Lecture II:
Introduction to Deep Learning

Michael Kagan

SLAC

CERN Openlab Summer Student Lectures
July 13, 2023

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**.

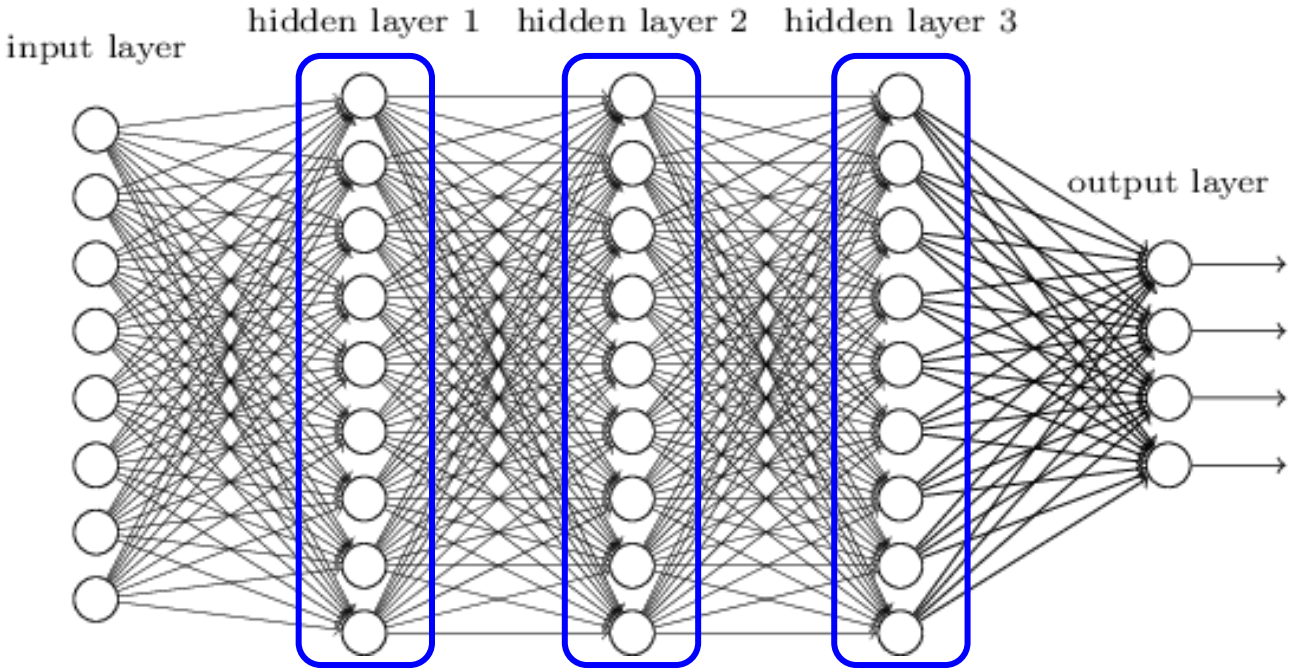
- Yann LeCun, 2018

- 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

The Plan

- Deep Learning is a HUGE field
  - O(10,000) papers submitted to conferences
- I'm will condense *some* parts of what you would find in *some lectures* of a Deep Learning course
- Highly recommend taking the time to go more slowly through lectures from a class. Online-available Recommendations:
  - Francois Fleuret course at University of Geneva
  - Gilles Louppe course at University of Liege
  - Yann LeCun & Alfredo Canziani course at NYU

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

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

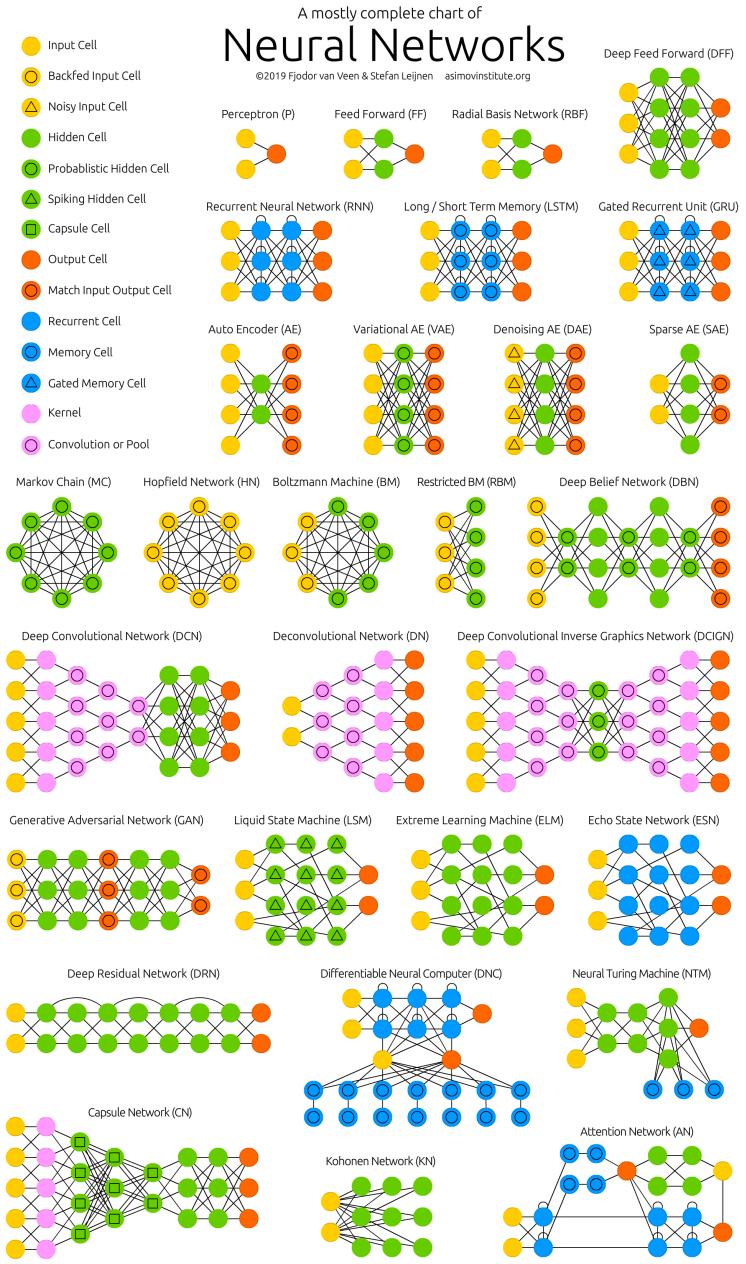


Image credit: [neural-network-zoo]

Neural Network Zoo – "Optimization" Perspective

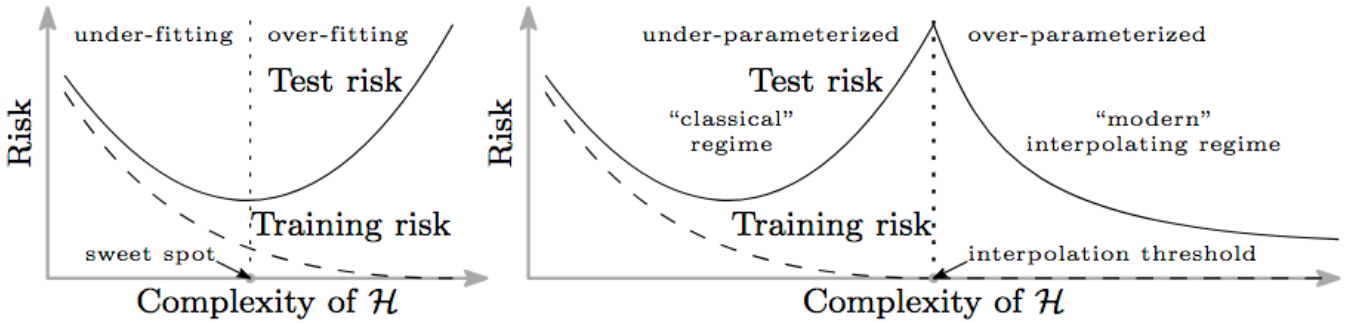
- A single layer network may need a width exponential in D to approximate a depth-D network's output
  - Simplified version of Telgarsky (2015, 2016)

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)

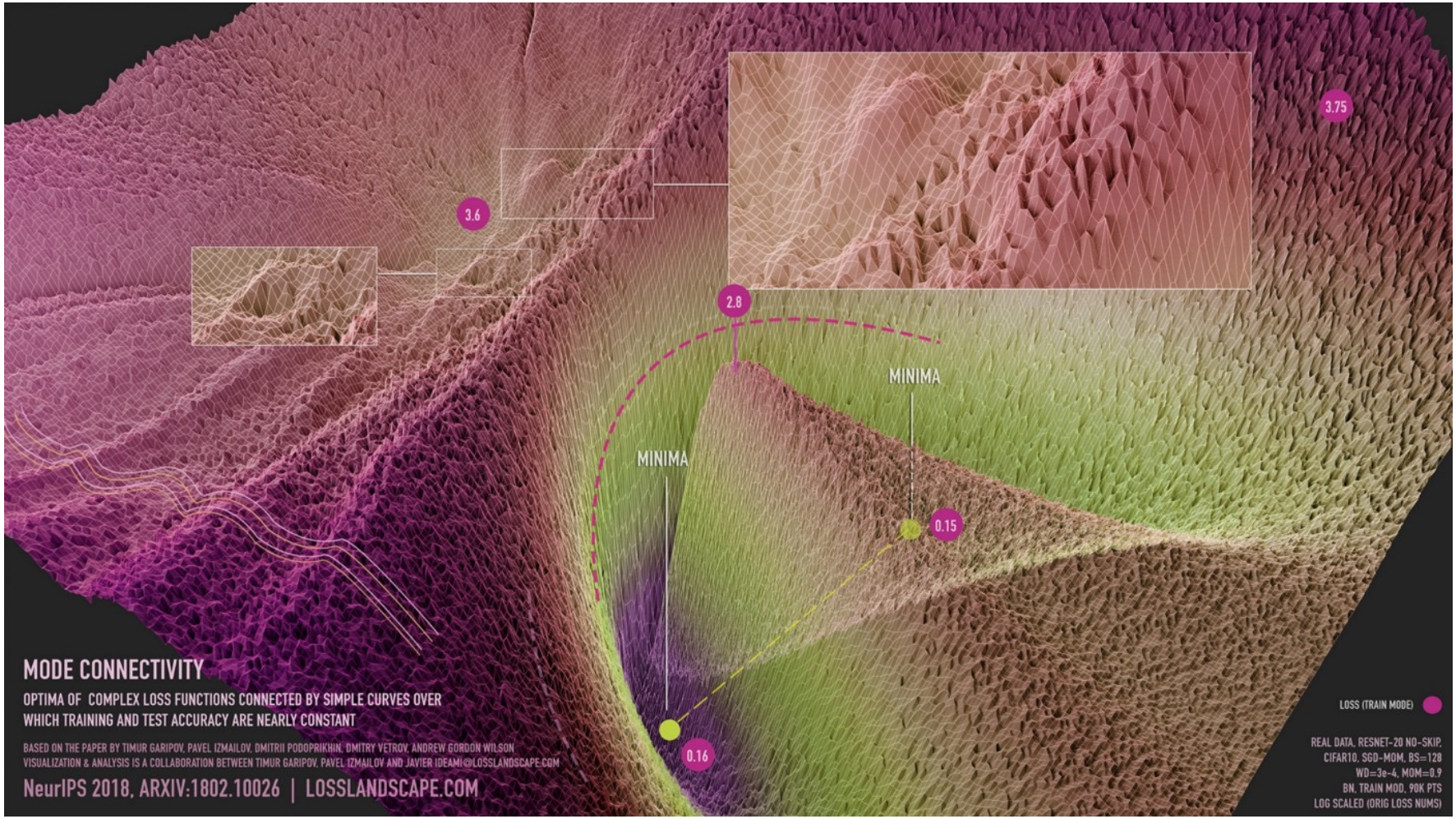
- Over-parametrizing a deep model often improves test performance, contrary to bias-variance tradeoff prediction

  - 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

Deep Neural Networks Loss Landscape

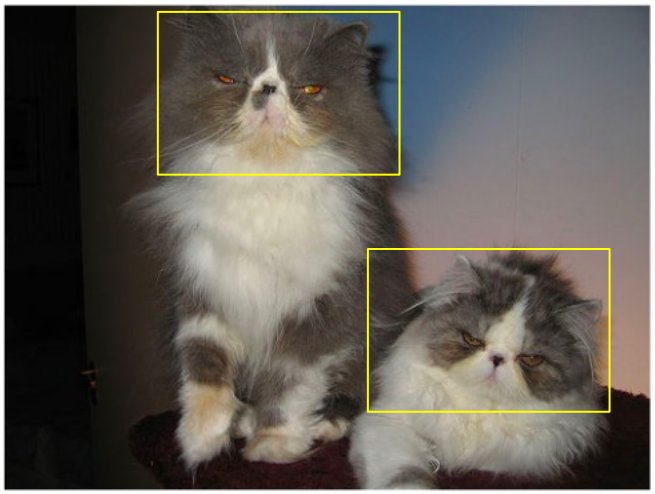


https://arxiv.org/abs/1802.10026

Convolutional Neural Networks

Convolutional Neural Networks

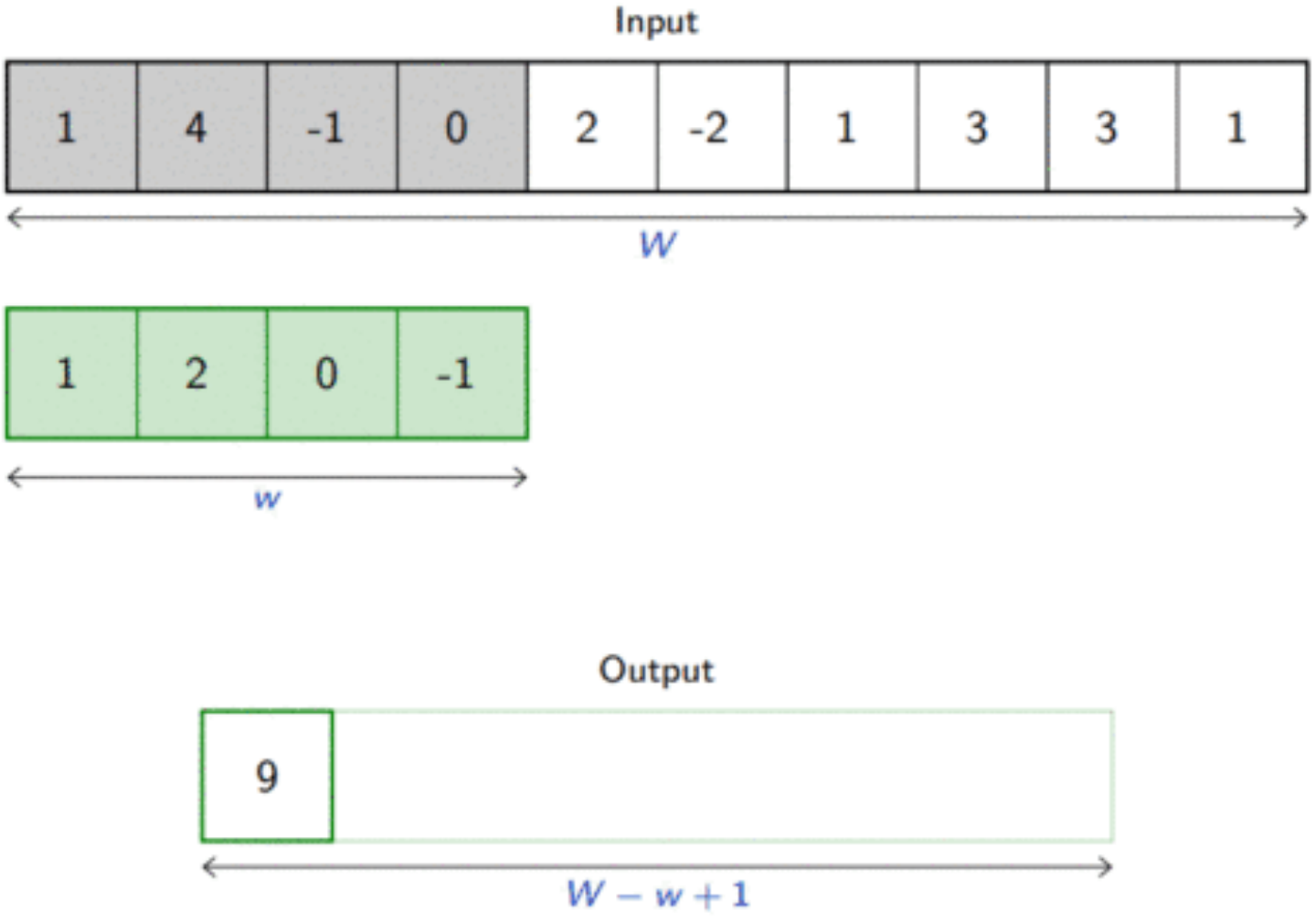
- When the structure of data includes "invariance to translation", a representation meaningful at a certain location can / should be used everywhere



- 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

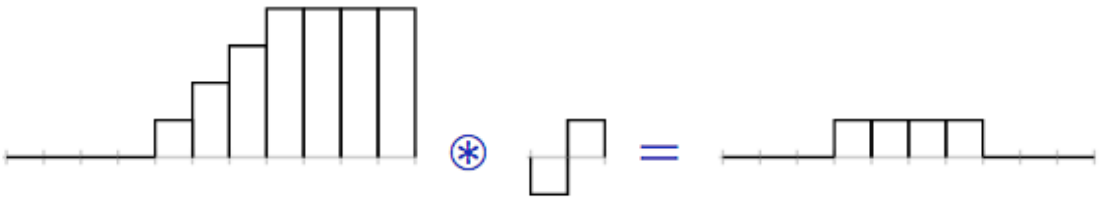


Fleuret, Deep Learning Course

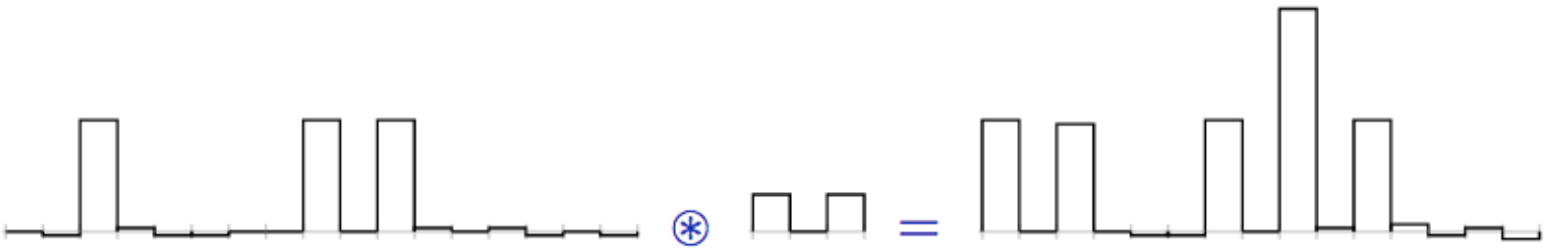
Convolutional Filters

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

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

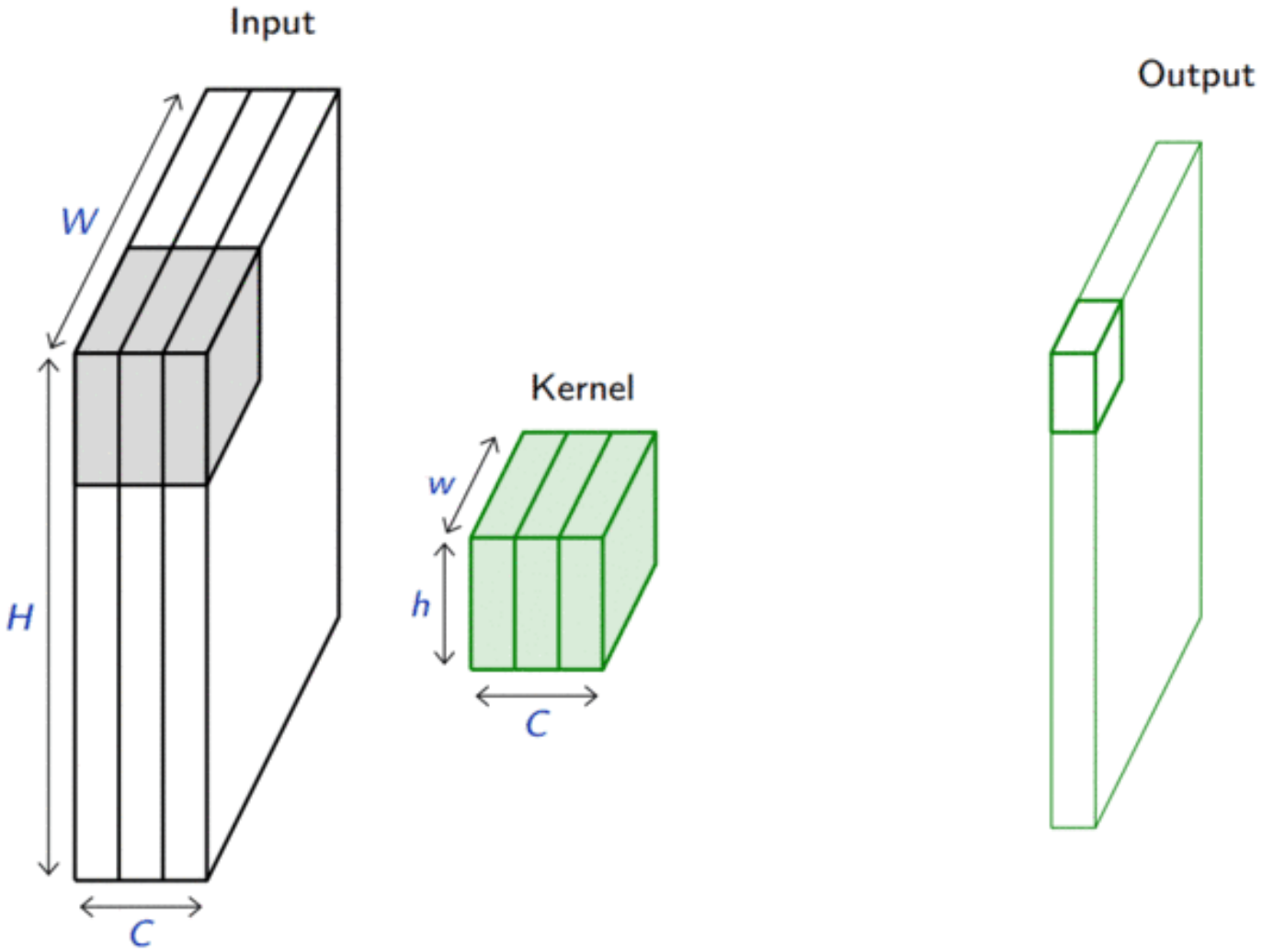


or crude "template matcher", *e.g.*



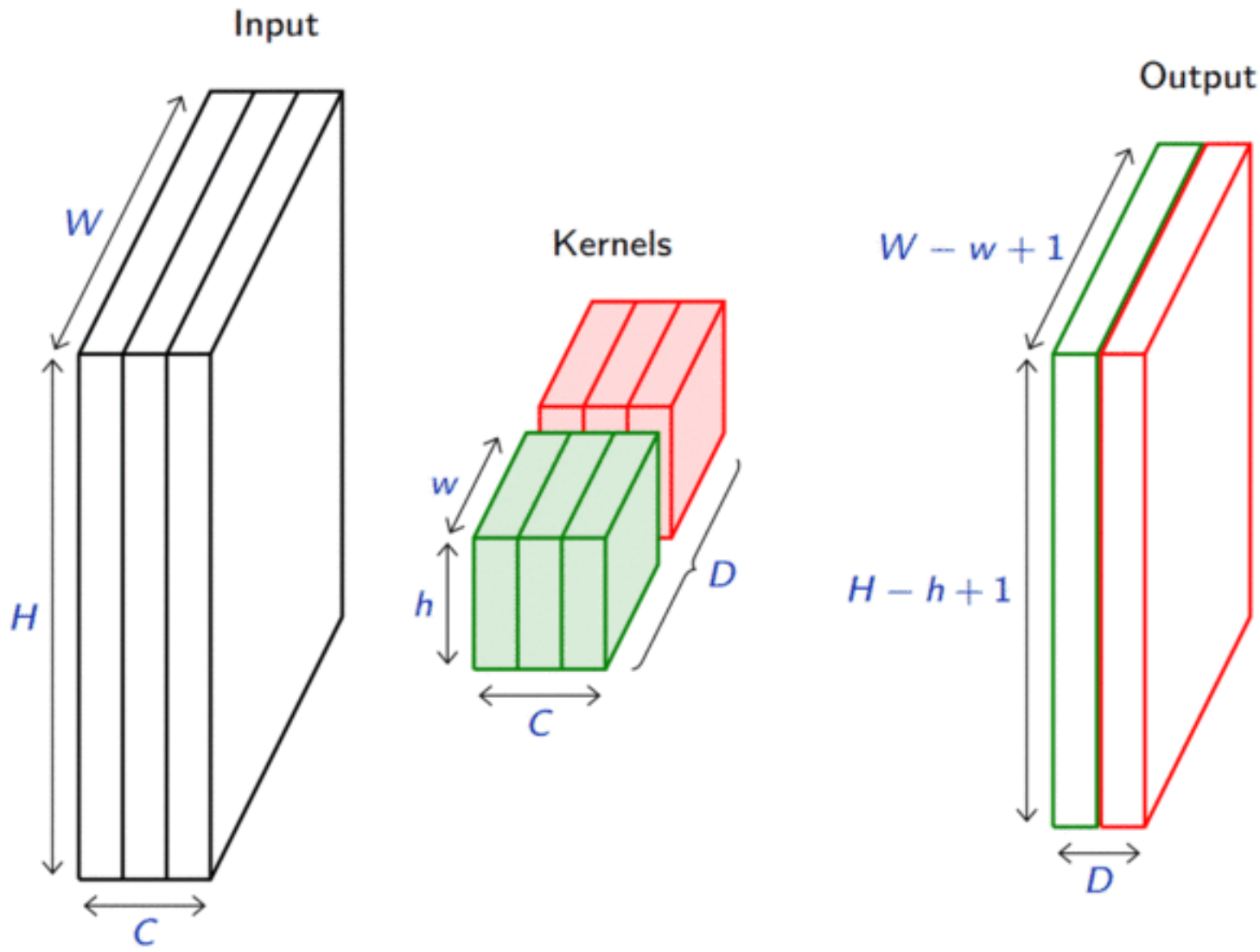
Fleuret, Deep Learning Course

2D Convolution Over Multiple Channels



Fleuret, Deep Learning Course

2D Convolution Over Multiple Channels



Fleuret, Deep Learning Course

2D Convolutional Layer

- Input data (tensor) $\mathbf{x}$ of size $C \times H \times W$
  - C channels (e.g. RGB in images)
- Learnable Kernel $\mathbf{u}$ 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*

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: 256×256×3 RGB image
  - Kernel: 3×3×3 → 27 weights
  - Fully connected layer:
    - 256×256×3 inputs → 256×256×3 outputs → $O(10^{10})$ weights

Y. LeCun et. al. 1998

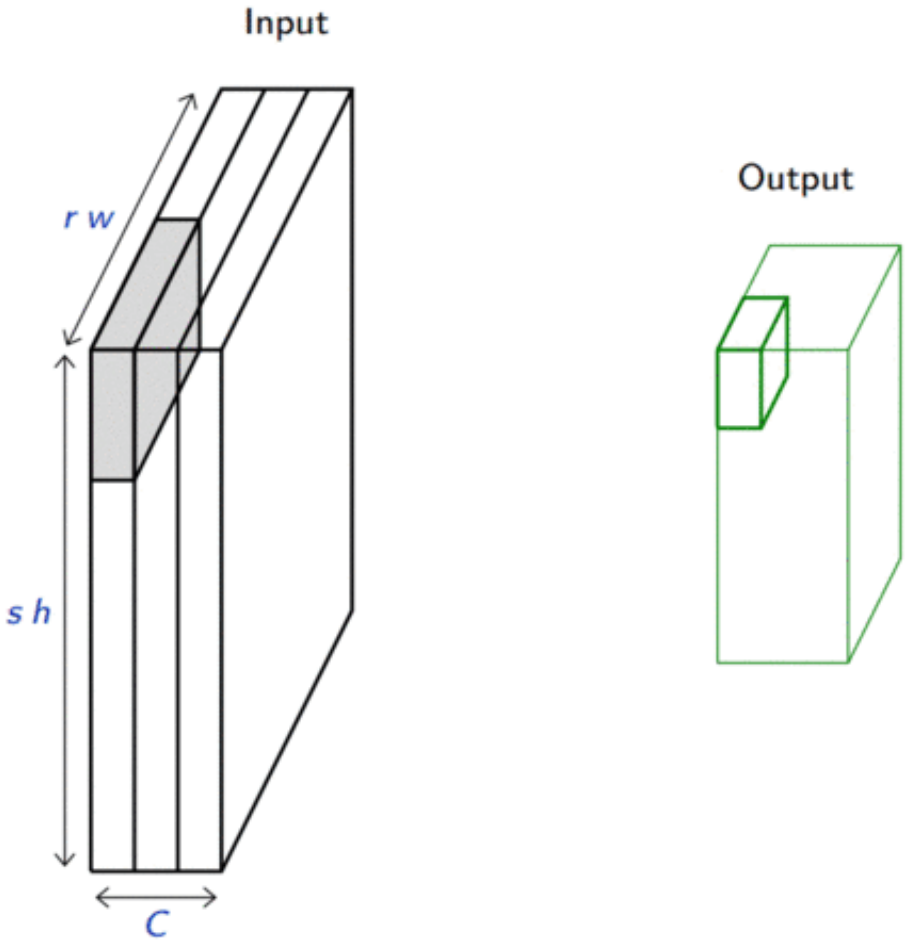
Shared Weights: Economic and Equivariant

- 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 $h \times w$

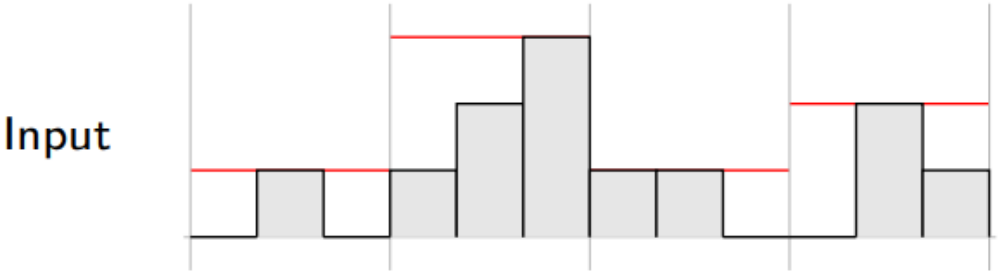


Fleuret, Deep Learning Course

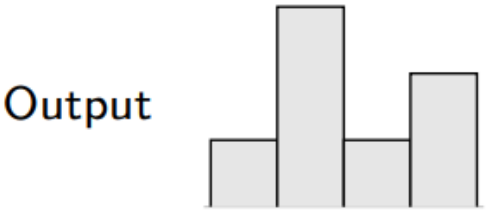
Pooling

- 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

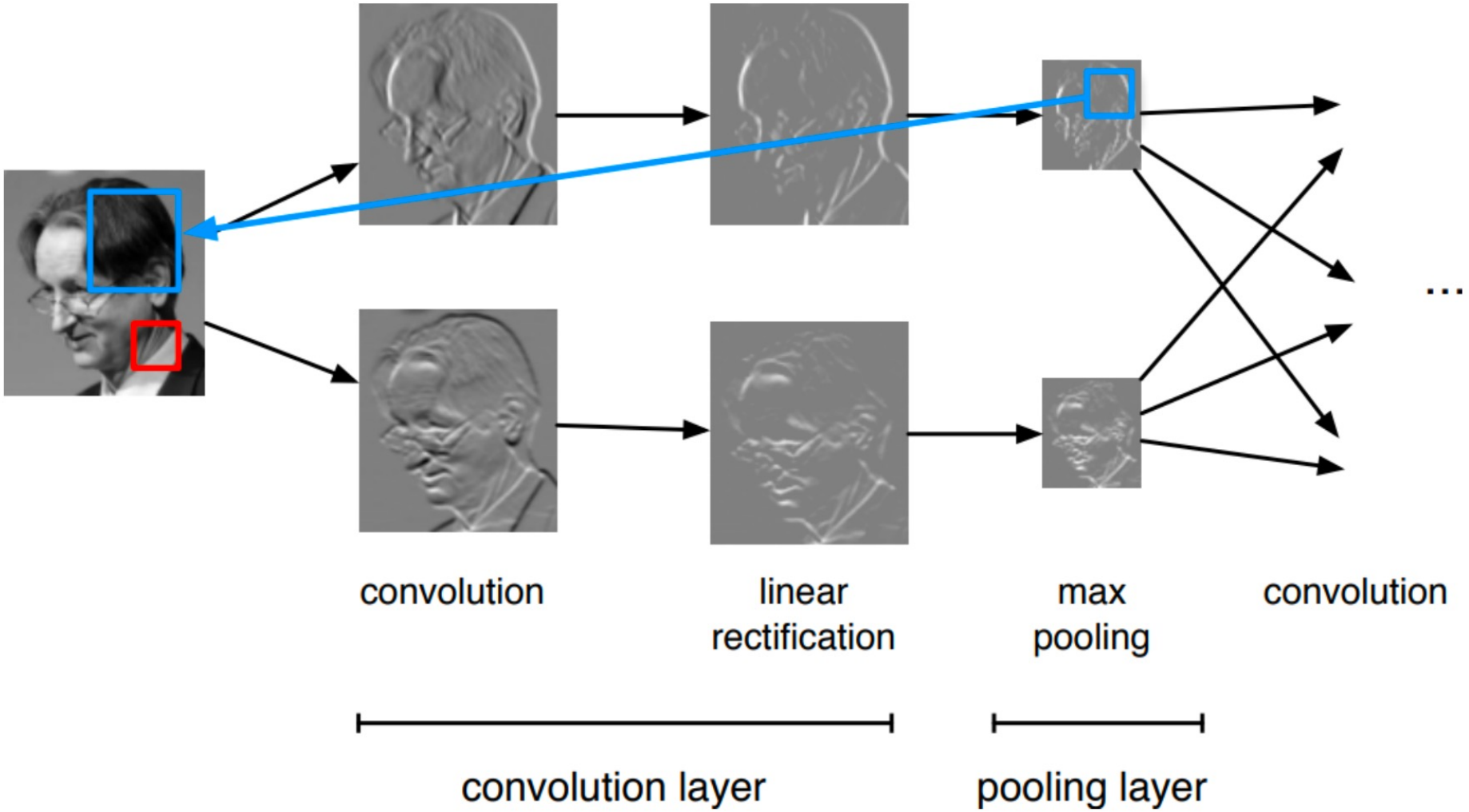


- Invariance to local perturbations

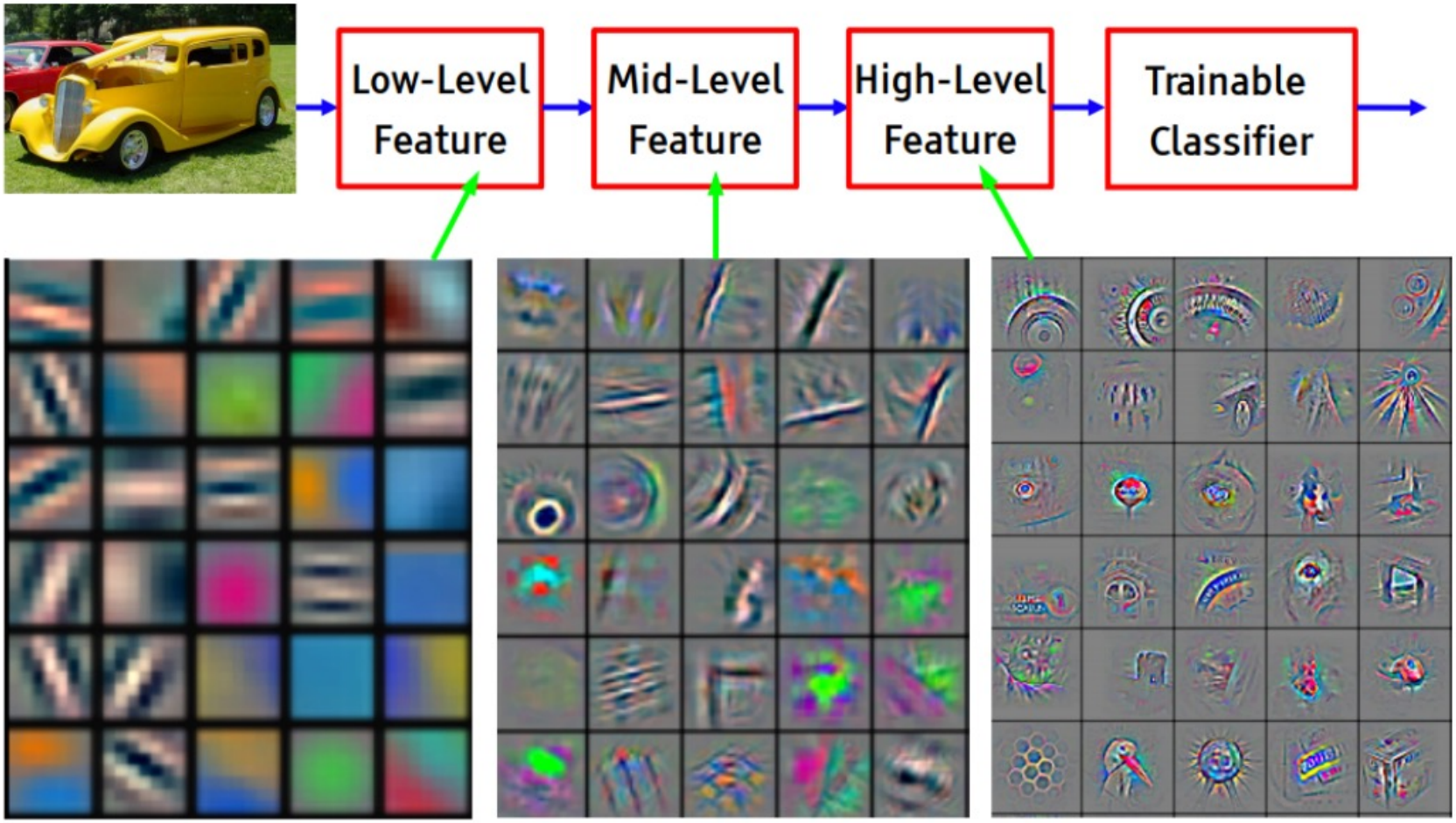


Convolutional Network

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

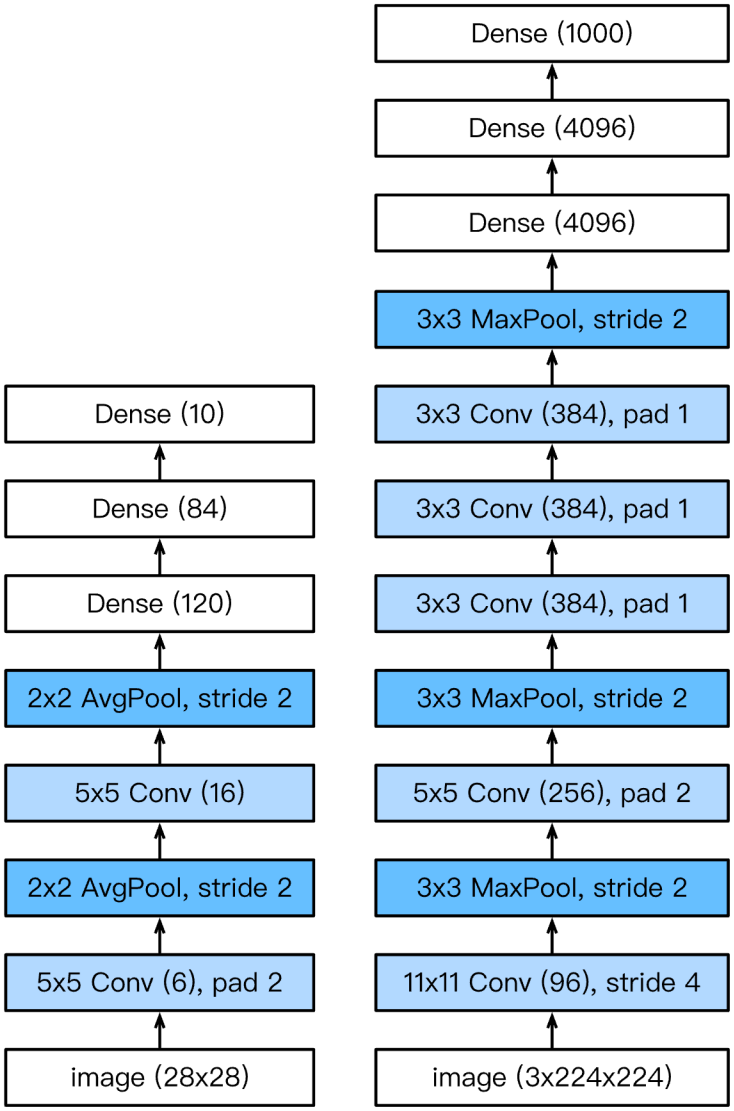


Hierarchical Composition of Features



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

Convolutional Networks



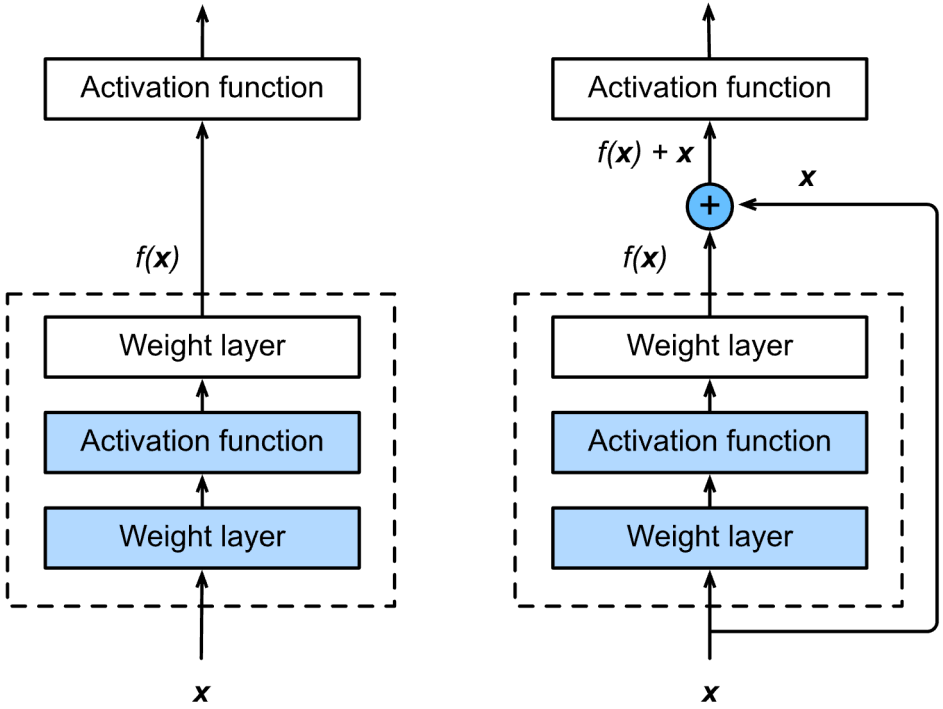
LeNet
(LeCun et al, 1998)

AlexNet
(Krizhevsky et al, 2012)

ImageNet Classification

Residual Connections

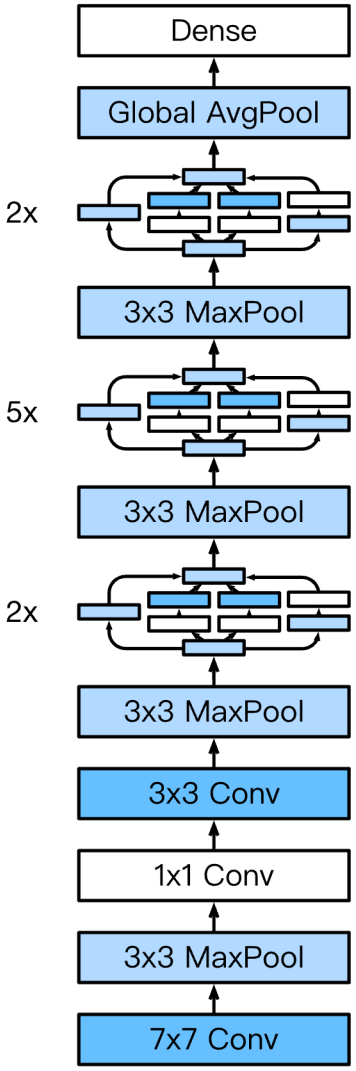
- Training very deep networks is made possible because of the **skip connections** in the residual blocks. Gradients can shortcut the layers and pass through without vanishing.



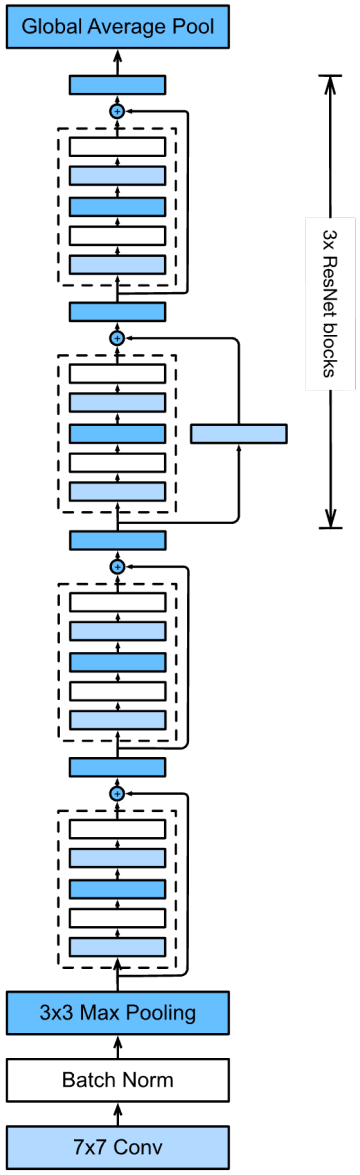
Credits: Deep Dive in Deep Learning , and G. Louppe

Deep CNNs

- To go deeper, architectures become much more complex
  - Multiple convolutions in parallel and recombined
  - Skip connections
- Recent ResNet-152 has 152 layers!



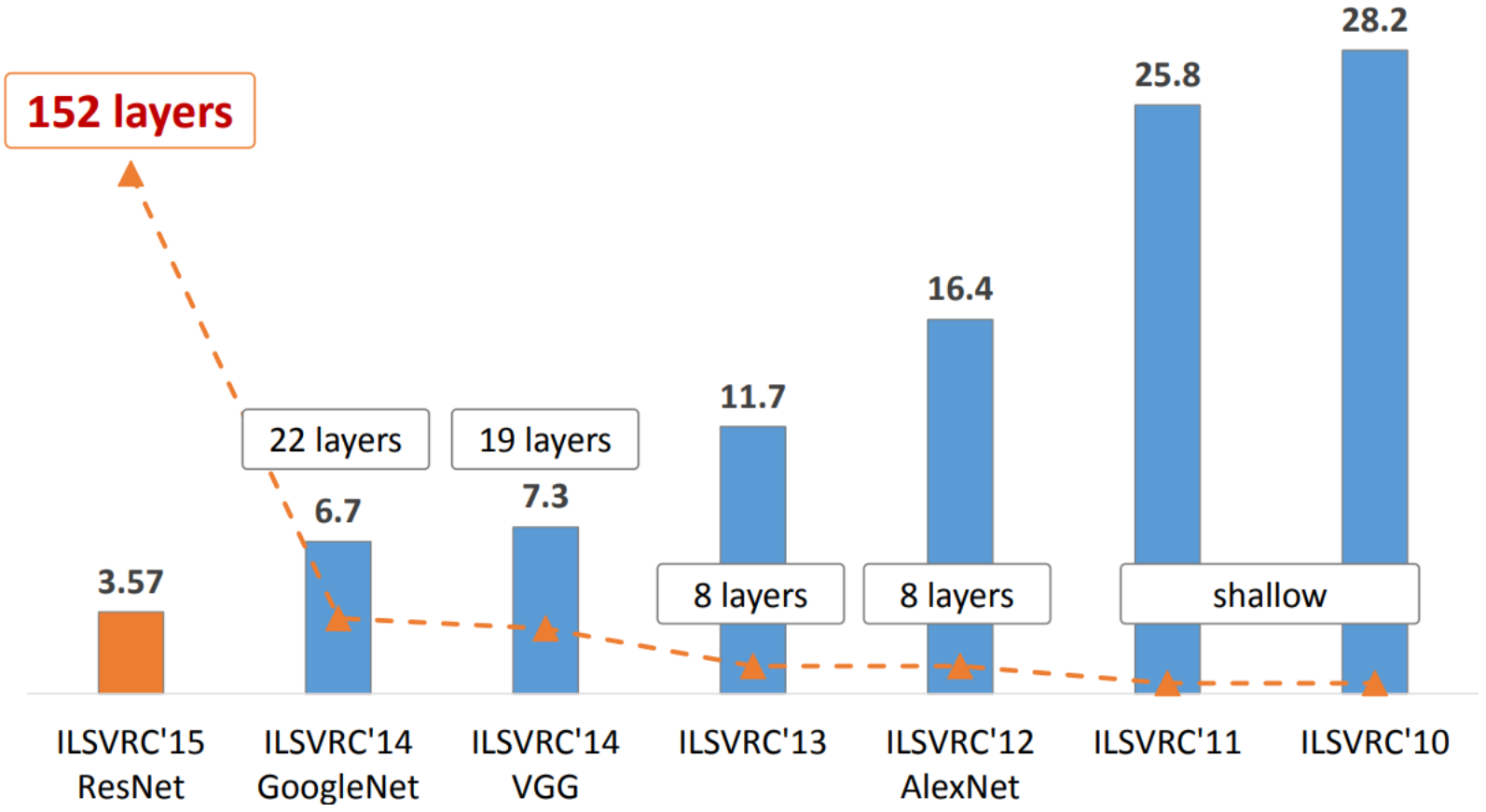
GoogLeNet
(Szegedy et al, 2014)



ResNet
(He et al, 2015)

Credits: Deep Dive in Deep Learning

Benefits of Depth



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 $x = \{x_1, x_2, \ldots, x_T\}$

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

Credit: F. Fleuret

Recurrent States

- Input sequence $x \in S(\mathbb{R}^m)$ of *variable* length $T(x)$
- Standard approach: use recurrent model that maintains a **recurrent state** $\boldsymbol{h}_t \in \mathbb{R}^q$ updated at each time step t. For $t = 1, \ldots, T(x)$:

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

  - 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

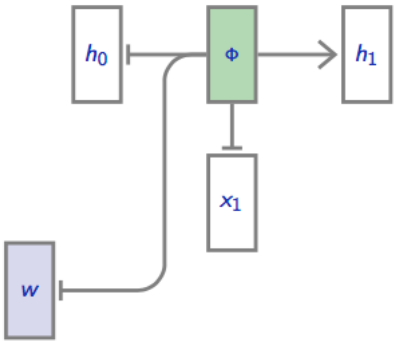
$$\boldsymbol{y}_t = \psi(\boldsymbol{h}_t; \theta)$$

Credit: F. Fleuret

Recurrent Neural Networks

Recurrent Model

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

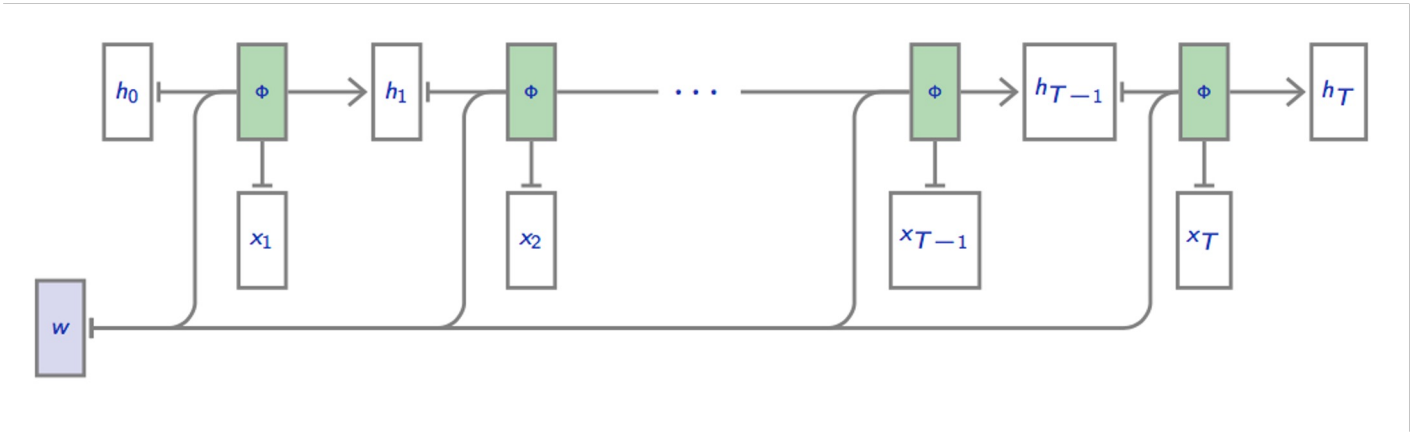


Credit: F. Fleuret

Recurrent Neural Networks

Recurrent Model

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

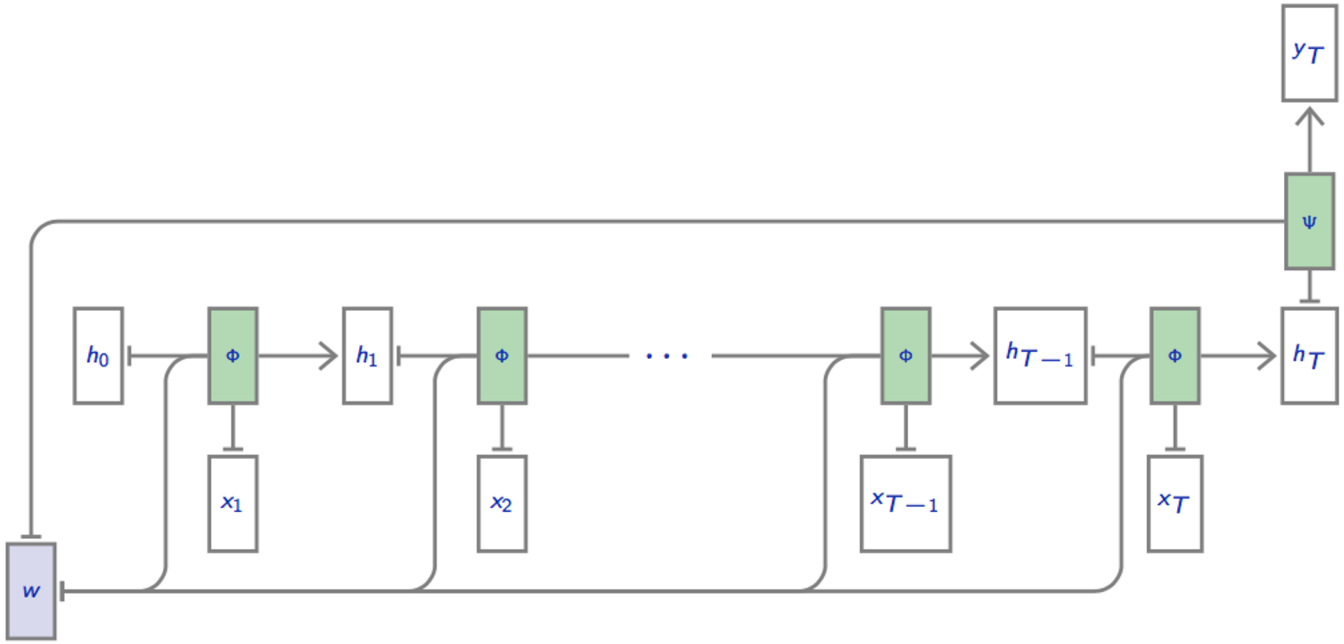


Credit: F. Fleuret

Recurrent Neural Networks

Prediction

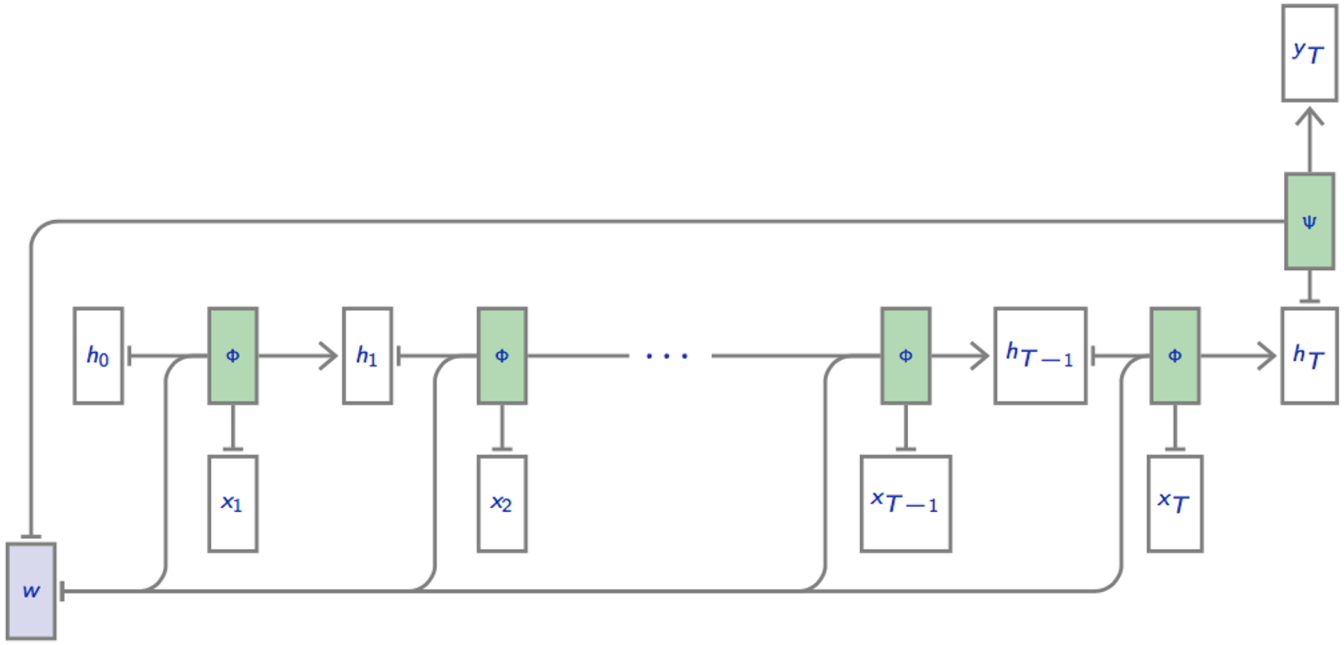
$$\boldsymbol{y}_t = \psi(\boldsymbol{h}_t; \theta)$$



Credit: F. Fleuret

Recurrent Neural Networks

[0.98] → Positive Sentiment

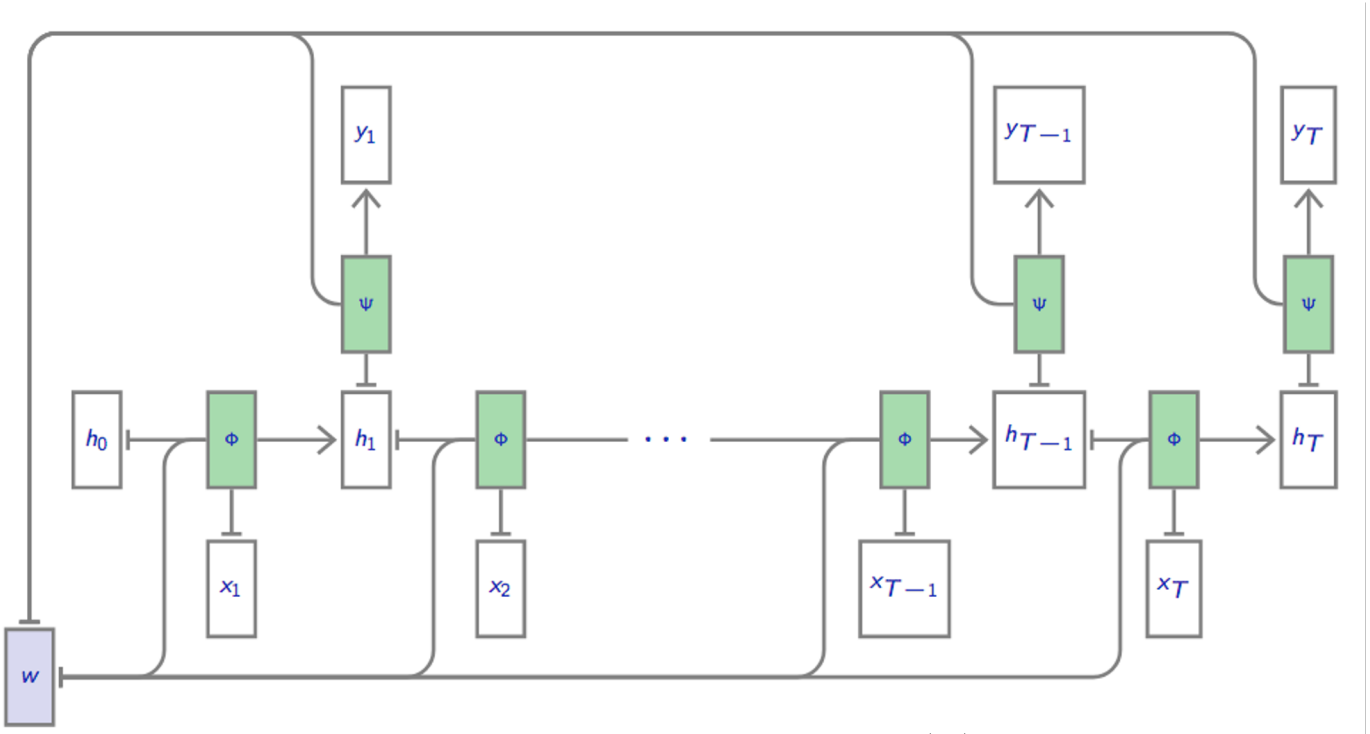


Sentiment Analysis

Credit: F. Fleuret

Recurrent Neural Networks

Prediction per sequence element

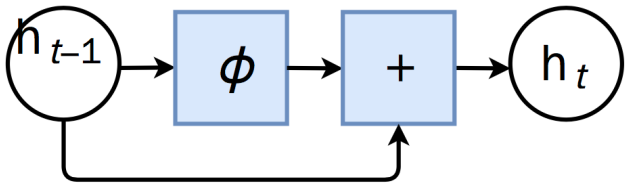


Although the number of steps $T(x)$ depends on x, this is a standard computational graph and automatic differentiation can deal with it as usual. This is known as "backpropagation through time" (Werbos, 1988)

Credit: F. Fleuret

Gating

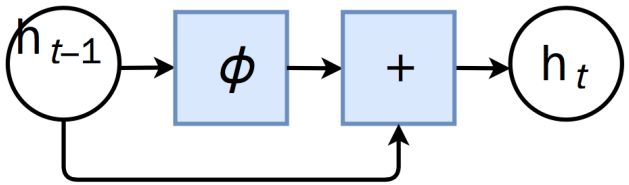
- Gating:
  - network can grow very deep, in time → vanishing gradients.
  - *Critical component*: add pass-through (additive paths) so recurrent state does not go repeatedly through squashing non-linearity.



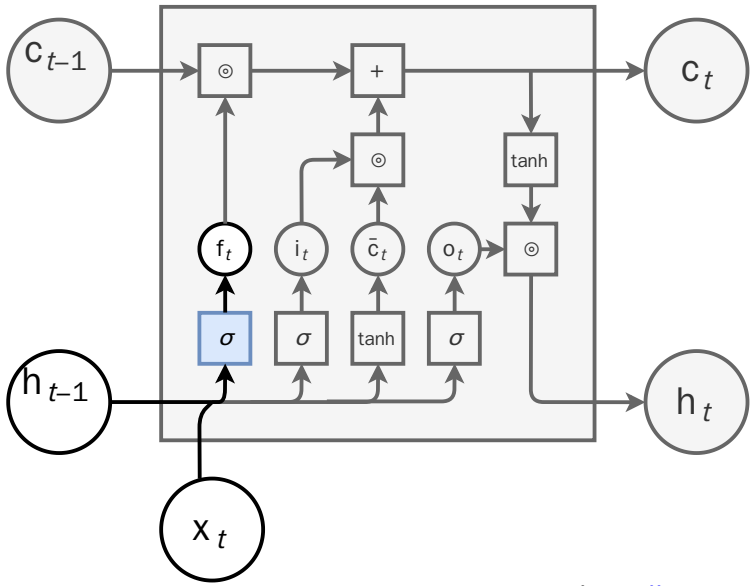
Credit: Gilles Louppe

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.

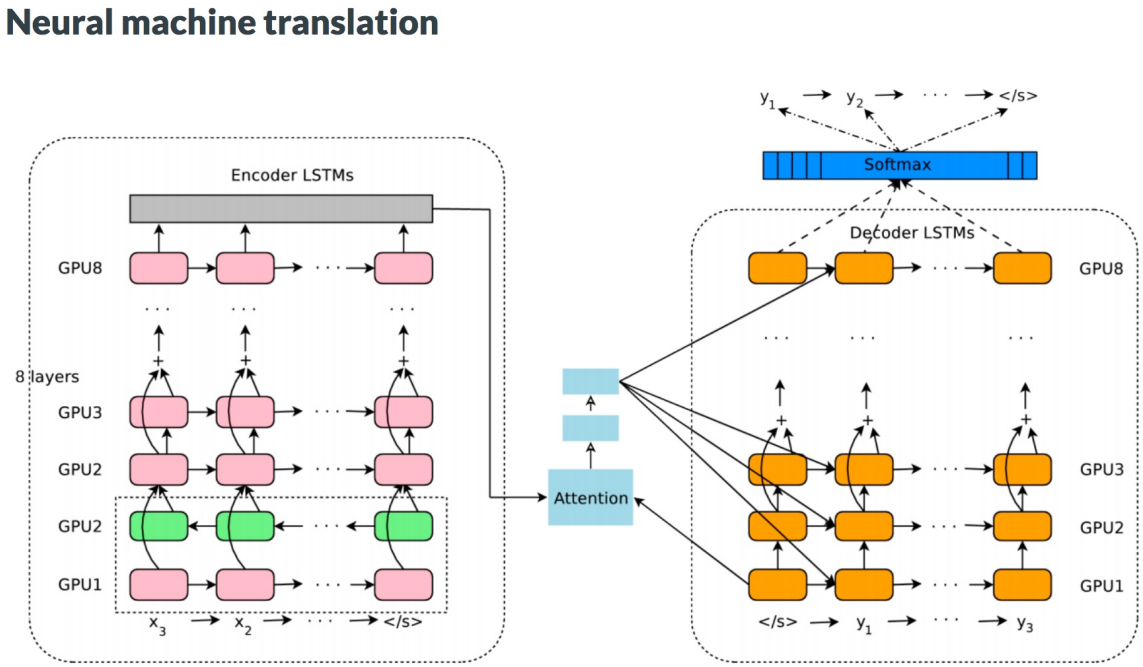


- LSTM:
  - Add internal state separate from output state
  - Add input, output, and forget gating



Credit: Gilles Louppe

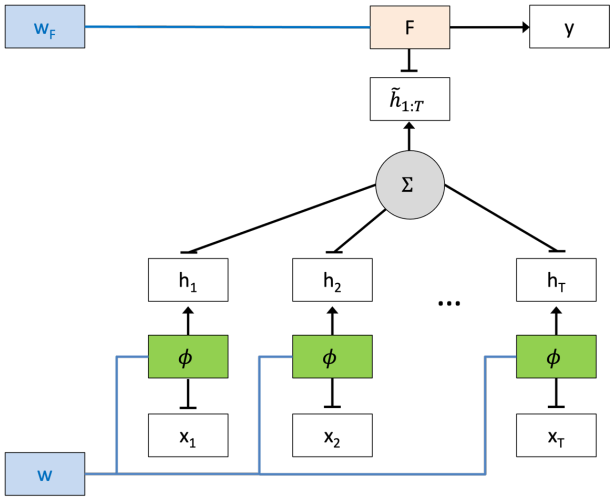
Examples



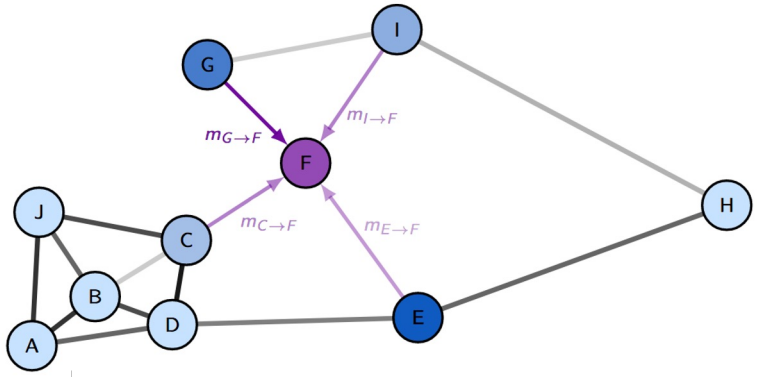
Y. Wu et al, 2016

Many Other Architecture Choices

Deep Sets



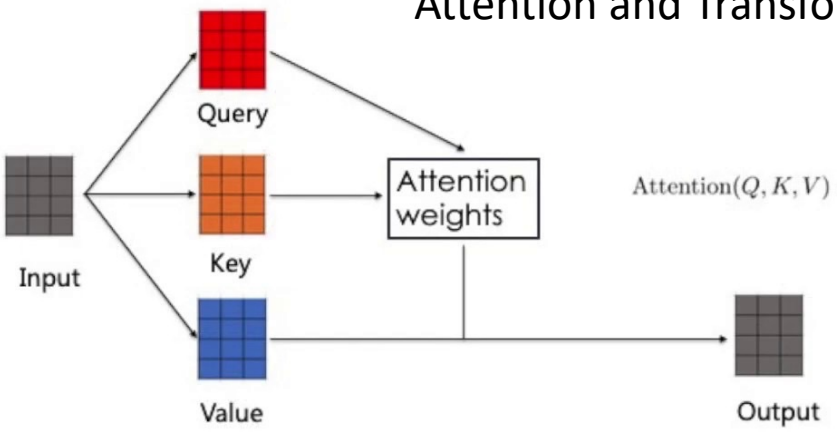
Graph Neural Networks



$$\tilde{m}_j^t = f(h_j^{t-1})$$
$$m_{j\to i}^t = \sigma(A_{ij}\tilde{m}_j^t)$$
$$h_i^t = \text{GRU}(h_i^{t-1}, \Sigma_j m_{j\to i}^t)$$

Image Credit: I. Henrion

Attention and Transformers



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

+ More…

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 explicitly model a high-dim. signal
  - Data synthesis / simulation
  - Density estimation, Anomaly detection
  - Denoising, Super Resolution, Data compression
  - …

- Often don't have labels → *Unsupervised Learning*

- **Generative models** aim to:
  - Learn a density $p(x)$ that explains the data
  - Draw samples of plausible data points

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

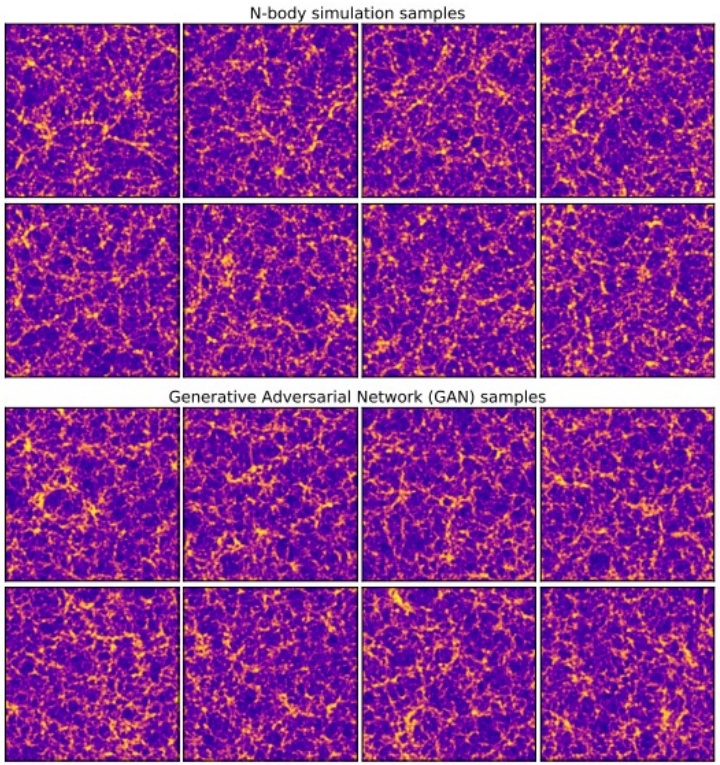
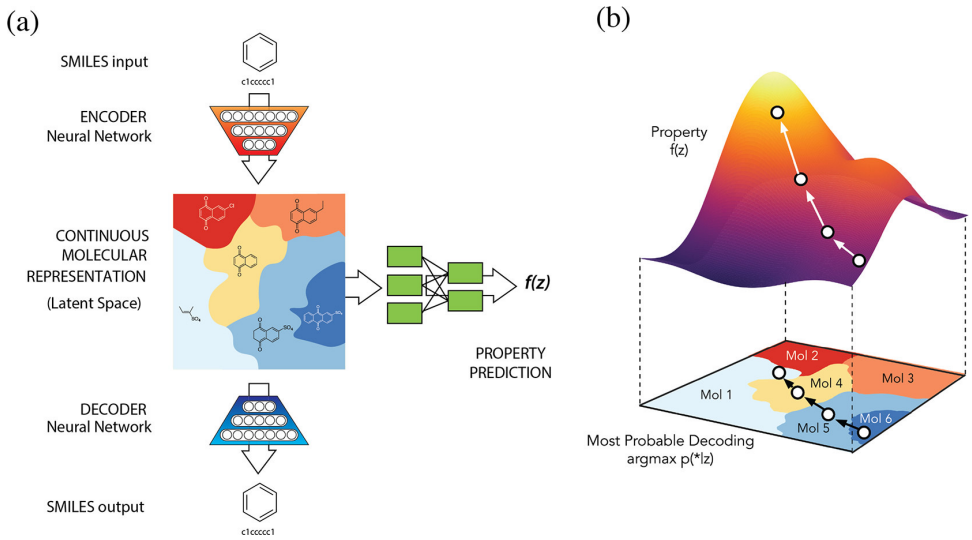
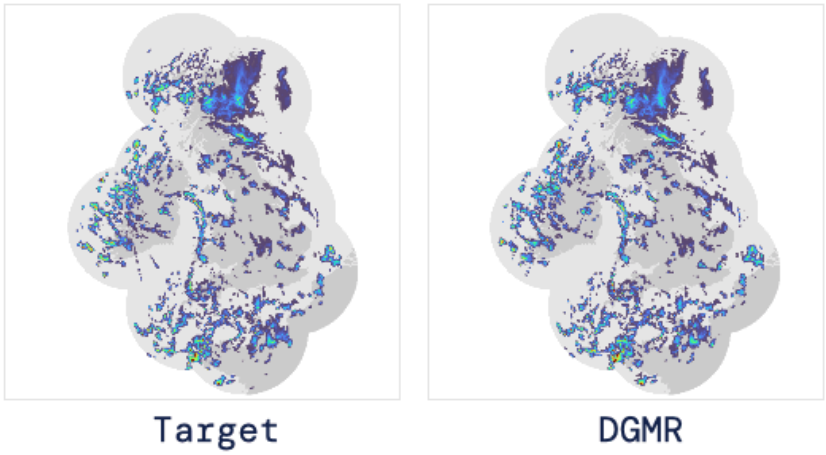


Figure 1: Samples from N-body simulation and from GAN for the box size of 500 Mpc. Note that the transformation in Equation 3.1 with $a = 20$ was applied to the images shown above for better clarity.

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)

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 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

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

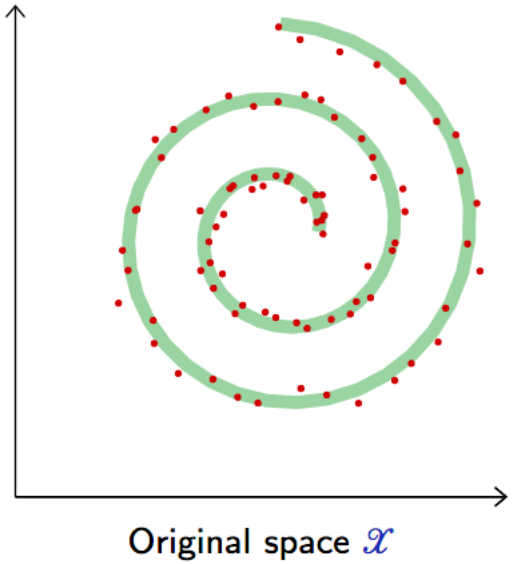
x is a random variable sampled from the distribution $q(x;\theta)$

Goal: Learn θ

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

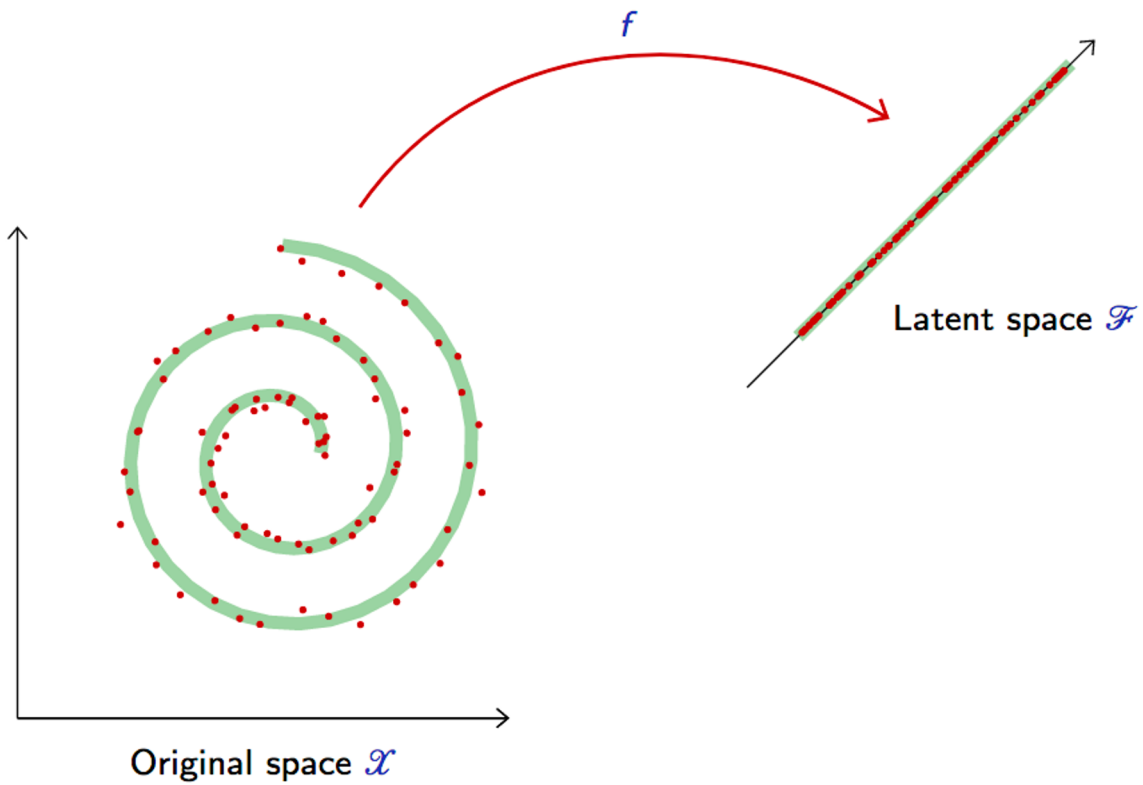
i.e. want to recreate the raw data distribution (such as the distribution of natural images).

Modeling Data and Meaningful Degrees of Freedom



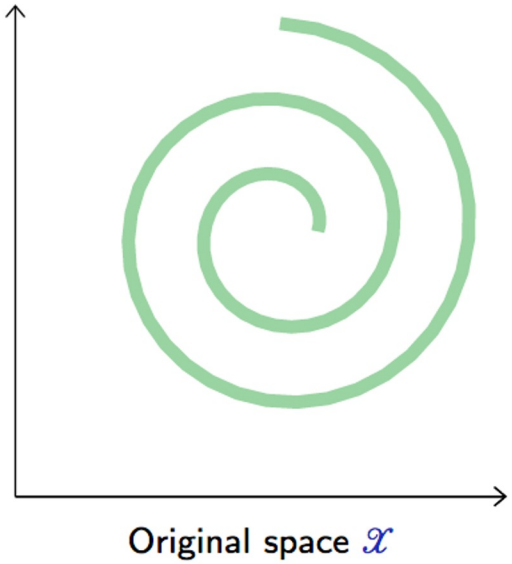
Fleuret, Deep Learning Course

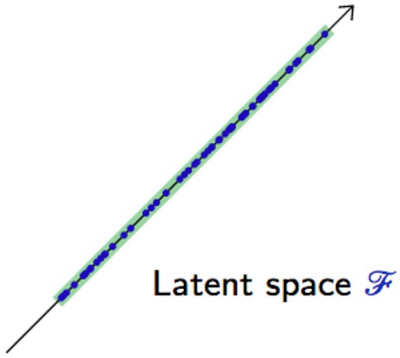
Modeling Data and Meaningful Degrees of Freedom



Fleuret, Deep Learning Course

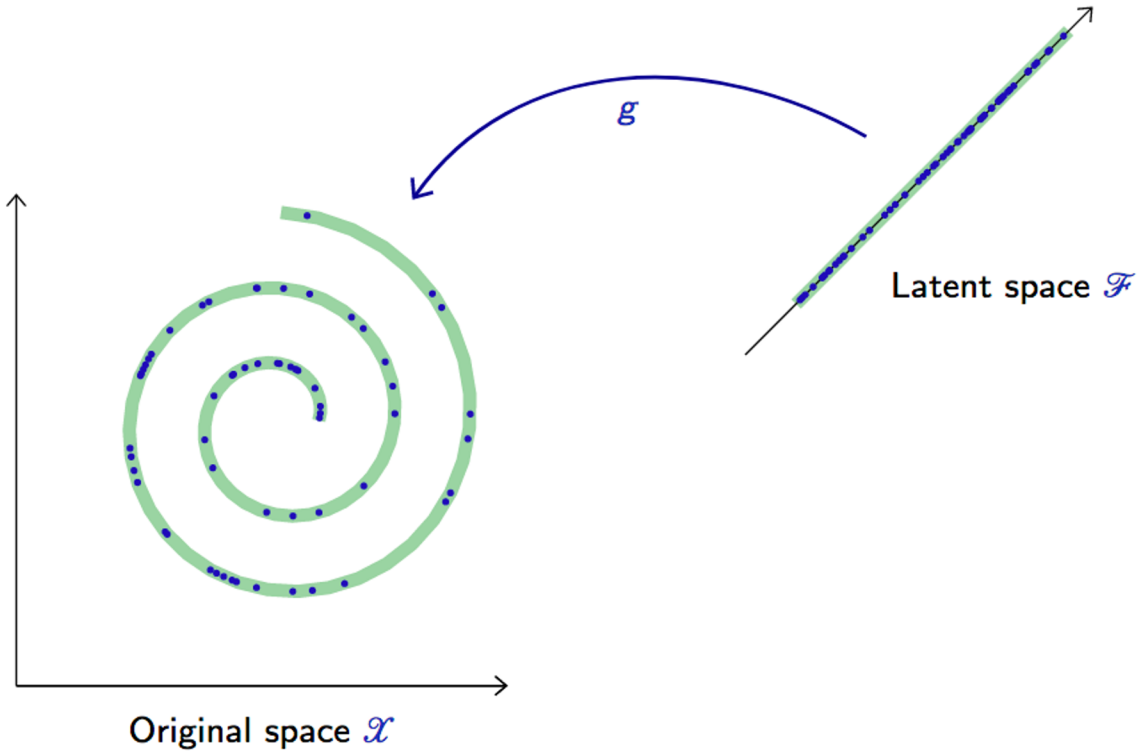
Modeling Data and Meaningful Degrees of Freedom





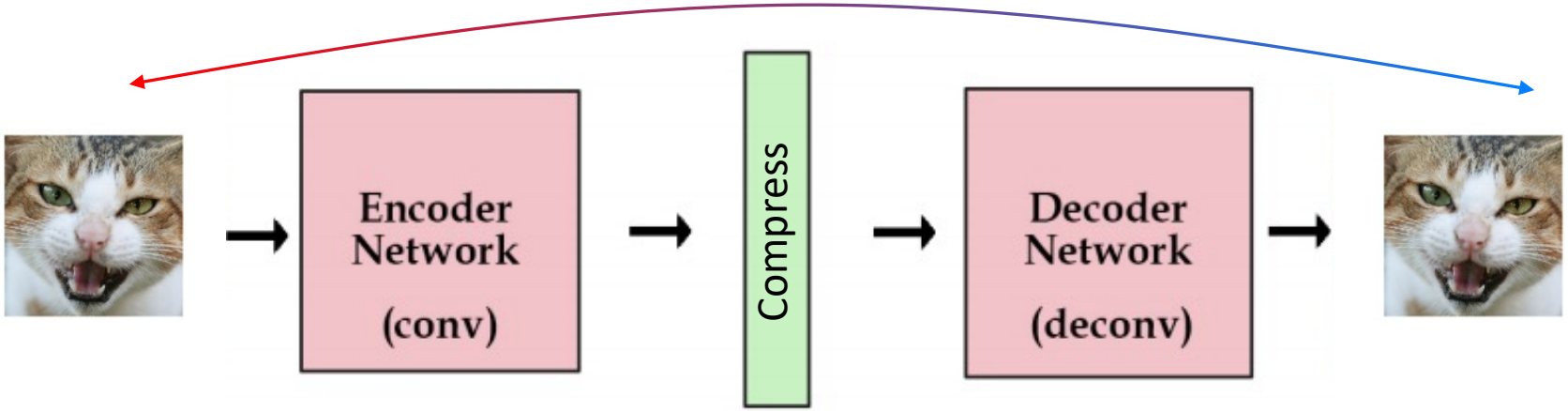
Fleuret, Deep Learning Course

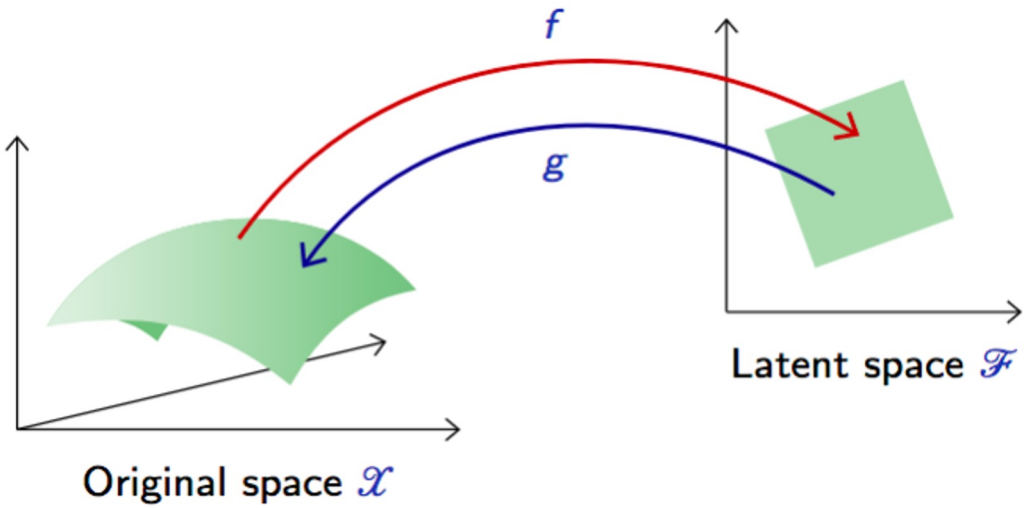
Modeling Data and Meaningful Degrees of Freedom



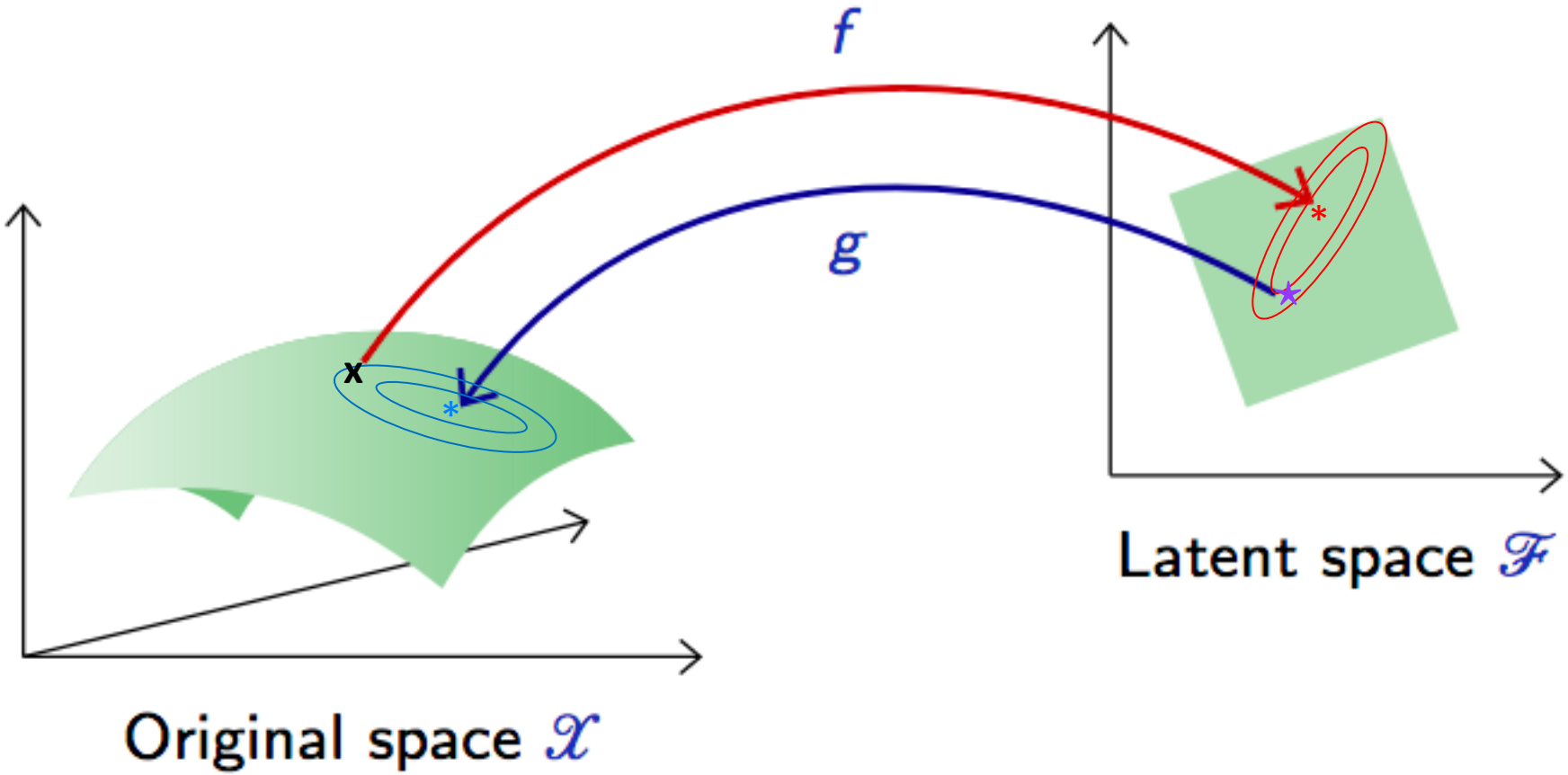
Fleuret, Deep Learning Course

AutoEncoders





Modeling Data and Meaningful Degrees of Freedom



Common Approach to Deep Generative Modeling:

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

Fleuret, Deep Learning Course

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

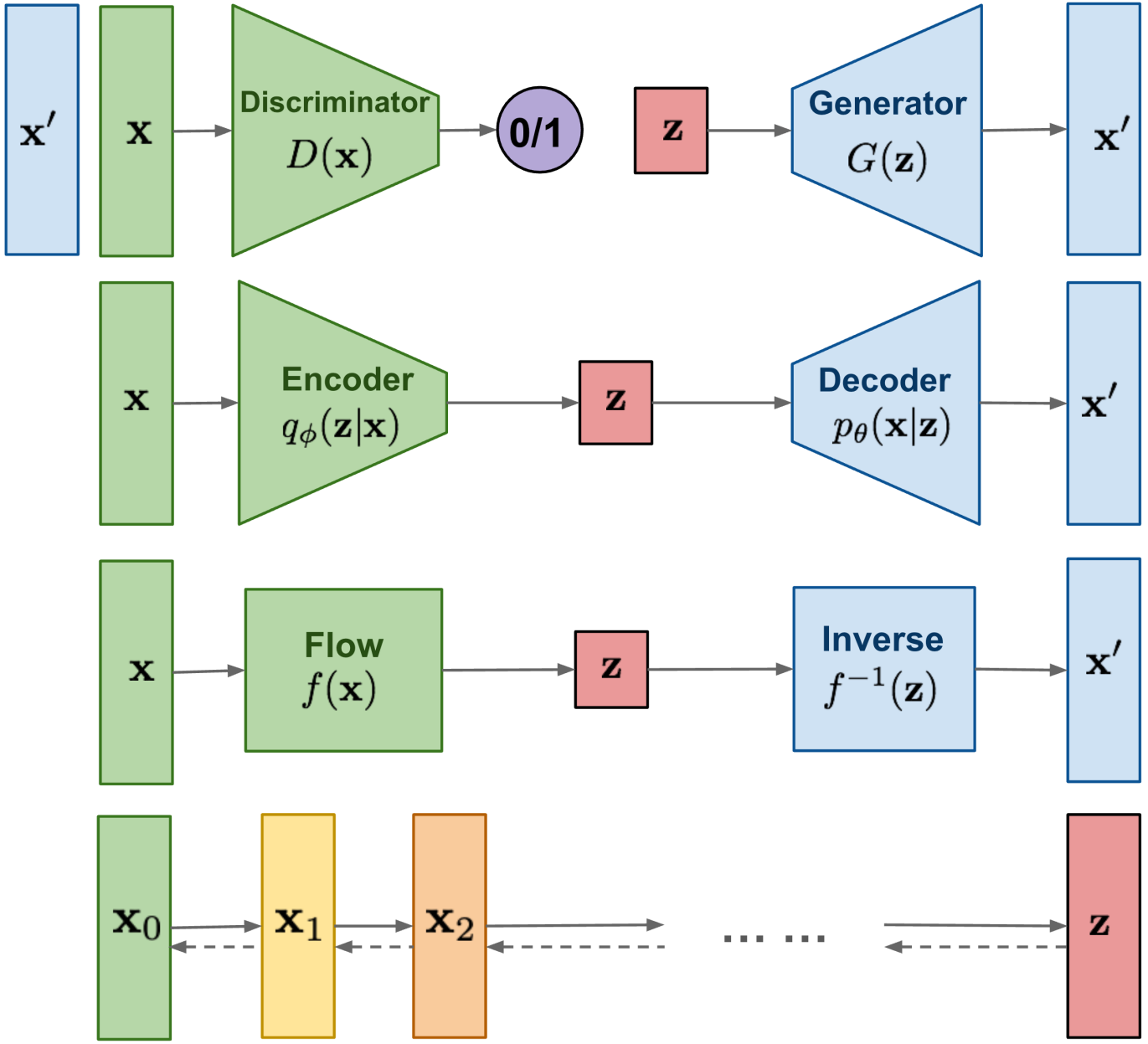
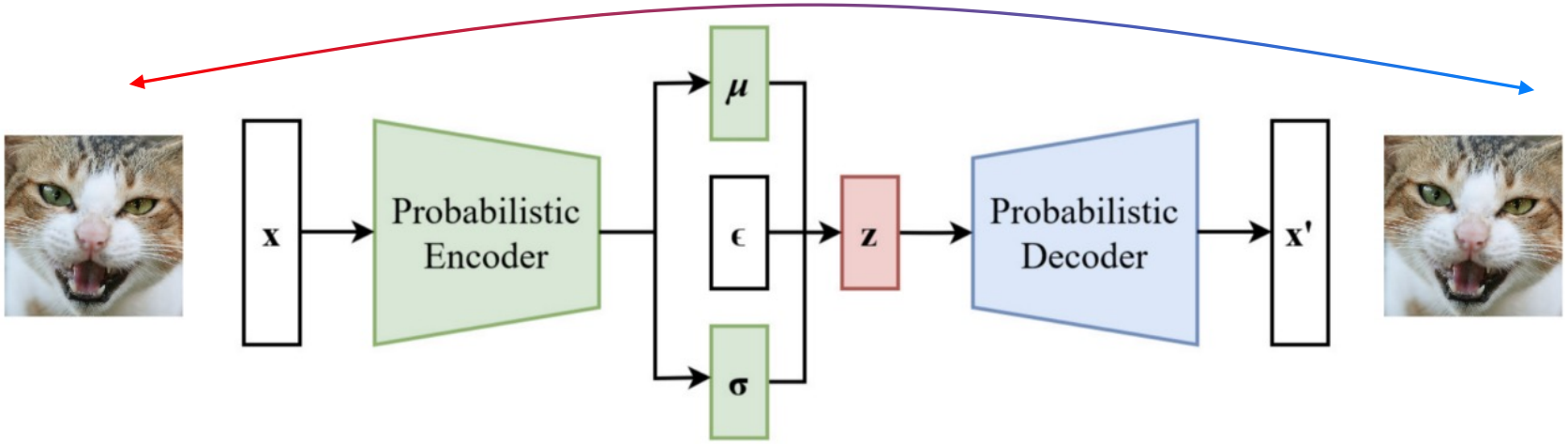
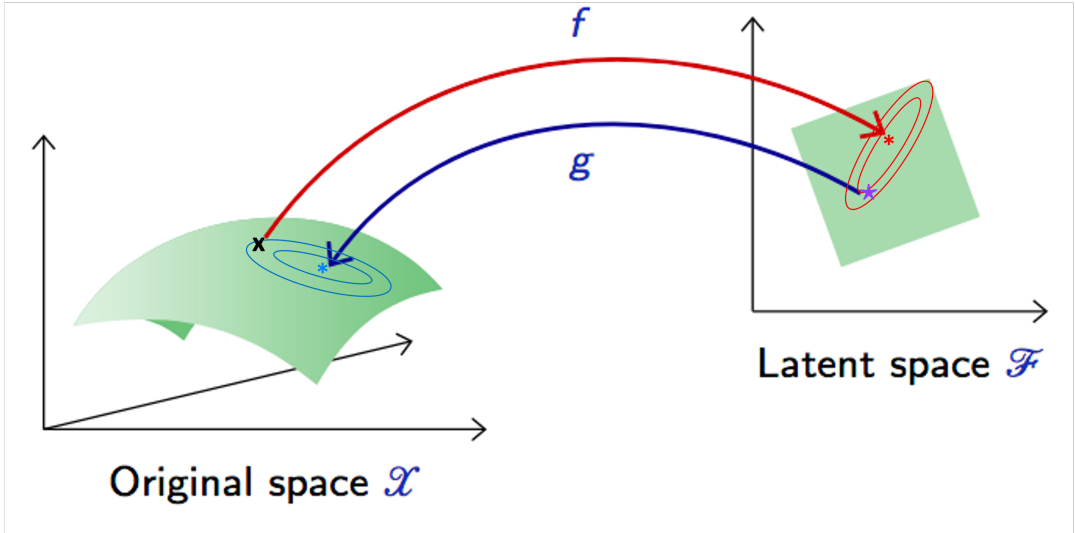


Image from Lilian Weng

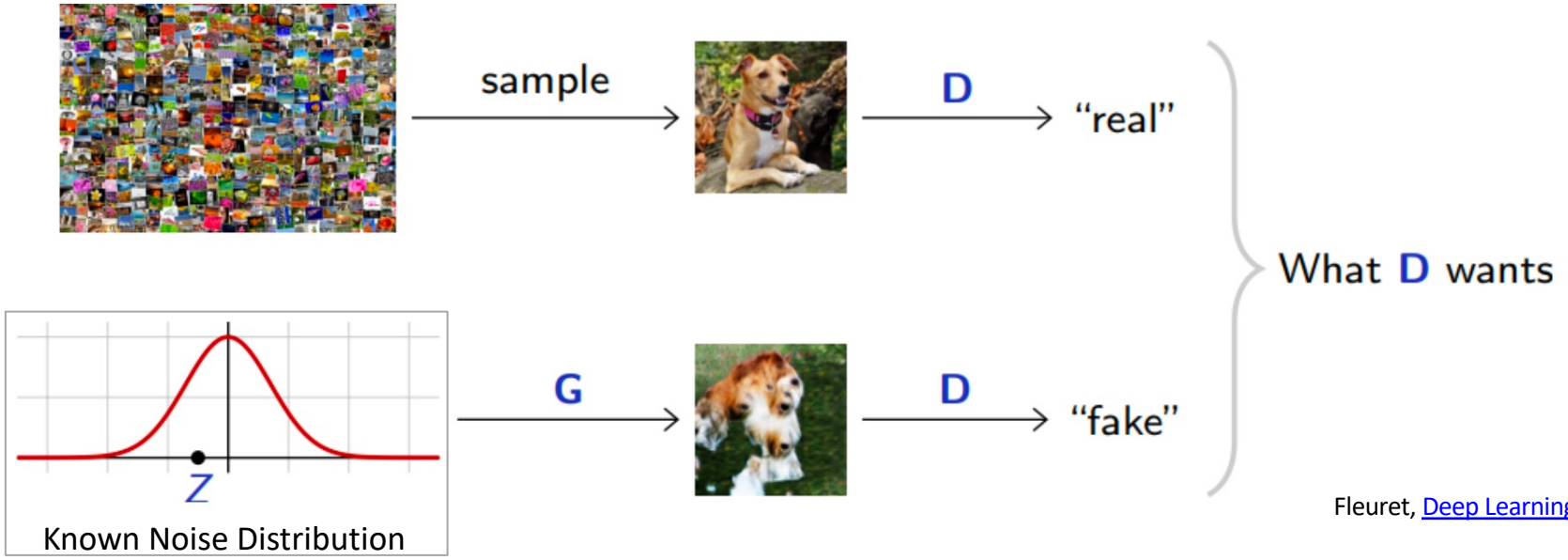
Variational AutoEncoders





Generative Adversarial Networks (GAN)

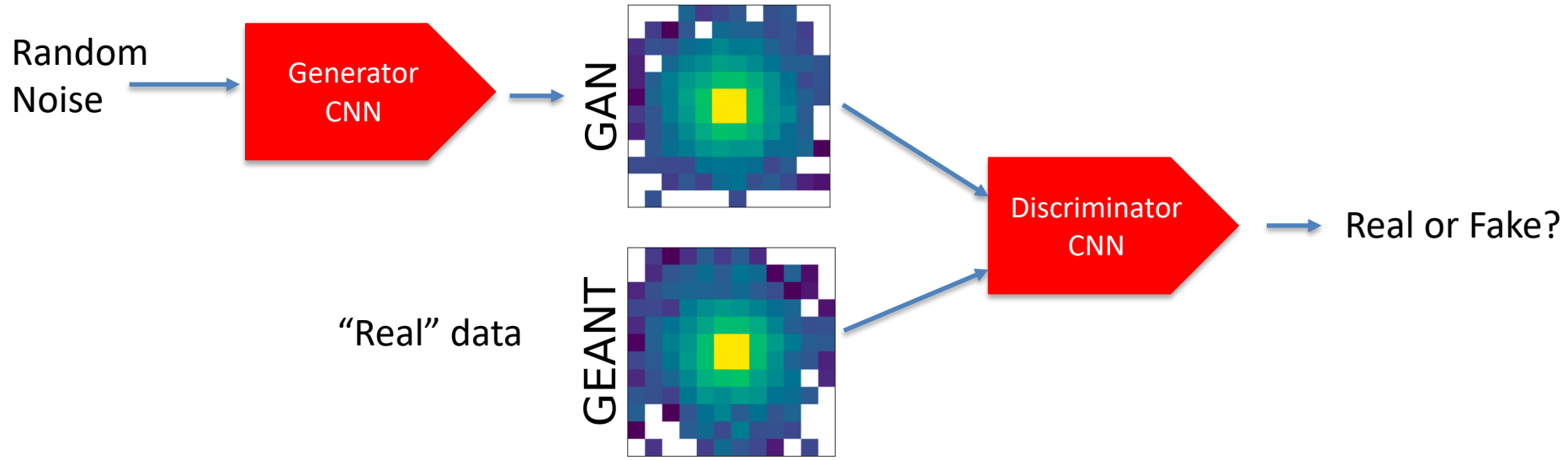
Goodfellow et. al., 2014

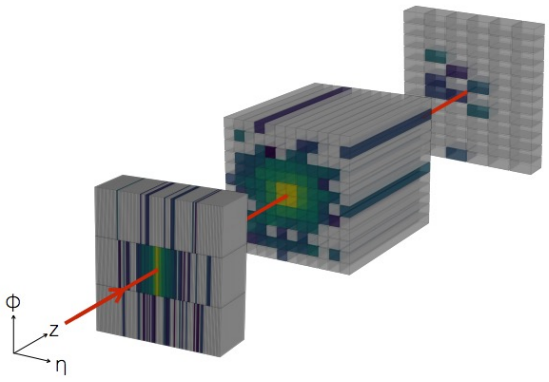


Fleuret, Deep Learning Course

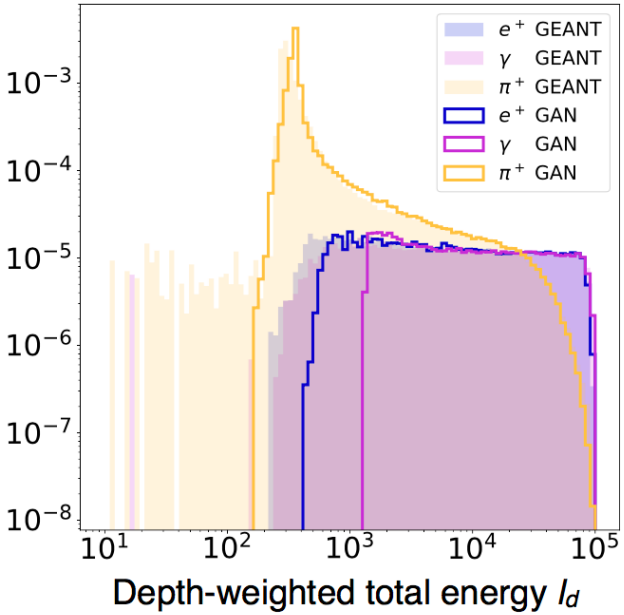
- Generator goal:
  - Produce *fake* data to trick discriminator to classify as *real*
- Discriminator goal:
  - Classify data as real or fake, as well as possible
- *Adversarial* setup
  - Two-player game with two networks w/ opposing objectives

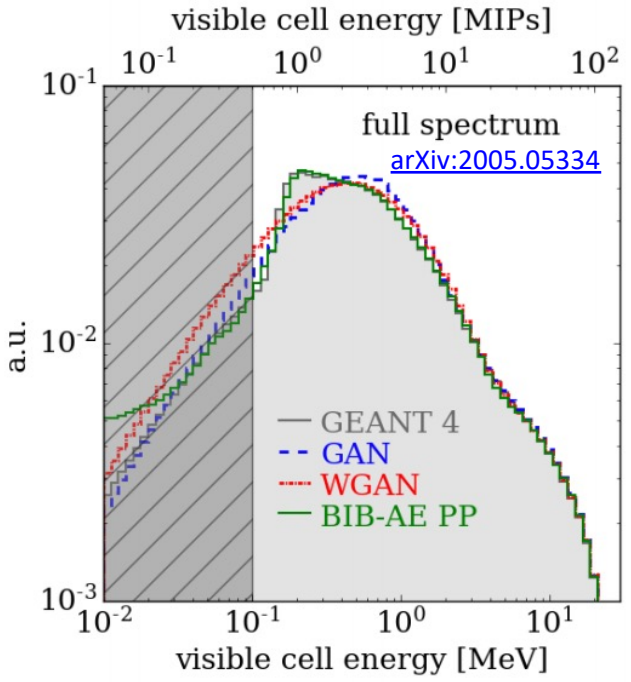
GANs for Calorimeter Energy Depositions





PRD97, 014021 (2018)
arXiv:1705.02355
arXiv:1701.05927

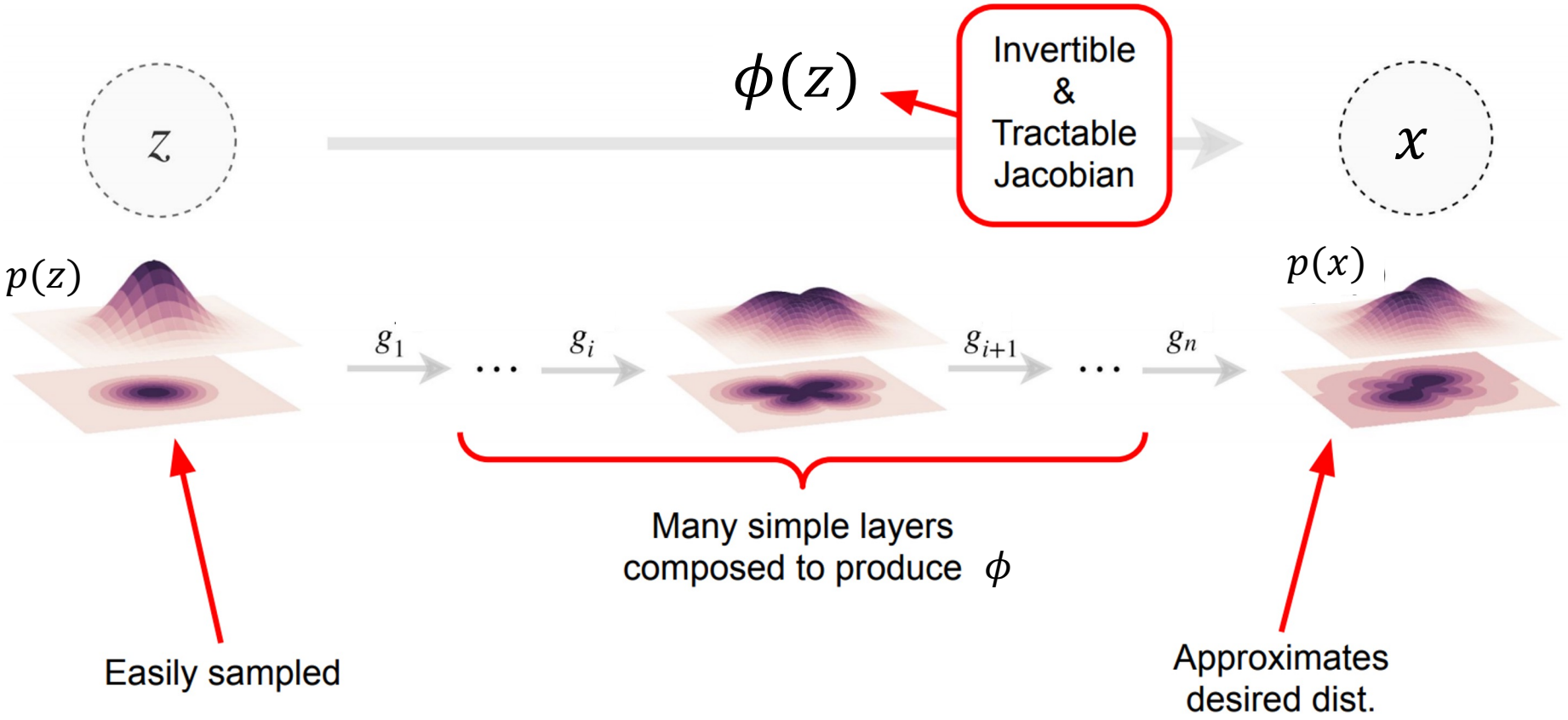




Normalizing Flows

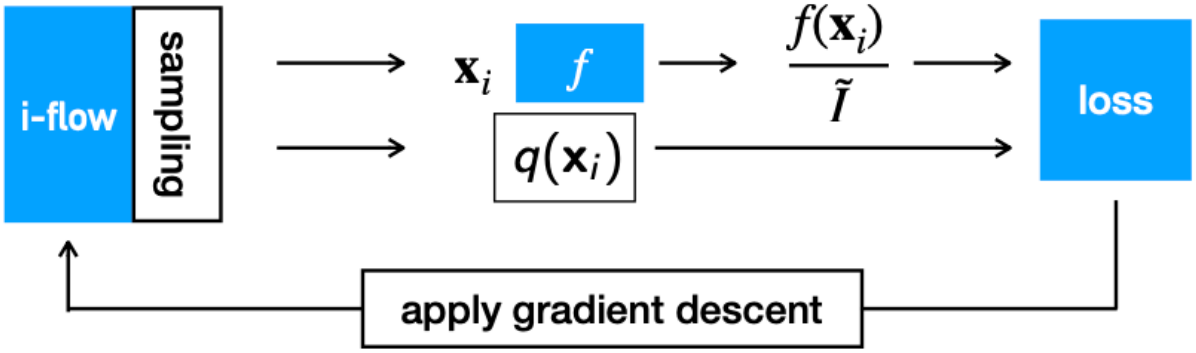
Normalizing flows allows *explicit density estimation*
We can evaluate density $p(x)$

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



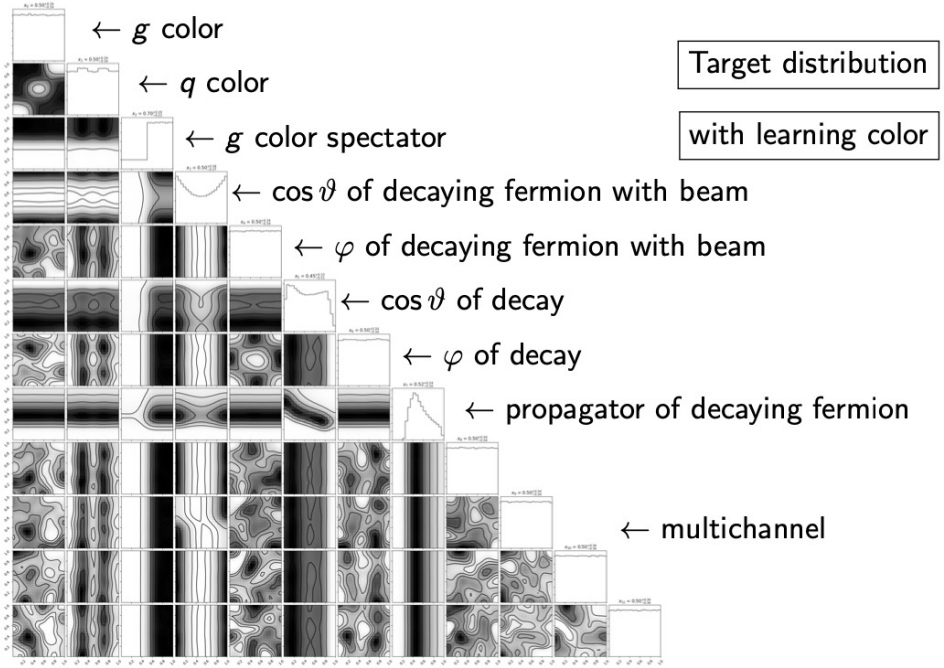
Slide credit: G. Kanwar

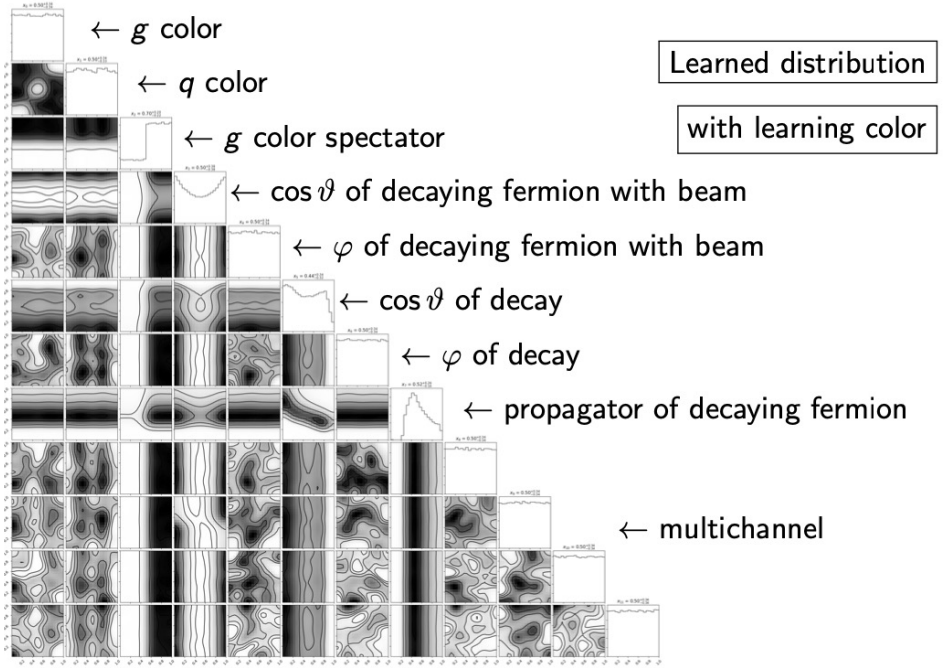
Event Generation with Normalizing Flows



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

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





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

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

An increasingly large number of people are **defining the networks procedurally in a data-dependent way** (with loops and conditionals), allowing them **to change dynamically as a function of the input data** fed to them. It's really very much **like a regular program, except it's parameterized**

- Yann LeCun, 2018

CNN

1D Convolutional Layers

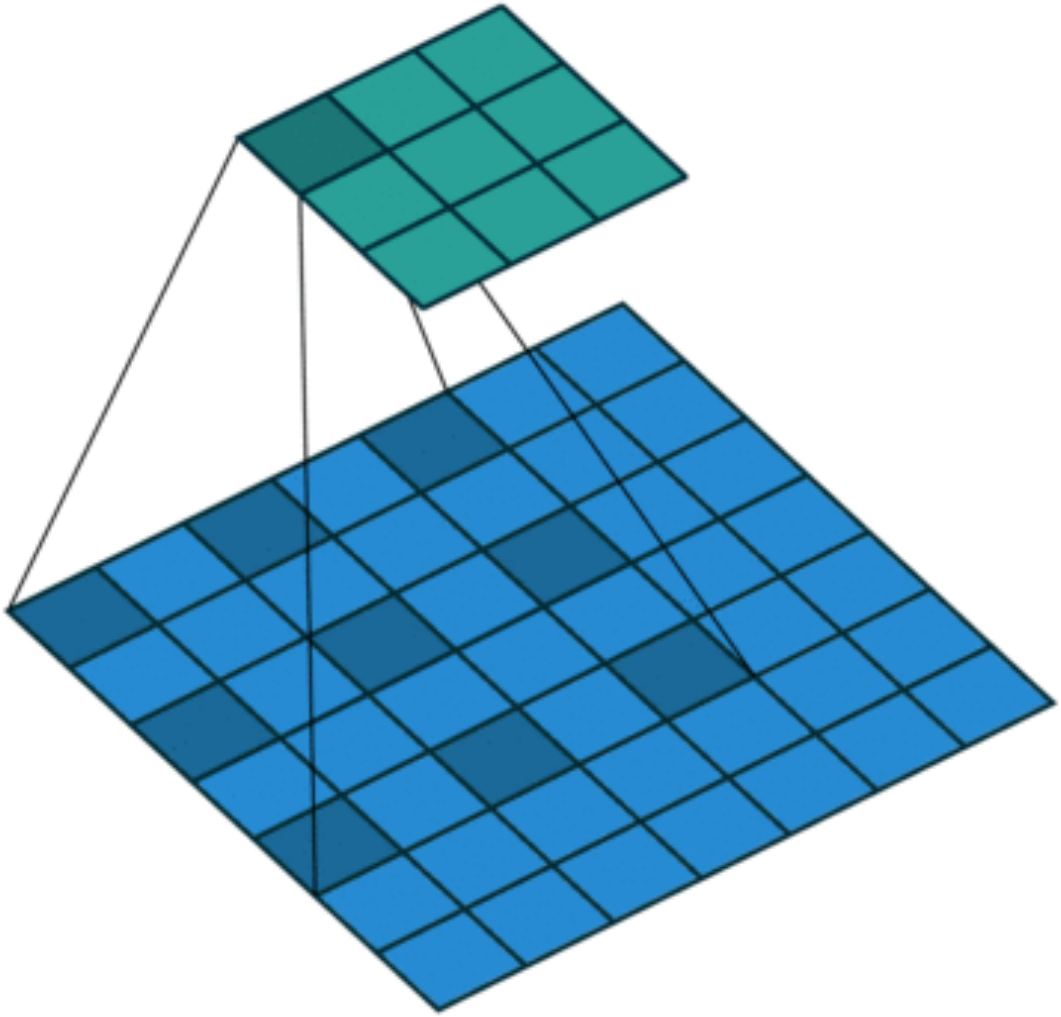
- **Data:** $x \in \mathbb{R}^M$
- **Convolutional kernel of width k:** $u \in \mathbb{R}^k$

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

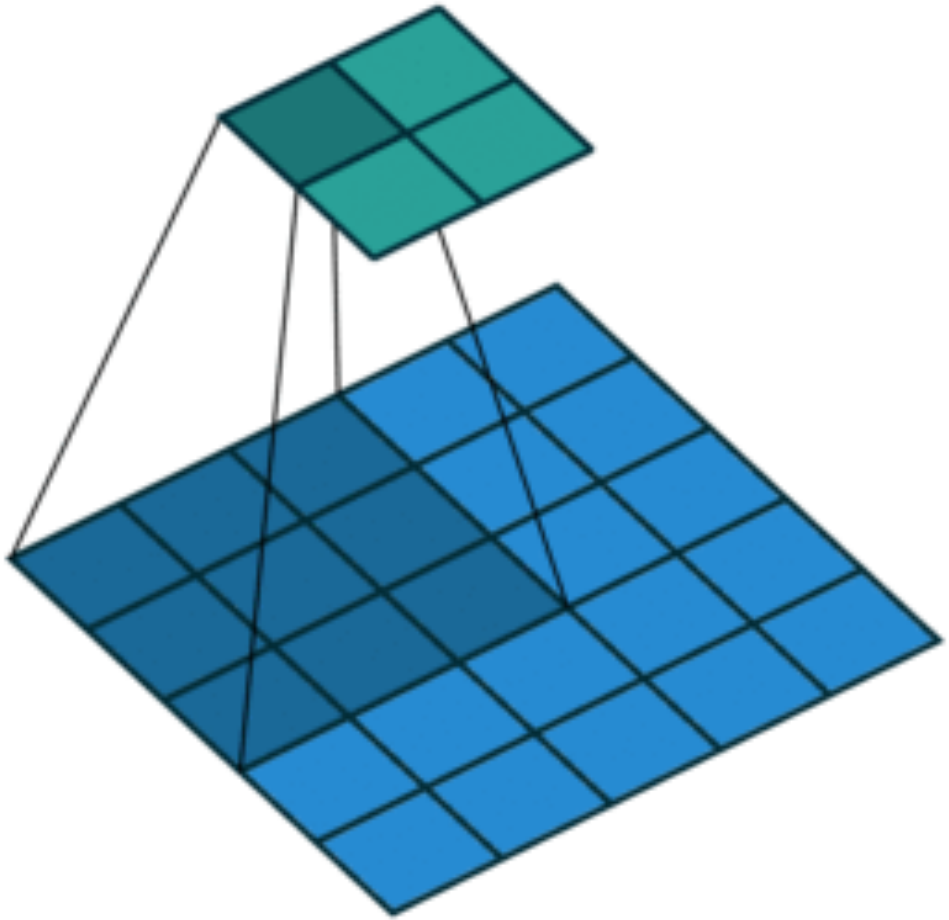
$$(x \circledast \mathrm{u})_i = \sum_{b=0}^{k-1} x_{i+b} u_b$$

- Scan across data and multiply by kernel elements

Dilation

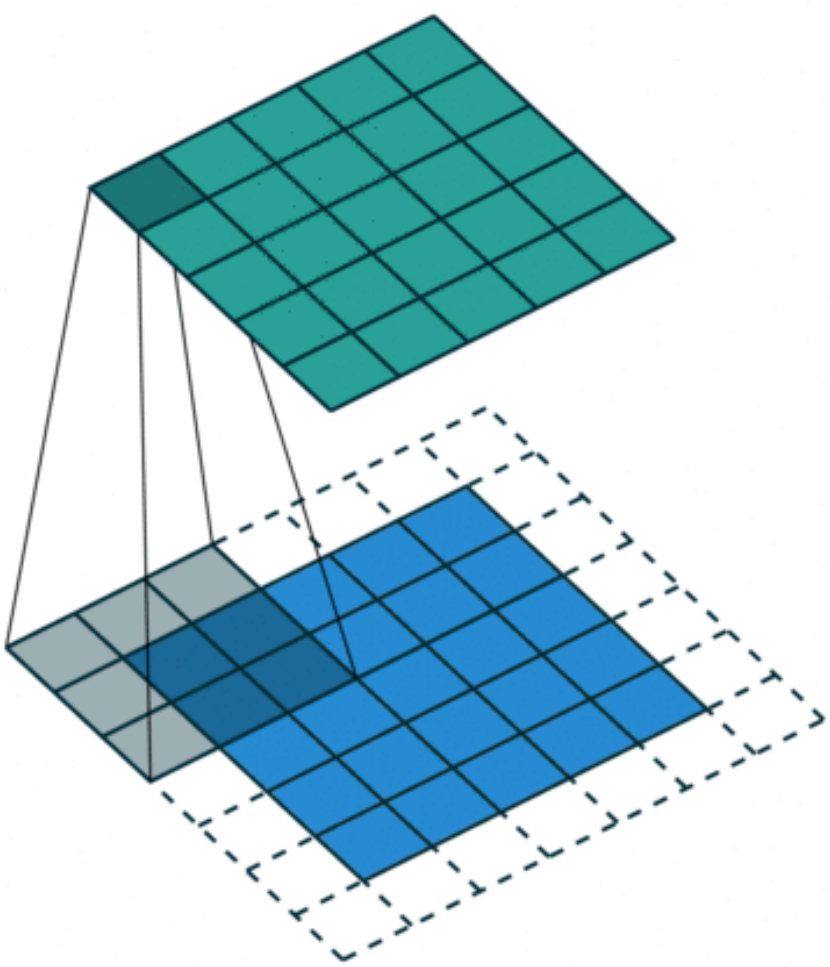


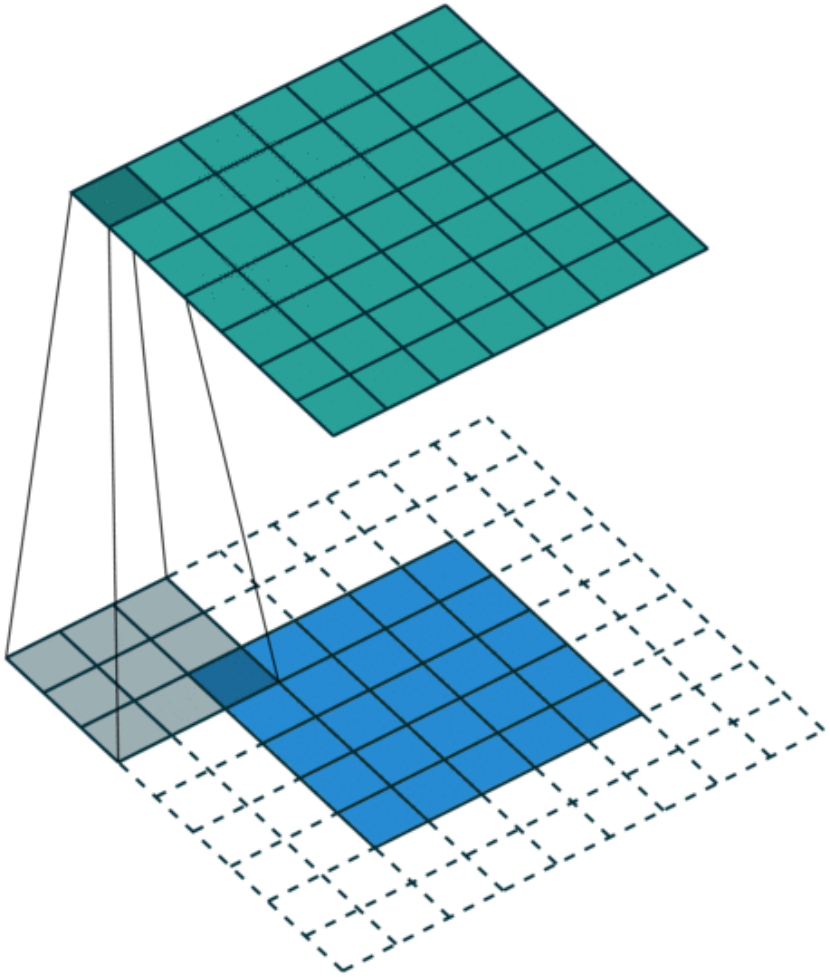
Stride – Step Size When Moving Kernel Across Input



Fleuret, Deep Learning Course

Padding – Size of Zero Frame Around Input





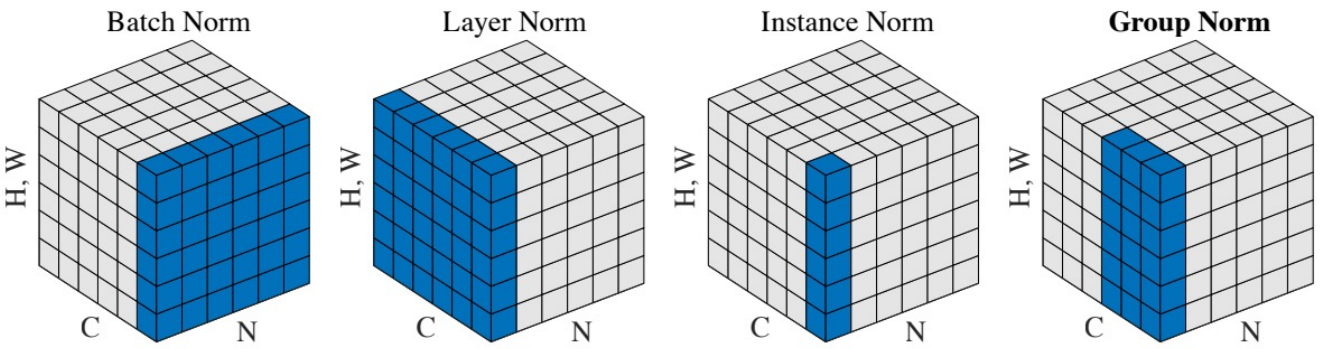
Fleuret, Deep Learning Course

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 $\forall b = 1 \ldots B$

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

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

  - γ and β are parameters to optimize

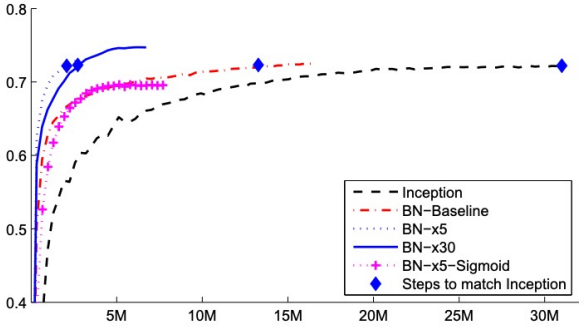
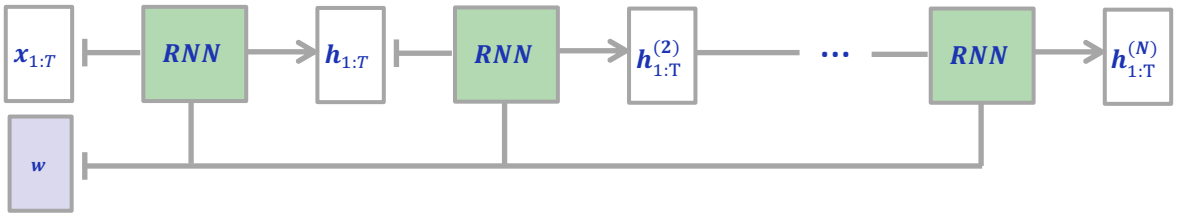


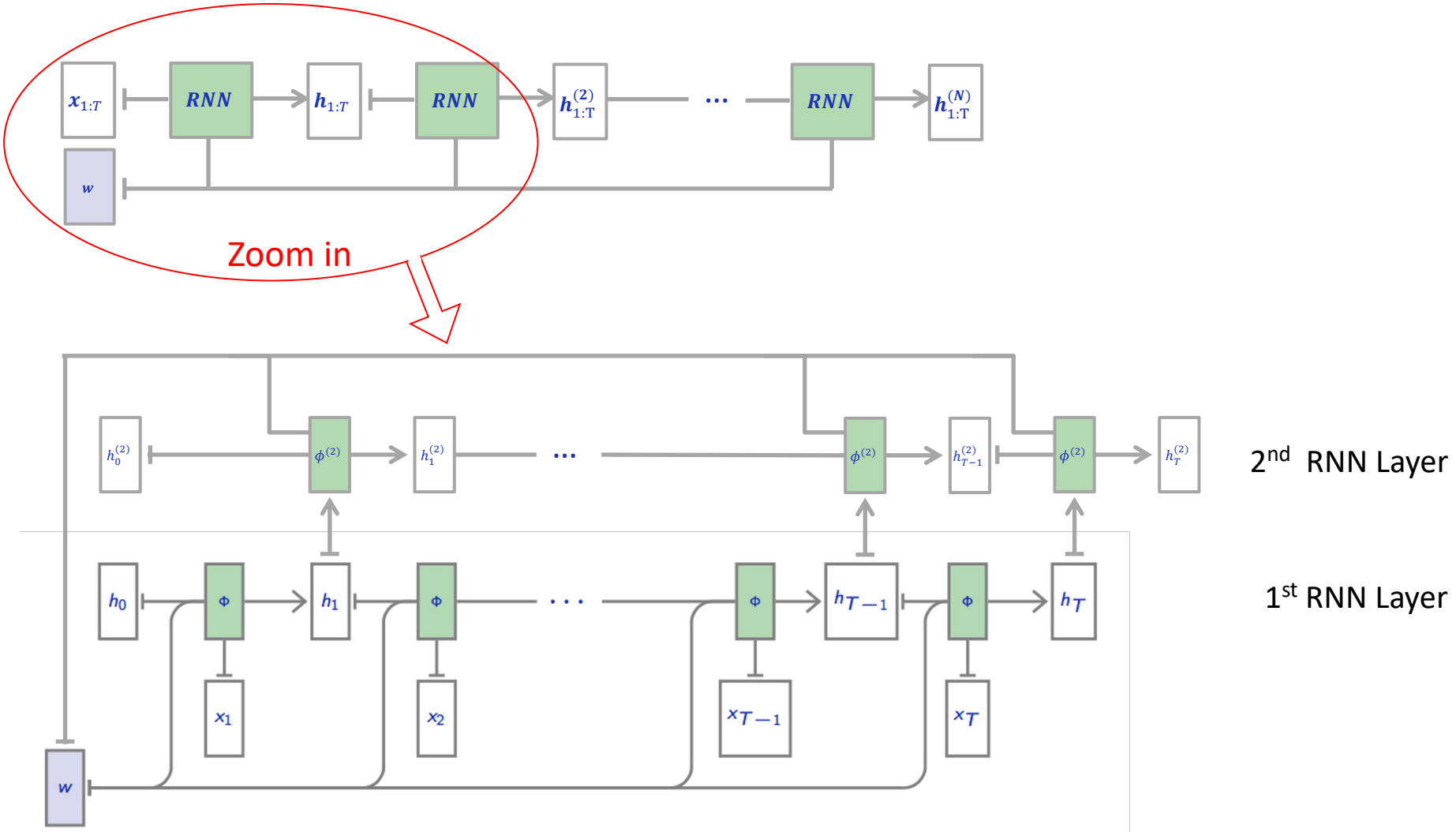
Figure 2: *Single crop validation accuracy of Inception and its batch-normalized variants, vs. the number of training steps.*

RNN

Stacked RNN



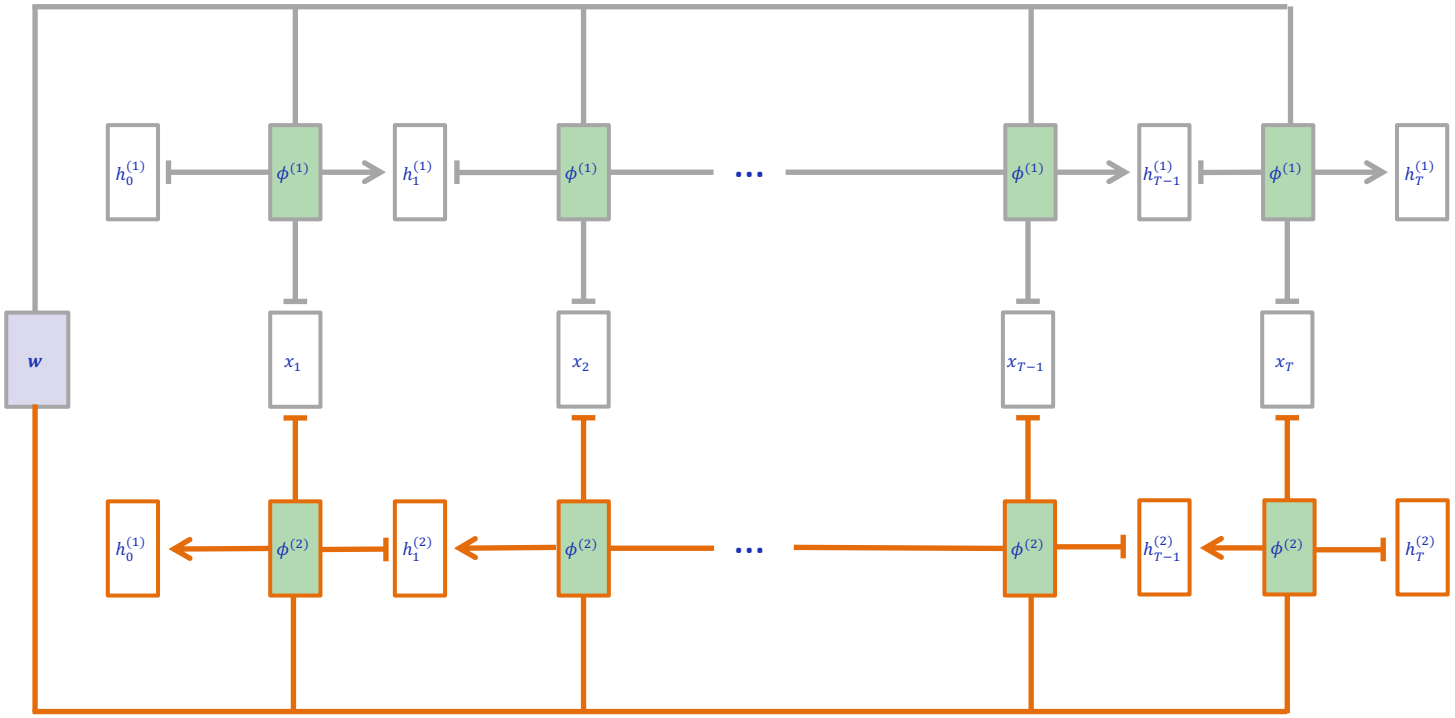
Stacked RNN



Two Stacked LSTM Layers

Bi-Directional RNN

Forward in time RNN Layer

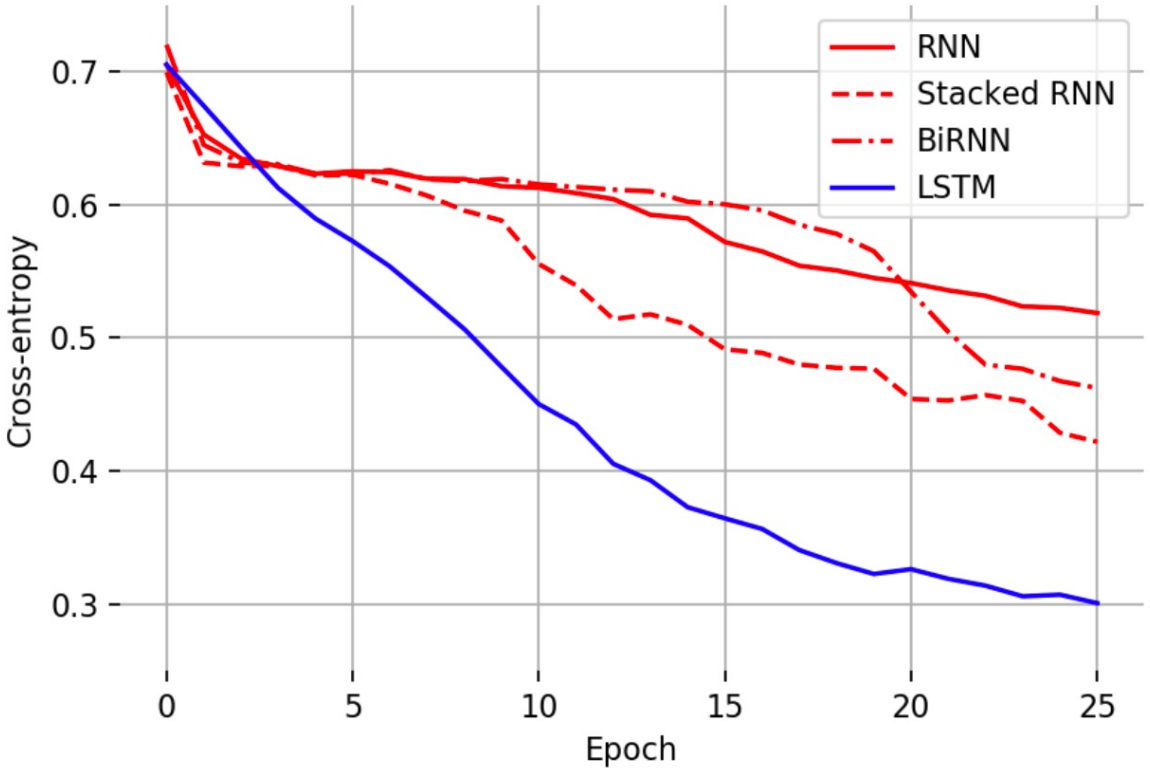


Backward in time RNN Layer

Comparison on Toy Problem

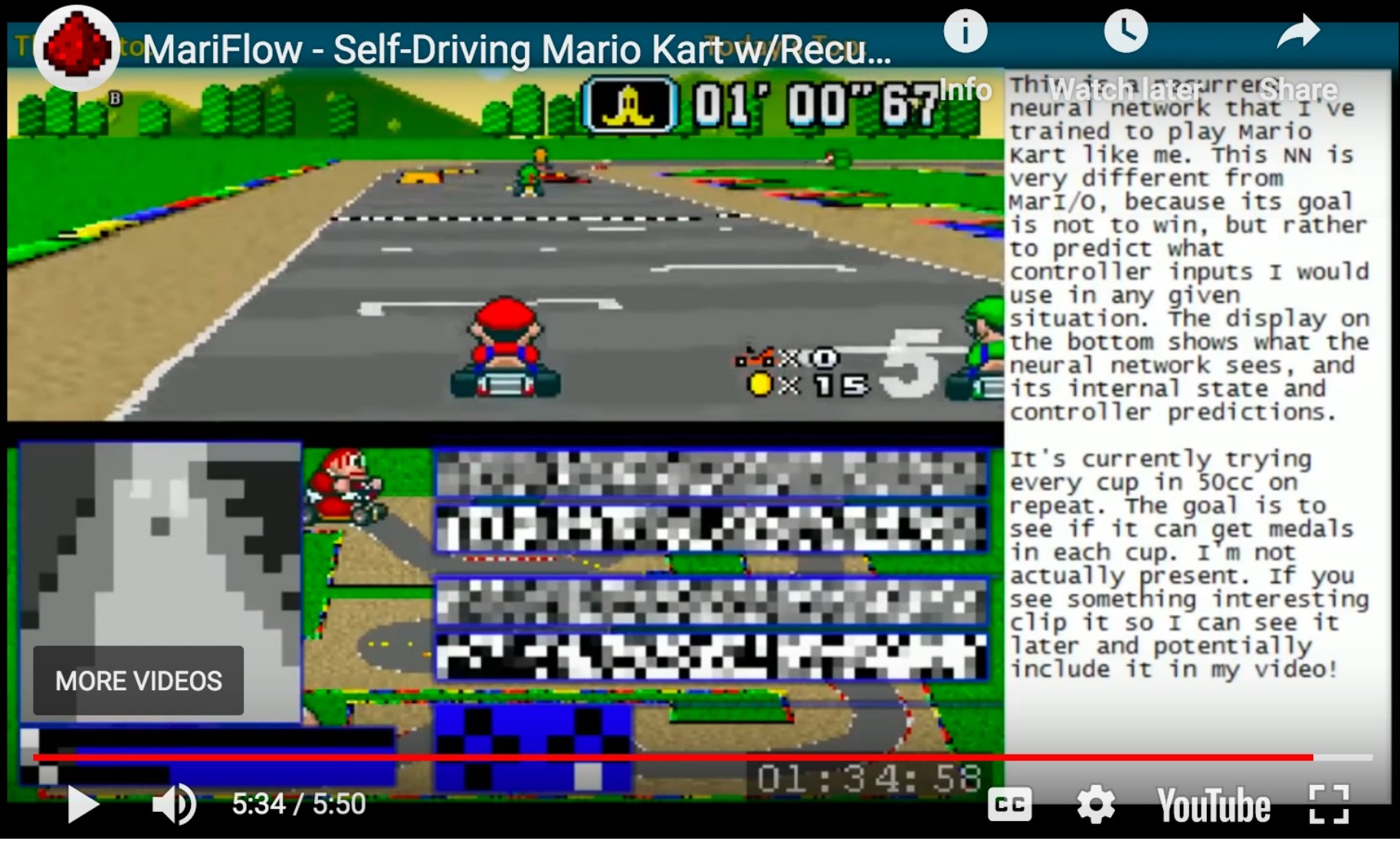
Learn to recognize palindrome
Sequence size between 1 to 10

$\mathbf{x}$	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

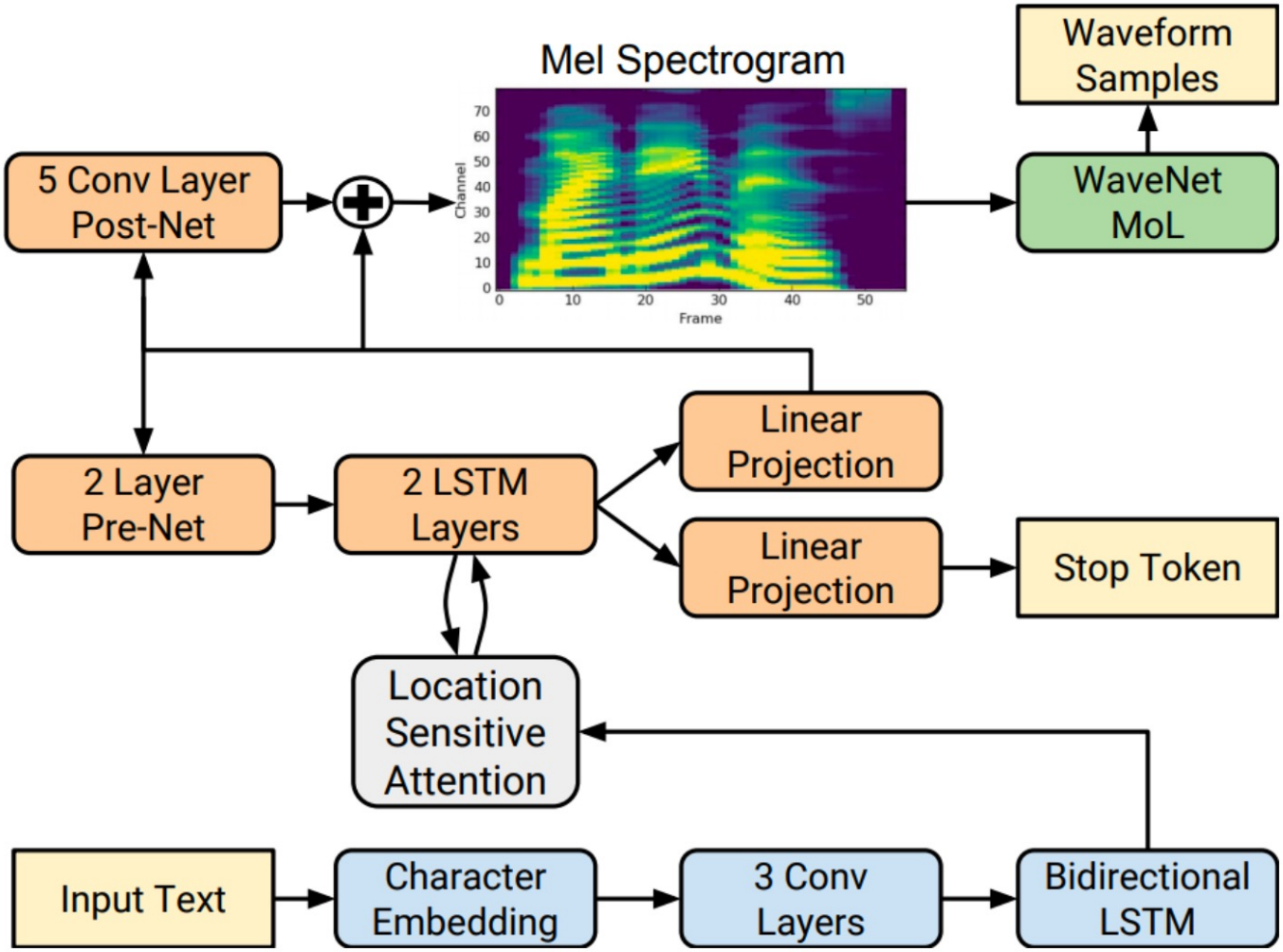
Examples



Self-driving Mario Kart with RNN: YouTube video

Examples

Text-to-speech synthesis



Shen et al., 2017

Deep Sets

What if our data has no time structure?

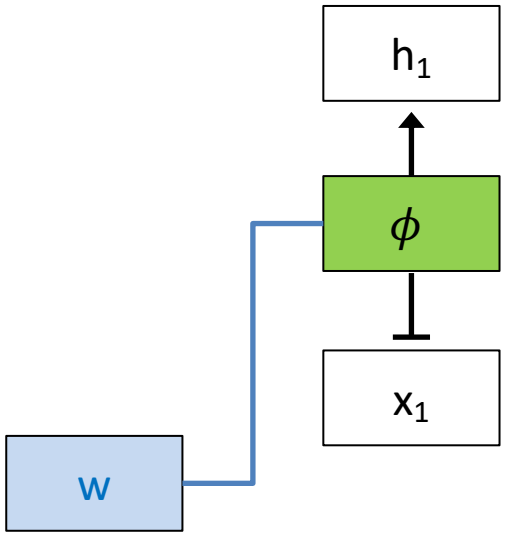
- Data may be variable in length but have no temporal structure → *Data are sets of values*

- *One option*: If we know about the data domain, could try to impose an ordering, then use RNN

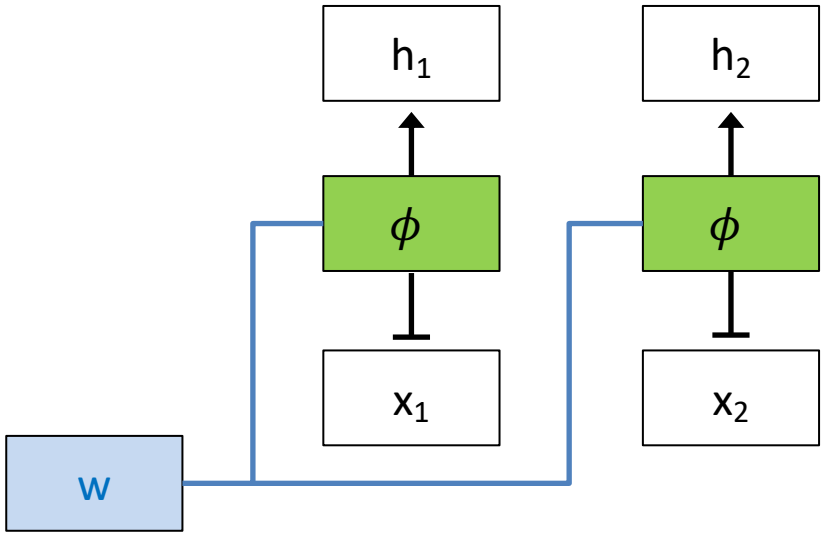
- *Better option*: use system that can operate on variable length sets in permutation invariant way

  – Why permutation invariant → so order doesn't matter

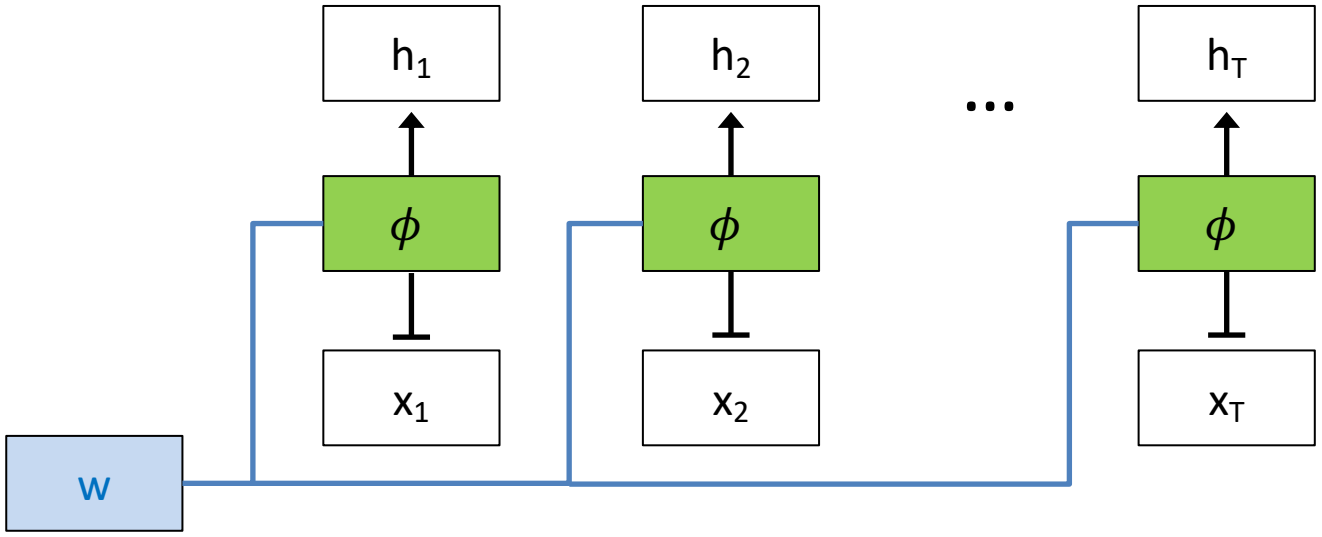
Deep Sets



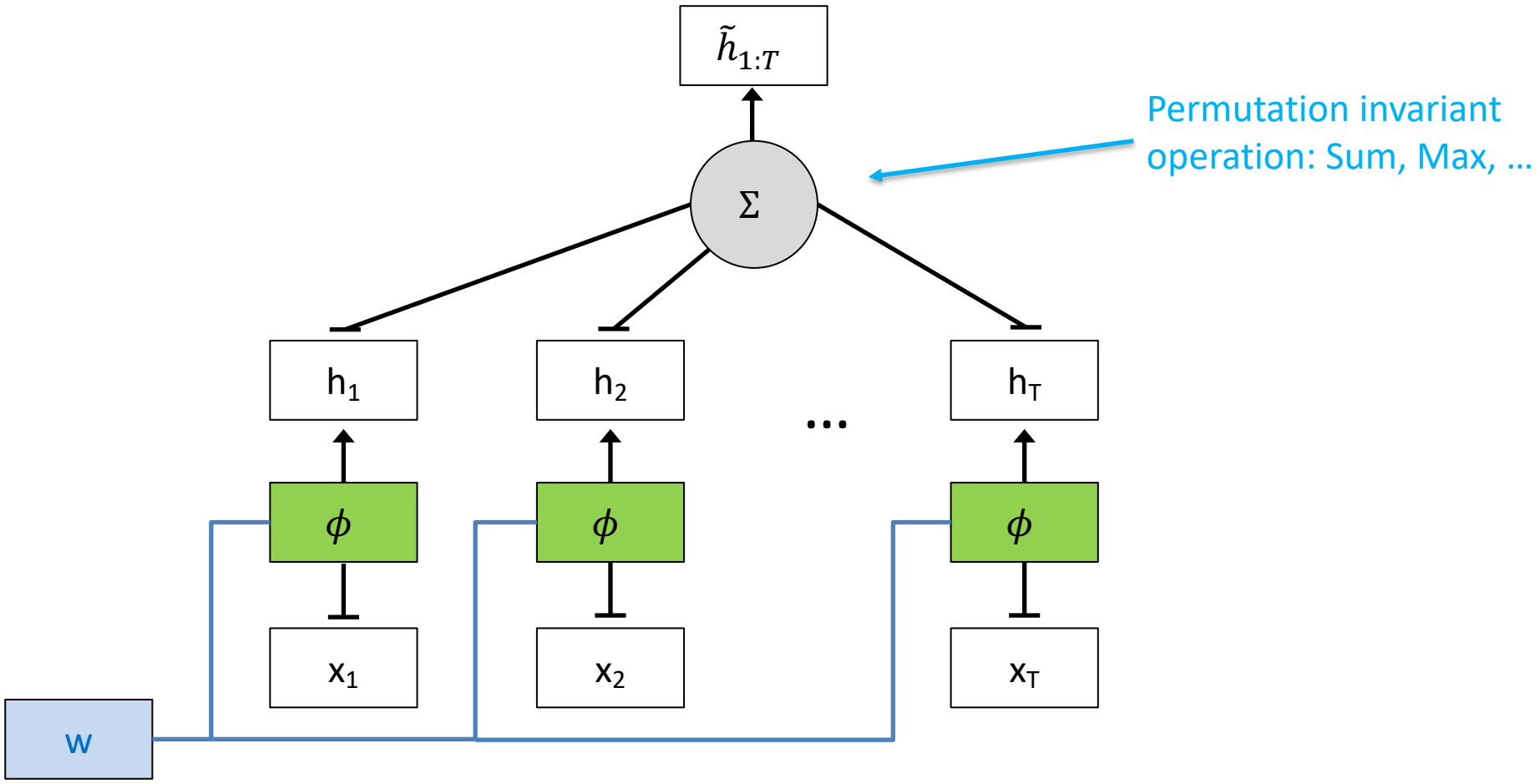
Deep Sets



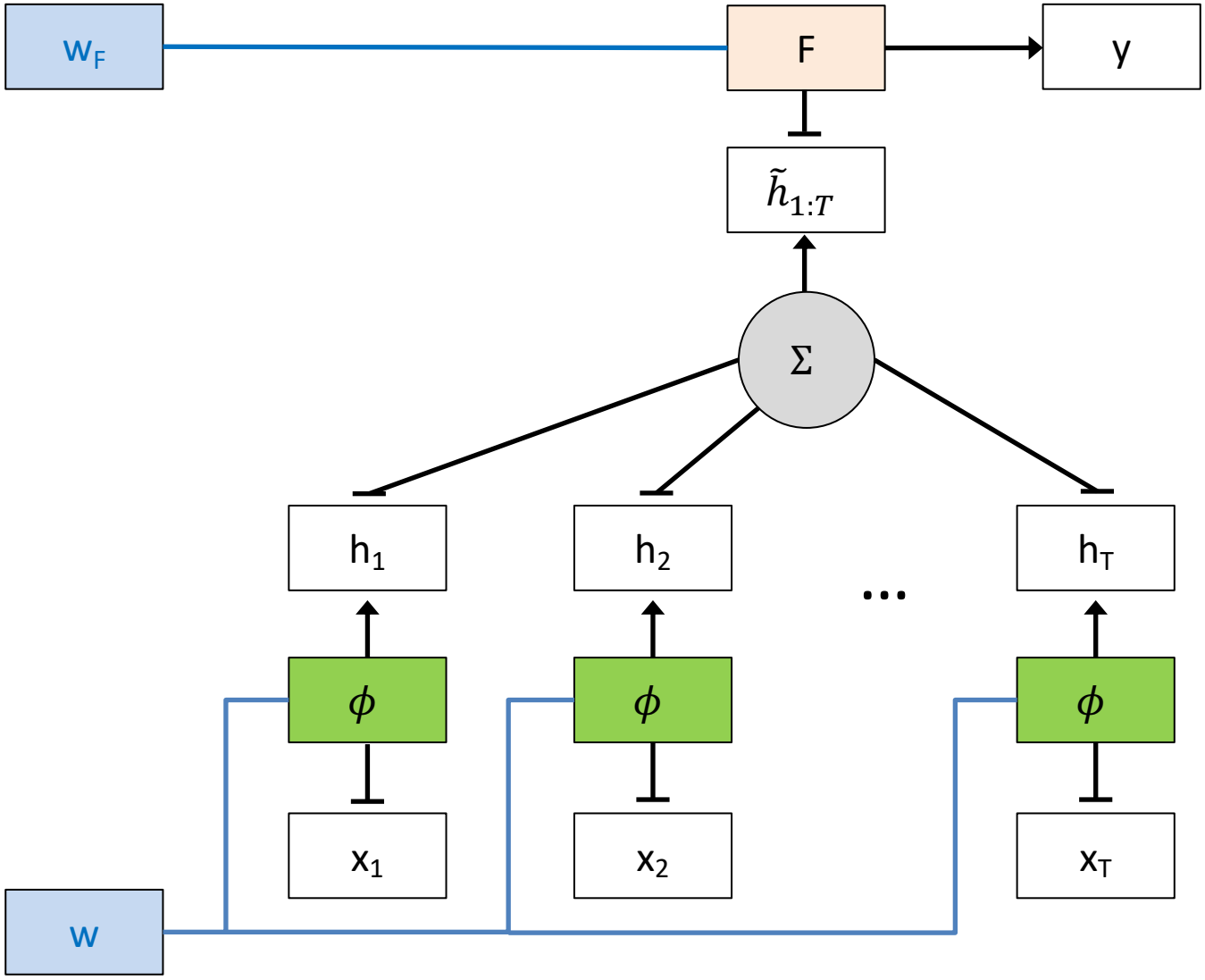
Deep Sets



Deep Sets


$\tilde{h}_{1:T}$
Σ
Permutation invariant operation: Sum, Max, …
h1
h2
…
hT
ϕ
ϕ
ϕ
x1
x2
xT
w

Deep Sets



Examples

Outlier detection

M. Zaheer et. al 2017

Medical Imaging

With more complex architecture

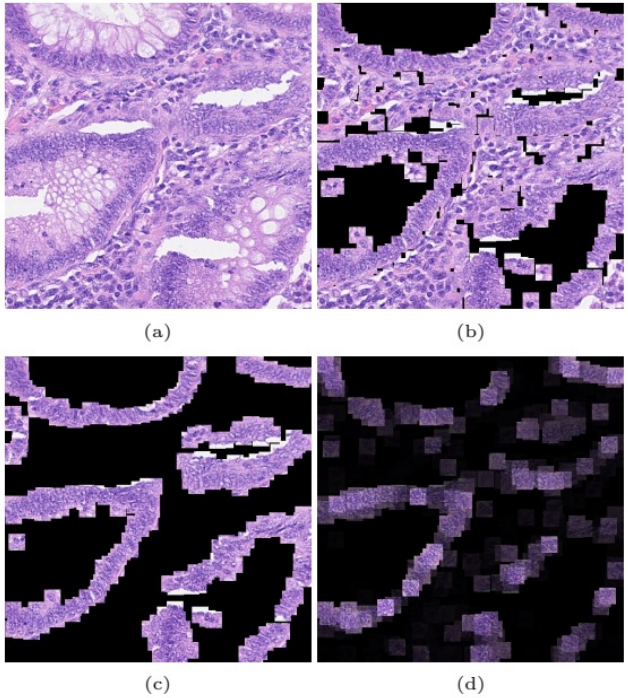
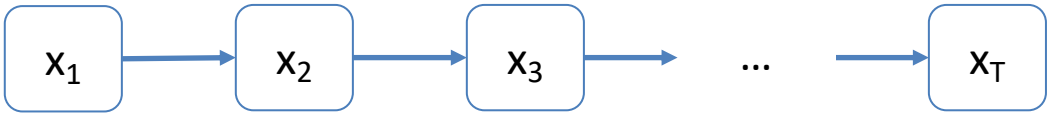


Figure 5. (a) H&E stained histology image. (b) 27×27 patches centered around all marked nuclei. (c) Ground truth: Patches that belong to the class epithelial. (d) Heatmap: Every patch from (b) multiplied by its corresponding attention weight, we rescaled the attention weights using $a'_k = (a_k - \min(\mathbf{a}))/(\max(\mathbf{a})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

M. Ilse et al., 2018

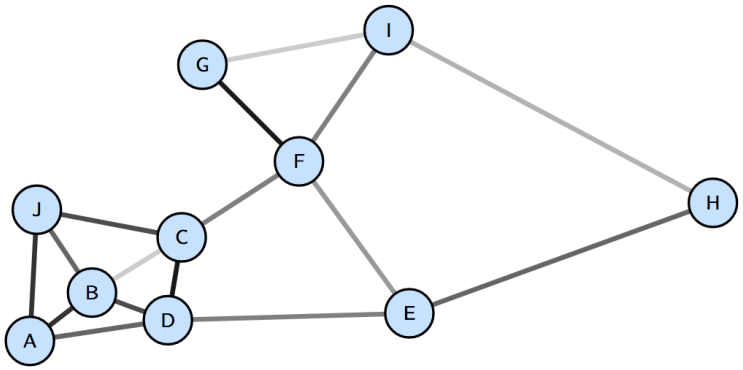
Graph Data

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



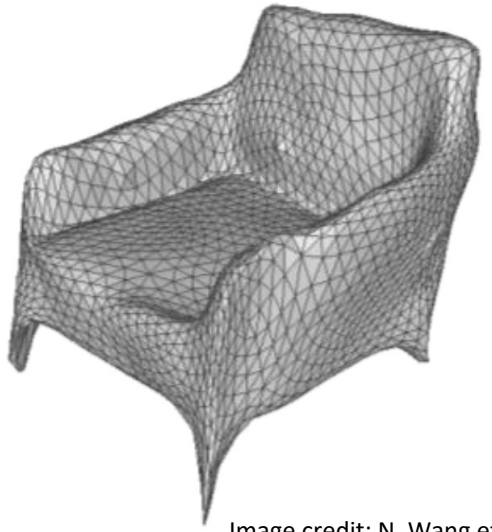
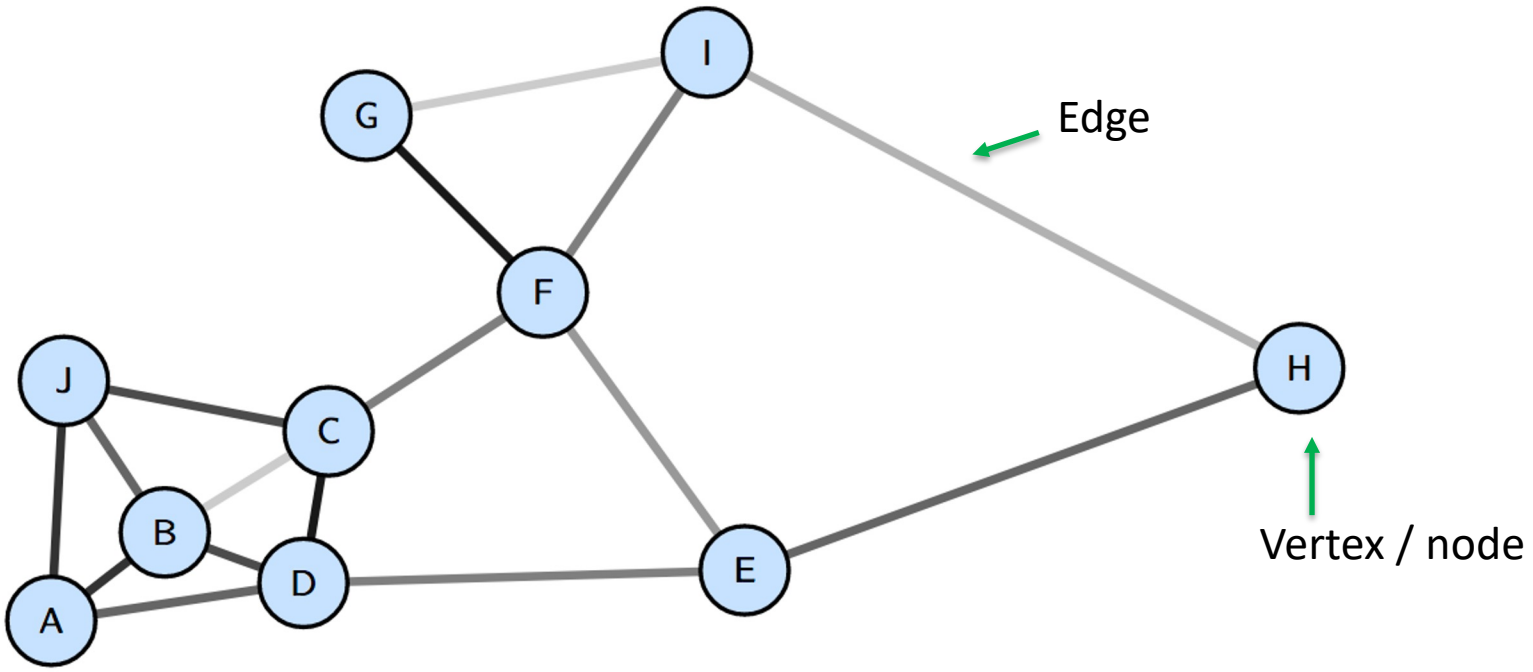


Image Credit: I. Henrion

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

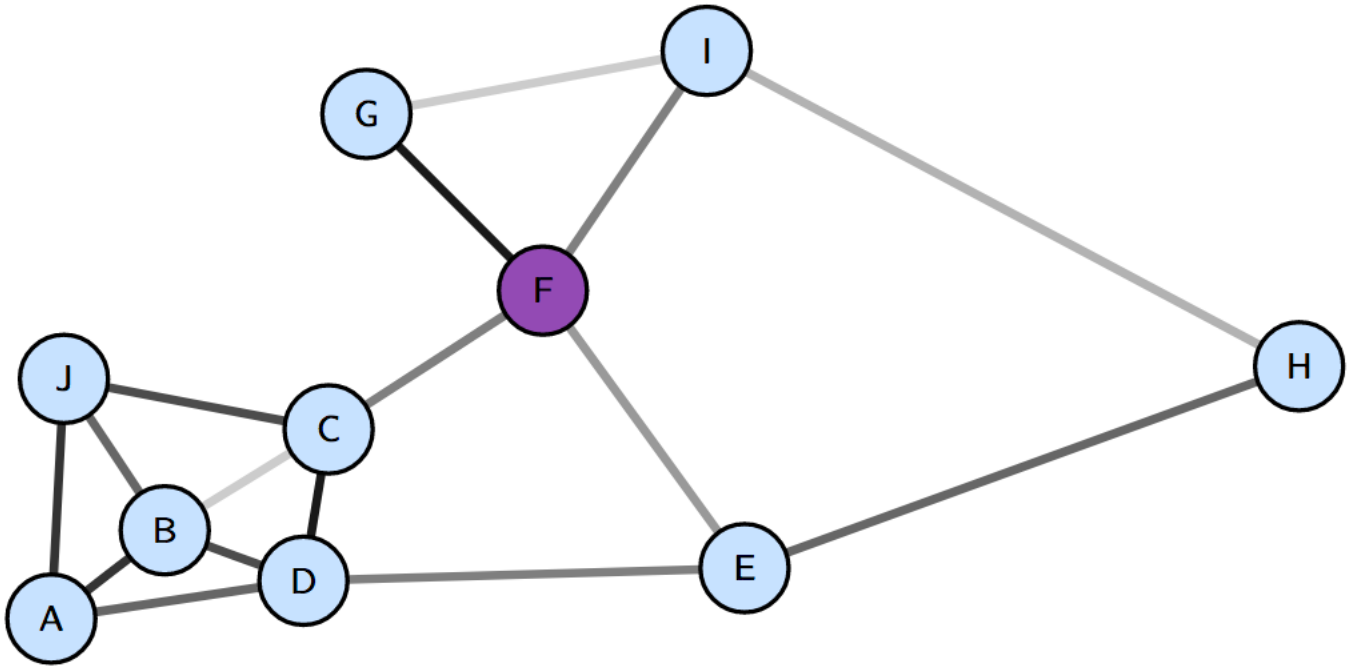
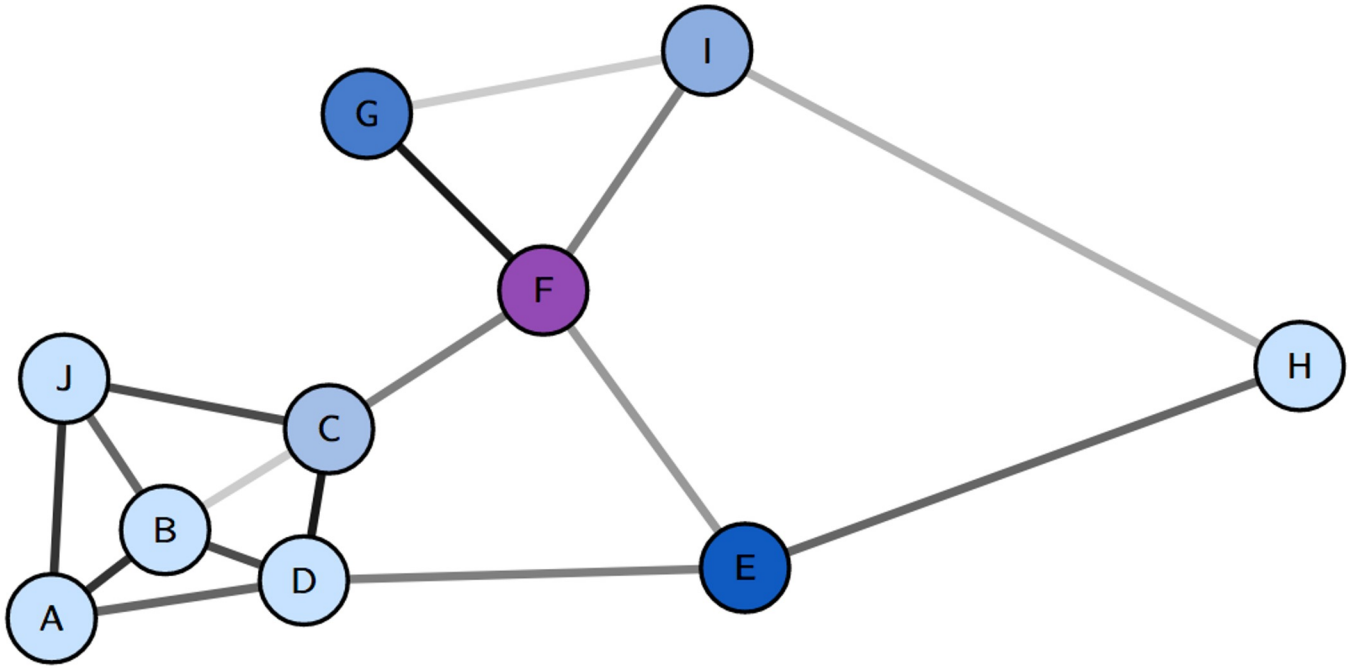


Image Credit: I. Henrion

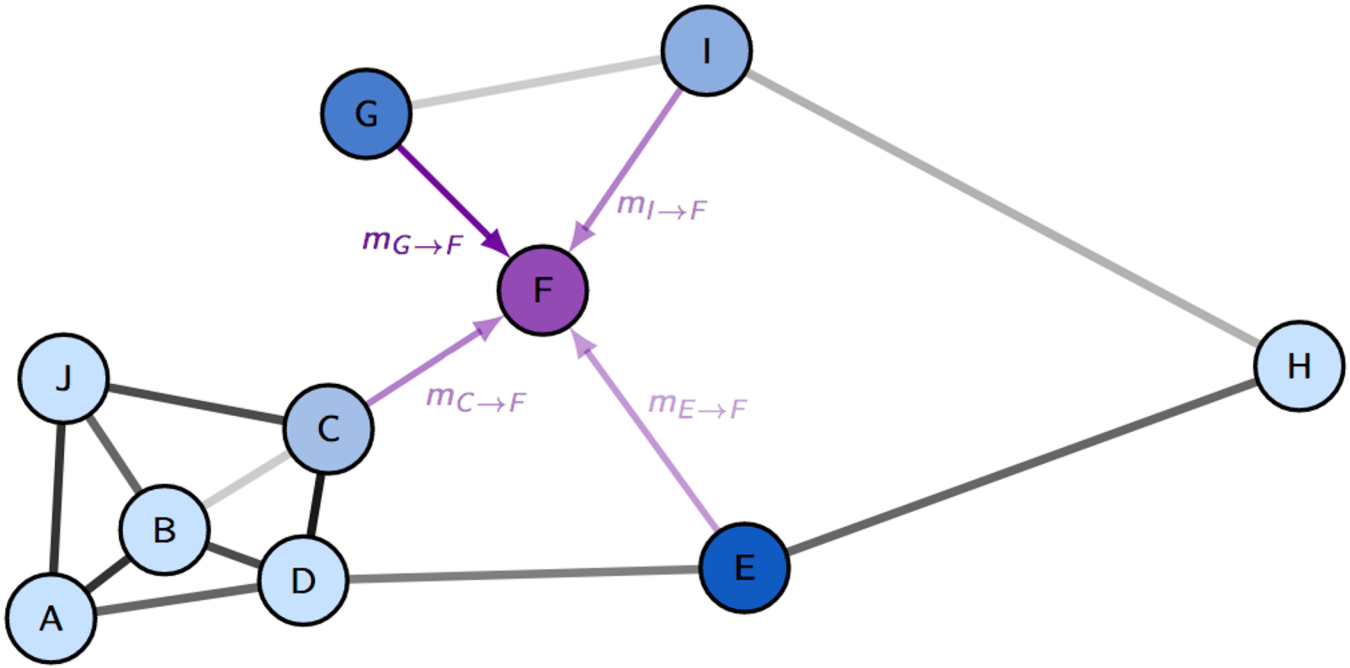
Neural Message Passing



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

Image Credit: I. Henrion

Neural Message Passing

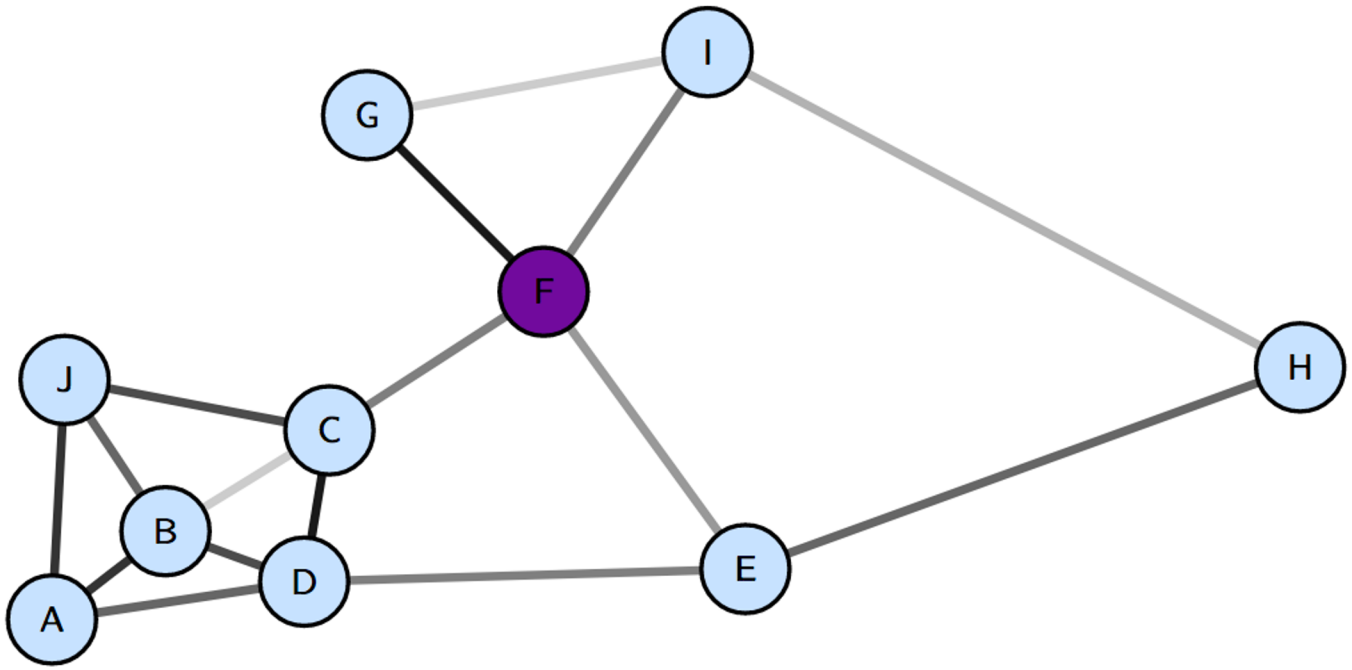


$$\tilde{m}_j^t = f(h_j^{t-1})$$

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

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

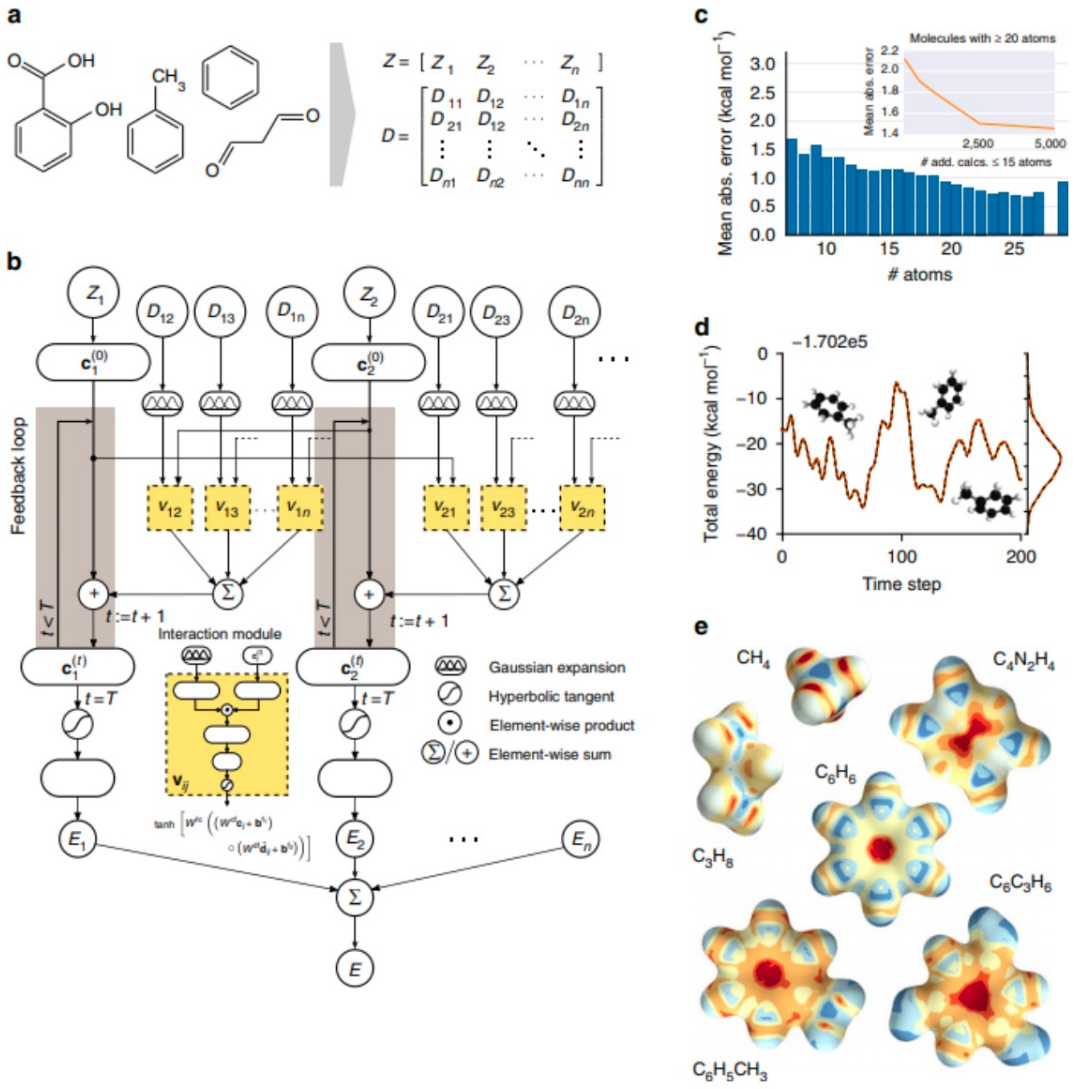
$\mathbf{r} = \text{Readout}(\mathbf{h})$

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

Image Credit: I. Henrion

Examples

Quantum chemistry with graph networks



Schutt et al. 2017

Examples

Learning to simulate physics with graph networks

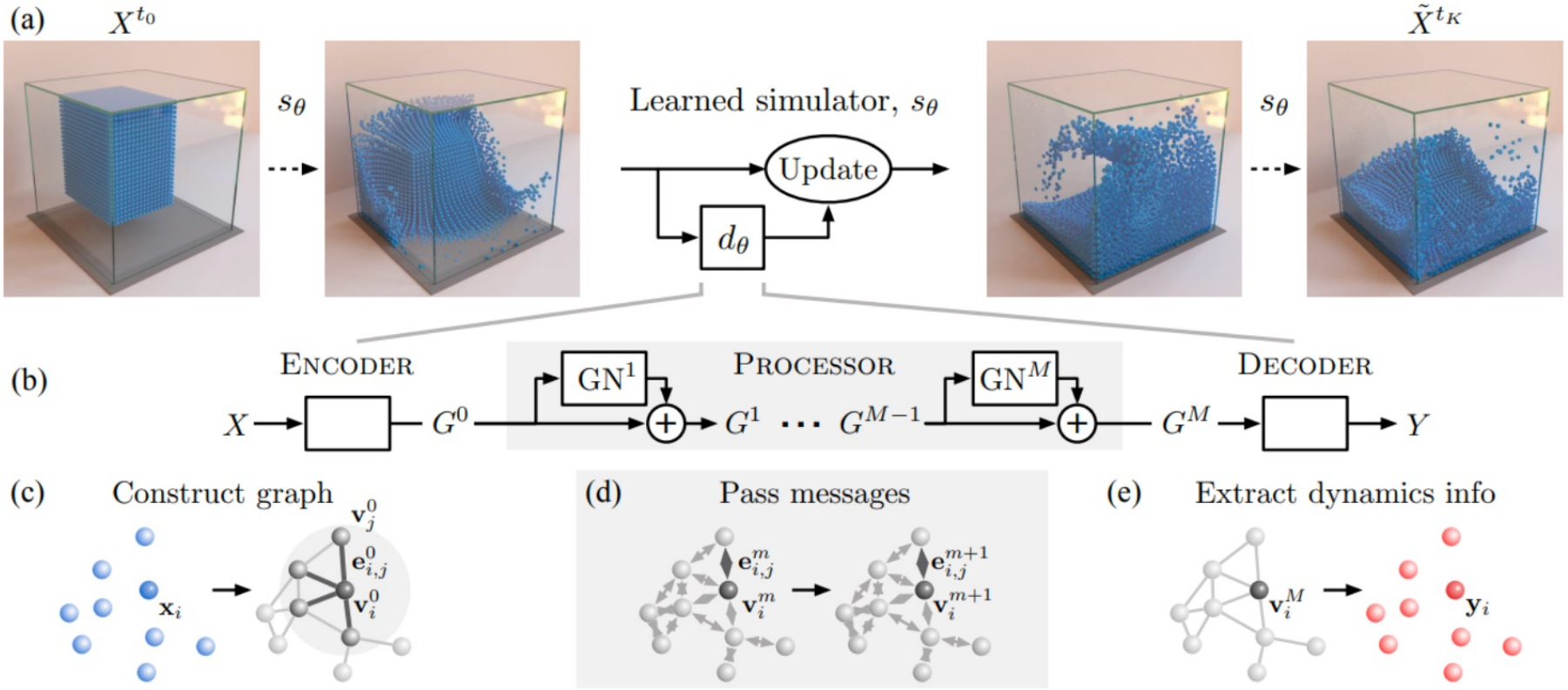


Figure 2. **(a)** Our GNS predicts future states represented as particles using its learned dynamics model, d_θ, and a fixed update procedure. **(b)** The d_θ uses an "encode-process-decode" scheme, which computes dynamics information, Y, from input state, X. **(c)** The ENCODER constructs latent graph, 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

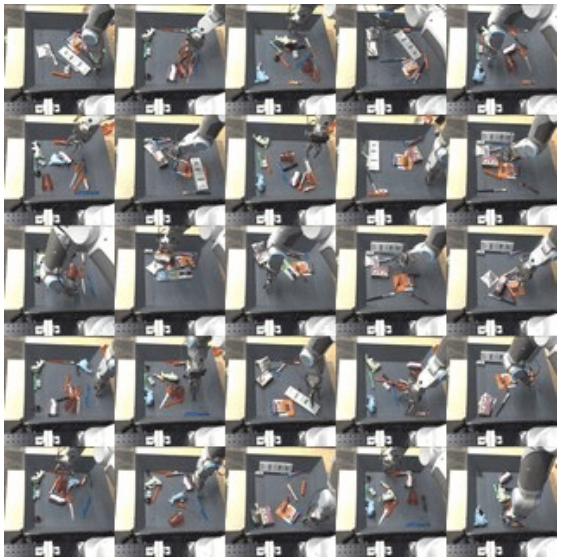
Deep Generative Model Examples

BigGan

(Brock et al, 2018)

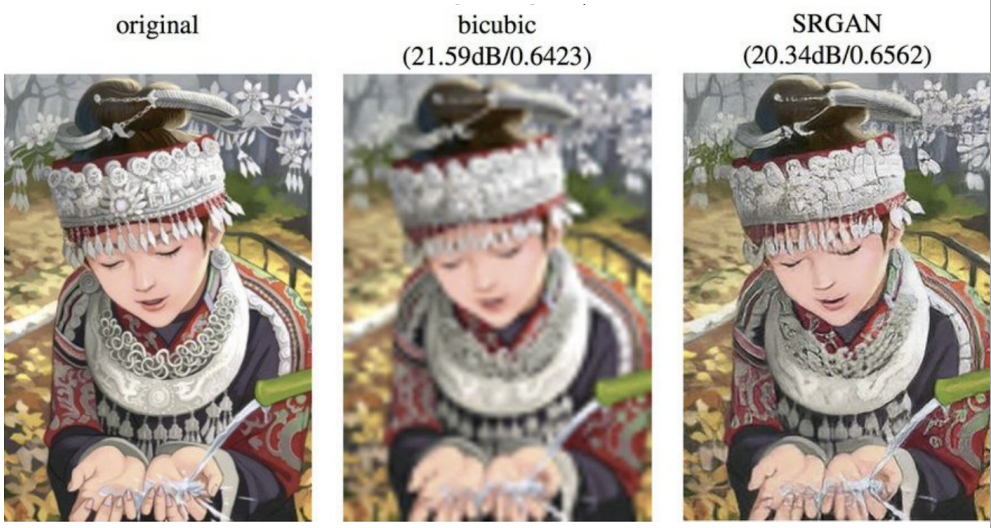
Image-to-Image Translation with CycleGAN

Zhu et. al. 2017

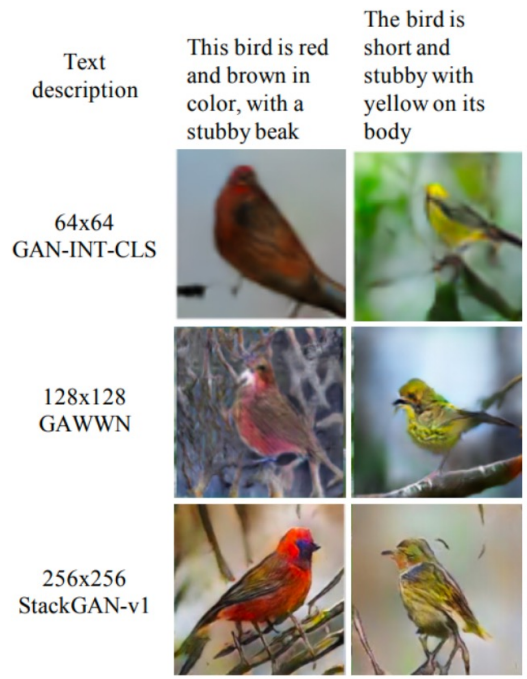


Simulate future trajectories of environments based on actions for planning. (Finn et al, 2016)

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



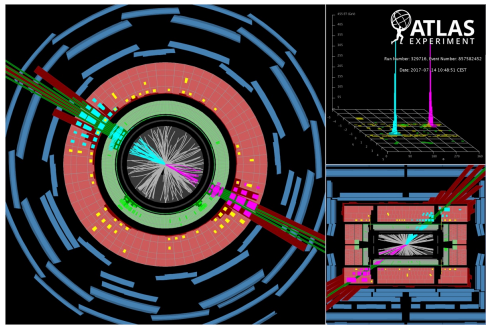
Text-to-Image Synthesis with StackGAN (Zhang et. al. 2017)

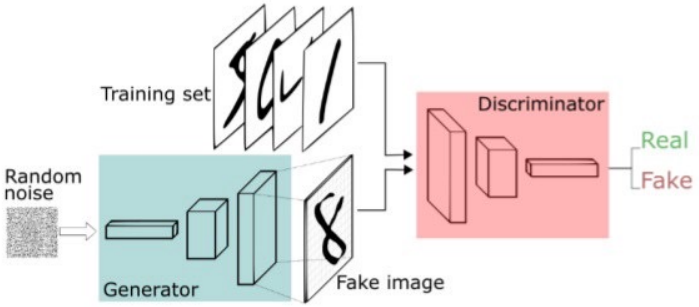


Generative Models

Different Kinds of Generative Models

Generative Models approximate and simulate the data generation process





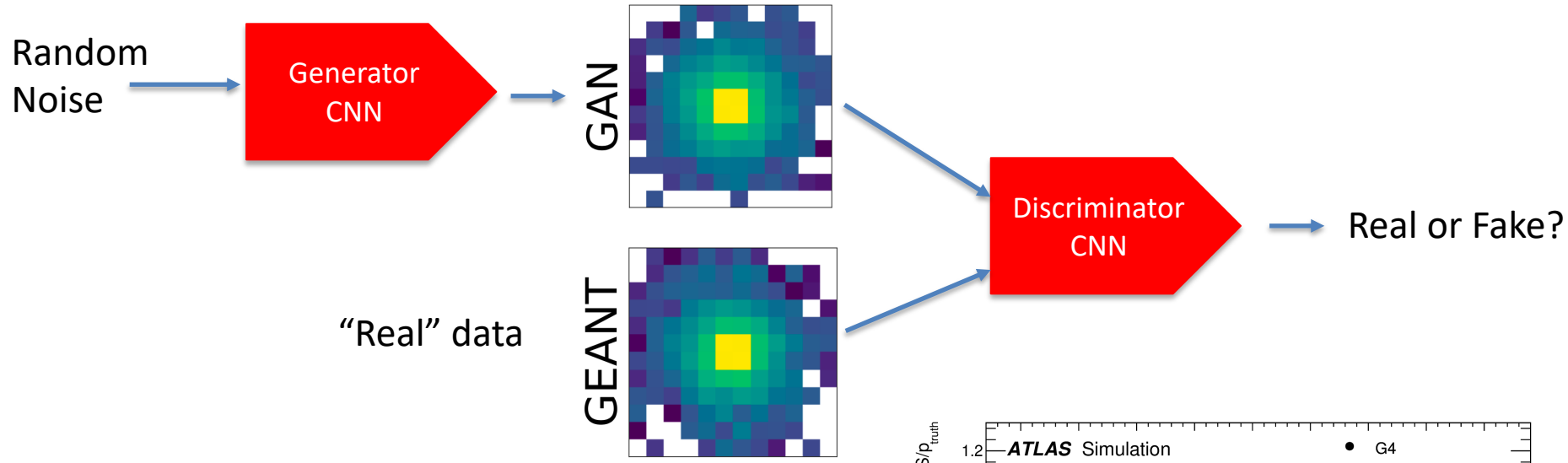
Scientific Simulators

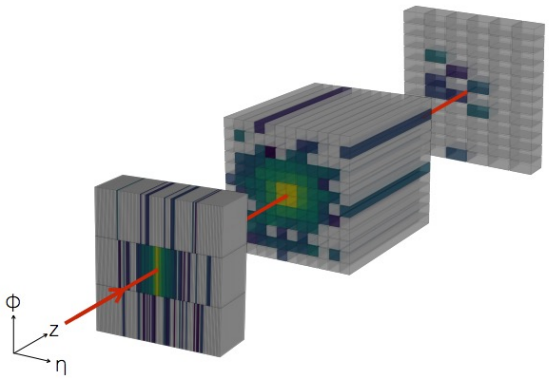
- Built from science knowledge
- Relatively few parameters, often interpretable
- High Fidelity, often computationally costly

Machine Learning

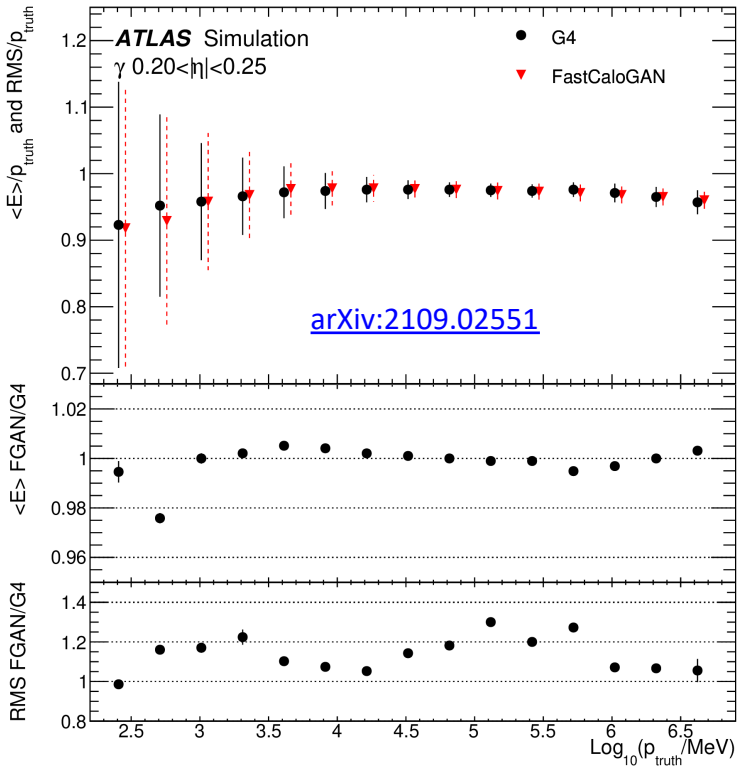
- Fit to data, inductive bias in model design / optimization
- Can have $>10^6$ parameters, often not interpretable
- Often slow to train, fast to evaluate

GANs for Calorimeter Energy Depositions





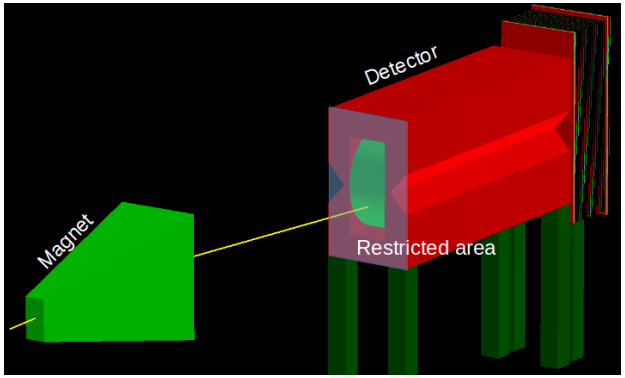
PRD97, 014021 (2018)
arXiv:1705.02355
arXiv:1701.05927

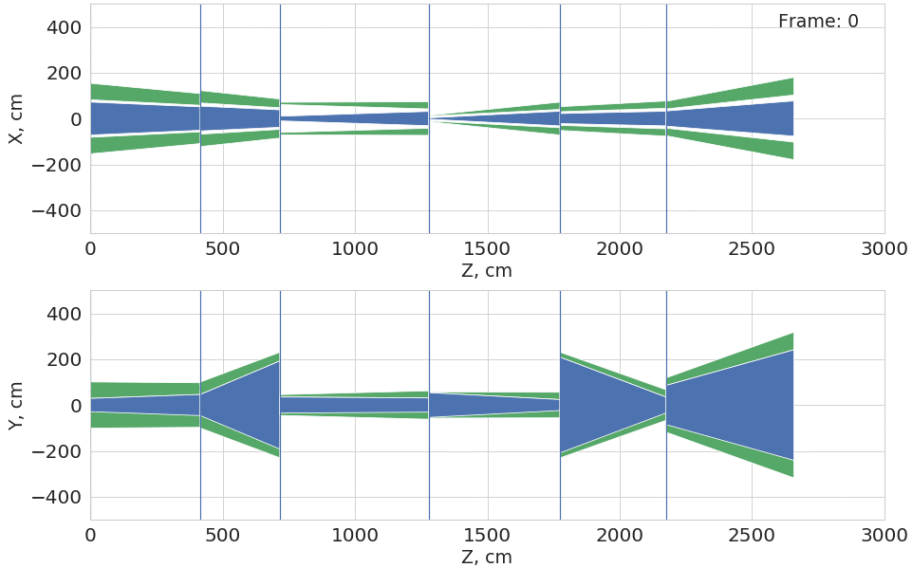


GANs for Detector Design

- 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

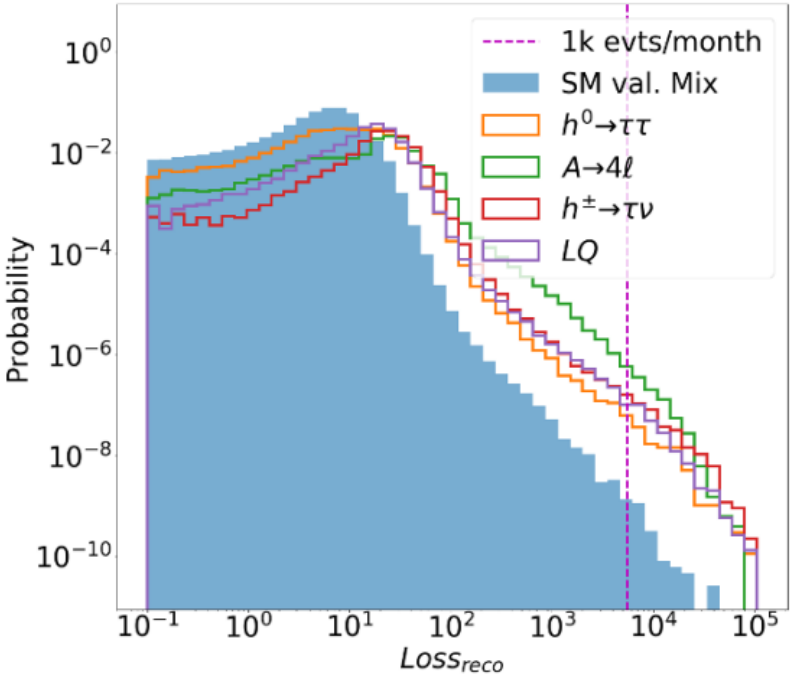
Magnet Optimization

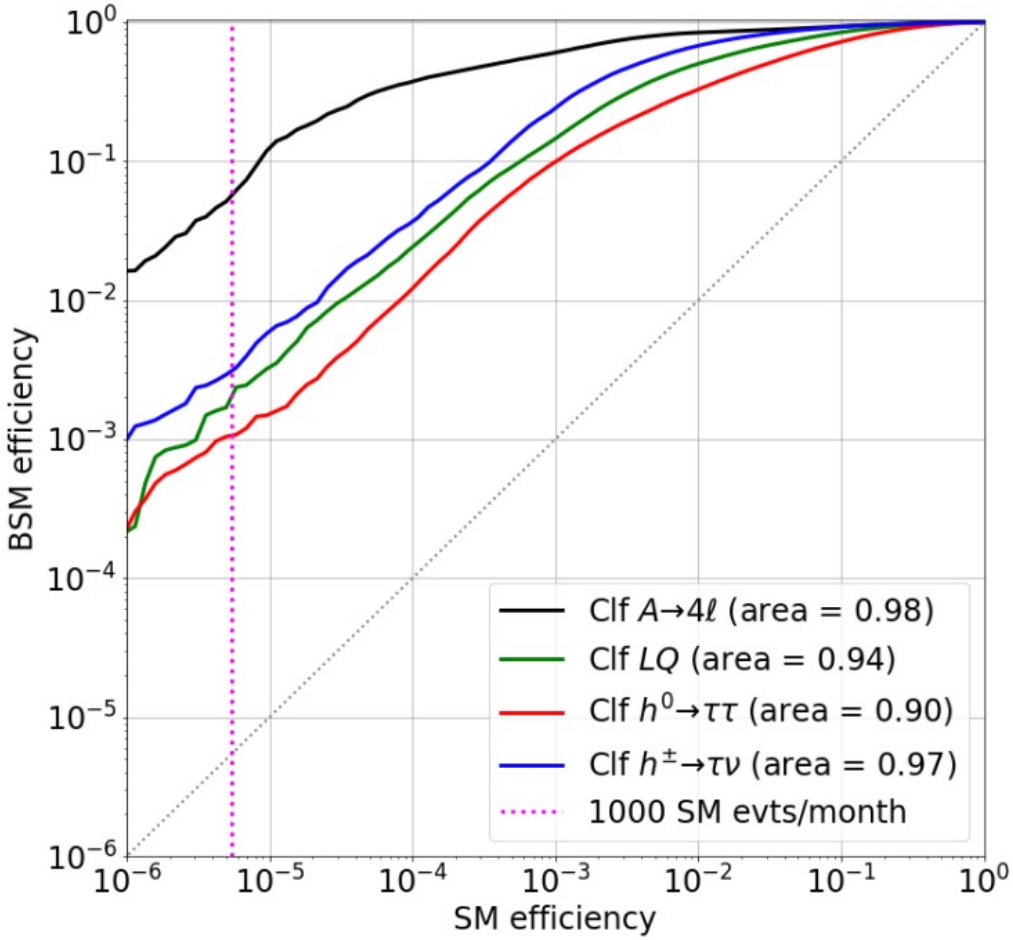




NeurIPS **33**, 14650-14662 (2020)

Anomaly Detection with VAE





- Train on SM
- Test if New Physics is poorly reconstructed

JHEP05(2019)036