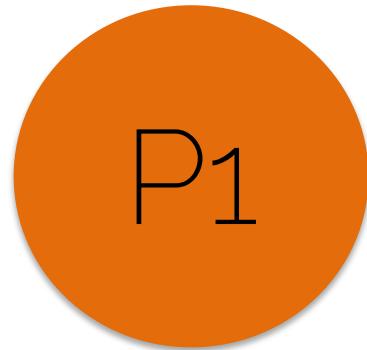


# Lecture 11 Recap

# Transfer Learning

Distribution

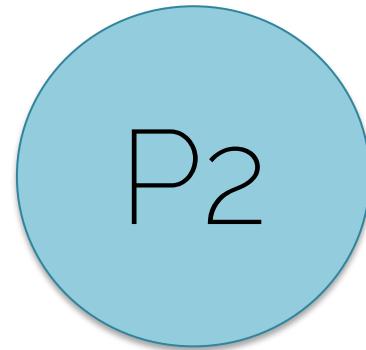


Large dataset



Use what has been  
learned for another  
setting

Distribution

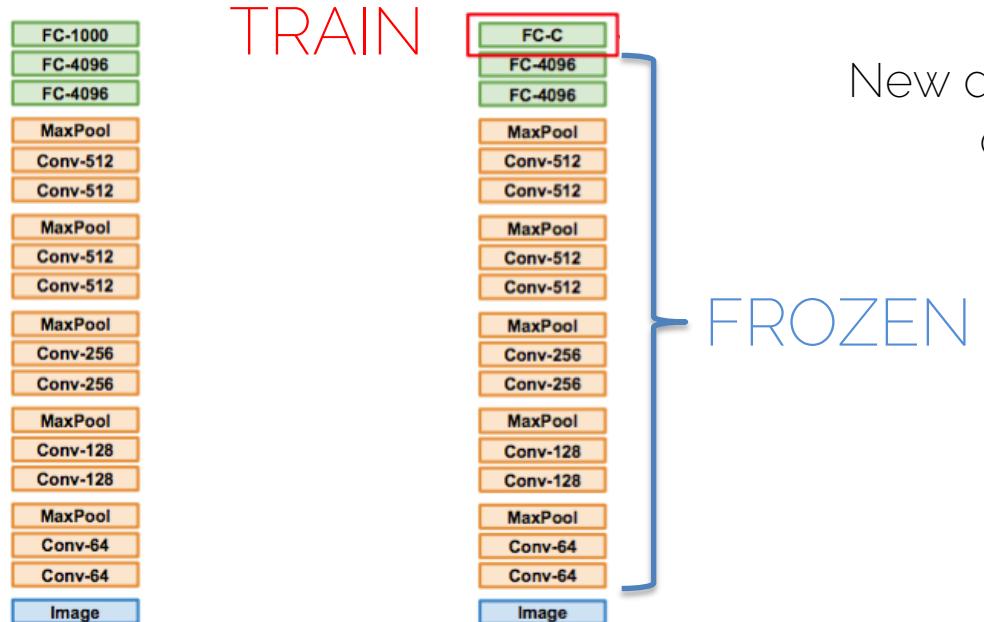


Small dataset



# Transfer Learning

Trained on ImageNet



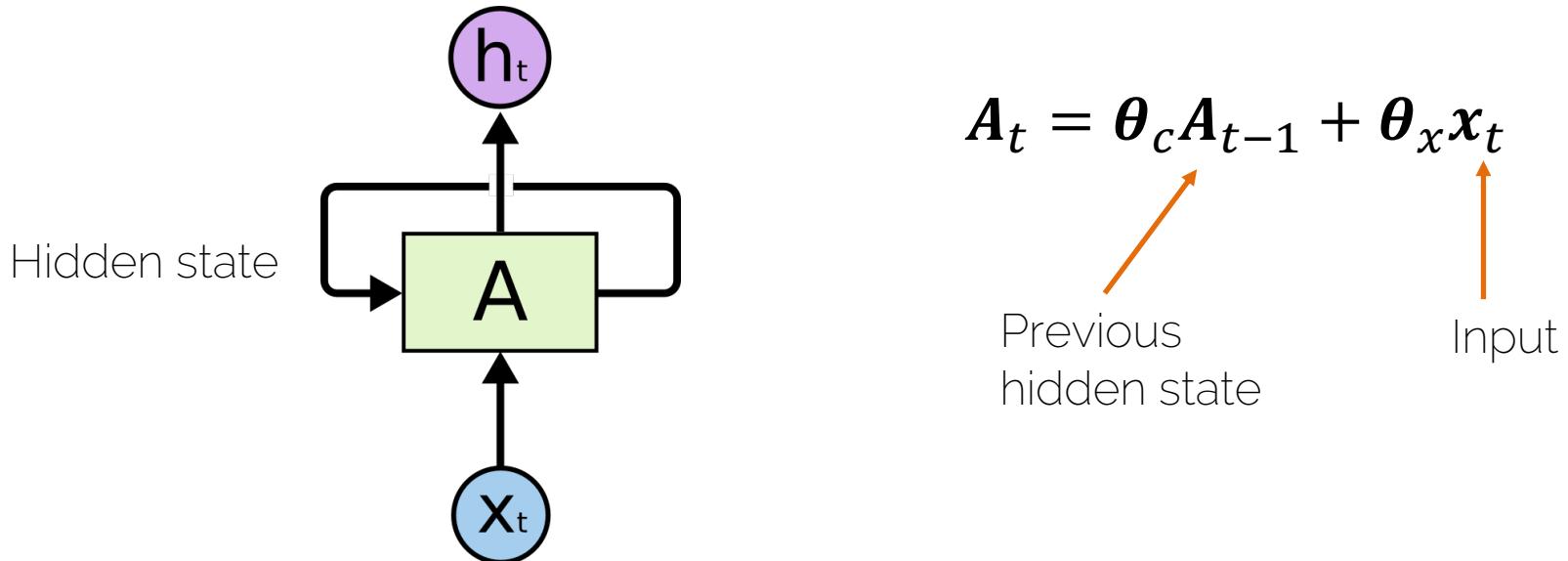
Source : [http://cs231n.stanford.edu/slides/2016/winter1516\\_lecture11.pdf](http://cs231n.stanford.edu/slides/2016/winter1516_lecture11.pdf)

[Donahue et al., ICML'14] DeCAF,

[Razavian et al., CVPRW'14] CNN Features off-the-shelf

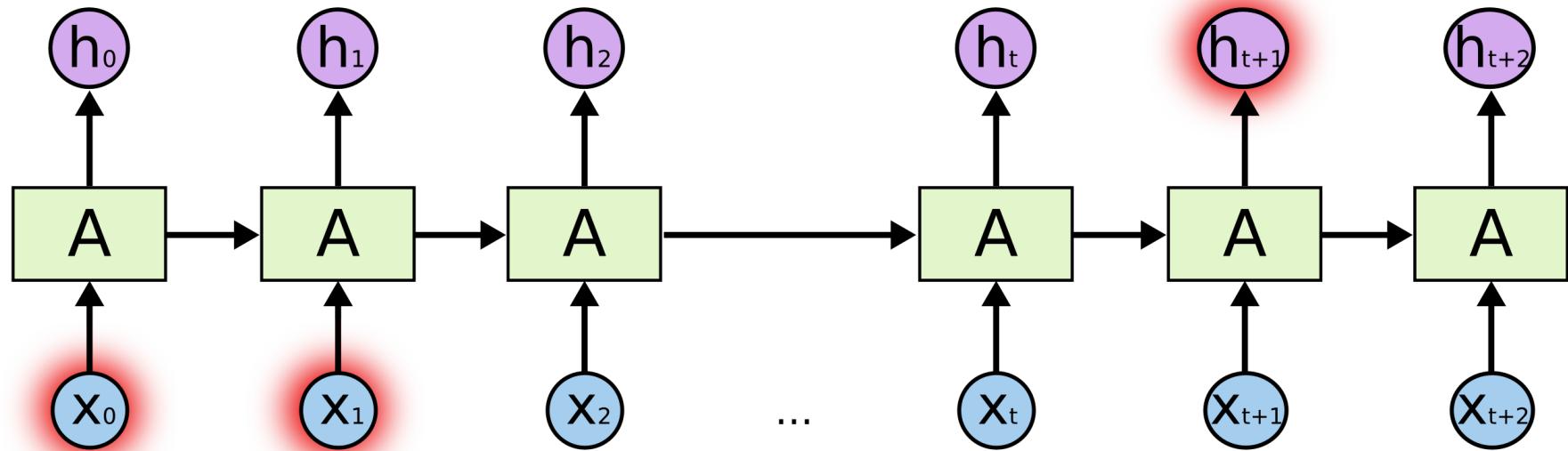
# Basic Structure of RNN

- We want to have notion of "time" or "sequence"



Source: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>

# Long-Term Dependencies

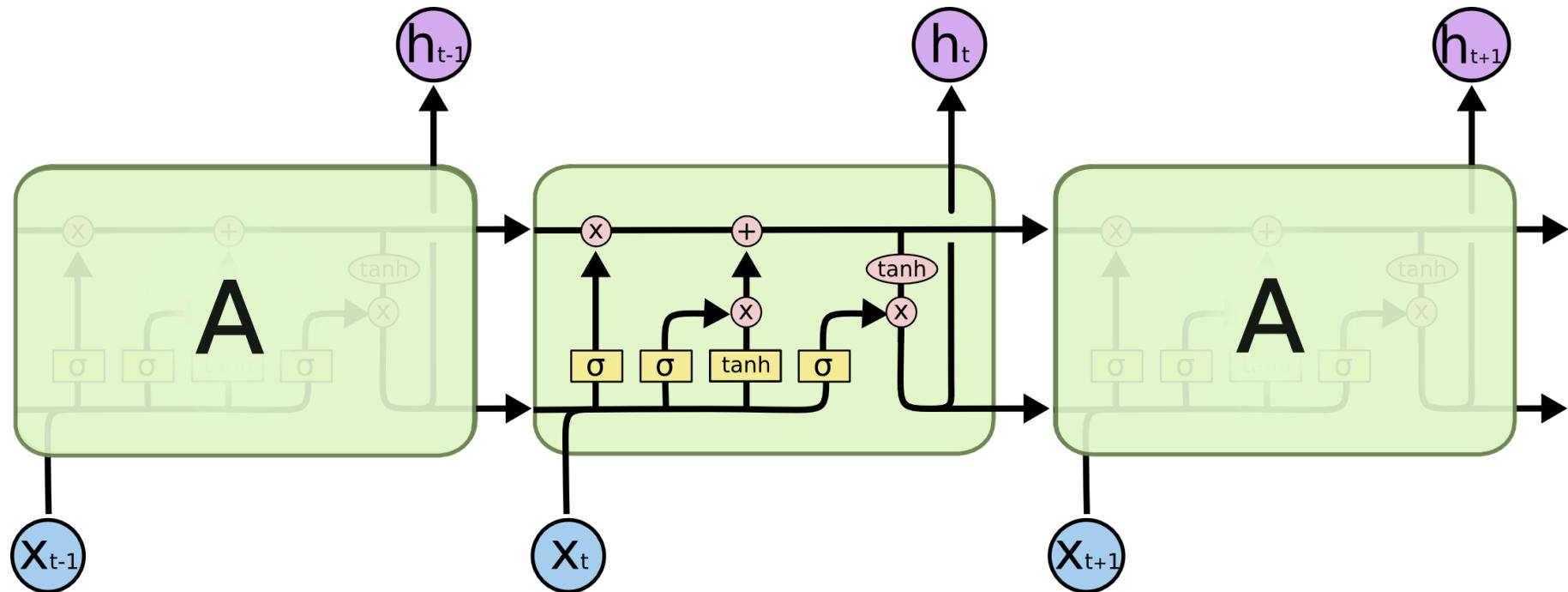


I moved to Germany ...

so I speak German fluently.

Source: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>

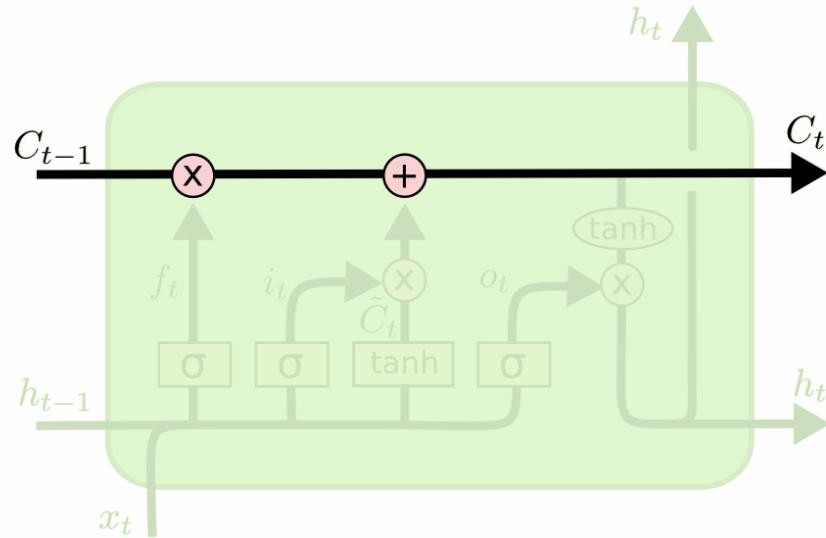
# Long-Short Term Memory Units(LSTM)



Source: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>

# Long-Short Term Memory Units

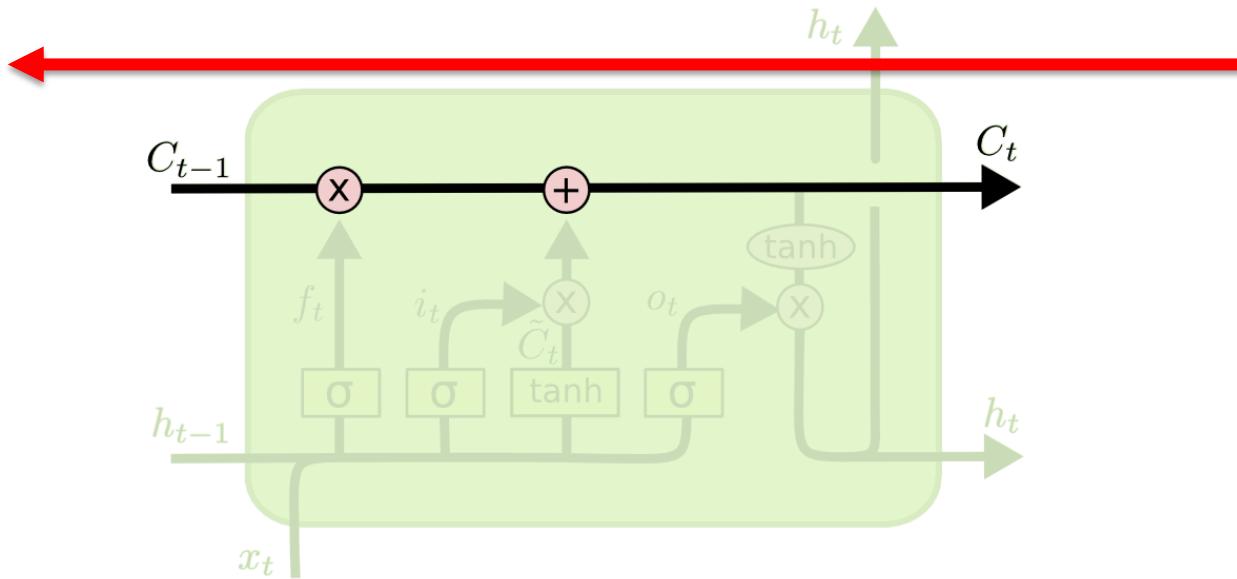
- Key ingredients
- Cell = transports the information through the unit



Source: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>

# LSTM

- Highway for the gradient to flow

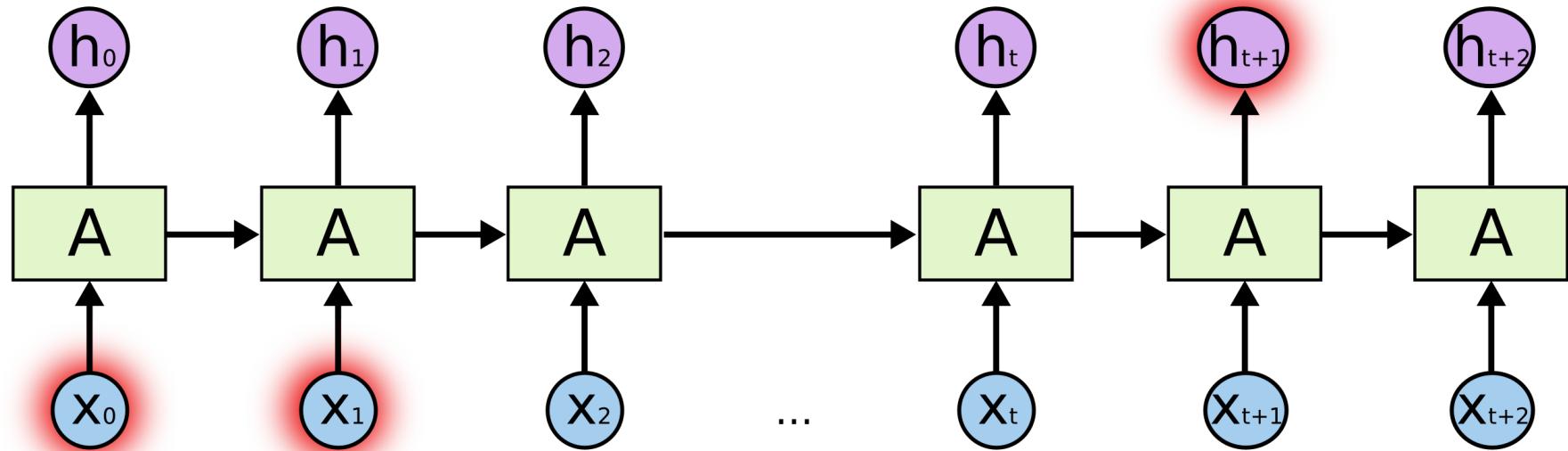


Source: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>

# Lecture 12: Advanced DL topics

# Attention

# Long-Term Dependencies

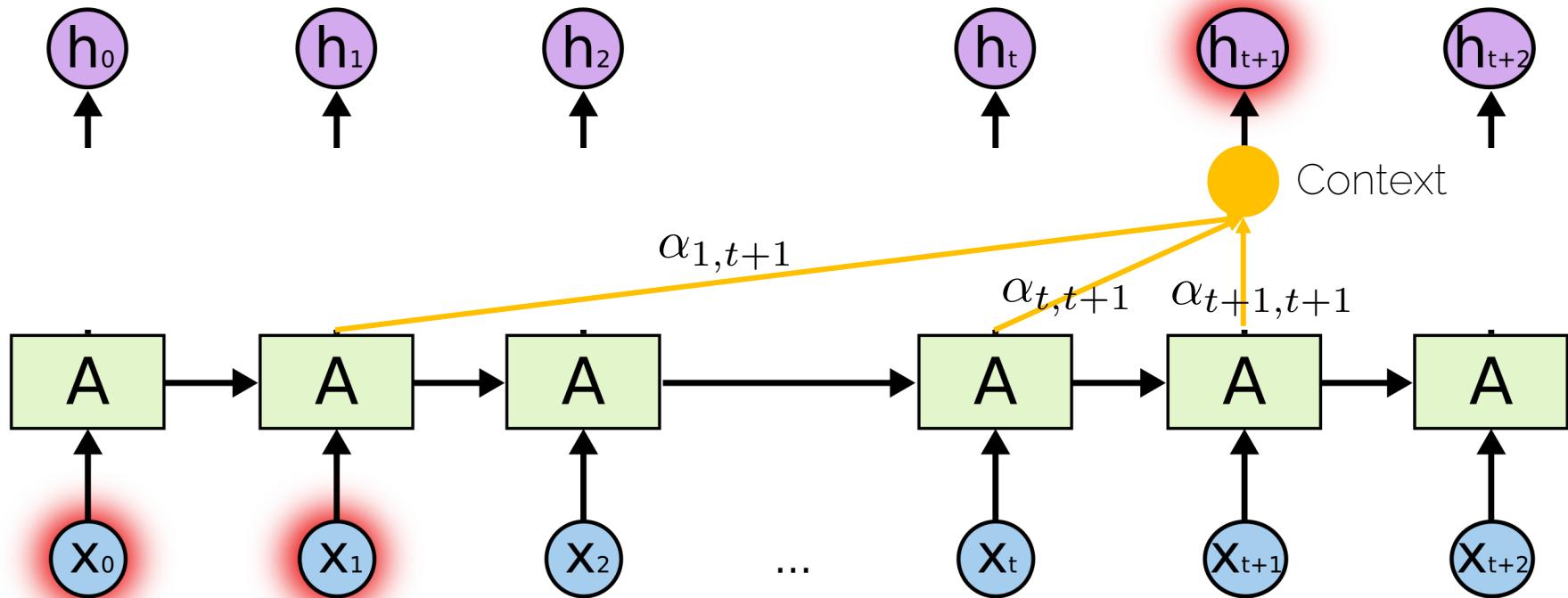


I moved to Germany ...

so I speak German fluently.

Source: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>

# Attention: intuition

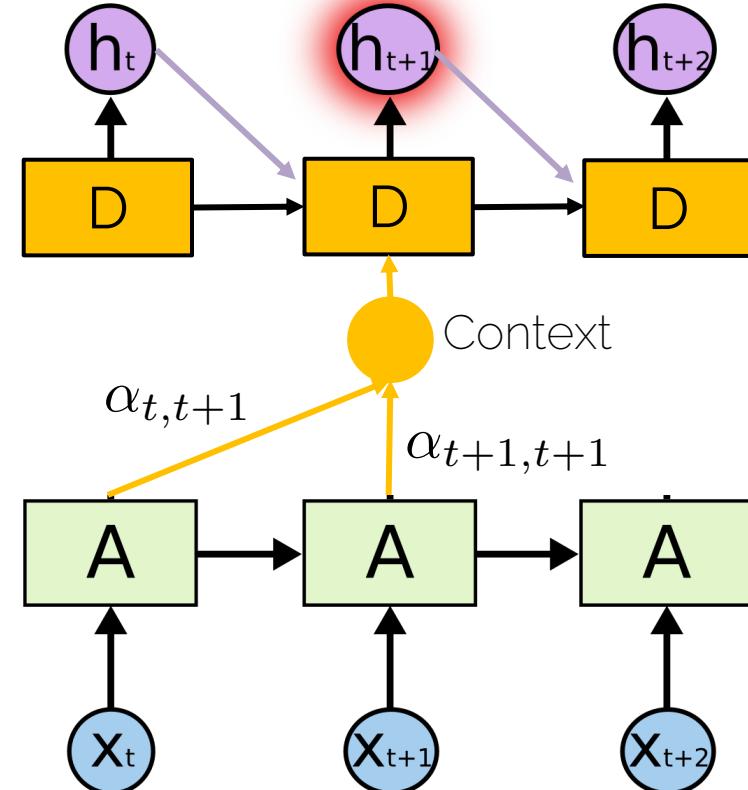


I moved to Germany ...

so I speak German fluently

# Attention: architecture

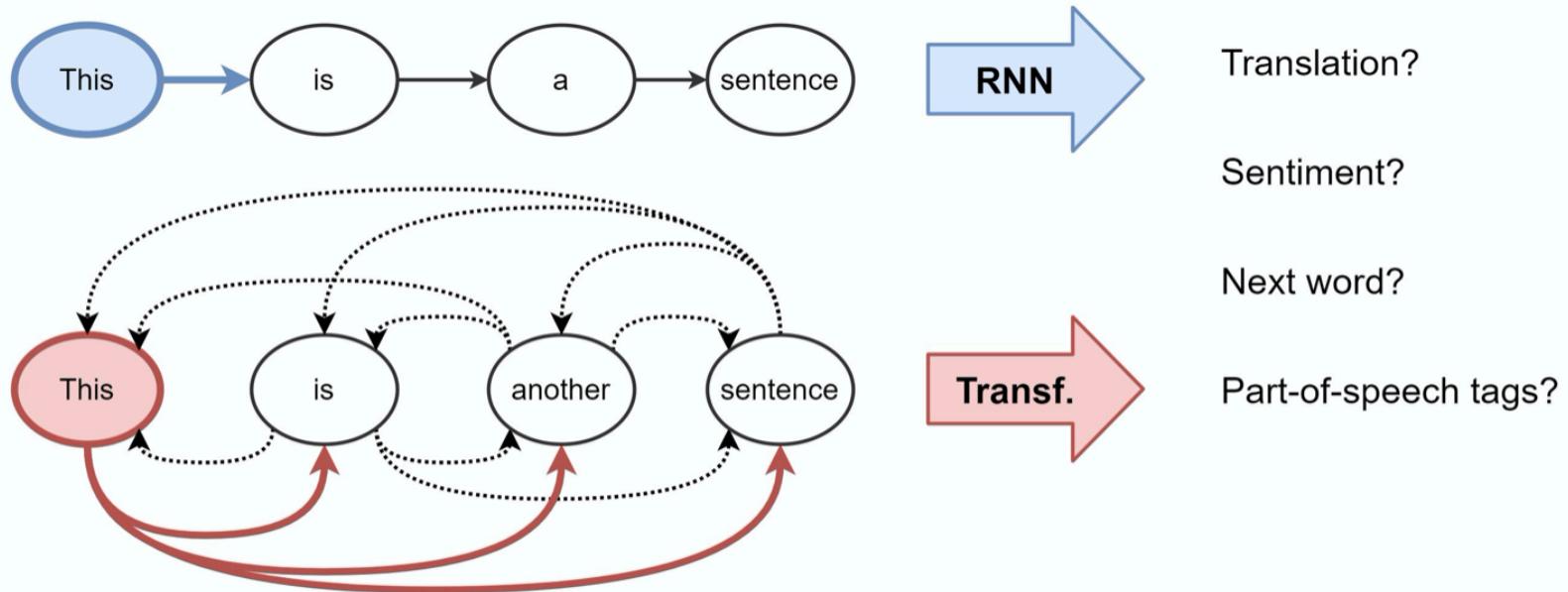
- A decoder processes the information
- Decoders take as input:
  - Previous decoder hidden state
  - Previous output
  - Attention



# Transformers

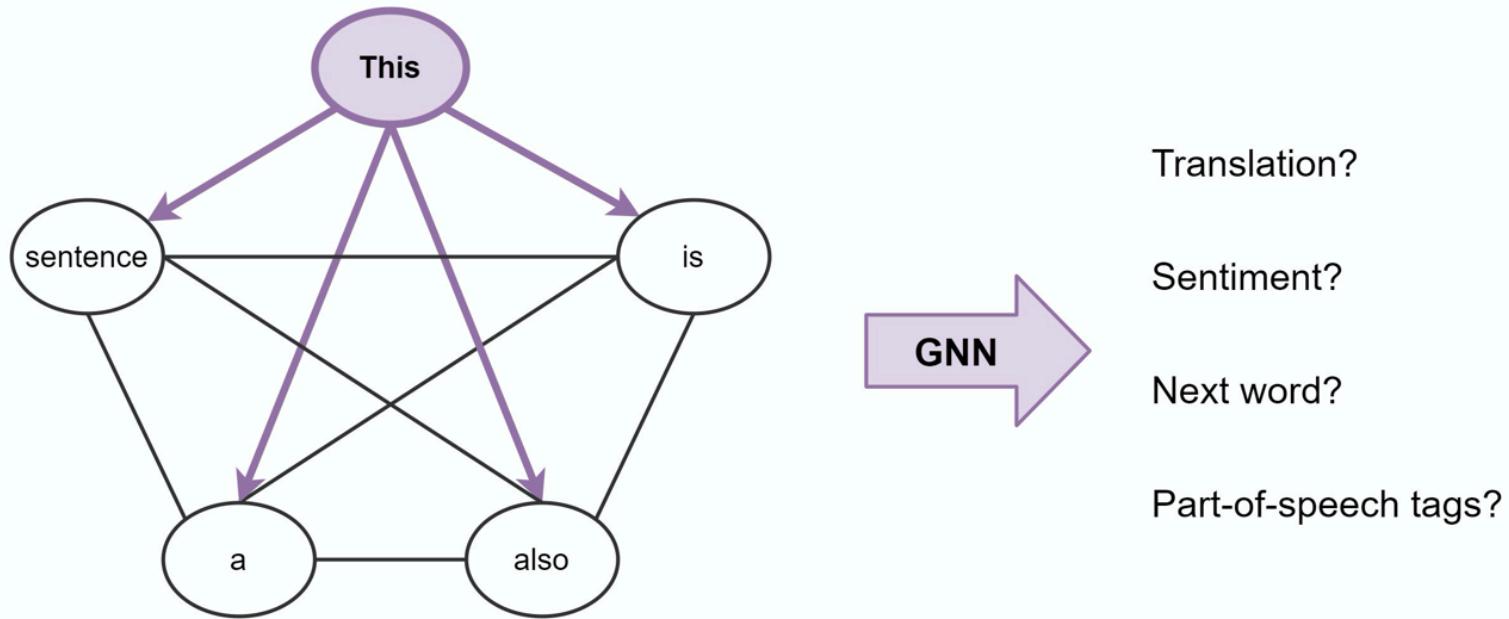
- What if we could get rid of the recurrent architecture and use only attention?
- All the memory problems of RNNs could disappear
- No RNN, no CNN, just attention!
- Current state-of-the-art in NLP!

# Transformers



# Transformers

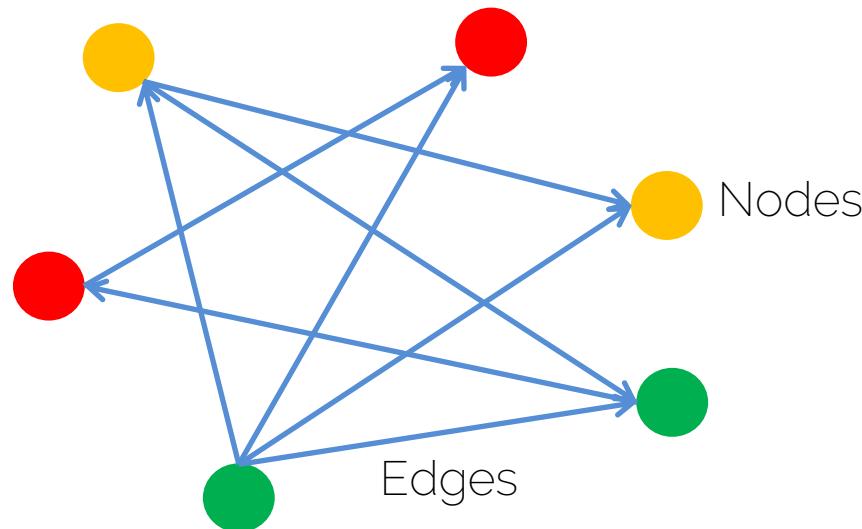
- This is very related to another important type of NN



# Graph Neural Networks

# A graph

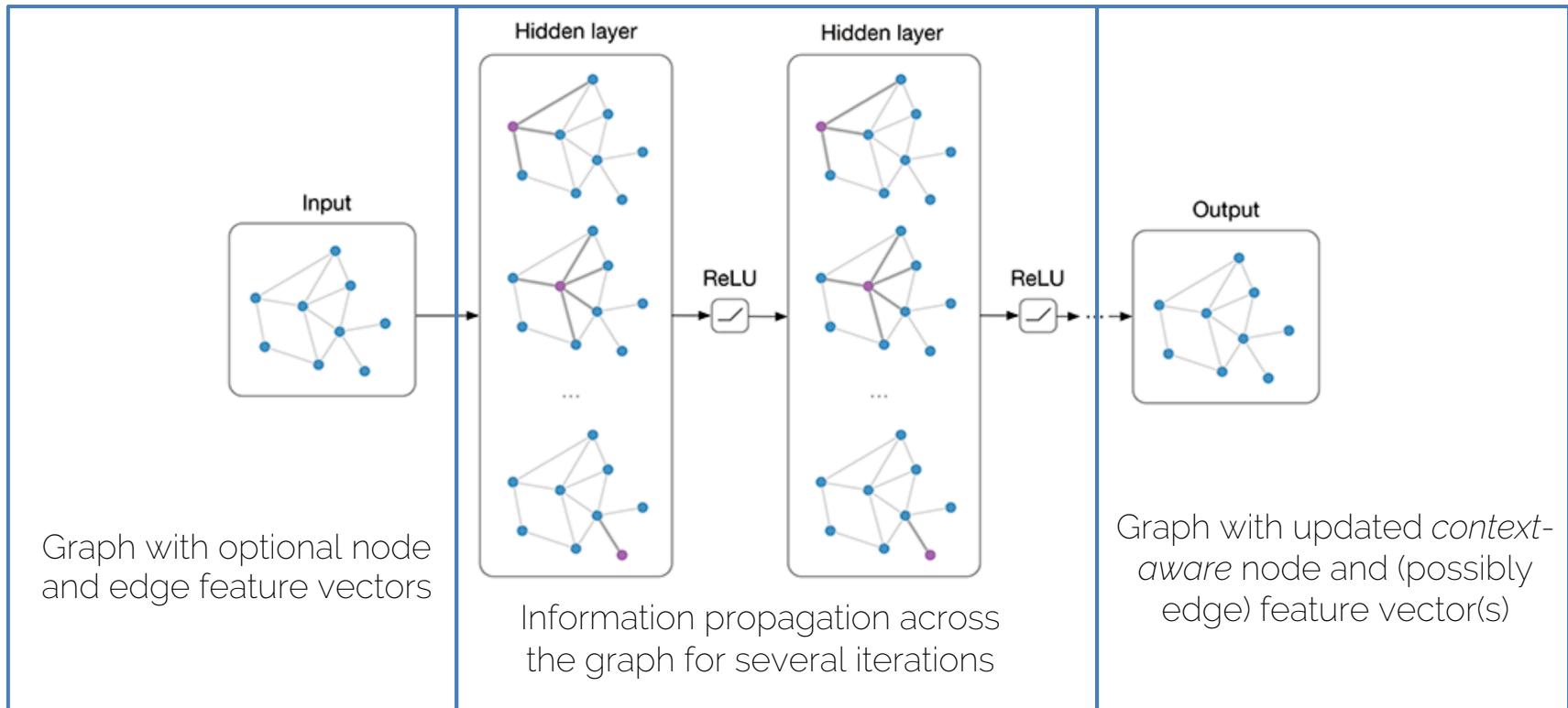
- Node: a concept
- Edge: a connection between concepts



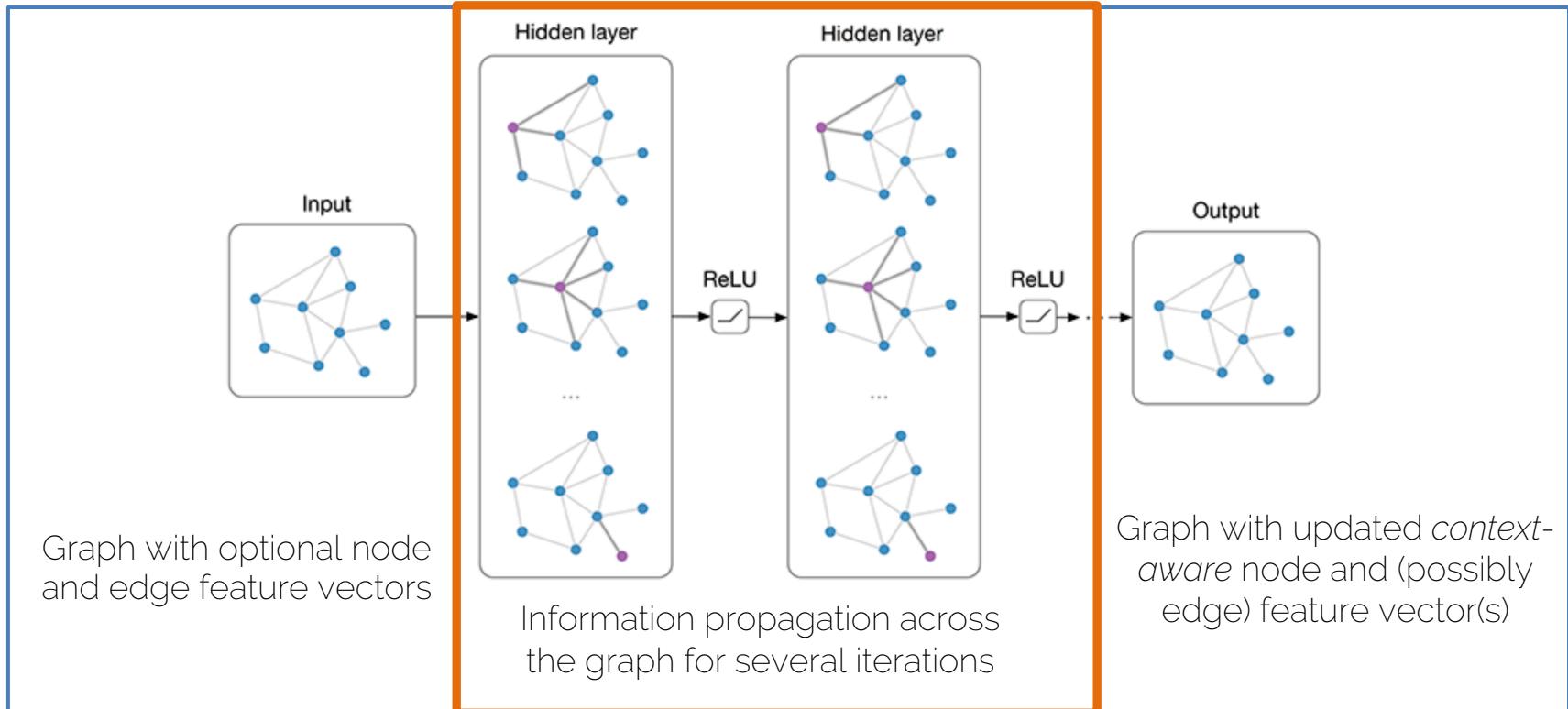
# Deep learning on graphs

- Generalizations of neural networks that can operate on graph-structured domains:
  - Scarselli et al. "The Graph Neural Network Model". IEEE Trans. Neur. Net 2009.
  - Kipf et al. "Semi-Supervised Classification with Graph Convolutional Networks. ICLR 2016.
  - Gilmer et al. "Neural Message Passing for Quantum Chemistry". ICML 2017
  - Battaglia et al. "Relational inductive biases, deep learning, and graph networks". arXiv 2018 (review paper)
- Key challenges:
  - Variable sized inputs (number of nodes and edges)
  - Need **invariance to node permutations**

# General Idea

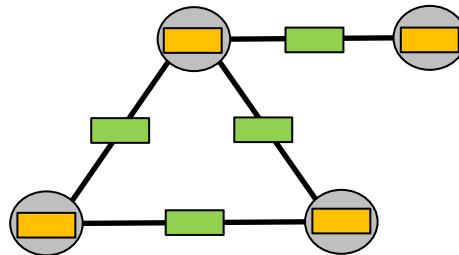


# General Idea

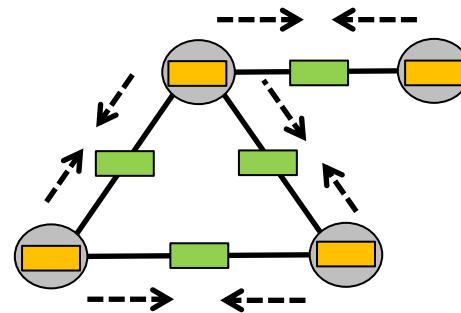


# Message Passing Networks

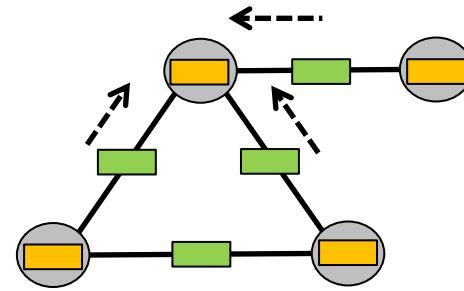
- We can divide the propagation process in two steps: 'node to edge' and 'edge to node' updates.



Initial Graph



'Node to edge' Update



'Edge to Node' Update

Node embeddings  
 Edge embeddings

# 'Node to edge' updates

- At every message passing step  $l$ , first do:

$$h_{(i,j)}^{(l)} = \mathcal{N}_e \left( [h_i^{(l-1)}, h_{(i,j)}^{(l-1)}, h_j^{(l-1)}] \right)$$


Embedding of node i in the previous message passing step

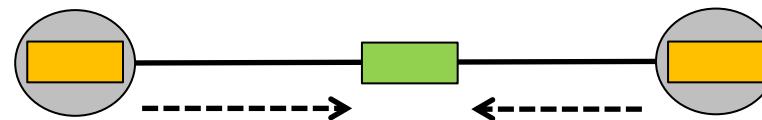
Embedding of edge (i,j) in the previous message passing step

Embedding of node j in the previous message passing step

# 'Node to edge' updates

- At every message passing step  $l$ , first do:

$$h_{(i,j)}^{(l)} = \mathcal{N}_e \left( [h_i^{(l-1)}, h_{(i,j)}^{(l-1)}, h_j^{(l-1)}] \right)$$

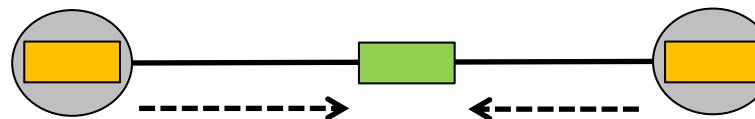


# 'Node to edge' updates

- At every message passing step  $l$ , first do:

$$h_{(i,j)}^{(l)} = \underbrace{\mathcal{N}_e}_{\text{Learnable function}} \left( [h_i^{(l-1)}, h_{(i,j)}^{(l-1)}, h_j^{(l-1)}] \right)$$

Learnable function (e.g.  
MLP) with shared  
weights across the  
entire graph



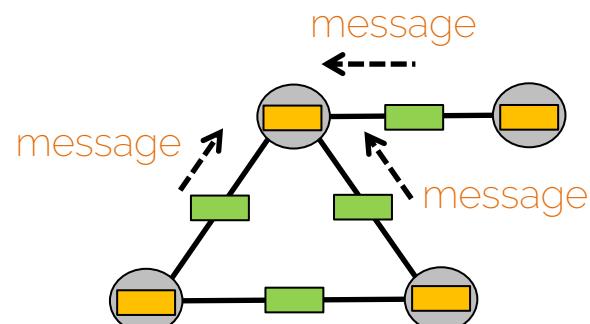
# 'Edge to node' updates

- After a round of edge updates, each edge embedding contains information about its pair of incident nodes
- Then, edge embeddings are used to update nodes:

$$m_i^{(l)} = \Phi \left( \left\{ h_{(i,j)}^{(l)} \right\}_{j \in N_i} \right)$$

Order invariant  
operation (e.g.  
sum, mean, max)

Neighbors of  
node i



# 'Edge to node' updates

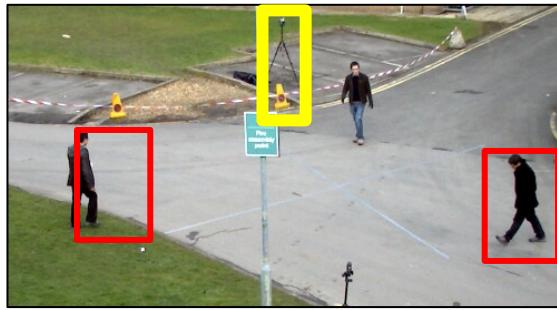
- After a round of edge updates, each edge embedding contains information about its pair of incident nodes
- Then, edge embeddings are used to update nodes:

$$m_i^{(l)} = \Phi \left( \left\{ h_{(i,j)}^{(l)} \right\}_{j \in N_i} \right)$$
$$h_i^{(l)} = \mathcal{N}_v \left( [m_i^{(l)}, h_i^{(l-1)}] \right)$$

The aggregation provides each node embedding with contextual information about its neighbors

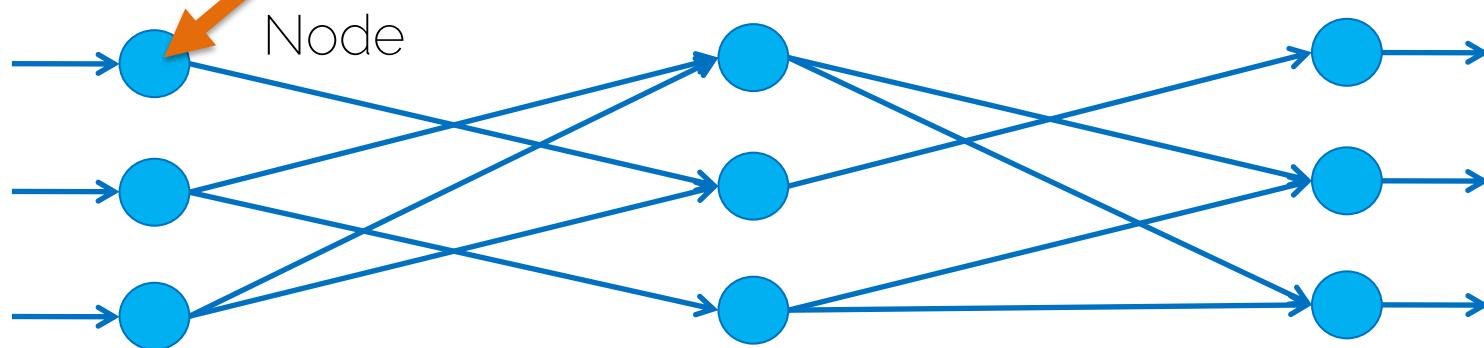
Learnable function (e.g. MLP) with shared weights across the entire graph

# Multi-object tracking with graphs



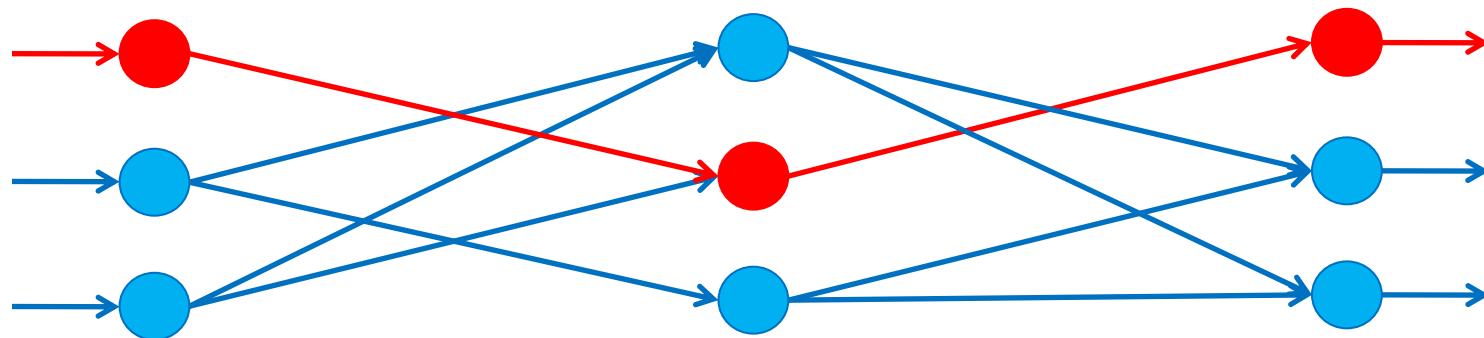
Step 1: Object detection

# Multi-object tracking with graphs



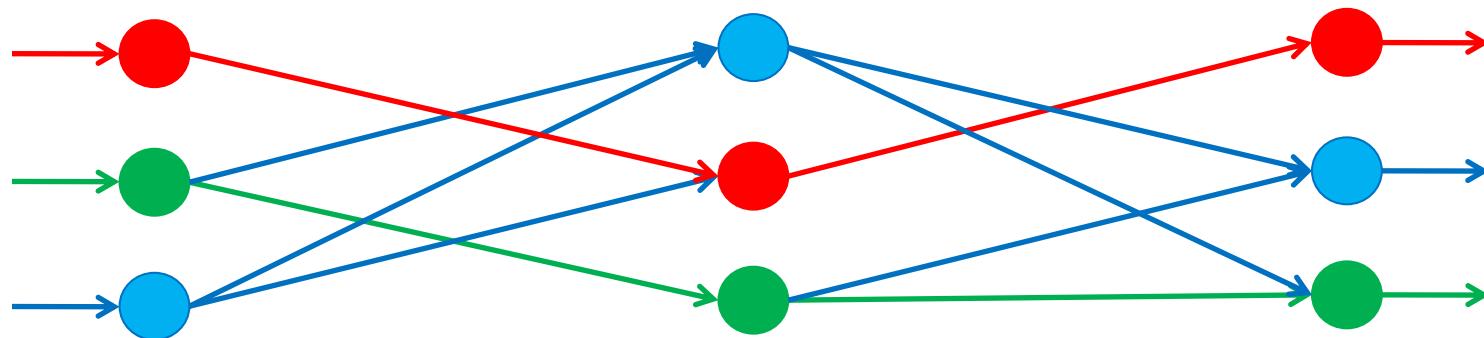
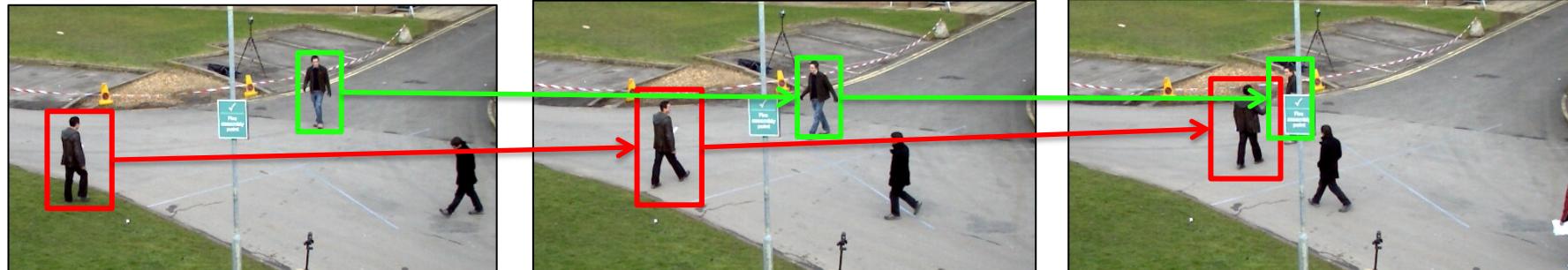
Graphical model

# Multi-object tracking with graphs



L. Leal-Taixé et al. "Everybody needs somebody: Modeling social and grouping behavior on a linear programming multiple people tracker." ICCVW/2011

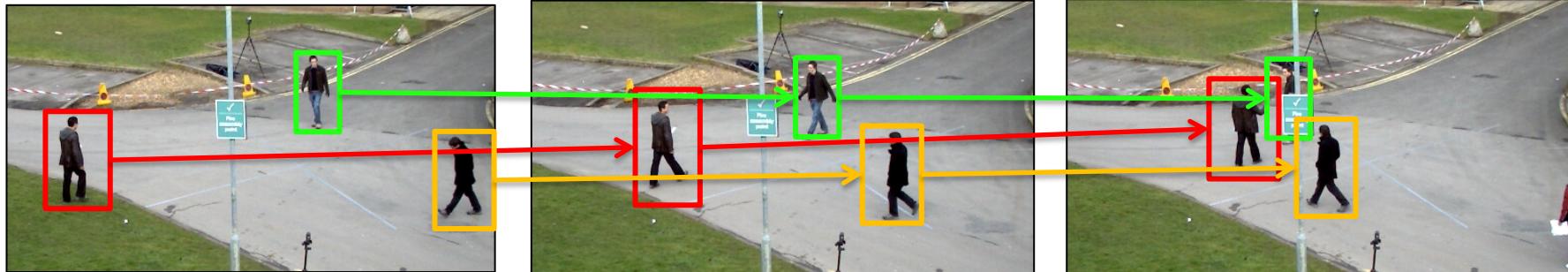
# Multi-object tracking with graphs



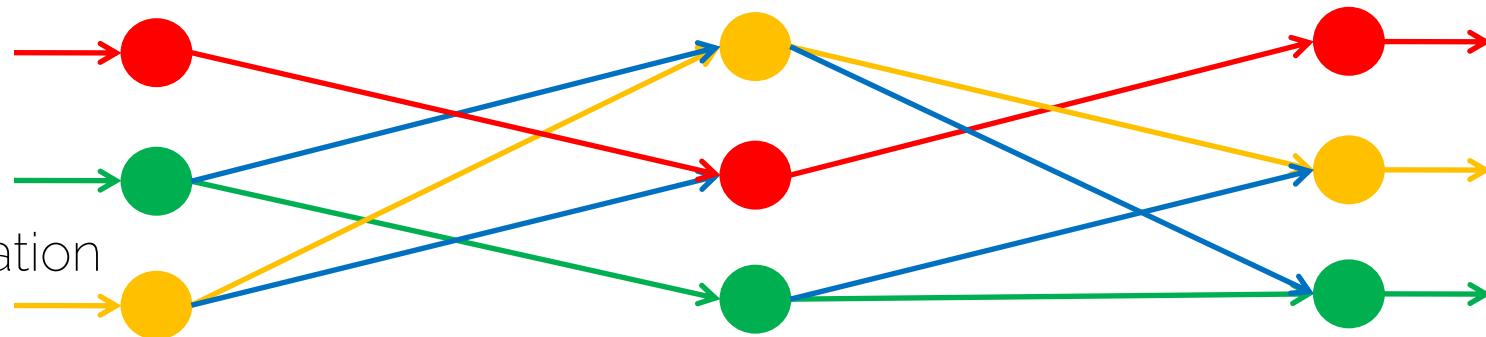
L. Leal-Taixé et al. "Everybody needs somebody: Modeling social and grouping behavior on a linear programming multiple people tracker." ICCVW/2011

# Multi-object tracking with graphs

Step 1: Object detection

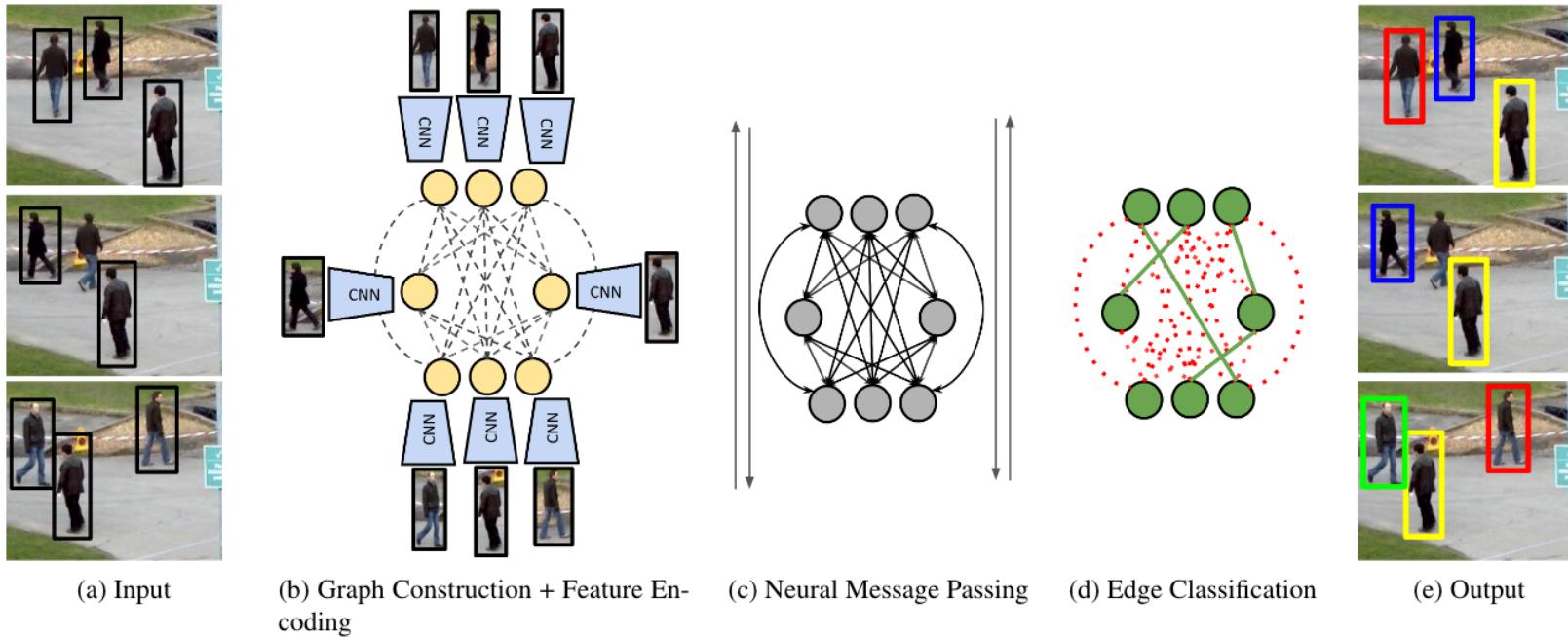


Step 2:  
Data  
association



L. Leal-Taixé et al. "Everybody needs somebody: Modeling social and grouping behavior on a linear programming multiple people tracker." ICCVW/2011

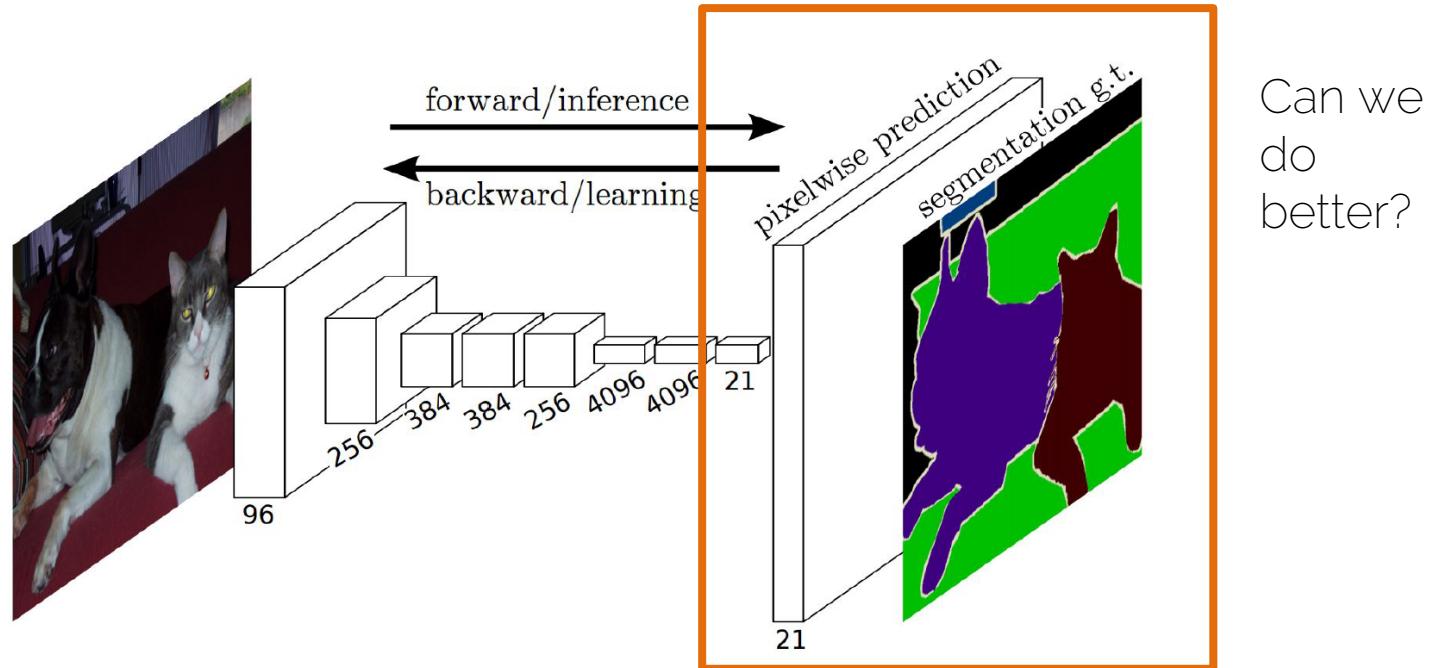
# MOT with MPN: Overview



# Generative Models

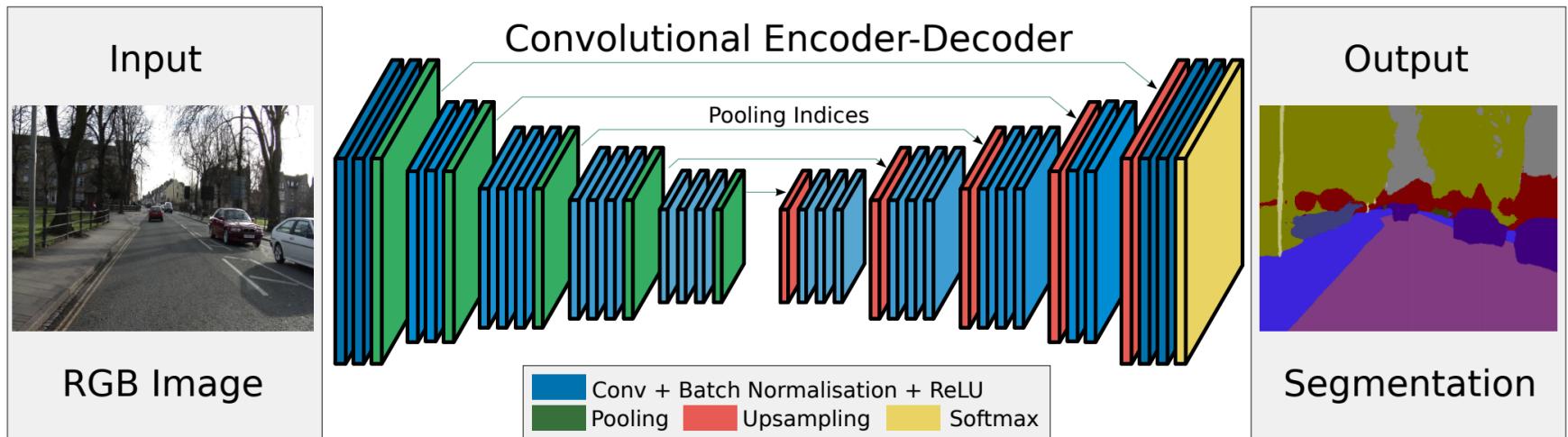
# Semantic Segmentation (FCN)

- Recall the Fully Convolutional Networks



[Long et al., CVPR'15] : Fully Convolutional Networks for Semantic Segmentation

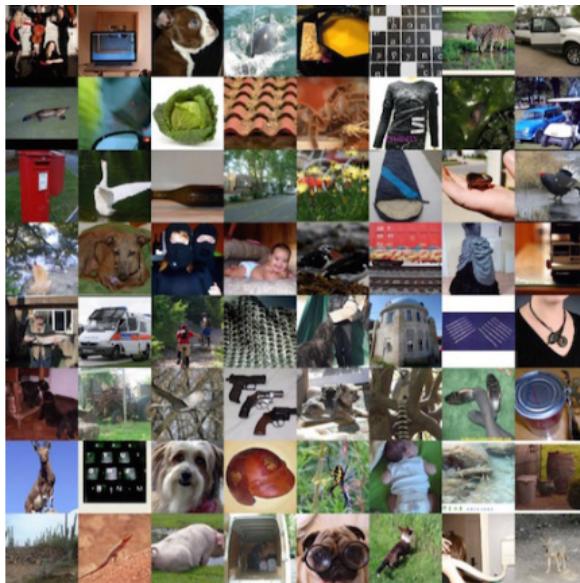
# SegNet



# Generative Models

- Given training data, how to generate new samples from the same distribution

Real Images



Generated Images



# Generative Models

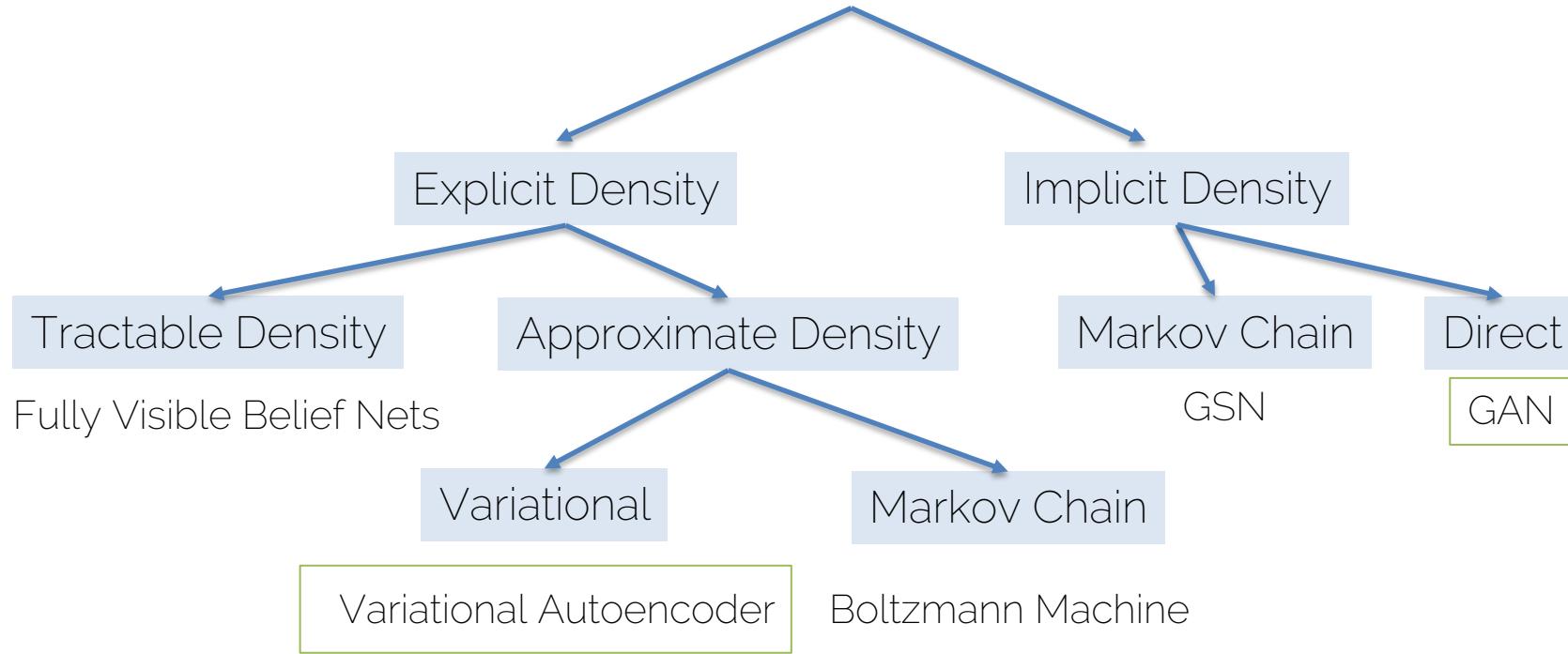
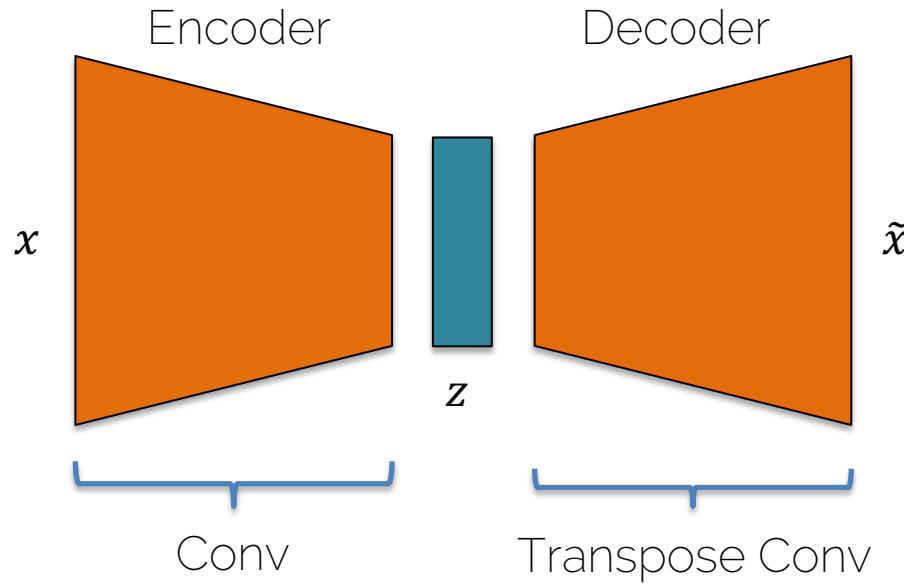


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017

# Variational Autoencoders

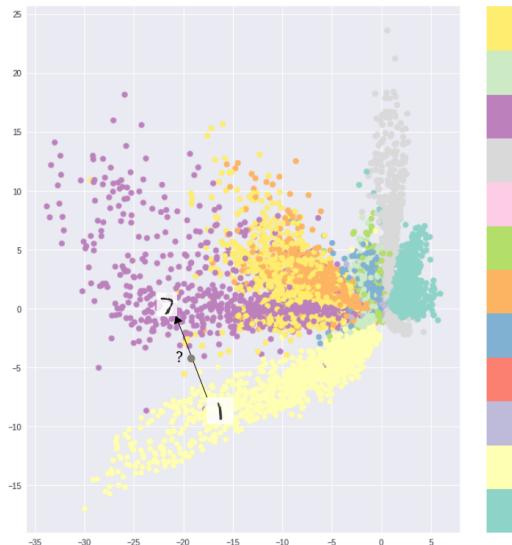
# Autoencoders

- Encode the input into a representation (bottleneck) and reconstruct it with the decoder



# Autoencoders

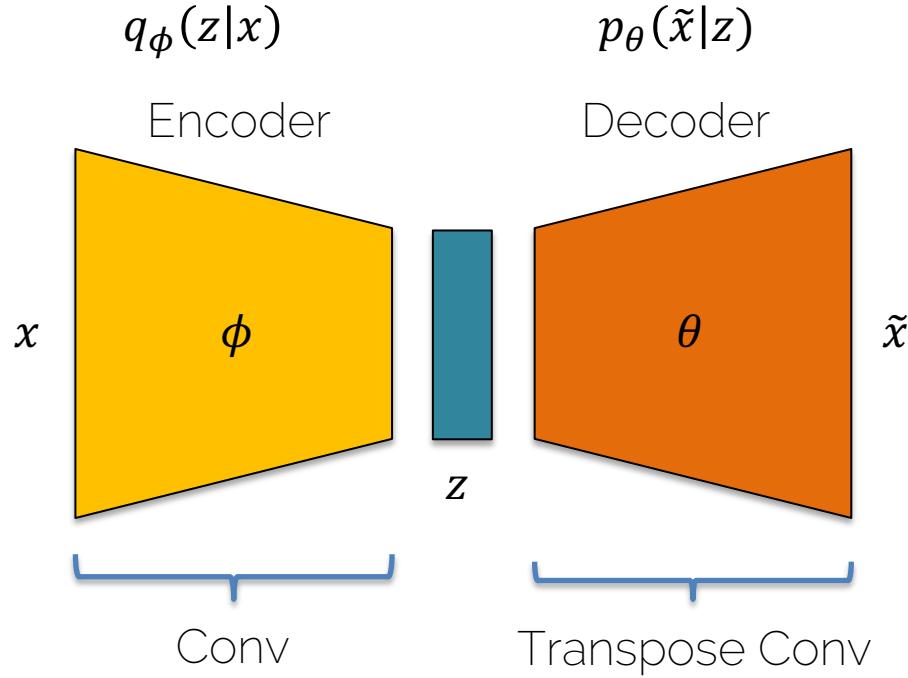
- Encode the input into a representation (bottleneck) and reconstruct it with the decoder



Latent space learned  
by autoencoder on MNIST

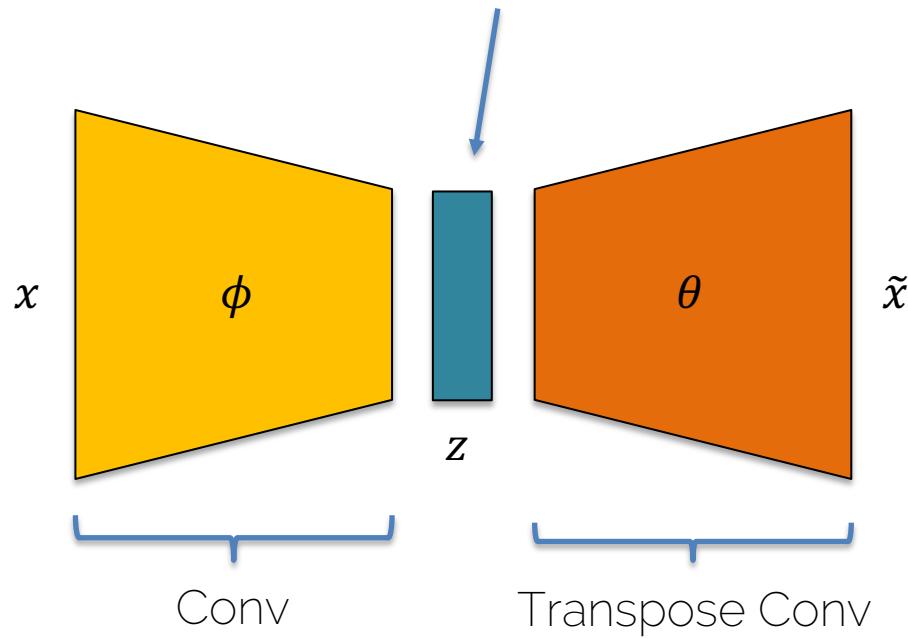
Source: <https://bit.ly/37ctFMS>

# Variational Autoencoder



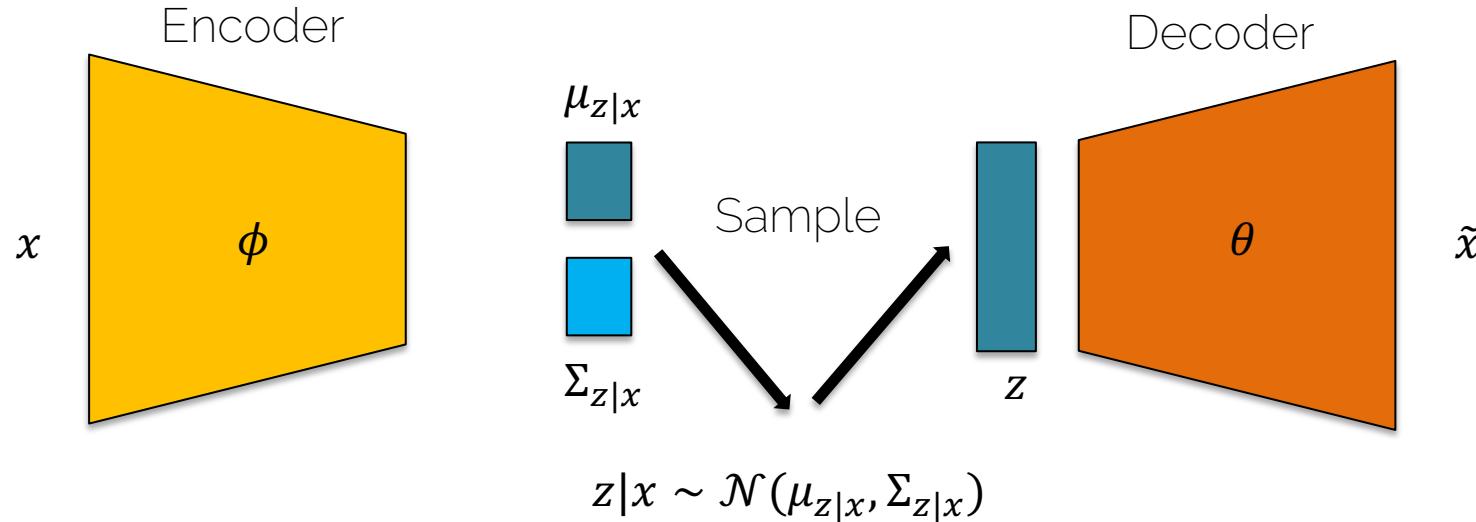
# Variational Autoencoder

Goal: Sample from the latent distribution to generate new outputs!



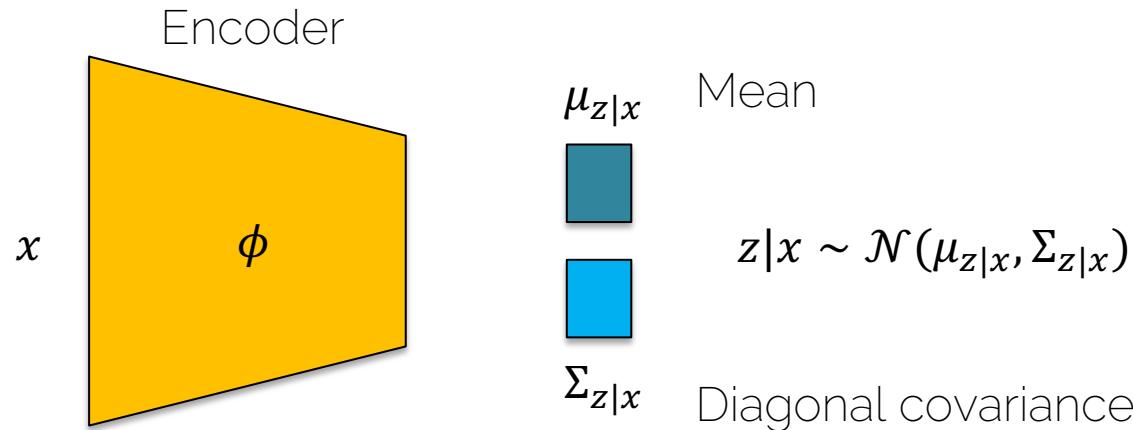
# Variational Autoencoder

- Latent space is now a distribution
- Specifically it is a Gaussian



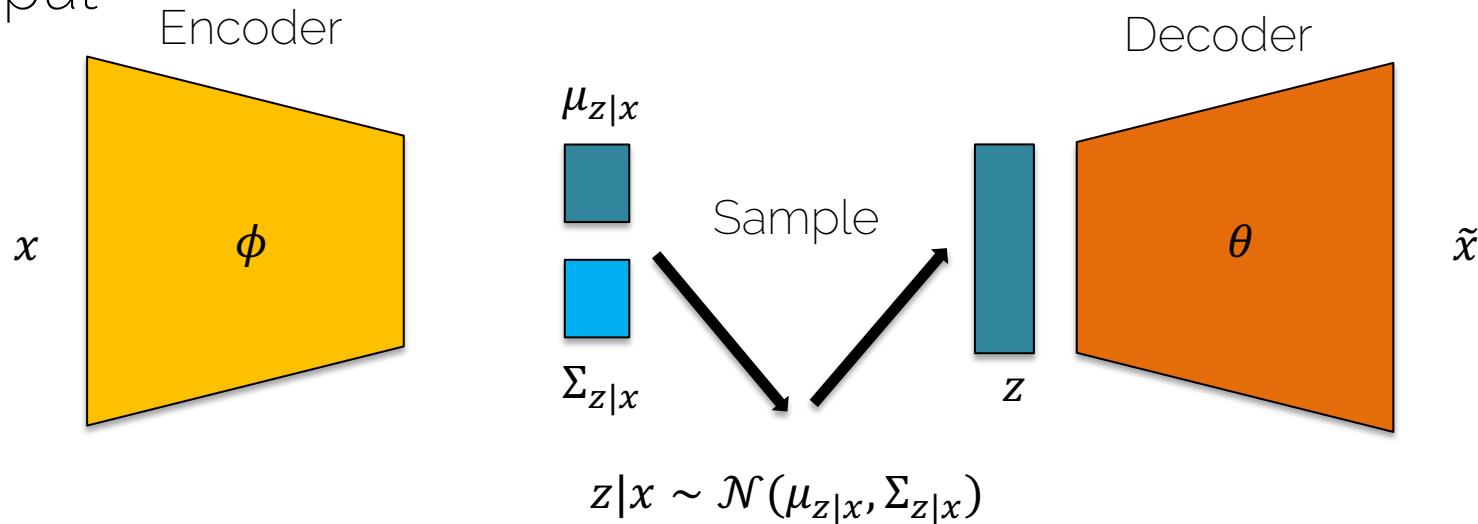
# Variational Autoencoder

- Latent space is now a distribution
- Specifically it is a Gaussian



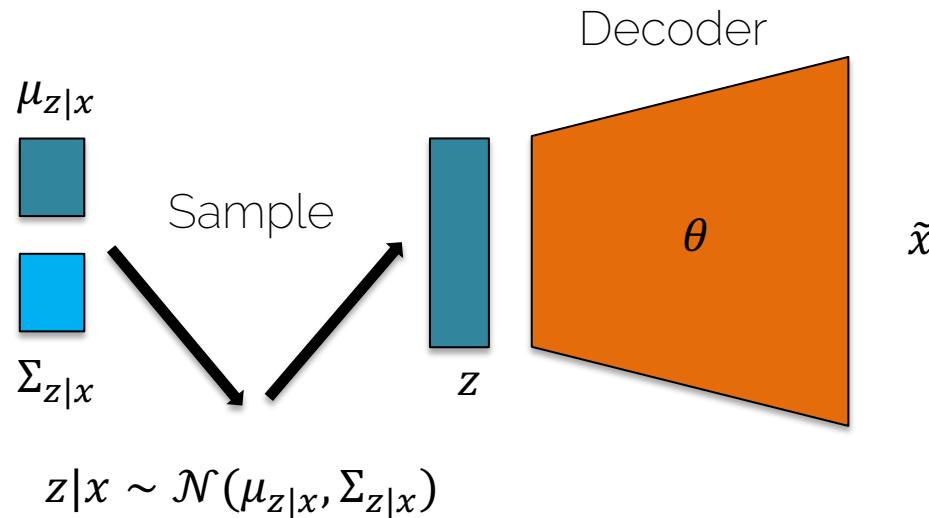
# Variational Autoencoder

- Training: loss makes sure the latent space is close to a Gaussian and the reconstructed output is close to the input



# Variational Autoencoder

- Test: Sample from the latent space



# Autoencoder vs VAE



Autoencoder



Variational Autoencoder



Ground Truth

Source: <https://github.com/kvfrans/variational-autoencoder>

# Generating data

Degree of smile



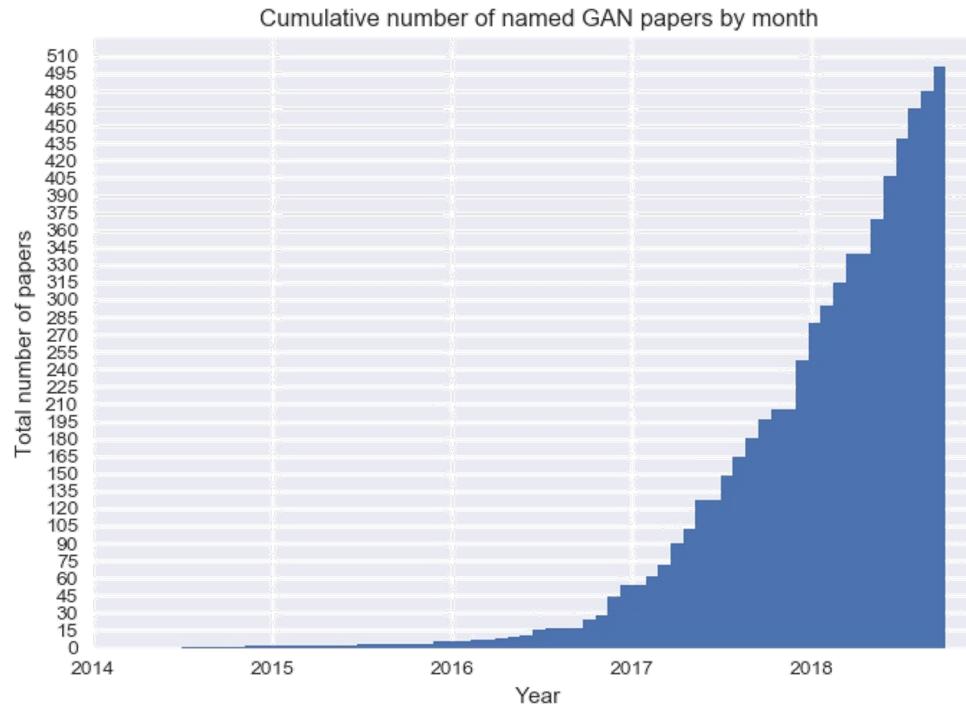
Head pose

# Autoencoder Overview

- Autoencoders (AE)
  - Reconstruct input
  - Unsupervised learning
- Variational Autoencoders (VAE)
  - Probability distribution in latent space (e.g., Gaussian)
  - Interpretable latent space (head pose, smile)
  - Sample from model to generate output

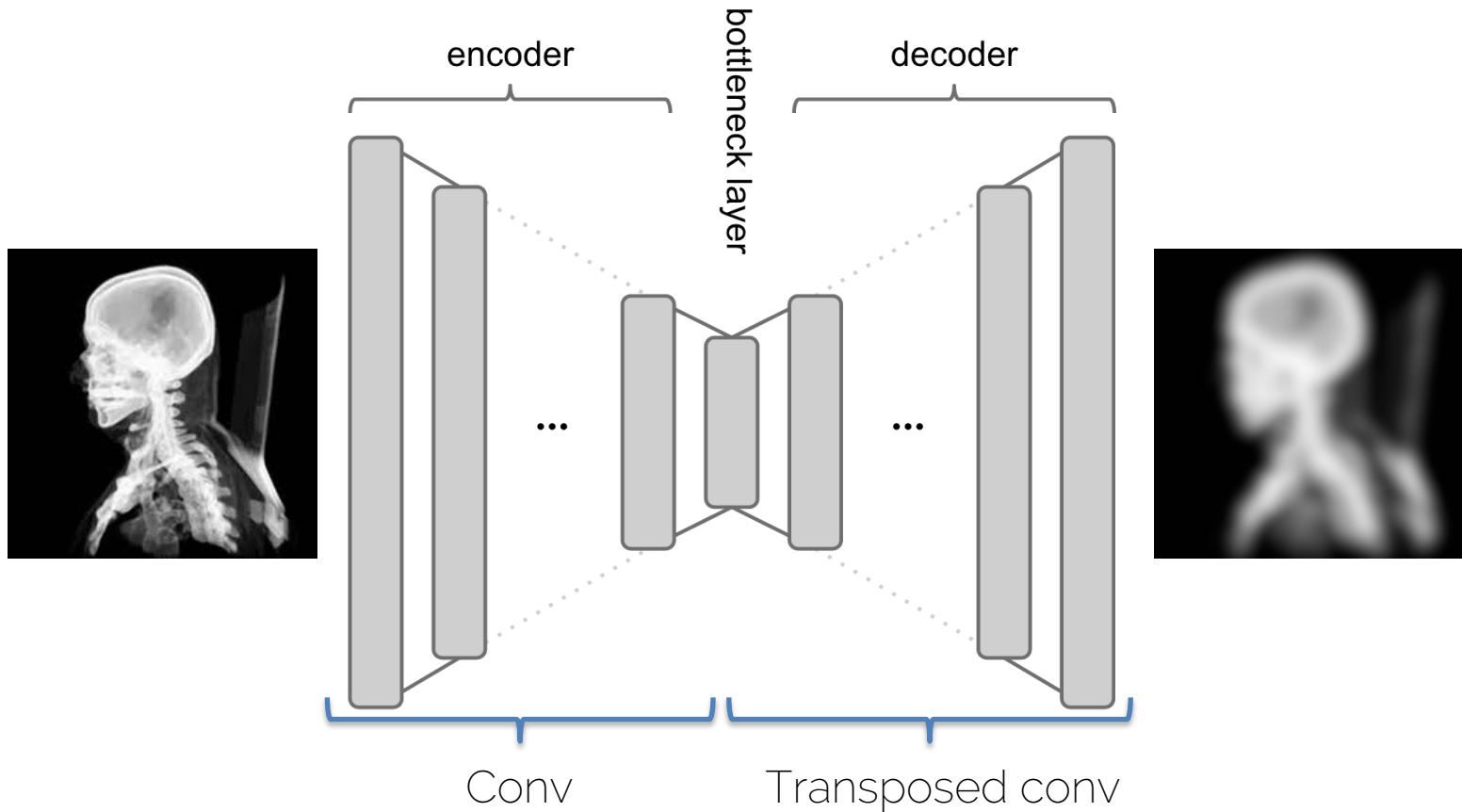
# Generative Adversarial Networks (GANs)

# Generative Adversarial Networks (GANs)



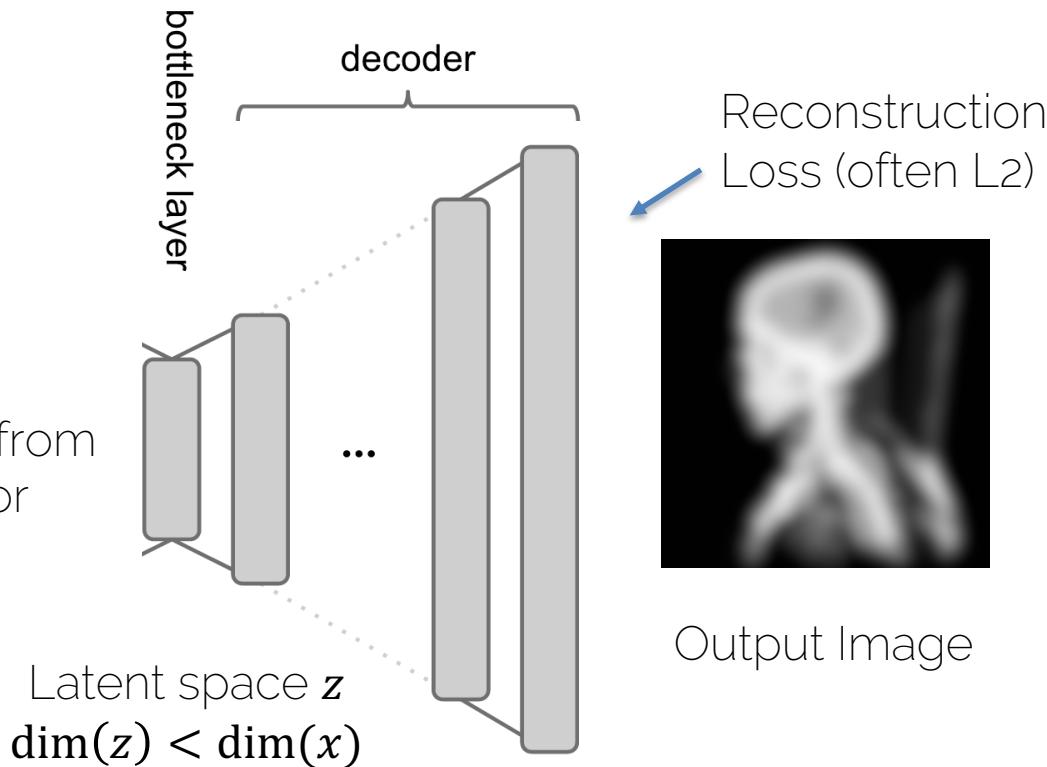
Source: <https://github.com/hindupuravinash/the-gan-zoo>

# Autoencoder



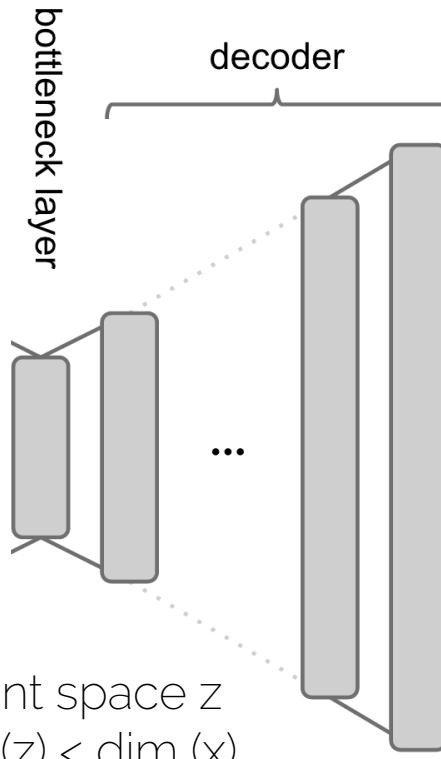
# Decoder as Generative Model

Test time:  
-> reconstruction from  
'random' vector

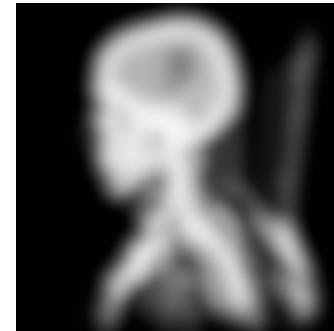


# Decoder as Generative Model

"Test time":  
-> reconstruction from  
'random' vector

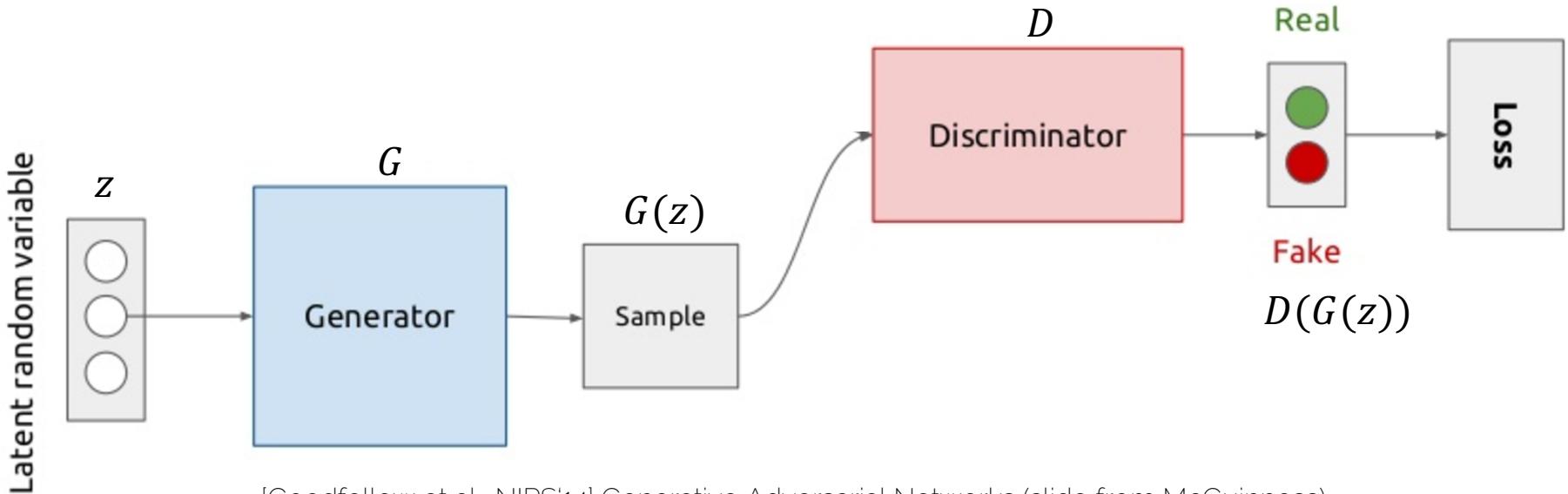


Reconstruction Loss  
Often L2, i.e., sum of squared dist.  
-> L2 distributes error equally  
-> mean is opt.  
-> res. is blurry



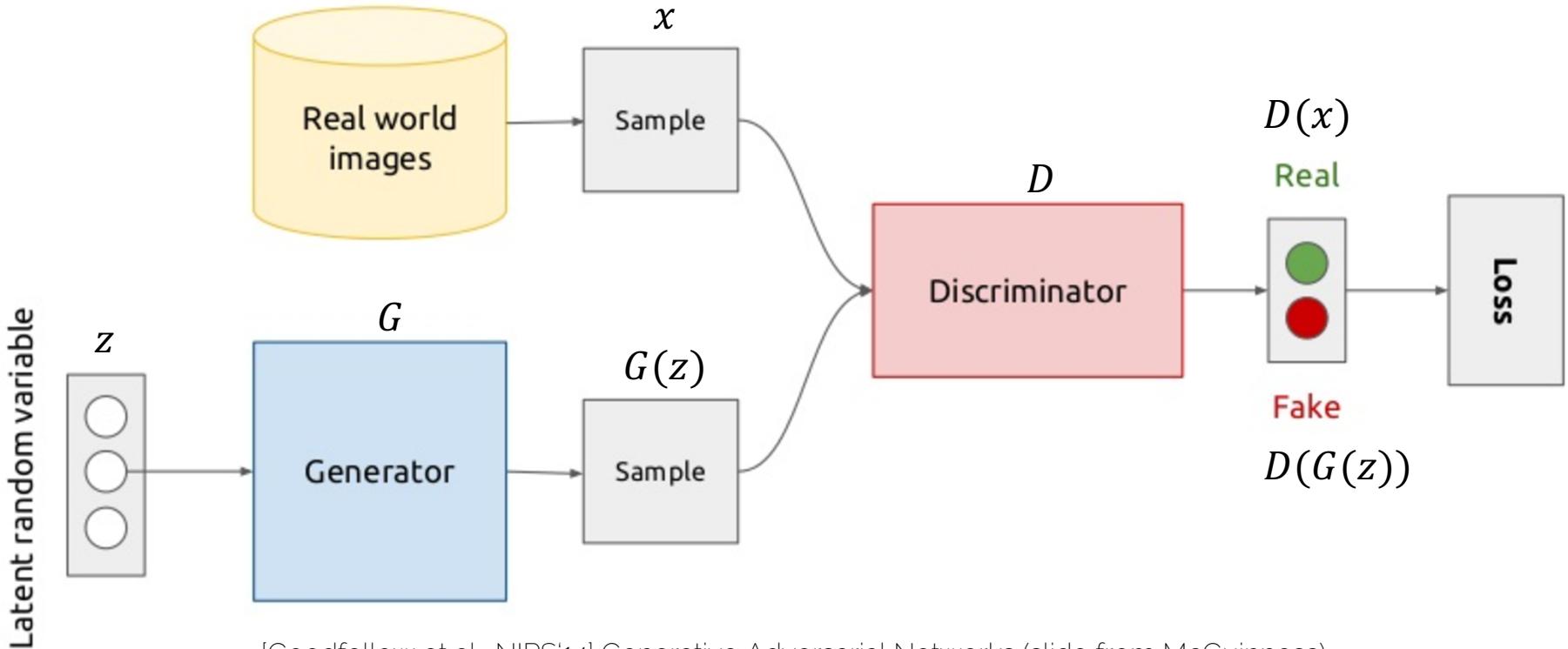
Instead of L2, can we  
"learn" a loss function?

# Generative Adversarial Networks (GANs)



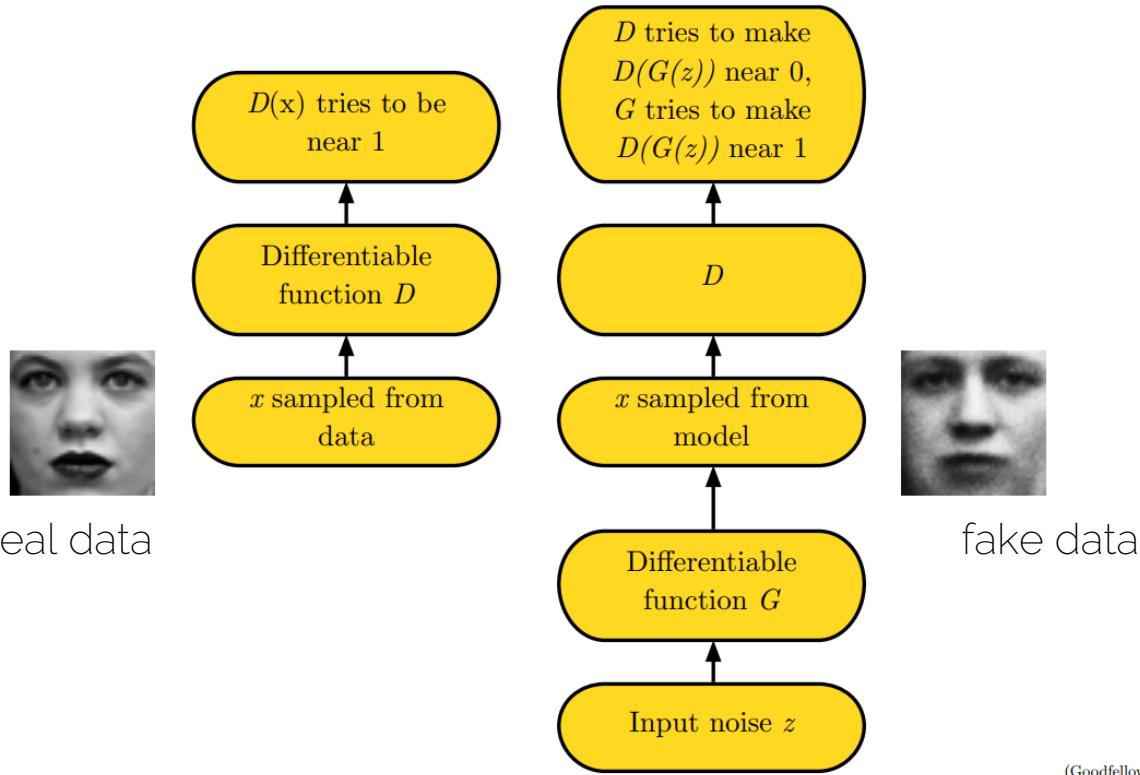
[Goodfellow et al., NIPS'14] Generative Adversarial Networks (slide from McGuinness)

# Generative Adversarial Networks (GANs)



[Goodfellow et al., NIPS'14] Generative Adversarial Networks (slide from McGuinness)

# Generative Adversarial Networks (GANs)



# GANs: Loss Functions

- Discriminator loss

$$J^{(D)} = -\frac{1}{2} \mathbb{E}_{\mathbf{x} \sim p_{data}} \log D(\mathbf{x}) - \frac{1}{2} \mathbb{E}_{\mathbf{z}} \log (1 - D(G(\mathbf{z})))$$

binary cross entropy

- Generator loss

$$J^{(G)} = -J^{(D)}$$

- Minimax Game:

- G minimizes probability that D is correct
  - Equilibrium is saddle point of discriminator loss
    - D provides supervision (i.e., gradients) for G

# GAN Applications

# BigGAN: HD Image Generation



[Brock et al., ICLR'18] BigGAN : Large Scale GAN Training for High Fidelity Natural Image Synthesis

# StyleGAN: Face Image Generation



[Karras et al., '18] StyleGAN : A Style-Based Generator Architecture for Generative Adversarial Networks

[Karras et al., '19] StyleGAN2 : Analyzing and Improving the Image Quality of StyleGAN

# Cycle GAN: Unpaired Image-to-Image Translation



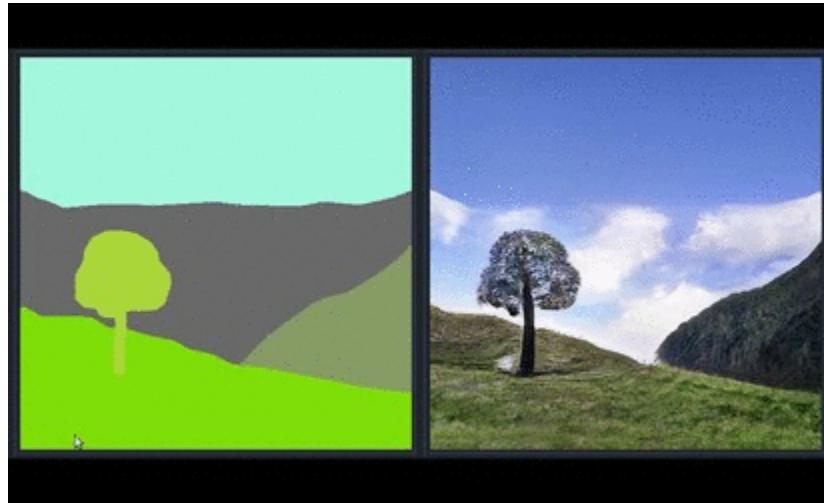
Generator  
A2B



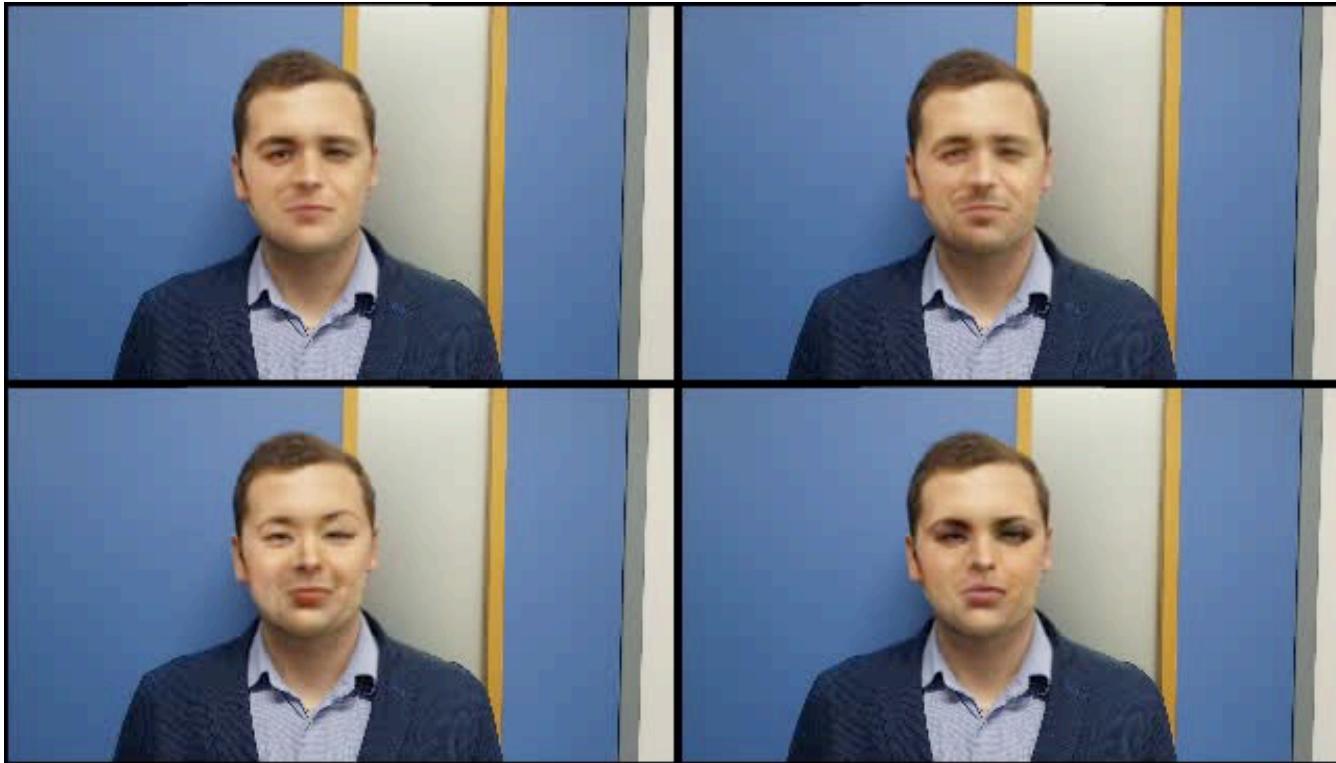
Generator  
A2B



# SPADE: GAN-Based Image Editing



# Face anonymization



[Maximov et al., CVPR'20] CIAGAN: Conditional Identity Anonymization Generative Adversarial Networks

# Reinforcement Learning

# Learning Paradigms in ML

## Supervised Learning

E.g., classification,  
regression

Labeled data

Find mapping from  
input to label

## Unsupervised Learning

E.g., clustering,  
anomaly detection

Unlabeled data

Find structure in data

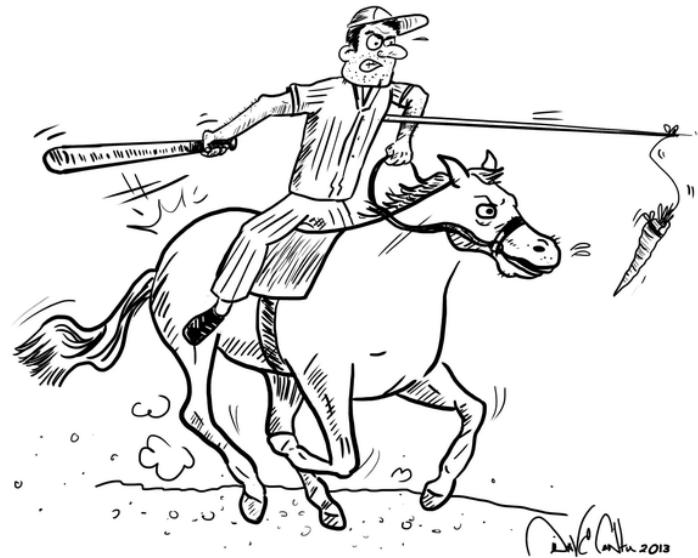
## Reinforcement Learning

Sequential data

Learning by  
interaction with  
the environment

# In a Nutshell

- RL-agent is trained using the “carrot and stick” approach
- Good behavior is encouraged by rewards
- Bad behavior is discouraged by punishment



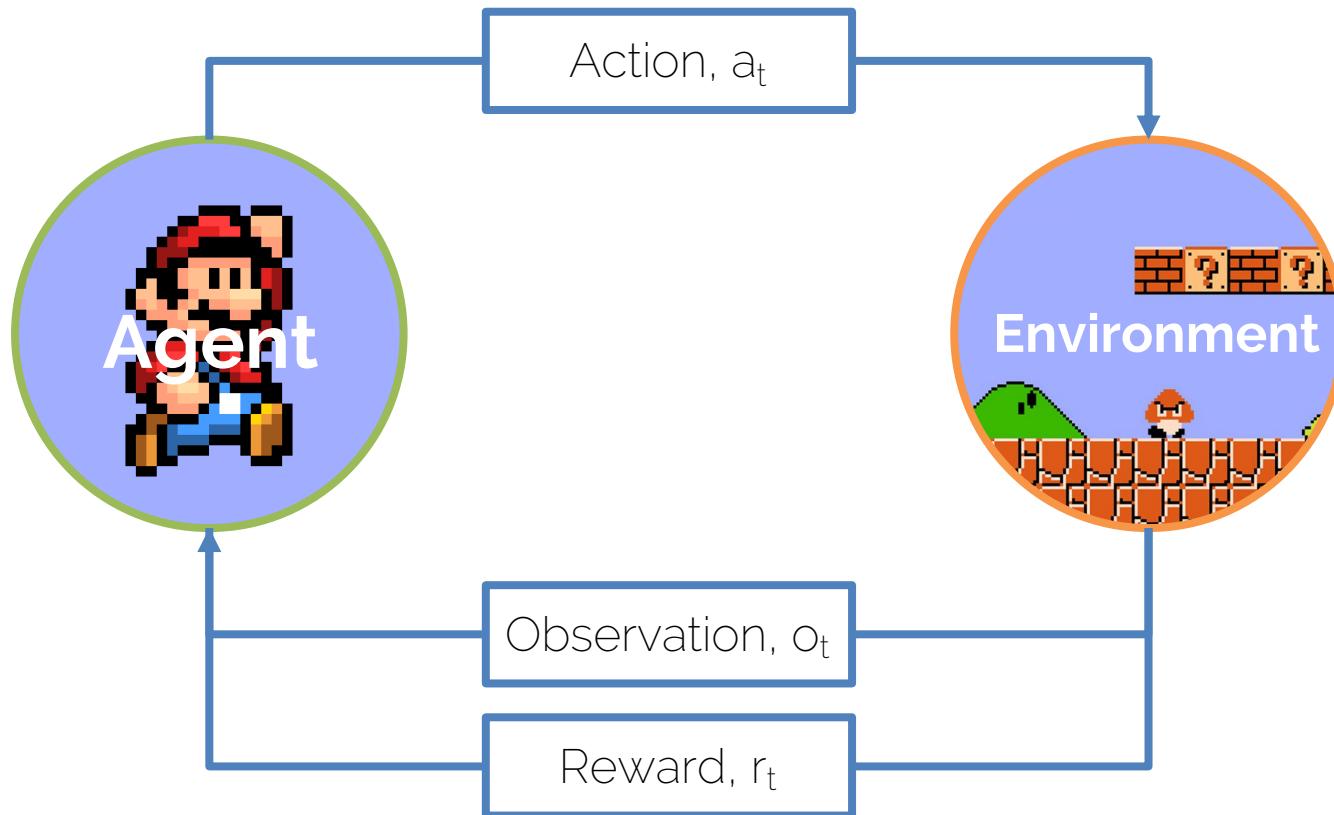
Source: quora.com

# Examples of RL: Learning to Walk



Source: Deepmind.com

# Agent and Environment



# Characteristics of RL

- Sequential, non i.i.d. data (time matters)
- Actions have an effect on the environment  
-> Change future input
- No supervisor, target is approximated by the reward signal

# History and State

- The agent makes decisions based on the history  $h$  of observations, actions and rewards up to time-step  $t$

$$h_t = o_1, a_1, r_1, \dots, a_{t-1}, r_{t-1}, o_t$$

- The state  $s$  contains all the necessary information from  $h \rightarrow s$  is a function of  $h$

$$s_t = f(h_t)$$

# Markov Assumption

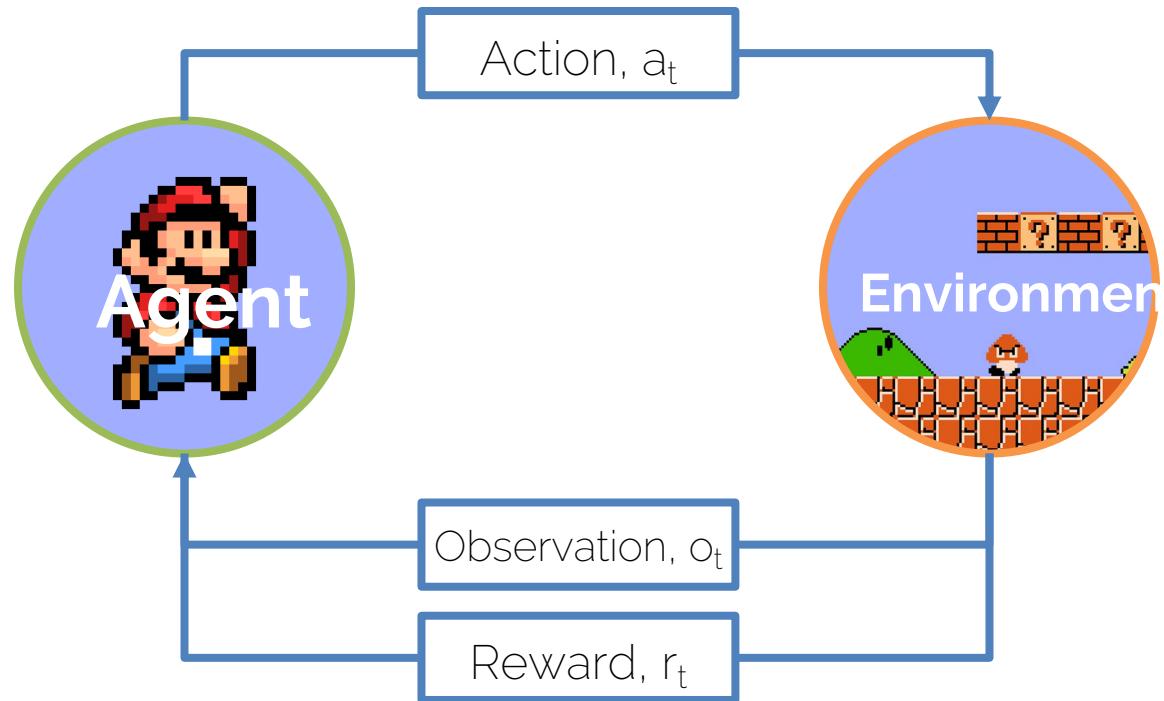
- Problem: History grows linearly over time
- Solution: Markov Assumption
- A state  $S_t$  is Markov if and only if:

$$\mathbb{P}[s_{t+1}|s_t] = \mathbb{P}[s_{t+1}|s_1, \dots s_t]$$

- “The future is independent of the past given the present”

# Agent and Environment

- Reward and next state are functions of current observation  $o_t$  and action  $a_t$  only



# Mathematical Formulation

- The RL problem is a Markov Decision Process (MDP) defined by:  $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathbb{P}, \gamma)$

$\mathcal{S}$  : Set of possible states

$\mathcal{A}$  : Set of possible actions

$\mathcal{R}$  : Distribution of reward given (state, action) pair

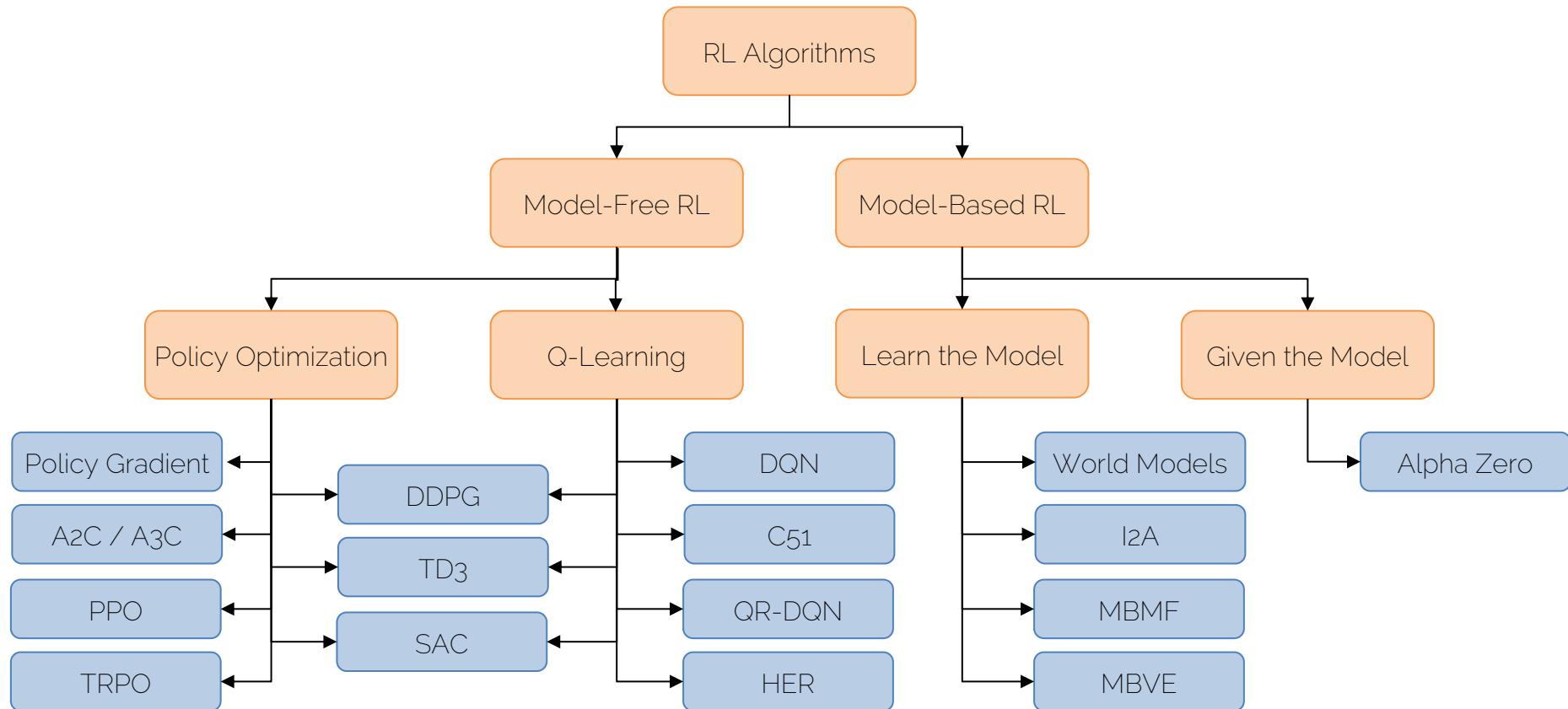
$\mathbb{P}$  : Transition probability of a (state, action) pair

$\gamma$  : Discount factor (discounts future rewards)

# Components of an RL Agent

- Policy  $\pi$ : Behavior of the agent  
-> Mapping from state to action:  $a = \pi(s)$
- Value-, Q-Function: How good is a state or (state, action) pair  
-> Expected future reward

# Taxonomy of RL Algorithms



# RL Milestones: Playing Atari



- Mnih et al. 2013, first appearance of DQN
- Successfully learned to play different Atari games like Pong, Breakout, Space Invaders, Seaquest and Beam Rider

# RL Milestones: AlphaZero (StarCraft)

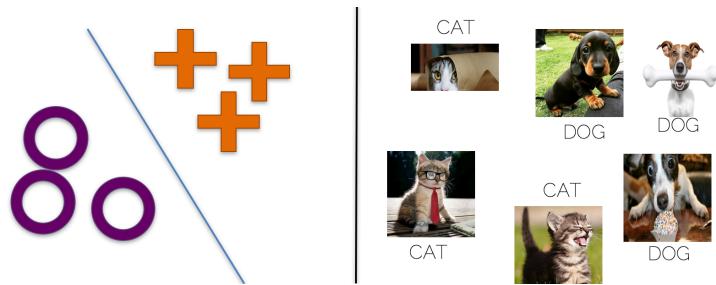
- Model: Transformer network with a LSTM core
- Trained on 200 years of StarCraft play for 14 days
- 16 Google v3 TPUs
- December 2018:  
Beats MaNa, a professional StarCraft player (world rank 13)



# I2DL Overview

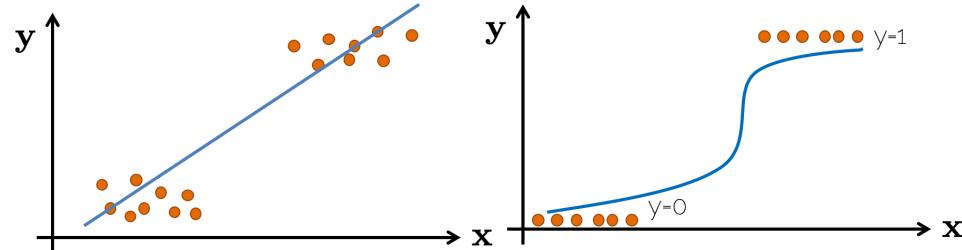
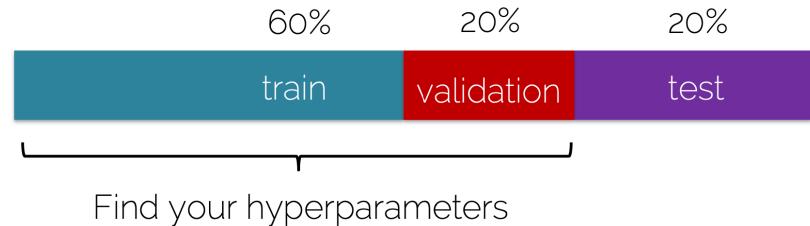
# Machine Learning Basics

- Unsupervised vs Supervised Learning



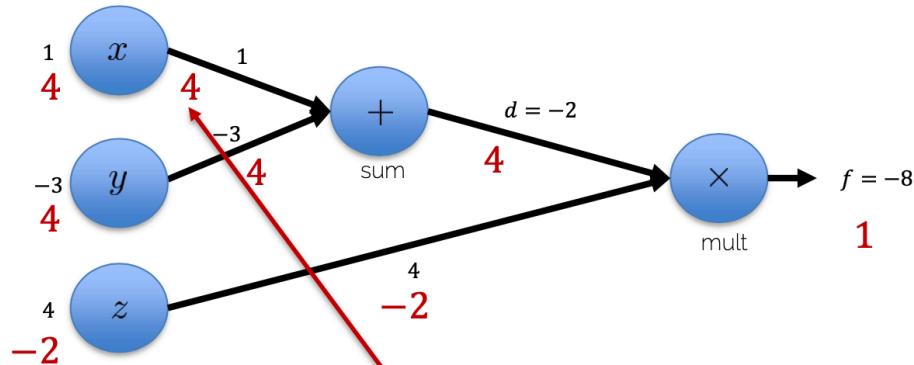
- Linear vs logistic regression

- Data splitting



# Intro to Neural Networks

- Backpropagation

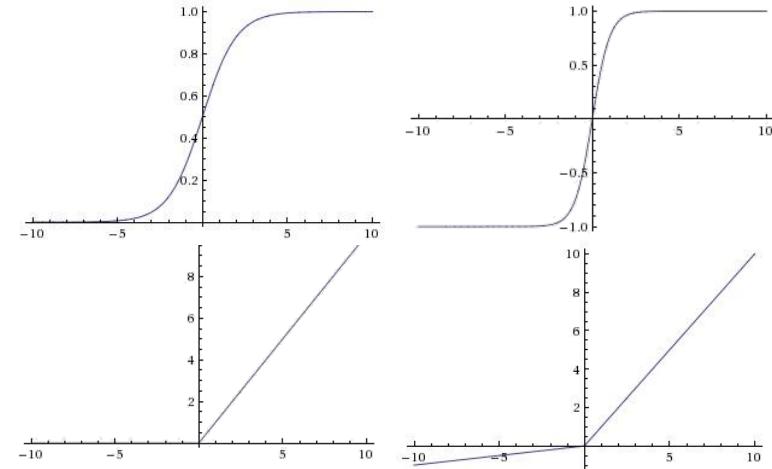


Chain Rule:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial d} \cdot \frac{\partial d}{\partial x}$$

$$\frac{\partial f}{\partial x}$$

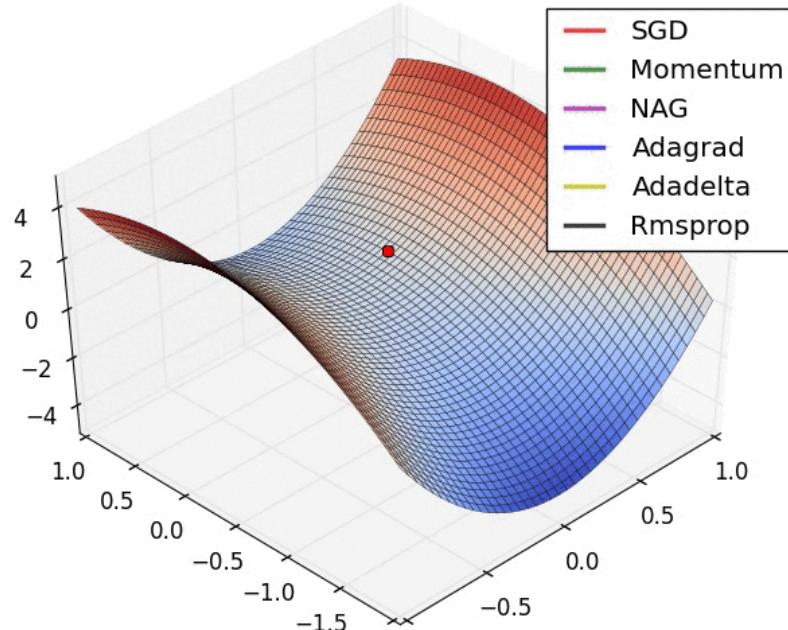
- Activation functions



- Loss functions
  - Comparison & effects

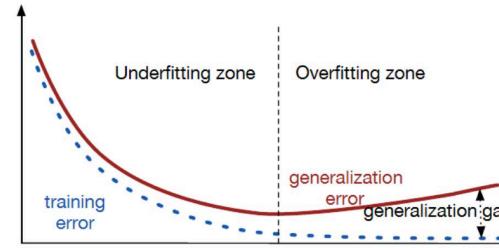
# Training Neural Networks

- Gradient Descent/ SGD

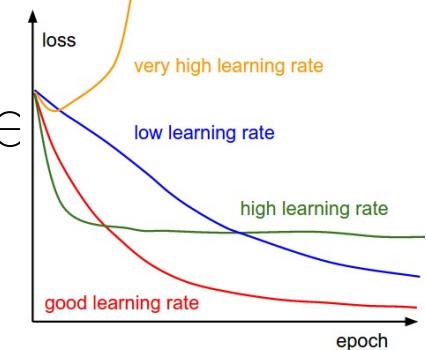


Source: <http://ruder.io/optimizing-gradient-descent/>,  
<https://srdas.github.io/DLBook/ImprovingModelGeneralization.html>,  
<http://cs231n.github.io/neural-networks-3/>

- Regularization

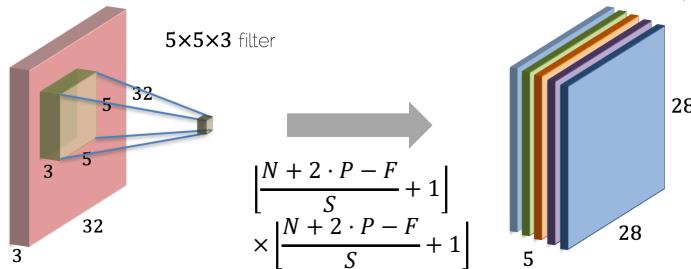


- Parameter & interpretation

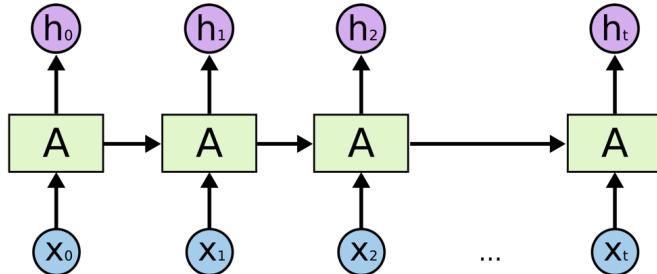


# Typology of Neural Networks

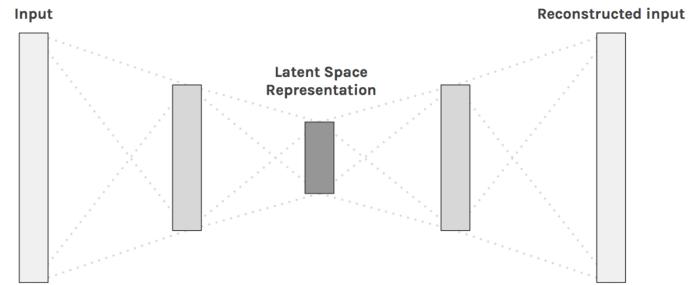
- CNNs



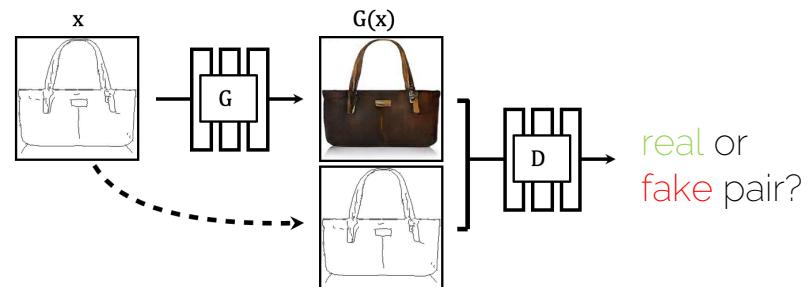
- RNNs



- Autoencoder

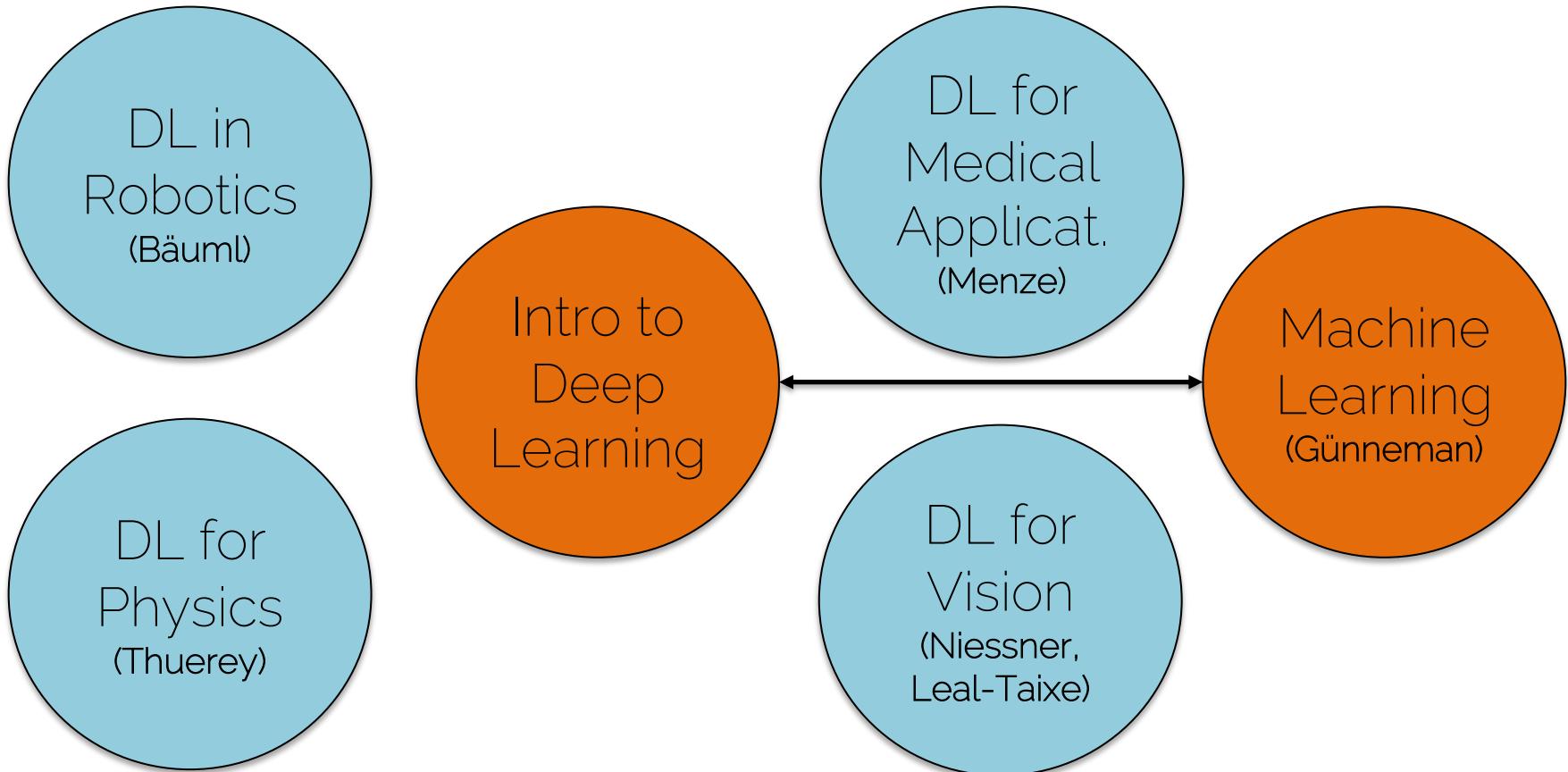


- GANs



# Other DL Courses

# Deep Learning at TUM



# Deep Learning at TUM

- Keep expanding the courses on Deep Learning
- This Introduction to Deep Learning course is the basis for a series of Advanced DL lectures on different topics
- Advanced topics are only for Master students
  - Preparation for MA theses, etc.

# Advanced DL for Computer Vision

- Deep Learning for Vision (Profs. Niessner/Leal-Taixé)
- Syllabus
  - Advanced architectures, e.g., Siamese neural networks
  - Variational Autoencoders
  - Generative models, e.g. GAN,
  - Multi-dimensional CNN
  - Graph neural networks
  - Domain adaptation

# Advanced DL for Computer Vision

- Deep Learning for Vision (WS19/20)
  - 2 V + 3 P
  - Must have attended the Intro to DL
  - Practical part is a project that will last the whole semester
  - Please do not sign up unless you are willing to spend a lot of time on the project!

# Detection, Segmentation and Tracking

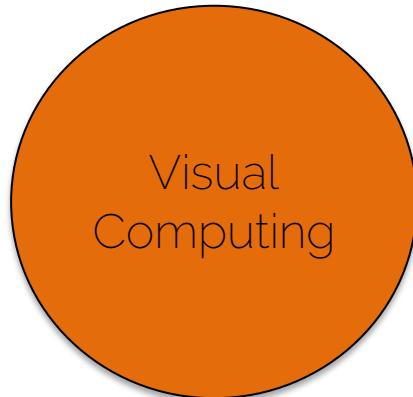
- Computer Vision 3: Detection, Segmentation and Tracking (Summer Semester Only!)
  - Object detection (one-stage vs two-stage)
  - Multi-object tracking
  - Semantic/Instance/Video segmentation
  - Trajectory prediction
  - 3D Detection/Segmentation/Tracking
  - Prof. Leal-Taixé

# Next Dates and Exam

- This is the last lecture!
- Friday (24.07): Guest Lecture, Prof. Antonio Torralba
  - Live event on Youtube, also available later
- Exam date: August 11<sup>th</sup> at 08:00-09:30
- There will NOT be a retake exam
- Neither cheat sheet nor calculator during the exam

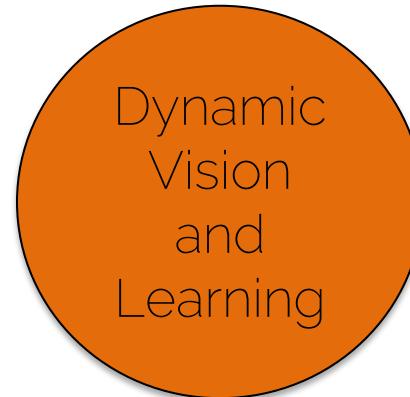
# Thank you

Niessner



- 3D scanning/reconstruction
- DL for 3D understanding
- Video Generation / Faces
- Neural Rendering

Leal-Taixé



- Video segmentation
- Multi-object tracking
- Camera localization
- Video anonymization

Good Luck  
in the Exam ☺

# References for Further Reading

- <https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf>
- <https://phillipi.github.io/pix2pix/>
- [http://cs231n.stanford.edu/slides/2017/cs231n\\_2017\\_lecure13.pdf](http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecure13.pdf)