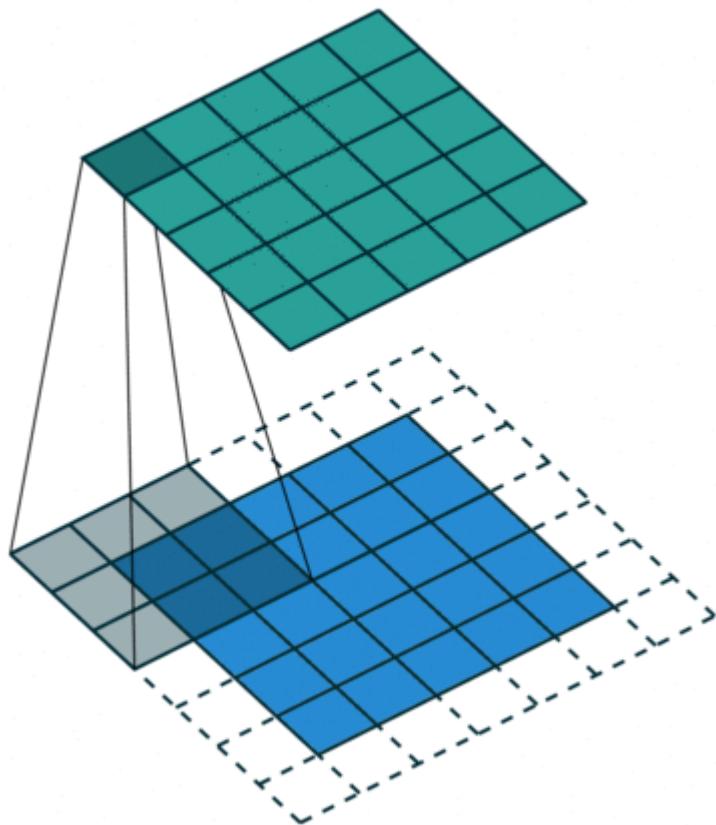
# Convolution



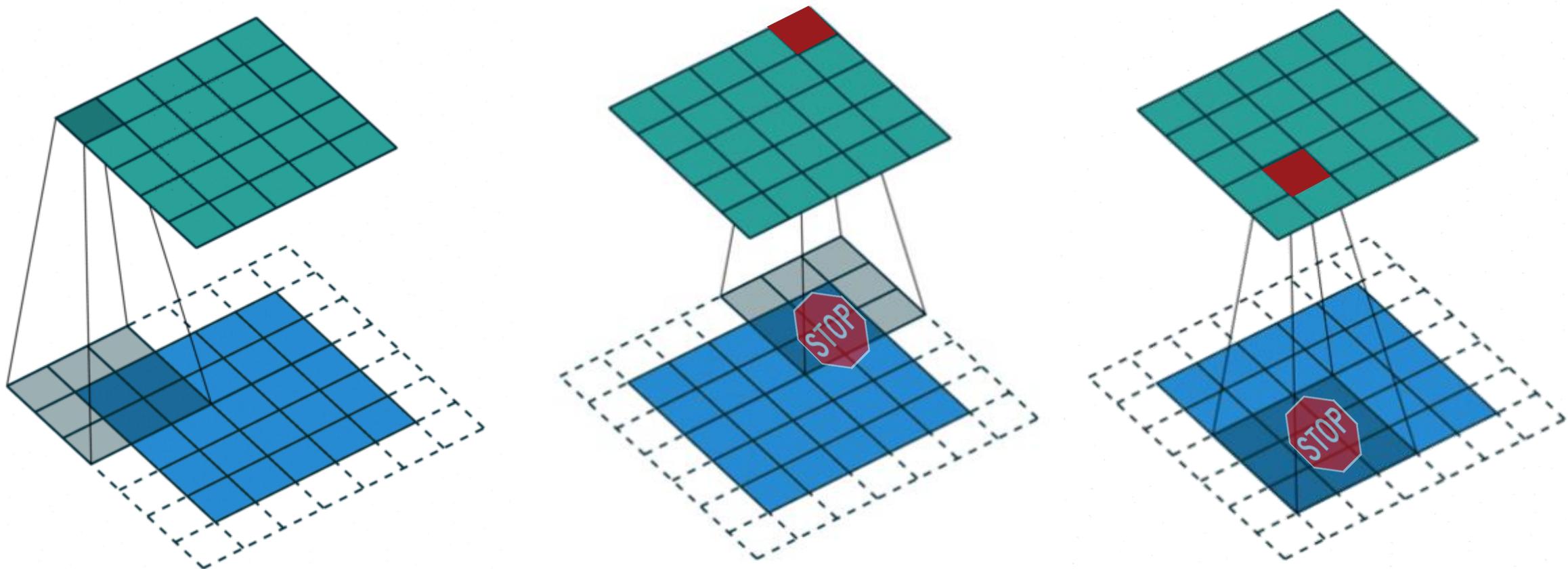
# Convolution



# Convolution in 2D

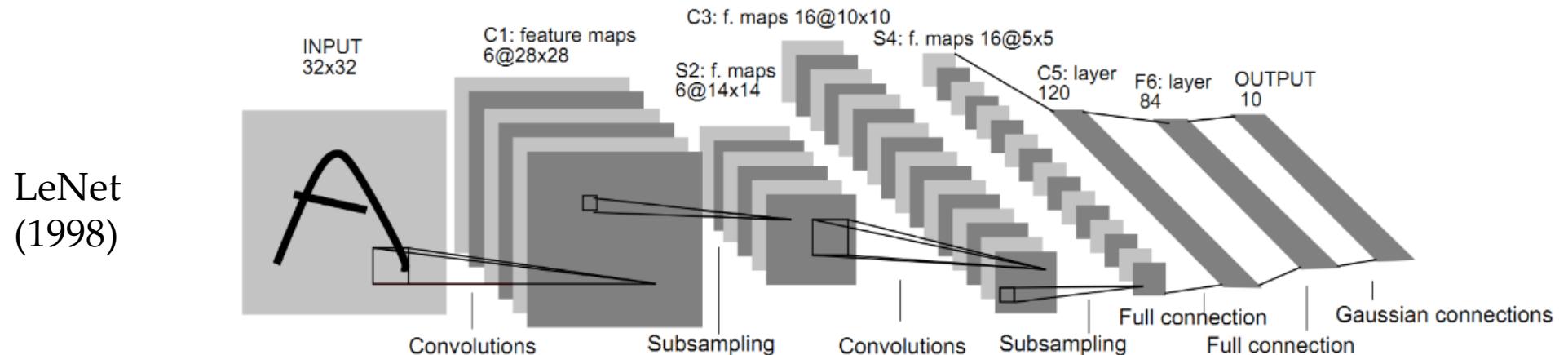
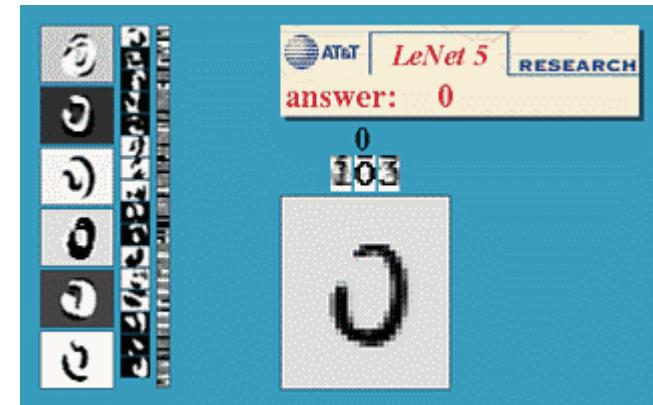


# Convolution in 2D

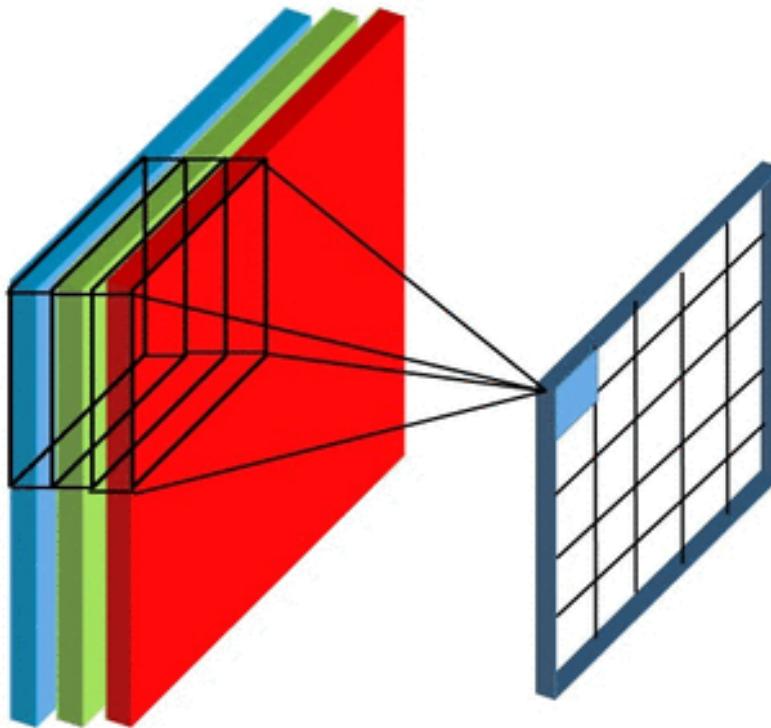


# Convolutional neural networks (CNN)

- Traditionally consists of:
  - Convolutional layers
  - Nonlinearity layers
  - Pooling layers
  - Fully-connected layers

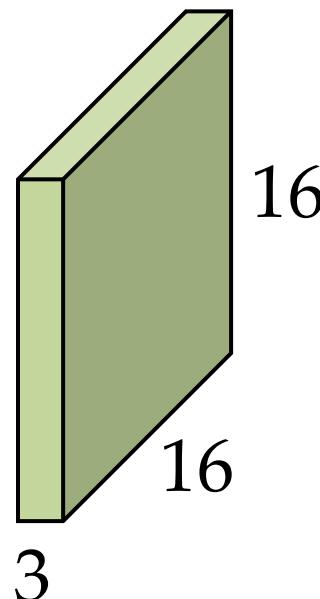


# Convolutional layer



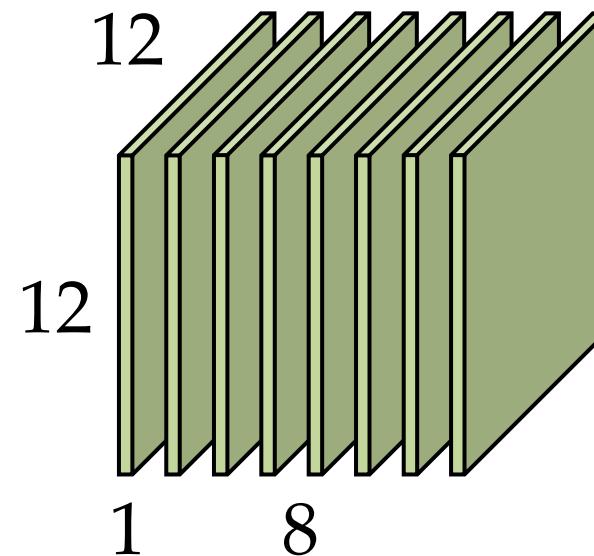
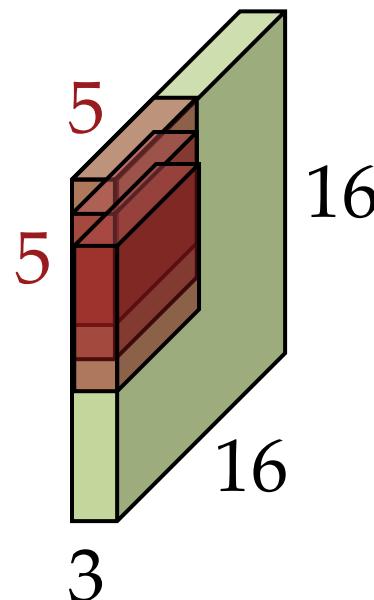
# Convolutional layer

Let's say we have an image with size:  $(16, 16, 3)$  and 8 kernels where each kernel is  $(5, 5)$ .

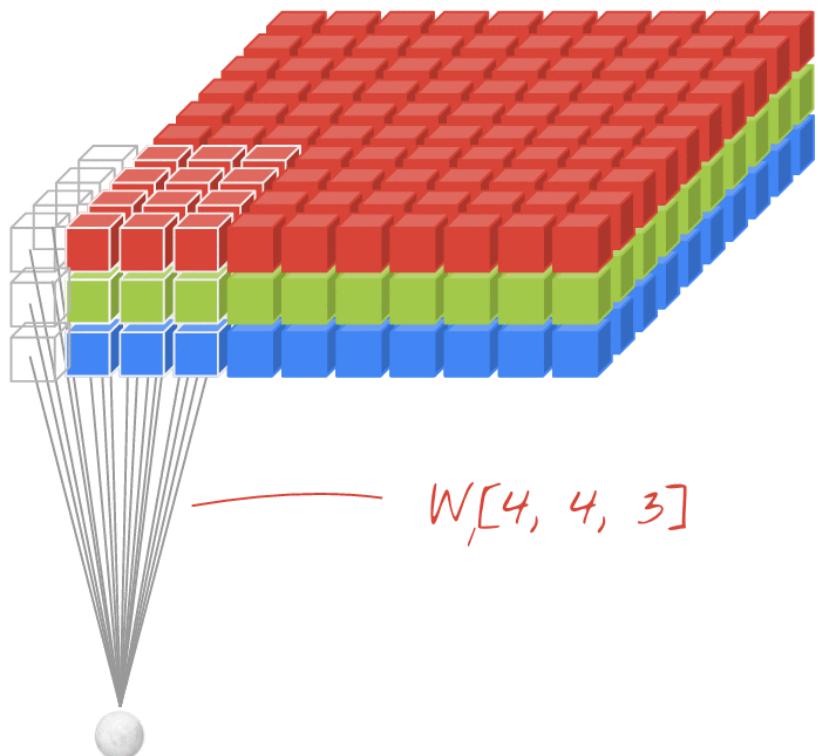


# Convolutional layer

Let's say we have an image with size:  $(16, 16, 3)$  and 8 kernels where each kernel is  $(5, 5)$ .

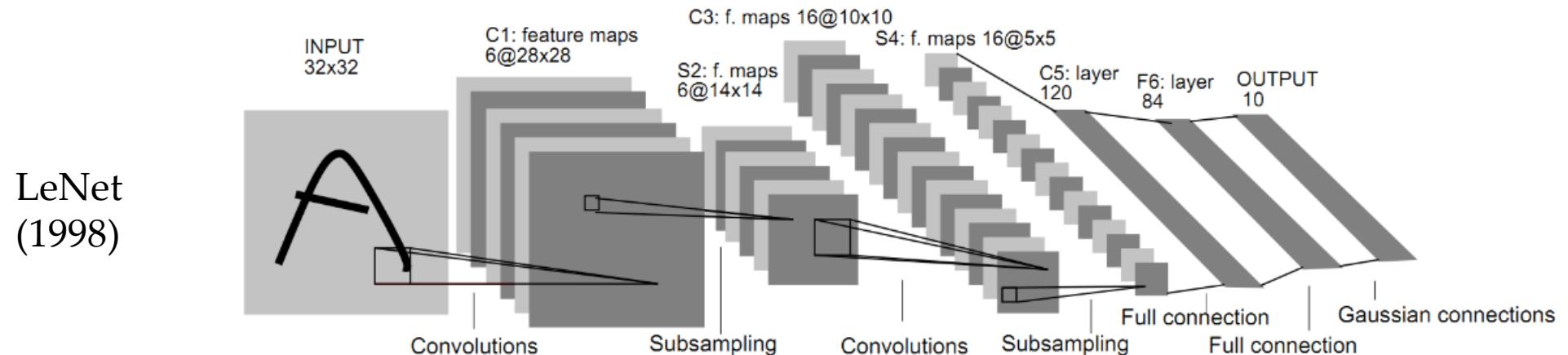
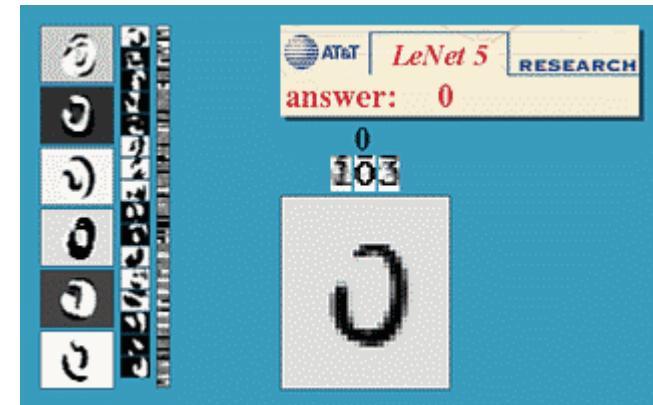


# Padding and stride



# Convolutional neural networks (CNN)

- Traditionally consists of:
  - Convolutional layers
  - Nonlinearity layers
  - Pooling layers
  - Fully-connected layers



Gradient-based learning applied to document recognition

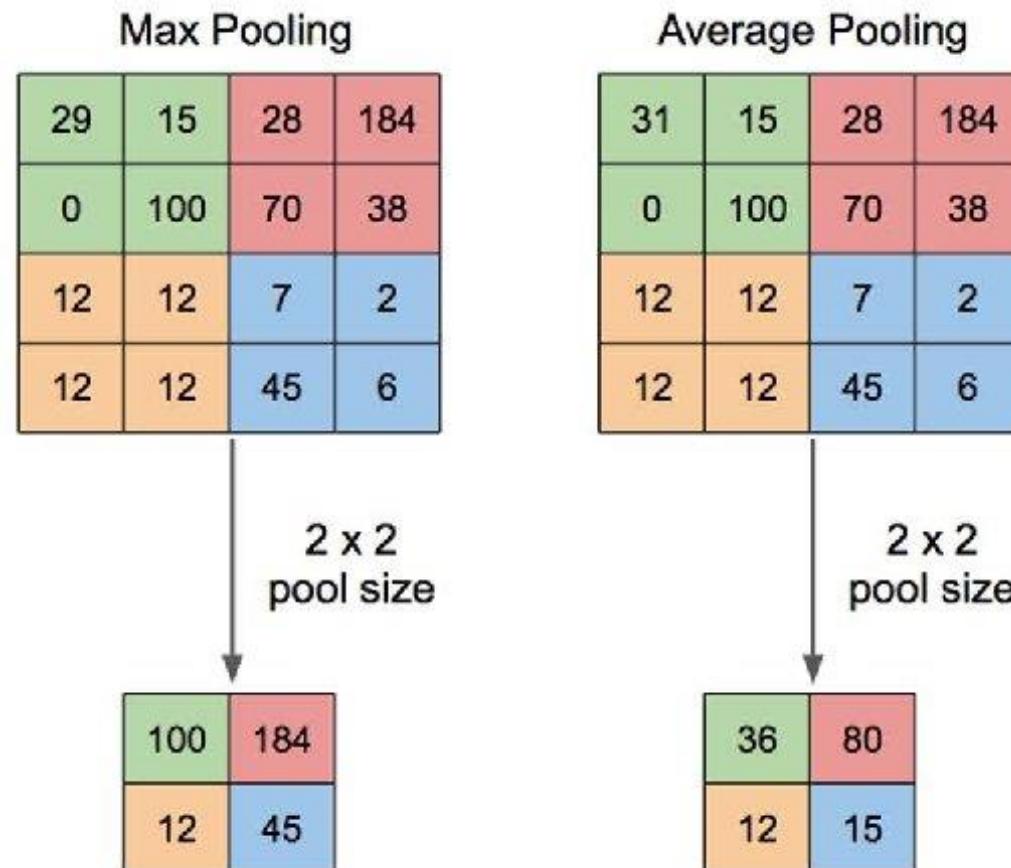
LeCun et al., Proceedings of the IEEE, 1998

Slide from: Marco Pavone (Stanford)

CSCI 699: Robot Learning - Lecture 2

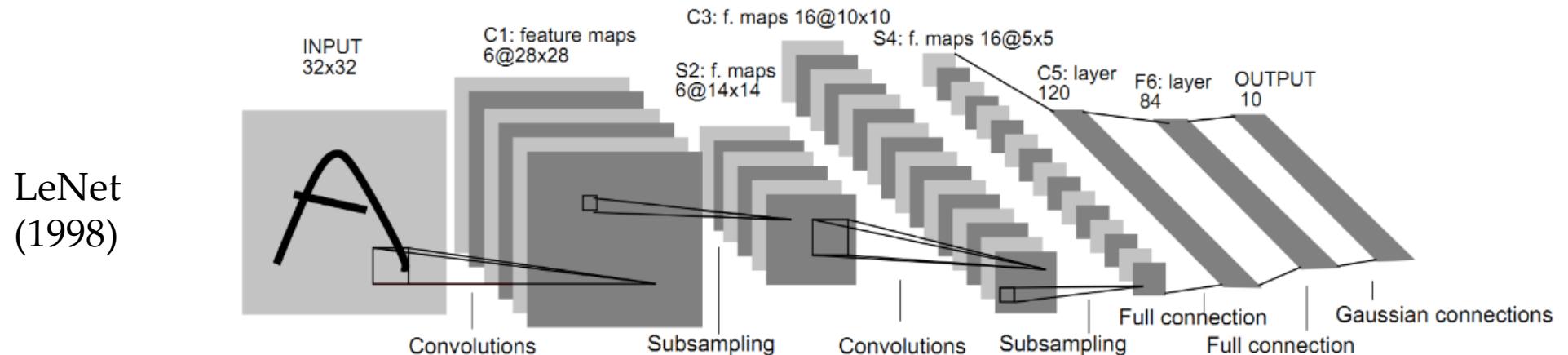
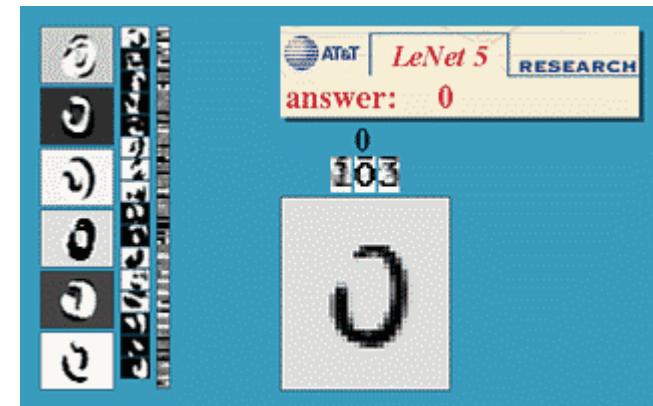
29

# Pooling layer



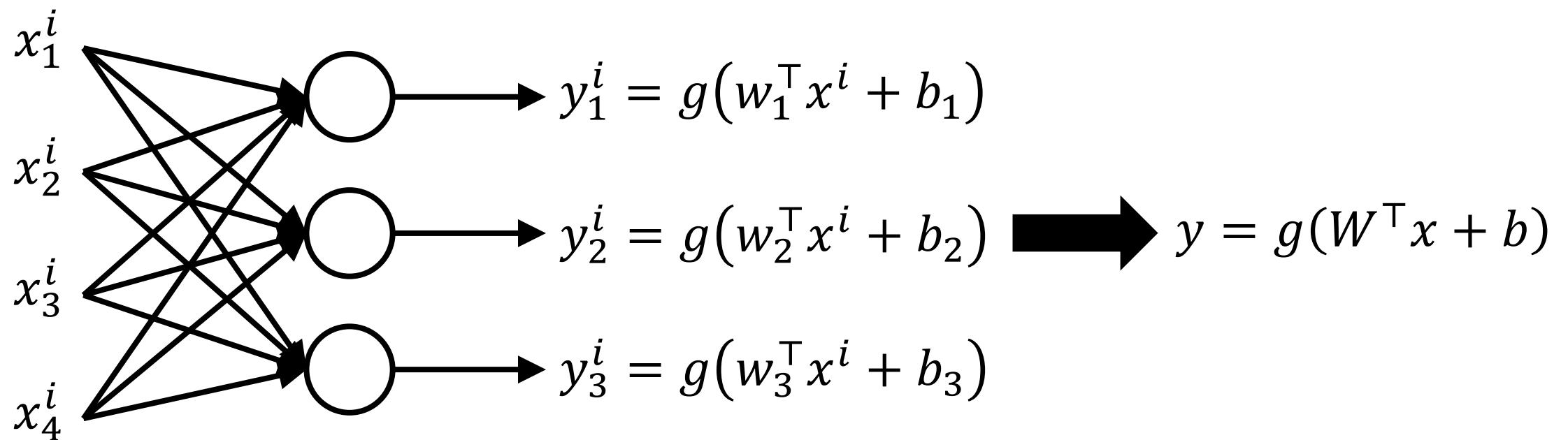
# Convolutional neural networks (CNN)

- Traditionally consists of:
  - Convolutional layers
  - Nonlinearity layers
  - Pooling layers
  - Fully-connected layers



# From lecture 1

A fully-connected layer

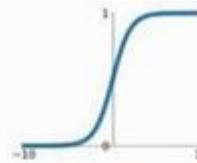


# From lecture 1

Different types of nonlinearity layers:

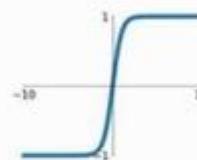
**Sigmoid**

$$\frac{1}{1+e^{-x}}$$



**tanh**

$$\tanh(x)$$



**ReLU**

$$\max(0, x)$$



**Leaky ReLU**

$$\max(0.1x, x)$$



**Maxout**

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

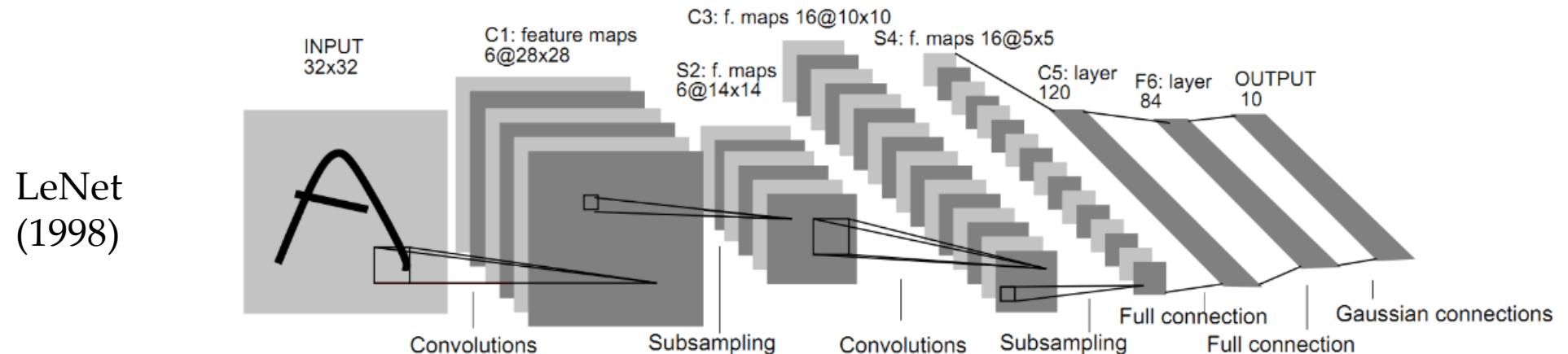
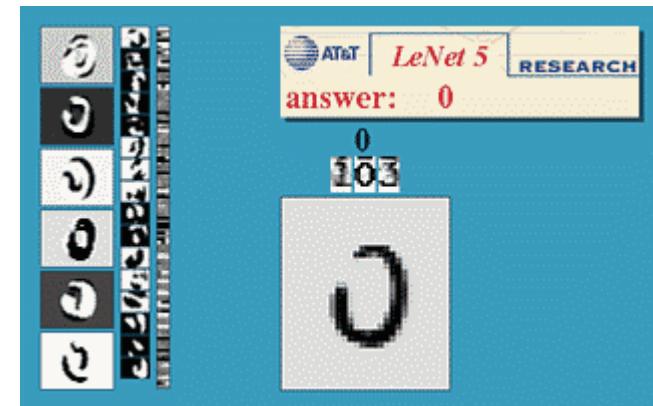
**ELU**

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



# Convolutional neural networks (CNN)

- Traditionally consists of:
  - Convolutional layers
  - Nonlinearity layers
  - Pooling layers
  - Fully-connected layers



# Classification with CNNs

## Classification



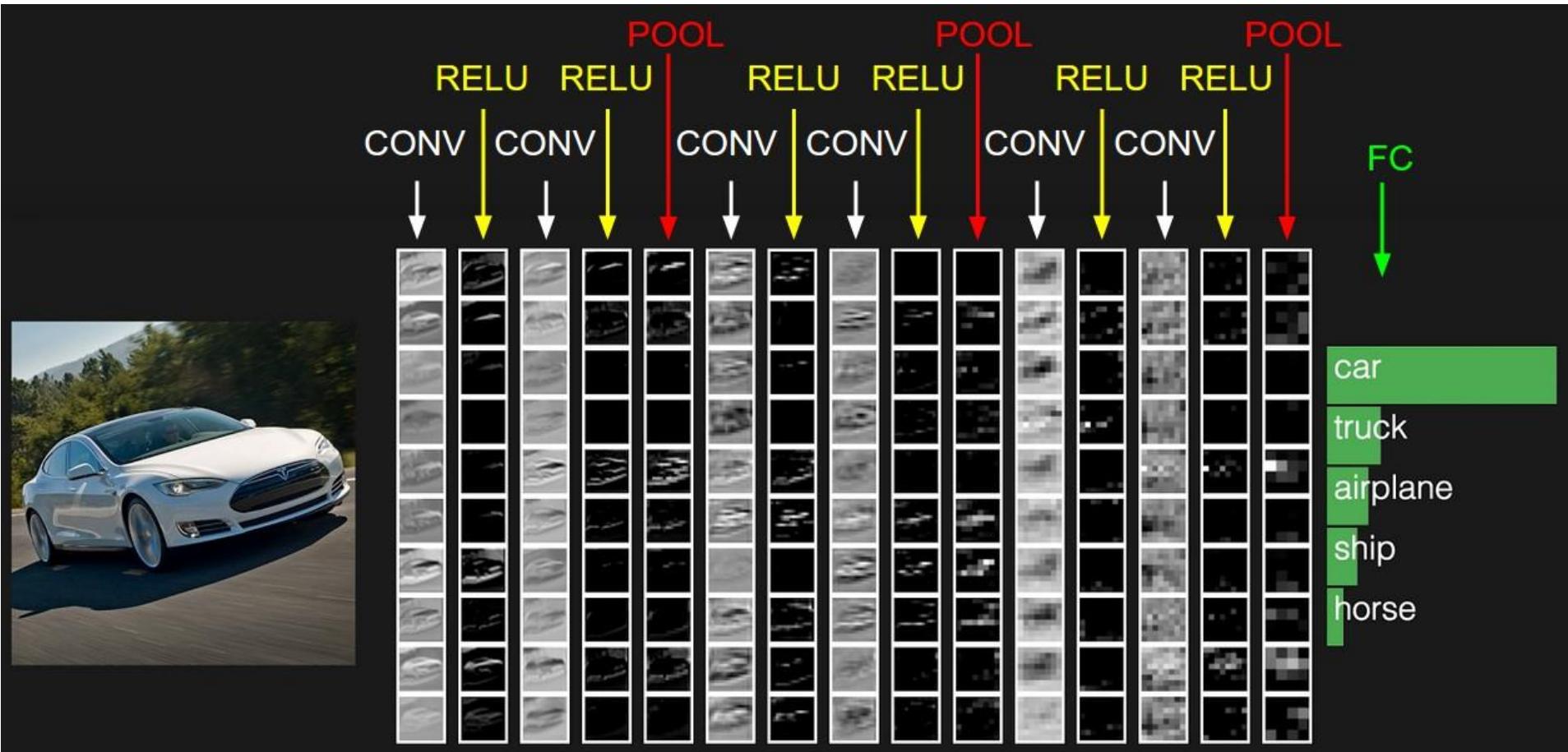
CAT

# Classification via hand-crafted features

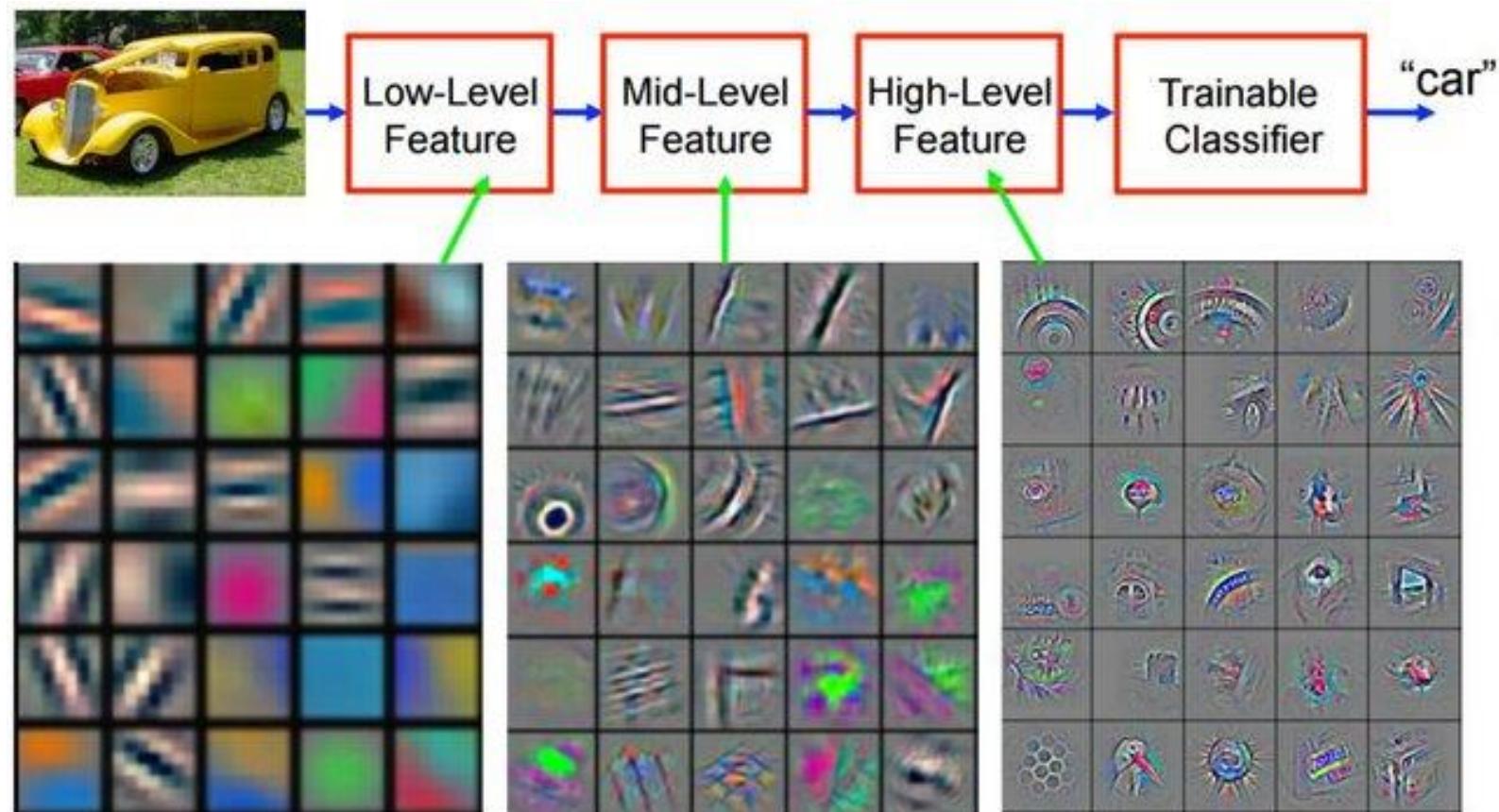
Uses image statistics and keypoints as features.

Trains a classifier over those features.

# End-to-end training



# Visualizing CNN features



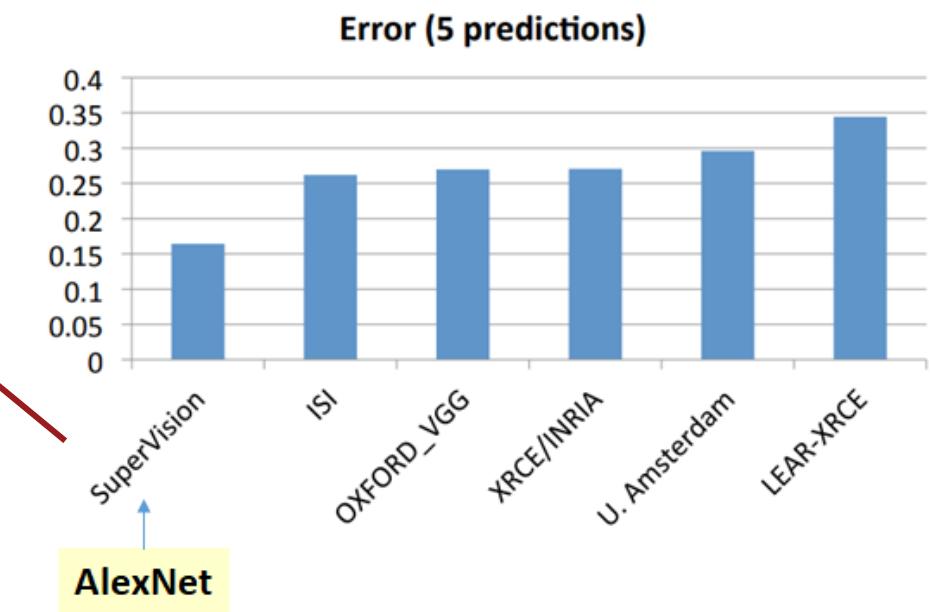
# Online CNN demo

[https://adamharley.com/nn\\_vis/](https://adamharley.com/nn_vis/)

# Comparison

Team name	Filename	Error (5 guesses)	Description
SuperVision	test-preds-141-146.2009-131-137-145-146.2011-145f.	0.15315	Using extra training data from ImageNet Fall 2011 release
SuperVision	test-preds-131-137-145-135-145f.txt	0.16422	Using only supplied training data
ISI	pred_FVs_wLACs_weighted.txt	0.26172	Weighted sum of scores from each classifier with SIFT+FV, LBP+FV, GIST+FV, and CSIFT+FV, respectively.
ISI	pred_FVs_weighted.txt	0.26602	Weighted sum of scores from classifiers using each FV.
ISI	pred_FVs_summed.txt	0.26646	Naive sum of scores from classifiers using each FV.
ISI	pred_FVs_wLACs_summed.txt	0.26952	Naive sum of scores from each classifier with SIFT+FV, LBP+FV, GIST+FV, and CSIFT+FV, respectively.
OXFORD_VGG	test_adhocmix_classification.txt	0.26979	Mixed selection from High-Level SVM scores and Baseline Scores, decision is performed by looking at the validation performance

Alex Krizhevsky,  
Ilya Sutskever,  
Geoffrey Hinton

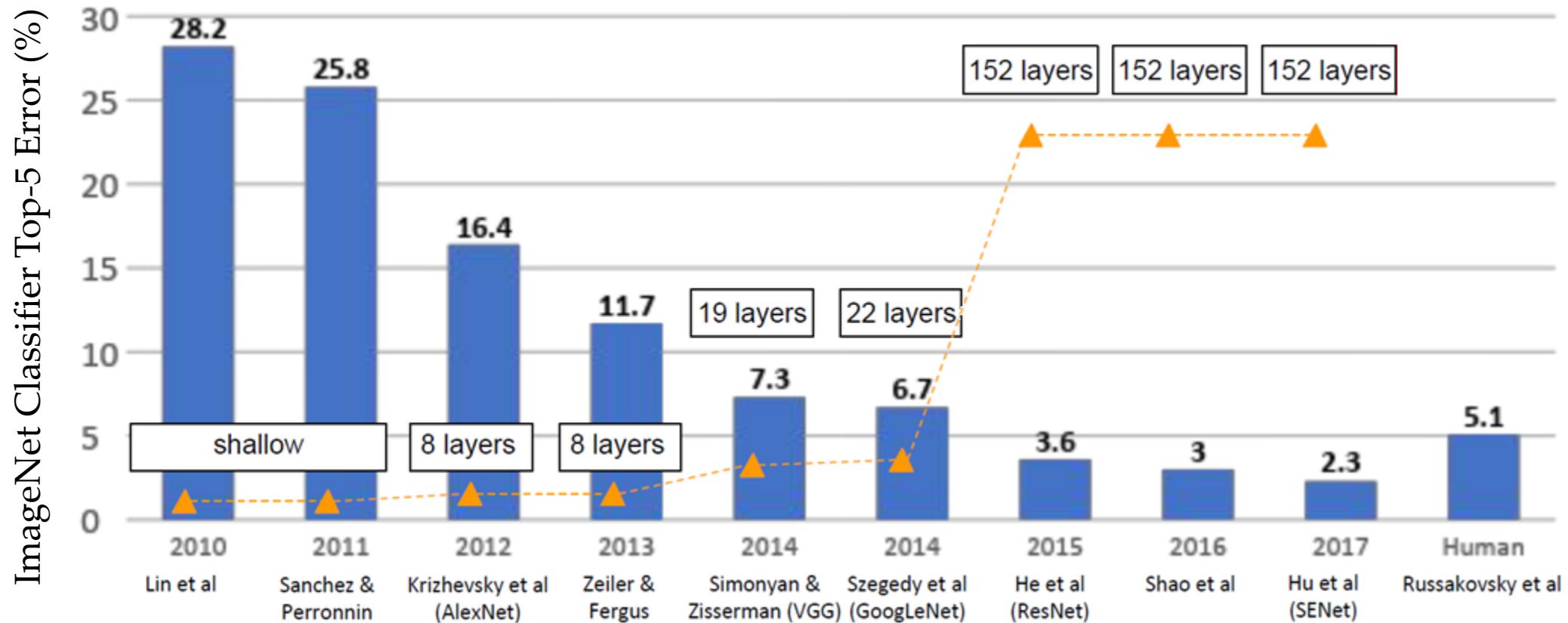


# Comparison

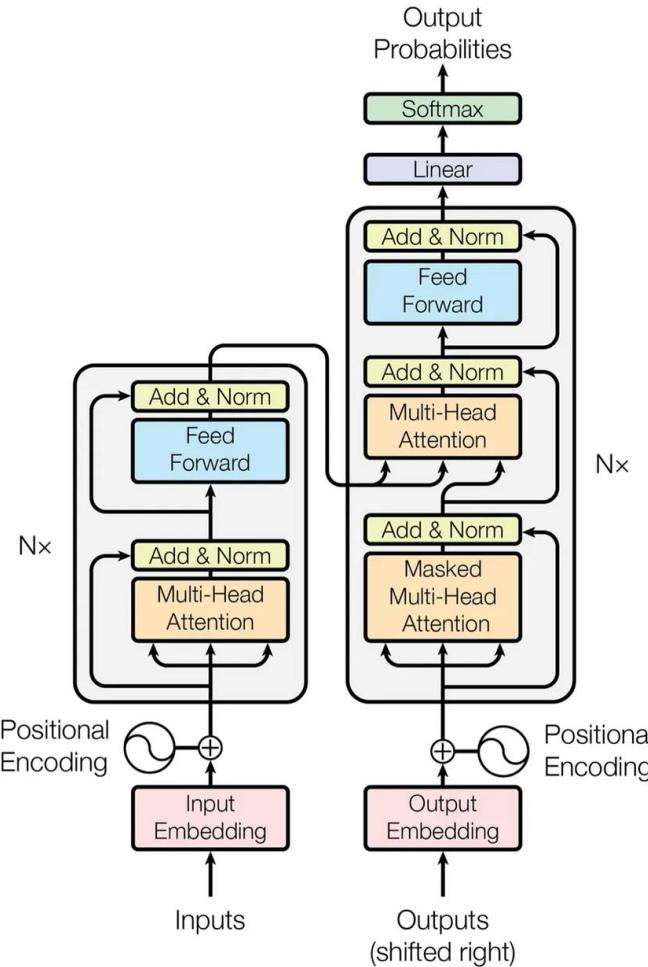
## The description of SuperVision:

*Our model is a large, deep convolutional neural network trained on raw RGB pixel values. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three globally-connected layers with a final 1000-way softmax. It was trained on two NVIDIA GPUs for about a week. To make training faster, we used non-saturating neurons and a very efficient GPU implementation of convolutional nets. To reduce overfitting in the globally-connected layers we employed hidden-unit "dropout", a recently-developed regularization method that proved to be very effective.*

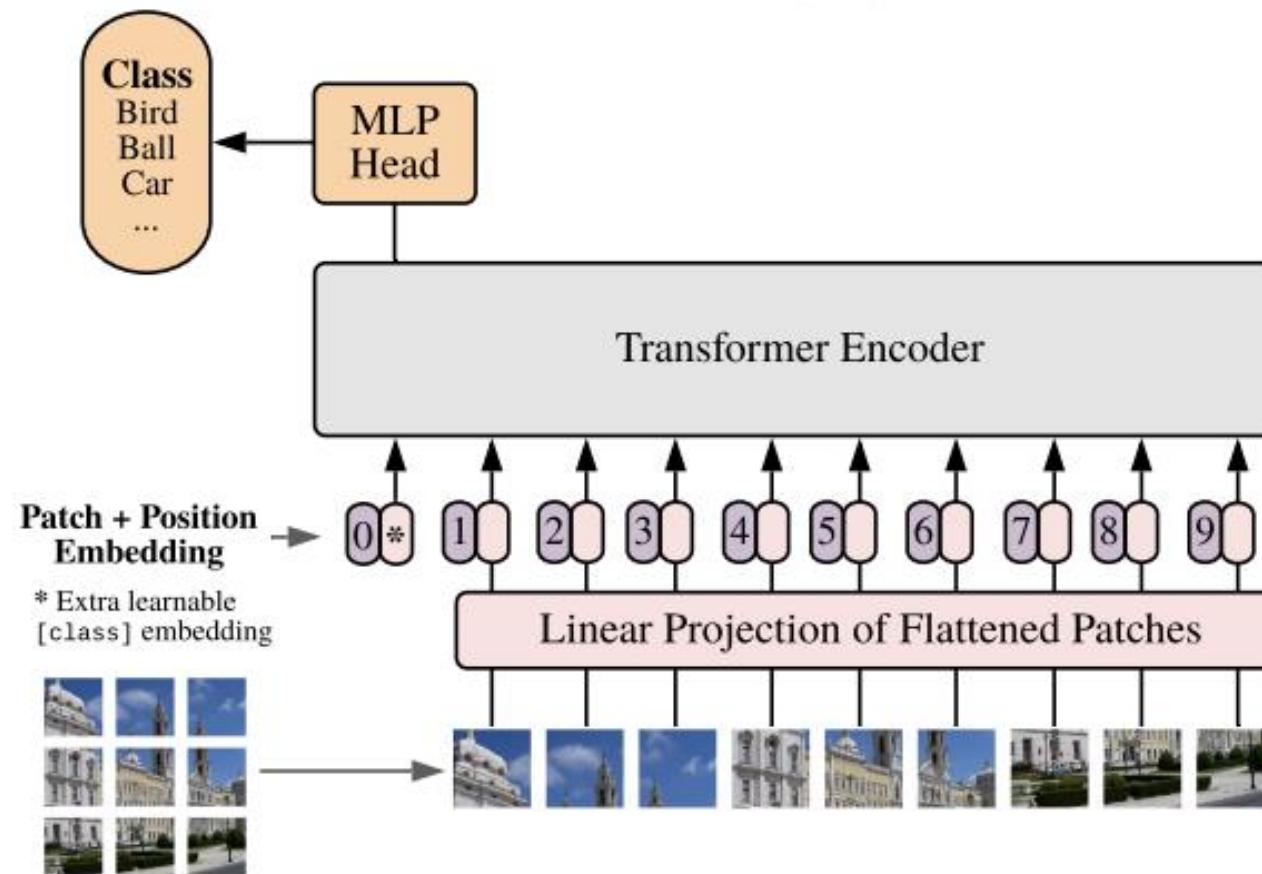
# Deeper and deeper



# Transformer



# Vision transformers (ViT)



# Robotics applications of vision

## Classification



CAT

# Robotics applications of vision

**Classification**



CAT

**Classification  
+ Localization**

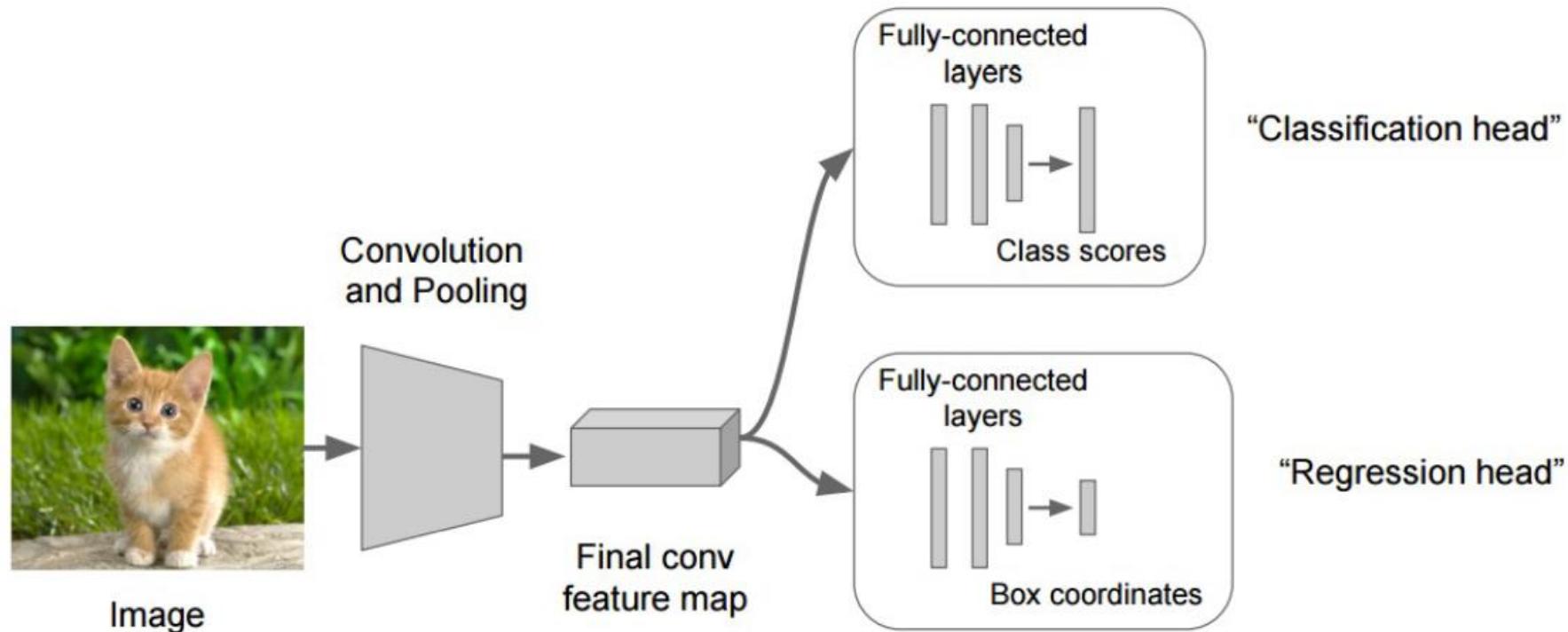


CAT

# Classification + Localization

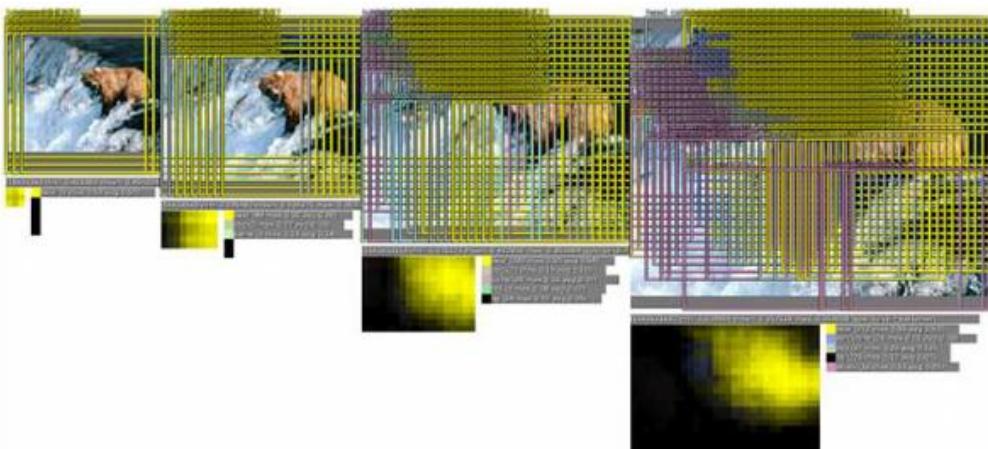
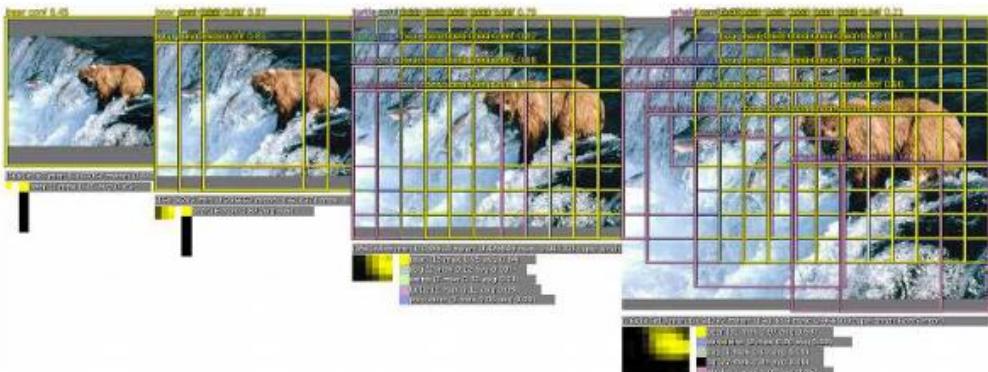
Do not just output the image class.

Output the class and 4 numbers, one for each corner of the bounding box.



# Classification + Localization

Or just use a classifier multiple times.



# Robotics applications of vision

**Classification**



CAT

**Classification  
+ Localization**



CAT

Single object

# Robotics applications of vision

**Classification**



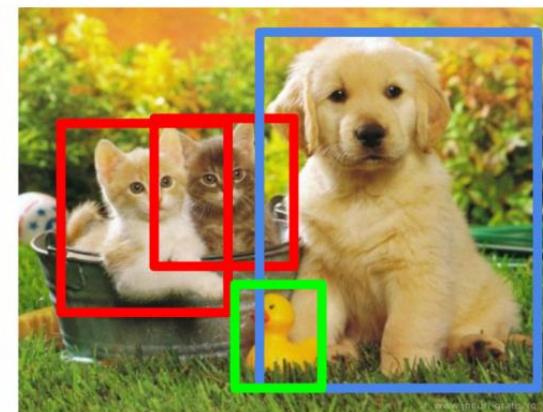
CAT

**Classification + Localization**



CAT

**Object Detection**



CAT, DOG, DUCK

Single object

# Robotics applications of vision

**Classification**



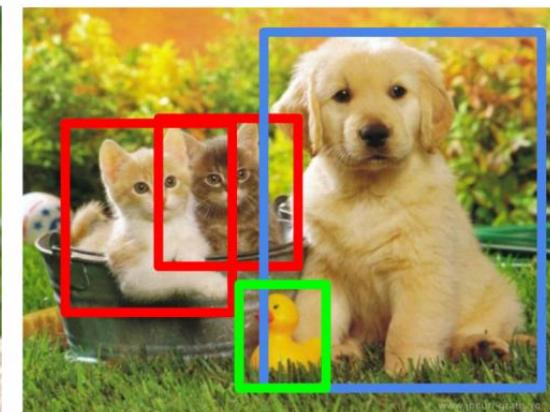
CAT

**Classification + Localization**



CAT

**Object Detection**



CAT, DOG, DUCK

**Instance Segmentation**

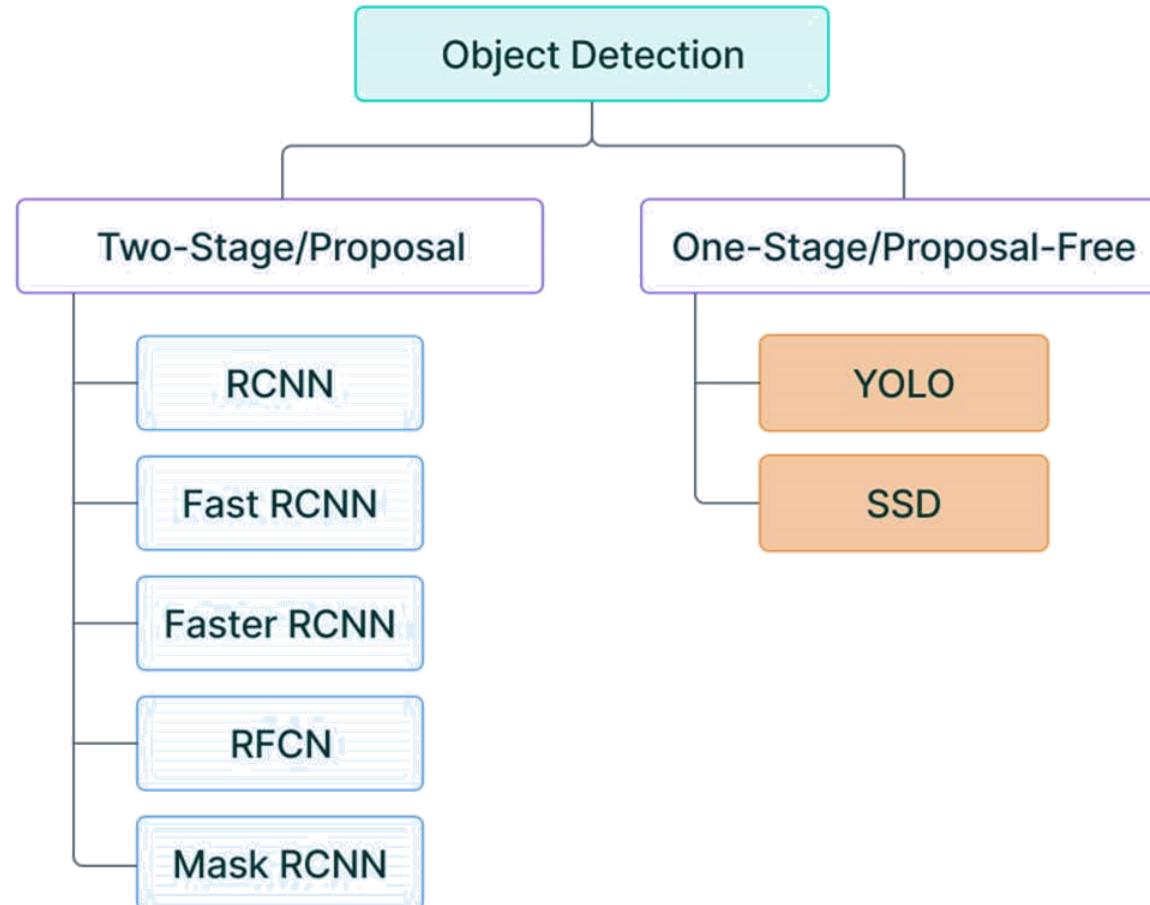


CAT, DOG, DUCK

Single object

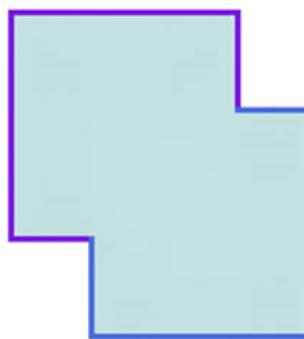
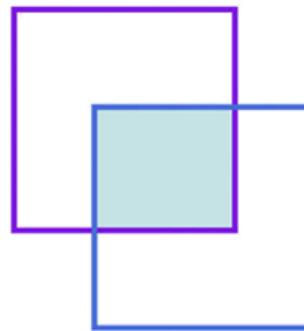
Multiple objects

# Object detection



# Metric for object detection

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}} =$$



# Object detection with YOLO

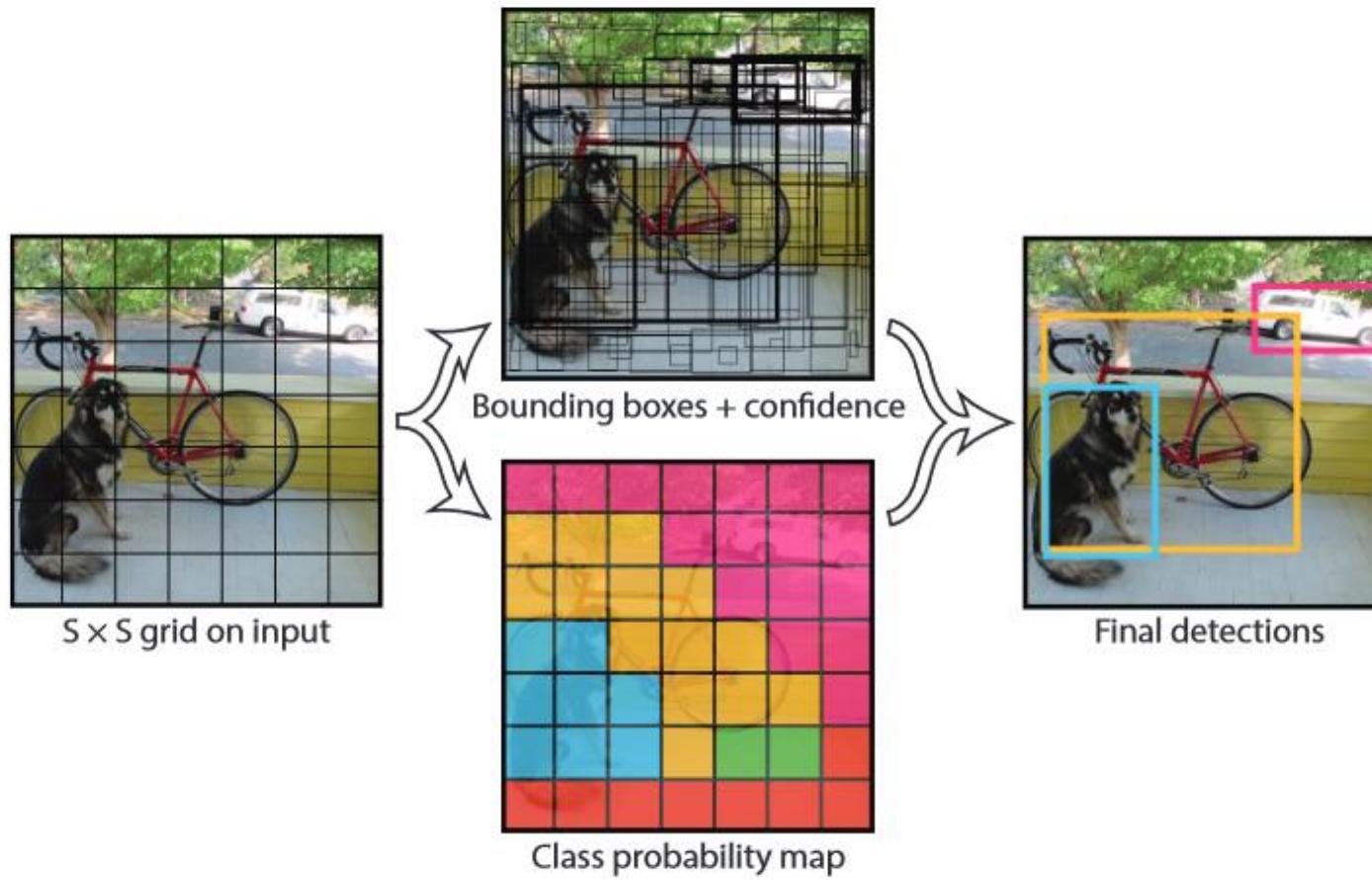
Divide image into  $S \times S$  grid. For each cell, predict:

- $B$  boxes with 4 coordinates and 1 confidence score each, and
- $C$  class scores.

This gives a tensor of  $S \times S \times (5B + C)$ .

Apply non-maximum suppression to finalize.

# Object detection with YOLO



# Object detection in robotics

Speed is crucial – we need to detect, track, and classify many objects with high frequency.

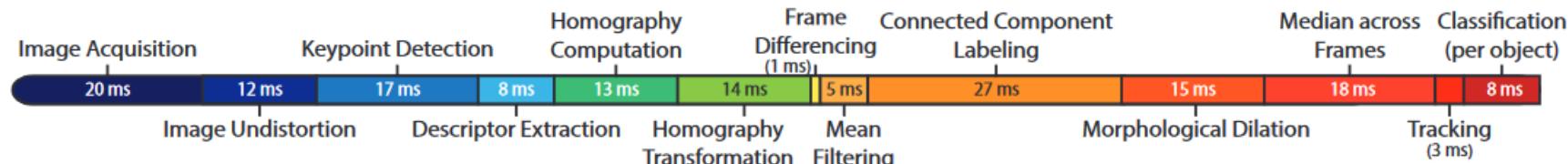
## Real-Time Detection, Tracking and Classification of Multiple Moving Objects in UAV Videos

Hüseyin C. Baykara\*, Erdem Biyik\*, Gamze Gül\*, Deniz Onural\*, Ahmet S. Öztürk\*, İlkay Yıldız\*

*International Conference on Tools with Artificial Intelligence (ICTAI), November 2017*

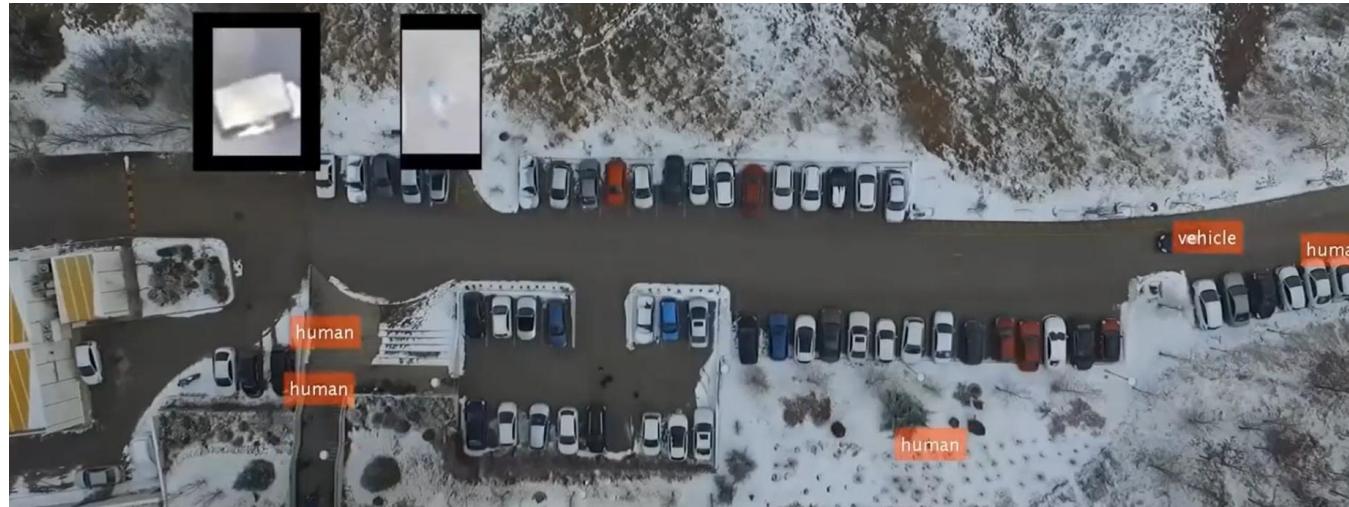


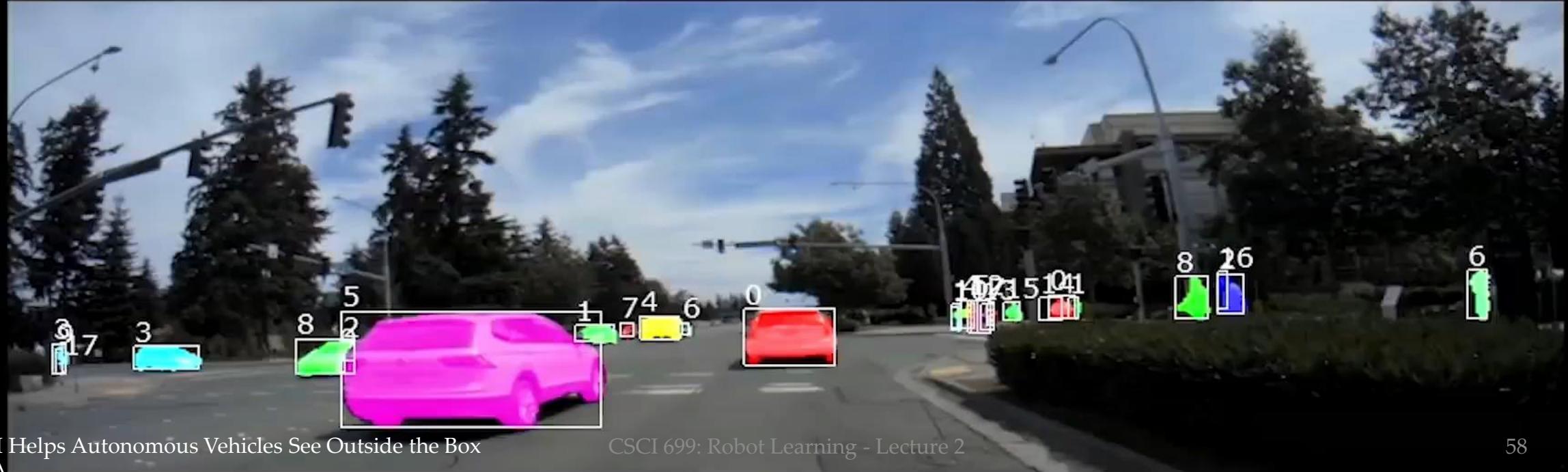
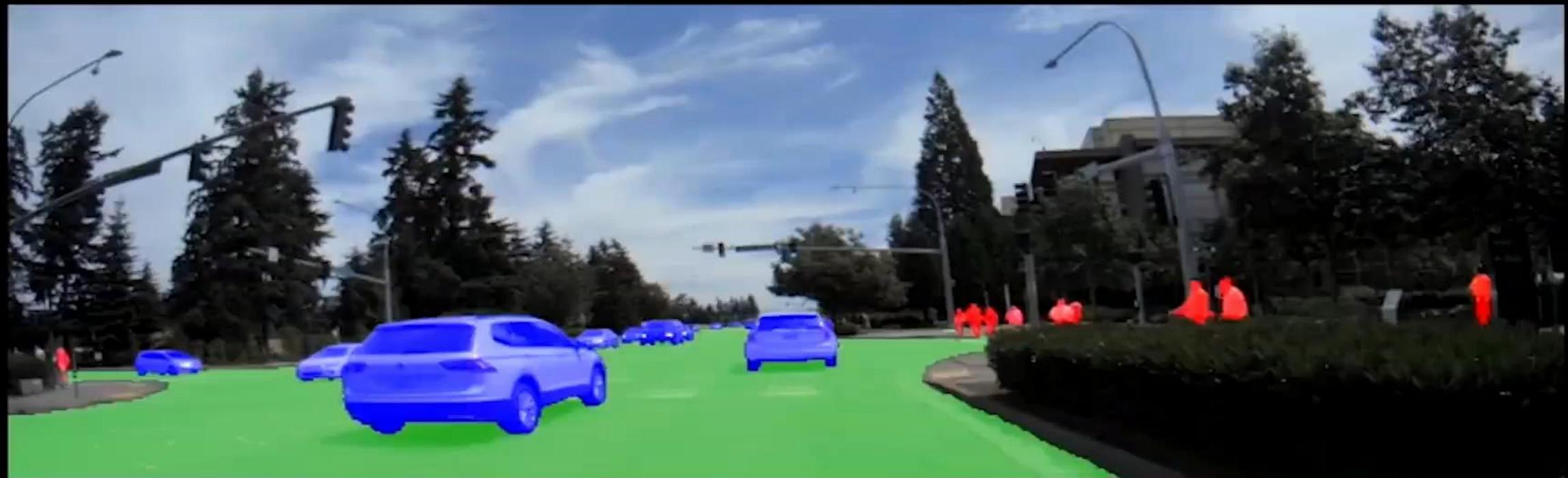
\* denotes equal contribution.



# Object detection in robotics

Speed is crucial – we need to detect, track, and classify many objects with high frequency.





# Today

- Basics of computer vision for robotics
- Representation learning

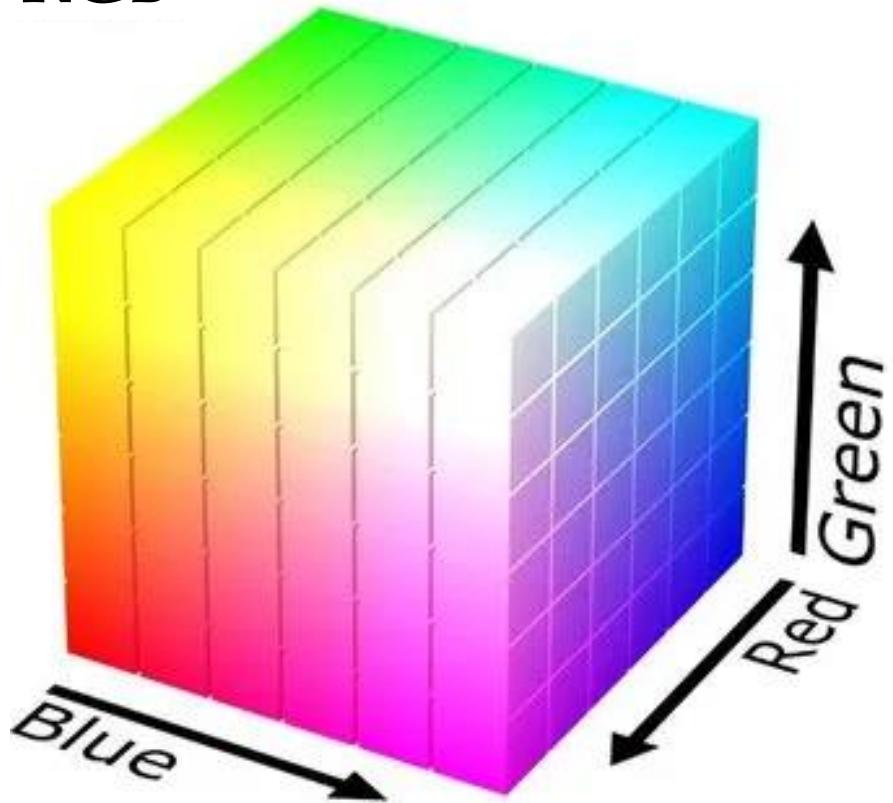
# Representation learning

A good representation:

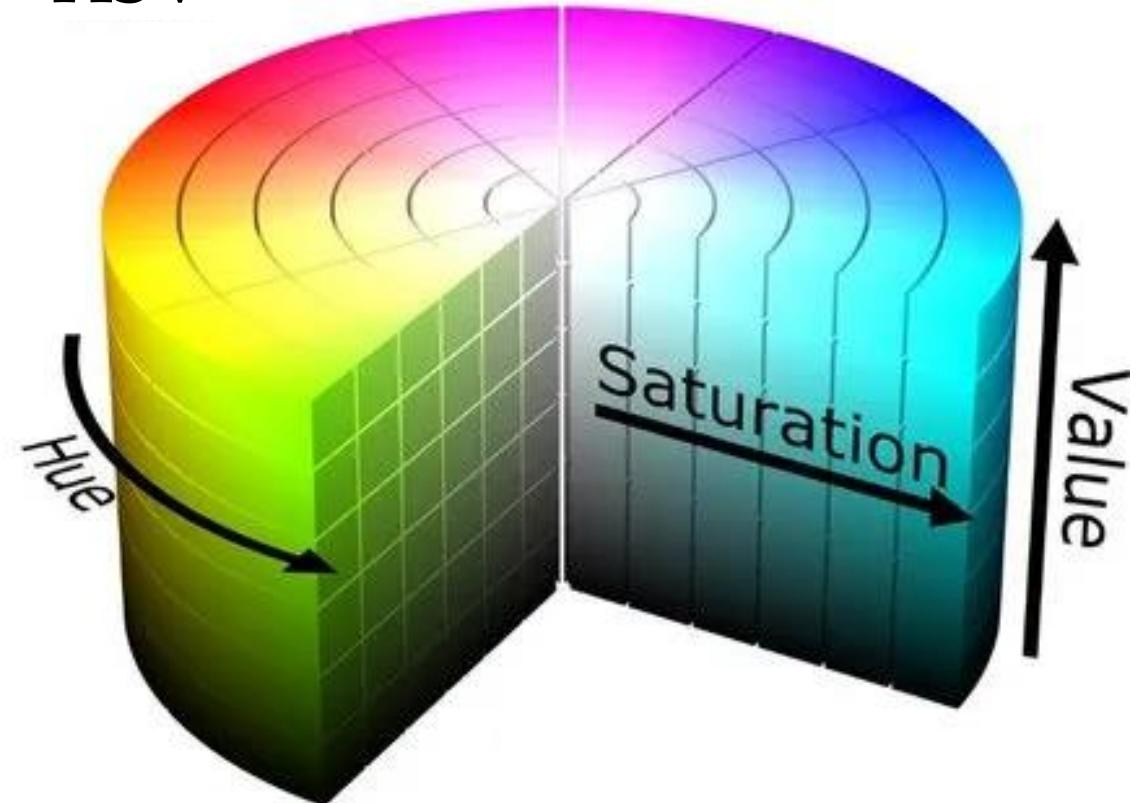
- contains the aspects of the raw data that are important for the downstream task
- is usually compact
- is ideally (but rarely) interpretable

# Examples of representations

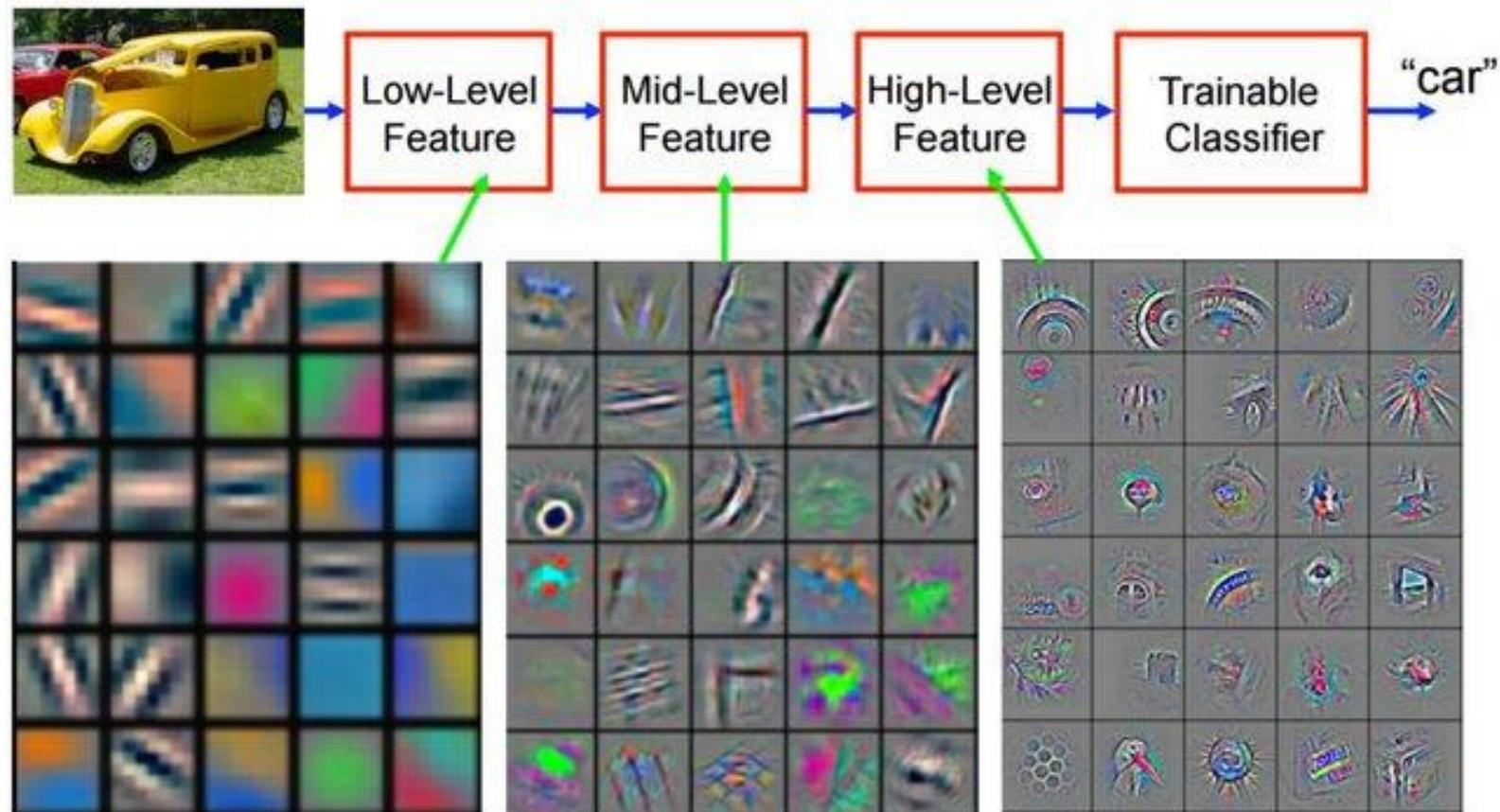
RGB



HSV



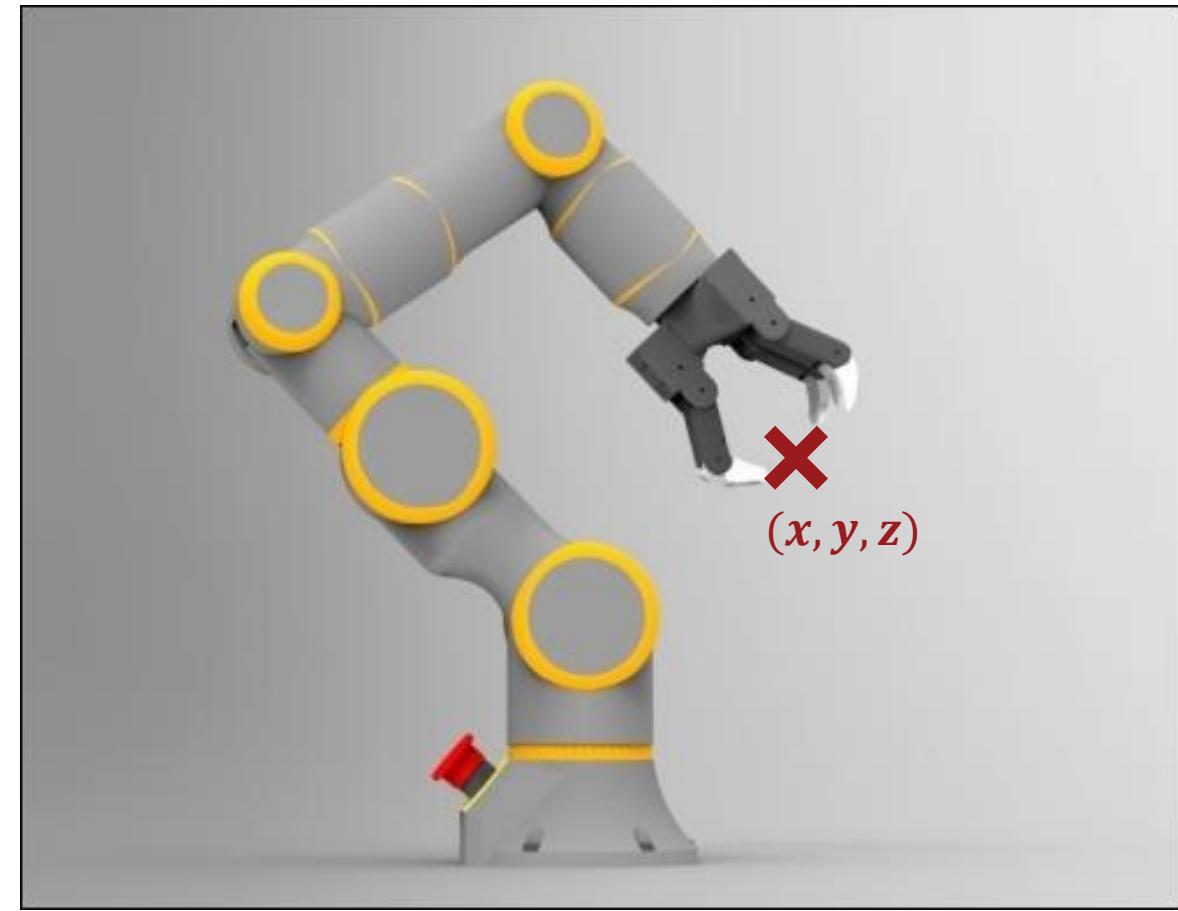
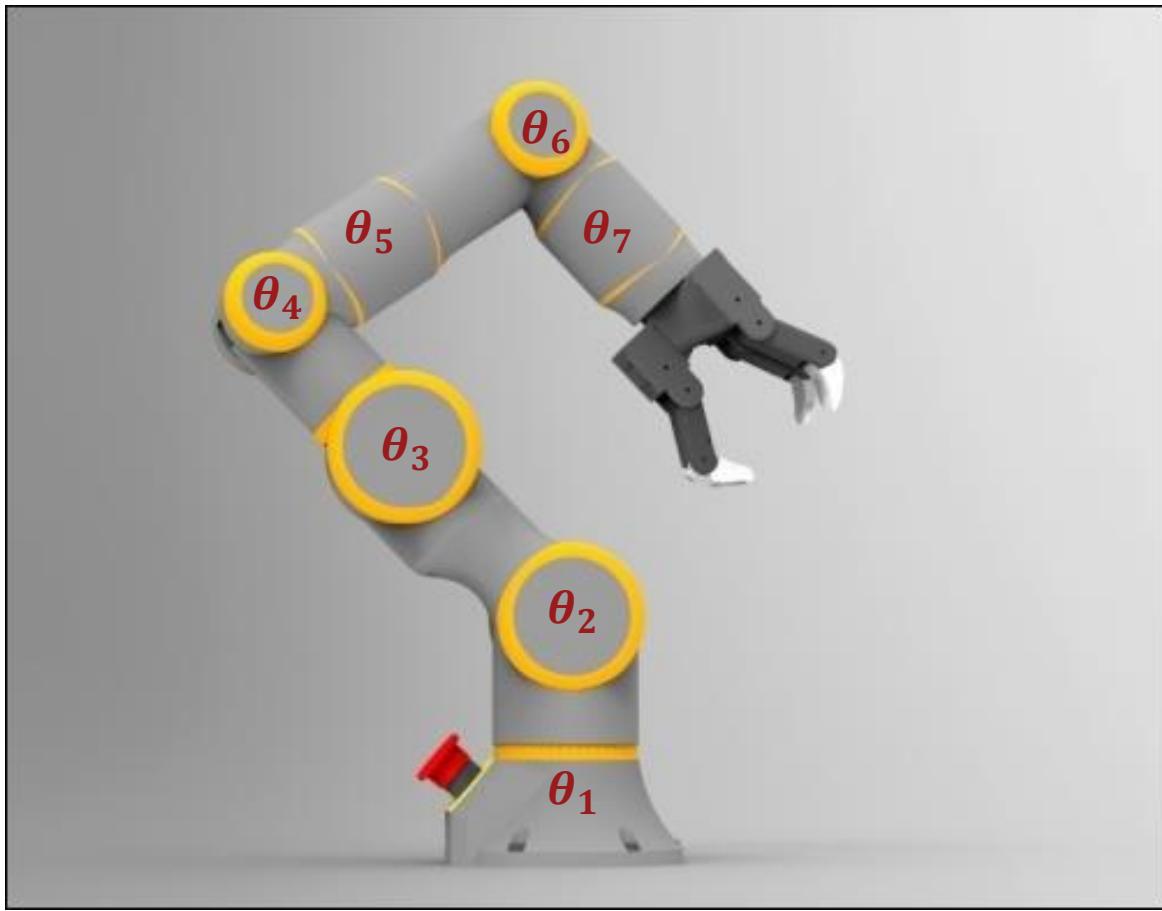
# Examples of representations



# Examples of representations



# Examples of representations



# How do we learn representations?

So many methods that it could be a course on its own.

<b>Dynamics of Representation Learning</b>							
30117D	048	4.0	Lecture	12:00-1:50pm	Mon, Wed	23 of 28	 Greg Ver Steeg

<b>Advanced Topics in Representation Learning for NLP</b>							
30111D	048	4.0	Lecture	2:00-5:20pm	Tuesday	16 of 18	 Xiang Ren

<b>Representation Learning: Theory and Practice</b>							
30220D	048	4.0	Lecture	10:00-11:50am	Mon, Wed	51 of 52	 Aram Galstyan,  Greg Ver Steeg

# How do we learn representations?

So many methods that it could be a course on its own.

We will cover only two:

- Autoencoders
- Variational autoencoders

# Principal component analysis (PCA)

PCA is a special case of autoencoders.

Raw data:

$$X \in \mathbb{R}^{n \times d}$$

Top  $k$  eigenvectors:

$$V \in \mathbb{R}^{d \times k}$$

Representation:

$$Z = XV \in \mathbb{R}^{n \times k}$$

Reconstruction:

$$\hat{X} = XVV^\top \in \mathbb{R}^{n \times d}$$

# Principal component analysis (PCA)

PCA is a special case of autoencoders.

This is not typical in  
a neural network

Raw data:

$$X \in \mathbb{R}^{n \times d}$$

Top  $k$  eigenvectors:

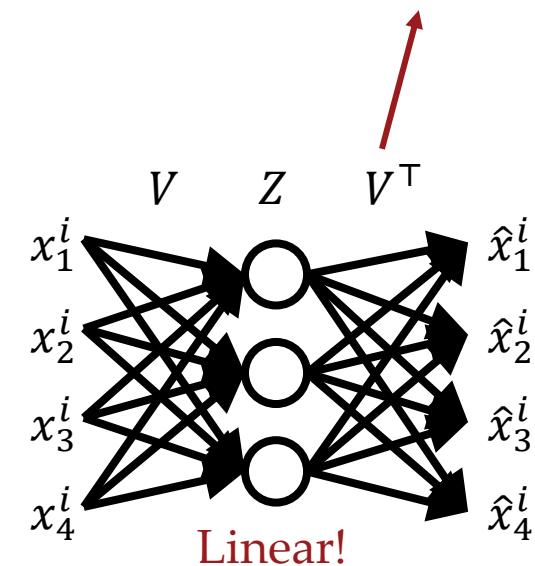
$$V \in \mathbb{R}^{d \times k}$$

Representation:

$$Z = X V \in \mathbb{R}^{n \times k}$$

Reconstruction:

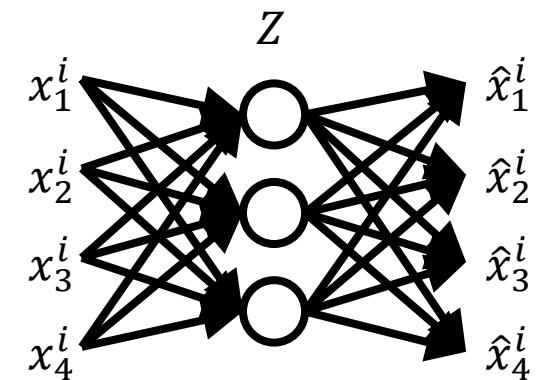
$$\hat{X} = X V V^\top \in \mathbb{R}^{n \times d}$$



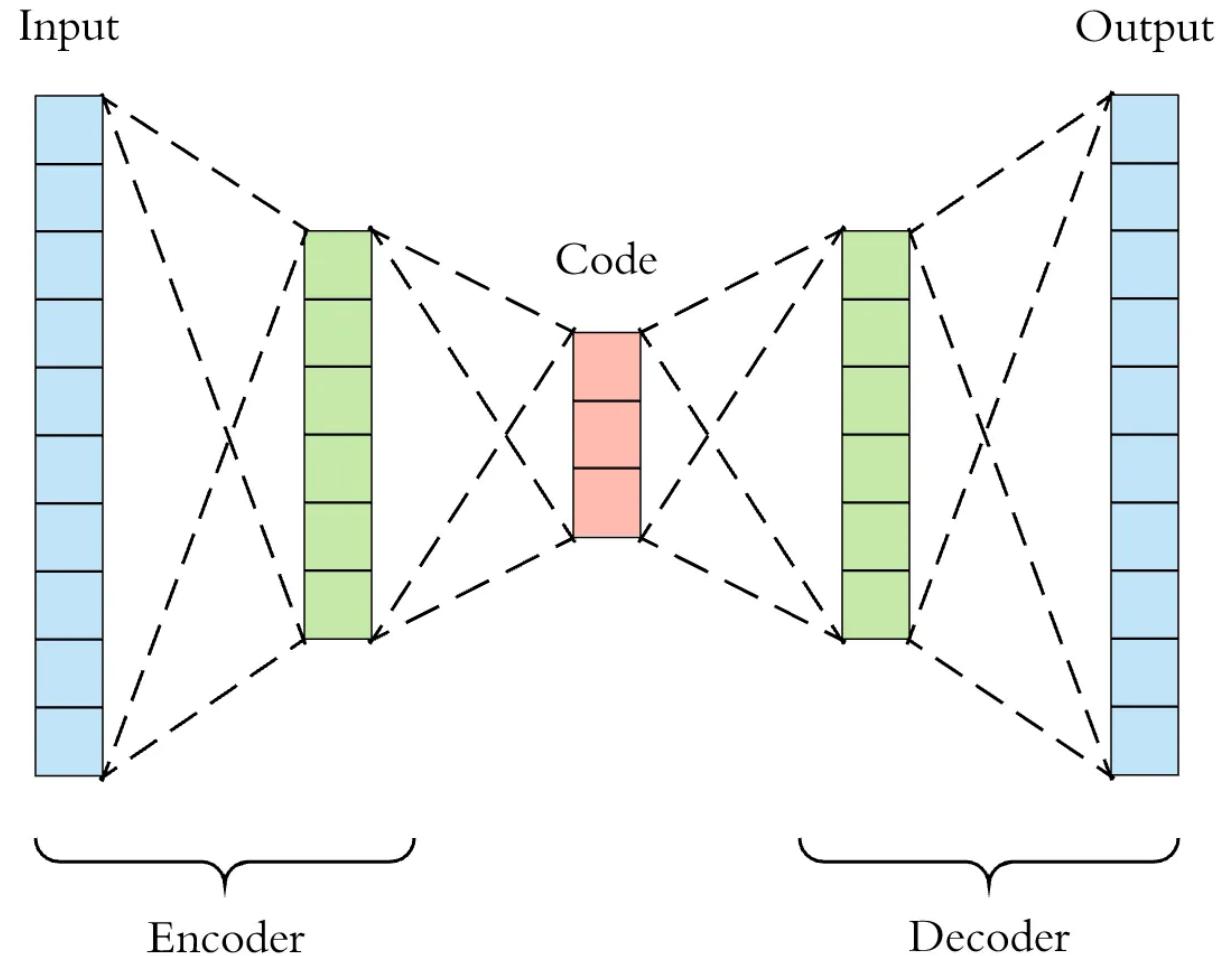
# Principal component analysis (PCA)

PCA is a special case of autoencoders.

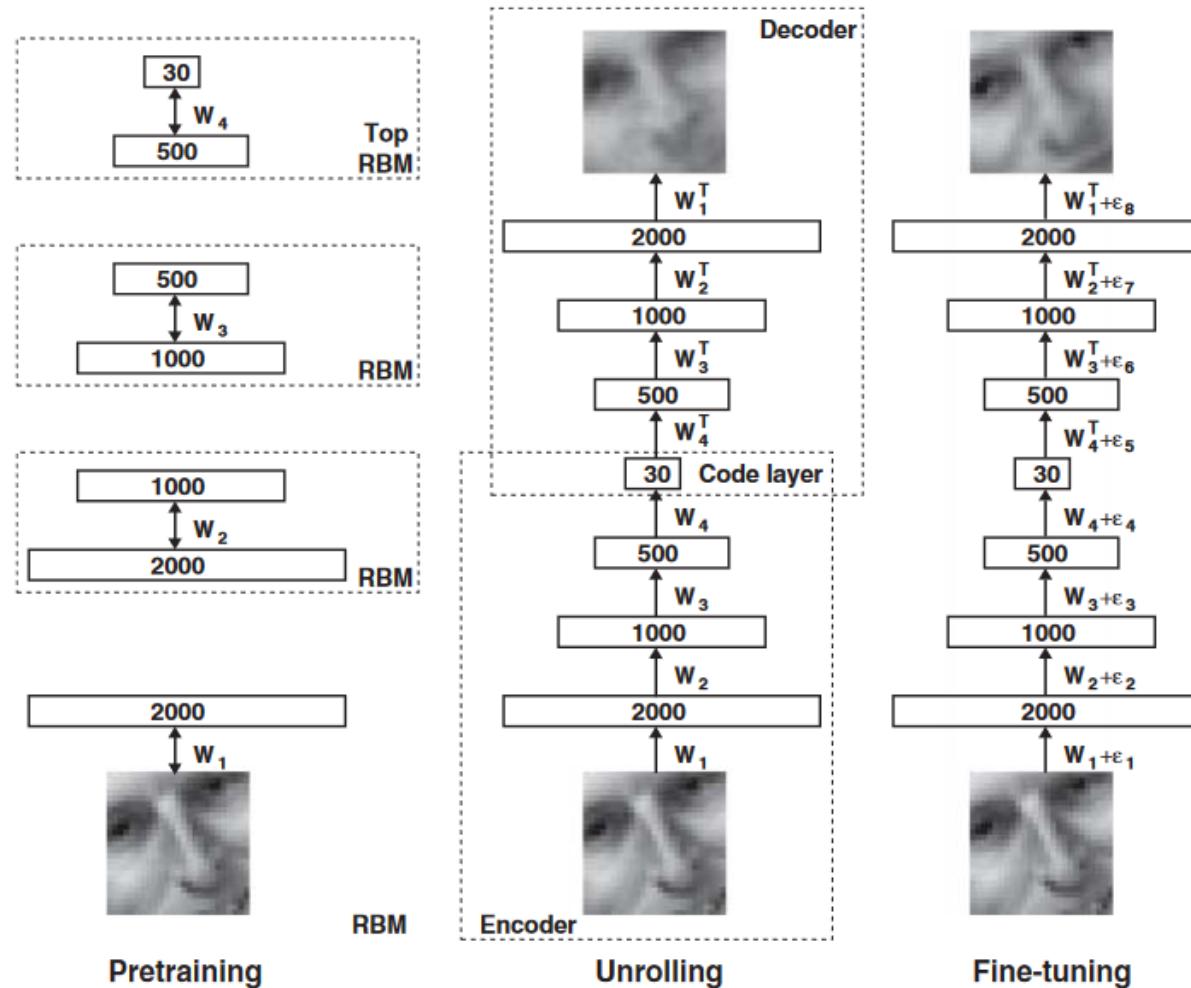
1. Add nonlinearity
  2. Remove the dependence between layers
  3. Possibly add more layers
- Now you have a more powerful autoencoder.



# Autoencoders

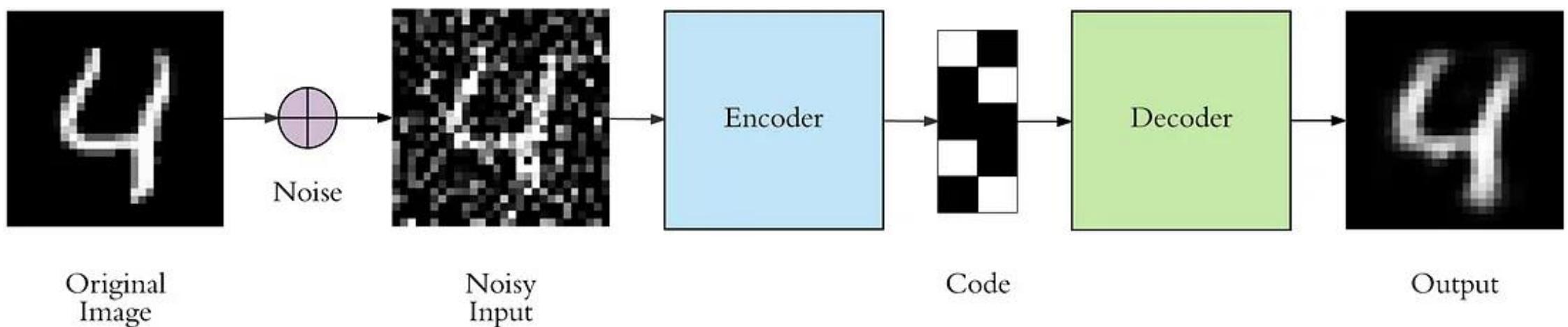


# Autoencoders



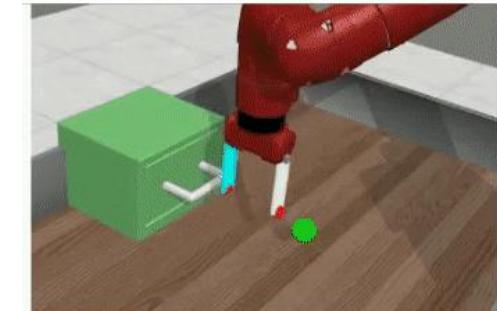
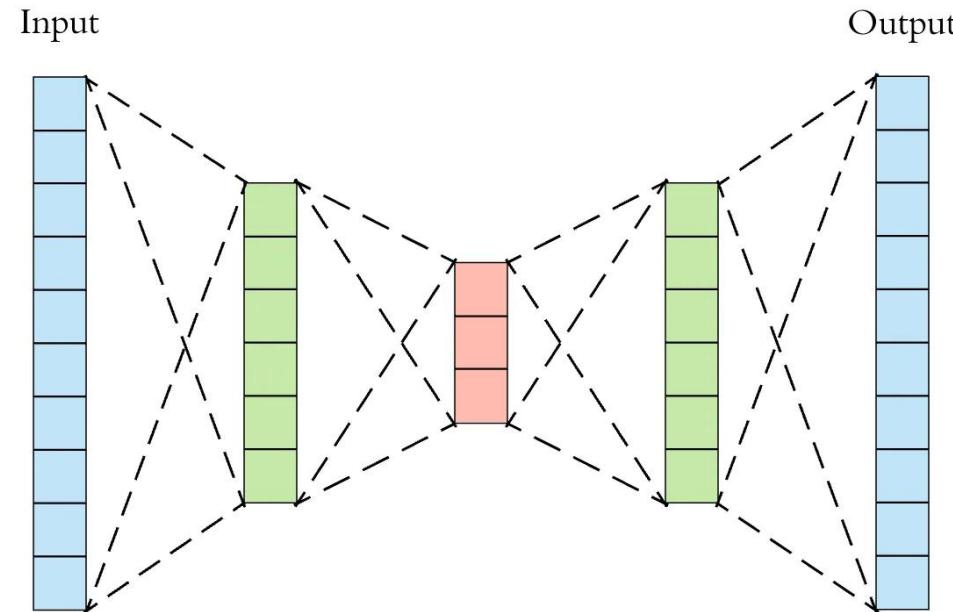
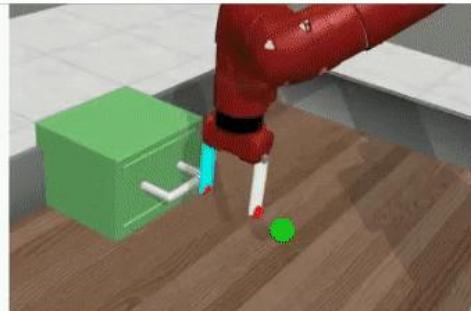
# Applications of autoencoders

- Denoising



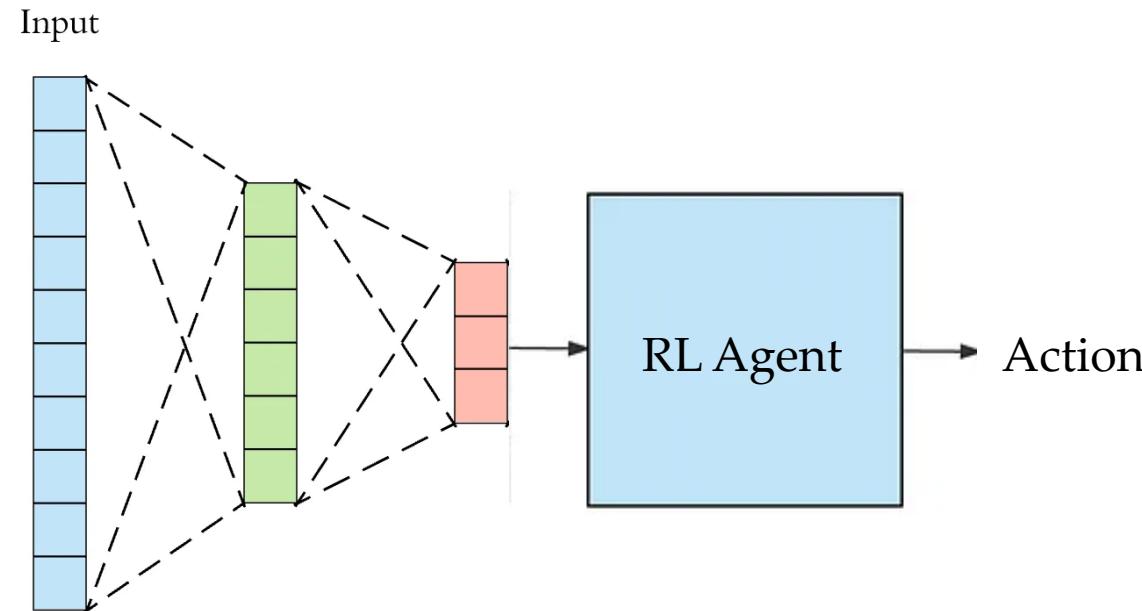
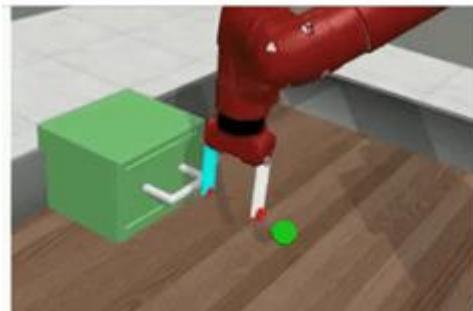
# Applications of autoencoders

- Reinforcement learning



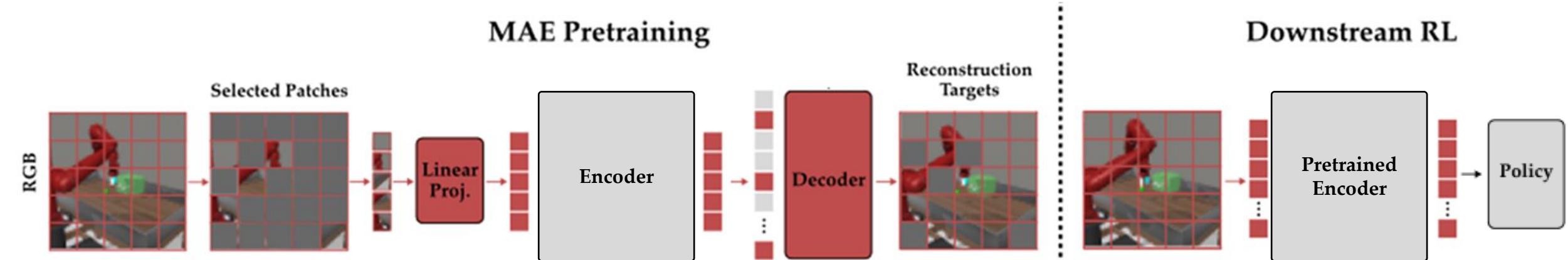
# Applications of autoencoders

- Reinforcement learning

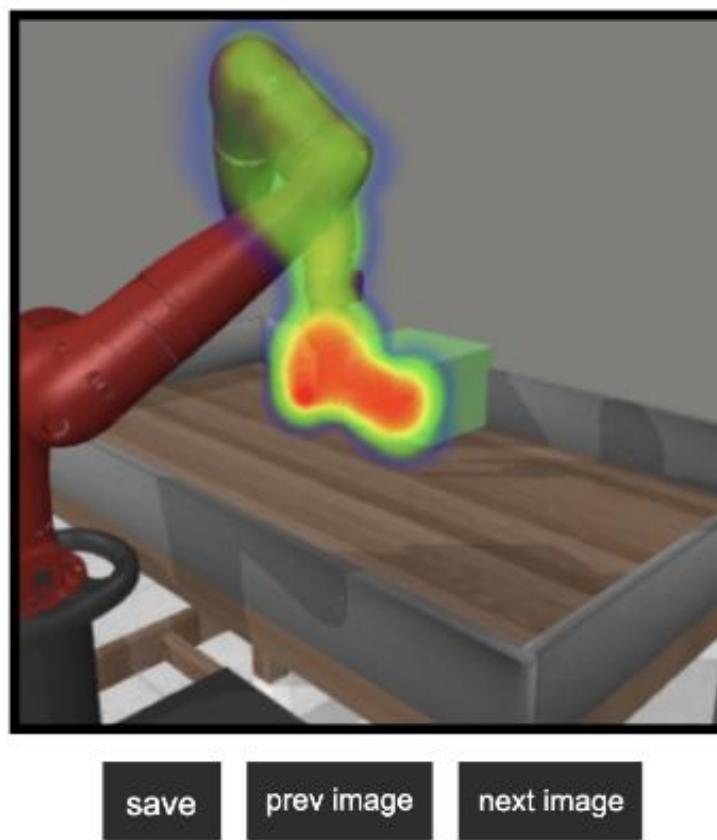


# Applications of autoencoders

- Reinforcement learning

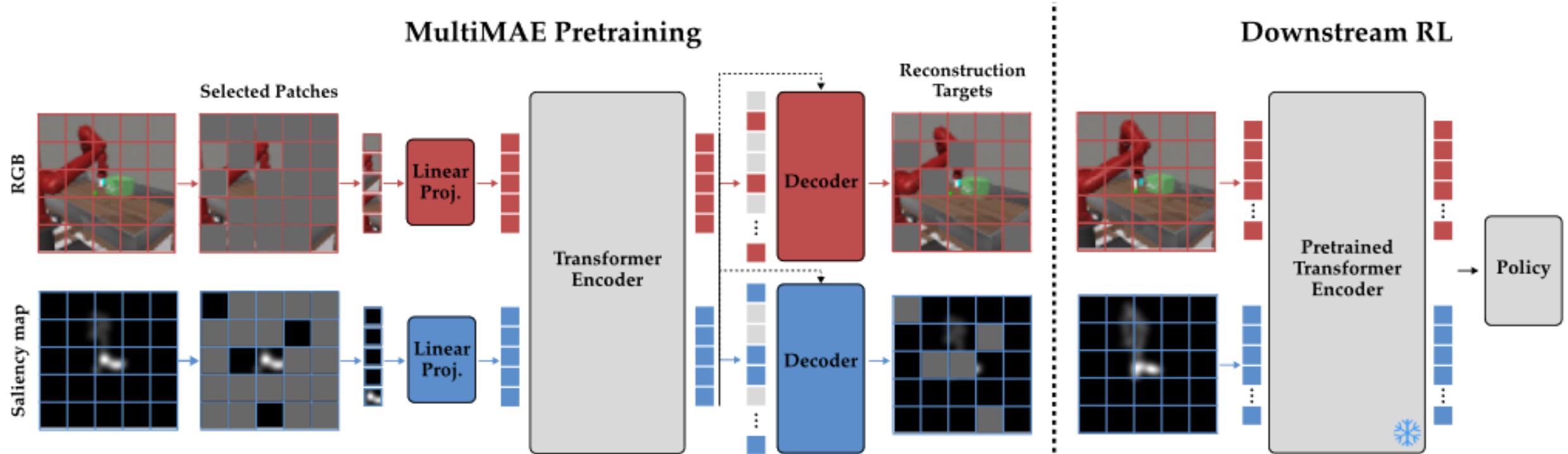


# Applications of autoencoders

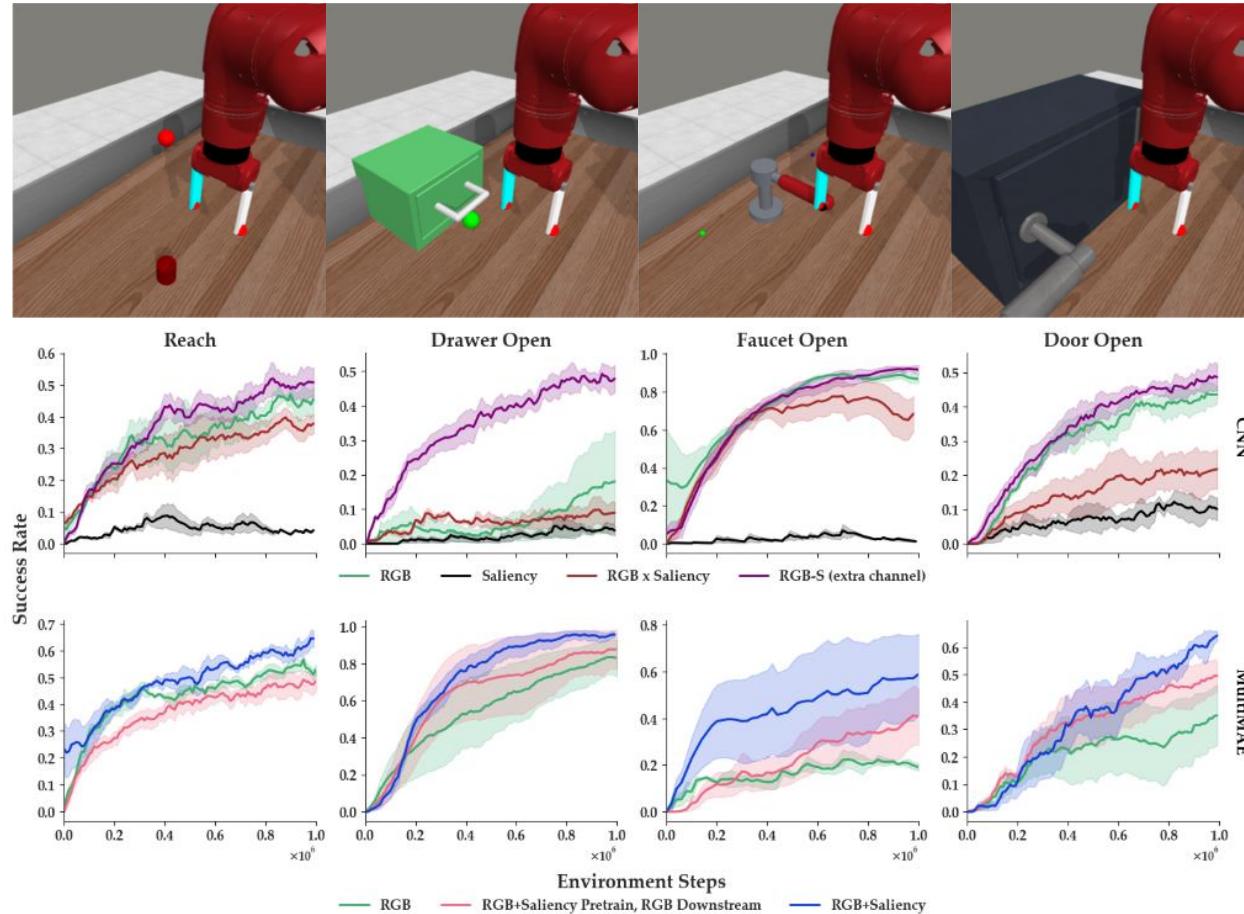


# Applications of autoencoders

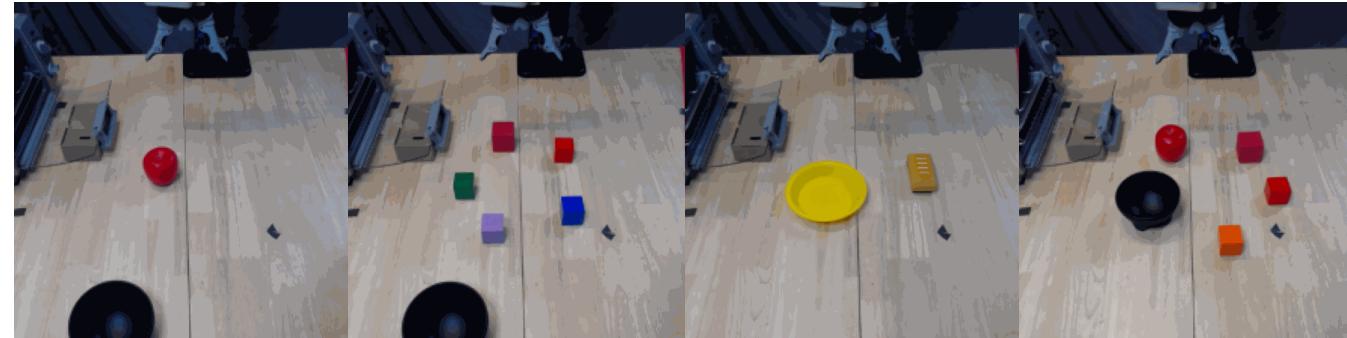
- Reinforcement learning (ViSaRL)



# ViSaRL results



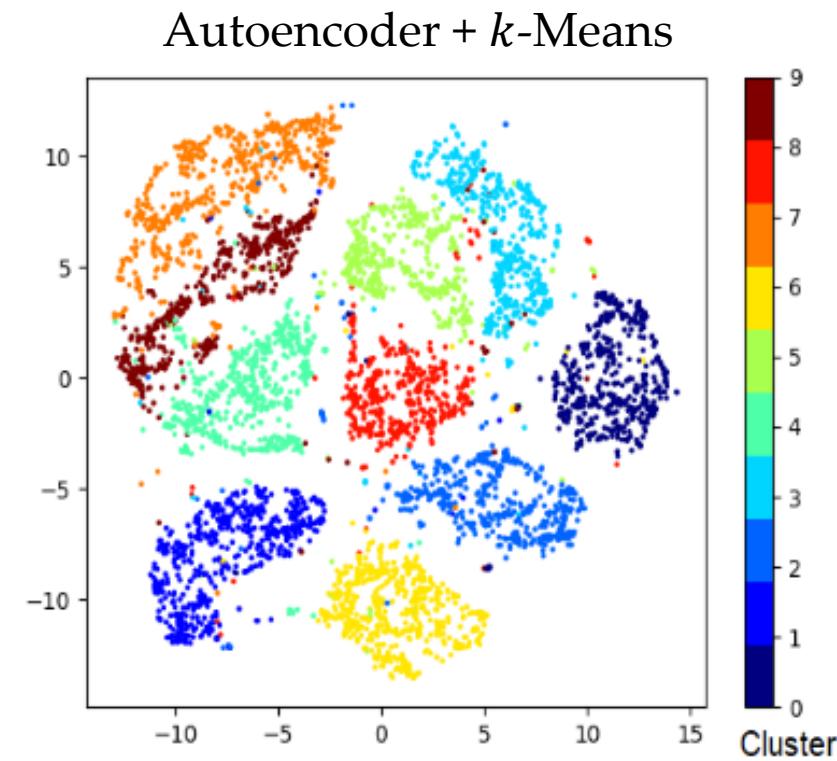
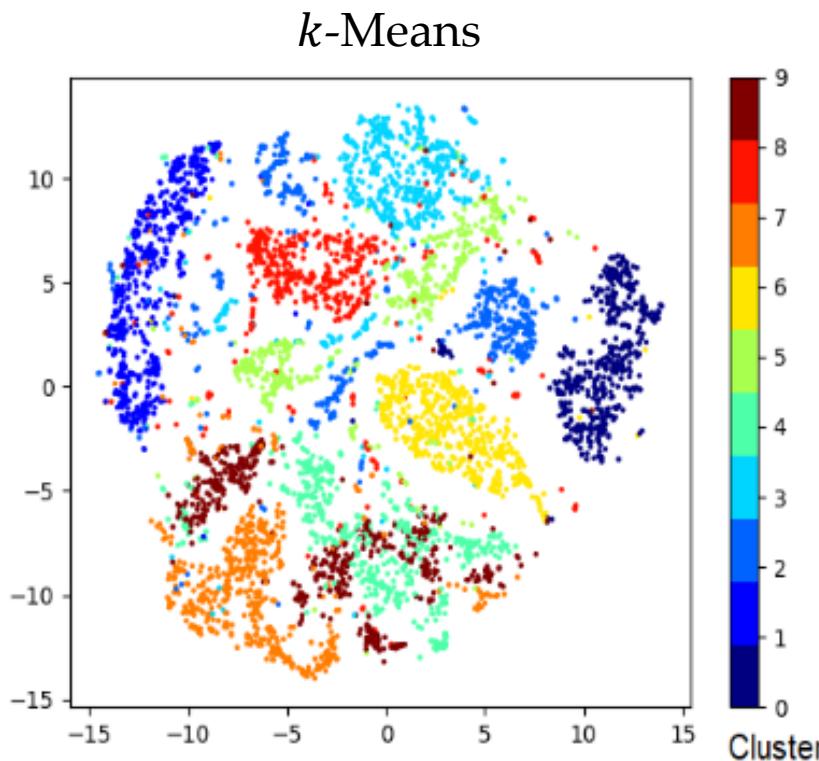
# ViSaRL results



Model	Apple	Red Block	Bread → Plate	Apple → Bowl
MultiMAE (RGB-only)	6/10	4/10	3/10	1/10
MultiMAE (RGB + Saliency)	8/10	7/10	6/10	6/10

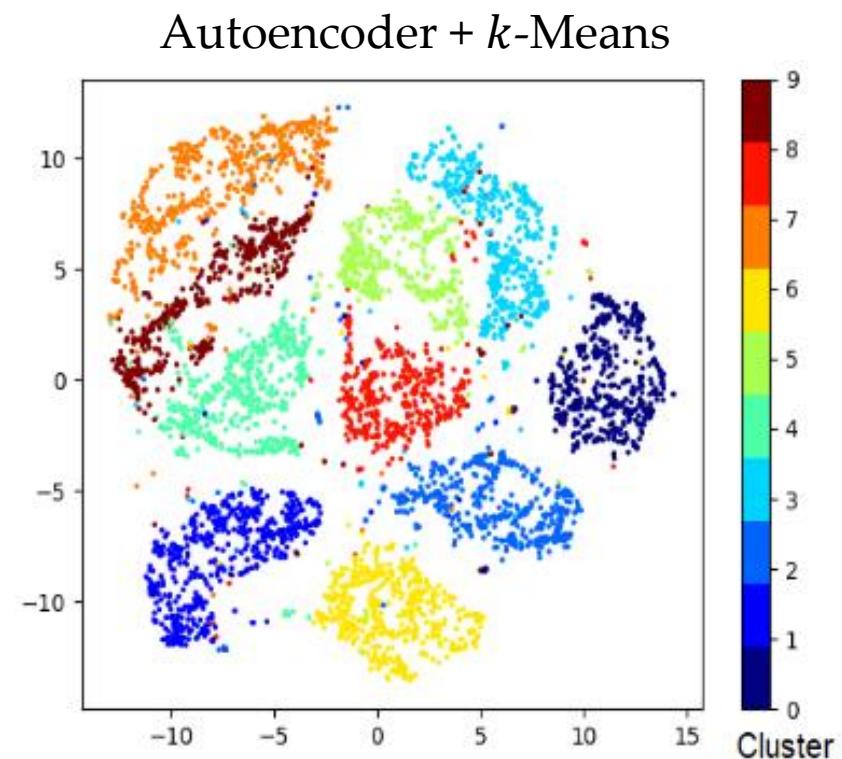
# Applications of autoencoders

- Clustering



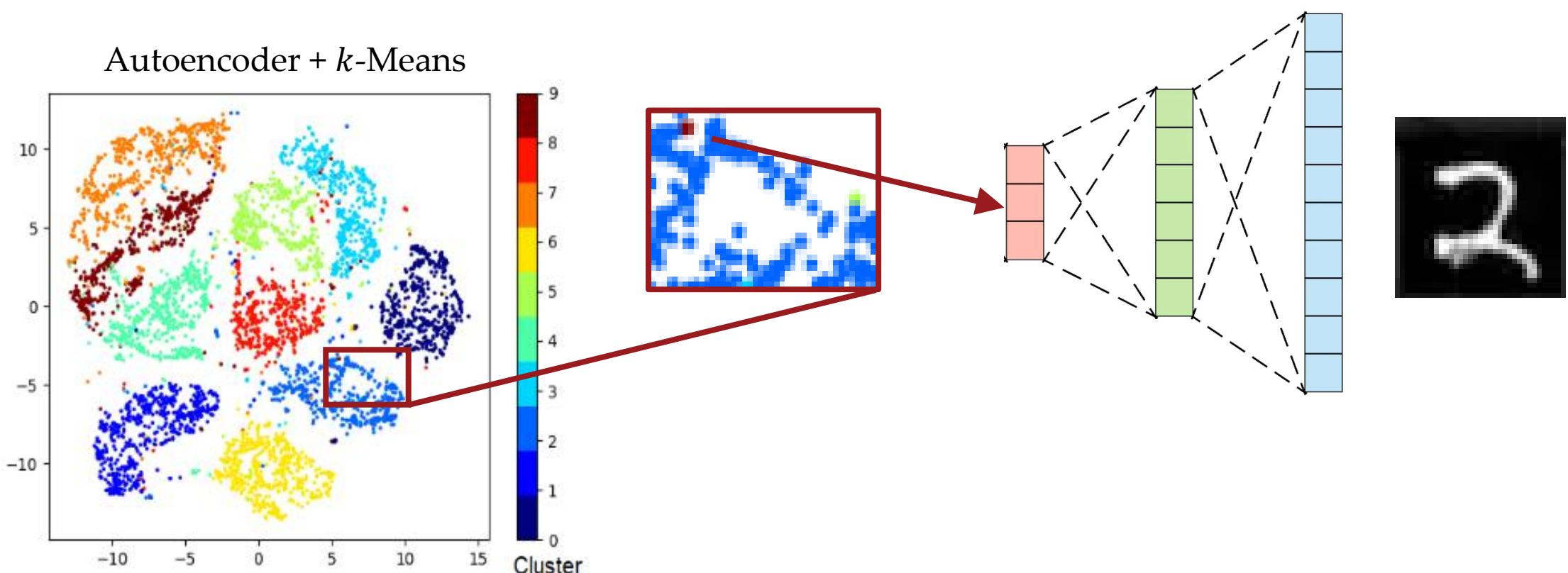
# Applications of autoencoders

- Generate new data??



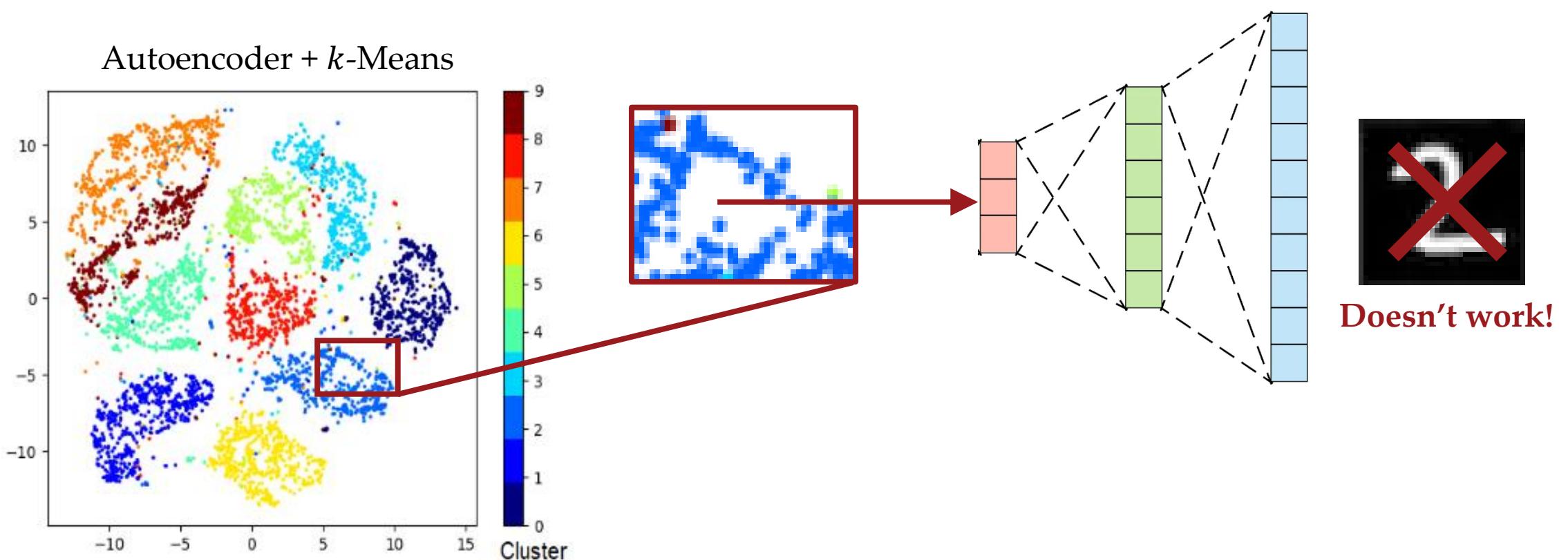
# Applications of autoencoders

- Generate new data??



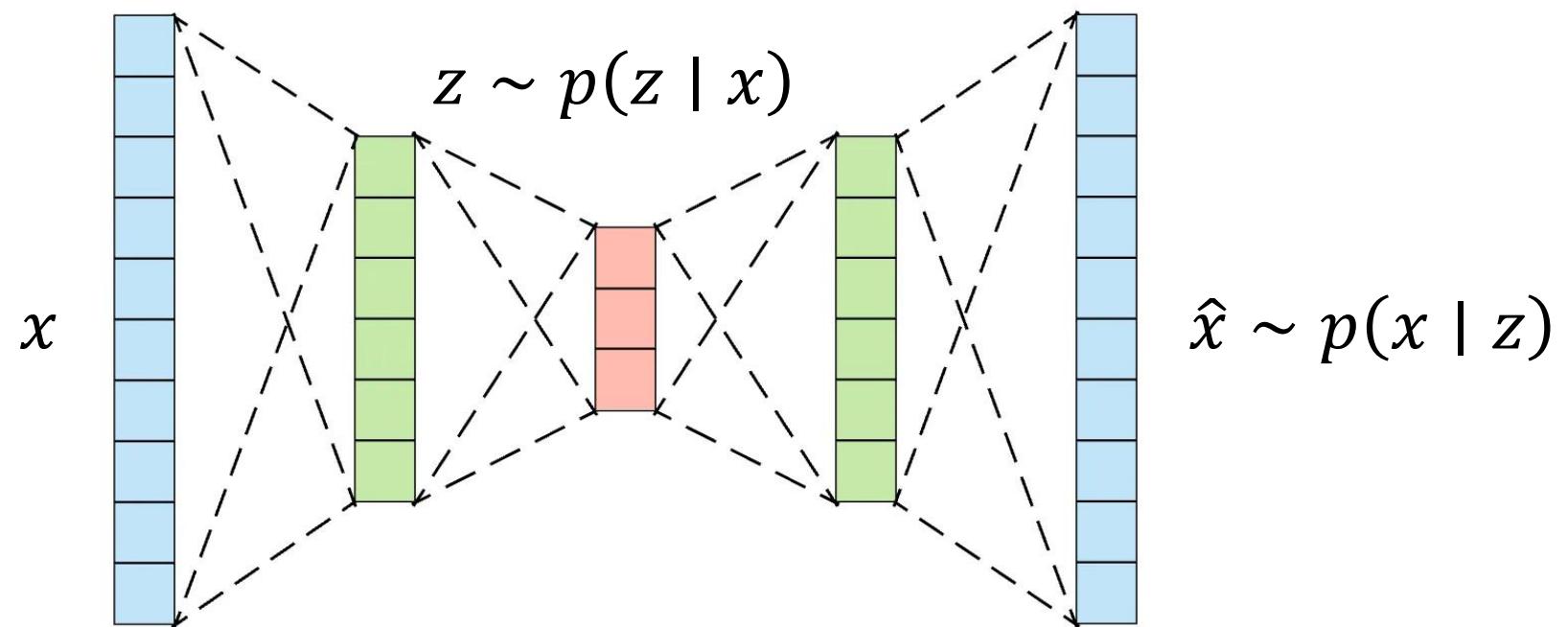
# Applications of autoencoders

- Generate new data??



# Variational autoencoders (VAE)

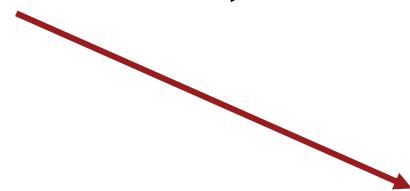
Same as autoencoders, but everything is a distribution now.



# Variational autoencoders (VAE)

How does a neural network output a distribution?

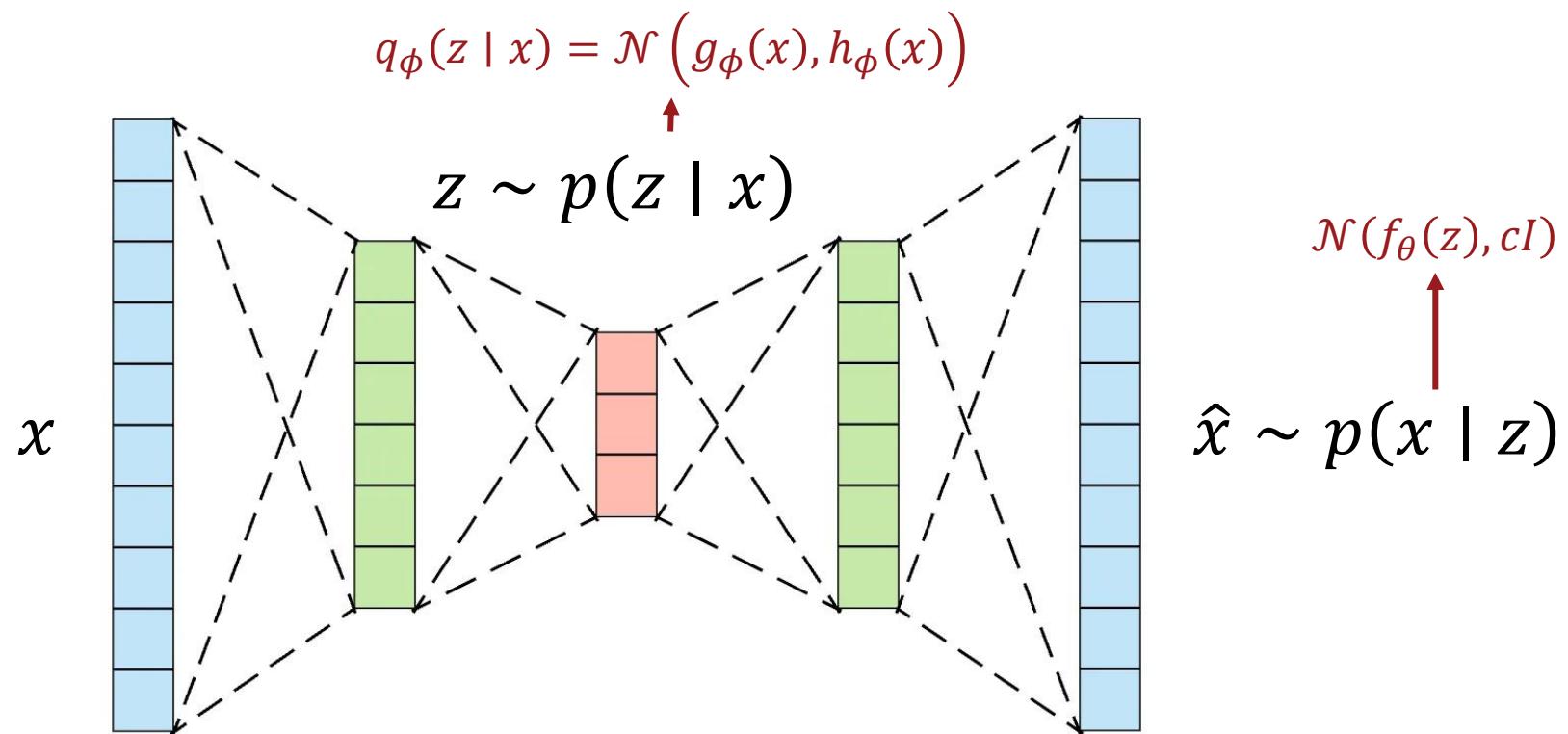
We assume a parameterized distribution, and the network outputs the parameters.



Most commonly a Gaussian distribution. Then, the network outputs the mean vector and the covariance matrix.

# Variational autoencoders (VAE)

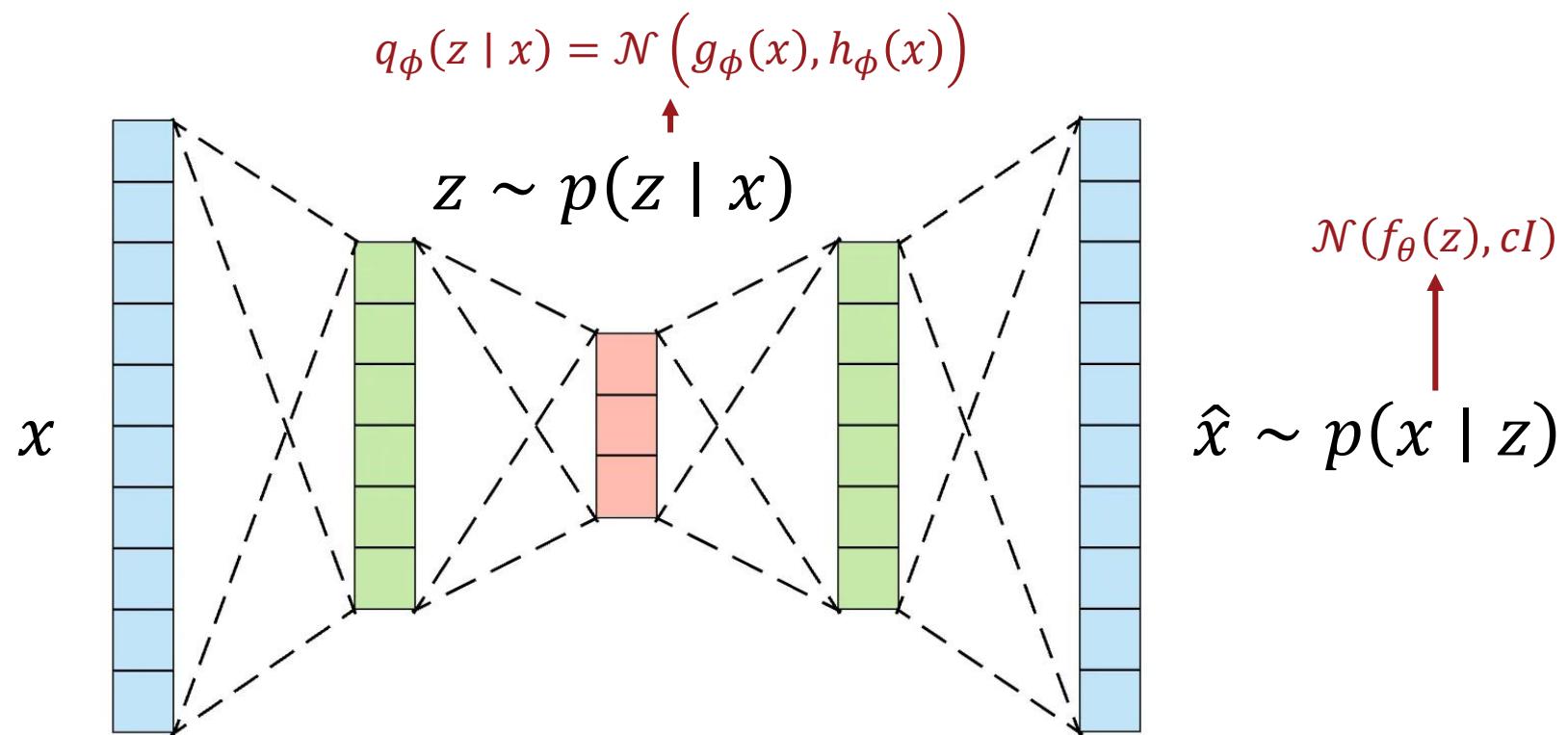
Same as autoencoders, but everything is a distribution now.



# Variational autoencoders (VAE)

Just minimize  $(x - \hat{x})^2$ ?

No.



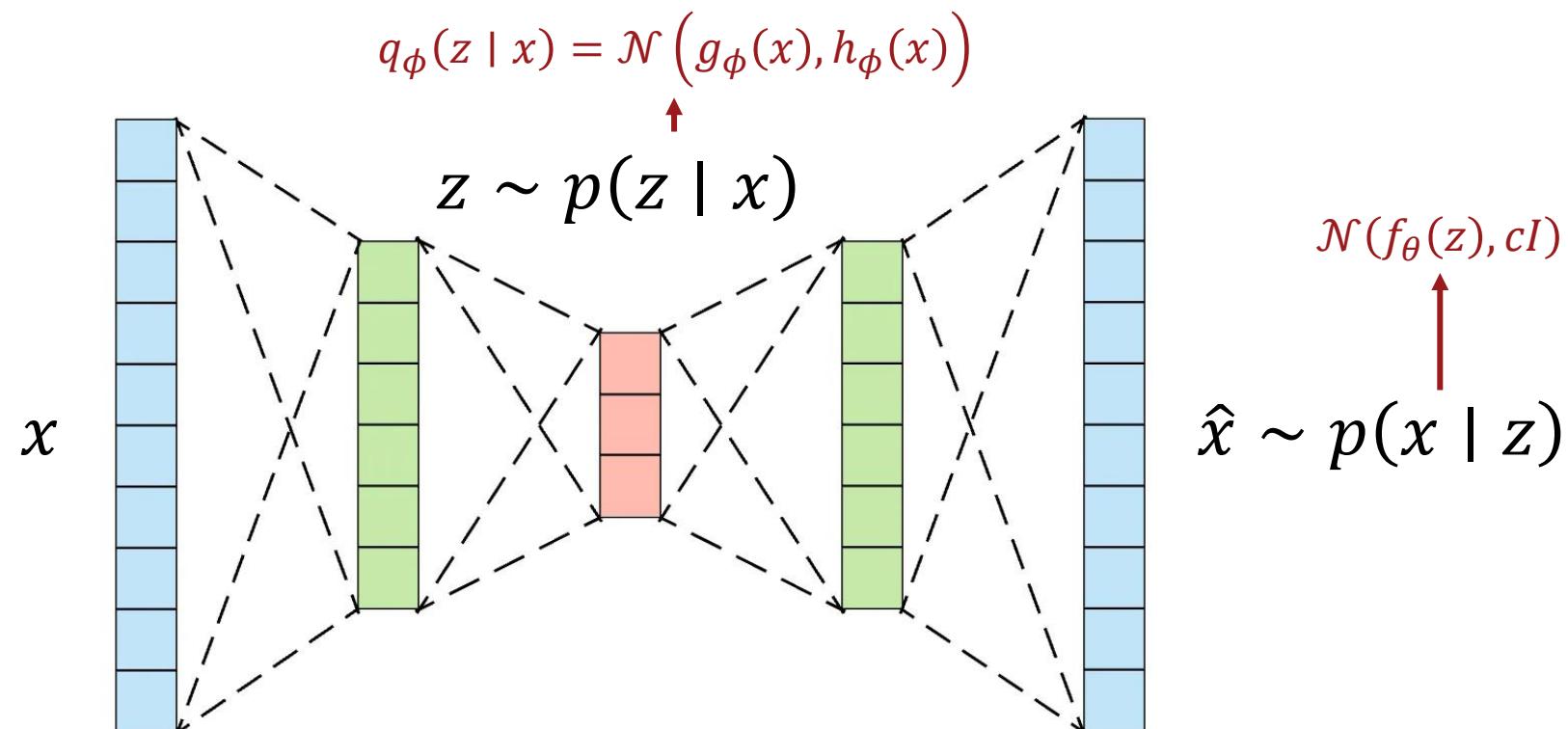
# Variational autoencoders (VAE)

Just minimize  $(x - \hat{x})^2$ ?

No.

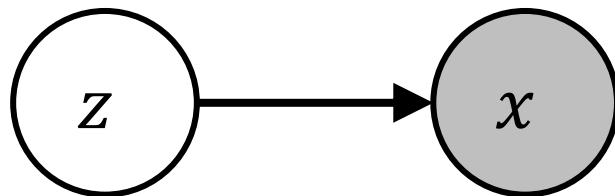
The network would learn Gaussians for  $z$  that are far from each other.

We need to regularize.



# Variational autoencoders (VAE)

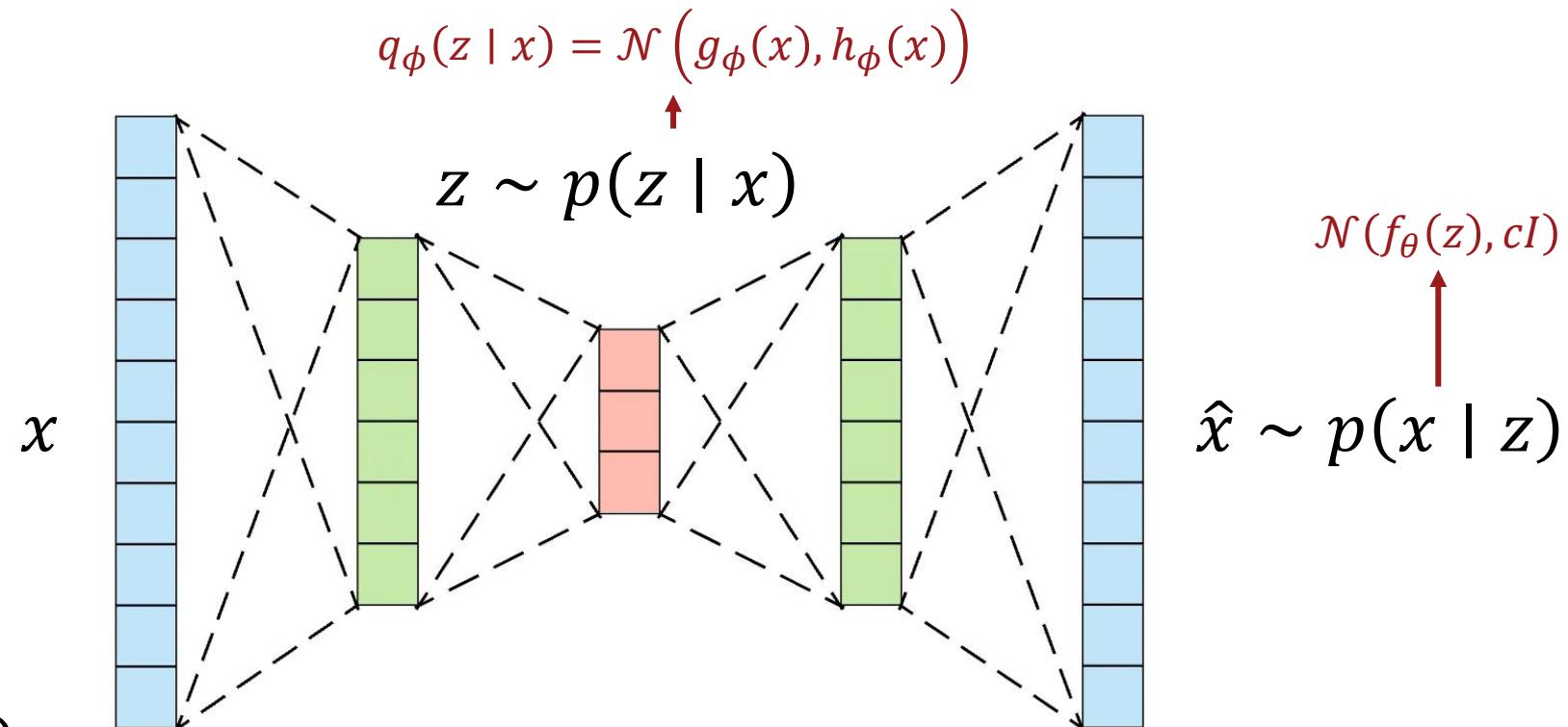
## Graphical model



$x$  is observed.

We want to maximize the likelihood of the observed data:  $p(x)$ .

But we can't compute  $p(x)$ .



# Variational autoencoders (VAE)

$$p(x)$$

Distribution of raw data.

MNIST example:

All samples from  $p(x)$  is a drawing of a 0-9 digit. For all such drawings  $p(x) > 0$ .

We don't have this.

# Variational autoencoders (VAE)

$p(x)$

Distribution of raw data.

MNIST example:

All samples from  $p(x)$  is a drawing of a 0-9 digit. For all such drawings  $p(x) > 0$ .

We don't have this.

$p(z)$

Distribution of the latent code.

We can assume anything.

This is what makes VAEs work!

We will assume  $\mathcal{N}(0, I)$  so that the network will not cheat by learning Gaussians that are far.

# Variational autoencoders (VAE)

Our goal is to maximize  $p(x)$ , or equivalently  $\log p(x)$ .

We will use **variational inference**.

# Variational inference

$$\begin{aligned} D_{KL}(q_\phi(z|x) \| p(z|x)) &= - \int q_\phi(z|x) \log \left( \frac{p(z|x)}{q_\phi(z|x)} \right) dz \geq 0 \\ &\quad - \int q_\phi(z|x) \log \left( \frac{p_\theta(x|z)p(z)}{q_\phi(z|x)p(x)} \right) dz \geq 0 \\ &\quad - \int q_\phi(z|x) \left[ \log \left( \frac{p_\theta(x|z)p(z)}{q_\phi(z|x)} \right) - \log p(x) \right] dz \geq 0 \\ \log p(x) \int q_\phi(z|x) dz - \int q_\phi(z|x) \log \left( \frac{p_\theta(x|z)p(z)}{q_\phi(z|x)} \right) dz &\geq 0 \end{aligned}$$

# Variational inference

$$\log p(x) \geq \int q_{\phi}(z | x) \log \left( \frac{p_{\theta}(x | z)p(z)}{q_{\phi}(z | x)} \right) dz$$

$$\log p(x) \geq \int q_{\phi}(z | x) \left[ \log p_{\theta}(x | z) + \log \left( \frac{p(z)}{q_{\phi}(z | x)} \right) \right] dz$$

$$\log p(x) \geq \mathbb{E}_{z \sim q_{\phi}(\cdot | x)} [\log p_{\theta}(x | z)] + \int q_{\phi}(z | x) \log \left( \frac{p(z)}{q_{\phi}(z | x)} \right) dz$$

$$\log p(x) \geq \mathbb{E}_{z \sim q_{\phi}(\cdot | x)} [\log p_{\theta}(x | z)] + D_{KL}(q_{\phi}(z | x) \| p(z))$$

# Variational inference

$$\log p(x) \geq \underbrace{\mathbb{E}_{z \sim q_\phi(\cdot|x)}[\log p_\theta(x|z)] + D_{KL}(q_\phi(z|x) \| p(z))}_{\text{Evidence Lower Bound (ELBO)}}$$

# Variational inference

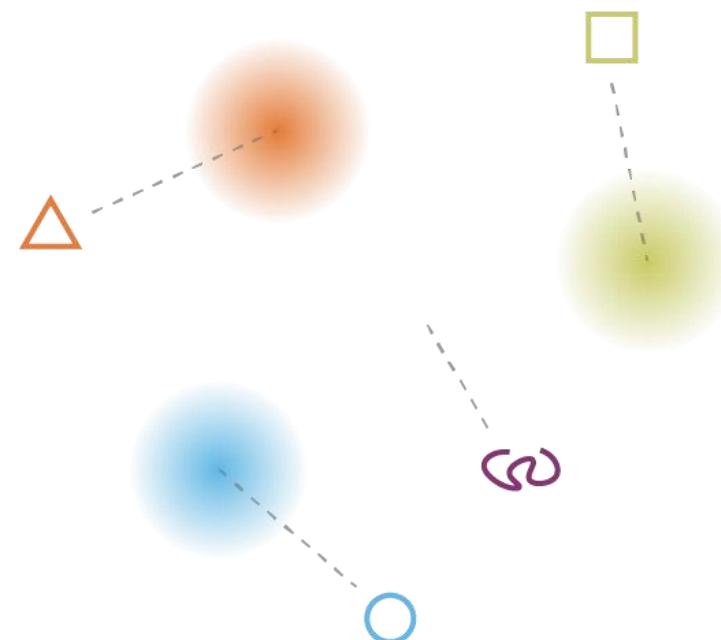
$$\log p(x) \geq \underbrace{\mathbb{E}_{z \sim q_\phi(\cdot|x)} [\log p_\theta(x|z)] + D_{KL}(q_\phi(z|x) \| p(z))}_{\text{Evidence Lower Bound (ELBO)}}$$

Practical version:

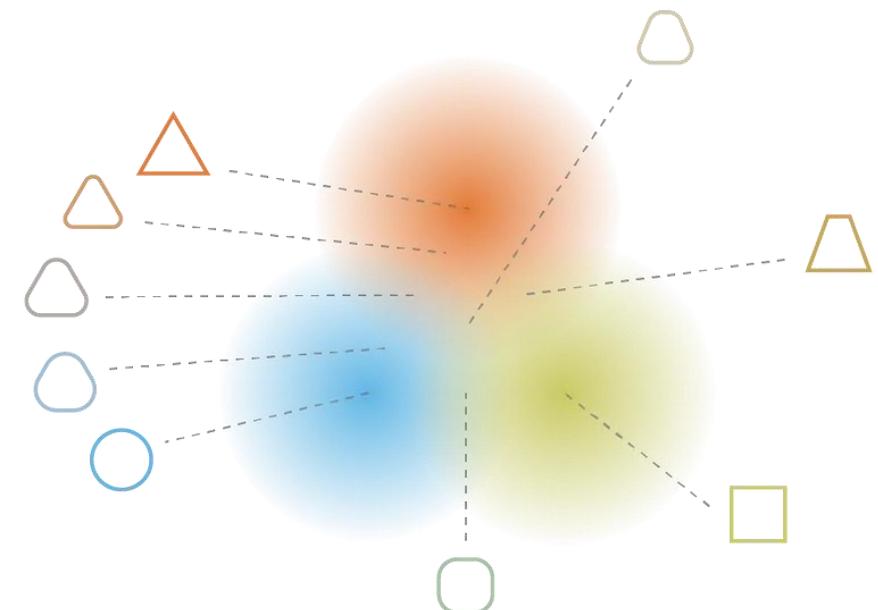
$$\log p(x) \geq \mathbb{E}_{z \sim q_\phi(\cdot|x)} [\log p_\theta(x|z) + \log p(z) - \log q_\phi(z|x)]$$

# Comparison

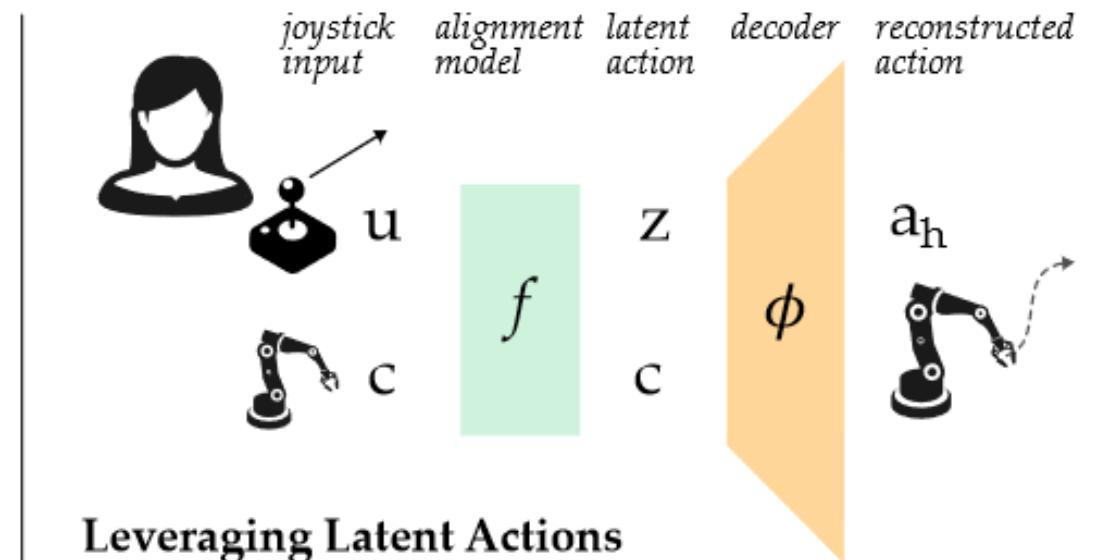
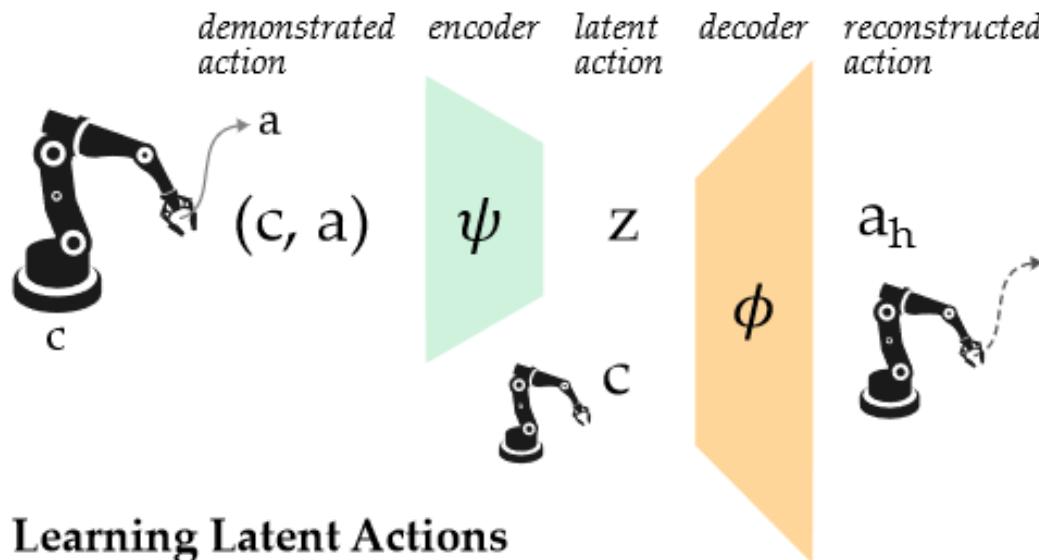
Autoencoder, or  
VAE without regularization



VAE with regularization

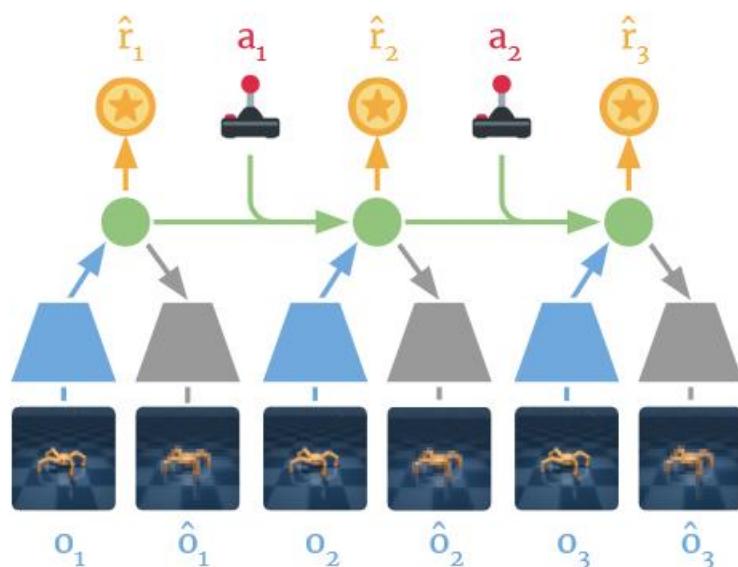


# Variational autoencoders in robotics

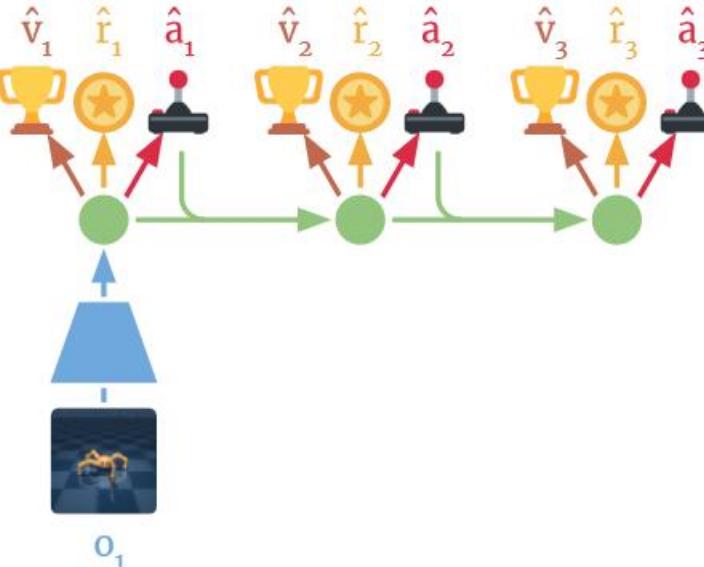


More on this in a few weeks!

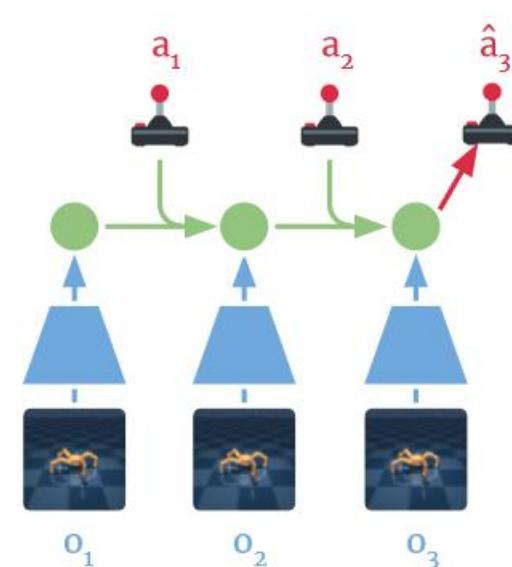
# Variational autoencoders in robotics



(a) Learn dynamics from experience



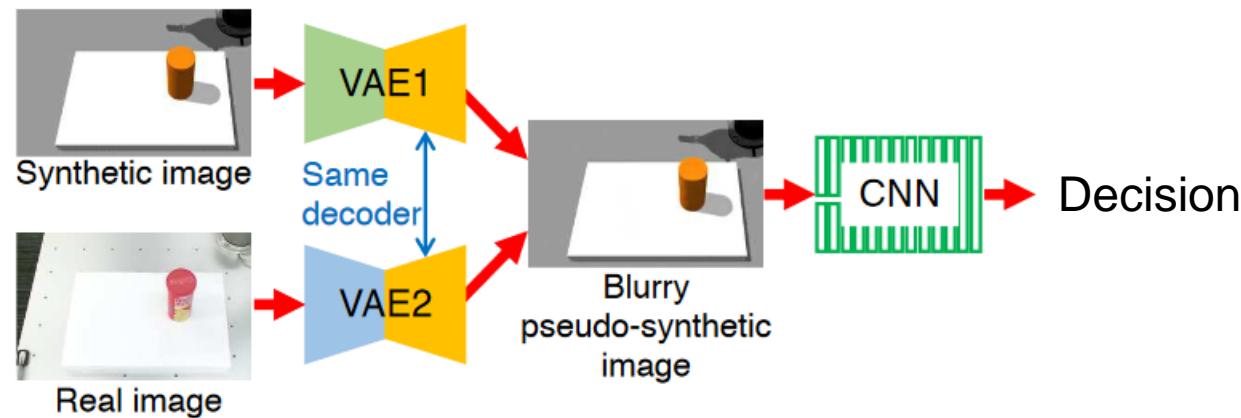
(b) Learn behavior in imagination



(c) Act in the environment

More on this in a few weeks!

# Variational autoencoders in robotics



More on this in a few weeks!

# Today

- Basics of computer vision for robotics
- Representation learning

# Next time...

- Presentations on representation learning
- Reinforcement learning