Introduction to Deep Learning

Michael Kagan

SLAC

CERN Openlab Summer Student Lectures
July 5, 2024

Modern Neural Networks

People are now building a **new kind of software** by assembling networks of **parameterized functional blocks** and by **training them from examples using some form of gradient-based optimization**.

- Yann LeCun, 2018

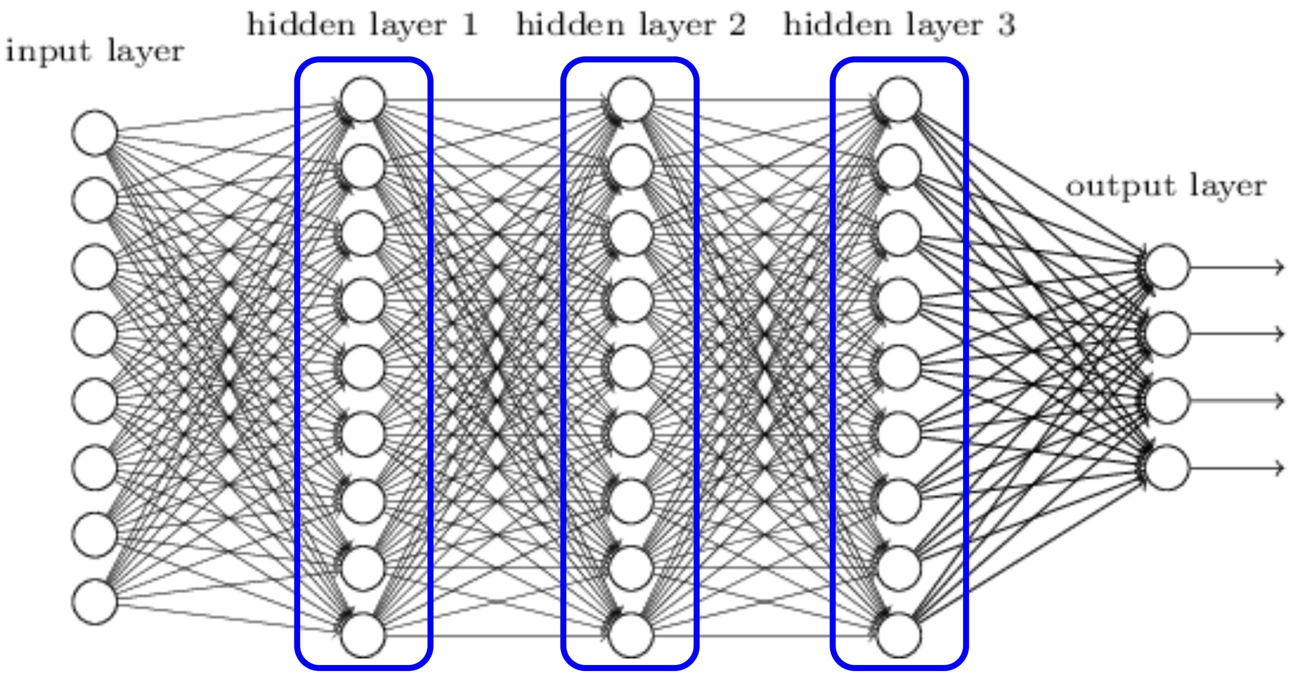
Modern Neural Networks

People are now building a **new kind of software** by assembling networks of **parameterized functional blocks** and by **training them from examples using some form of gradient-based optimization**.

- Yann LeCun, 2018

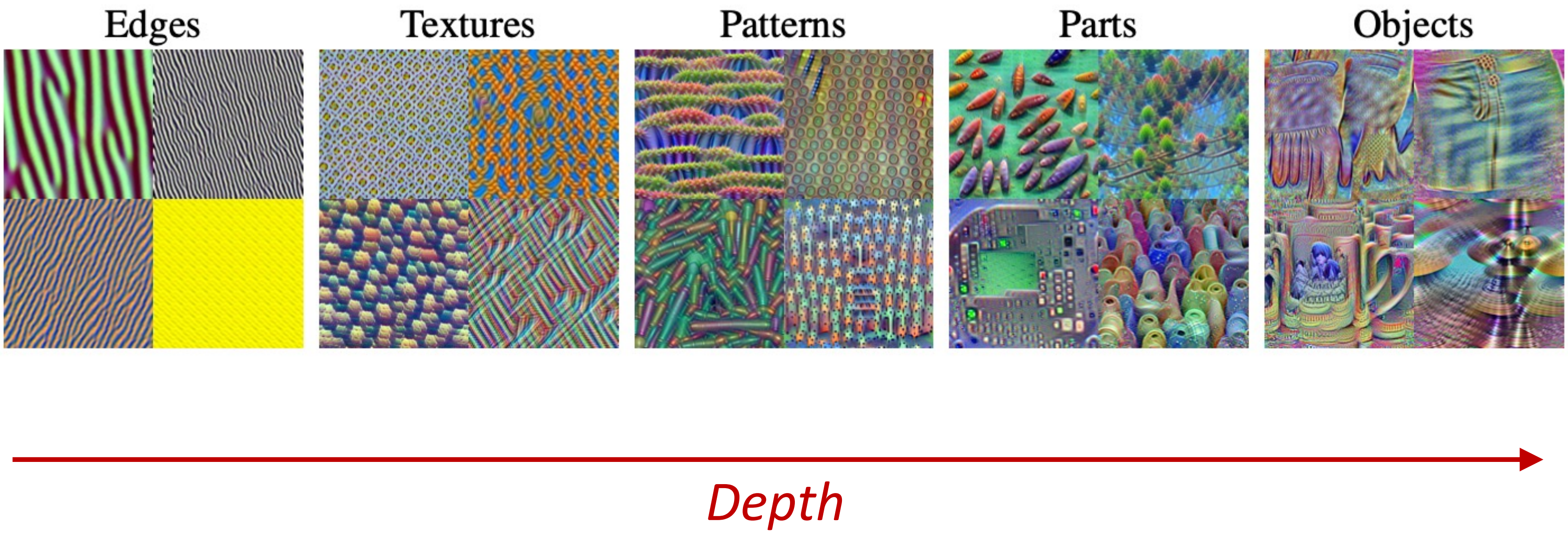
- Non-linear operations of data with parameters
- Layers (set of operations) designed to perform specific mathematical operations
- Chain together layers to perform desired computation
- Train system (with examples) for desired computation using gradient descent

Deep Neural Networks



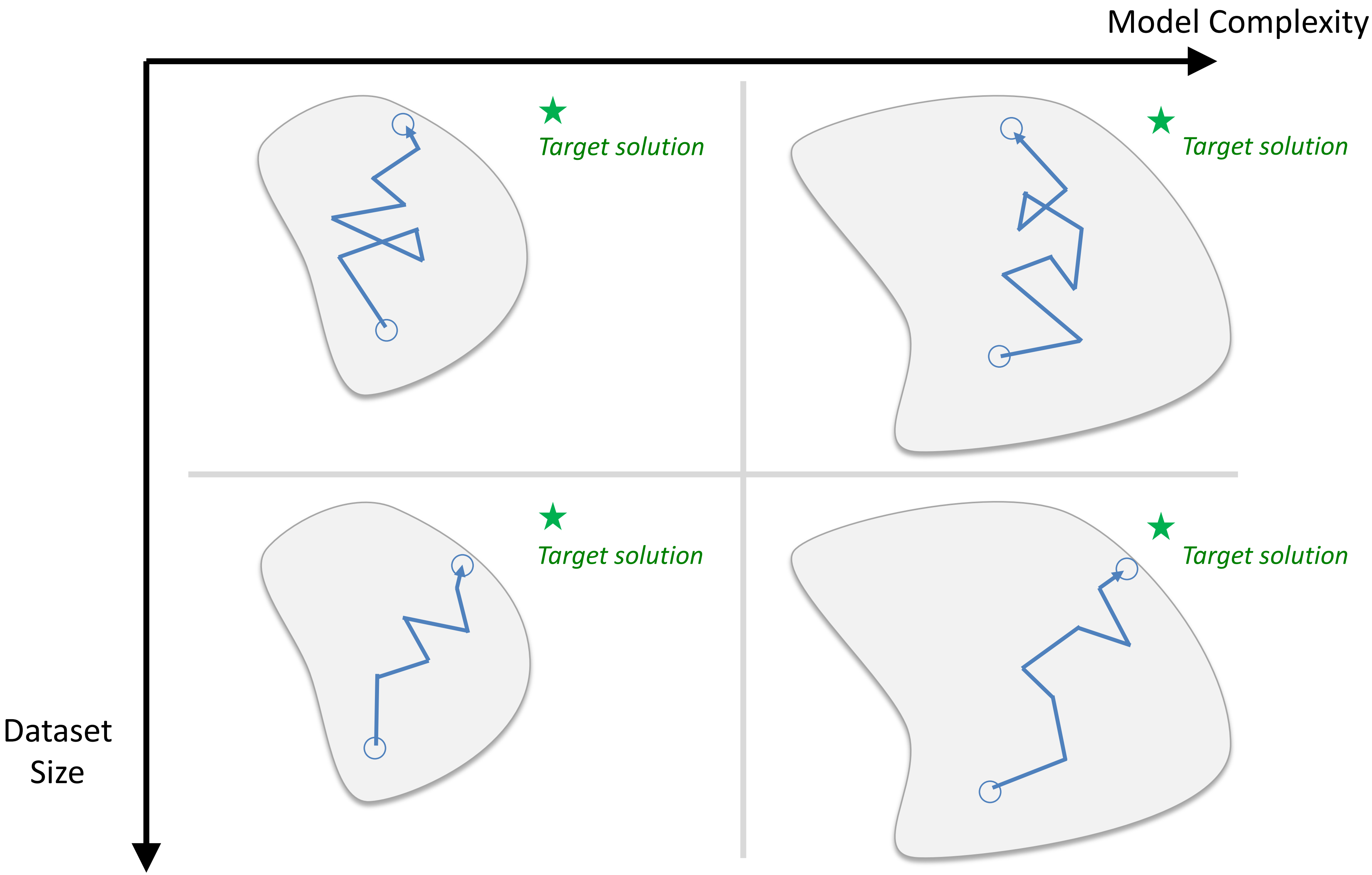
- As data complexity grows, need exponentially large number of neurons in a single-hidden-layer network to capture all structure in data
- Deep networks *factorize learning* of structure in data across layers
- Large datasets, fast computing (GPU / TPU) and new training procedures / network structures made training possible

Hierarchical Learning of Features

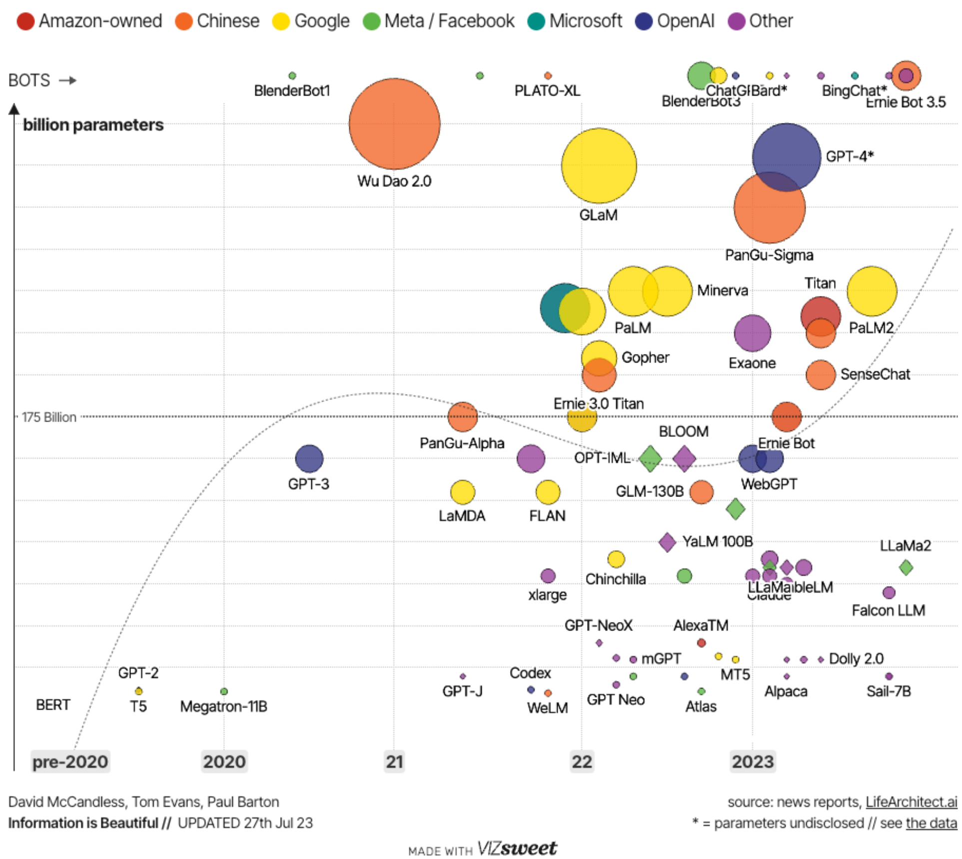


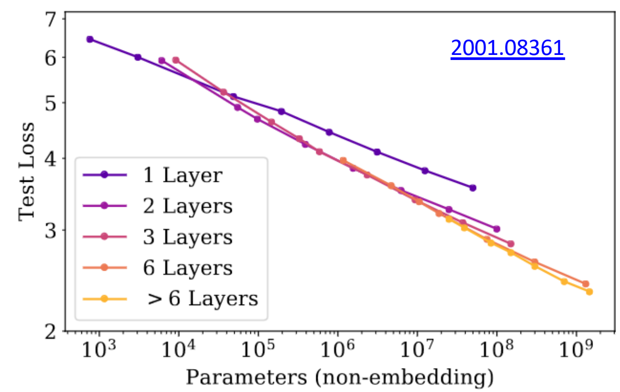
[2212.06727](https://arxiv.org/abs/2212.06727)

More Complex Models – Bigger Search Space
More Data – Find Better Solutions



The Power of Scale: Large Models, Data, Compute





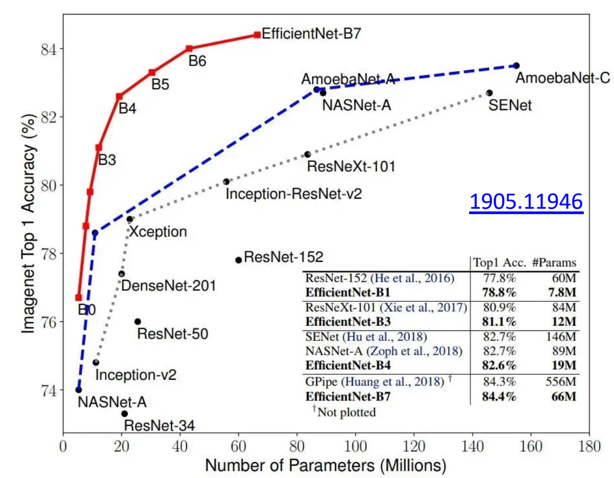
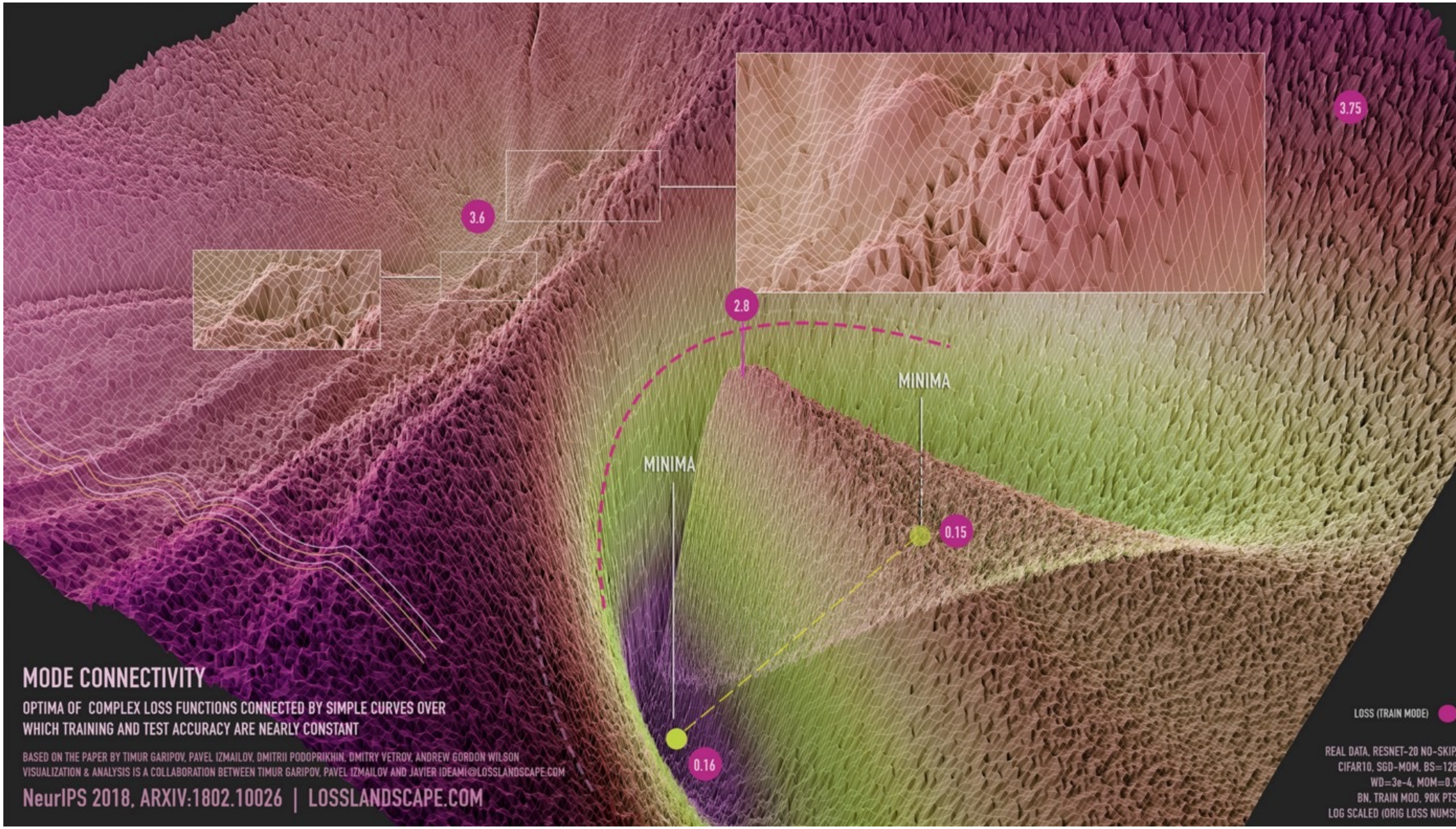


Image credit: D. McCandless, T. Evans, P. Barton

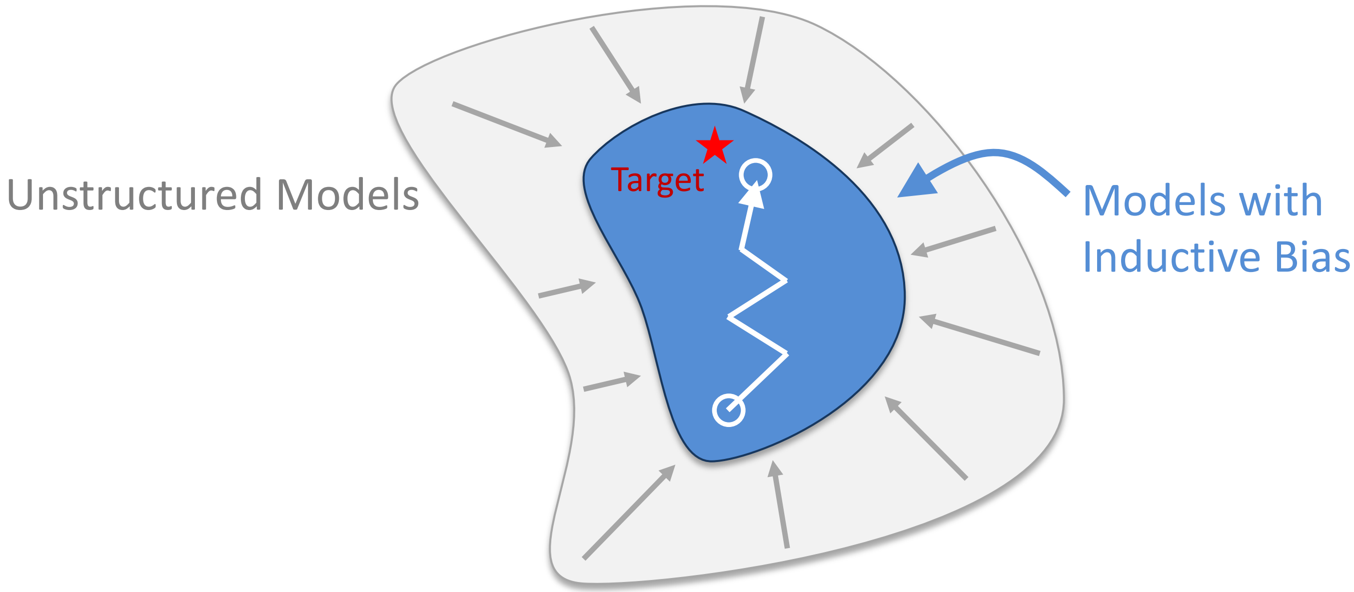
Deep Neural Networks Loss Landscape



https://arxiv.org/abs/1802.10026

Choosing the right function…

- We know a lot about our data
  - What transformations shouldn't affect predictions
  - Symmetries, structures, geometry, …
- **Inductive Bias:** we can match models to this knowledge
  - Throw out irrelevant functions we know aren't the solution
  - Bias the learning process towards good solutions



Choosing the right function…

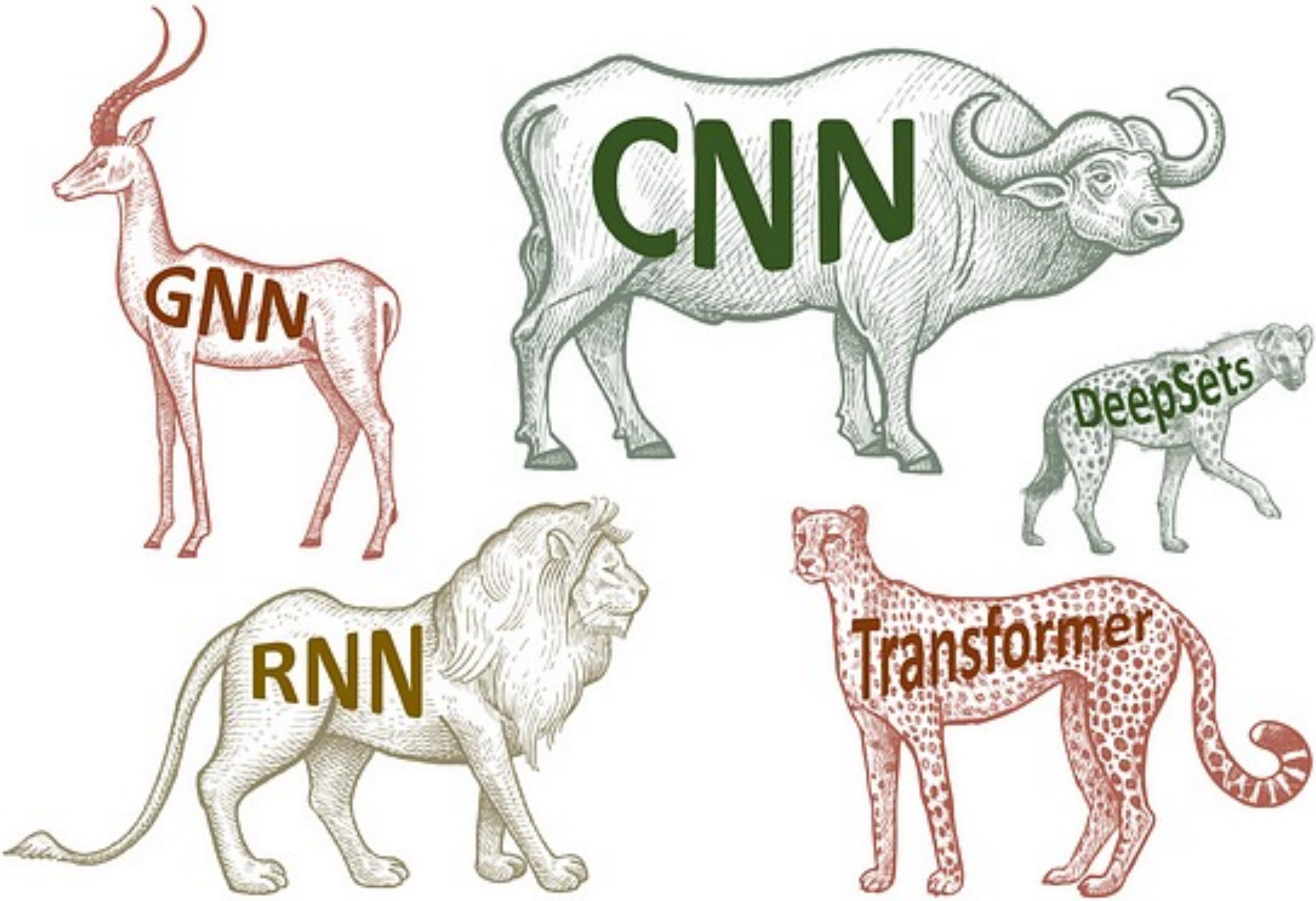
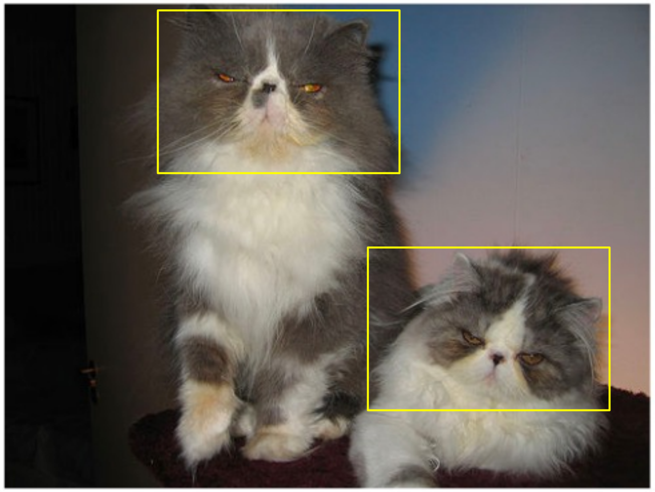


Image credit: Michael Bronstein

Convolutional Neural Networks

Convolutional Neural Networks

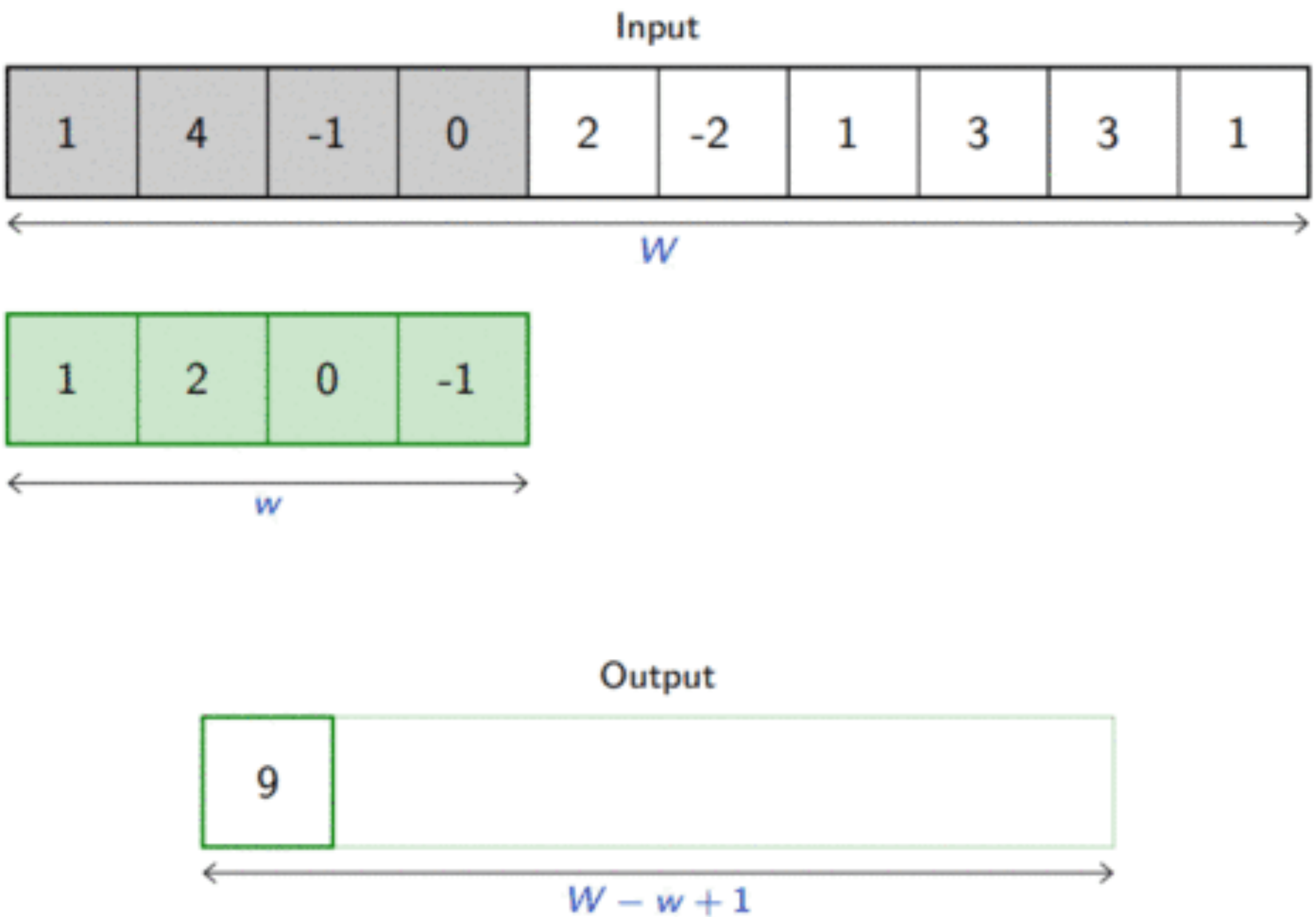
- When the structure of data includes “invariance to translation”, a representation meaningful at a certain location can / should be used everywhere



- Convolutional layers build on this idea, that the same “local” transformation is applied everywhere and preserves the signal structure

Fleuret, Deep Learning Course

1D Convolutional Layer Example

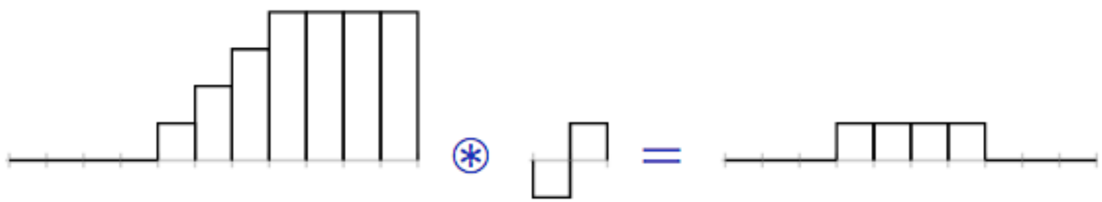


Fleuret, [Deep Learning Course]

Convolutional Filters

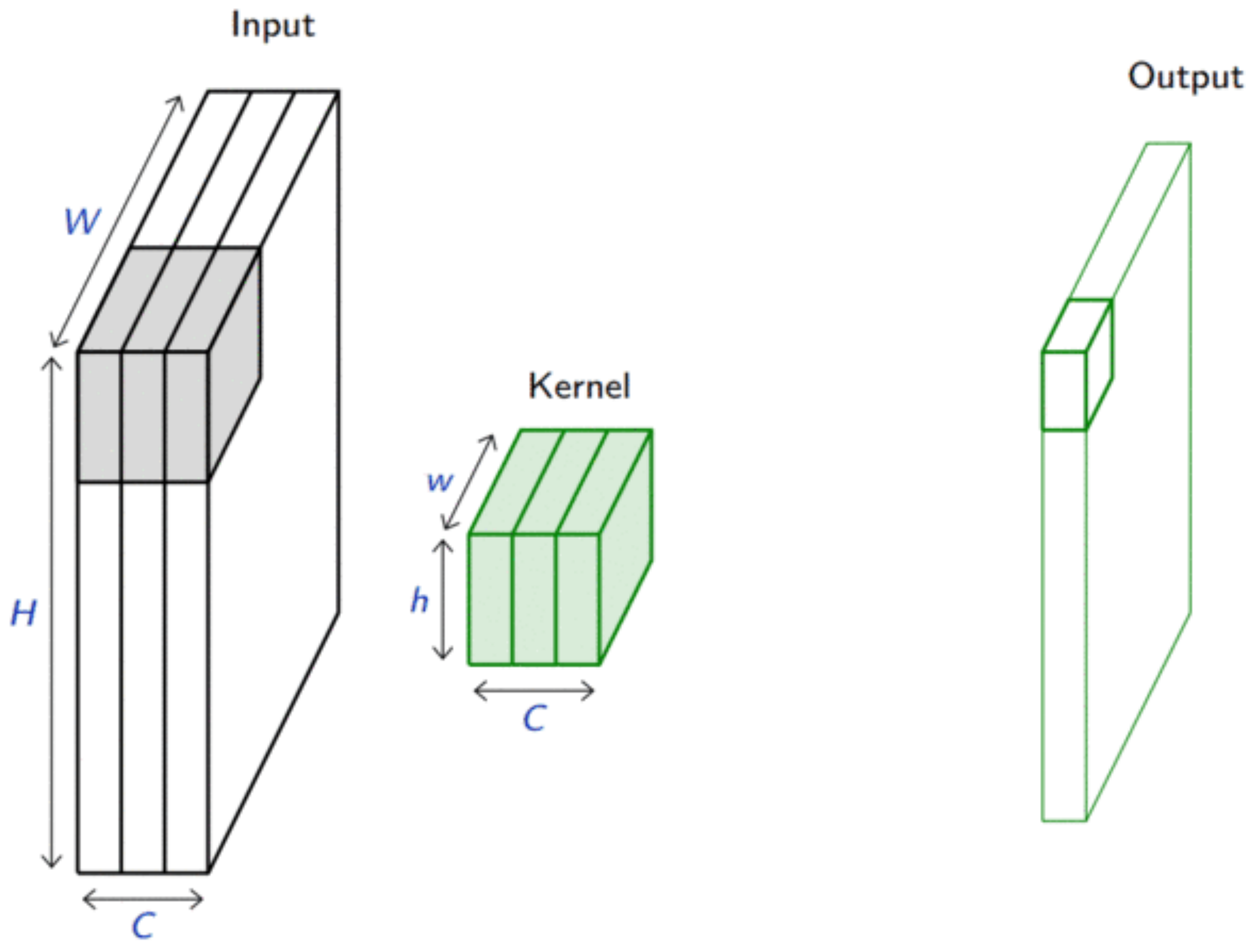
Convolution can implement in particular differential operators, *e.g.*

$$(0,0,0,0,1,2,3,4,4,4,4) \circledast (-1,1) = (0,0,0,1,1,1,1,0,0,0).$$



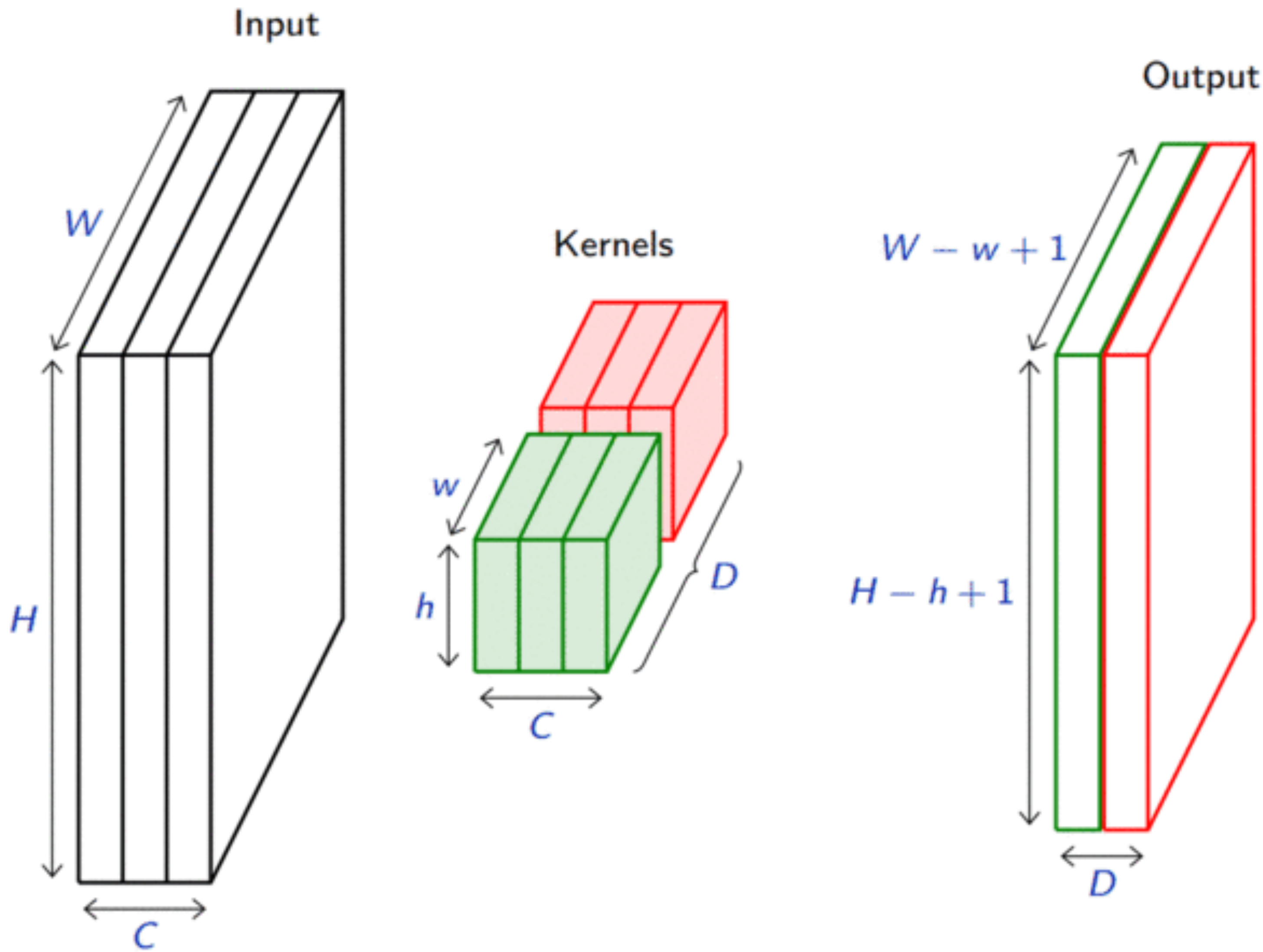
or crude "template matcher", *e.g.*

2D Convolution Over Multiple Channels



Fleuret, [Deep Learning Course]

2D Convolution Over Multiple Channels



Fleuret, Deep Learning Course

Shared Weights: Economic and Equivariant

- Parameters are *shared* by each neuron producing an output in the activation map

- Dramatically reduces number of weights needed to produce an activation map
  - Data: $256 \times 256 \times 3$ RGB image
  - Kernel: $3 \times 3 \times 3 \rightarrow 27$ weights
  - Fully connected layer:
    - $256 \times 256 \times 3$ inputs → $256 \times 256 \times 3$ outputs → $O(10^{10})$ weights

Y. LeCun et. al. 1998

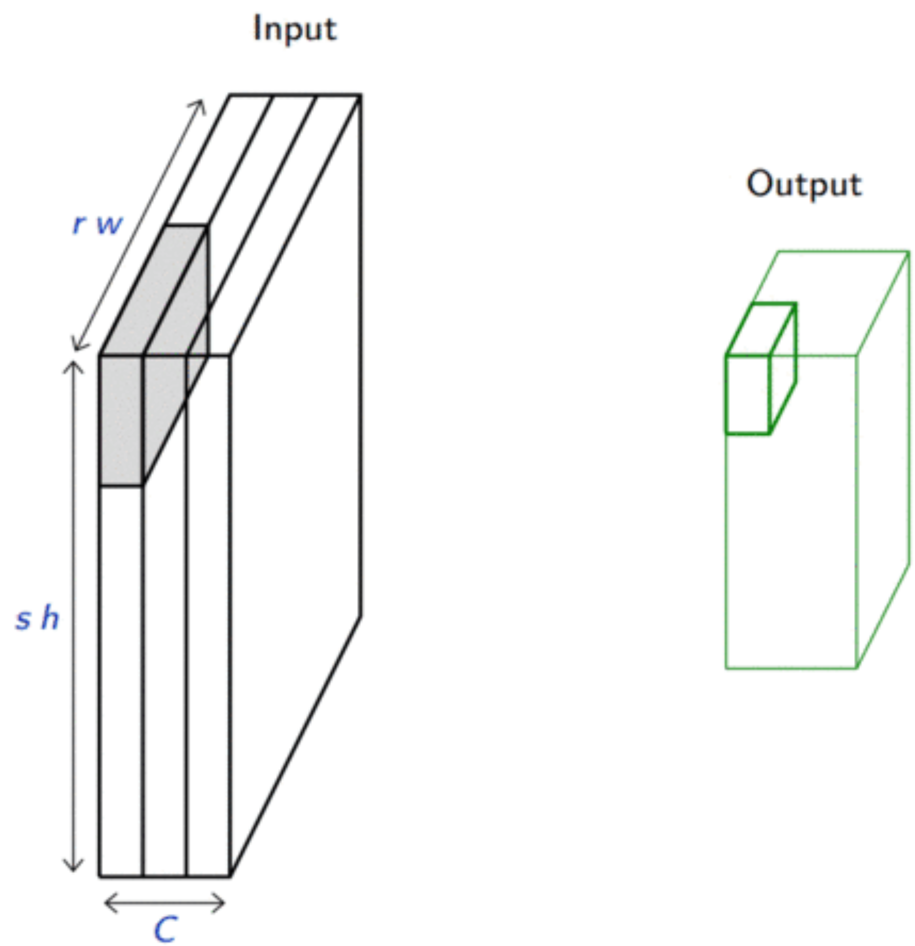
Shared Weights: Economic and Equivariant

- Parameters are *shared* by each neuron producing an output in the activation map
- Dramatically reduces number of weights needed to produce an activation map
- Convolutional layer does pattern matching at any location → Equivariant to translation

Y. LeCun et. al. 1998

Pooling

- In each channel, find *max* or *average* value of pixels in a pooling area of size $h \times w$



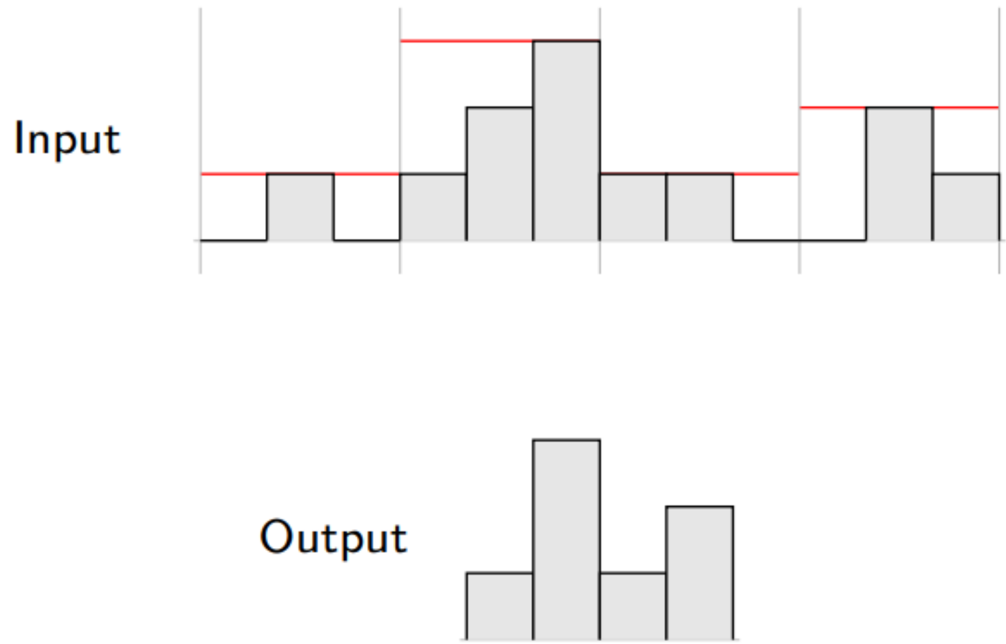
Fleuret, Deep Learning Course

Pooling

- In each channel, find *max* or *average* value of pixels in a pooling area of size $h \times w$

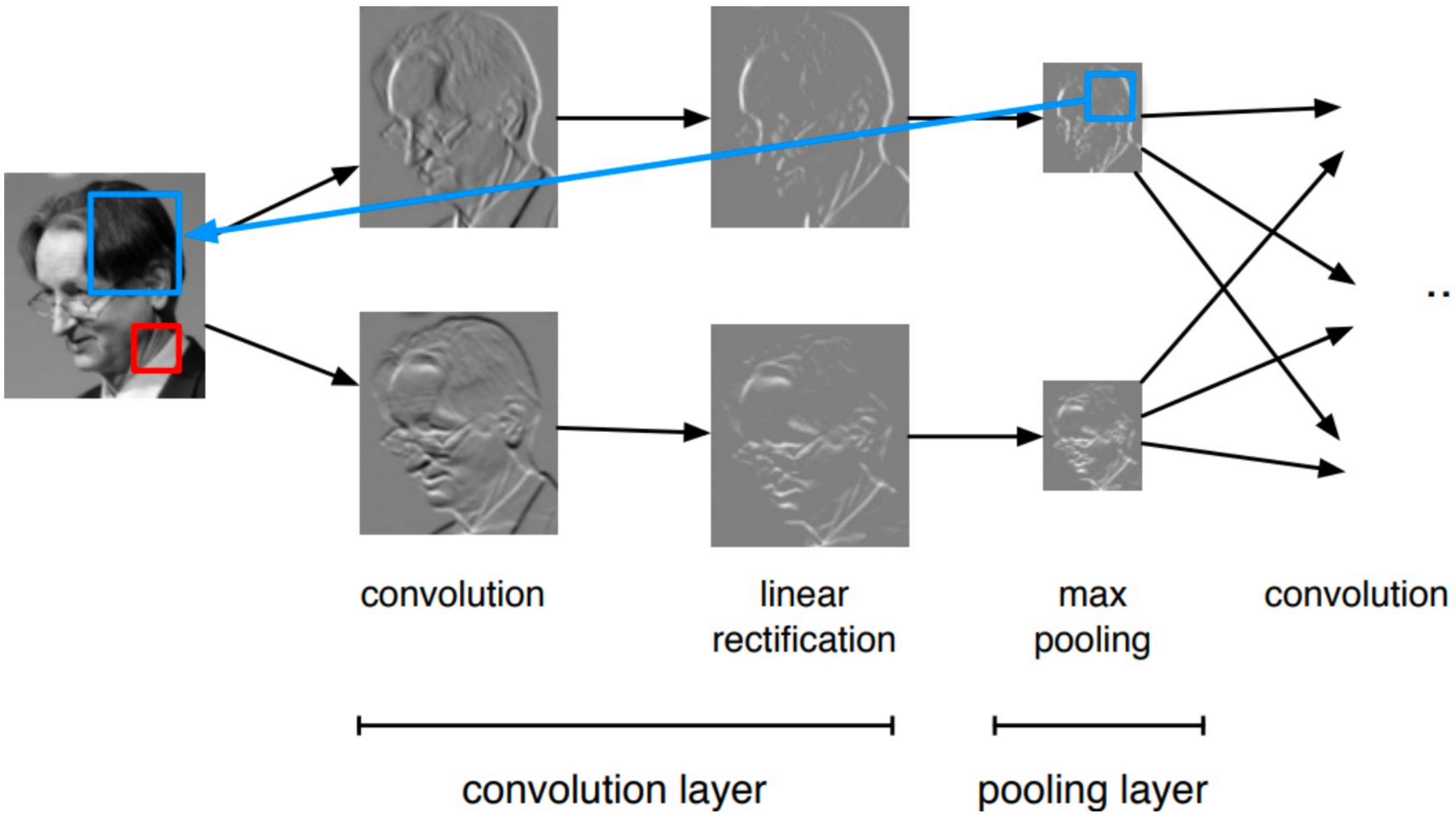
- Invariance to permutation within pooling area

- Invariance to local perturbations

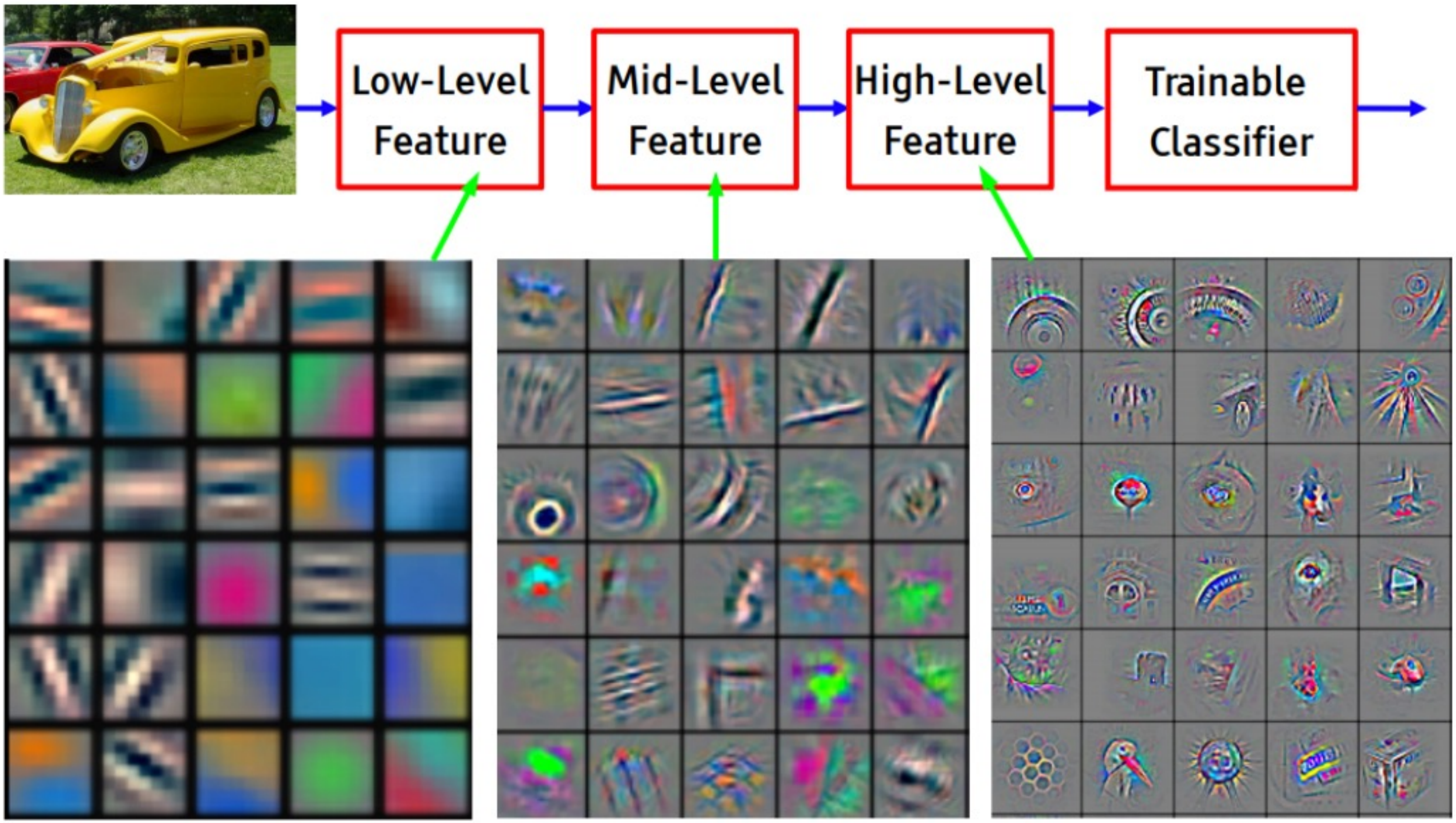


Convolutional Network

- A combination of convolution, pooling, ReLU, and fully connected layers

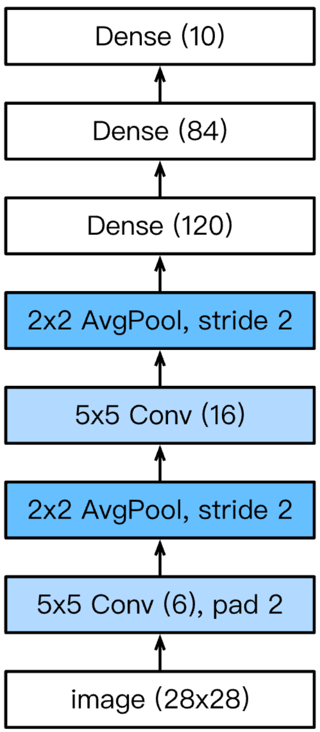


Hierarchical Composition of Features

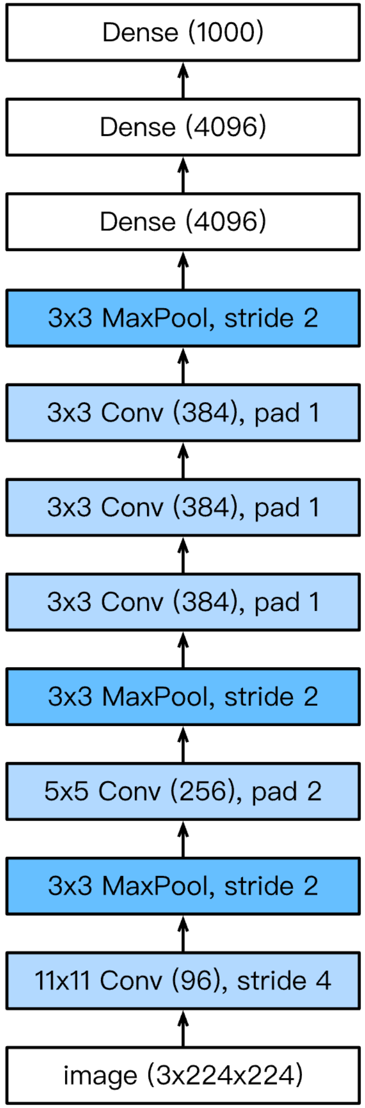


Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Convolutional Networks



LeNet
(LeCun et al, 1998)

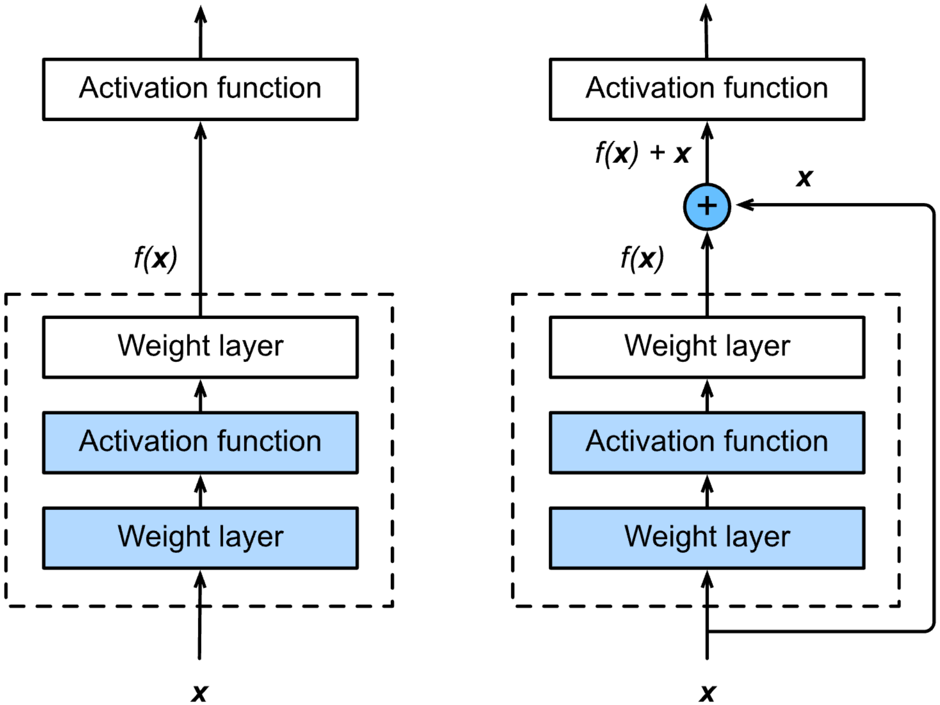


AlexNet
(Krizhevsky et al, 2012)

ImageNet Classification

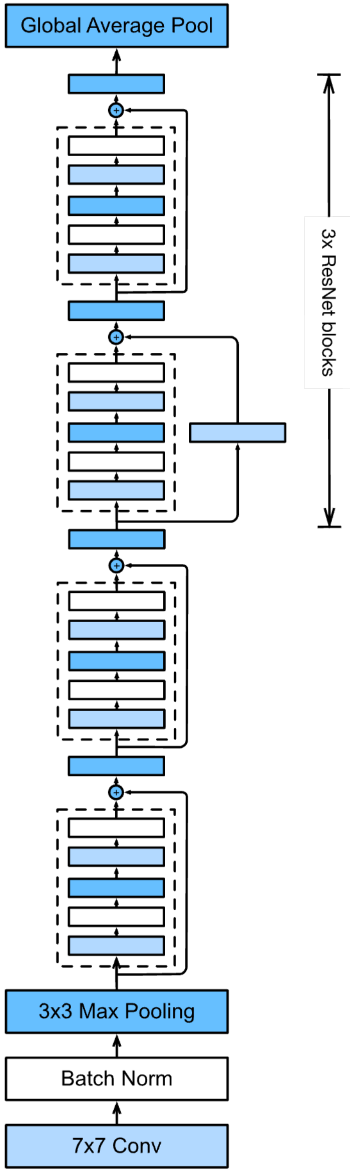
Residual Connections

- Training very deep networks is made possible because of the **skip connections** in the residual blocks. Gradients can shortcut the layers and pass through without vanishing.



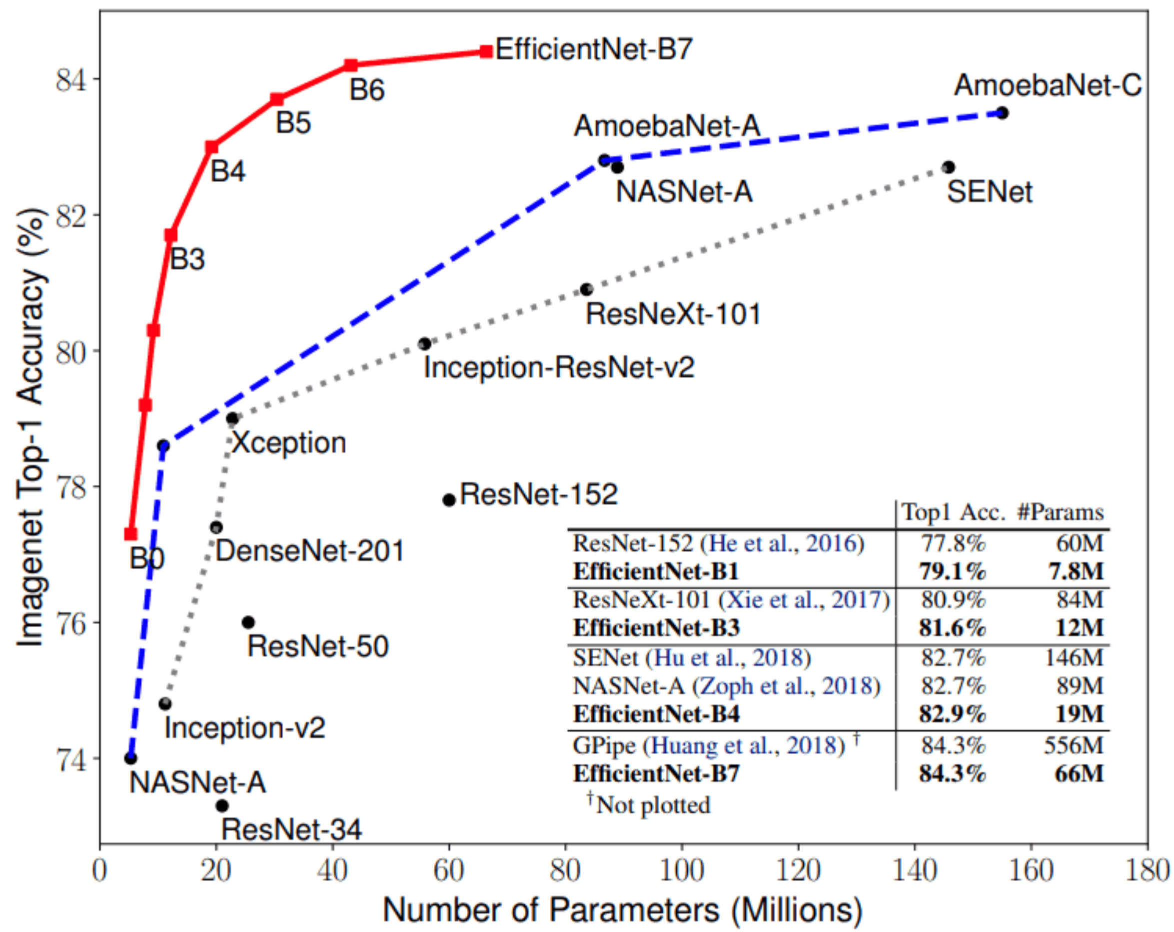
Credits: Deep Dive in Deep Learning , and G. Louppe

Deep CNNs



ResNet
(He et al, 2015)

1905.11946



Recurrent Neural Networks

Sequential Data

- Many types of data are not fixed in size
- Many types of data have a temporal or sequence-like structure
  - Text
  - Video
  - Speech
  - DNA
  - …
- MLP expects fixed size data
- How to deal with sequences?

Sequential Data

- Given a set $\mathcal{X}$, let $S(\mathcal{X})$ be the set of sequences, where each element of the sequence $x_i \in \mathcal{X}$
  - $\mathcal{X}$ could reals $\mathbb{R}^M$, integers $\mathbb{Z}^M$, etc.
  - Sample sequence $x = \{x_1, x_2, \dots, x_T\}$

- Tasks related to sequences:
  - Classification $f \colon S(\mathcal{X}) \rightarrow \{\boldsymbol{p} \mid \sum_{c=1}^{N} p_i = 1\}$
  - Generation $f \colon \mathbb{R}^d \rightarrow S(\mathcal{X})$
  - Seq.-to-seq. translation $f \colon S(\mathcal{X}) \rightarrow S(\mathcal{Y})$

Credit: F. Fleuret

Recurrent States

- Input sequence $x \in S(\mathbb{R}^m)$ of *variable* length $T(x)$

- Recurrent model maintain a **recurrent state** $\boldsymbol{h}_t \in \mathbb{R}^q$ updated at each time step t. For $t = 1, \dots, T(x)$:

$$\boldsymbol{h}_{t+1} = \phi(\boldsymbol{x}_t, \boldsymbol{h}_t; \theta)$$

  - Simplest model:

$$\phi(\boldsymbol{x}_t, \boldsymbol{h}_t; W, U) = \sigma(W\boldsymbol{x}_t + U\boldsymbol{h}_t)$$

- Predictions can be made at any time t from the recurrent state

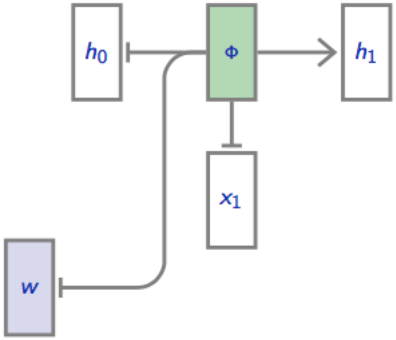
$$\boldsymbol{y}_t = \psi(\boldsymbol{h}_t; \theta)$$

Credit: F. Fleuret

Recurrent Neural Networks

Recurrent Model

$$\boldsymbol{h}_{t+1} = \phi(\boldsymbol{x}_t, \boldsymbol{h}_t; \theta)$$

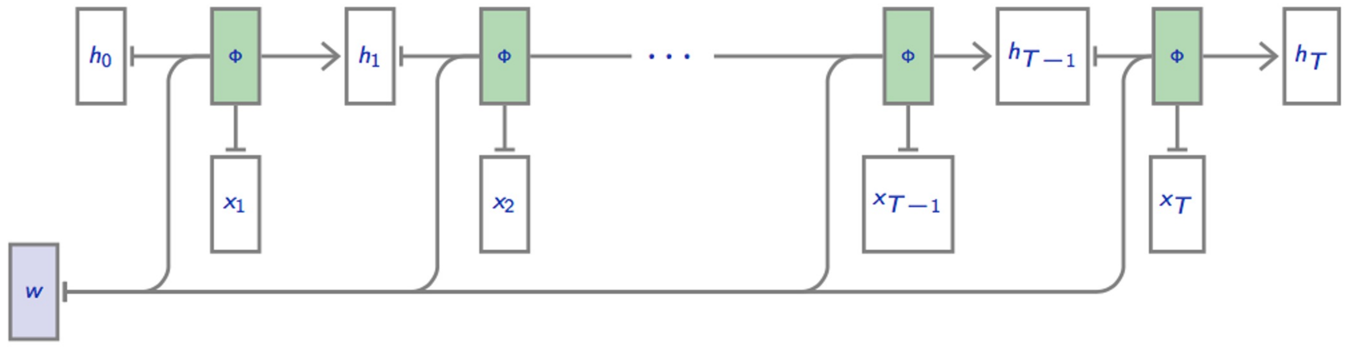


Credit: F. Fleuret

Recurrent Neural Networks

Recurrent Model

$$\boldsymbol{h}_{t+1} = \phi(\boldsymbol{x}_t, \boldsymbol{h}_t; \theta)$$

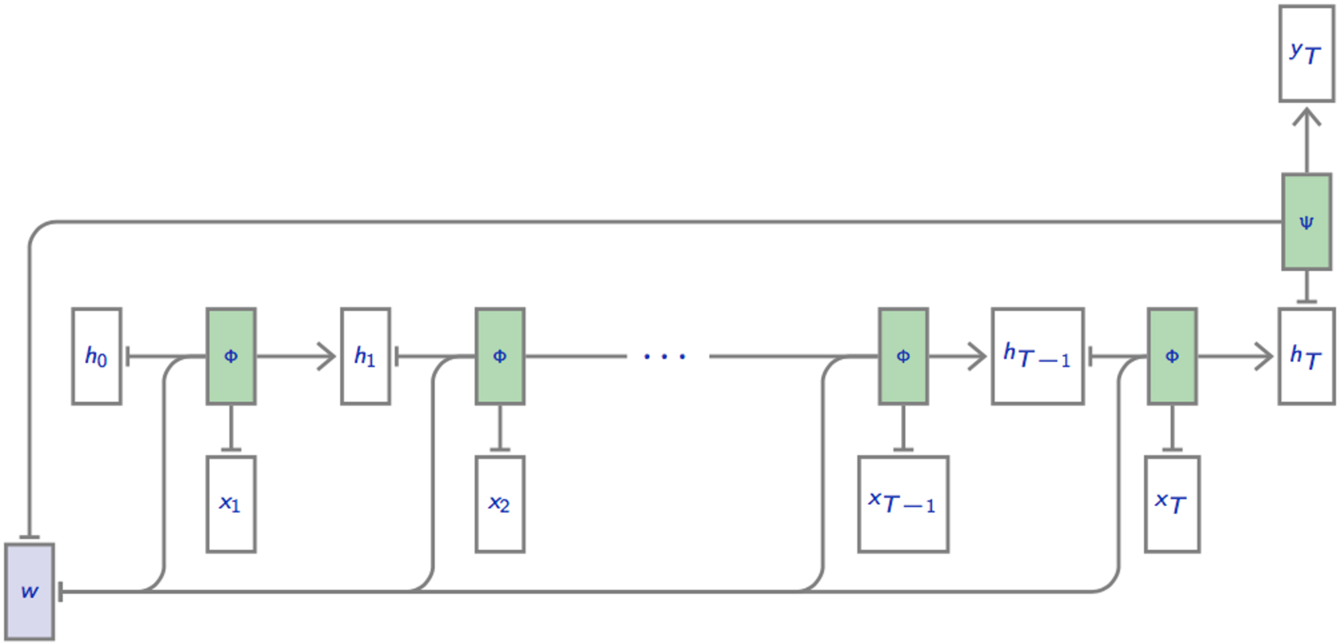


Credit: F. Fleuret

Recurrent Neural Networks

Prediction

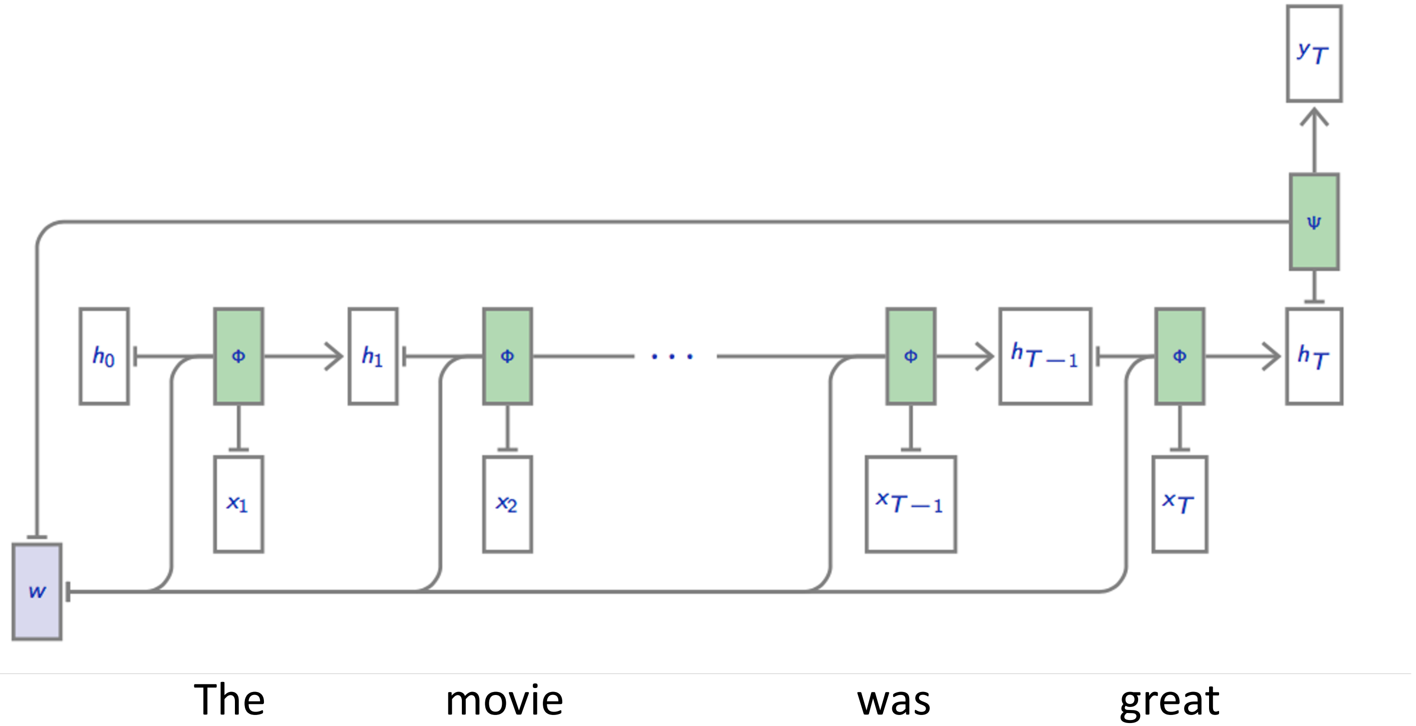
$$\boldsymbol{y}_t = \psi(\boldsymbol{h}_t; \theta)$$



Credit: [F. Fleuret]()

Recurrent Neural Networks

[0.98] → Positive Sentiment

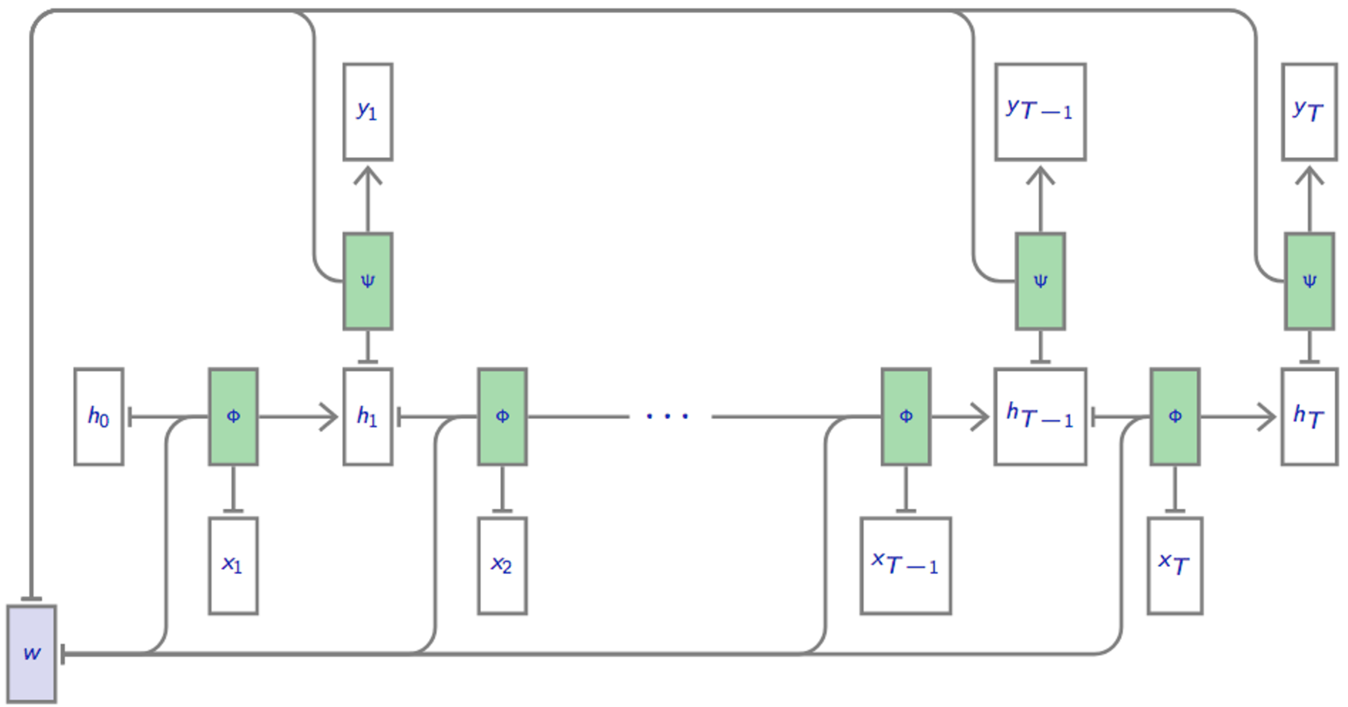


Sentiment Analysis

Credit: F. Fleuret

Recurrent Neural Networks

Prediction per sequence element

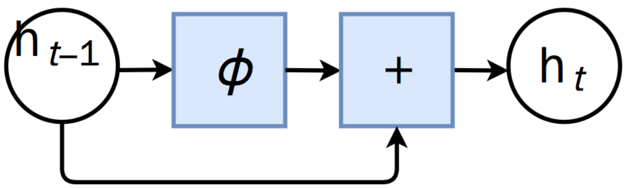


Although the number of steps $T(x)$ depends on x, this is a standard computational graph and automatic differentiation can deal with it as usual. This is known as “backpropagation through time” (Werbos, [1988](#))

Credit: [F. Fleuret](#)

Gating

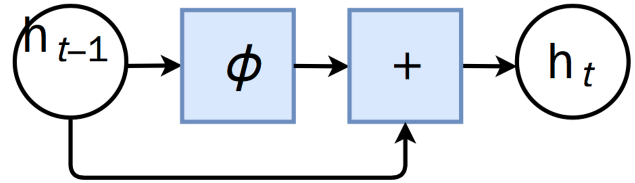
- Gating:
  - network can grow very deep, in time → vanishing gradients.
  - *Critical component*: add pass-through (additive paths) so recurrent state does not go repeatedly through squashing non-linearity.



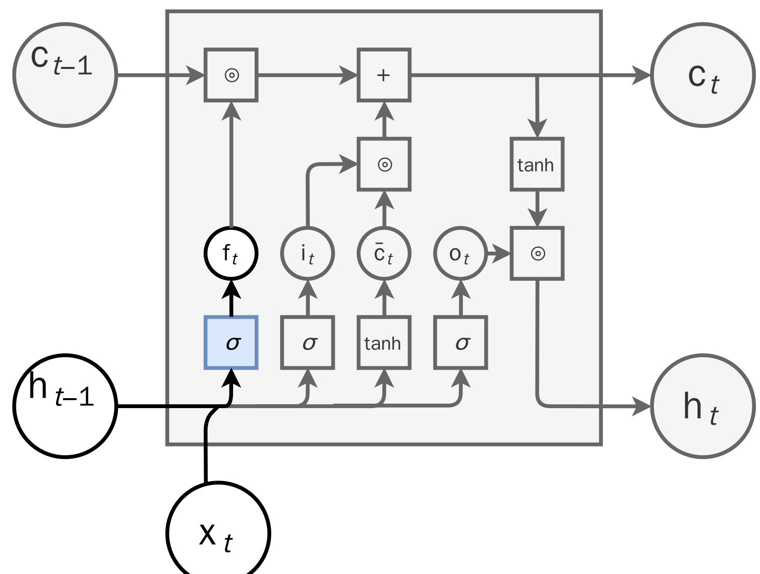
Credit: Gilles Louppe

Long Short Term Memory (LSTM)

- Gating:
  - network can grow very deep, in time → vanishing gradients.
  - *Critical component*: add pass-through (additive paths) so recurrent state does not go repeatedly through squashing non-linearity.



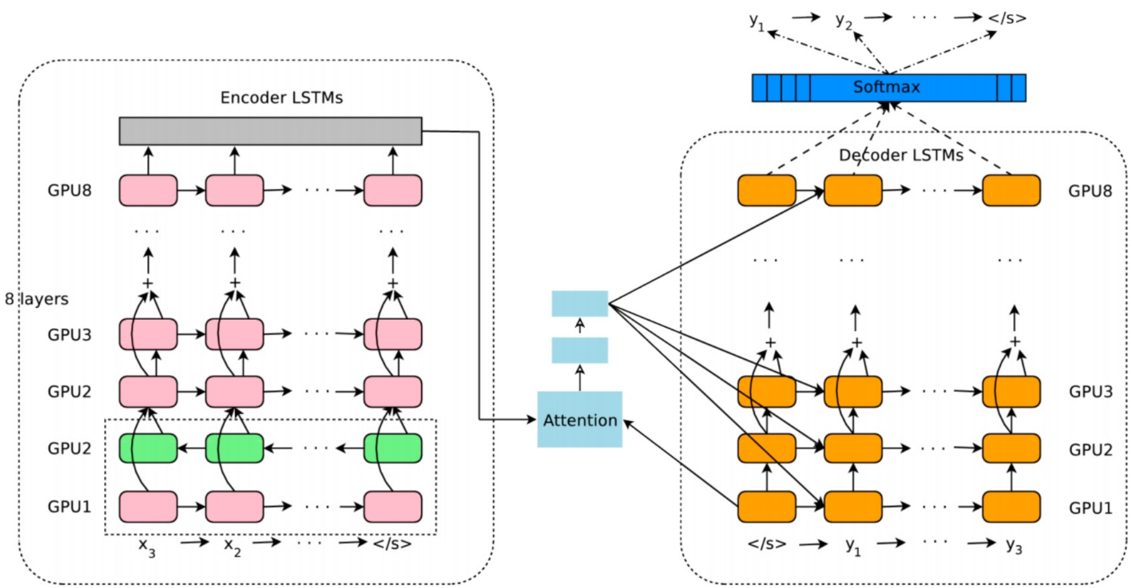
- LSTM:
  - Add internal state separate from output state
  - Add input, output, and forget gating

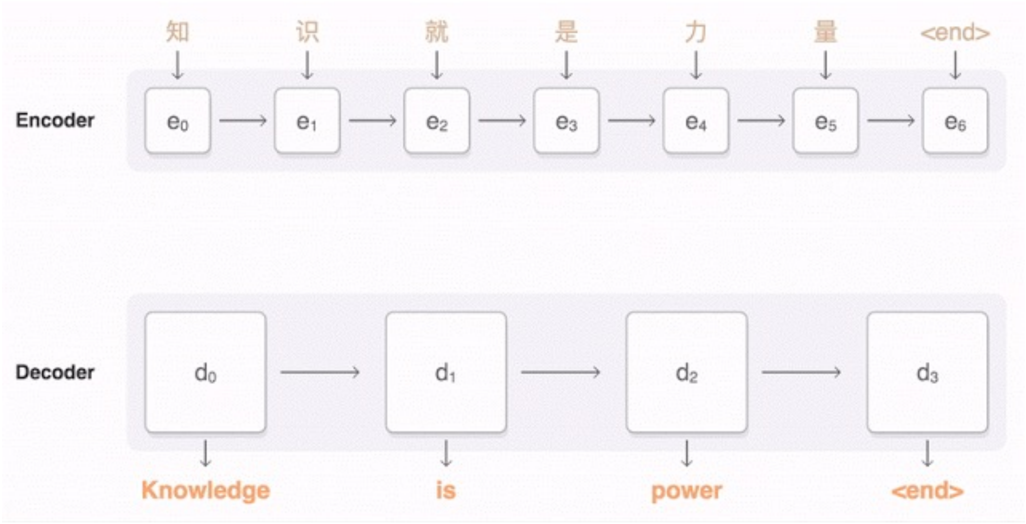


Credit: Gilles Louppe

Examples

Neural machine translation

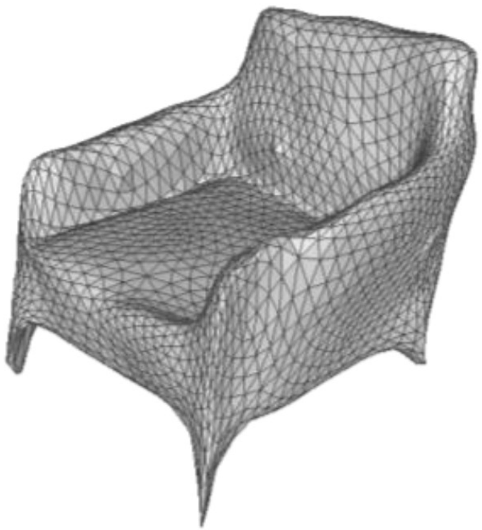




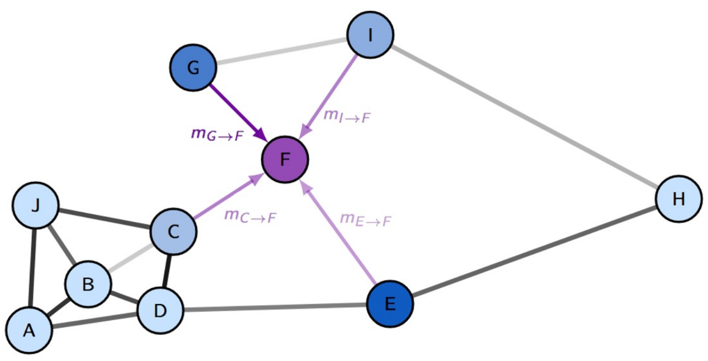
Y. Wu et al, 2016

Many Other Architectures: Graph Neural Networks

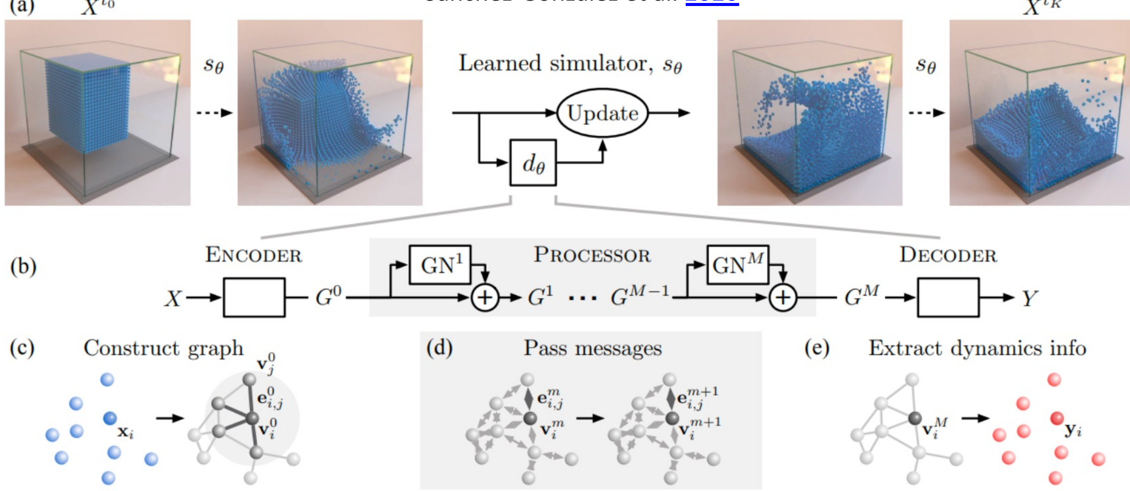
- Permutation invariant data with geometric relationships
  - Features can be local on graph, but meaningful anywhere on graph



- Graph layers can encode these relationships on nodes & edges



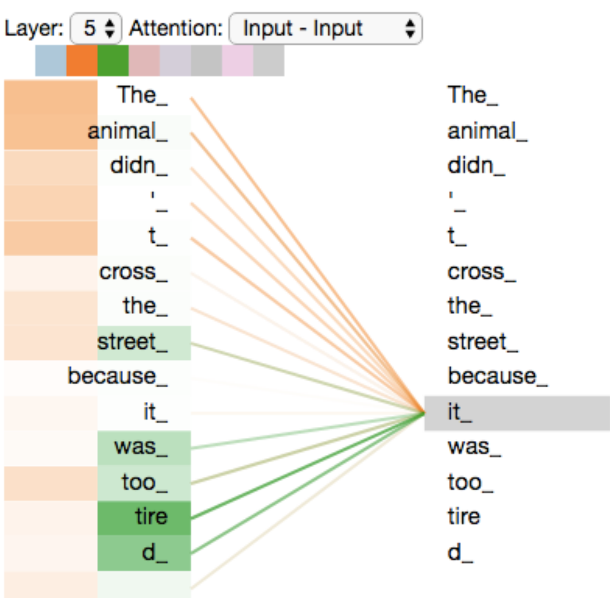
Sanchez-Gonzalez et al. [2020]



Many Other Architectures: Transformers & Deep Sets

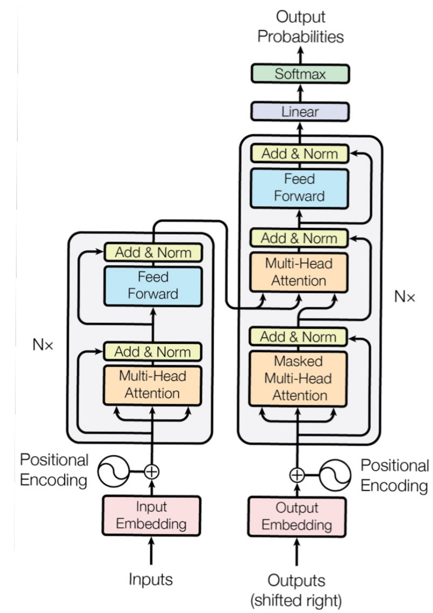
- **Deep Sets** and **Transformers** can process permutation invariant sets of data

- *Transformers are very adaptable*:
  Built using layers of ***attention***, they can also process sequences, images, and other data



Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com



Beyond Regression and Classification

Beyond Regression and Classification

- Not all tasks are predicting a label from features, as in classification and regression

- May want to model a high-dim. signal
  - Data synthesis / simulation
  - Density estimation
  - Anomaly detection
  - Denoising, super resolution
  - Data compression
  - …

- Often don't have labels → Unsupervised Learning

Unsupervised Learning

- Our goal is to study the data density $p(x)$
- Even w/o labels, aim to characterize the distribution

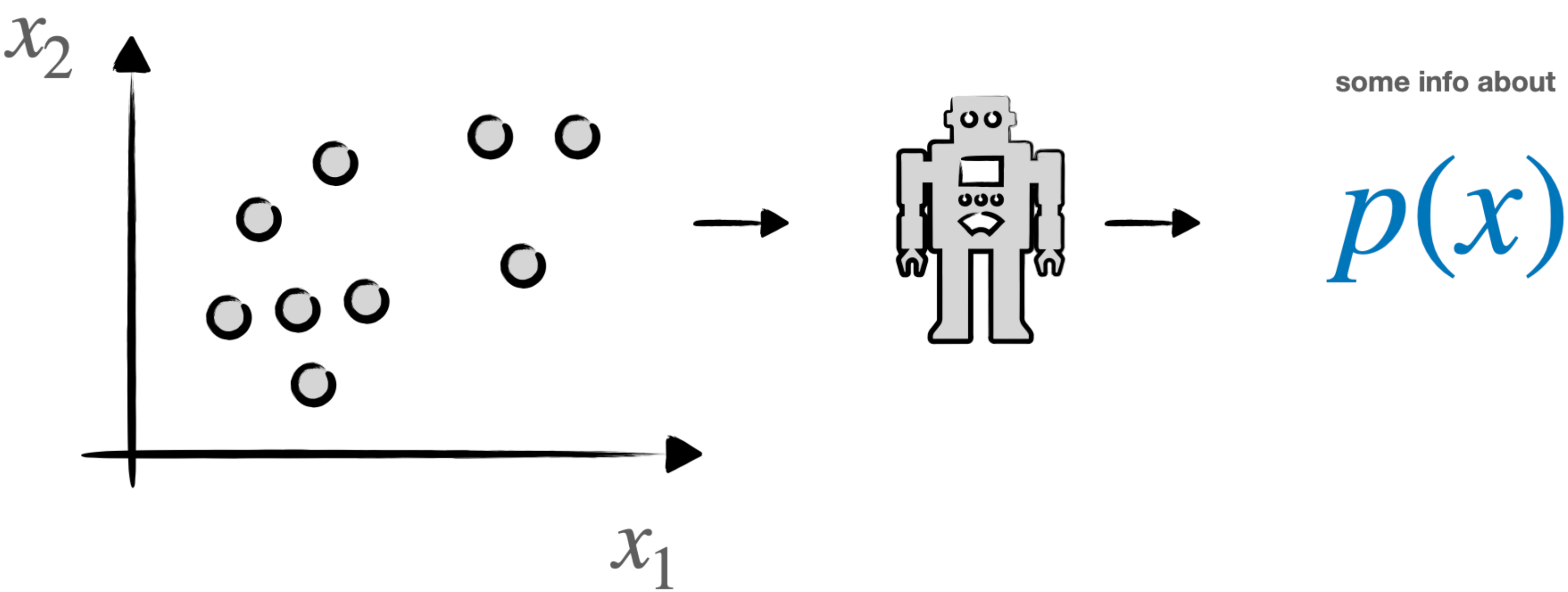


Image credit: L. Heinrich

Probability Models

A process

$\rightarrow \mathbb{R}^2$

Generating new samples from randomness

A formula

$\mathbb{R}^2 \rightarrow \mathbb{R}$

$$p_{\mu,\Sigma}(x) = \frac{1}{\sqrt{|2\pi\Sigma|}} exp\left(-\frac{1}{2}(x-\mu)^T\Sigma(x-\mu)\right)$$

Evaluating the Probability for a given sample

"Understanding $p(x)$" – ability to do either or both of these

Image credit: L. Heinrich

Probability Models as Sampling a Process

- In many cases, we don't have a theory of the underlying process → *Can still learn to sample*

- Deep learning can be very good at this!

$$\text{face} \sim p(\text{face})$$

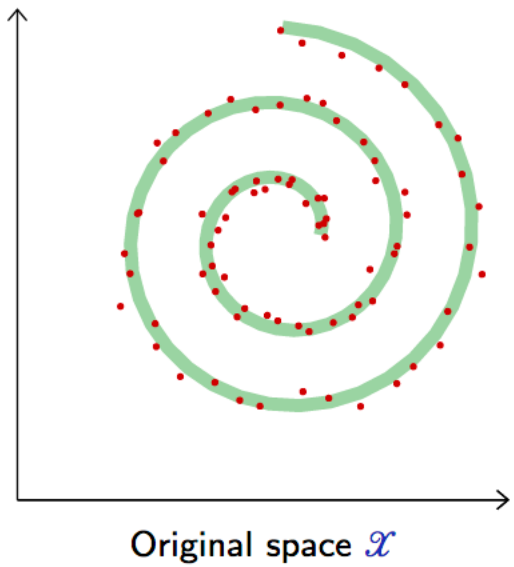


https://thispersondoesnotexist.com/

Learning Objective

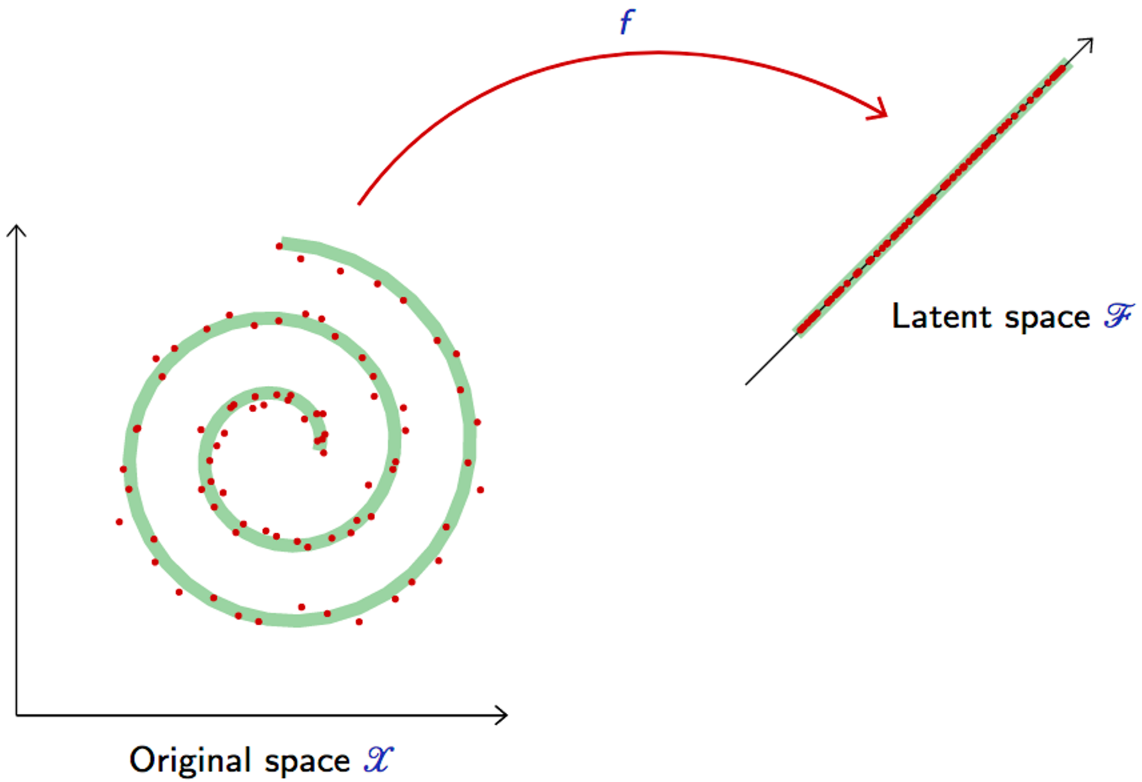
- Unsupervised learning is more heterogeneous than supervised learning
- Many architectures, losses, learning strategies
- Often constructed so model converges to $p(x)$
  - Variational inference, Adversarial learning, Self-supervision, …
- Often framed as **modeling the lower dimensional "meaningful degrees of freedom"** that describe the data

Modeling Data and Meaningful Degrees of Freedom



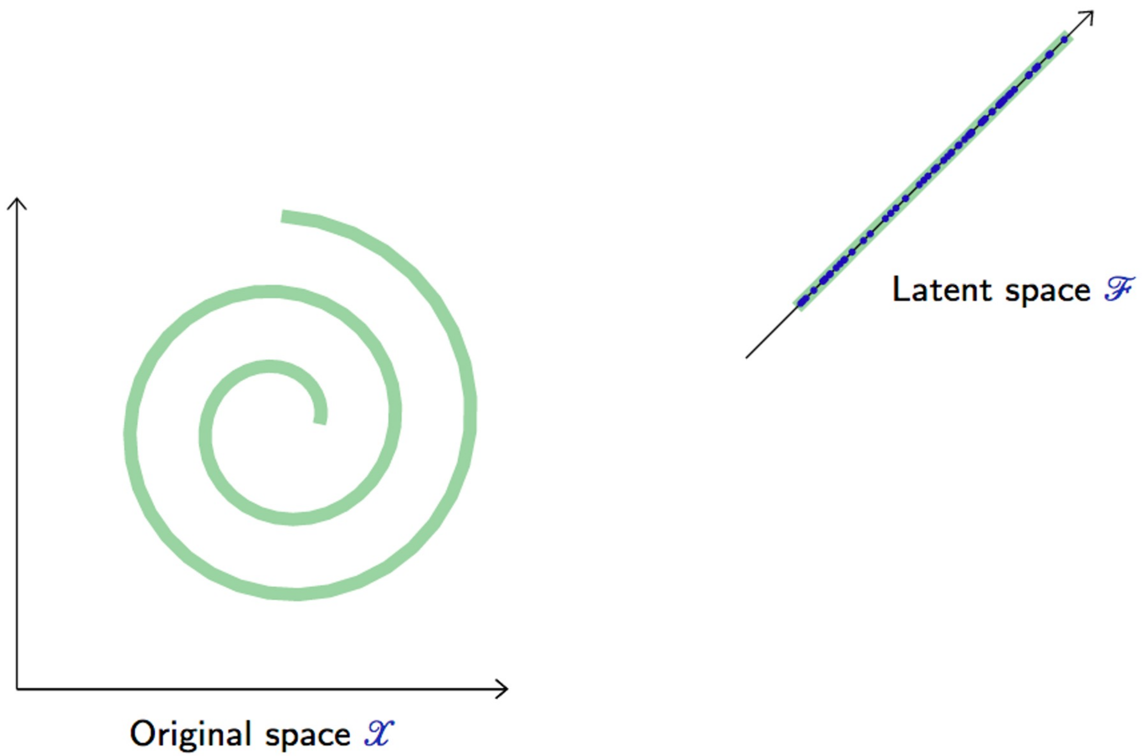
Fleuret, Deep Learning Course

Modeling Data and Meaningful Degrees of Freedom



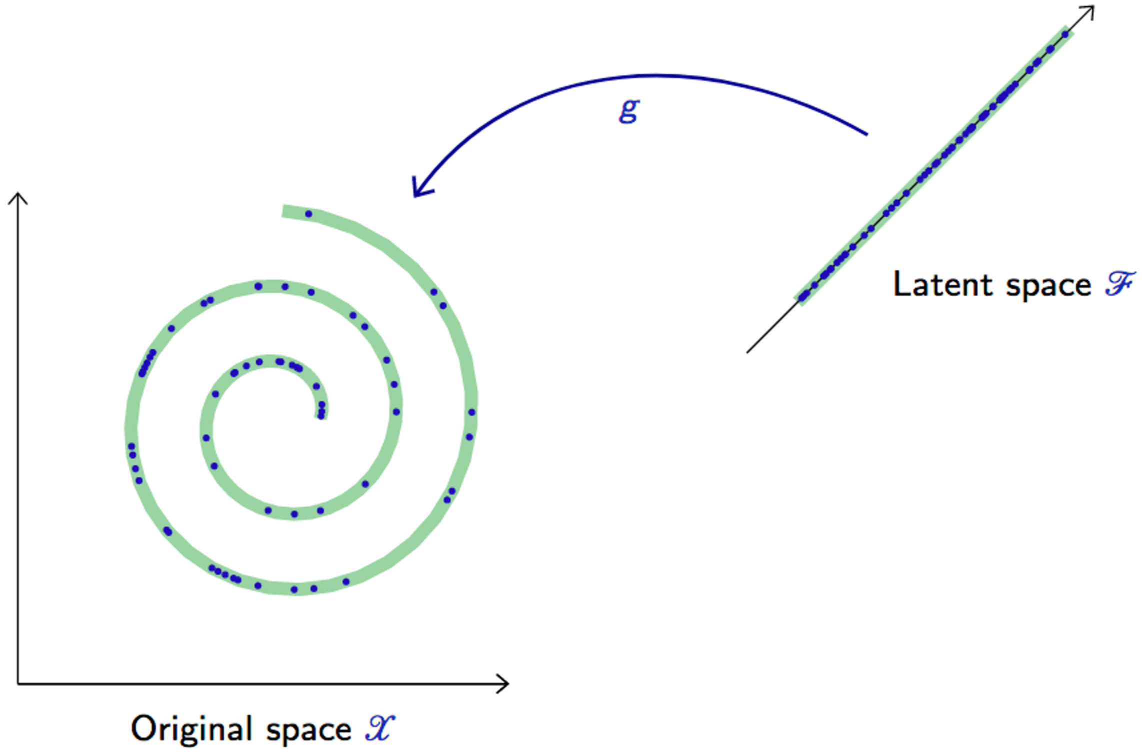
Fleuret, [Deep Learning Course]

Modeling Data and Meaningful Degrees of Freedom



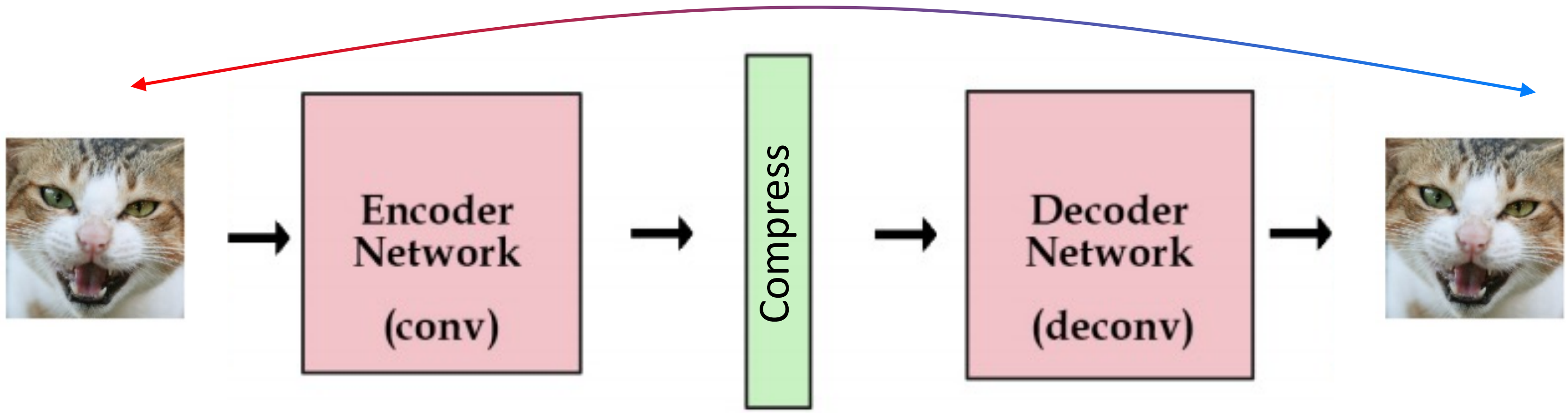
Fleuret, Deep Learning Course

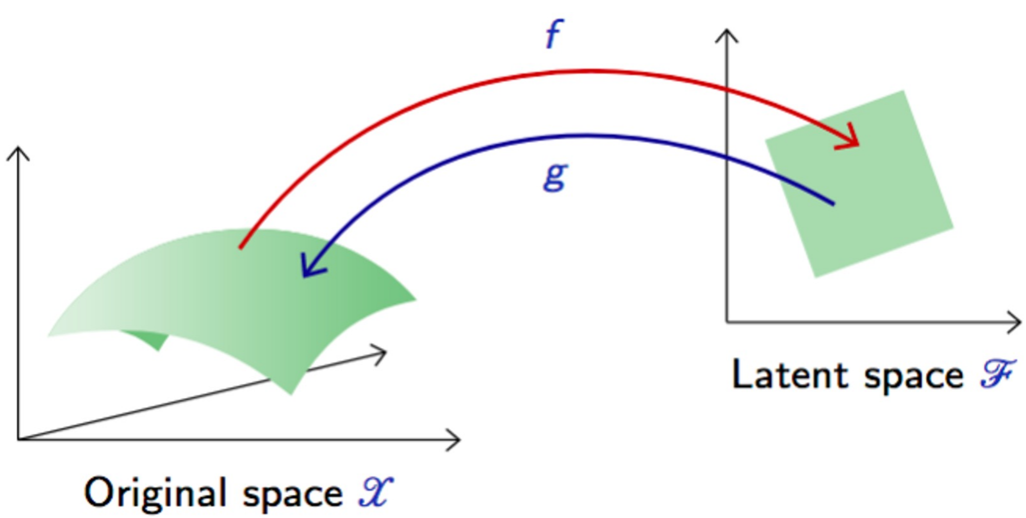
Modeling Data and Meaningful Degrees of Freedom



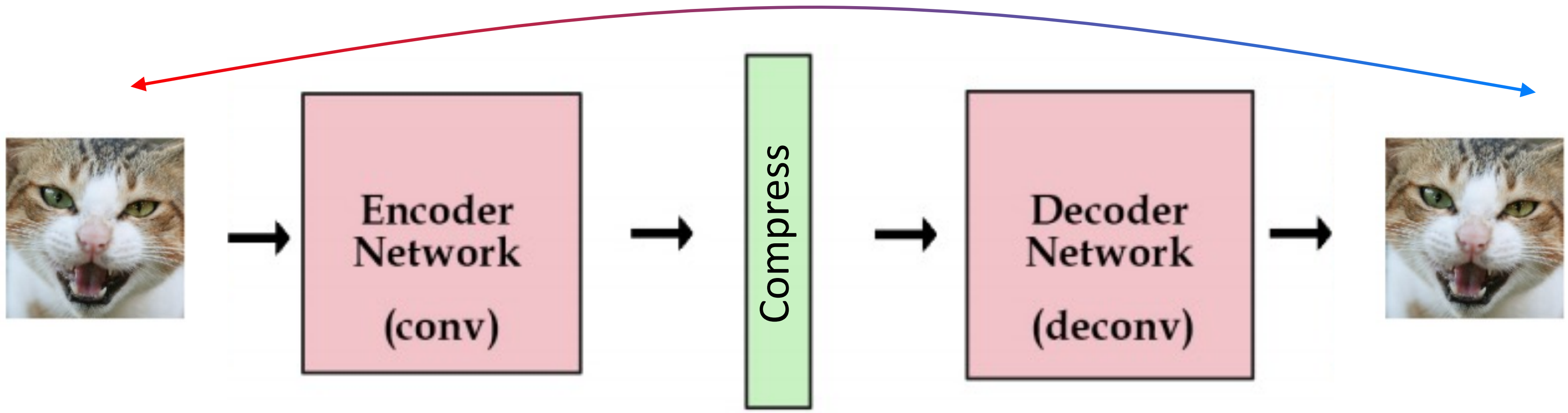
Fleuret, Deep Learning Course

AutoEncoders

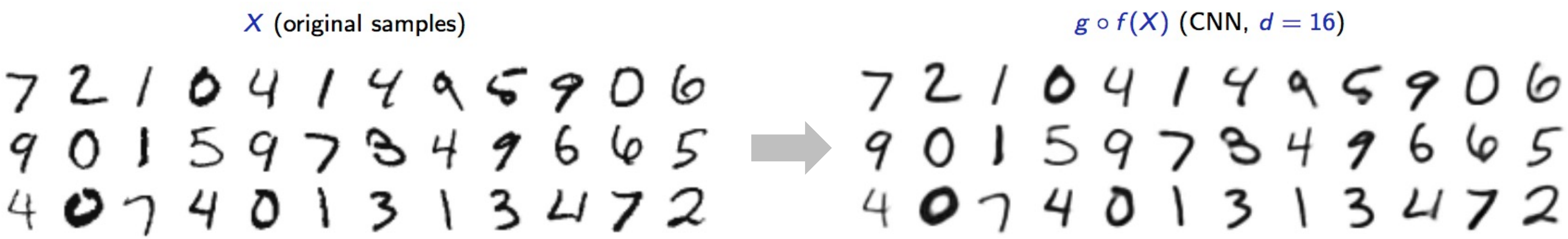




AutoEncoders

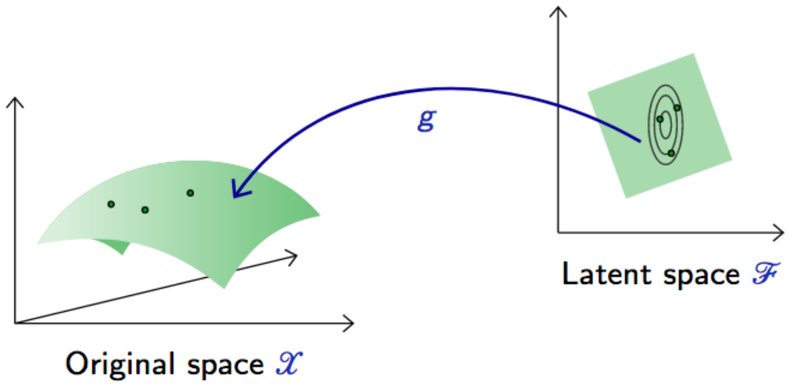


$$L(\theta, \psi) = \frac{1}{N} \sum_n \left\| x_n - g_\psi(f_\theta(x_n)) \right\|^2$$



Can We Generate Data with Decoder?

- Can we sample in latent space and decode to generate data?



- What distribution to sample from in latent space?
  - Try Gaussian with mean and variance from data

Autoencoder sampling ($d = 16$)

- Don't know the right latent space density

Fleuret, [Deep Learning Course]

Generative Models Goal

A generative model is a probabilistic model q that can be used as a simulator of the data.

Goal: generate synthetic, realistic high-dimensional data

$$x \sim q(x; \theta)$$

that is as close as possible to the unknown data distribution $p(x)$ for which we have empirical samples.

i.e. want to recreate the raw data distribution (such as the distribution of natural images).

Generative Models Are Everywhere These Days

ChatGPT

Stable Diffusion

GitHub Copilot

Prompt:

street style photo of a woman selling pho at a Vietnamese street market, sunset, shot on fujifilm

Deep Generative Model Examples in Science

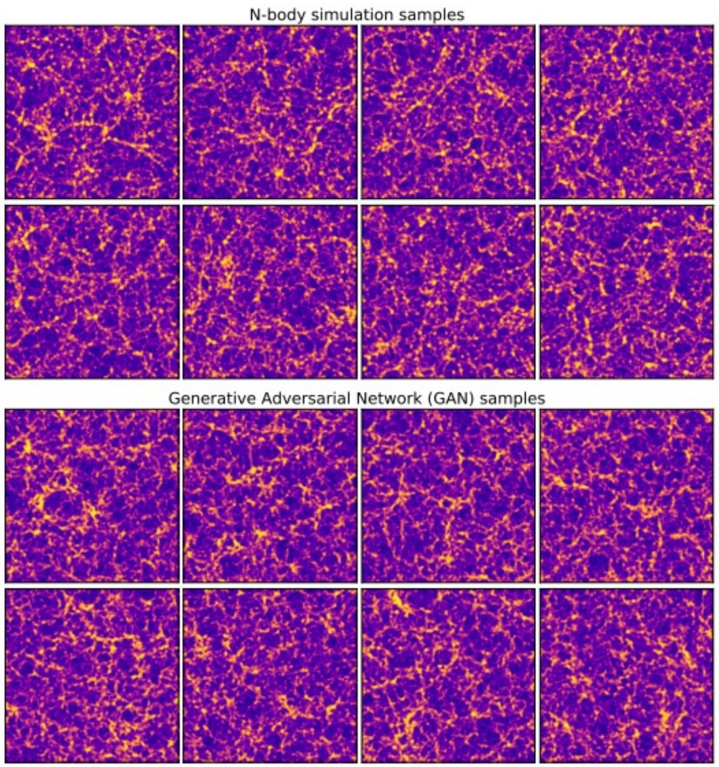
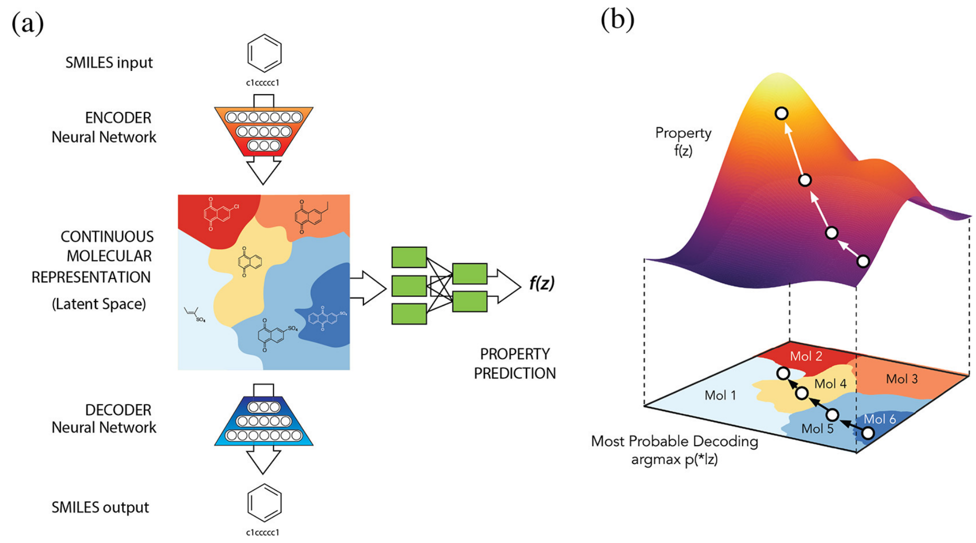
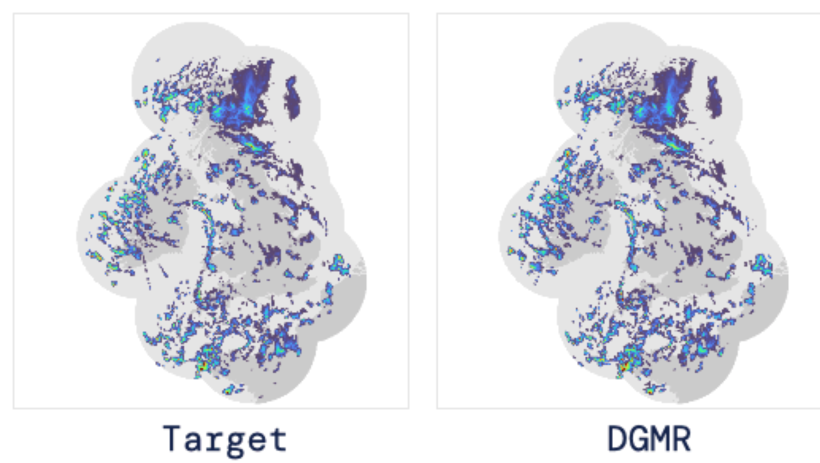


Figure 1: Samples from N-body simulation and from GAN for the box size of 500 Mpc. Note that the transformation in Equation 3.1 with $a = 20$ was applied to the images shown above for better clarity.

Learning cosmological models
(Rodriguez et al, 2018)



Design of new molecules with desired chemical properties.
(Gomez-Bombarelli et al, 2016)



Deep Generative Model of Rainfall (Ravuri et. al. 2021)

What Deep Generative Models Are There?

GAN: Adversarial training

VAE: maximize variational lower bound

Flow-based models: Invertible transform of distributions

Diffusion models: Gradually add Gaussian noise and then reverse

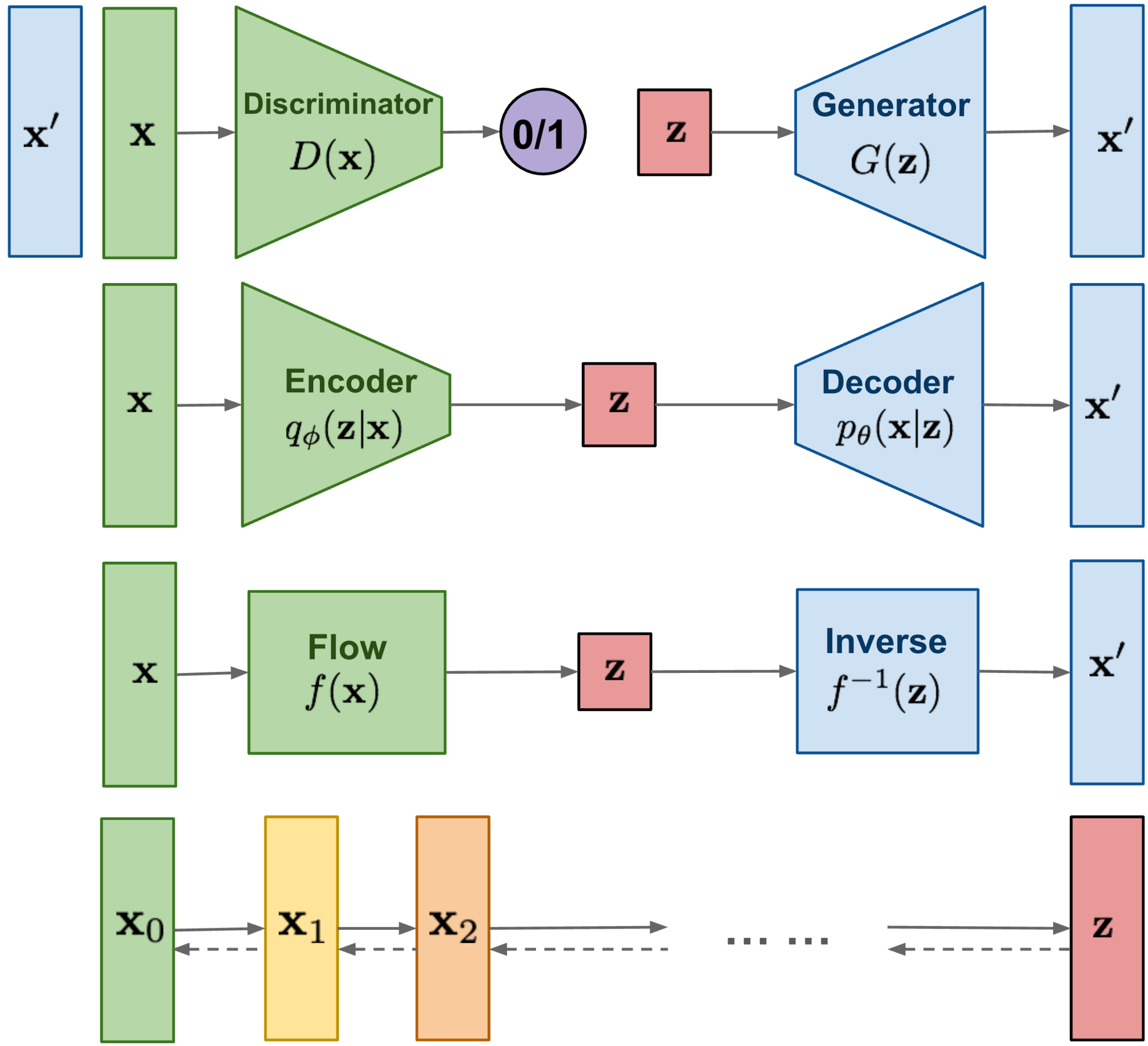
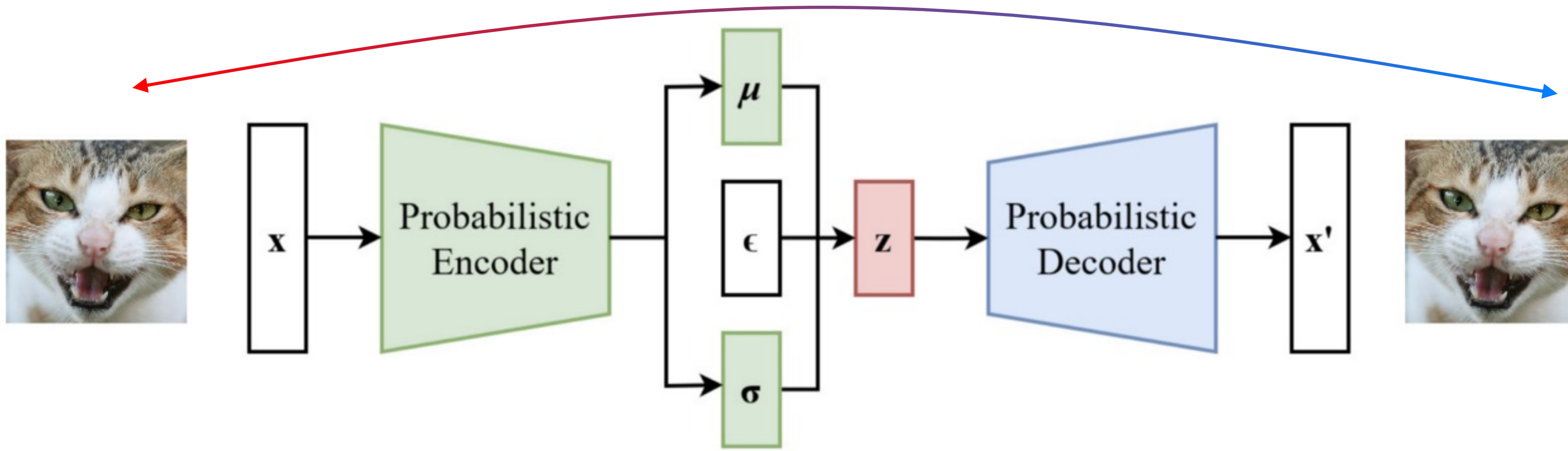
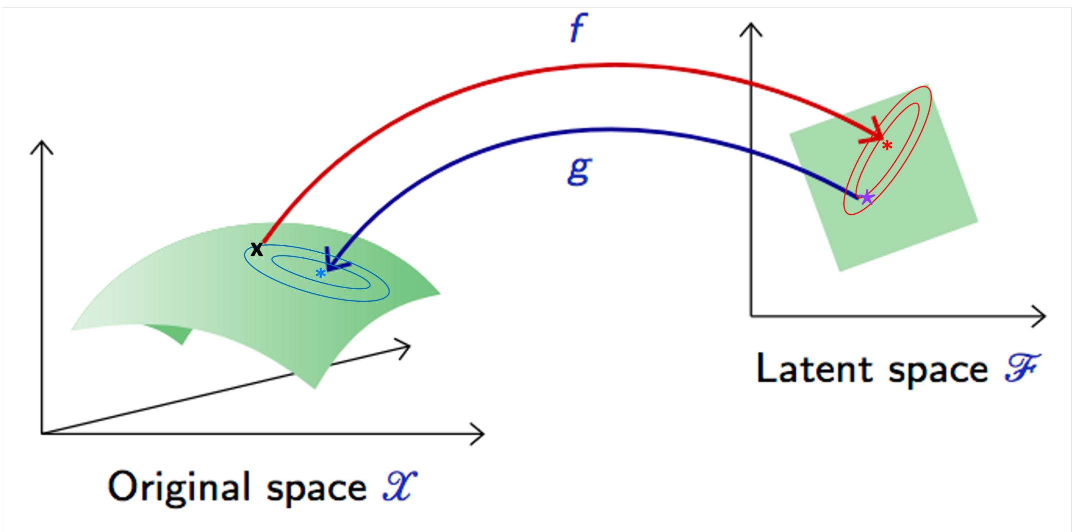


Image from [Lilian Weng]

Variational AutoEncoders

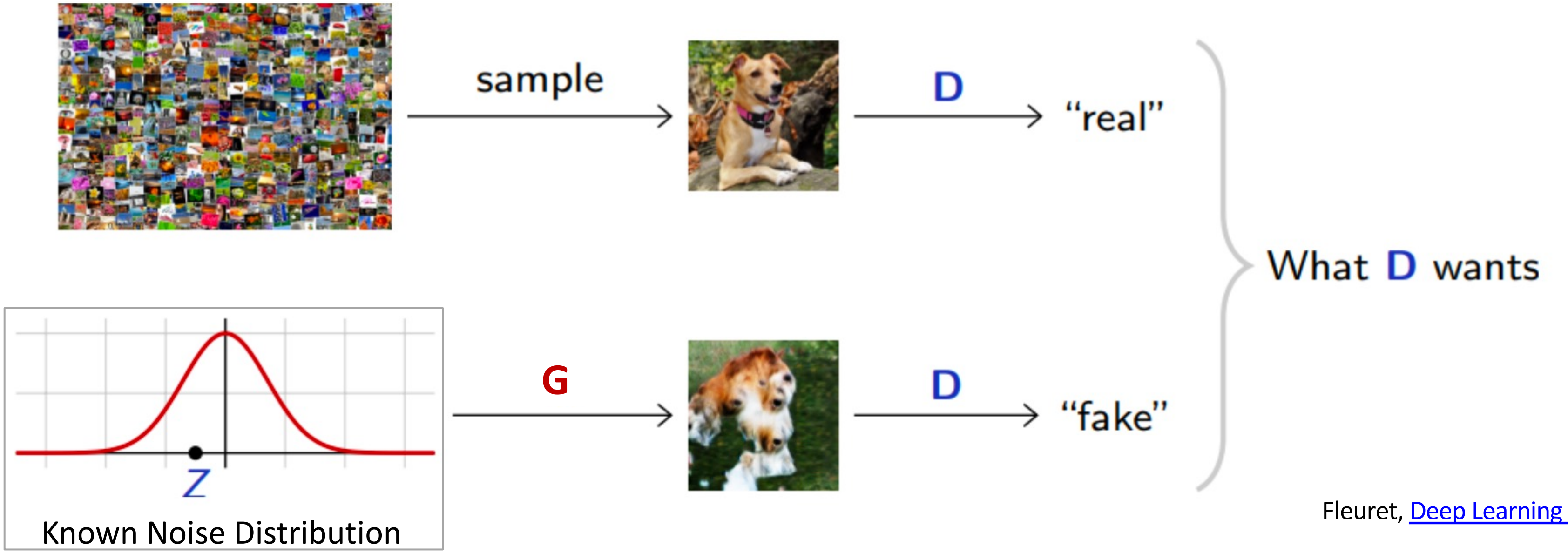




Choose known distribution for latent space and learn map to data space

Generative Adversarial Networks (GAN)

Goodfellow et. al., 2014



Fleuret, Deep Learning Course

- Generator creates data from noise, trained to trick Discriminator that classifies data as real or fake

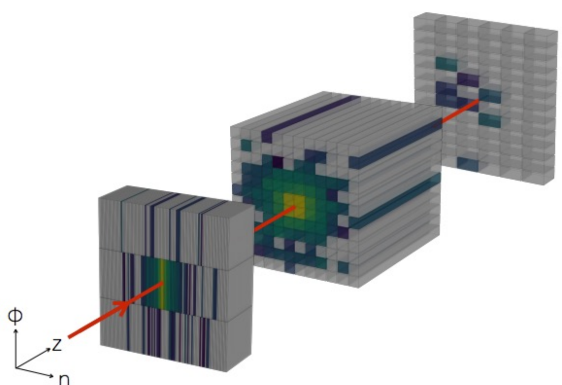
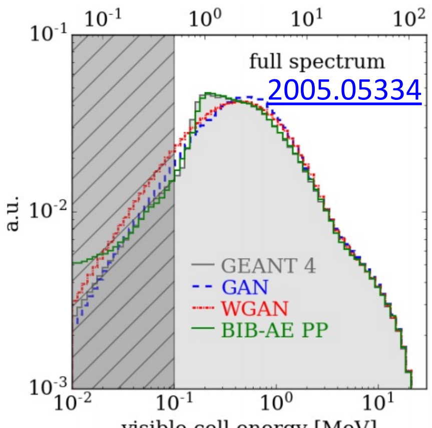
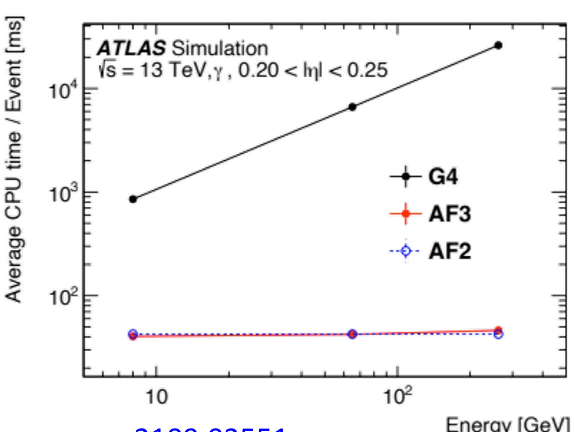


Image credit: 1705.02355

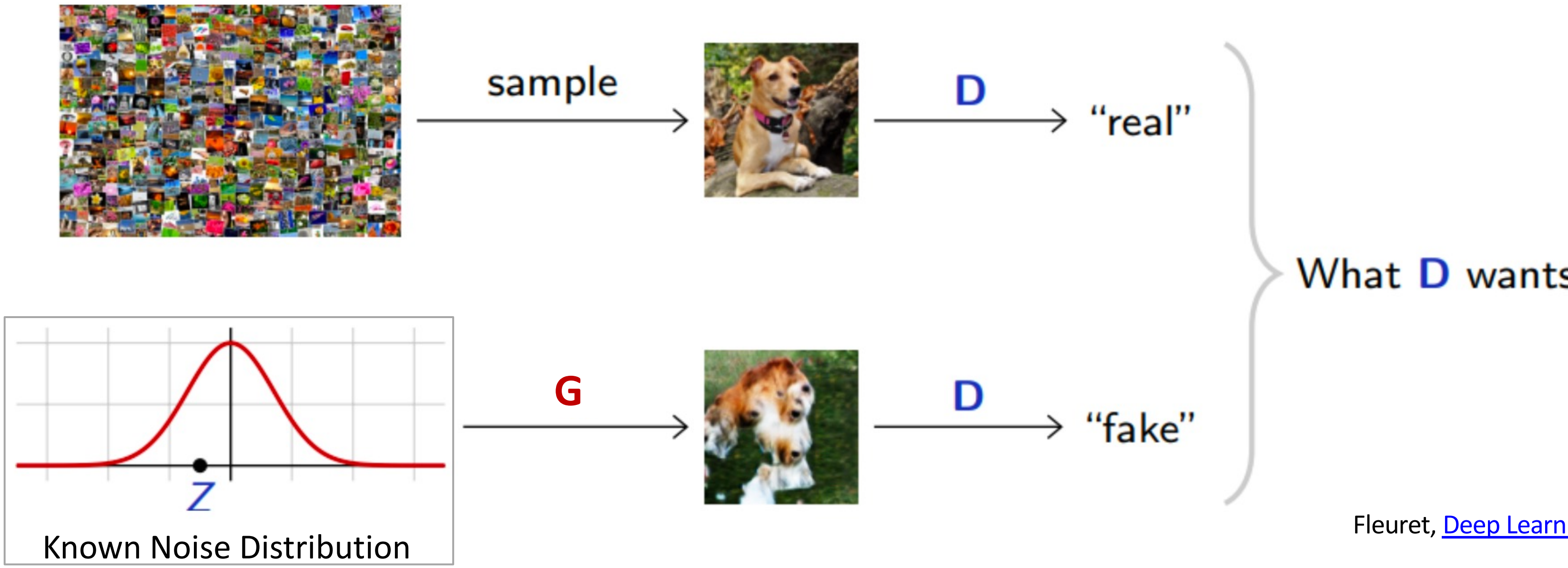




2109.02551

Generative Adversarial Networks (GAN)

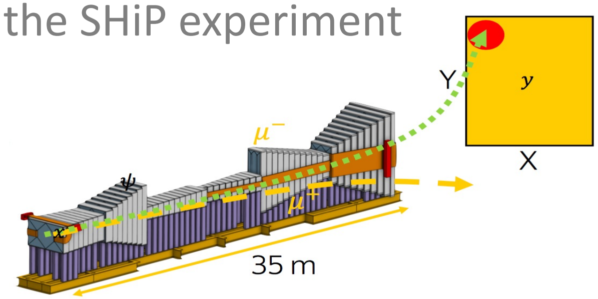
Goodfellow et. al., [2014]

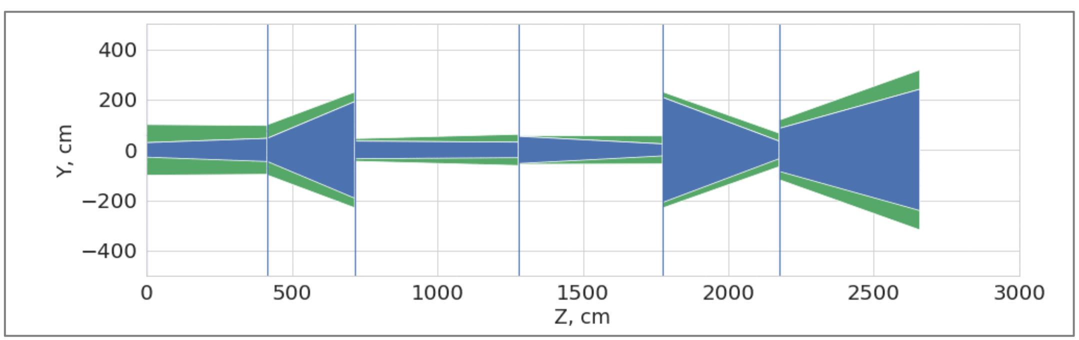


Fleuret, Deep Learning Course

- Generator creates data from noise, trained to trick Discriminator that classifies data as real or fake

Optimization of the magnet system
For the SHiP experiment





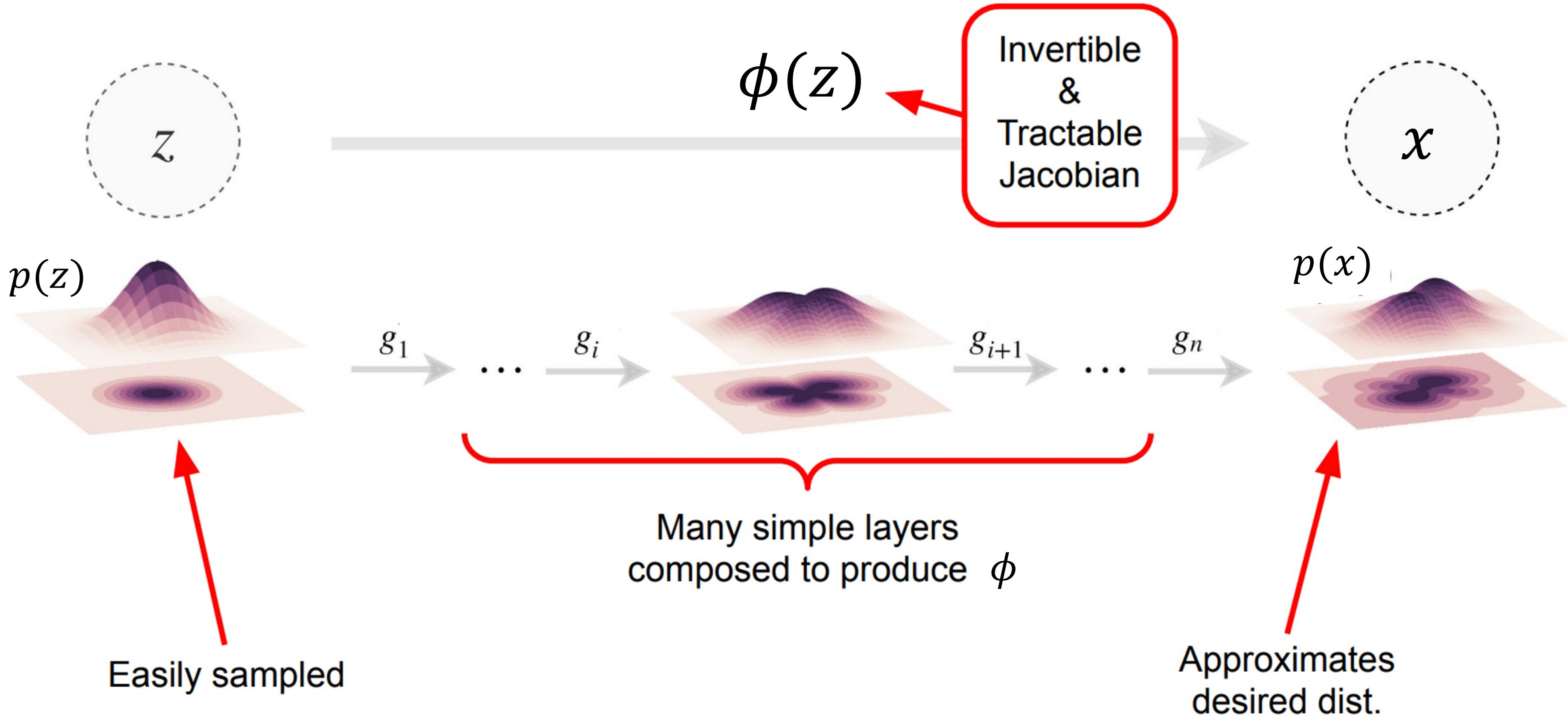
Shirbokov, **MK**, et al., NeurIPS **33**, 14650-14662 (2020)

Normalizing Flows

Explicit density estimation

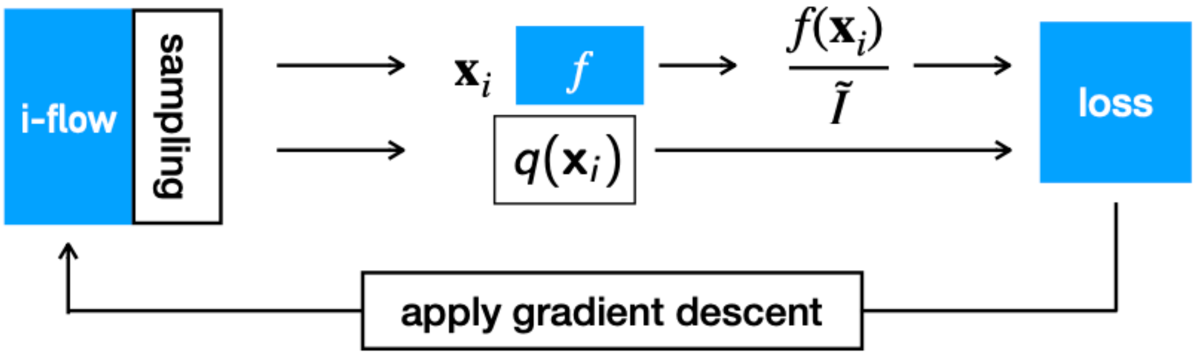
We can evaluate density $p(x)$

$$p_x(\boldsymbol{x}) = p_z(\boldsymbol{z}) \left| \det\left(\frac{\partial \phi(\boldsymbol{z})}{d\boldsymbol{z}}\right)^{-1} \right|$$



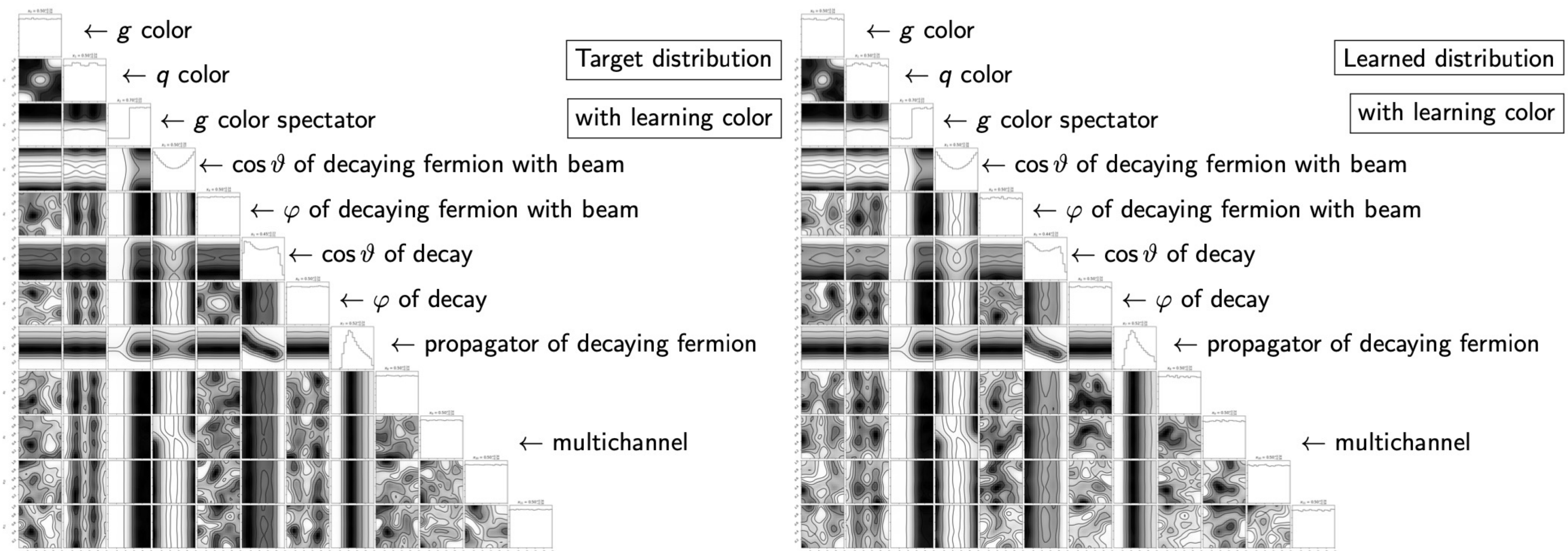
Slide credit: G. Kanwar

Event Generation with Normalizing Flows



arXiv: 2001.05486, ML:ST
arXiv: 2001.10028, PRD
Slide credit: C. Krause

Example: Learning $e^+e^- \to 3j$



Diffusion Models

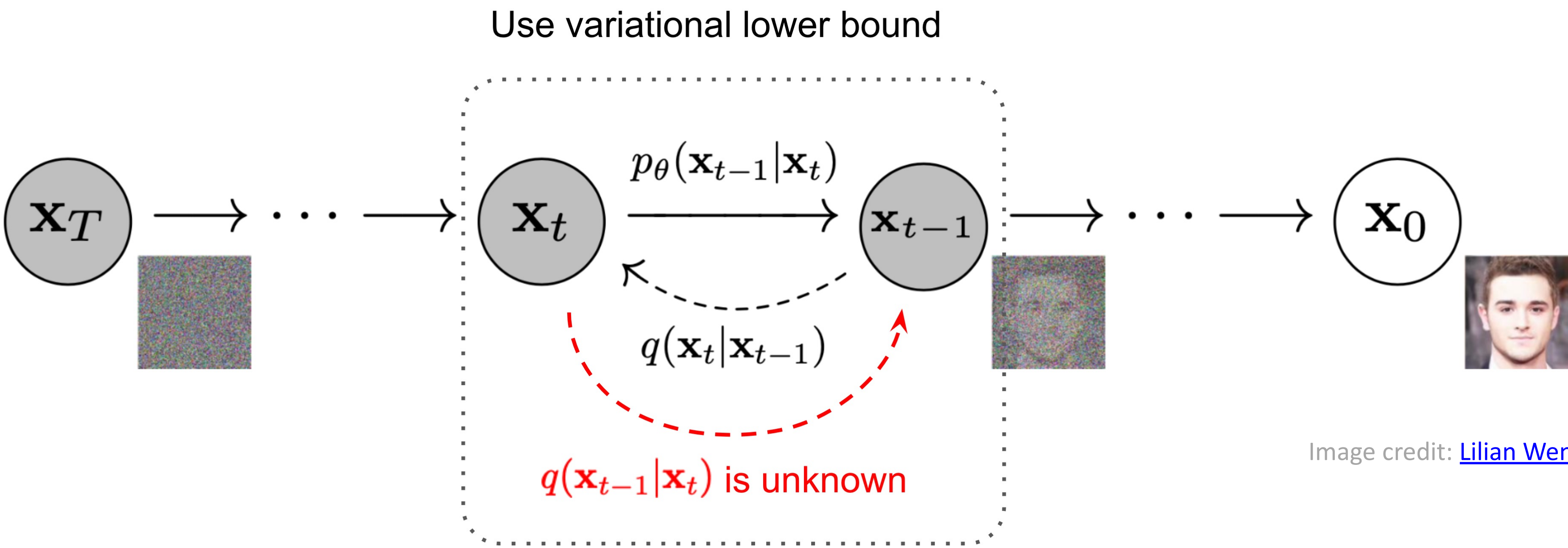
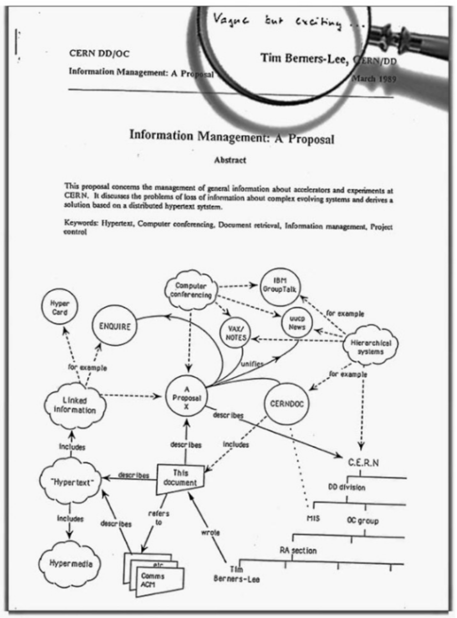


Image credit: [Lilian Weng]

- Iteratively add noise to data,
  Train model to learn how to denoise step by step

Wrapping Up

Since Tim Berners-Lee Invented the World Wide Web…


Vague but exciting …

CERN DD/OC

Information Management: A Proposal

Tim Berners-Lee, CERN/DD

March 1989

Information Management: A Proposal

Abstract

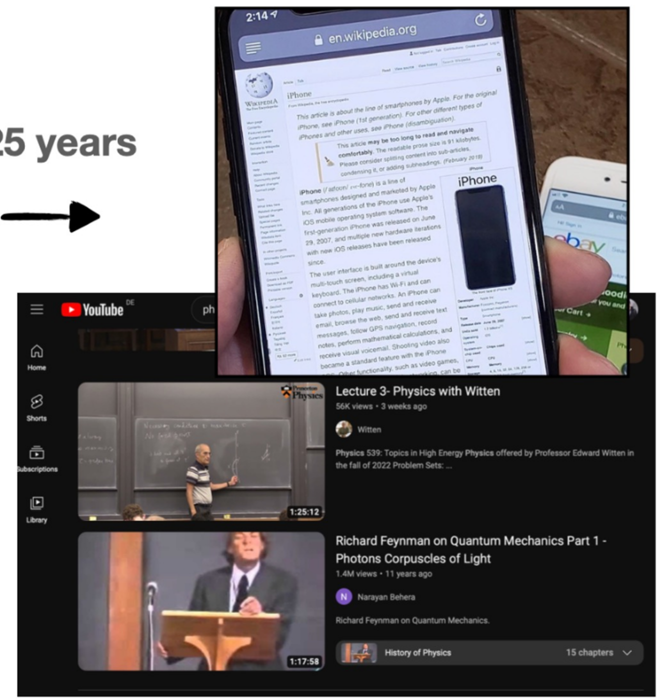
This proposal concerns the management of general information about accelerators and experiments at CERN. It discusses the problems of loss of information about complex evolving systems and derives a solution based on a distributed hypertext sytstem.

Keywords: Hypertext, Computer conferencing, Document retrieval, Information management, Project control

1989

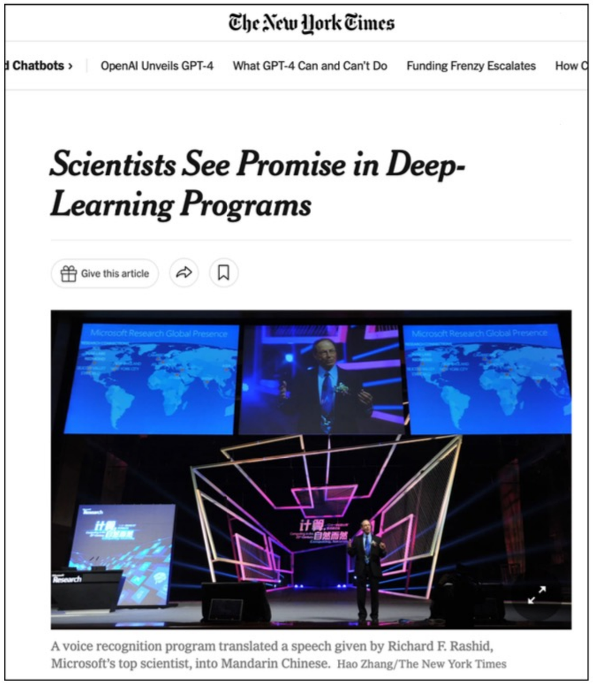
~10 years →

~25 years →



Slide credit: L. Heinrich

Still Early for Deep Learning, Where Will We be in 25 Years?


The New York Times

Chatbots › | OpenAI Unveils GPT-4 | What GPT-4 Can and Can't Do | Funding Frenzy Escalates | How C

Scientists See Promise in Deep-Learning Programs

Give this article

A voice recognition program translated a speech given by Richard F. Rashid, Microsoft's top scientist, into Mandarin Chinese. Hao Zhang/The New York Times

2012

~10 years

Prompt: `Several giant wooly mammoths approach treading through a snowy meadow [...]`

[OpenAI Sora]

~25 years

?

Slide credit: L. Heinrich

Do These Models Know Physics?... Maybe Not Yet

Credit: [Jim Fan](#) + Sora

Summary

- Deep neural networks allow us to learn complex function by hierarchically structuring the feature learning
- We can express our inductive bias about a system in terms of model design, and can be adapted to a many types of data
- Many neural networks structures are available for training models on a wide array of data types.
- Beyond classification and regression, deep neural networks allow for powerful generative models to enable us to model and generate data

Backup

Deep Neural Networks

Neural Network Zoo

- Structure of the networks, and the node connectivity can be adapted for problem at hand
- Moving inductive bias from feature engineering to model design
  - *Inductive bias*:
    Knowledge about the problem
  - *Feature engineering*:
    Hand crafted variables
  - *Model design*:
    The data representation and the structure of the machine learning model / network

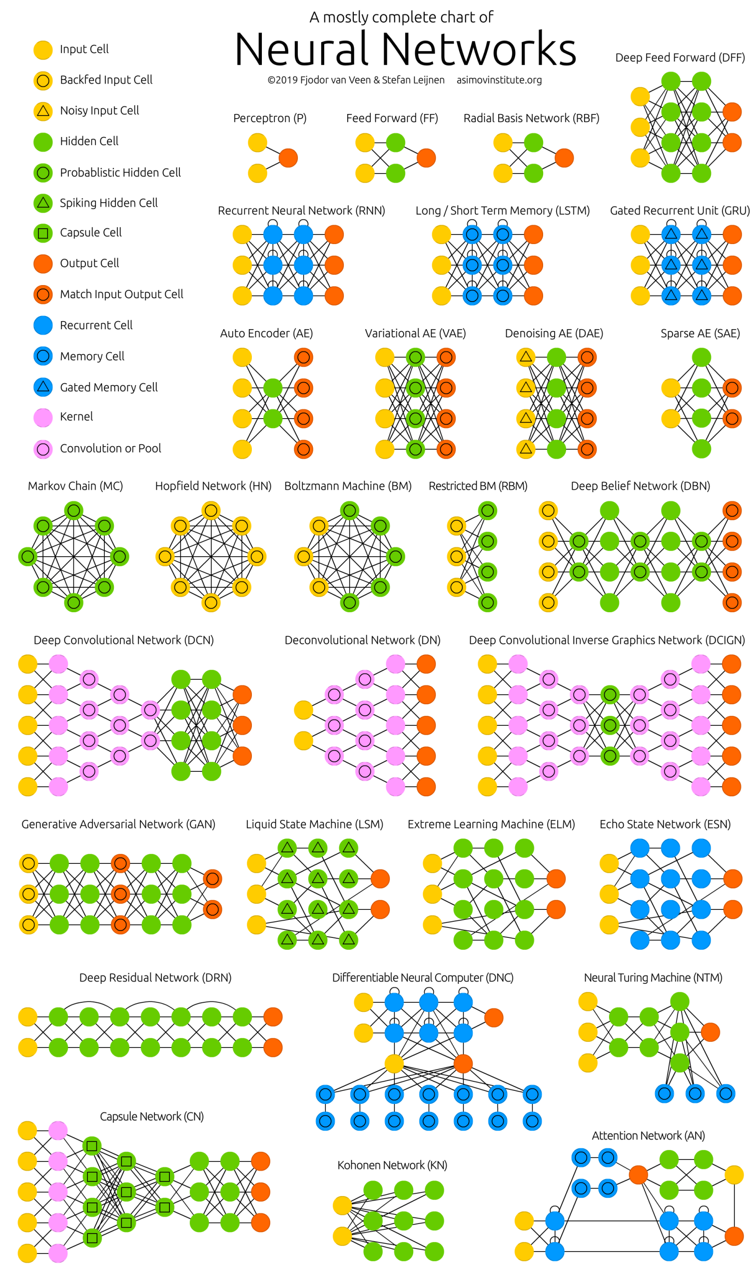


Image credit: [neural-network-zoo]

Neural Network Zoo – “Optimization” Perspective

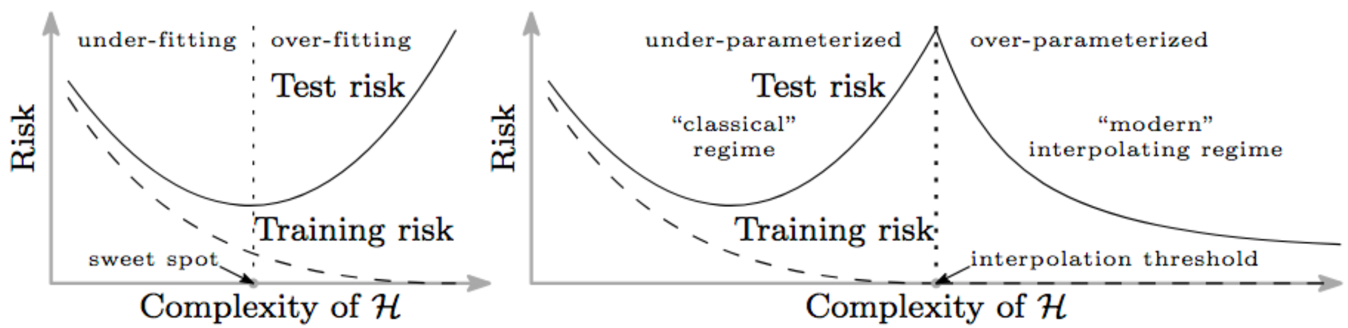
- A single layer network may need a width exponential in D to approximate a depth-D network’s output
  - Simplified version of Telgarsky (2015, 2016)

Neural Network Zoo – "Optimization" Perspective

- A single layer network may need a width exponential in D to approximate a depth-D network's output
  - Simplified version of Telgarsky (2015, 2016)

- Over-parametrizing a deep model often improves test performance, contrary to bias-variance tradeoff prediction

Belkin et. al. 2018



(a) U-shaped "bias-variance" risk curve

(b) "double descent" risk curve

Figure 1: Curves for training risk (dashed line) and test risk (solid line). (a) The classical *U-shaped risk curve* arising from the bias-variance trade-off. (b) The *double descent risk curve*, which incorporates the U-shaped risk curve (i.e., the "classical" regime) together with the observed behavior from using high complexity function classes (i.e., the "modern" interpolating regime), separated by the interpolation threshold. The predictors to the right of the interpolation threshold have zero training risk.

Neural Network Zoo – "Optimization" Perspective

- A single layer network may need a width exponential in D to approximate a depth-D network's output
  - Simplified version of Telgarsky (2015, 2016)

- Over-parametrizing a deep model often improves test performance, contrary to bias-variance tradeoff prediction

  - But we must control that:
    - Gradients don't vanish
    - Gradient amplitude is homogeneous across network
    - Gradients are under control when weights change

Neural Network Zoo – "Optimization" Perspective

- A single layer network may need a width exponential in D to approximate a depth-D network's output
  - Simplified version of Telgarsky (2015, 2016)

- Over-parametrizing a deep model often improves test performance, contrary to bias-variance tradeoff prediction

- Major part of deep learning is choosing the right function

  - Need to make gradient descent work, even if substantial engineering required

Convolutional Neural Networks

1D Convolutional Layers

- **Data:** $x \in \mathbb{R}^M$
- **Convolutional kernel of width k:** $u \in \mathbb{R}^k$

- Convolution $x \circledast u$ is vector of size M-k+1

$$(x \circledast \mathrm{u})_i = \sum_{b=0}^{k-1} x_{i+b} u_b$$

- Scan across data and multiply by kernel elements

2D Convolutional Layer

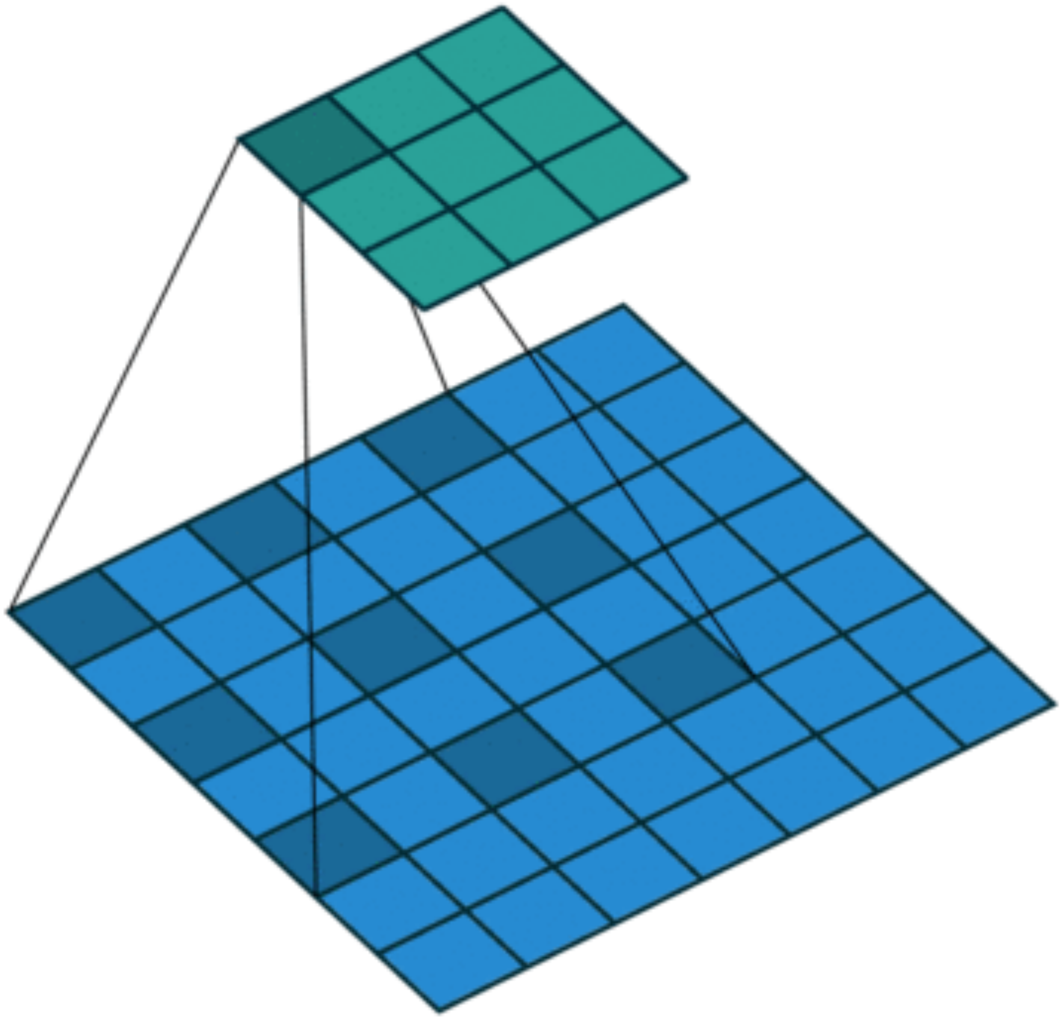
- Input data (tensor) **x** of size $C \times H \times W$
  - C channels (e.g. RGB in images)

- Learnable Kernel **u** of size $C \times h \times w$
  - The size $h \times w$ is the *receptive field*

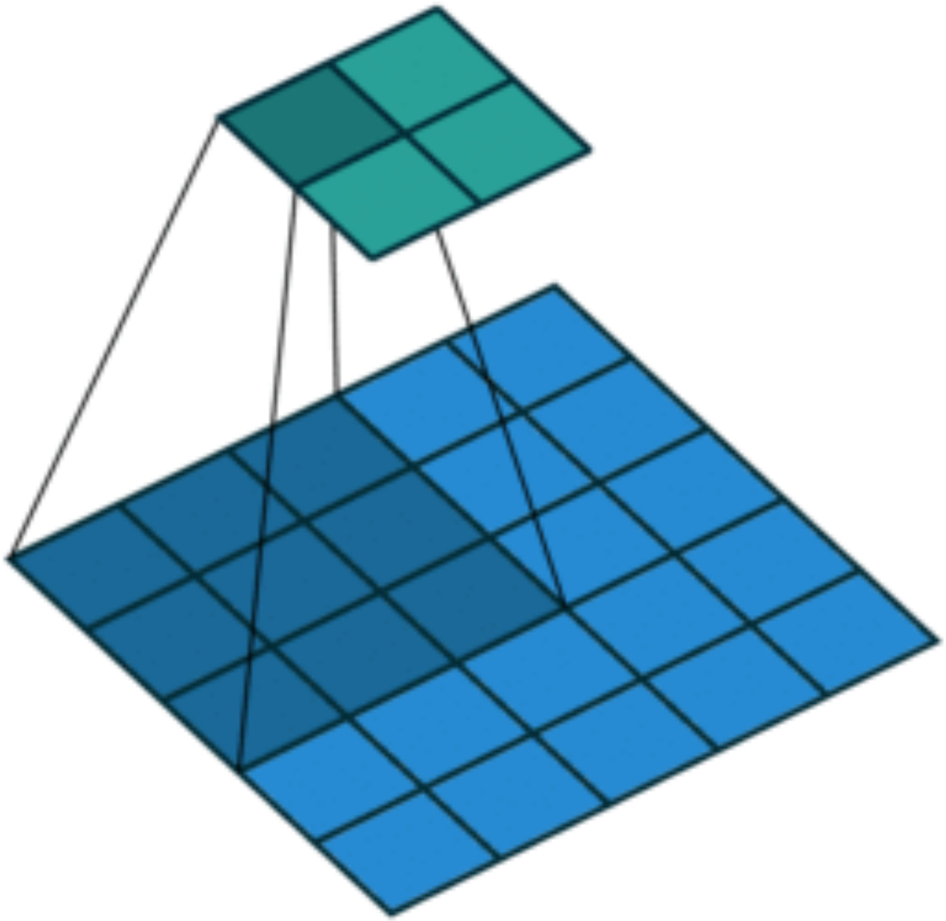
$$(\boldsymbol{x} \circledast \boldsymbol{u})_{i,j} = \sum_{c=0}^{C-1} (\boldsymbol{x}_c \circledast \boldsymbol{u}_c)_{i,j} = \sum_{c=0}^{C-1} \sum_{n=0}^{h-1} \sum_{m=0}^{w-1} \boldsymbol{x}_{c,n+i,m+j} \boldsymbol{u}_{c,n,m}$$

- Output size $(H - h + 1) \times (W - w + 1)$ for each kernel
  - Often called *Activation Map* or *Output Feature Map*

Dilation

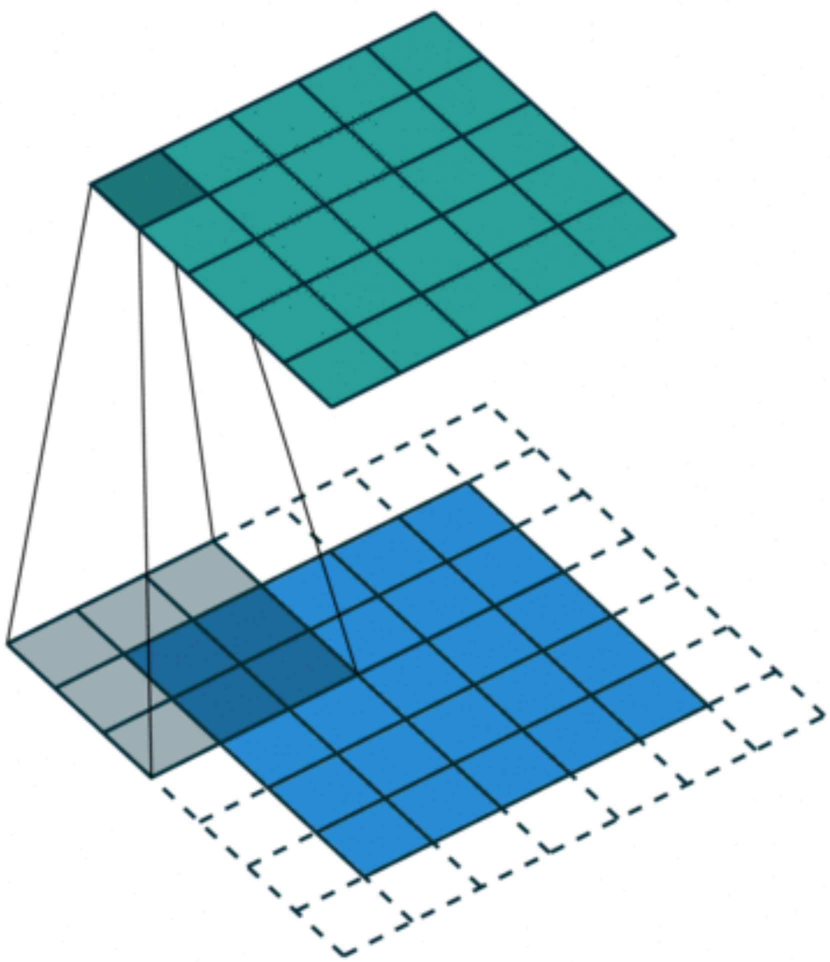


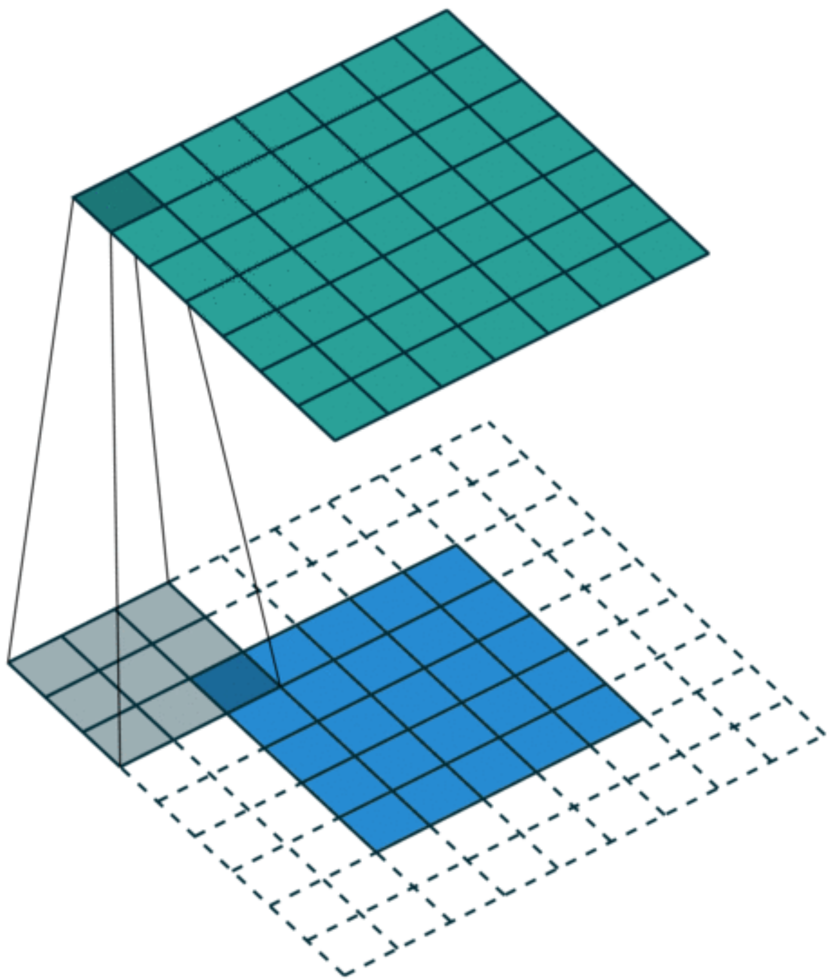
Stride – Step Size When Moving Kernel Across Input



Fleuret, Deep Learning Course

Padding – Size of Zero Frame Around Input





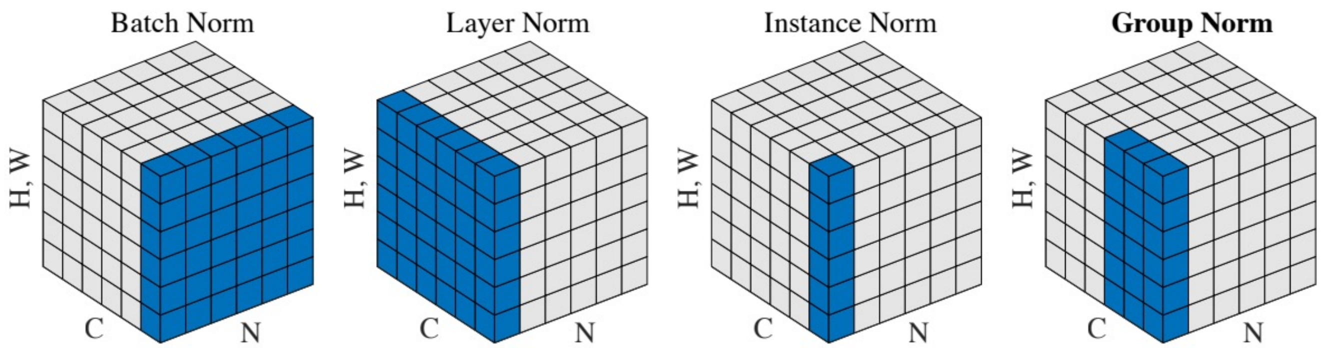
Fleuret, [Deep Learning Course]

Normalization

- Maintaining proper statistics of the activations and derivatives is a critical issue to allow the training of deep architectures

"Training Deep Neural Networks is complicated by the fact that **the distribution of each layer's inputs changes during training, as the parameters of the previous layers change**. This slows down the training by requiring lower learning rates and careful parameter initialization …"

Ioffe, Szegedy,
Batch Normalization, ICML 2015



Wu, He, *Group Normalization*, CoRR 2018

Batch Normalization

- During training batch normalization shifts and rescales according to the mean and variance estimated on the batch.
  - During test, use empirical moments estimated during training

- Per-component mean and variance on the batch

$$m_{batch} = \frac{1}{B}\sum_{b=1}^{B} x_b$$

$$v_{batch} = \frac{1}{B}\sum_{1}^{B} (x_b - m_{batch})^2$$

- Normalize and compute output $\forall b = 1 \dots B$

$$z_b = \frac{x_b - m_{batch}}{\sqrt{v_{batch} + \epsilon}}$$

$$y_b = \gamma \odot z_b + \beta$$

  - γ and β are parameters to optimize

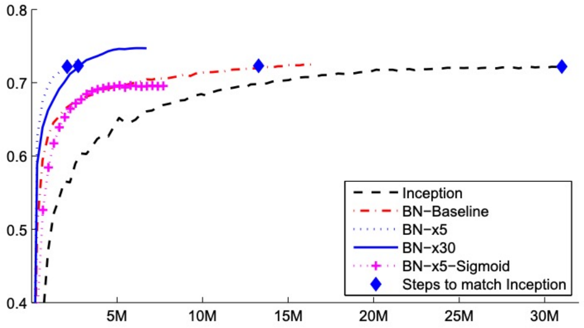
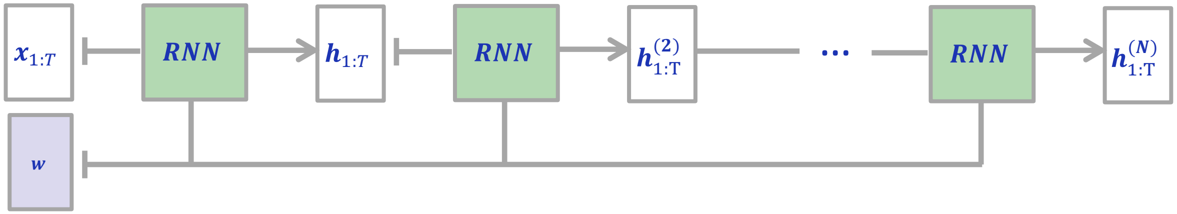


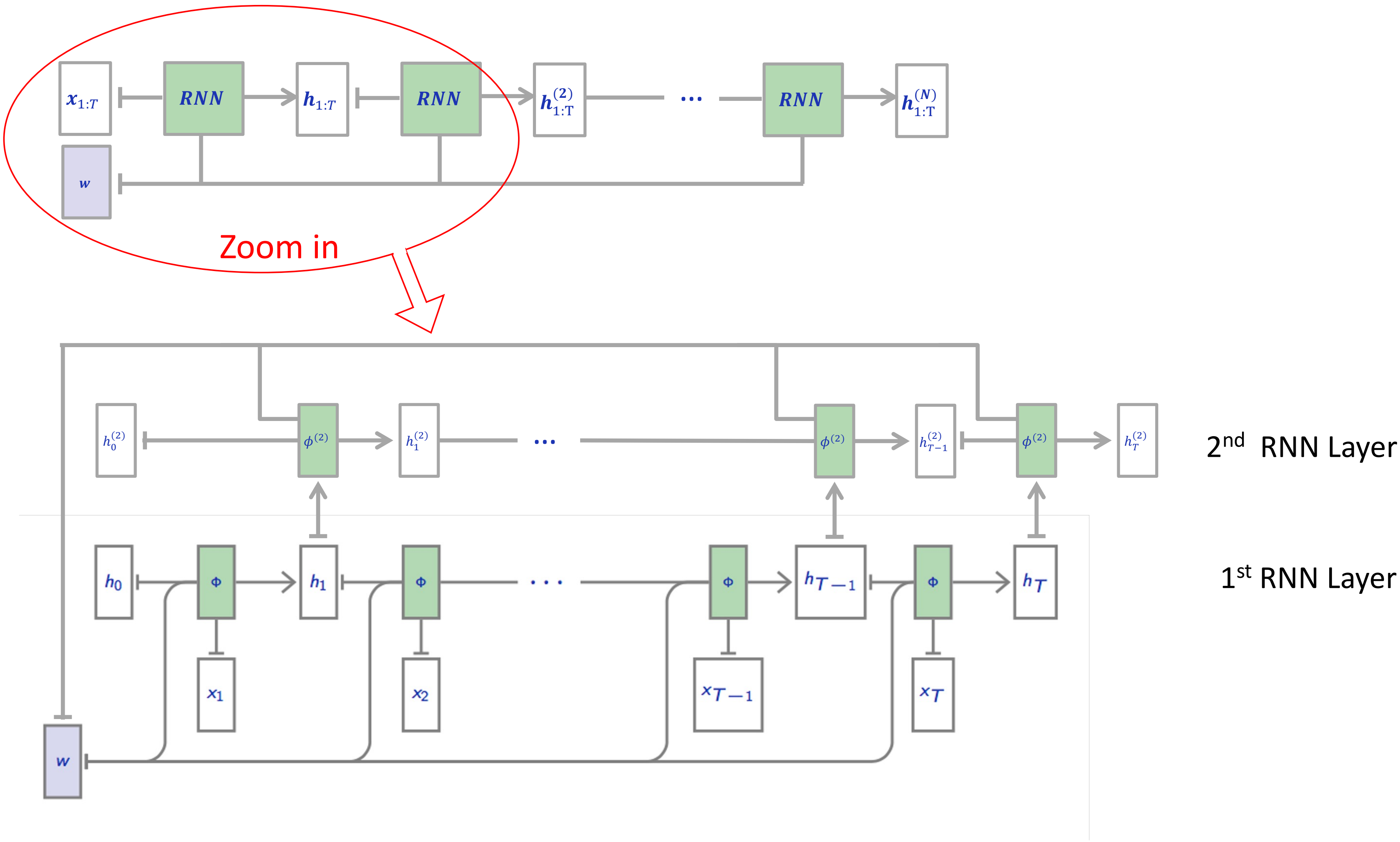
Figure 2: *Single crop validation accuracy of Inception and its batch-normalized variants, vs. the number of training steps.*

RNN

Stacked RNN



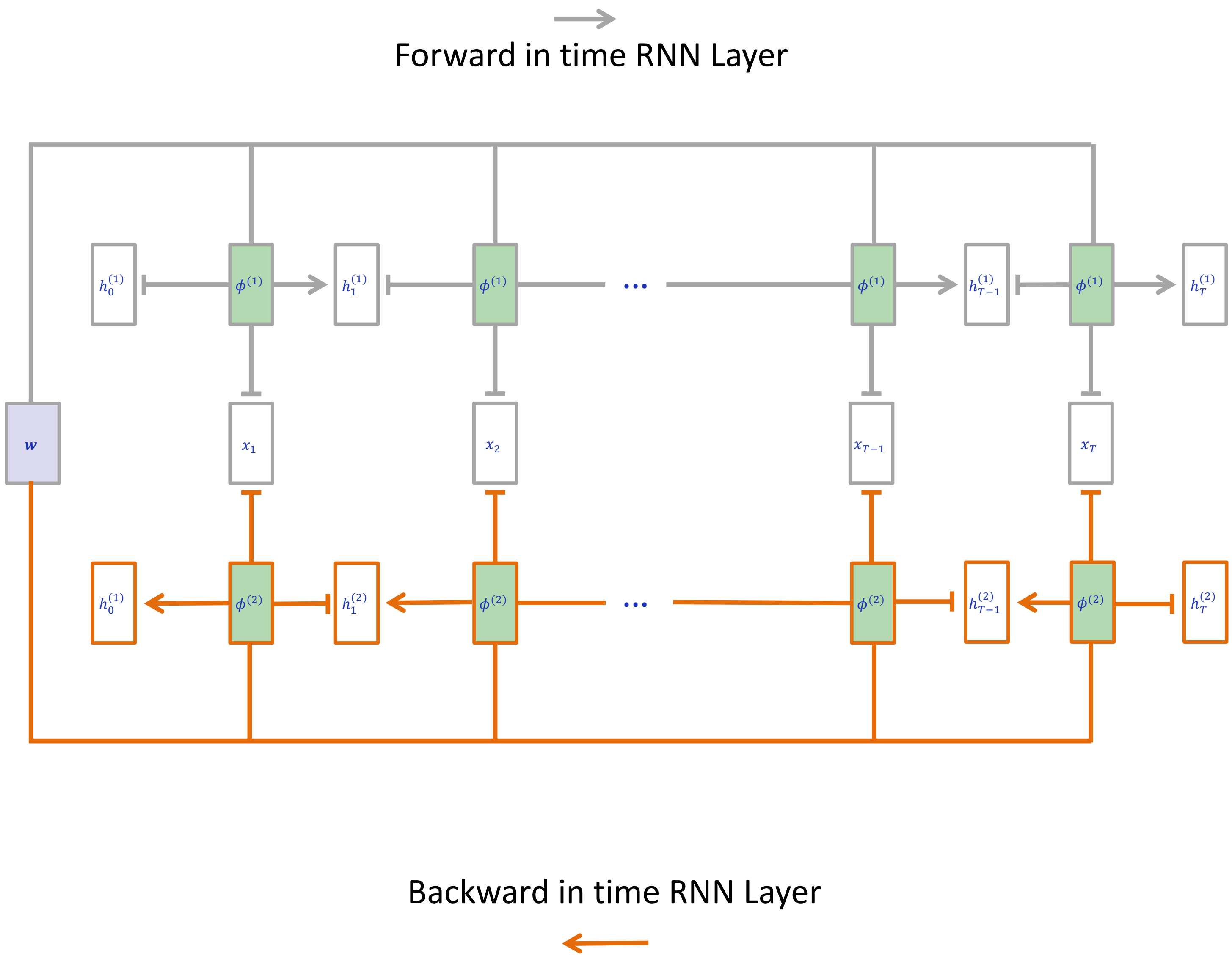
Stacked RNN



Two Stacked LSTM Layers

Bi-Directional RNN

Forward in time RNN Layer

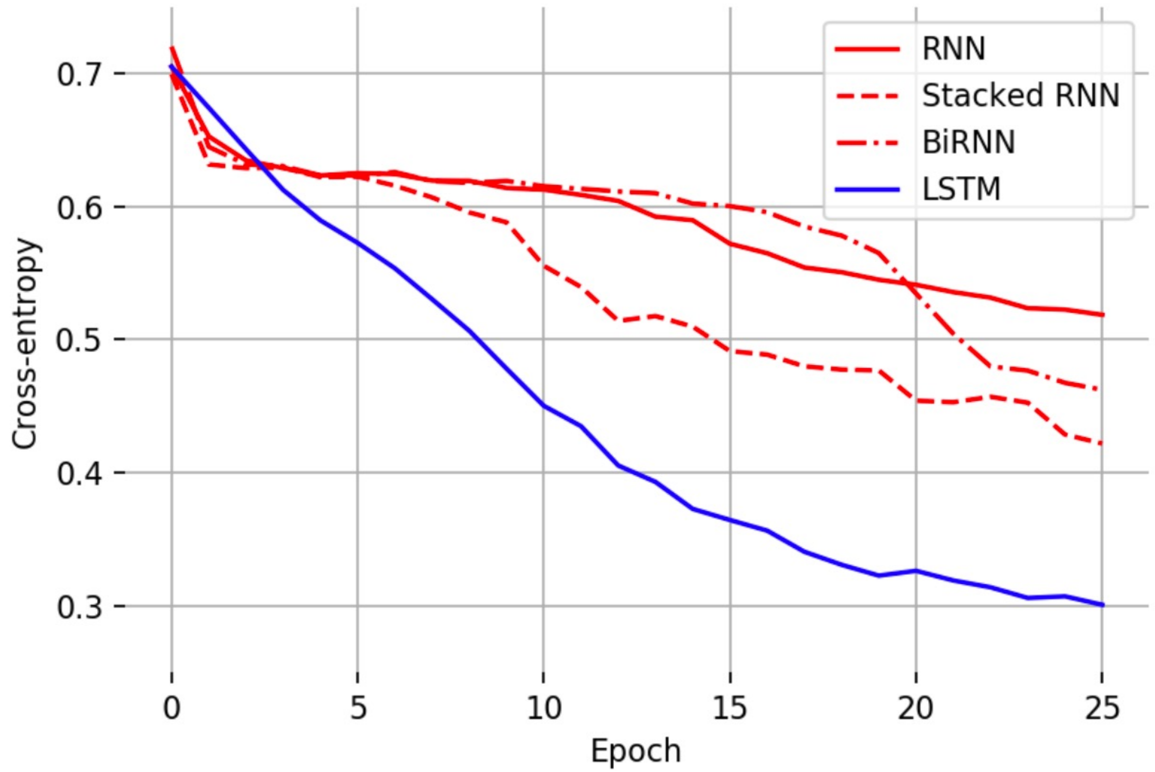


Backward in time RNN Layer

Comparison on Toy Problem

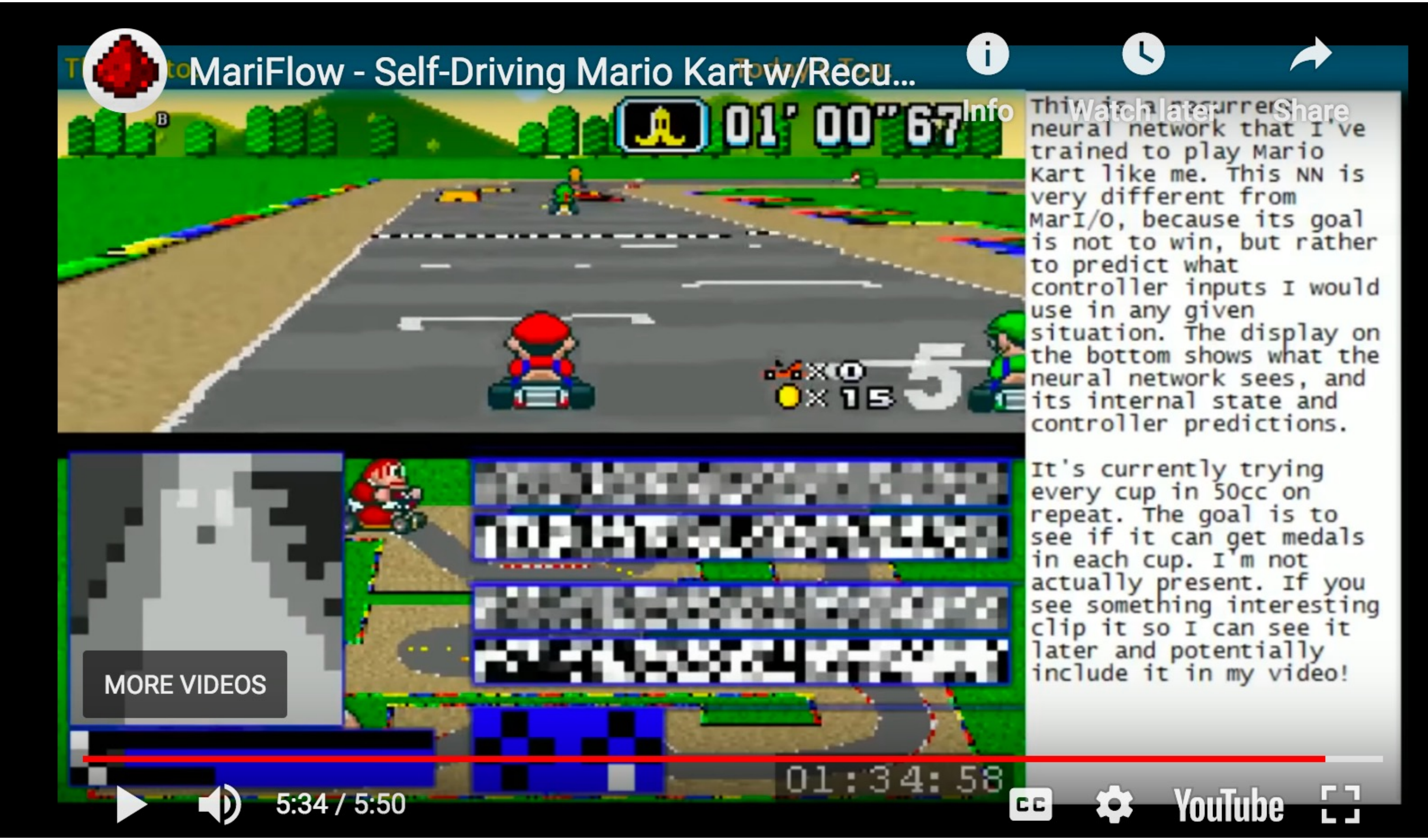
Learn to recognize palindrome
Sequence size between 1 to 10

$\mathbf{x}$	y
$(1, 2, 3, 2, 1)$	1
$(2, 1, 2)$	1
$(3, 4, 1, 2)$	0
(0)	1
$(1, 4)$	0



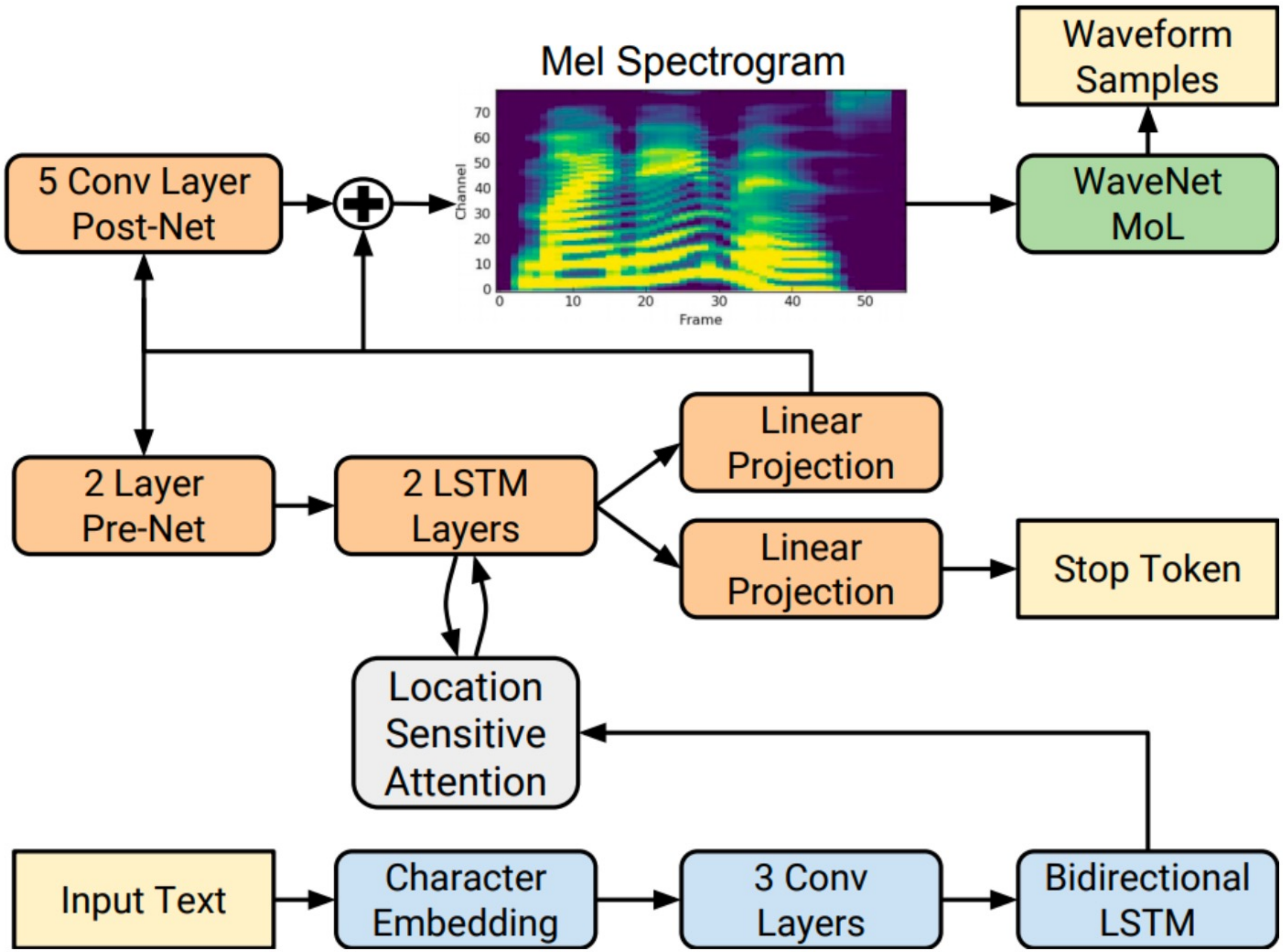
Slide credit: G. Louppe

Examples



Self-driving Mario Kart with RNN: [YouTube video]

Text-to-speech synthesis



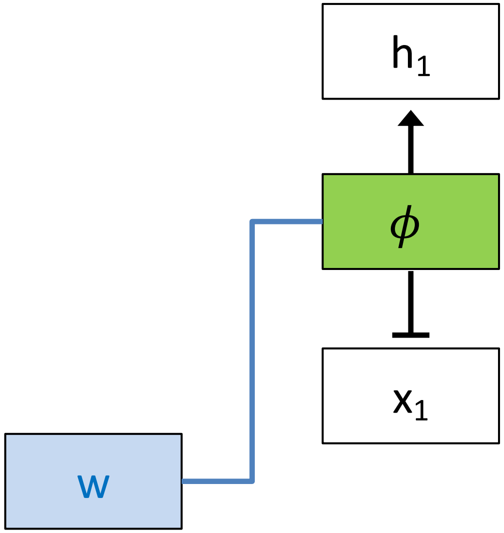
Shen et al., [2017]

Deep Sets

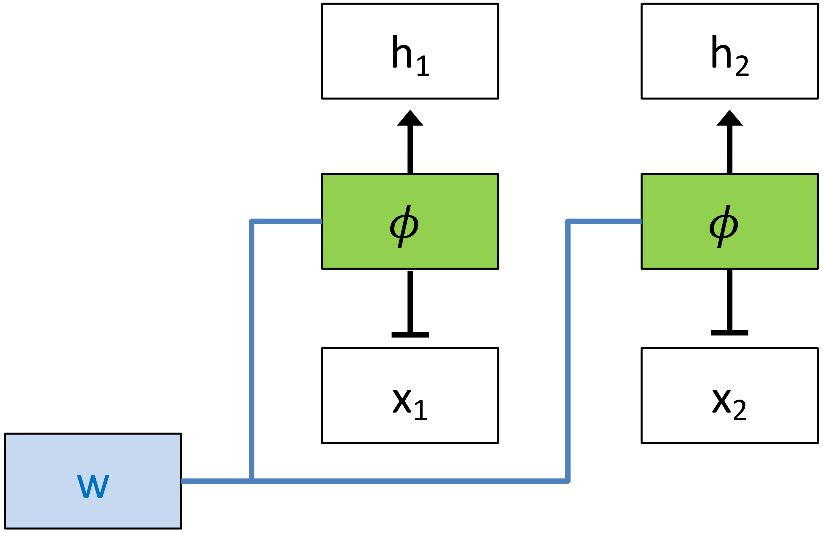
What if our data has no time structure?

- Data may be variable in length but have no temporal structure → *Data are sets of values*
- *One option*: If we know about the data domain, could try to impose an ordering, then use RNN
- *Better option*: use system that can operate on variable length sets in permutation invariant way
  – Why permutation invariant → so order doesn't matter

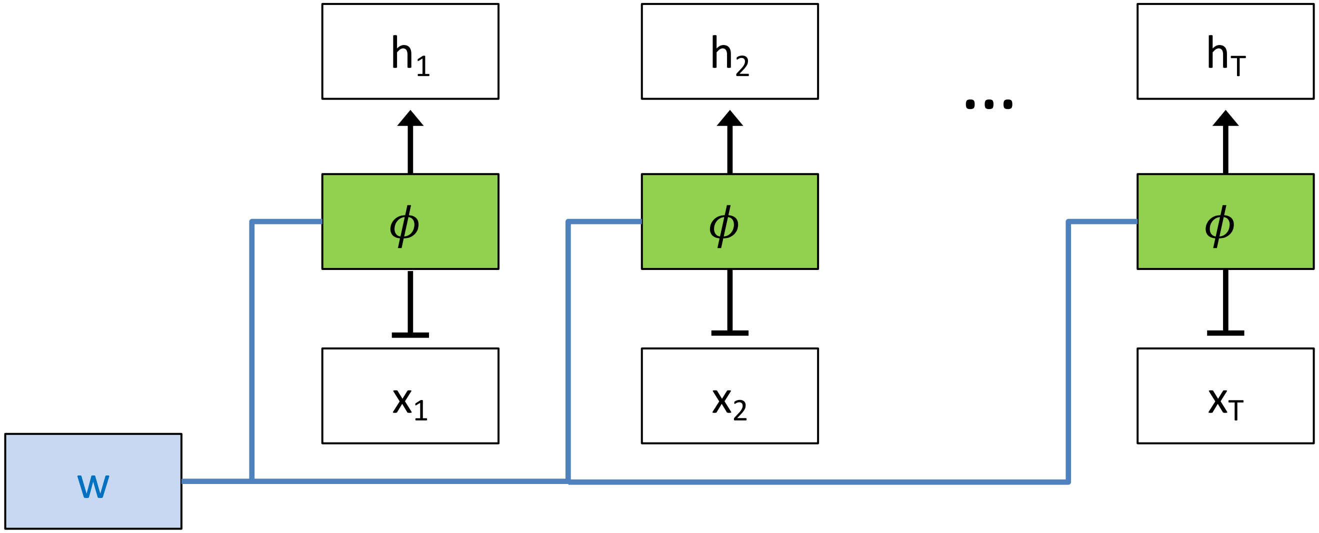
Deep Sets



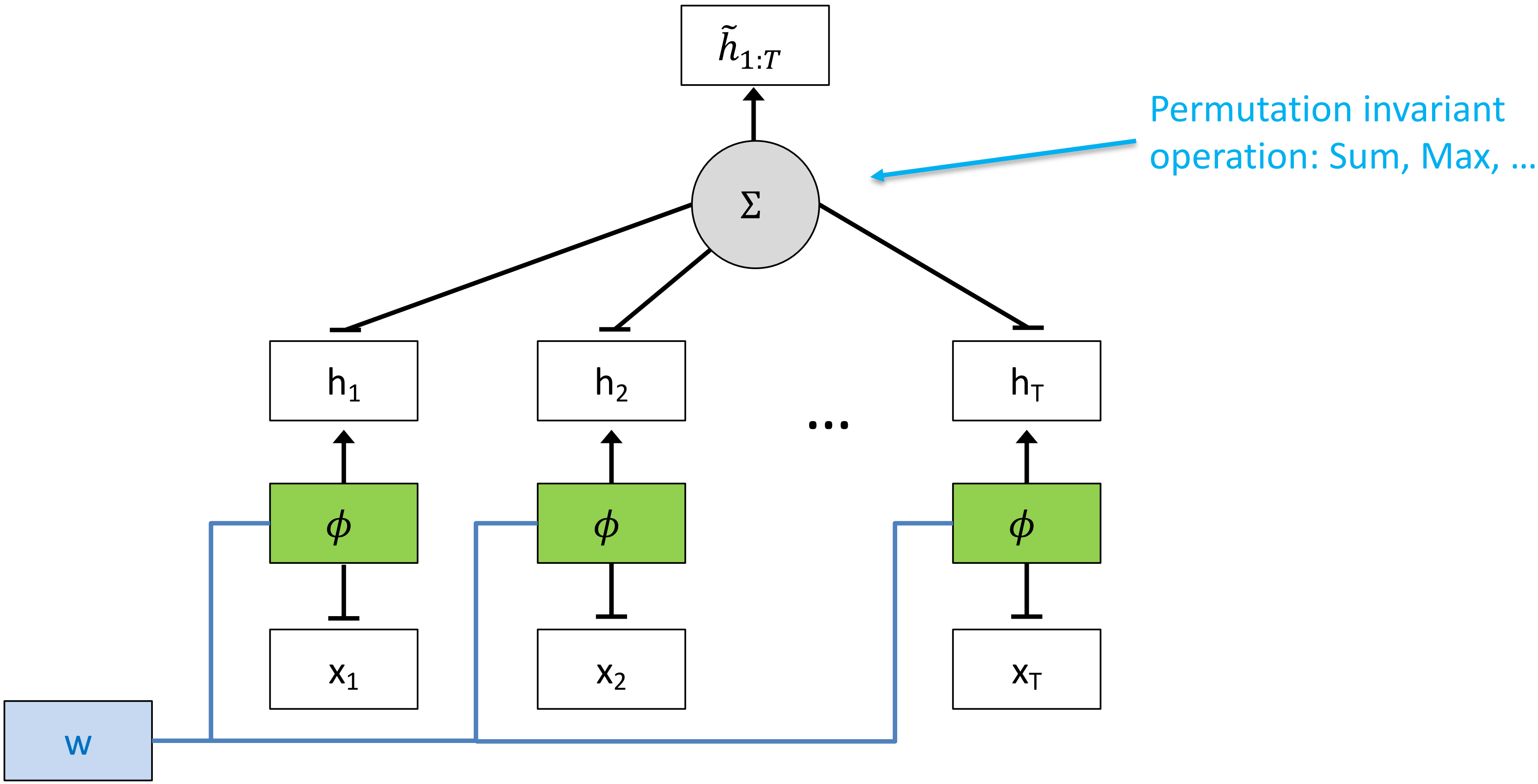
Deep Sets



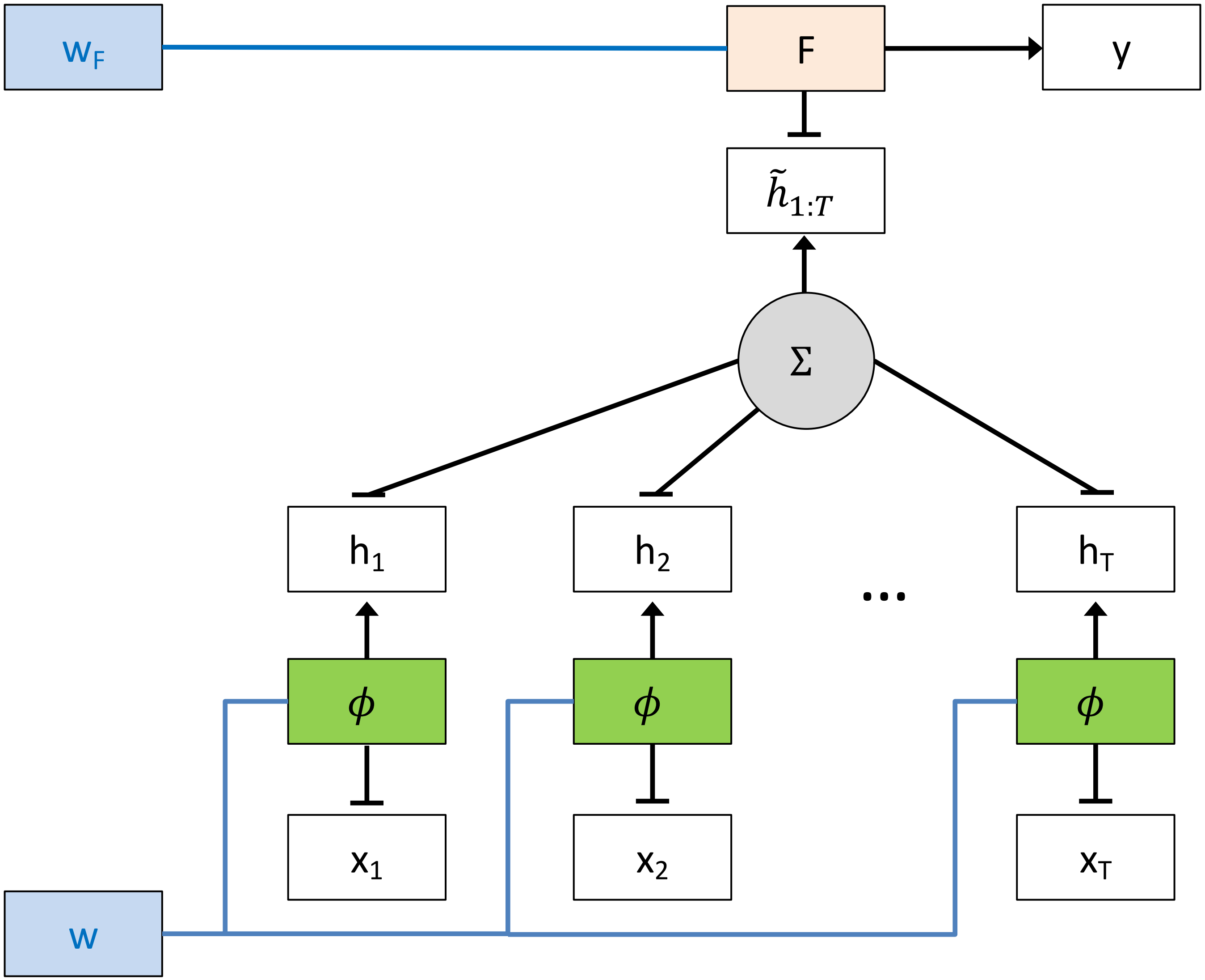
Deep Sets



Deep Sets


$\tilde{h}_{1:T}$
Σ
Permutation invariant operation: Sum, Max, …
h_1
h_2
…
h_T
ϕ
ϕ
ϕ
x_1
x_2
x_T
w

Deep Sets



Examples

Outlier detection

M. Zaheer et. al 2017

Medical Imaging

With more complex architecture

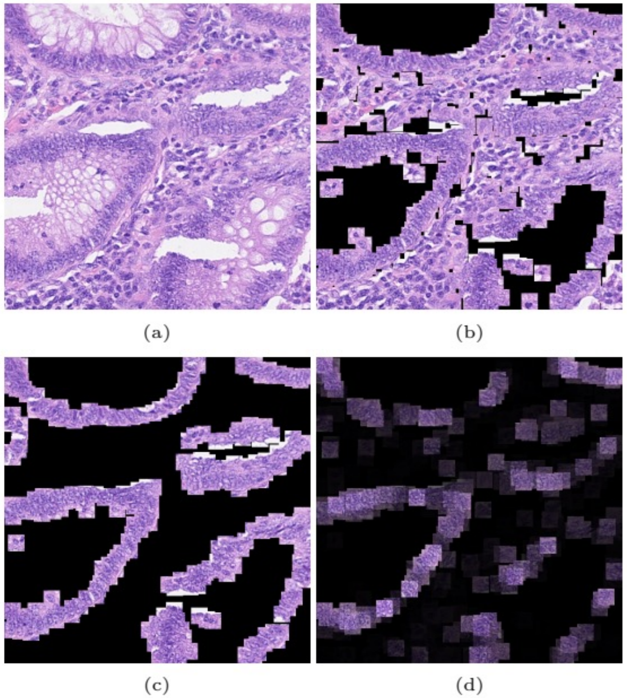


Figure 5. (a) H&E stained histology image. (b) 27×27 patches centered around all marked nuclei. (c) Ground truth: Patches that belong to the class epithelial. (d) Heatmap: Every patch from (b) multiplied by its corresponding attention weight, we rescaled the attention weights using $a'_k = (a_k - \min(\mathbf{a}))/(\max(\mathbf{a}) - \min(\mathbf{a}))$.

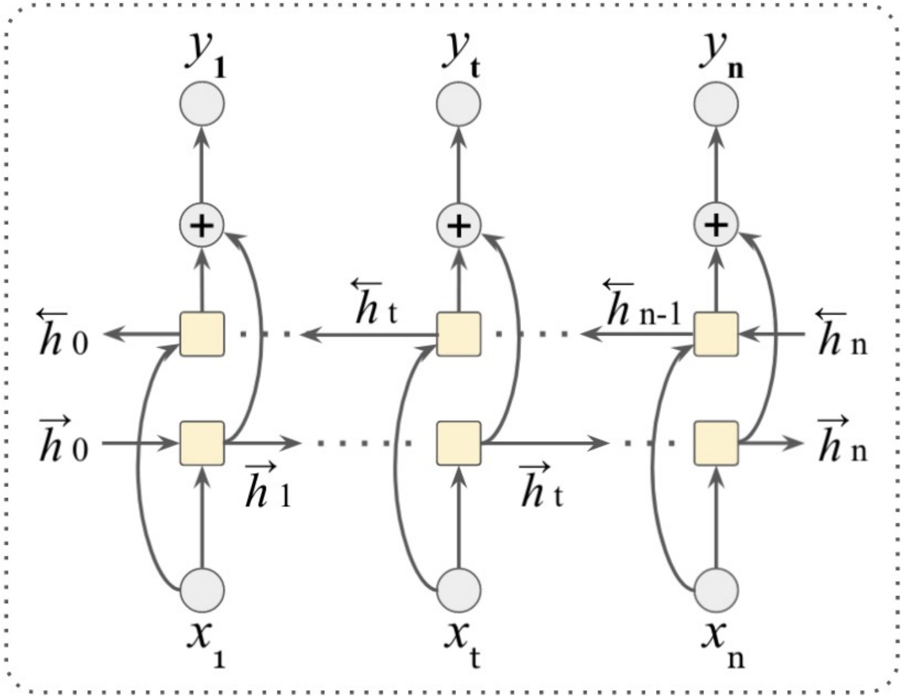
M. Ilse et al., 2018

Transformers

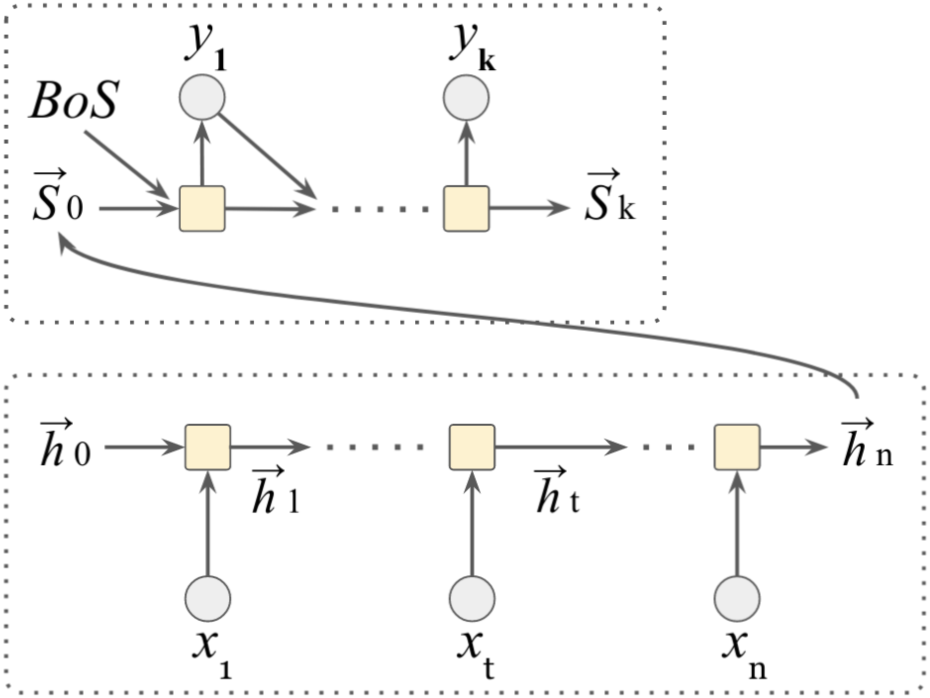
Challenges of Long Sequences

- Gradients may not explode or vanish, but managing a meaningful context over a long sequence is challenging.

- Bottleneck: fixed length array in model with long input



Bi-Directional



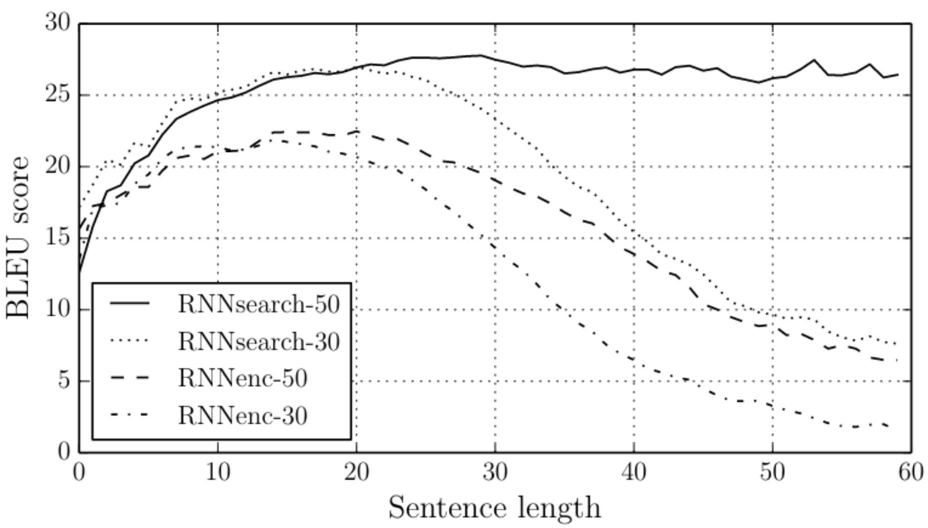
RNN Encoder-Decoder

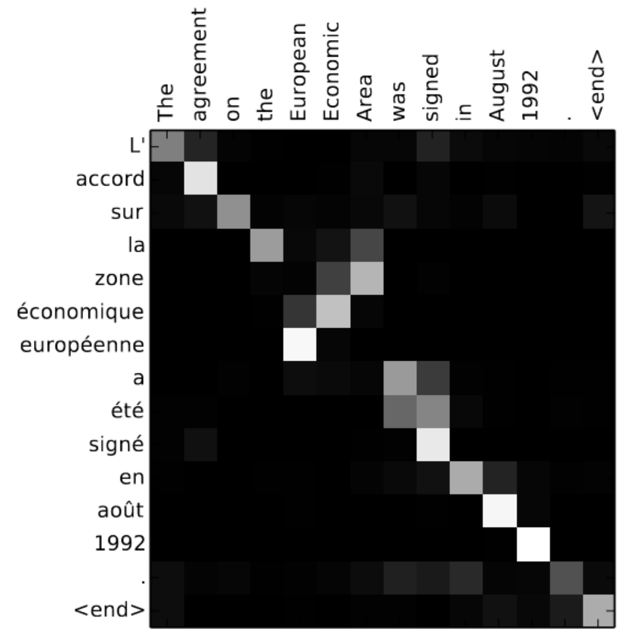
Additive Attention Mechanism

- Idea: allow RNN to look at all the hidden state sequence when producing an output. Output is generated from context c

$$c_i = \sum_{j=1}^{T} \alpha_{ij} h_j \quad \text{where} \quad \alpha_{ij} = softmax(\beta_{ij})_{over\ j}$$

$$\text{and} \quad \beta_{ij} = U \tanh(W s_{i-1} + \widetilde{W} h_j + b_i)$$

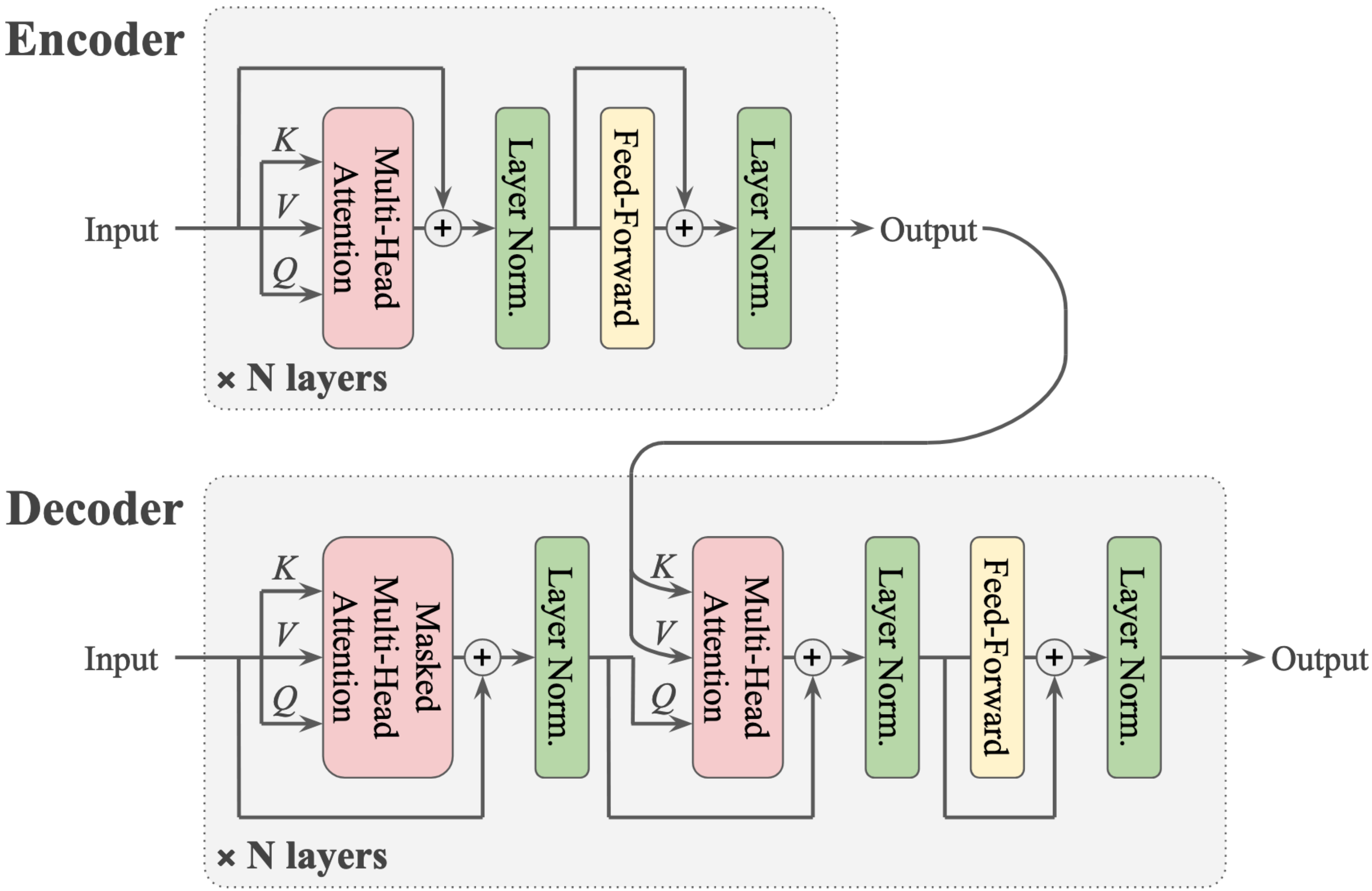




1409.0473

Transformers

- Idea: Get rid of the RNN and only use attention



Scaled Dot-Product Attention

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V$$

where

$$Q \in \mathbb{R}^{n \times d}$$
$$K \in \mathbb{R}^{m \times d}$$
$$V \in \mathbb{R}^{m \times d_v}$$

Query

$$Q = \begin{pmatrix} \vec{q}_1 \\ \vdots \\ \vec{q}_n \end{pmatrix}_{n \times d}$$

Key

$$K = \begin{pmatrix} \vec{k}_1 \\ \vdots \\ \vec{k}_m \end{pmatrix}_{m \times d}$$

Value

$$V = \begin{pmatrix} \vec{v}_1 \\ \vdots \\ \vec{v}_m \end{pmatrix}_{m \times d_v}$$

- Project the input "query" onto a "key" to compute the weights for the corresponding "value"
- Return the weighted value

Scaled Dot-Product Attention

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V$$

where

$$Q \in \mathbb{R}^{n\times d}$$
$$K \in \mathbb{R}^{m\times d}$$
$$V \in \mathbb{R}^{m\times d_v}$$

- ***Self-Attention***: using input X to define Q,K,V

$$Q = XW_Q \qquad K = XW_K \qquad V = XW_V$$

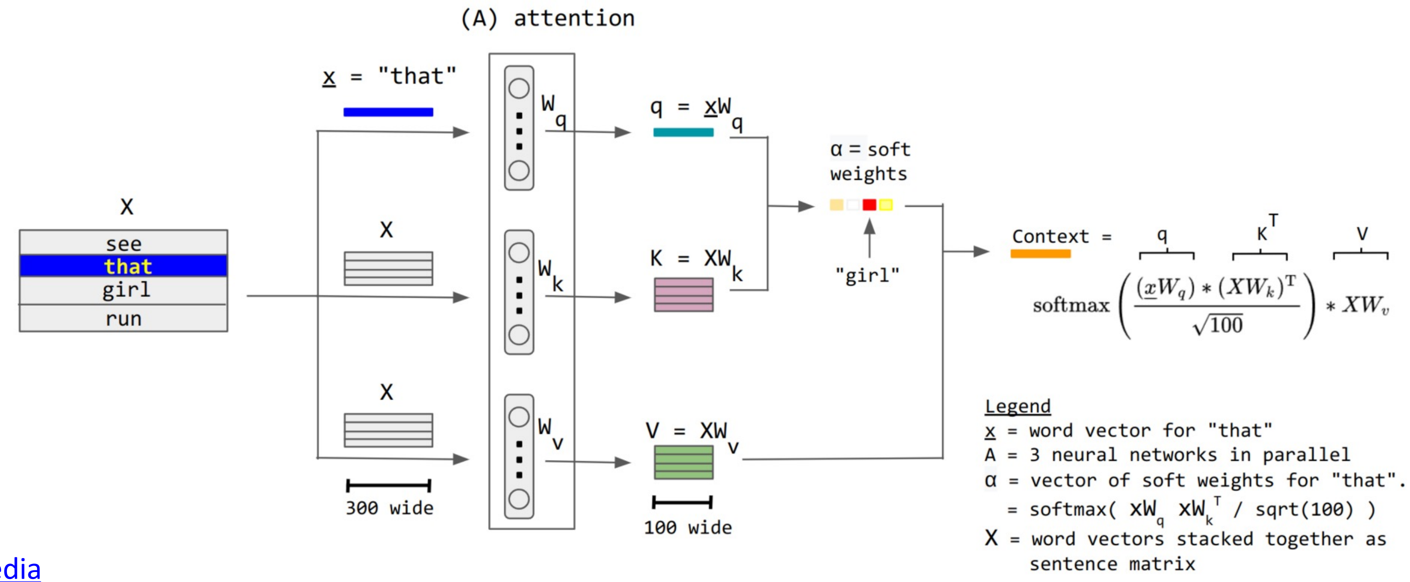


Image credit: Wikipedia

- Lets look at a single query

$$\frac{qK^T}{\sqrt{d}} = \left(\frac{\vec{q}_1 \cdot \vec{k}_1}{\sqrt{d}}, \frac{\vec{q}_1 \cdot \vec{k}_2}{\sqrt{d}}, \cdots, \frac{\vec{q}_1 \cdot \vec{k}_m}{\sqrt{d}}\right)_{1\times m}$$

$$\text{softmax}\left(\frac{qK^T}{\sqrt{d}}\right) = (p_1, p_2, ..., p_m)_{1\times m} = \vec{p} \quad \text{where} \quad p_i = \frac{\exp \frac{\vec{q}_1 \cdot \vec{k}_i}{\sqrt{d}}}{\sum_{j=1}^m \exp \frac{\vec{q}_1 \cdot \vec{k}_j}{\sqrt{d}}}$$

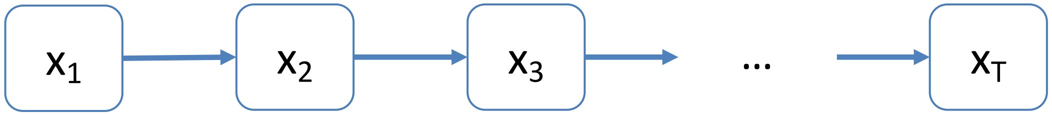
$$\text{Attention}(q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right) V = \vec{p}V = \sum_{i=1}^{n} p_i \vec{v}_i$$

- Generalize input to length n

$$\text{Attention}(Q, K, T) = \begin{pmatrix} p_{11}\vec{v}_1 + p_{12}\vec{v}_2 + \cdots + p_{1m}\vec{v}_m \\ p_{21}\vec{v}_1 + p_{22}\vec{v}_2 + \cdots + p_{2m}\vec{v}_m \\ \vdots \\ p_{n1}\vec{v}_1 + p_{n2}\vec{v}_2 + \cdots + p_{nm}\vec{v}_m \end{pmatrix} = \begin{pmatrix} \sum_i^m p_{1i}\vec{v}_i \\ \sum_i^m p_{2i}\vec{v}_i \\ \vdots \\ \sum_i^m p_{ni}\vec{v}_i \end{pmatrix}_{n\times d_v}$$

Graph Neural Networks

Graph Data



- Sequential data has single (directed) connections from data at current time to data at next time

- What about data with more complex dependencies

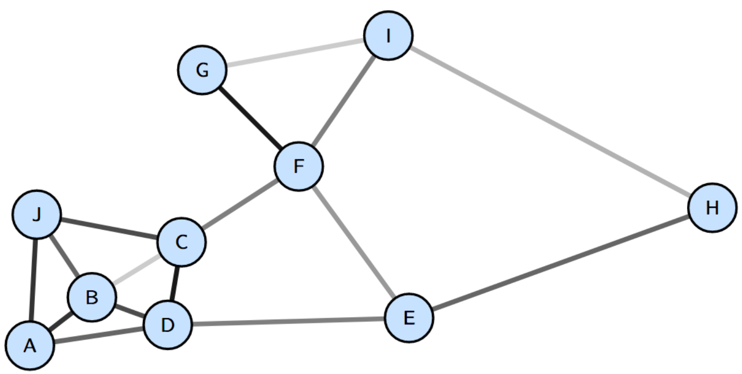
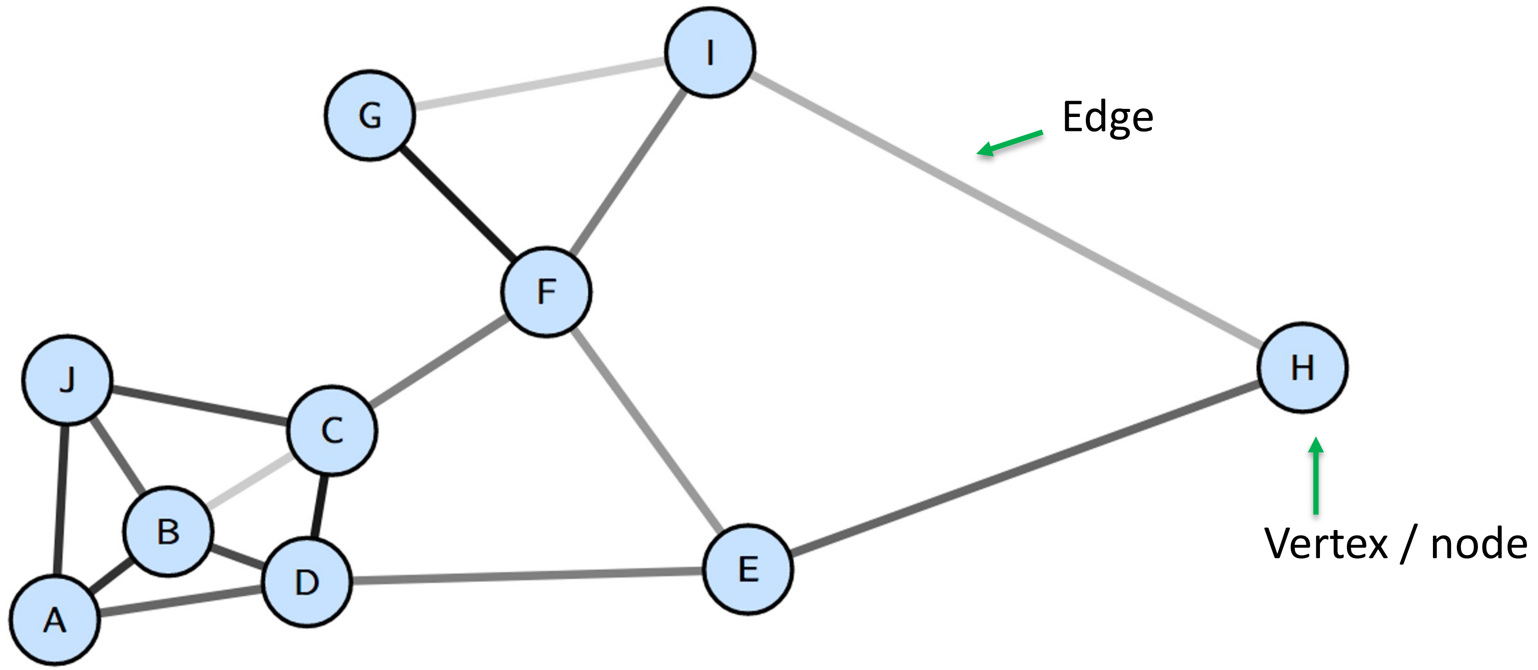


Image Credit: [I. Henrion]

Image credit: N. Wang et al., [2018]

Graphs



- Adjacency matrix: $A_{ij} = \delta(edge\ between\ vertex\ i\ and\ j)$
- Each node can have features
- Each edge can have features, e.g. distance between nodes

Image Credit: I. Henrion

Neural Message Passing

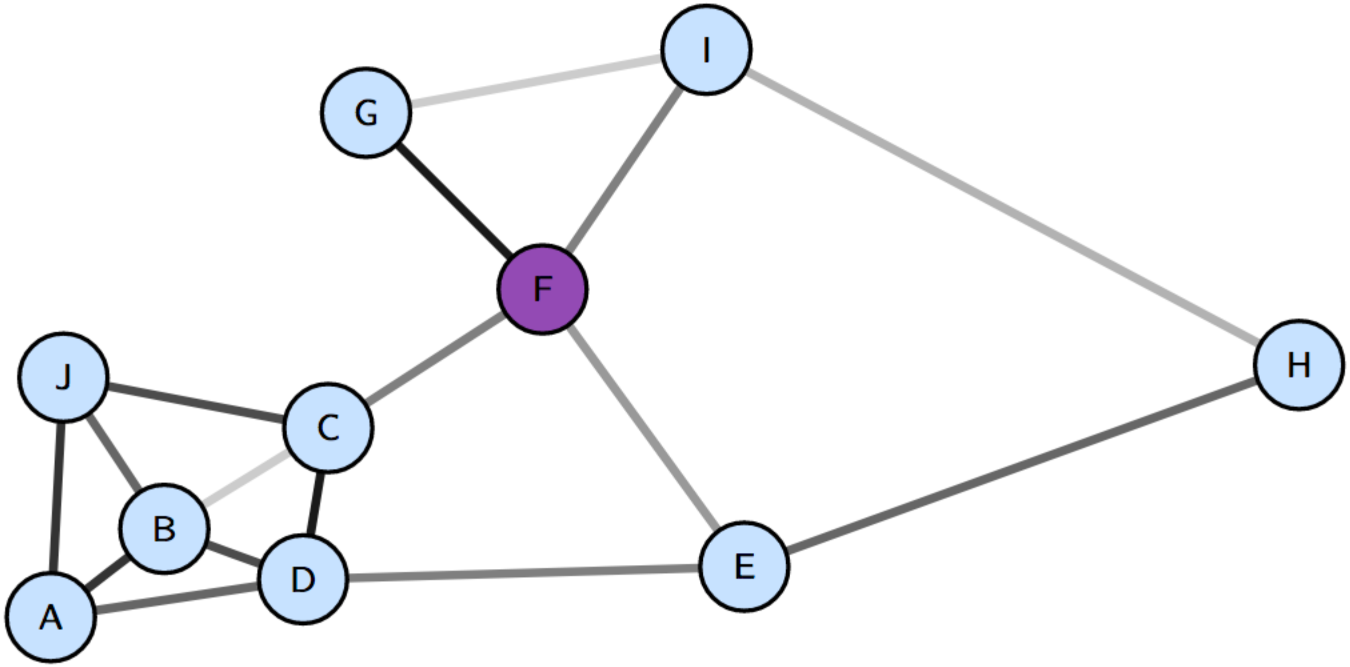
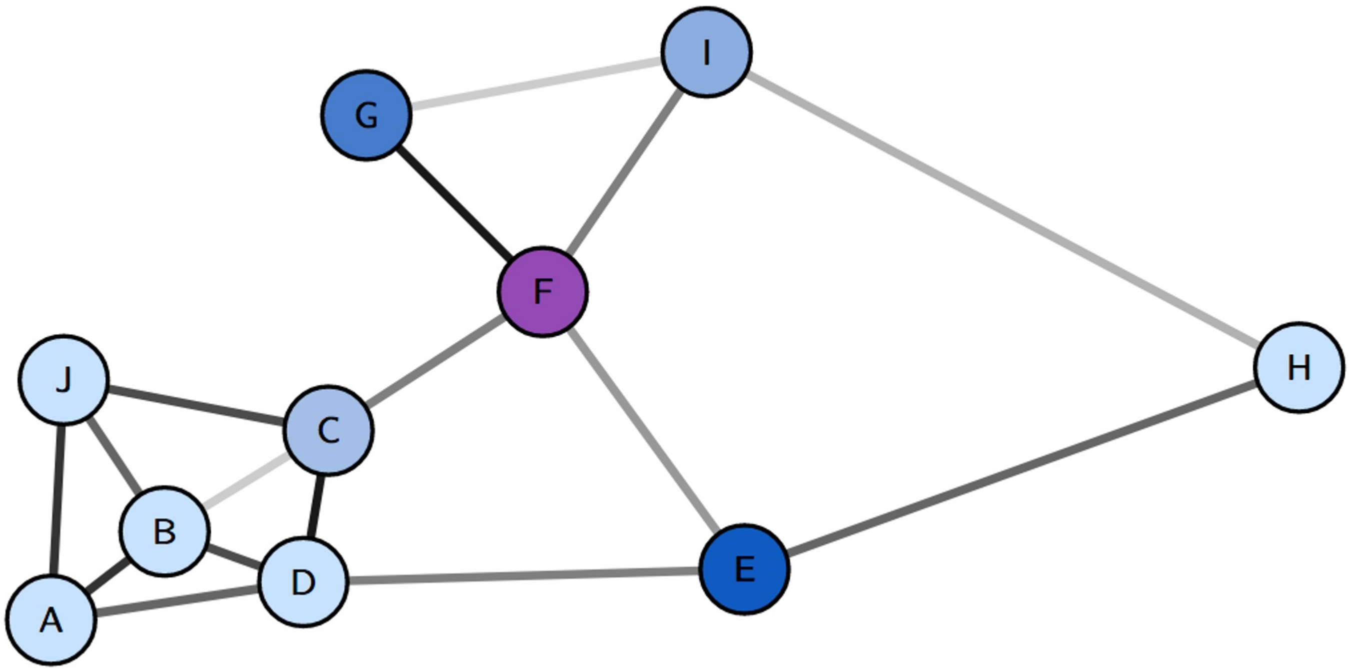


Image Credit: [I. Henrion]

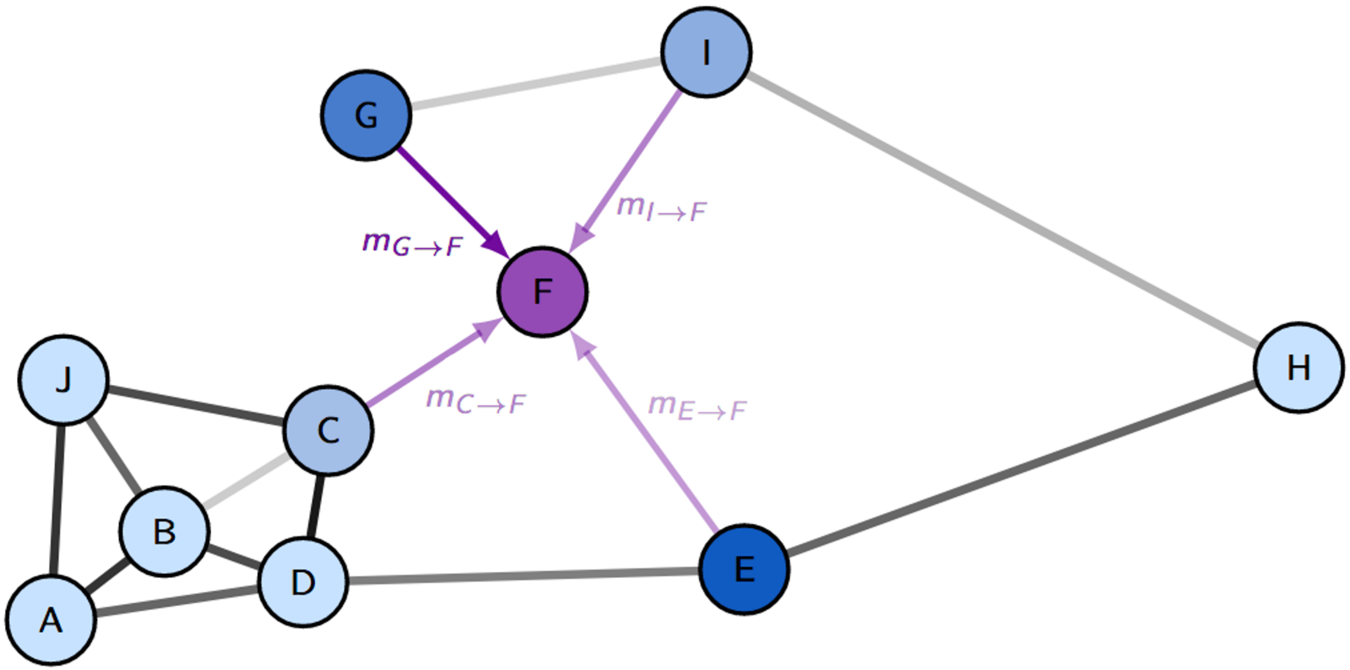
Neural Message Passing



$$\tilde{m}_j^t = f(h_j^{t-1})$$

Image Credit: [I. Henrion]

Neural Message Passing

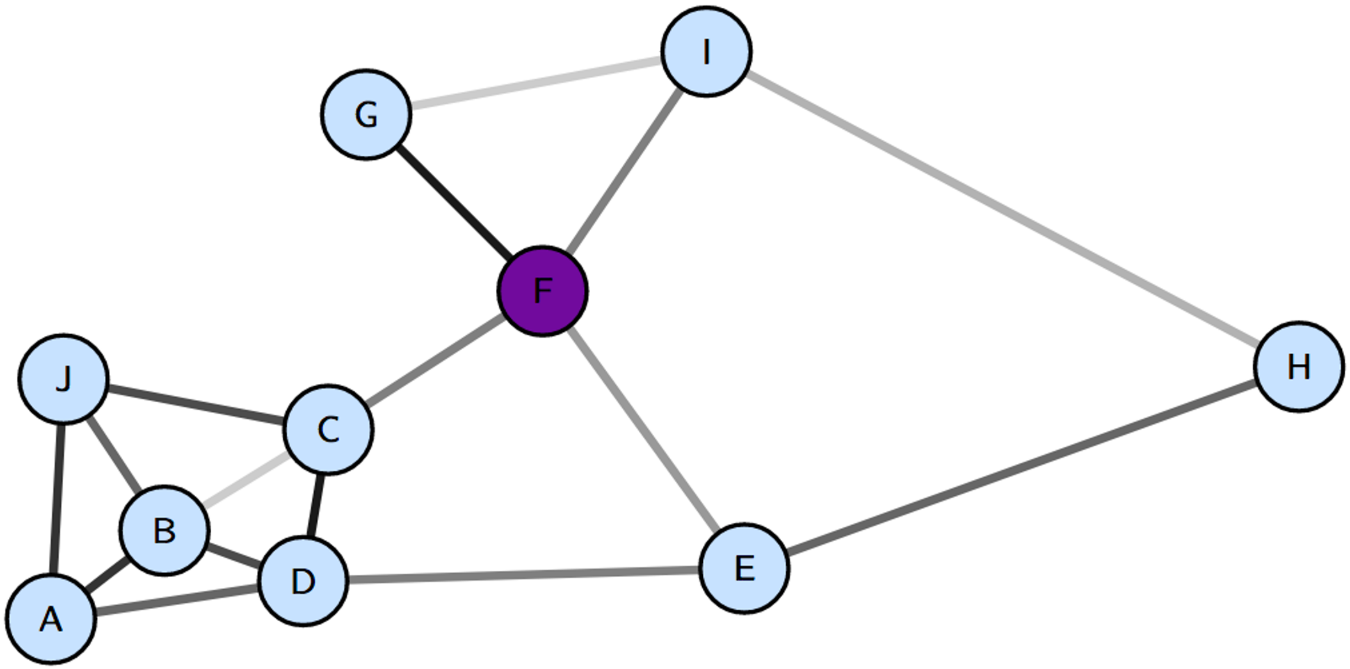


$$\tilde{m}_j^t = f(h_j^{t-1})$$

$$m_{j \to i}^t = \sigma(A_{ij} \tilde{m}_j^t)$$

Image Credit: I. Henrion

Neural Message Passing



$$\tilde{m}_j^t = f(h_j^{t-1})$$

$$m_{j \to i}^t = \sigma(A_{ij} \tilde{m}_j^t)$$

$$h_i^t = \mathrm{GRU}(h_i^{t-1}, \Sigma_j m_{j \to i}^t)$$

Image Credit: [I. Henrion]

Neural Message Passing

Algorithm 1 Message passing neural network

Require: $N \times D$ nodes $\mathbf{x}$, adjacency matrix A

$\mathbf{h} \leftarrow \text{Embed}(\mathbf{x})$

for $t = 1, \ldots, T$ **do**

  $\mathbf{m} \leftarrow \text{Message}(A, \mathbf{h})$

  $\mathbf{h} \leftarrow \text{VertexUpdate}(\mathbf{h}, \mathbf{m})$

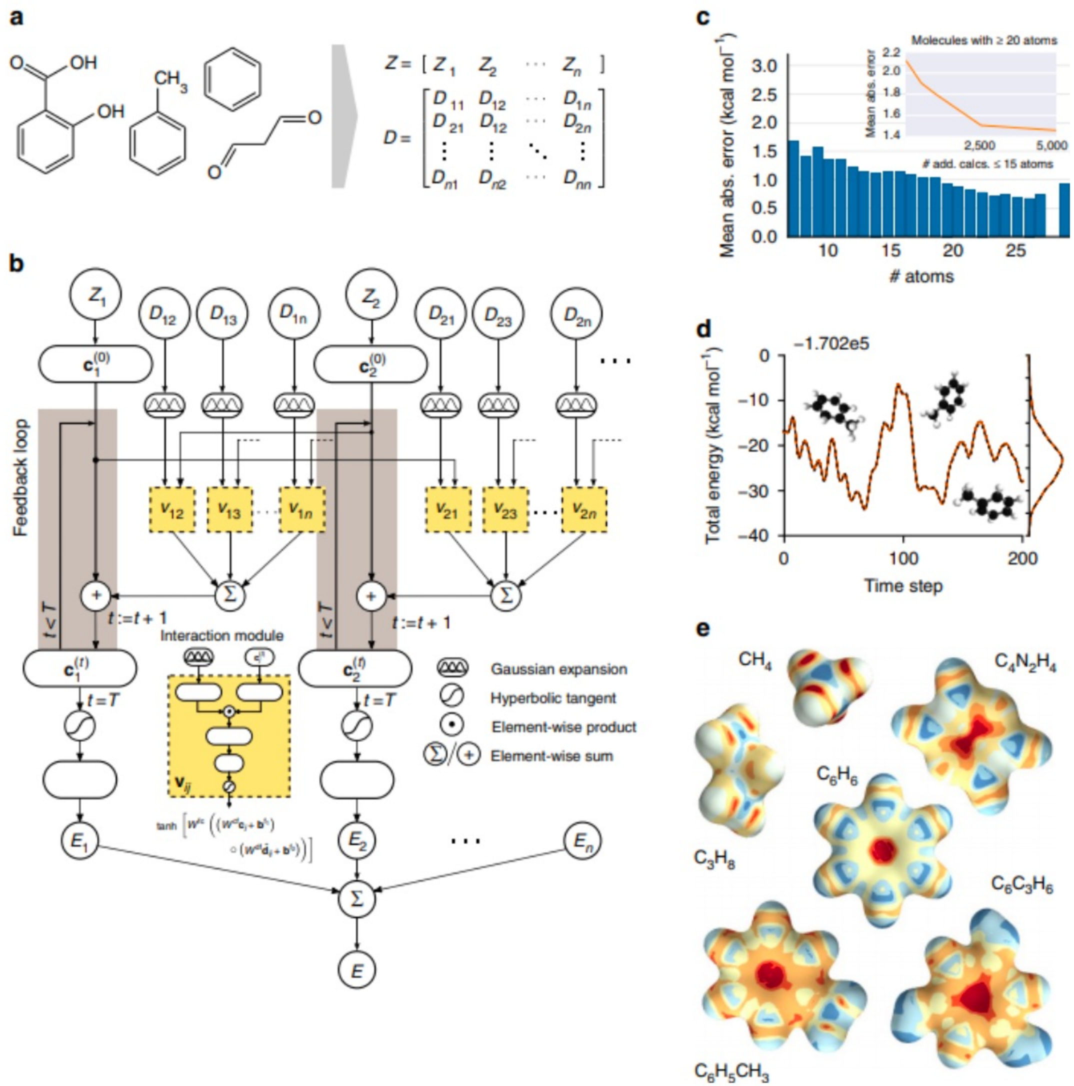
end for

$\mathbf{r} = \text{Readout}(\mathbf{h})$

return $\text{Classify}(\mathbf{r})$

Image Credit: [I. Henrion]

Quantum chemistry with graph networks



Schutt et al. 2017

Examples

Learning to simulate physics with graph networks

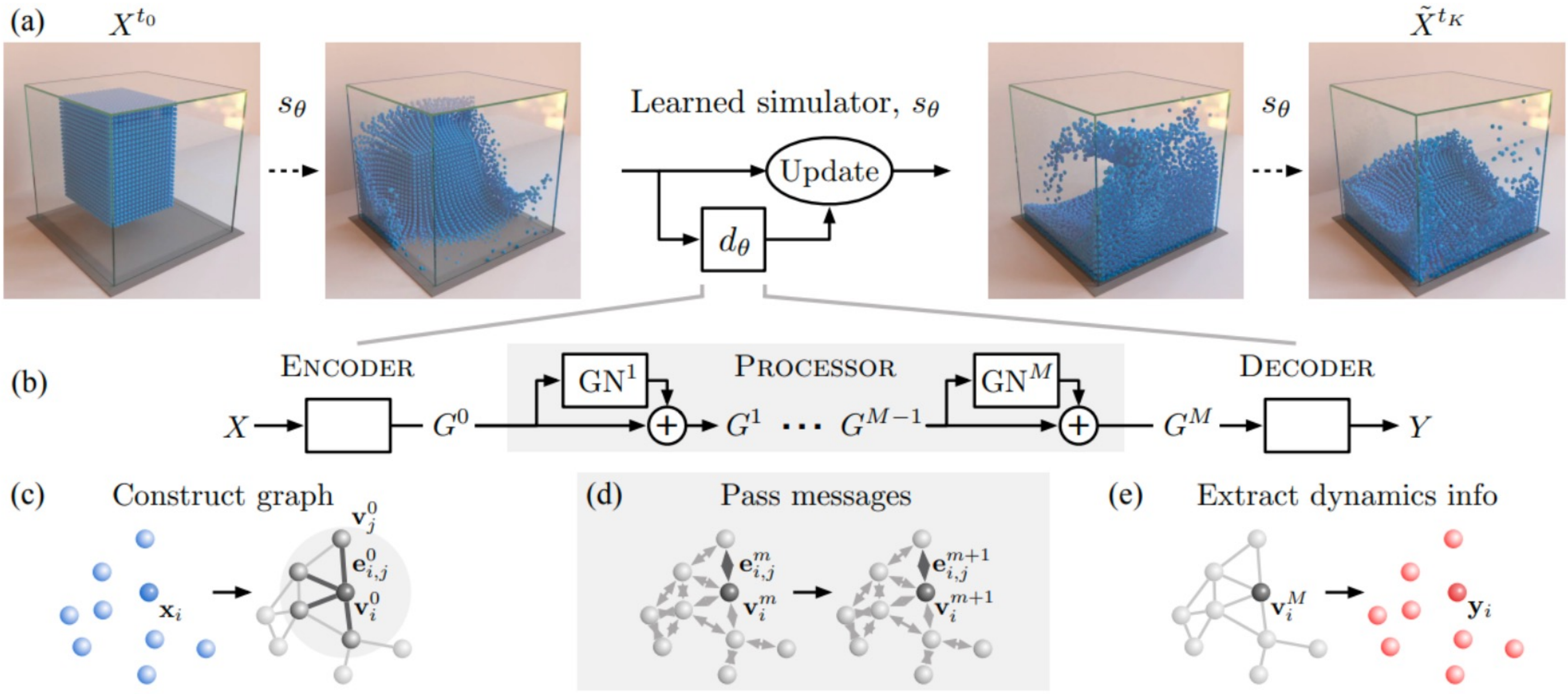


Figure 2. **(a)** Our GNS predicts future states represented as particles using its learned dynamics model, d_θ, and a fixed update procedure. **(b)** The d_θ uses an "encode-process-decode" scheme, which computes dynamics information, Y, from input state, X. **(c)** The ENCODER constructs latent graph, G^0, from the input state, X. **(d)** The PROCESSOR performs M rounds of learned message-passing over the latent graphs, $G^0, \ldots, G^M$. **(e)** The DECODER extracts dynamics information, Y, from the final latent graph, G^M.

Sanchez-Gonzalez et al. [2020](#)

Deep Generative Model Examples

BigGan

(Brock et al, 2018)

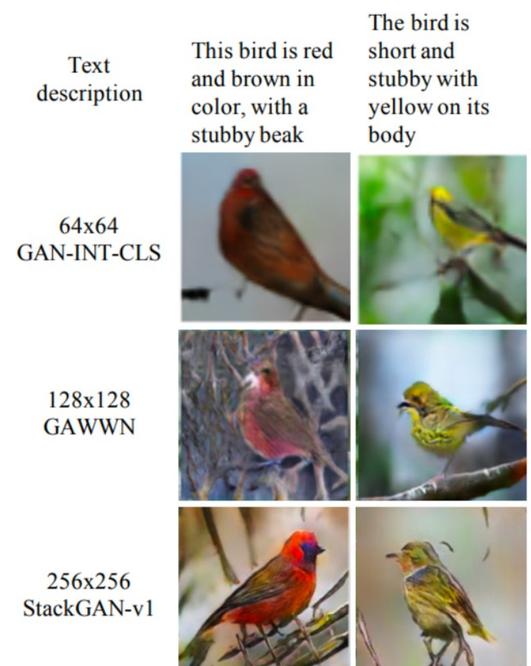
Image-to-Image Translation with CycleGAN

[Zhu et. al. 2017]

Single Image Super-Resolution (Ledig et al, 2016)

Simulate future trajectories of environments based on actions for planning.
(Finn et al, 2016)

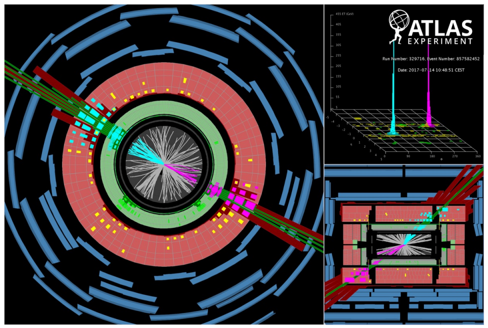
Text-to-Image Synthesis with StackGAN
(Zhang et. al. 2017)

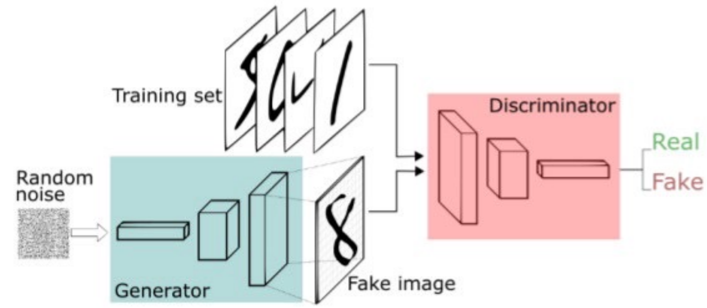


Generative Models

Different Kinds of Generative Models

Generative Models approximate and simulate the data generation process





Scientific Simulators

- Built from science knowledge
- Relatively few parameters, often interpretable
- High Fidelity, often computationally costly

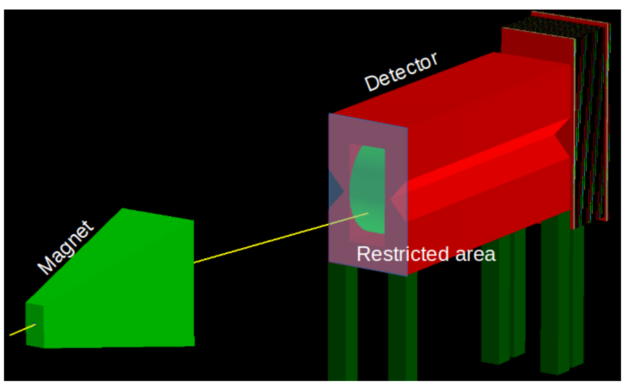
Machine Learning

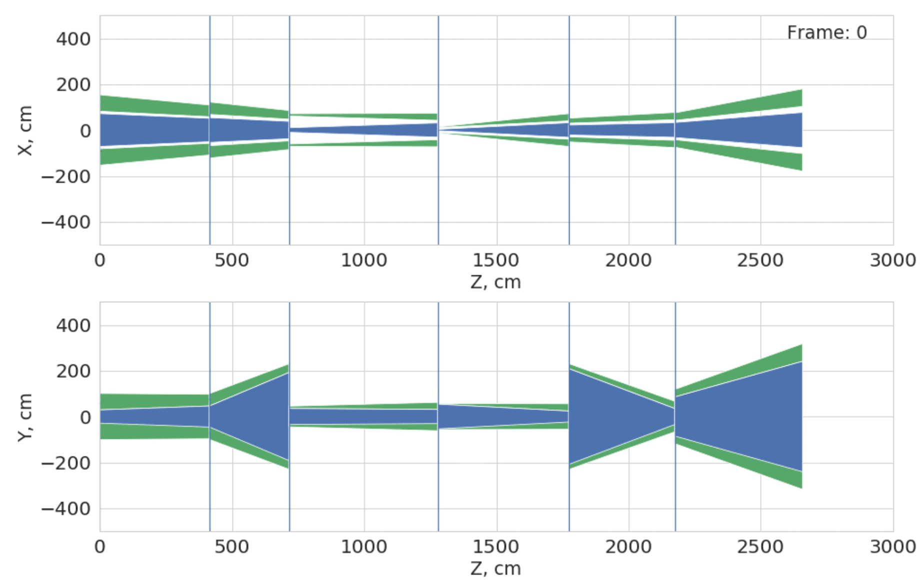
- Fit to data, inductive bias in model design / optimization
- Can have >10^6 parameters, often not interpretable
- Often slow to train, fast to evaluate

GANs for Detector Design

- GAN to emulate detector simulation $\tilde{x} = g(z|\psi)$ given detector parameters ψ (e.g. magnet shape below)
- Design objective C to minimize: $\min_{\psi} \mathbb{E}_{\tilde{x}}[C(\tilde{x} = g(z|\psi))]$
- GAN is differentiable → Minimize with gradient descent

Magnet Optimization





NeurIPS **33**, 14650-14662 (2020)

Variational Autoencoders

Denoising Autoencoders

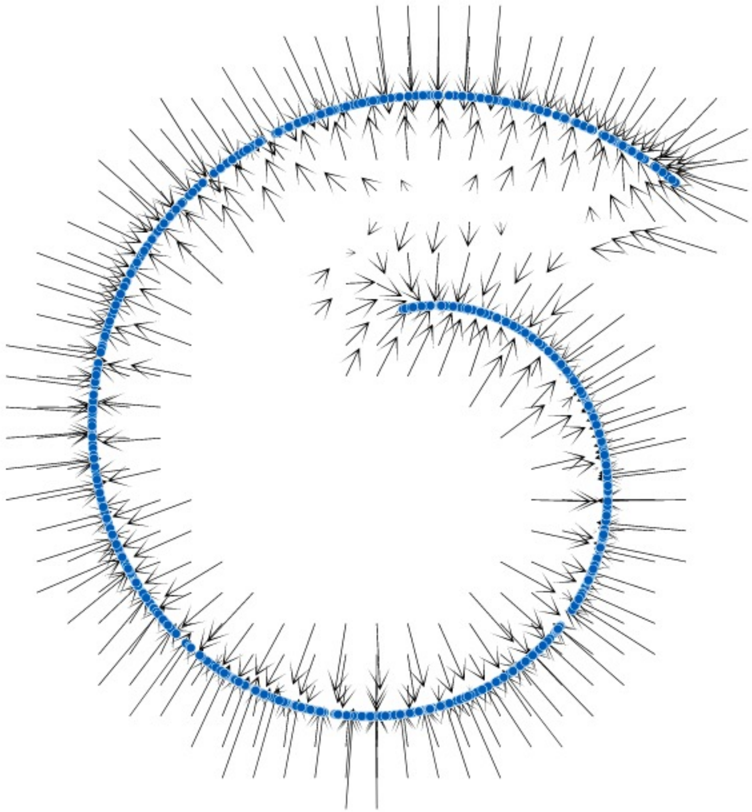
- Learn a mapping from corrupted data space $\tilde{\mathcal{X}}$ back to original data space
  - Mapping $\phi_w(\tilde{\mathcal{X}}) = \mathcal{X}$
  - ϕ_w will be a neural network with parameters w

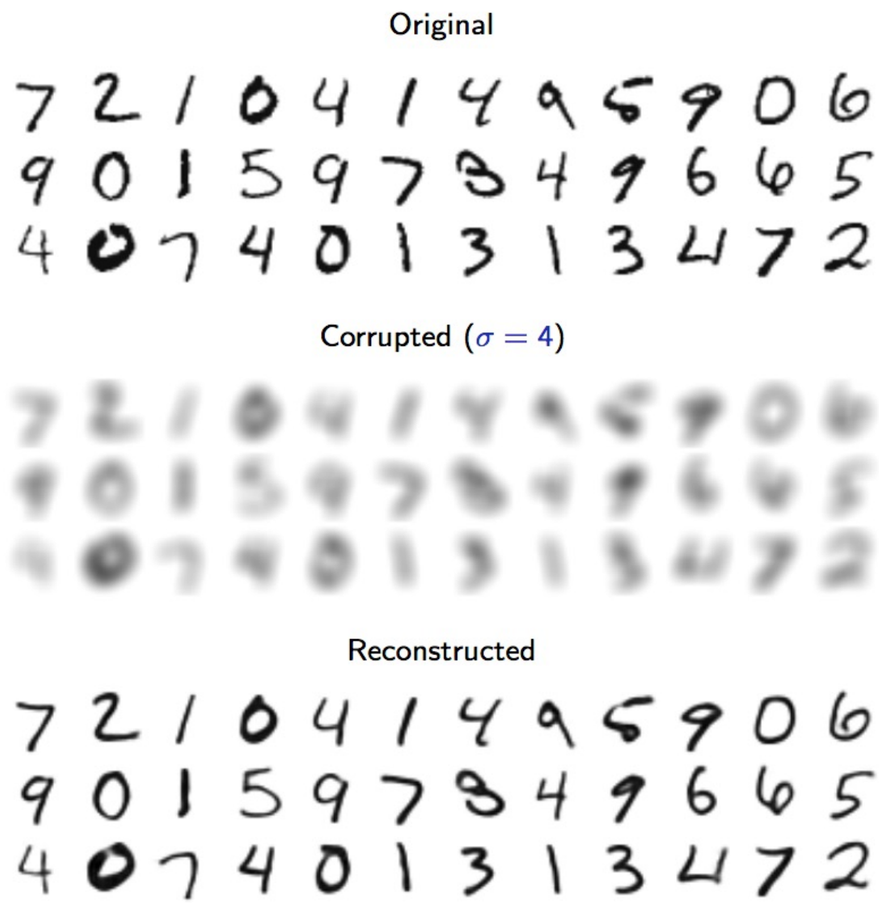
- Loss:

$$\mathrm{L} = \frac{1}{N} \sum_{n} \| x_n - \phi_w(x_n + \epsilon_n) \|$$

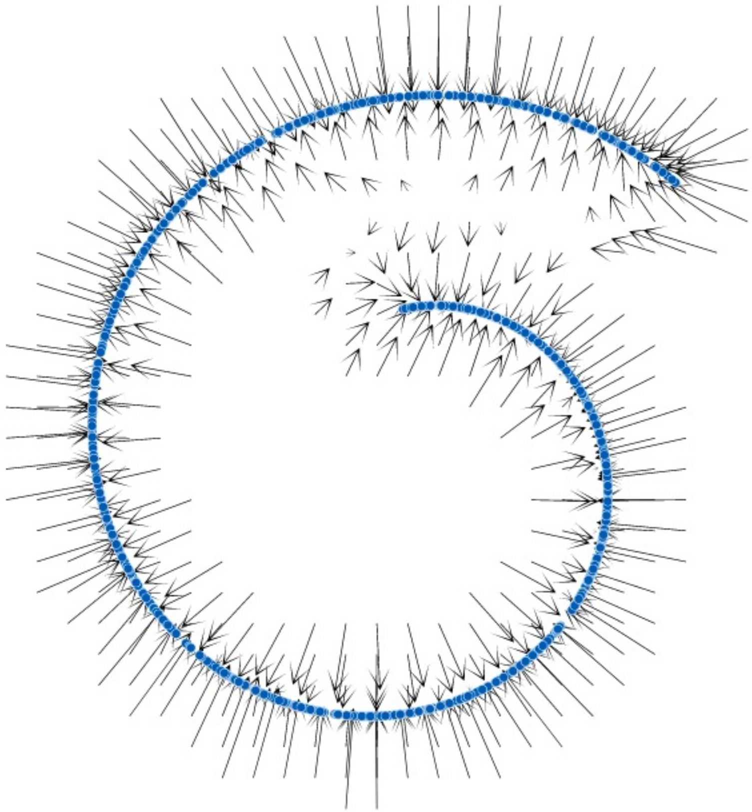
Perturbation, e.g. Gaussian noise

Denoising Autoencoders Examples



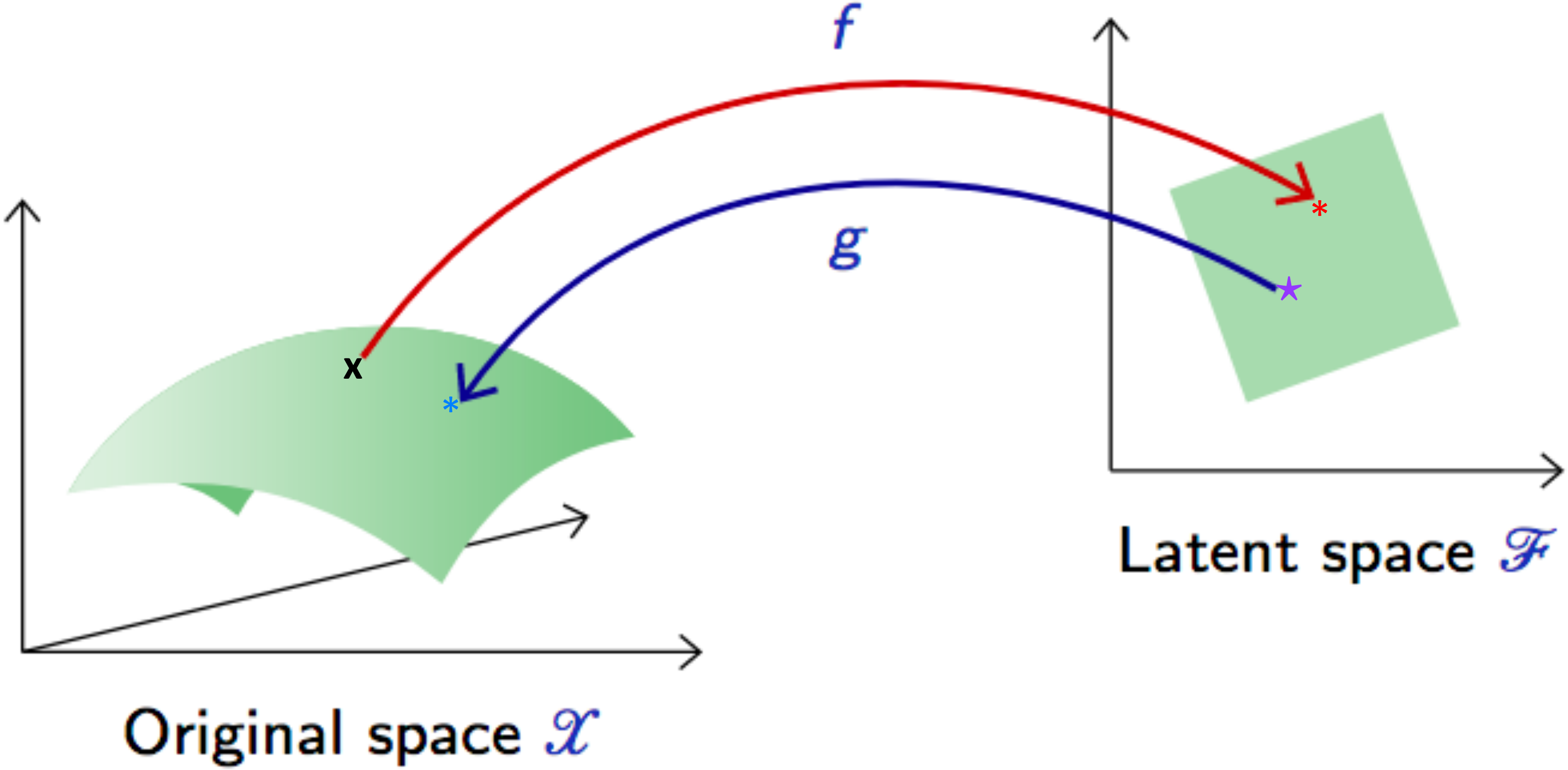


Denoising Autoencoders Examples

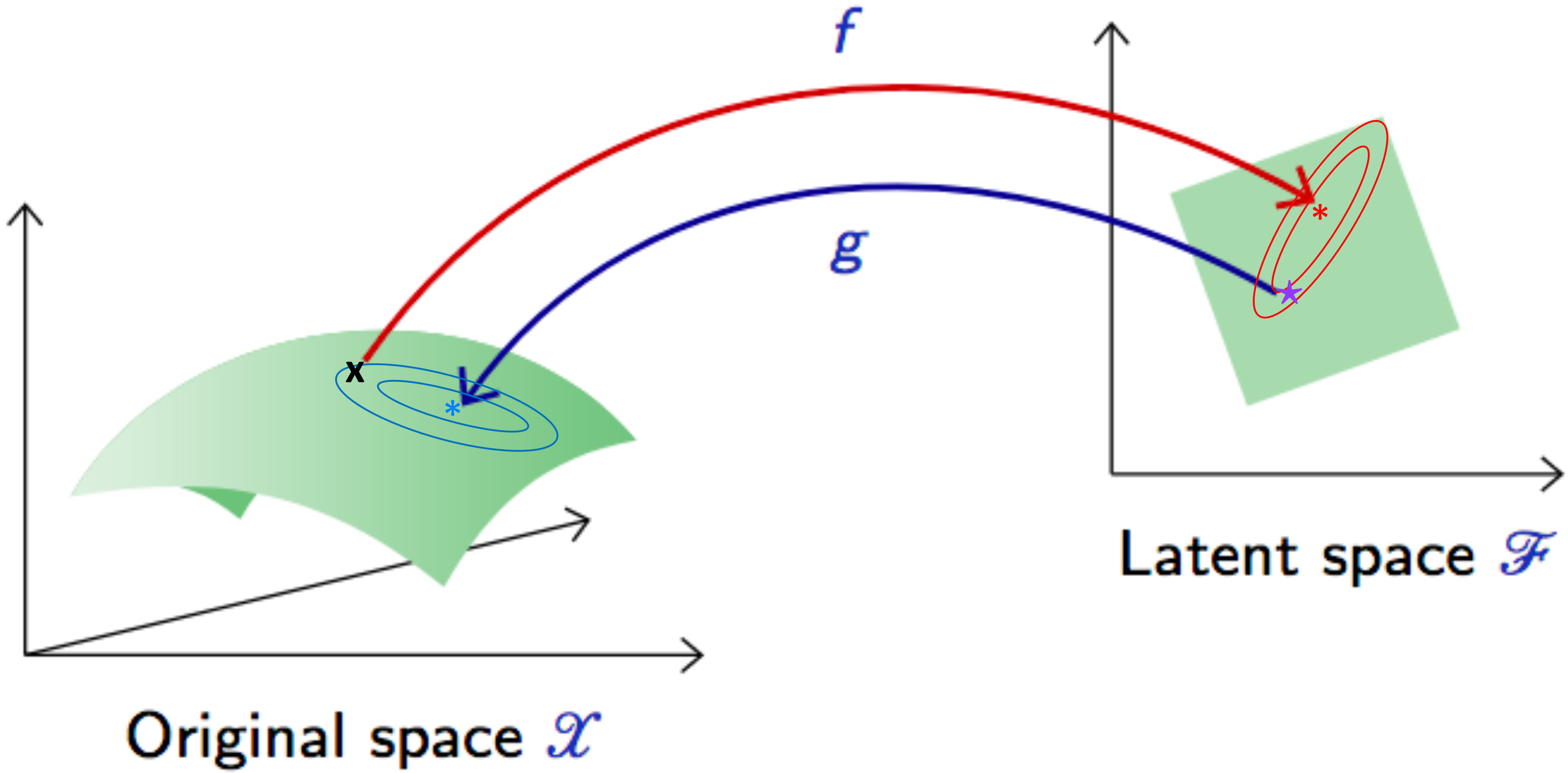


- **Autoencoder learns the average behavior**
- What if we care about these variations?
- Can we add a notion of variation in the autoencoder?

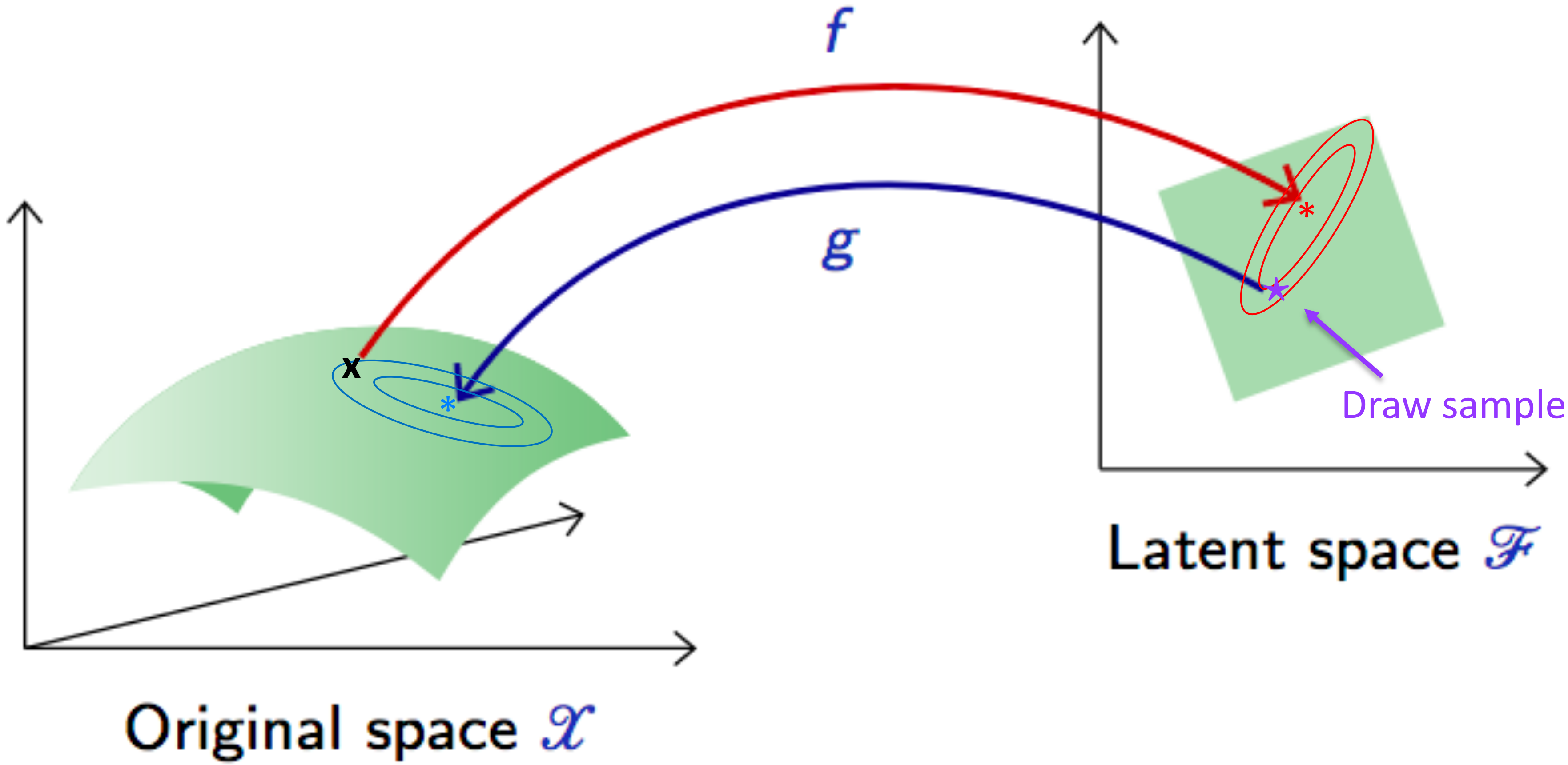
Autoencoder



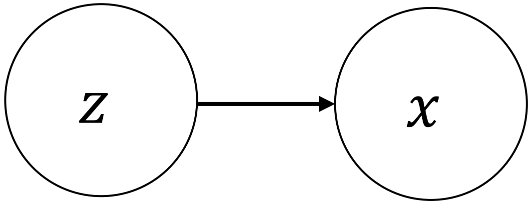
Variational Autoencoder



Variational Autoencoder



Latent Variable Models



- Observed random variable x depends on unobserved latent random variable z

- Joint probability: $p(x, z) = p(x|z)p(z)$

- $p(x|z)$ is stochastic generation process from $z \rightarrow x$

From Deterministic to Probabilistic Autoencoder

- Probabilistic relationship between data and latents

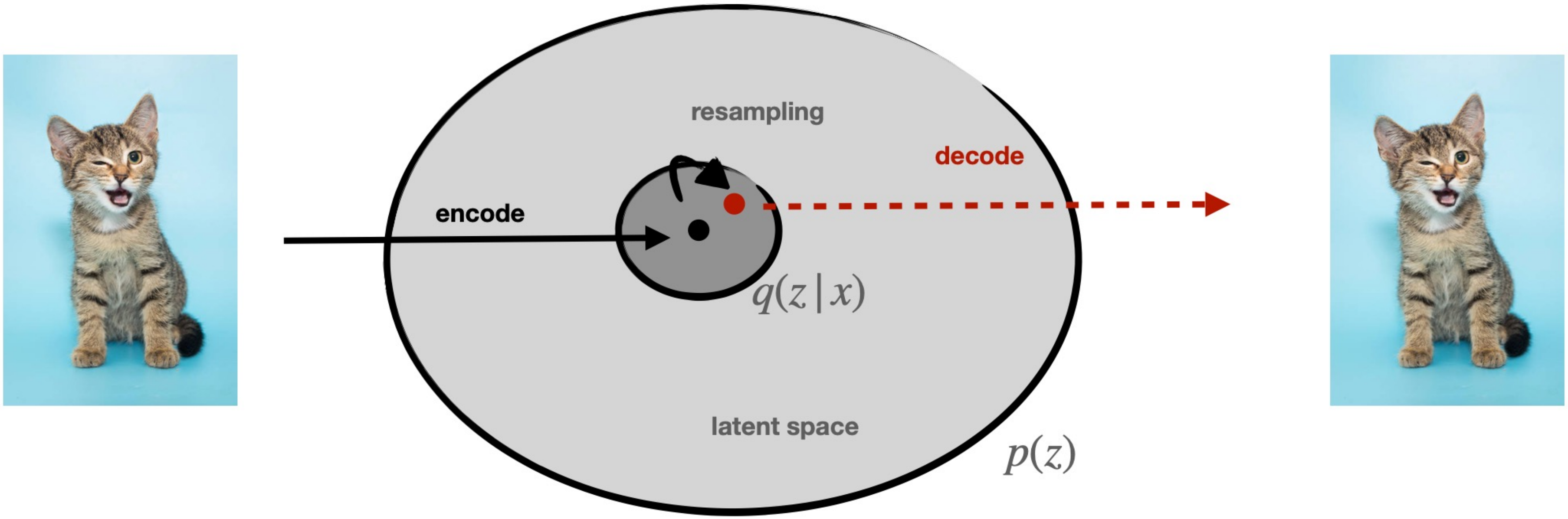
$$x, z \sim p(x, z) = p(x|z)p(z)$$

- Autoencoding

$$x \rightarrow q(z|x) \xRightarrow[sample]{} z \rightarrow p(x|z)$$

  - **Encoder:** Learn what latents can produced data: $q(z|x)$
  - **Decoder:** Learn what data is produced by latent: $p(x|z)$

Variational Autoencoder



- Close-by points must decode to similar images

Image credit: L. Heinrich

How do we design Encoder and Decoder

- Classification / regression models make single predictions…

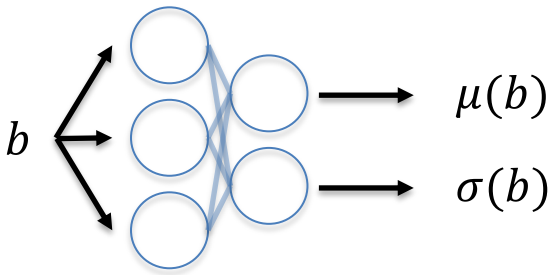
How to model a conditional density $p(a|b)$?

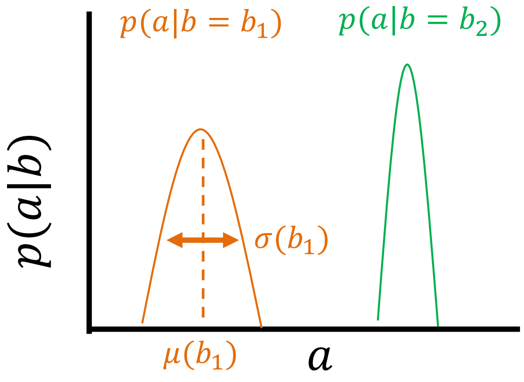
- Assume a known form of density, e.g. normal

$$p(a|b) = \mathcal{N}\big(a; \mu(b), \sigma(b)\big)$$

  – Parameters of density depend on conditioned variable

- Use neural network to model density parameters

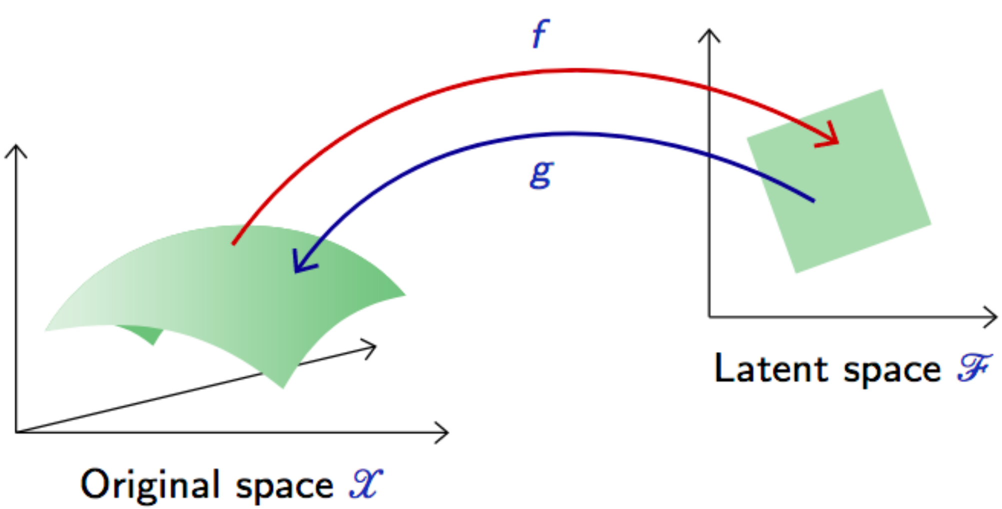




Encoding

- Typical encoder maps input x to "average" point in latent space

$$f(x) = \mu(x)$$

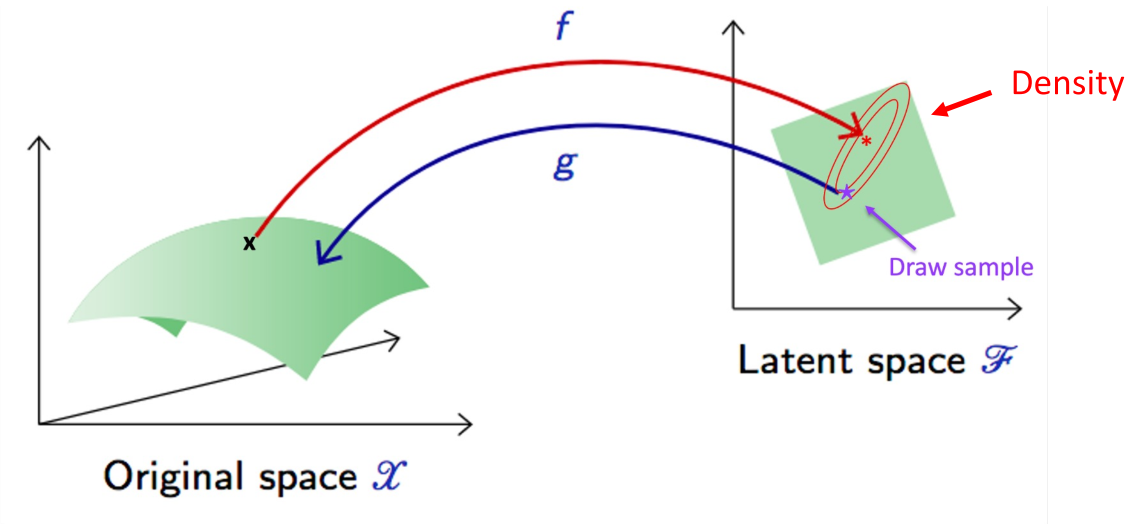


Encoding

- A VAE Encoder has two outputs: mean & variance function

$$f_\psi(x) = \{\mu_\psi(x), \sigma^2_\psi(x)\}$$

ψ are parameters of the NN



Encoding

- A VAE Encoder has two outputs: mean & variance function

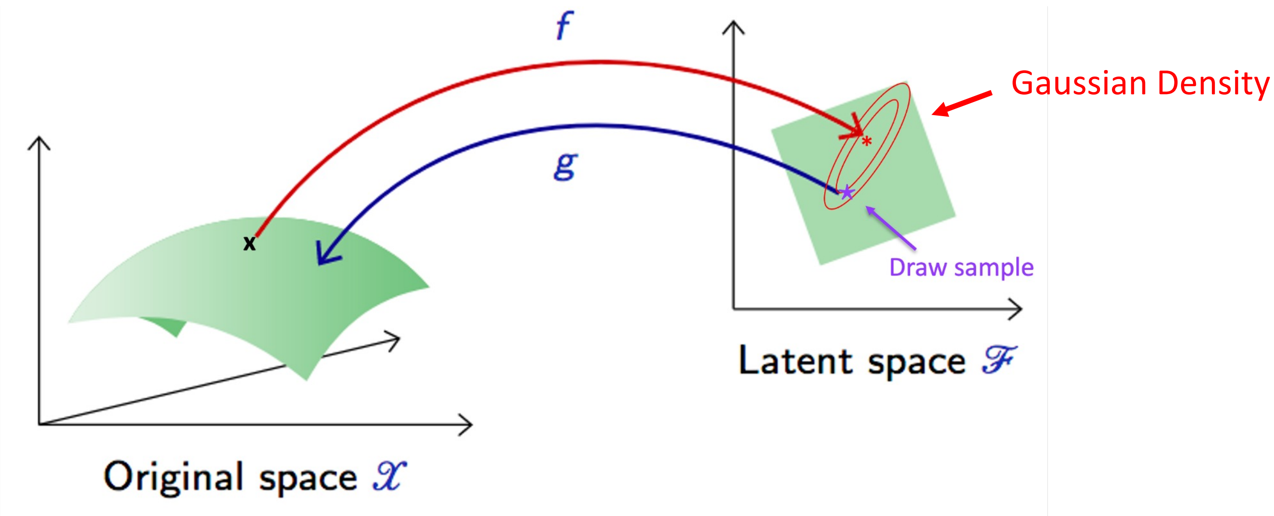
$$f_\psi(x) = \{\mu_\psi(x), \sigma_\psi^2(x)\}$$

ψ are parameters of the NN

- What is the probability of a point in latent space?

$$p_\psi(z|x) = N(z \mid \mu_\psi(x), \sigma_\psi^2(x))$$

Could choose different density
Gaussian is easiest



Re-parameterization trick

- Given $x \sim p(x|\theta)$

- Sometimes, we can rewrite x as a function of the parameters and a simpler distribution without parameter dependence

$$x = g(\epsilon, \theta) \qquad \epsilon \sim p(\epsilon)$$

- Example:

$$x \sim N(x|\mu, \sigma) \ \rightarrow \ x = \sigma * \epsilon + \mu \ \text{ with } \ \epsilon \sim N(0,1)$$

Encoding

- A VAE Encoder has two outputs: mean & variance function

$$f_\psi(x) = \{\mu_\psi(x), \sigma_\psi^2(x)\}$$

ψ are parameters of the NN

- What is the probability of a point in latent space?

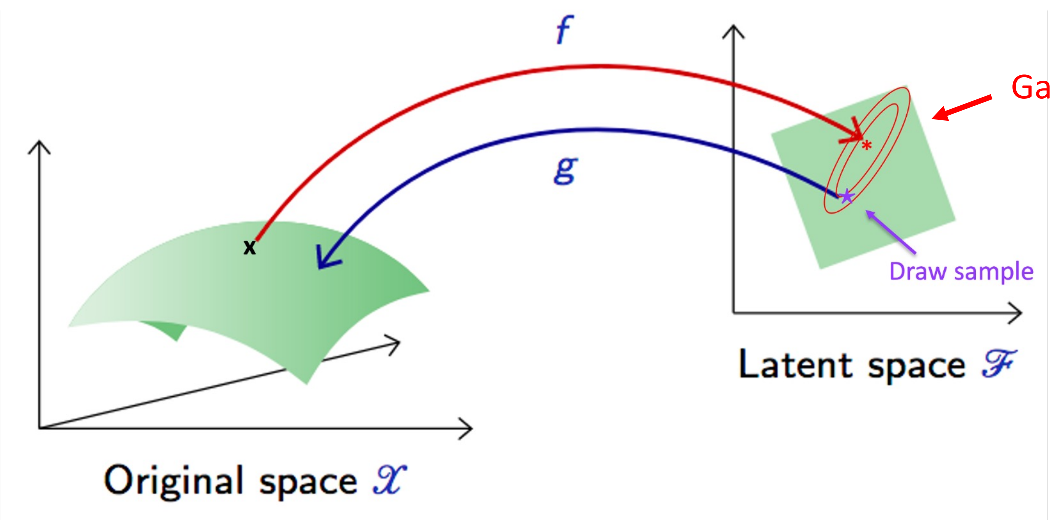
$$p_\psi(z|x) = N(z \mid \mu_\psi(x), \sigma_\psi^2(x))$$

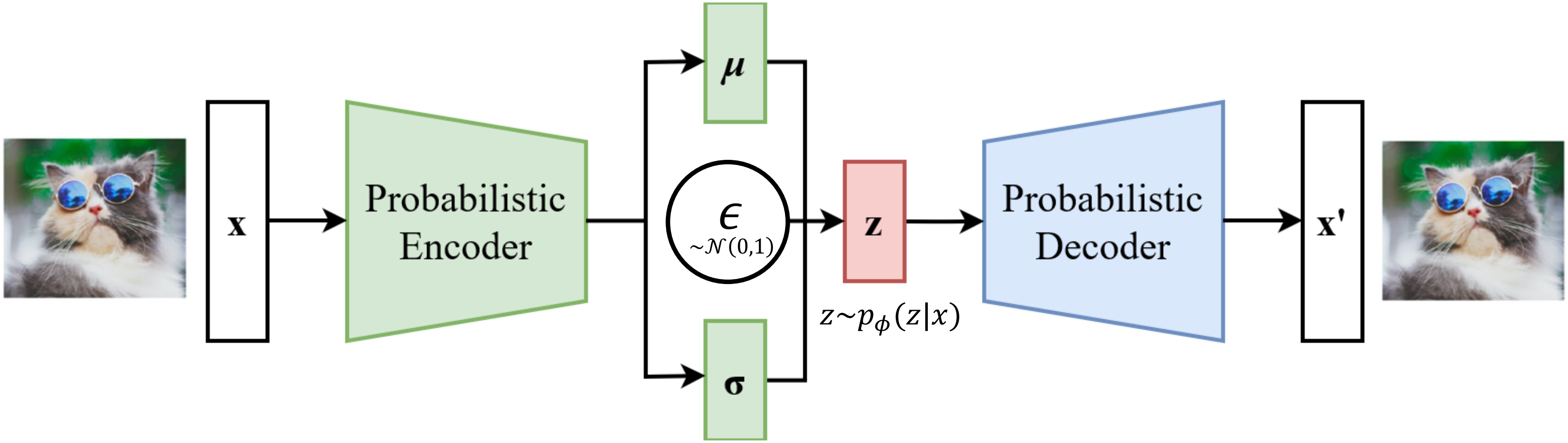
Could choose different density
Gaussian is easiest

- How do we draw a sample in latent space?

$$z = \sigma_\psi(x) * \epsilon + \mu_\psi(x) \qquad \epsilon \sim N(0, I)$$

Re-parameterization trick





Kingma, Welling, 1312.6114
Rezende, Mohamed, Wierstra, 1401.4082

Decoding

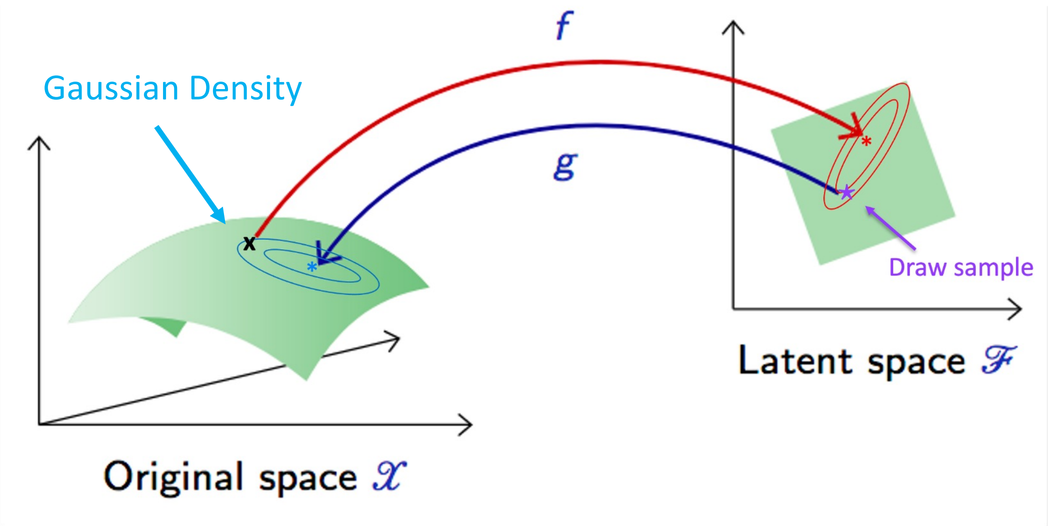
- Same as autoencoder

$$g_\theta(z) \equiv \mu_\theta(z)$$

θ are parameters of the NN

- Likelihood of an observation x

$$p_\theta(x|z) = N(x \mid \mu_\theta(z), I)$$



Decoding

- Same as autoencoder

$$g_\theta(z) \equiv \mu_\theta(z)$$

θ are parameters of the NN

- Likelihood of an observation x

$$p_\theta(x|z) = N(x \mid \mu_\theta(z), I)$$

- "**Reconstruction Loss**": Maximum likelihood

$$L_{reco} = \mathbb{E}_{z\sim q(z|x)}[\log p(x|z)]$$

Decoding

- Same as autoencoder

$$g_\theta(z) \equiv \mu_\theta(z)$$

θ are parameters of the NN

- Likelihood of an observation x

$$p_\theta(x|z) = N(x \mid \mu_\theta(z), I)$$

- "**Reconstruction Loss**": Maximum likelihood

$$L_{reco} = \mathbb{E}_{z\sim q(z|x)}[\log p(x|z)] \approx \frac{1}{N} \sum_{z_i \sim q(z|x)} \log N(x \mid g_\theta(z_i), I)$$

Decoding

- Same as autoencoder

$$g_\theta(z) \equiv \mu_\theta(z)$$

θ are parameters of the NN

- Likelihood of an observation x

$$p_\theta(x|z) = N(x \mid \mu_\theta(z), I)$$

- "**Reconstruction Loss**": Maximum likelihood

$$L_{reco} = \mathbb{E}_{z\sim q(z|x)}[\log p(x|z)] \approx -\frac{1}{N}\sum_{z_i\sim q(z|x)} \left(x - g_\theta(z_i)\right)^2$$

Same as the autoencoder loss

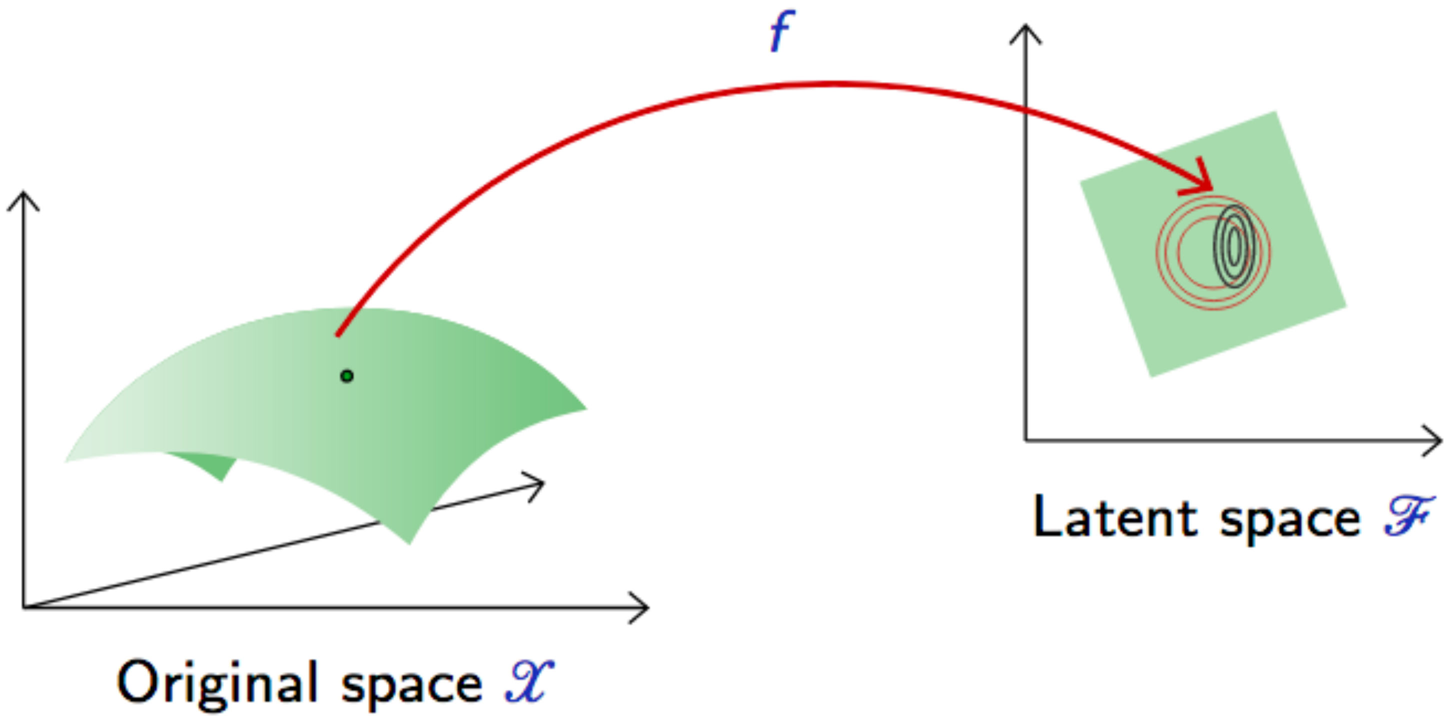
Variational Autoencoder Training Loss

- How do we make sure system doesn't collapse to an autoencoder (i.e. VAE encoder only predicts mean)?

Variational Autoencoder Training Loss

- How do we make sure system doesn't collapse to an autoencoder (i.e. VAE encoder only predicts mean)?

- Use prior $p(z)$ for the latent space distribution, **need to ensure the encoder is consistent with prior**



Variational Autoencoder Training Loss

- Constrain difference between distributions with **Kullback–Leibler divergence**

$$D_{KL}[q(z|x)|p(z)] = \mathbb{E}_{q(z|x)}\left[\log\frac{q(z|x)}{p(z)}\right] = \int q(z|x)\log\frac{q(z|x)}{p(z)}\,dz$$

  - $D_{KL}[q|p] \geq 0$ and is only 0 when $q = p$

Variational Autoencoder Training Loss

- Constrain difference between distributions with **Kullback–Leibler divergence**

$$D_{KL}[q(z|x)|p(z)] = \mathbb{E}_{q(z|x)}\left[\log\frac{q(z|x)}{p(z)}\right] = \int q(z|x)\log\frac{q(z|x)}{p(z)}\,dz$$

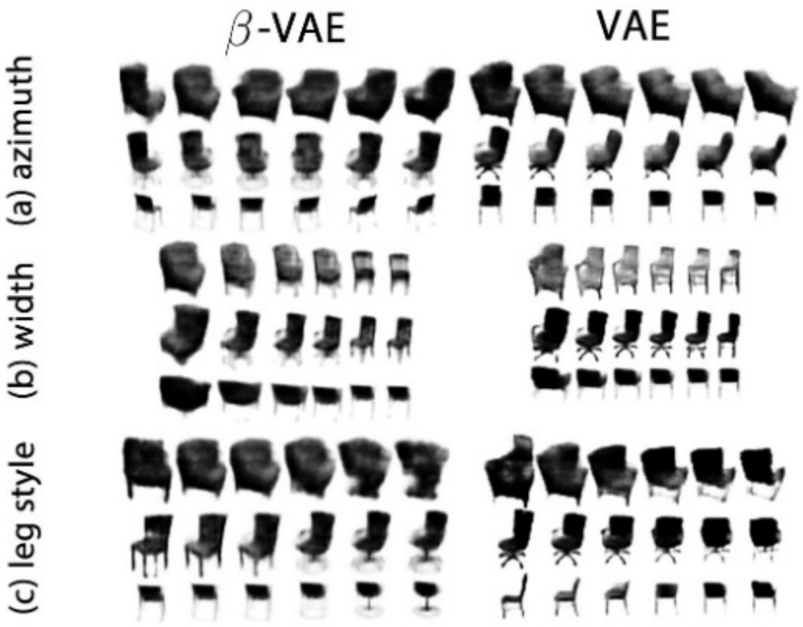
- VAE full objective

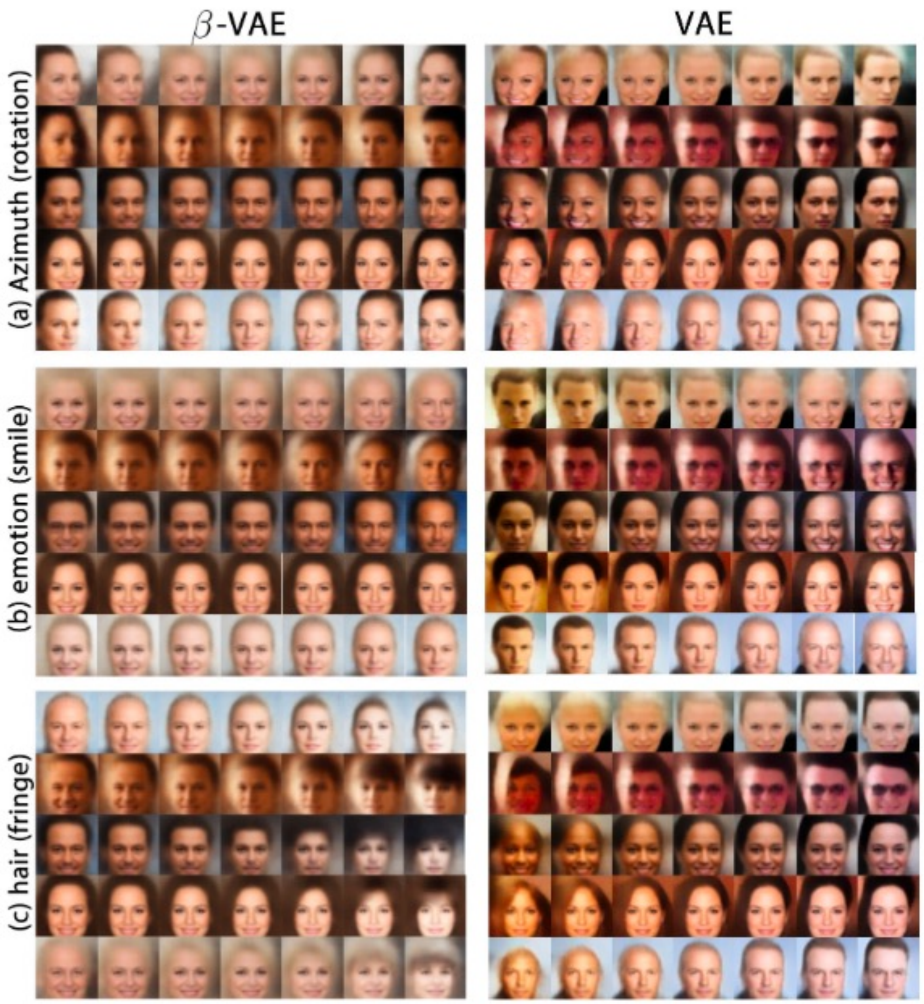
Reconstruction Loss → $\mathbb{E}_{q_\psi(z|x)}[\log p_\theta(x|z)]$

Regularization of Encoder → $D_{KL}[q_\psi(z|x)|p(z)]$

$$\max_{\theta,\psi} L(\theta,\psi) = \max_{\theta,\psi}\left[\mathbb{E}_{q_\psi(z|x)}[\log p_\theta(x|z)] - D_{KL}[q_\psi(z|x)|p(z)]\right]$$

Examples

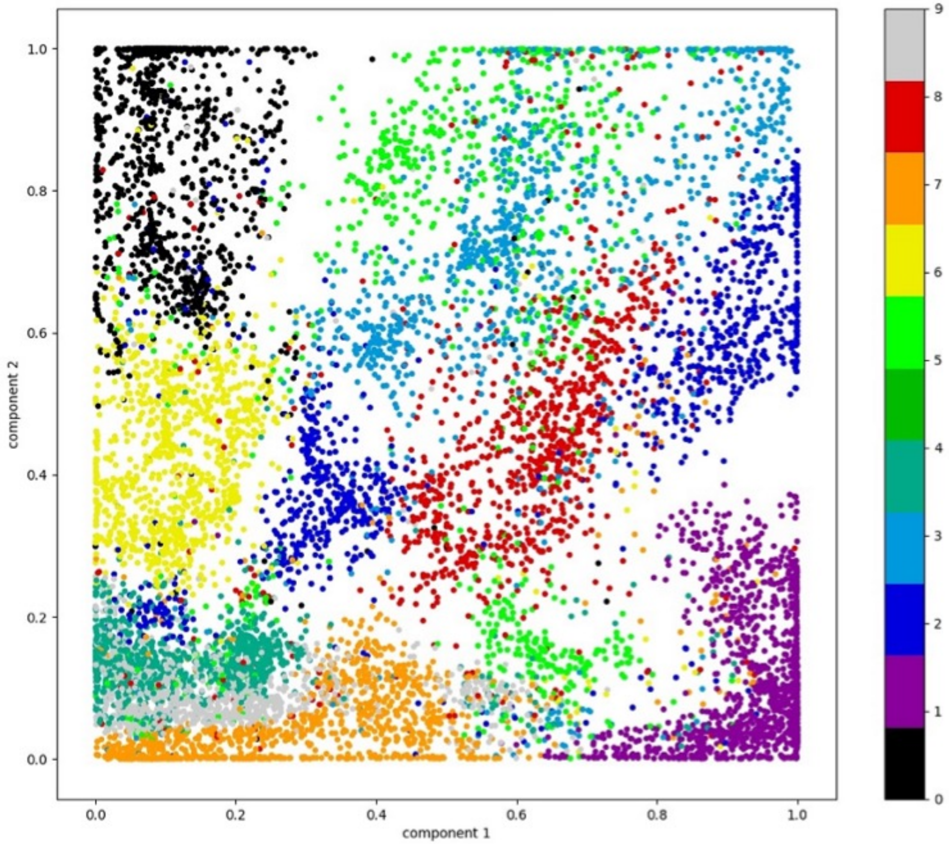




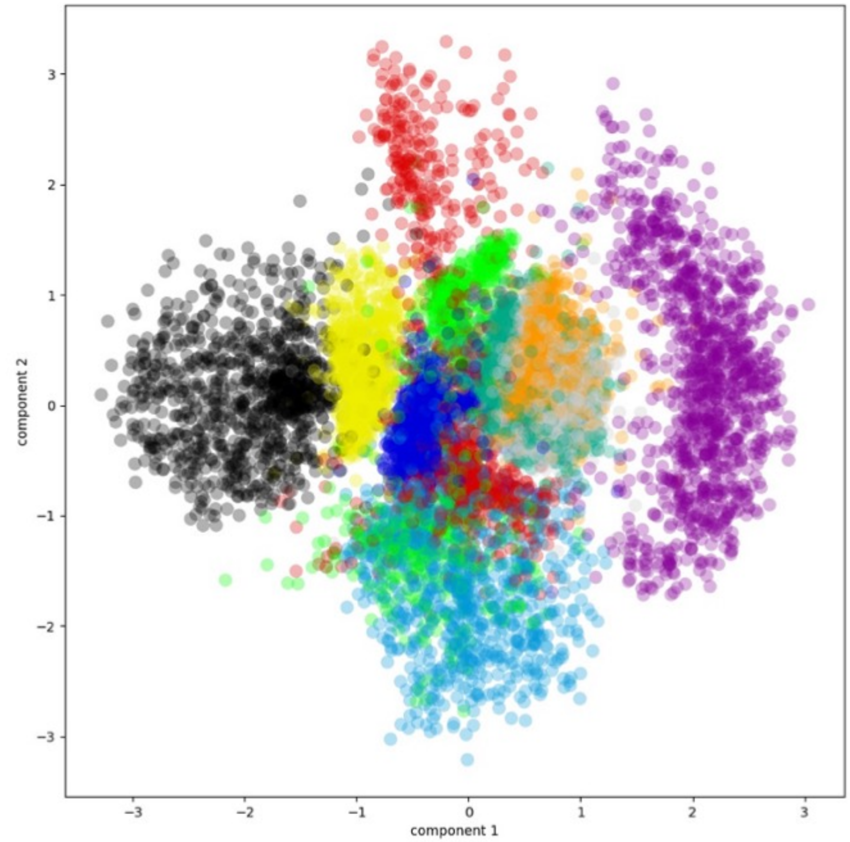
Higgins et al., [2017]

Comparing Latent Spaces

Autoencoder



Variational Autoencoder



Data: MNIST data set of hand-written digits