Introduction to Machine Learning:

Lecture 3 – Intro to Deep Learning

Michael Kagan

SLAC NATIONAL ACCELERATOR LABORATORY

TRISEP Summer School
July 8-12, 2024

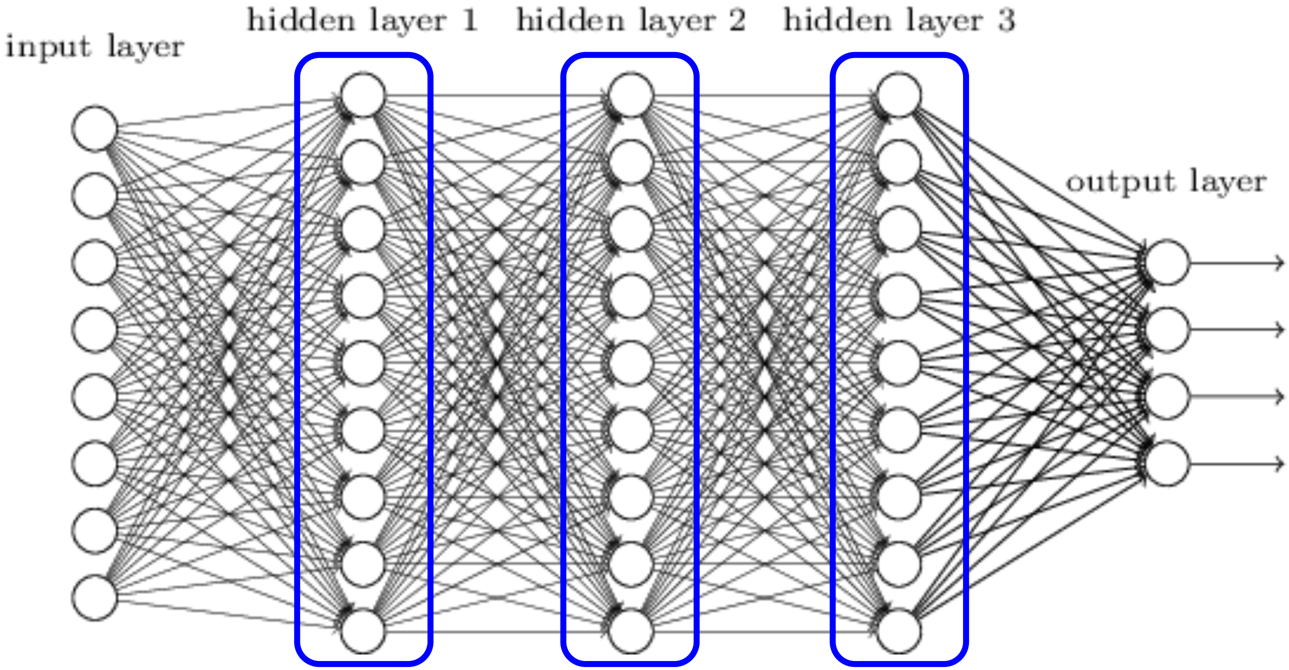
The Plan

- Lecture 1 – Machine Learning Fundamentals
- Lecture 2 – Intro to Neural Networks
- Lecture 3 – Intro to Deep Learning
- Lecture 4 – Intro to Unsupervised Learning
- Lecture 5 – Intro to Deep Generative Models

Deep Learning Resourses

- Deep Learning is a HUGE field
  - O(10,000) papers submitted to conferences
- I only condensed *some* parts of what you would find in *some lectures* of a Deep Learning course
  - More details from other lecturers!
- Highly recommend Online-available lectures:
  - Francois Fleuret course at University of Geneva
  - Gilles Louppe course at University of Liege

Deep Neural Networks



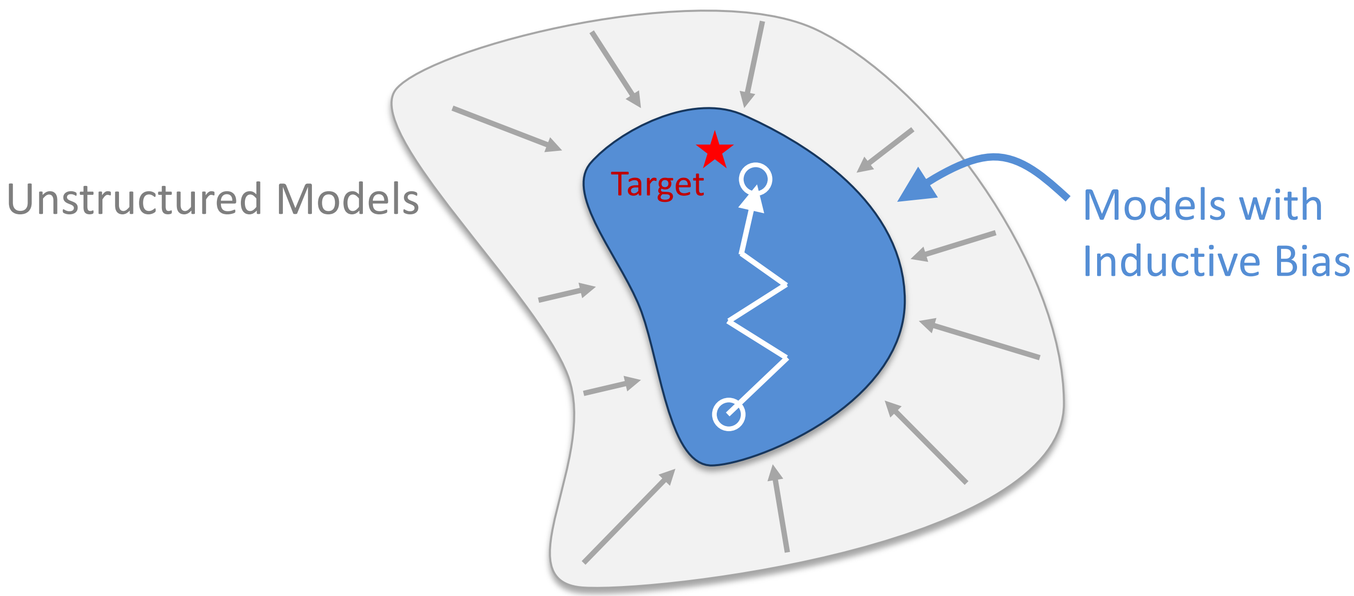
- As data complexity grows, need exponentially large number of neurons in a single-layer network to capture all structure in data

- Deep networks ***factorize the learning*** of structure across layers

- Difficult to train, recently possible with large datasets, fast computing (GPU/TPU) & new training algs. / network structures

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…

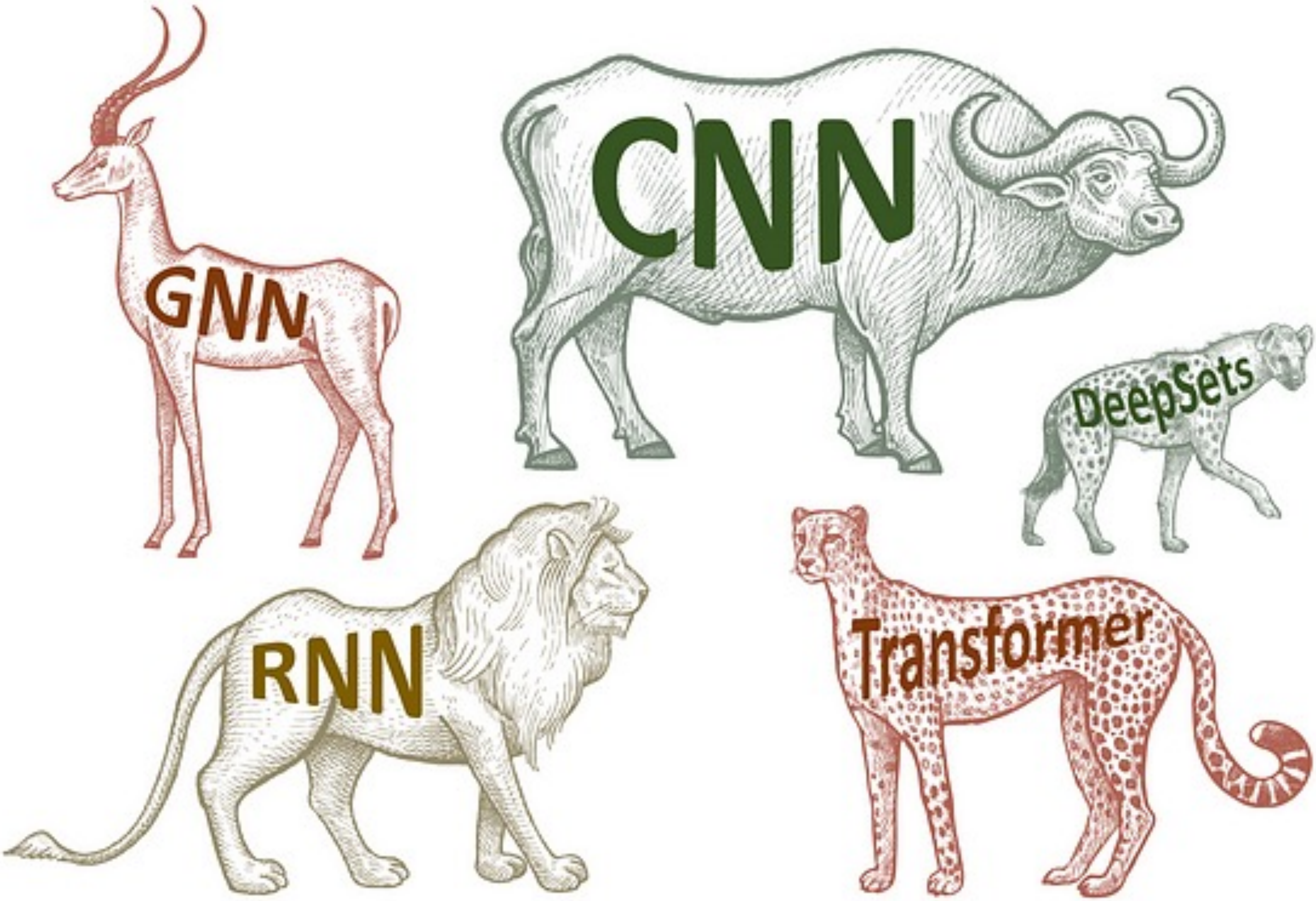
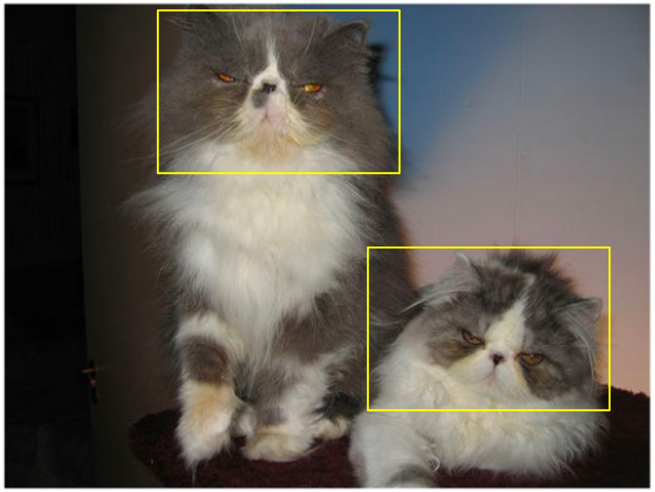


Image credit: Michael Bronstein

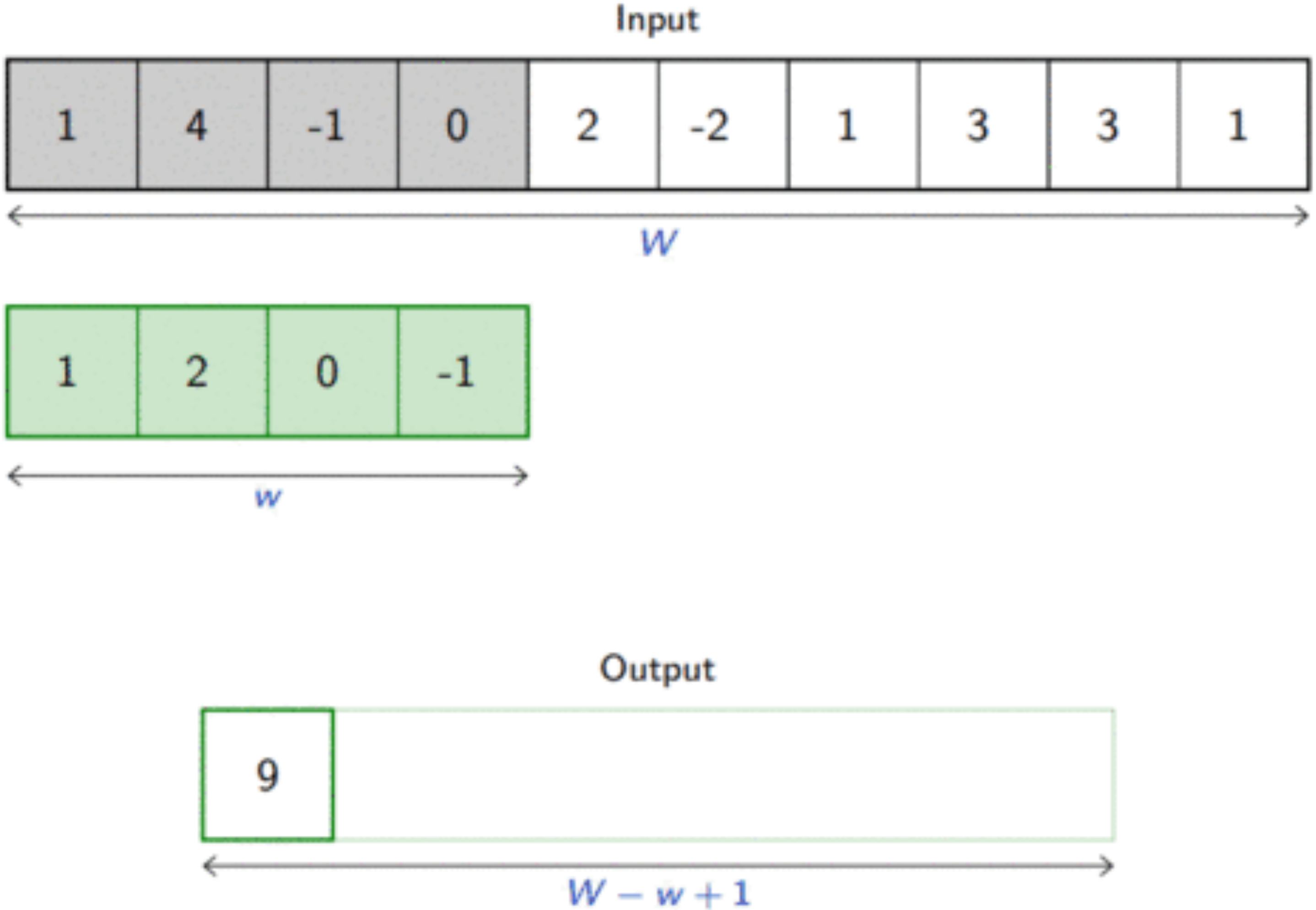
Convolutional Neural Networks

- 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

1D Convolutional Layer Example



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

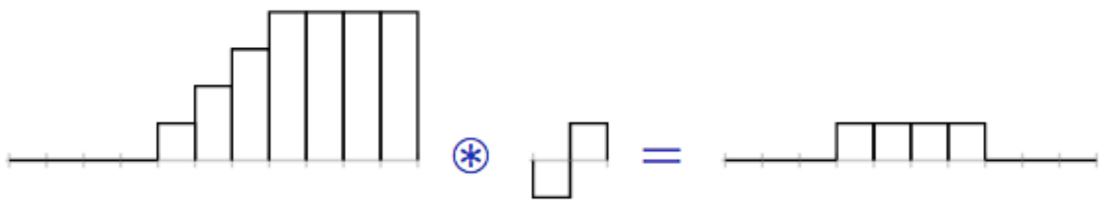
$$(x \circledast \mathrm{u})_i = \sum_{b=0}^{k-1} x_{i+b} u_b$$

- Scan across data and multiply by kernel elements

Convolutional Filters

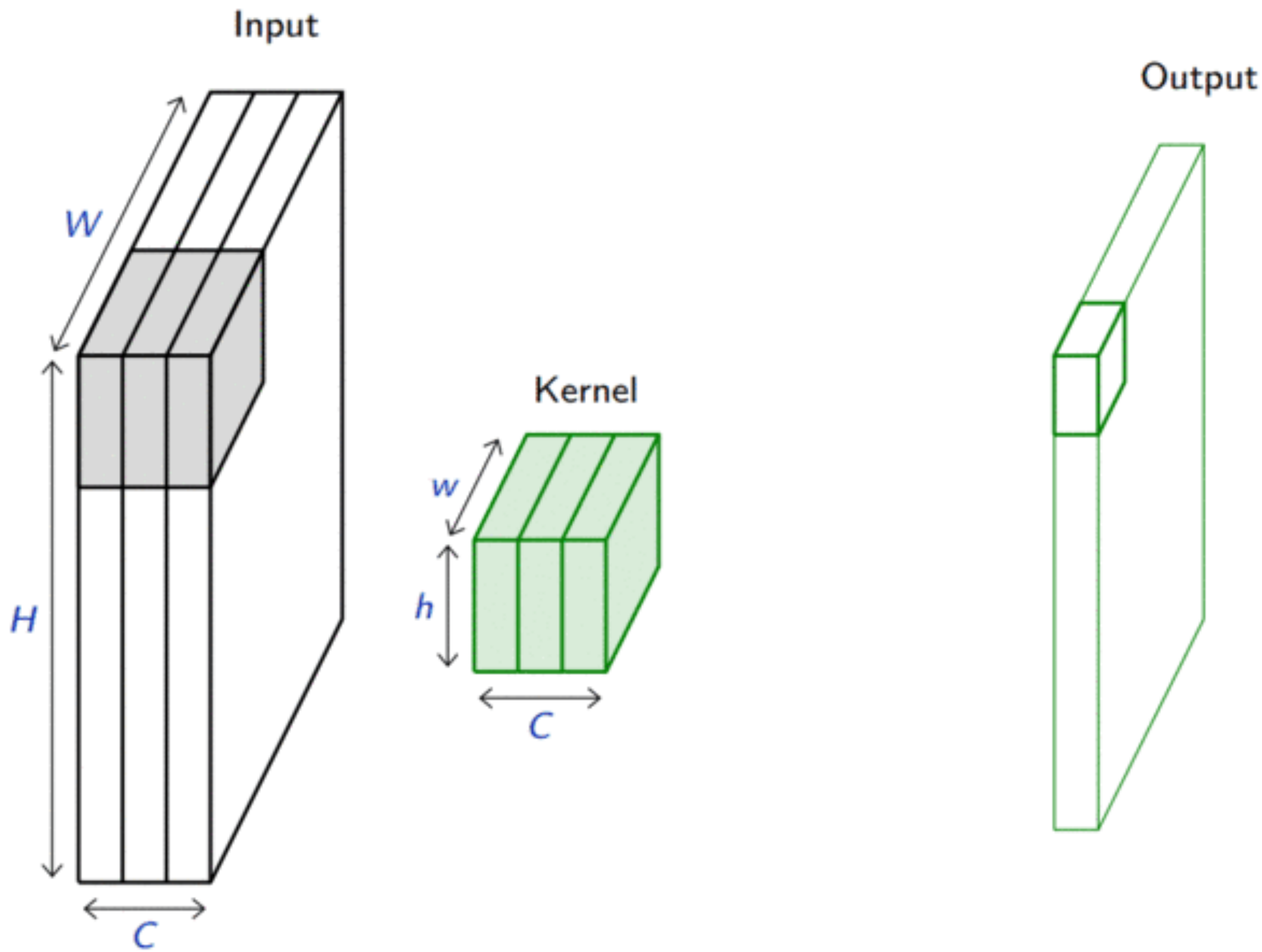
Convolution can implement in particular differential operators, *e.g.*

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



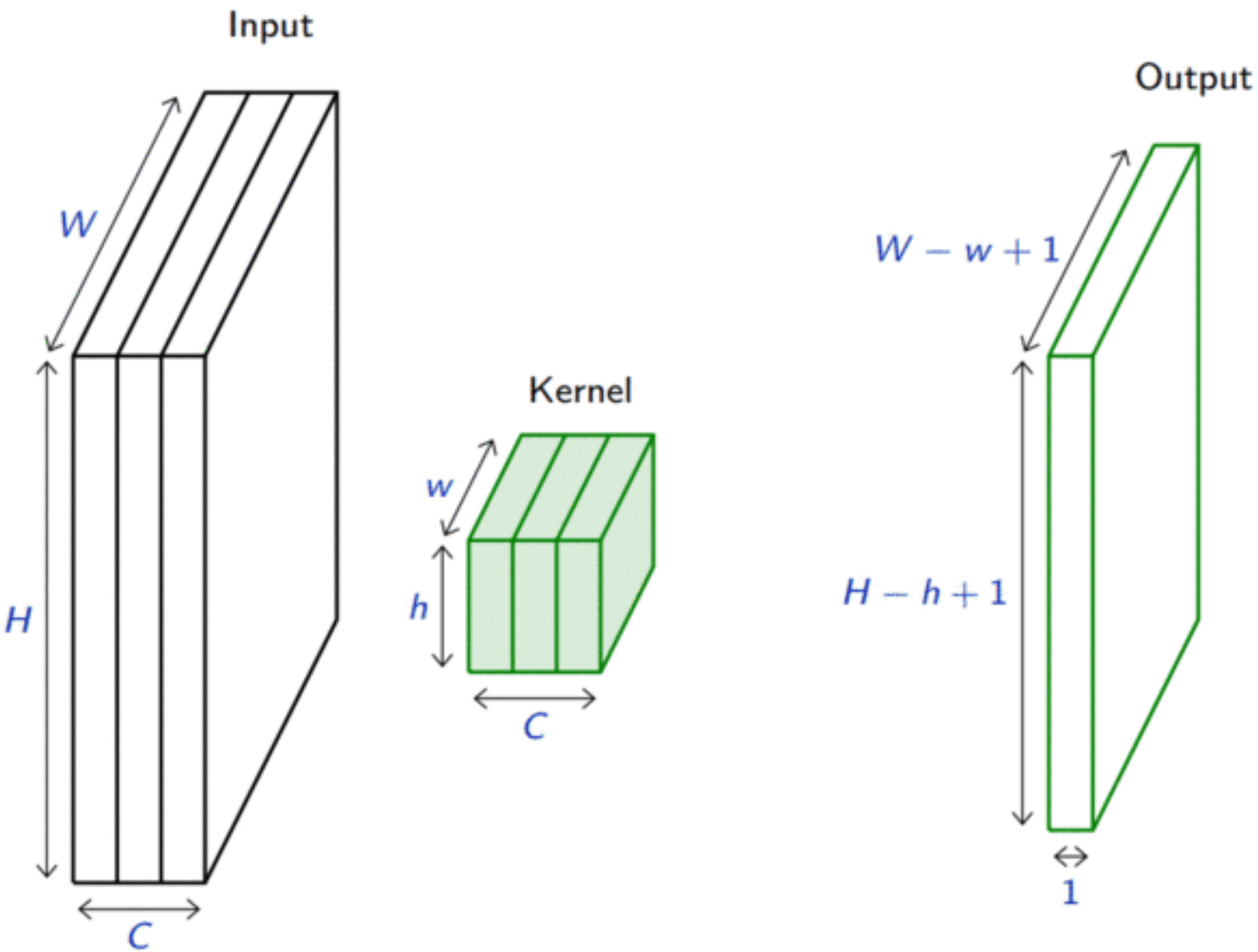
or crude "template matcher", *e.g.*

2D Convolution Over Multiple Channels



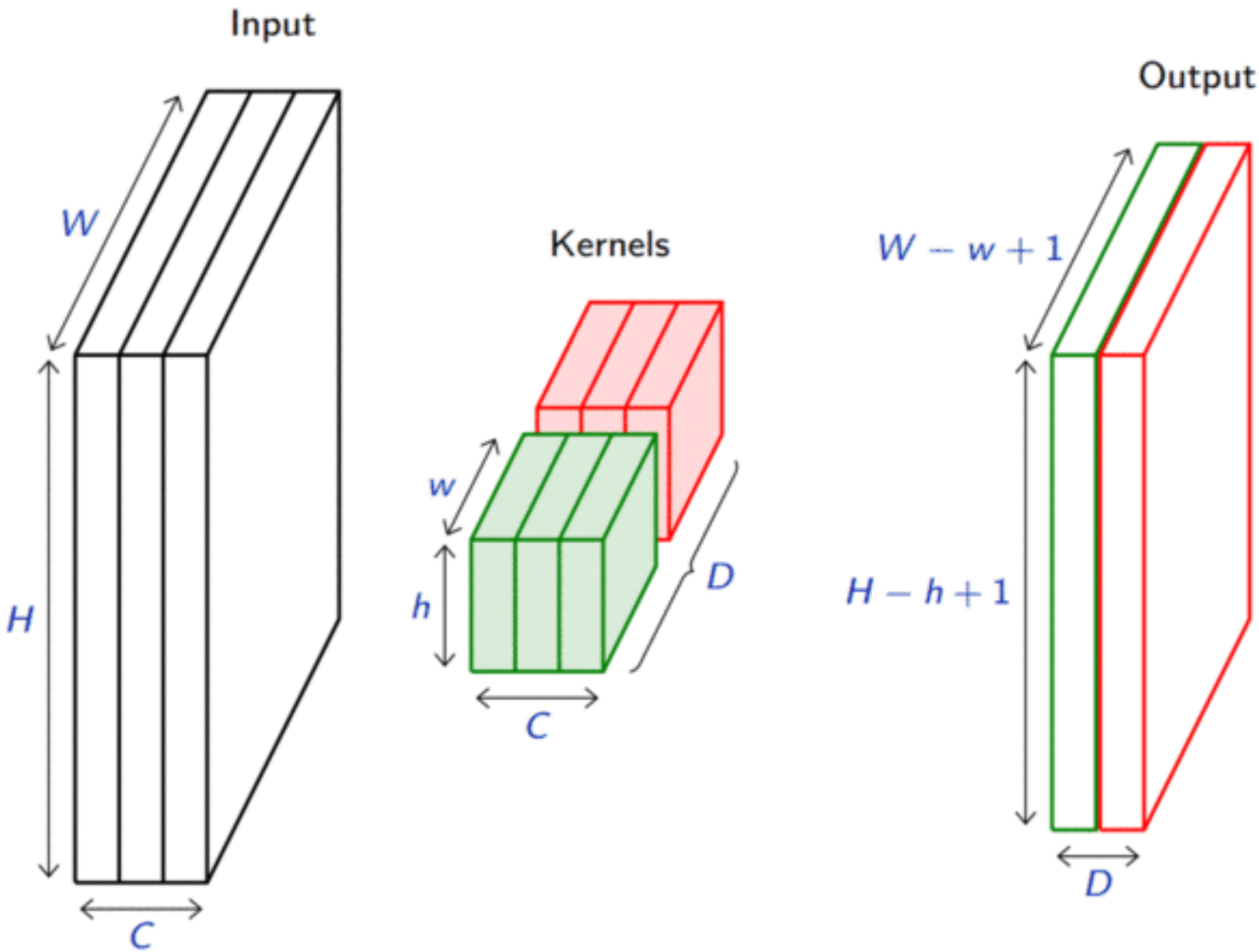
Fleuret, Deep Learning Course

2D Convolution Over Multiple Channels



Fleuret, Deep Learning Course

2D Convolution Over Multiple Channels



Fleuret, [Deep Learning Course]

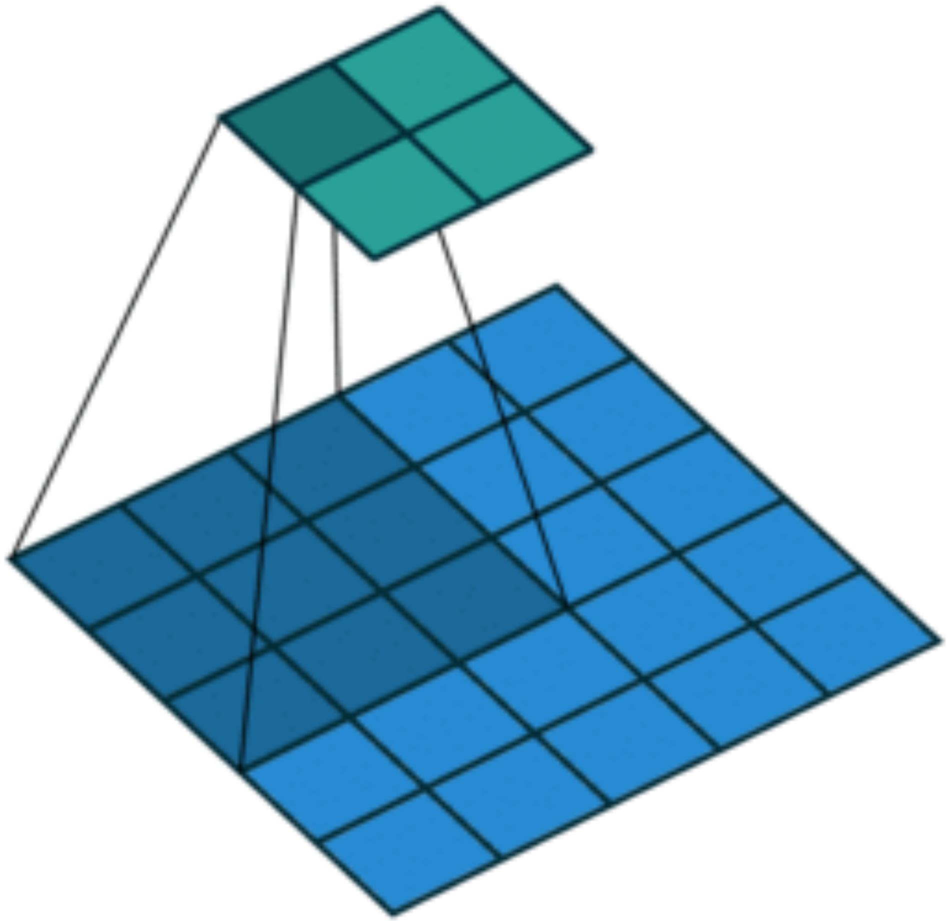
2D Convolutional Layer

- Input data (tensor) **x** of size C×H×W
  - C channels (e.g. RGB in images)
- Learnable Kernel **u** of size C×h×w
  - The size h×w is the *receptive field*

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

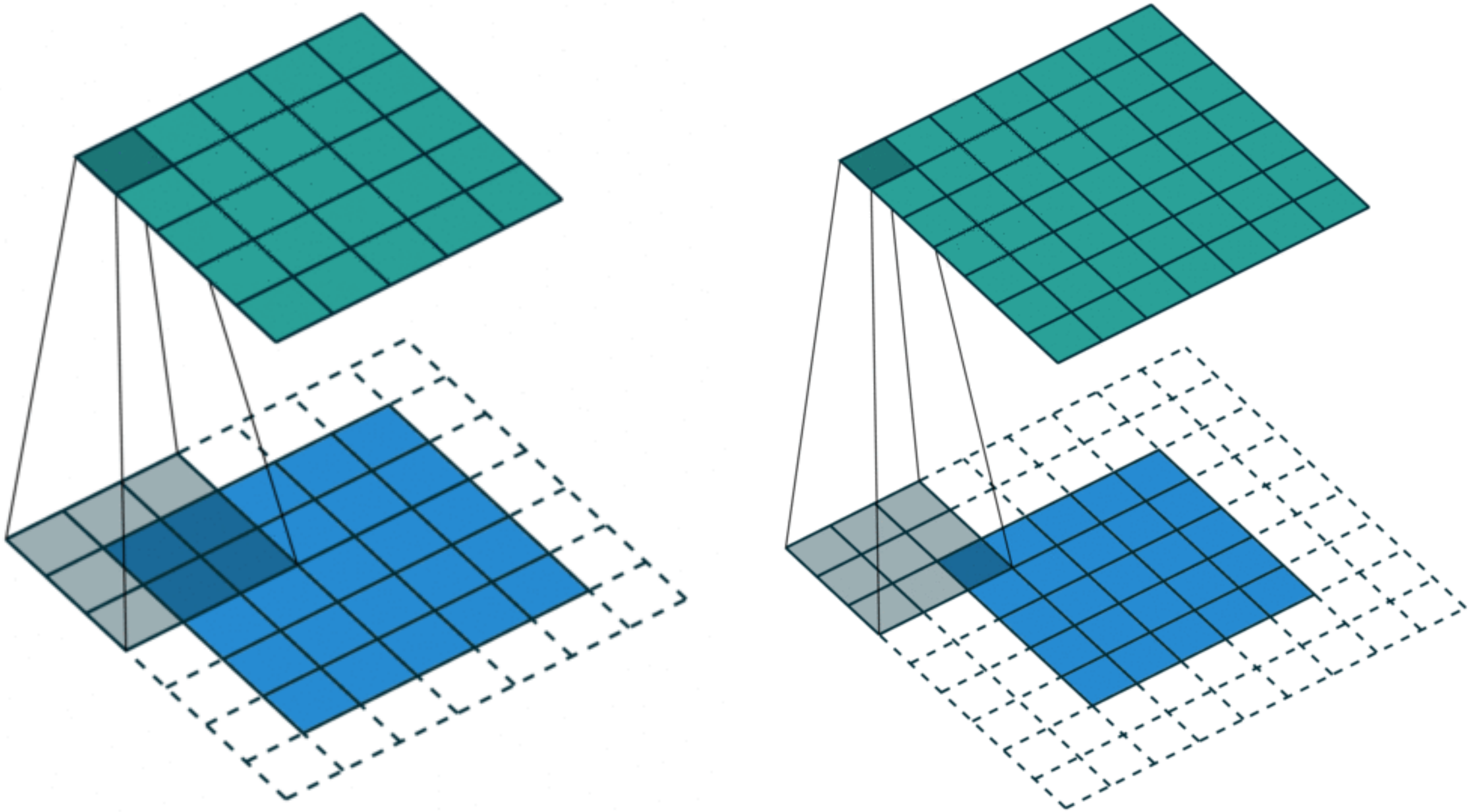
- Output size $(H - h + 1)\times(W - w + 1)$ for each kernel
  - Often called *Activation Map* or *Output Feature Map*

Stride – Step Size When Moving Kernel Across Input



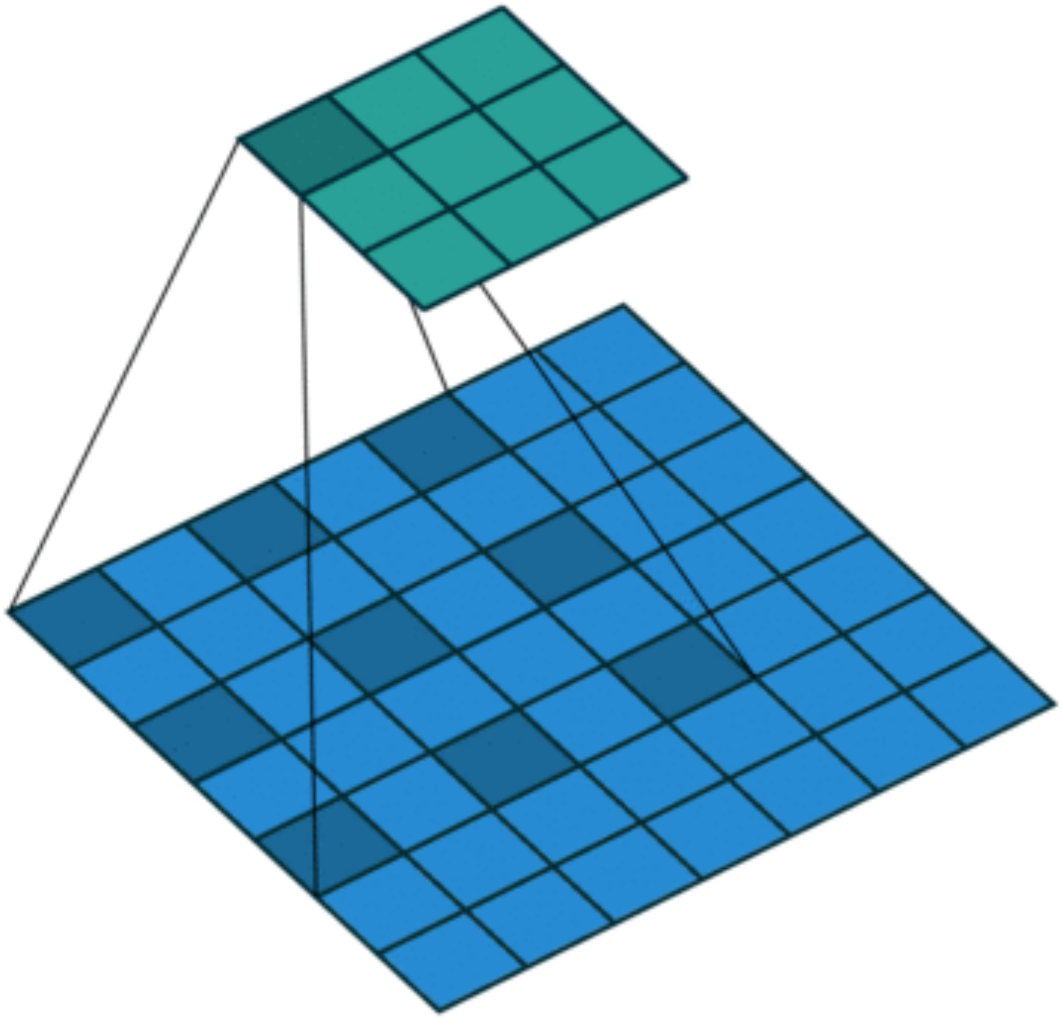
Fleuret, [Deep Learning Course]

Padding – Size of Zero Frame Around Input



Fleuret, [Deep Learning Course]

Dilation



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

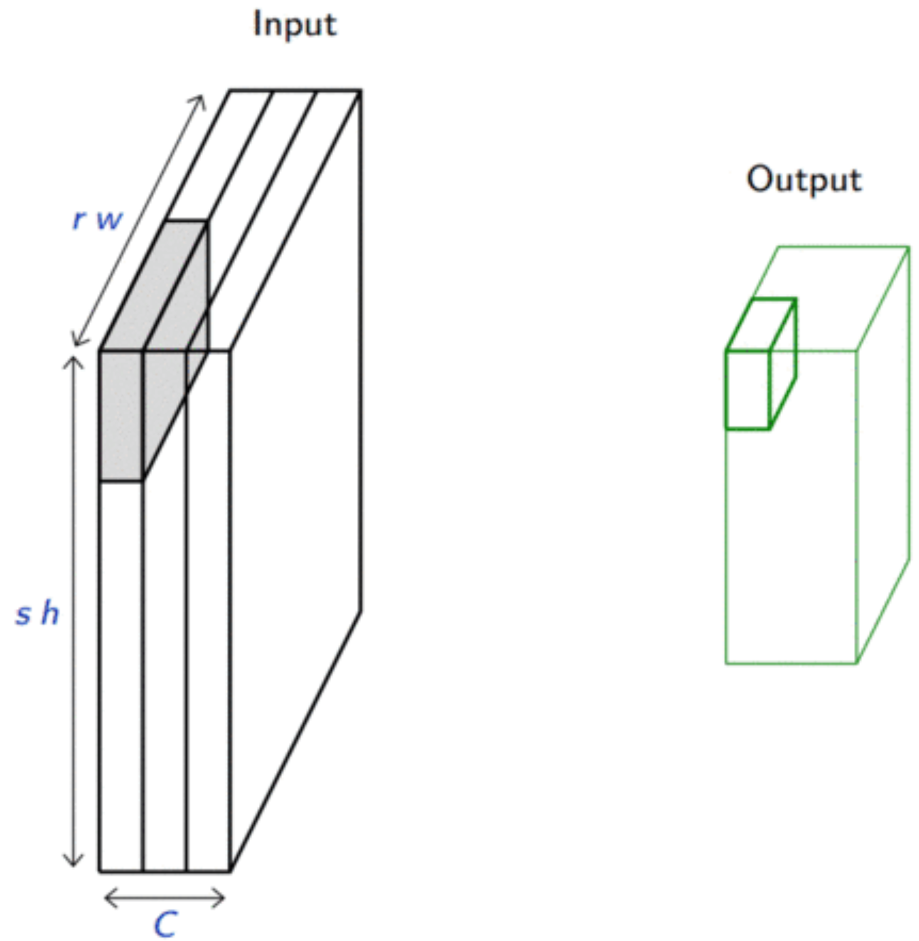
Shared Weights: Economic and Equivariant

- 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 $h \times w$



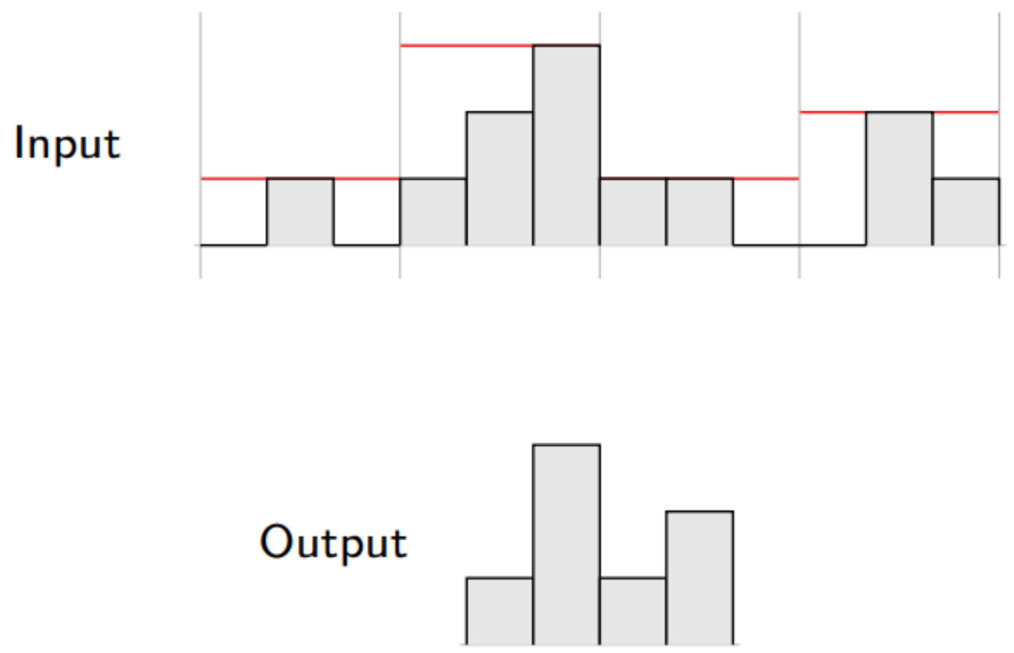
Fleuret, Deep Learning Course

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

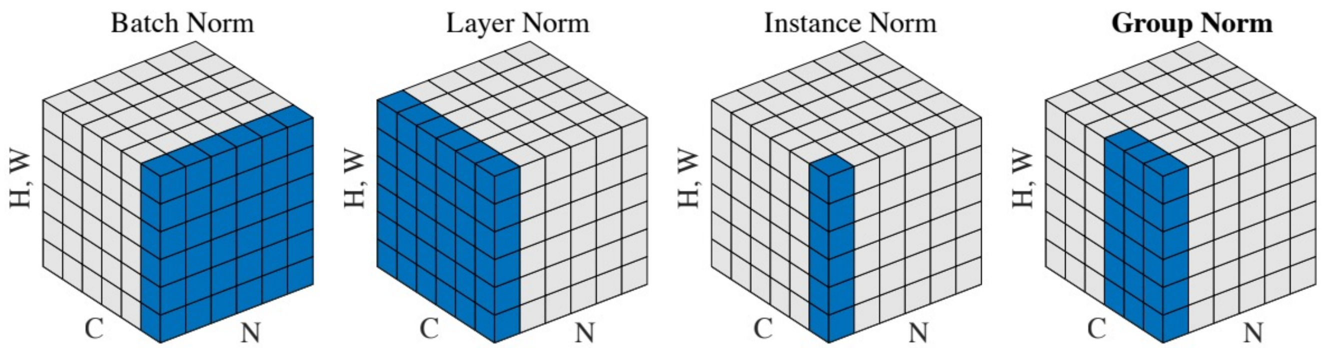


Normalization

- 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

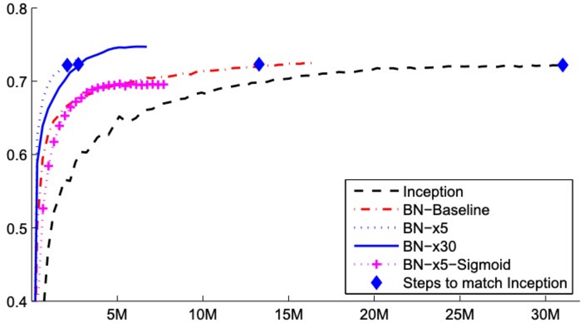
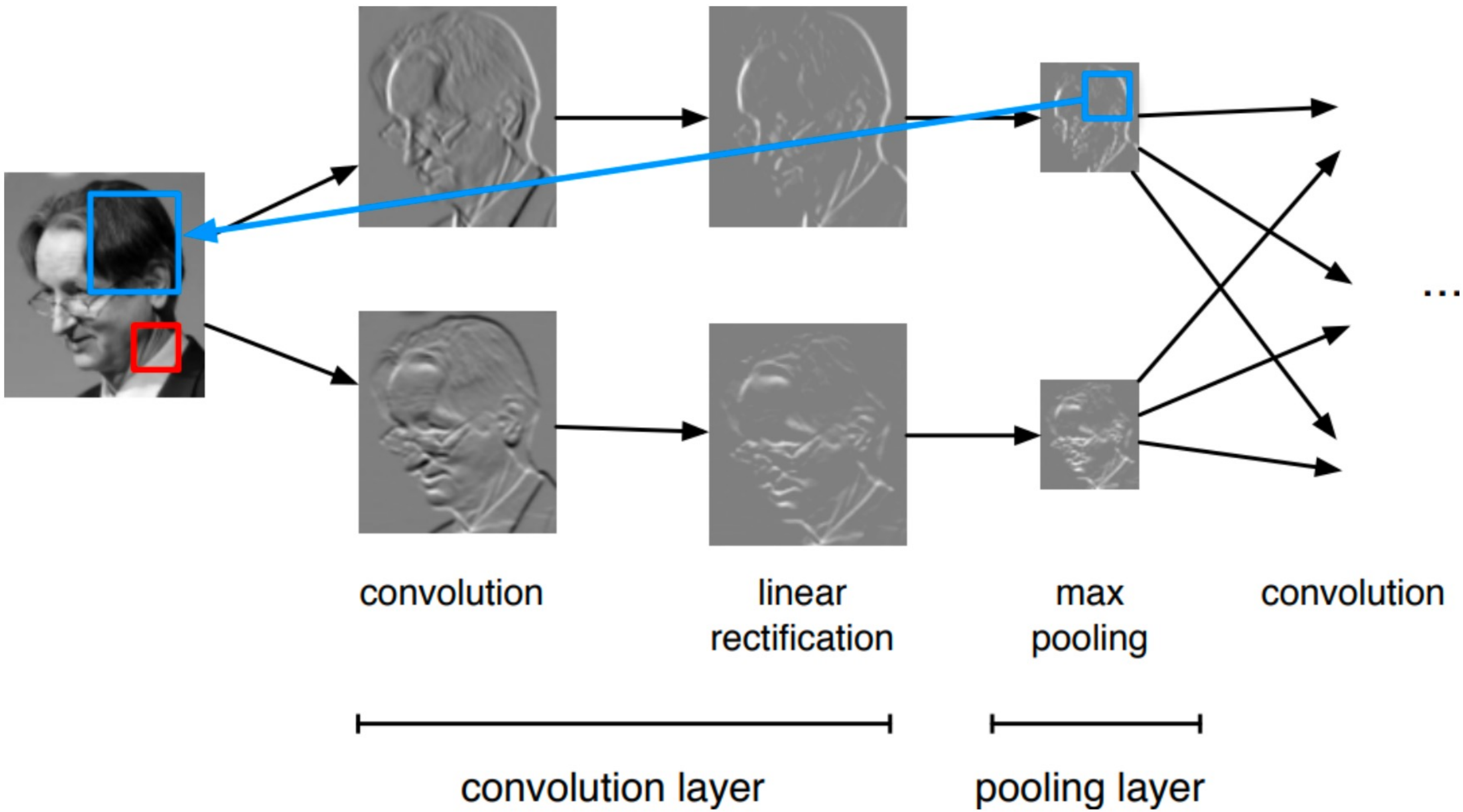


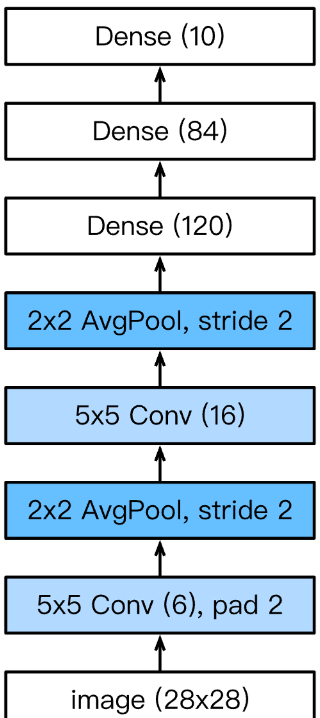
Figure 2: *Single crop validation accuracy of Inception and its batch-normalized variants, vs. the number of training steps.*

Convolutional Network

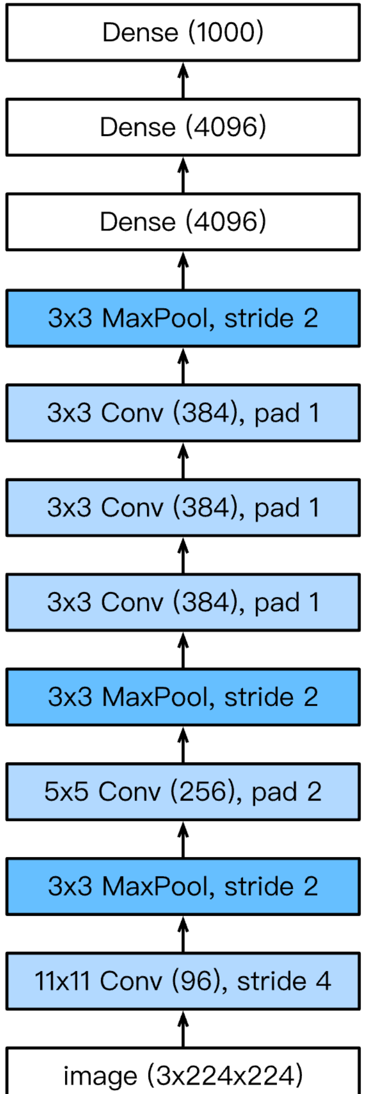
- A combination of convolution, pooling, ReLU, and fully connected layers



Convolutional Networks



LeNet
(LeCun et al, 1998)

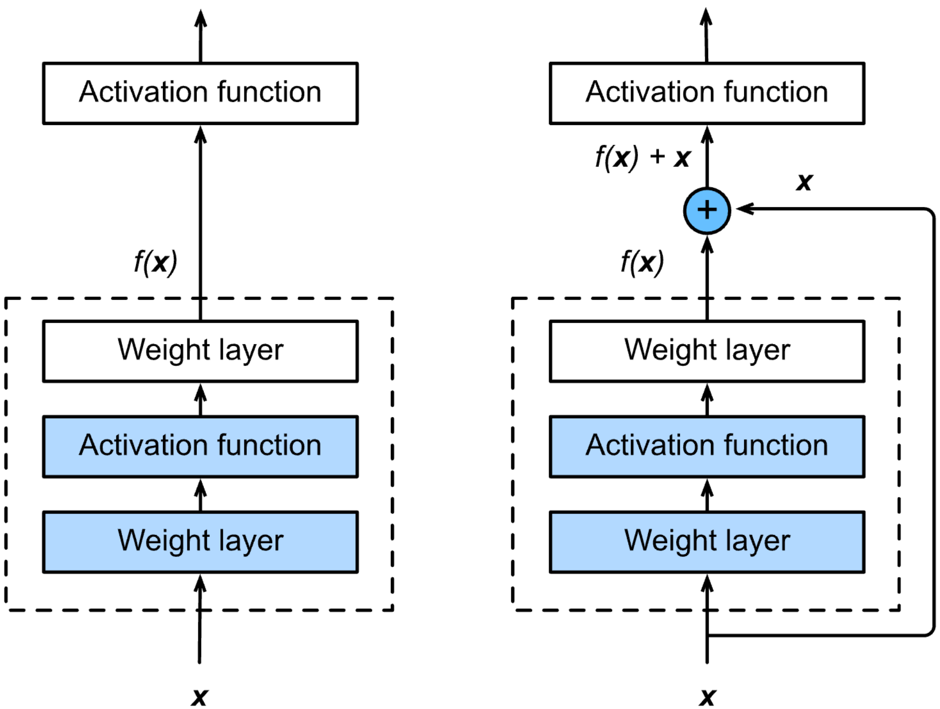


AlexNet
(Krizhevsky et al, 2012)

ImageNet Classification

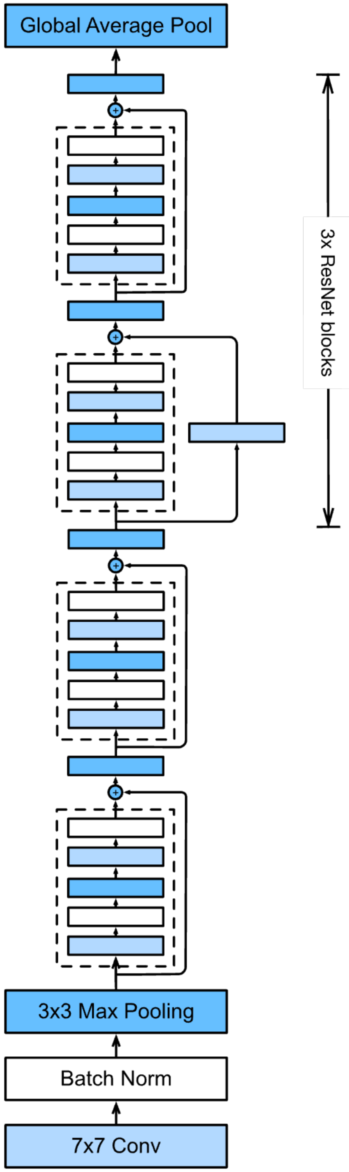
Residual Connections

- 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.



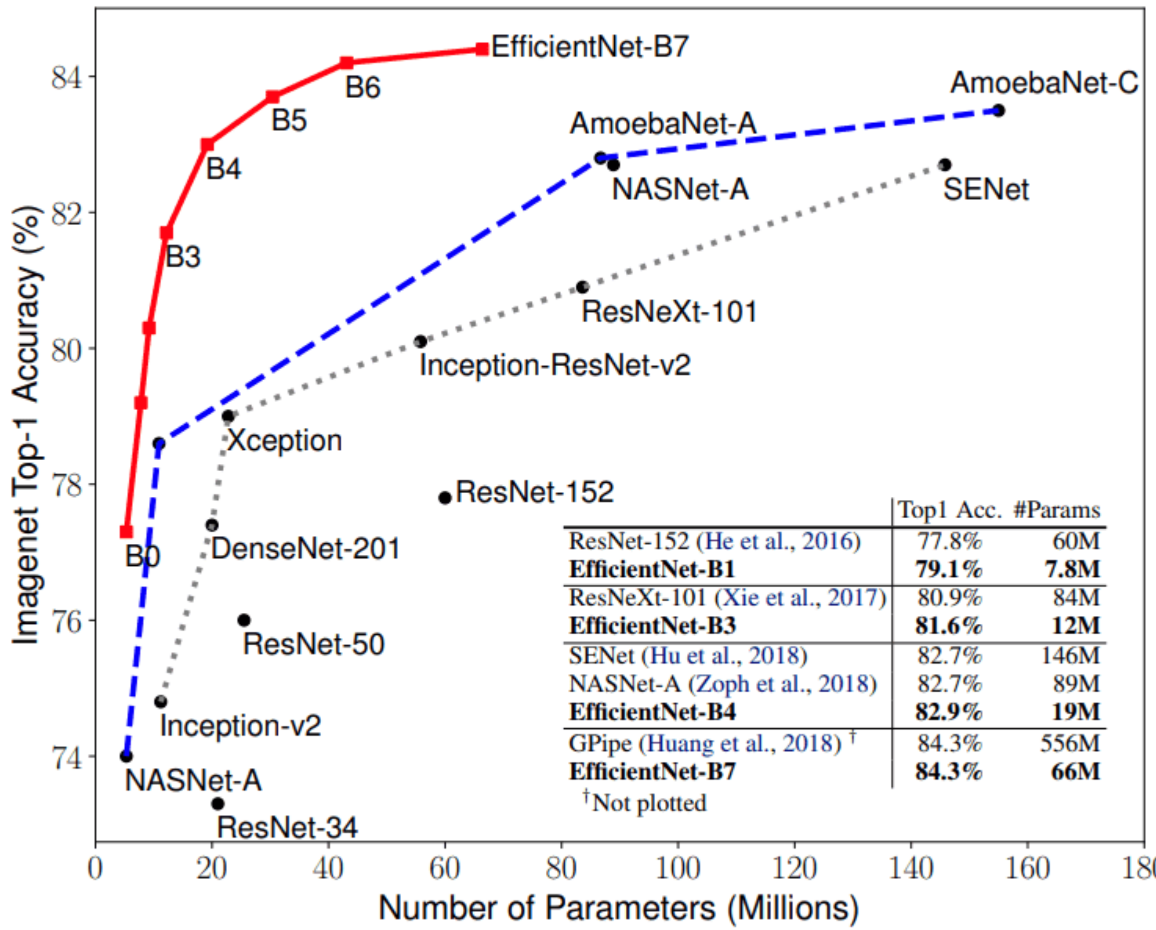
Credits: Deep Dive in Deep Learning , and G. Louppe

Deep CNNs



ResNet
(He et al, 2015)

[1905.11946](1905.11946)



	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**79.1%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.6%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.9%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.3%**	**66M**

† Not plotted

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

Credit: F. Fleuret

Recurrent States

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

- Recurrent model maintains **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:

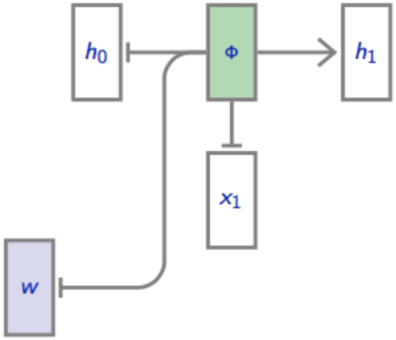
$$\phi(\boldsymbol{x}_t, \boldsymbol{h}_t; W, U) = \sigma(W\boldsymbol{x}_t + U\boldsymbol{h}_t)$$

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

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

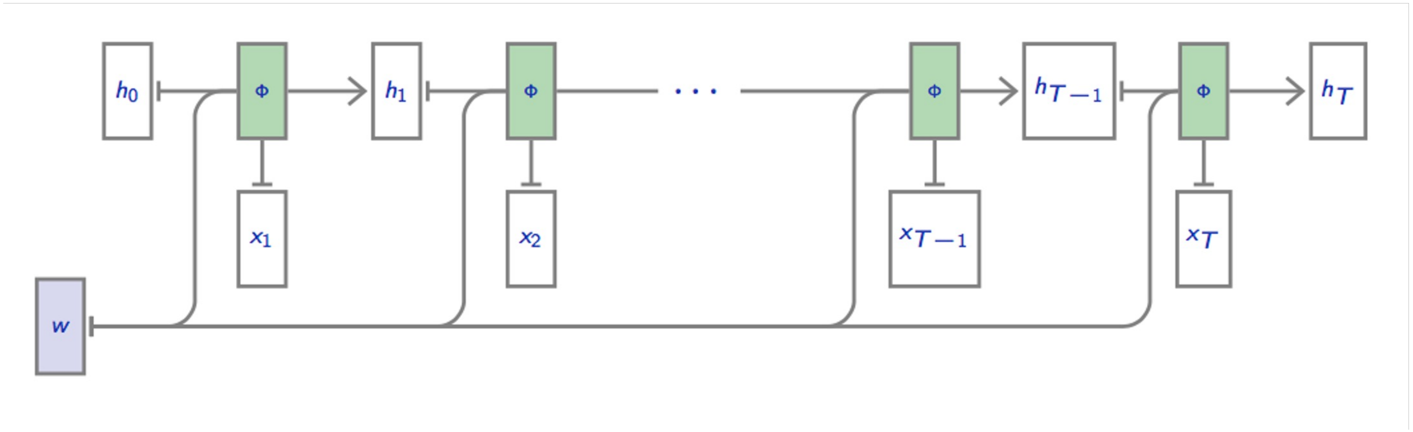
Credit: F. Fleuret

Recurrent Neural Networks



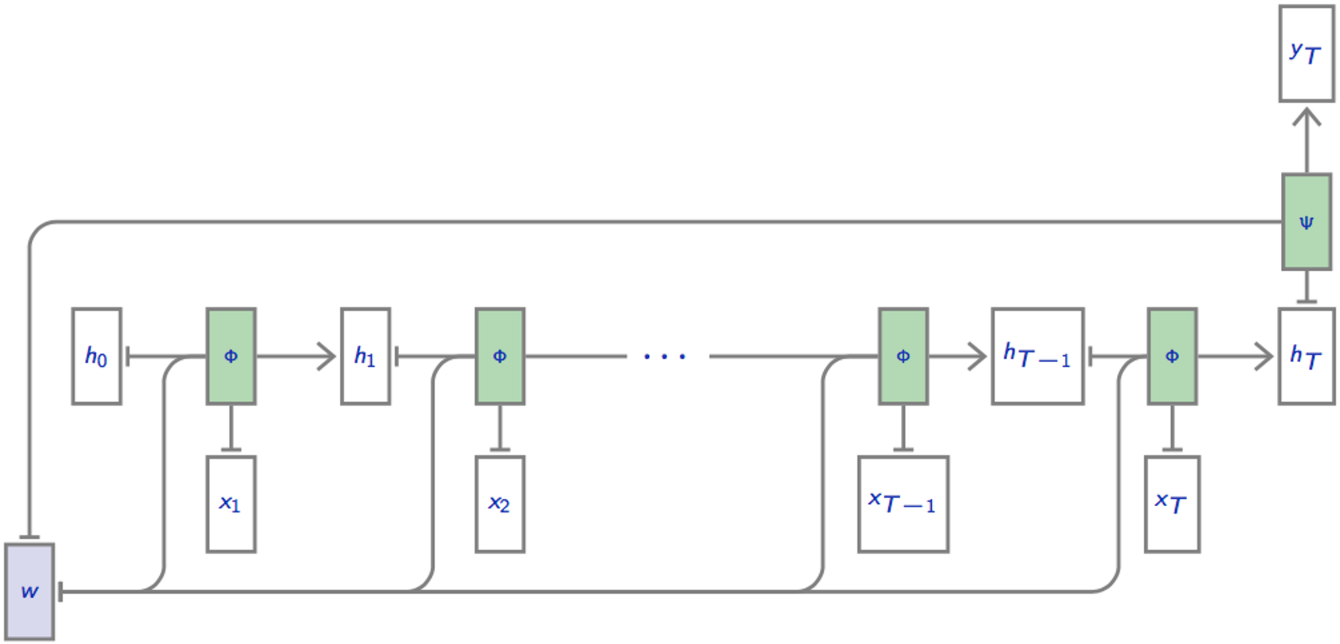
Credit: [F. Fleuret]

Recurrent Neural Networks



Credit: F. Fleuret

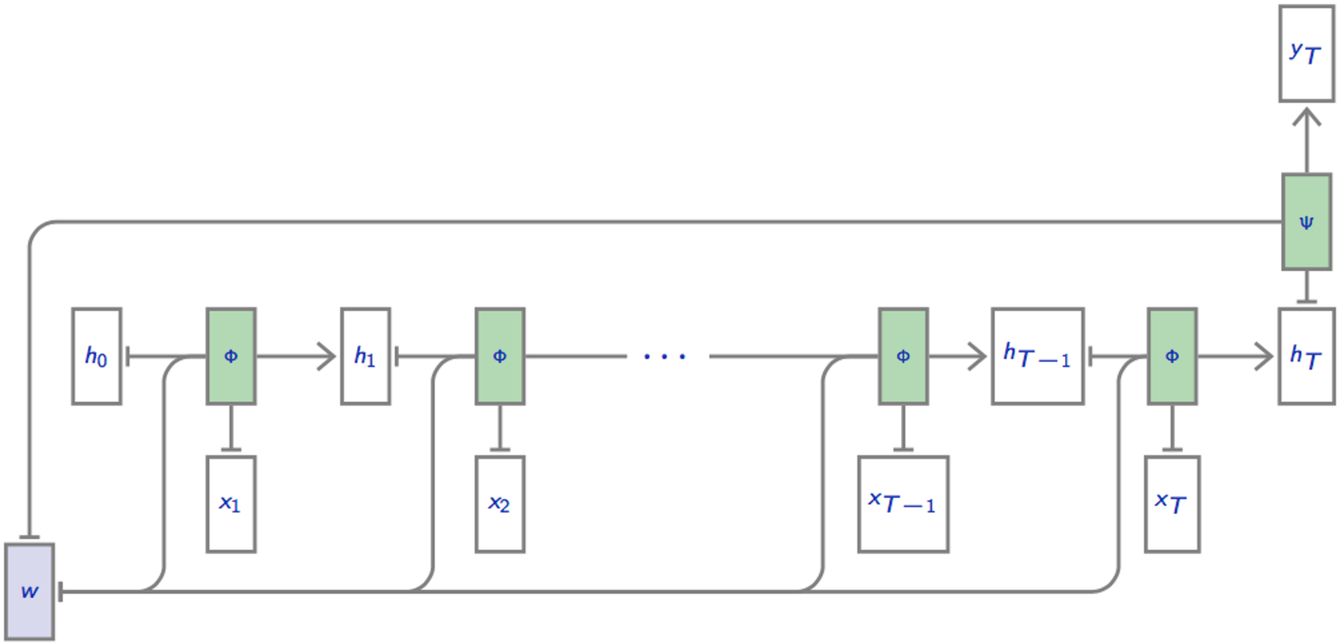
Recurrent Neural Networks



Credit: [F. Fleuret]

Recurrent Neural Networks

[0.98] → Positive Sentiment



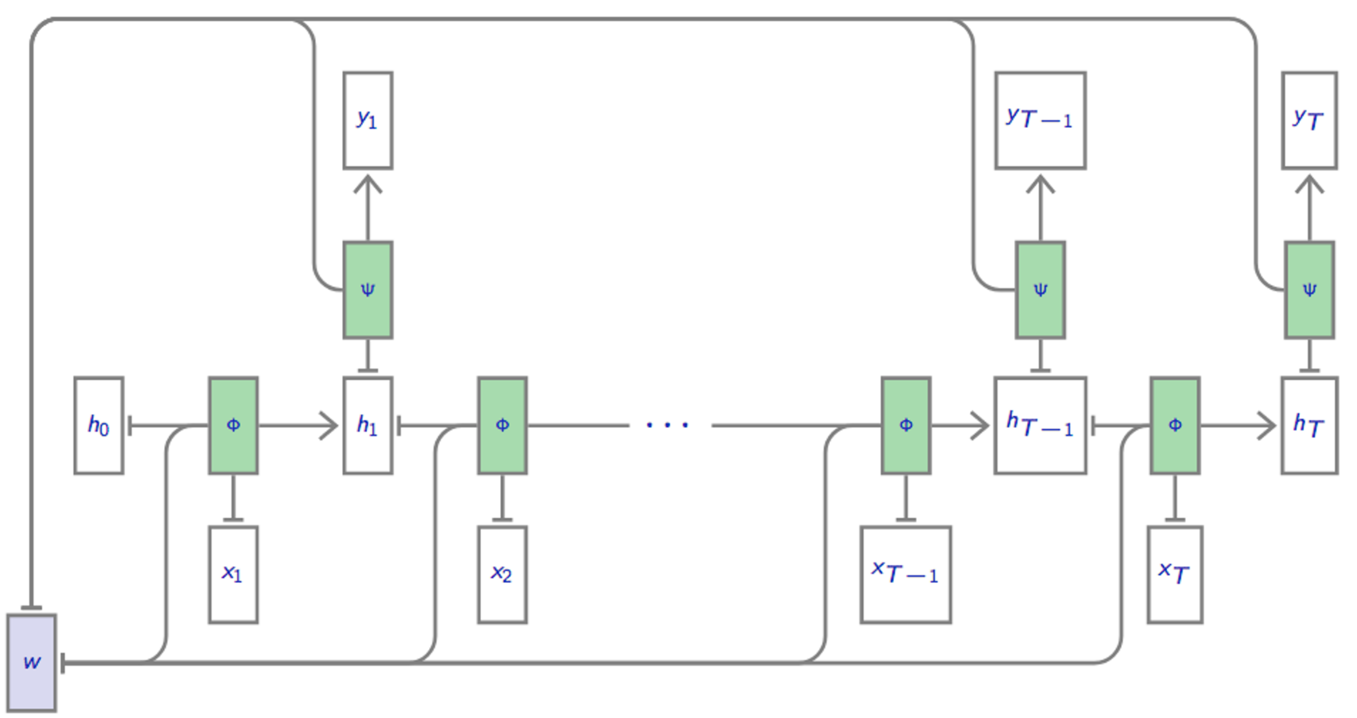
The movie was great

Sentiment Analysis

Credit: F. Fleuret

Recurrent Neural Networks

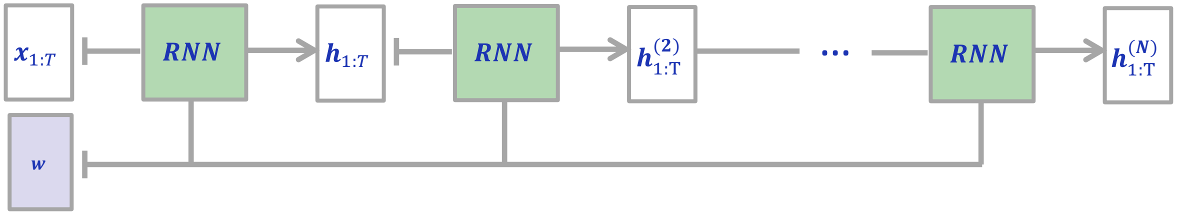
Prediction per sequence element



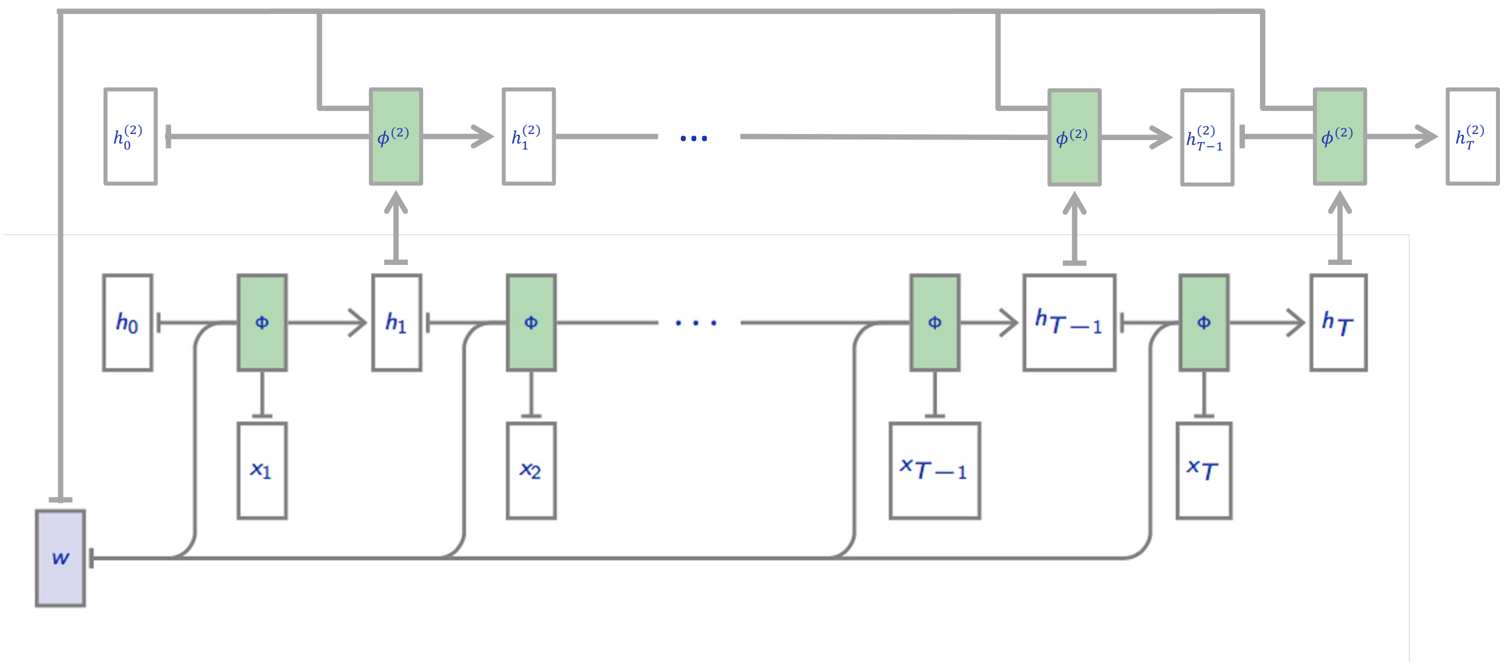
Although the number of steps $T(x)$ depends on x, this is a standard computational graph and automatic differentiation can deal with it as usual. This is known as "backpropagation through time" (Werbos, 1988)

Credit: F. Fleuret

Stacked RNN



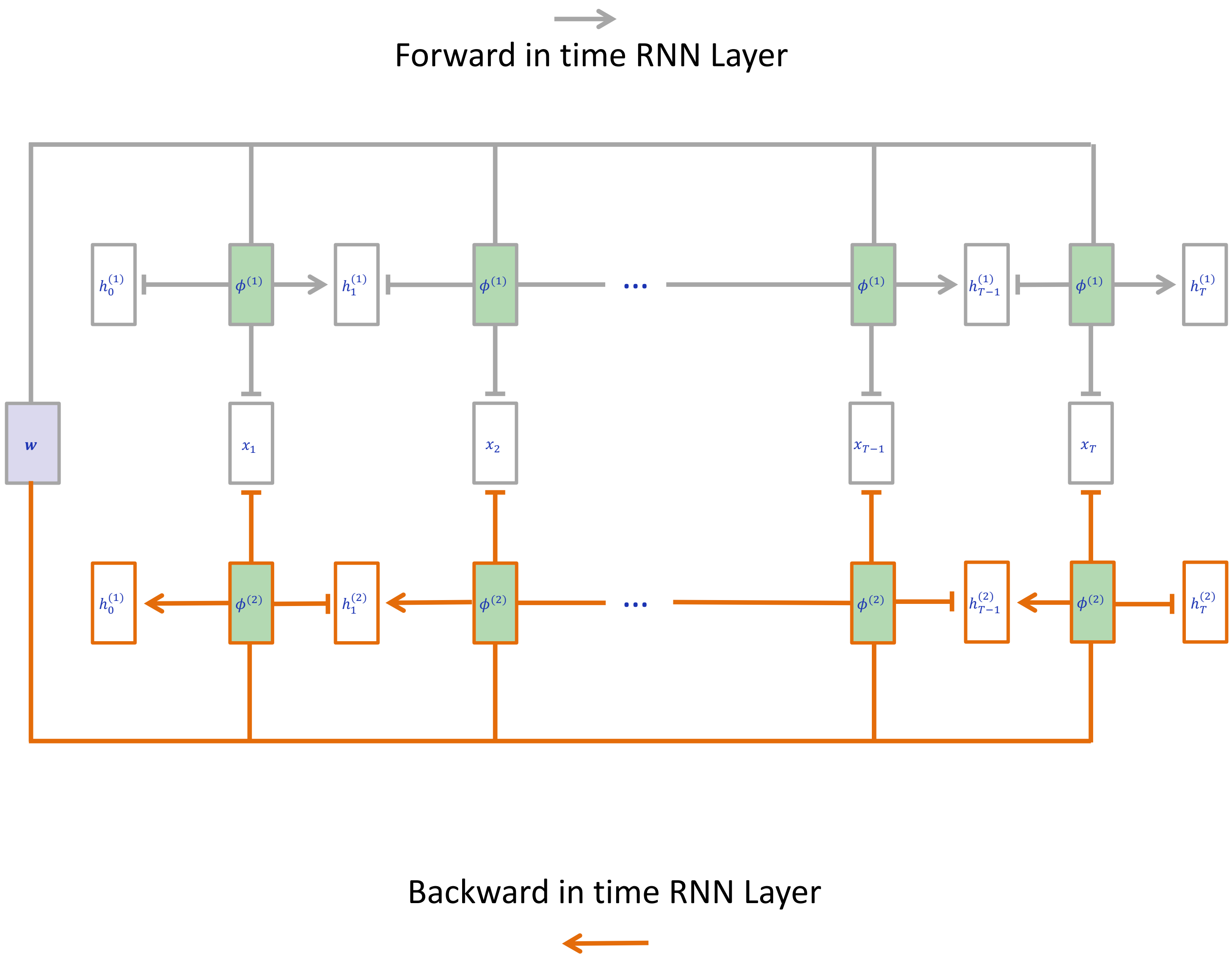
Stacked RNN



Two Stacked LSTM Layers

Bi-Directional RNN

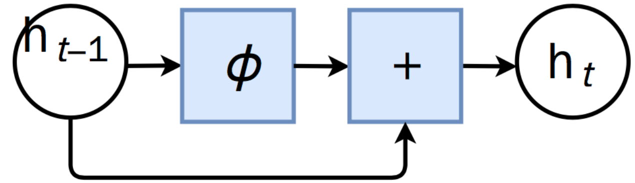
Forward in time RNN Layer



Backward in time RNN Layer

Gating

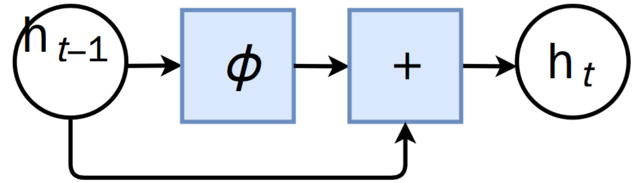
- Gating:
  - network can grow very deep, in time → vanishing gradients.
  - *Critical component*: add pass-through (additive paths) so recurrent state does not go repeatedly through squashing non-linearity.



Credit: Gilles Louppe

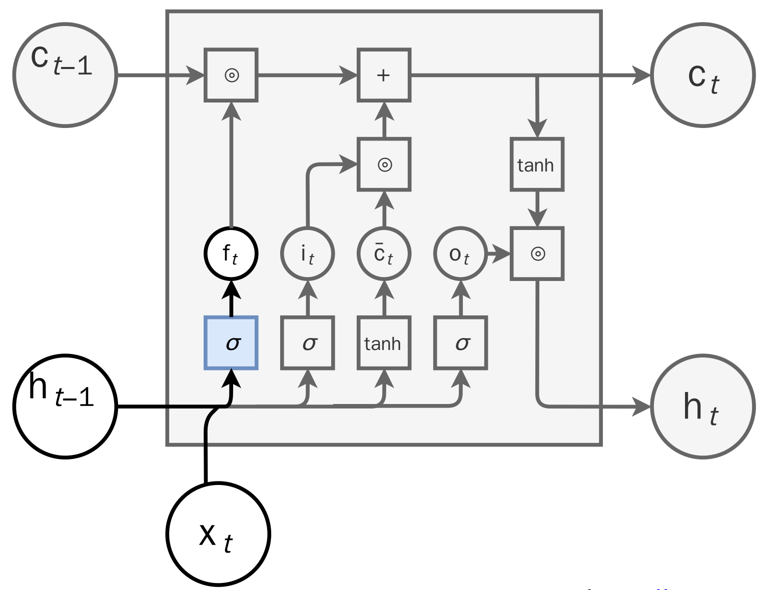
Long Short Term Memory (LSTM)

- 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

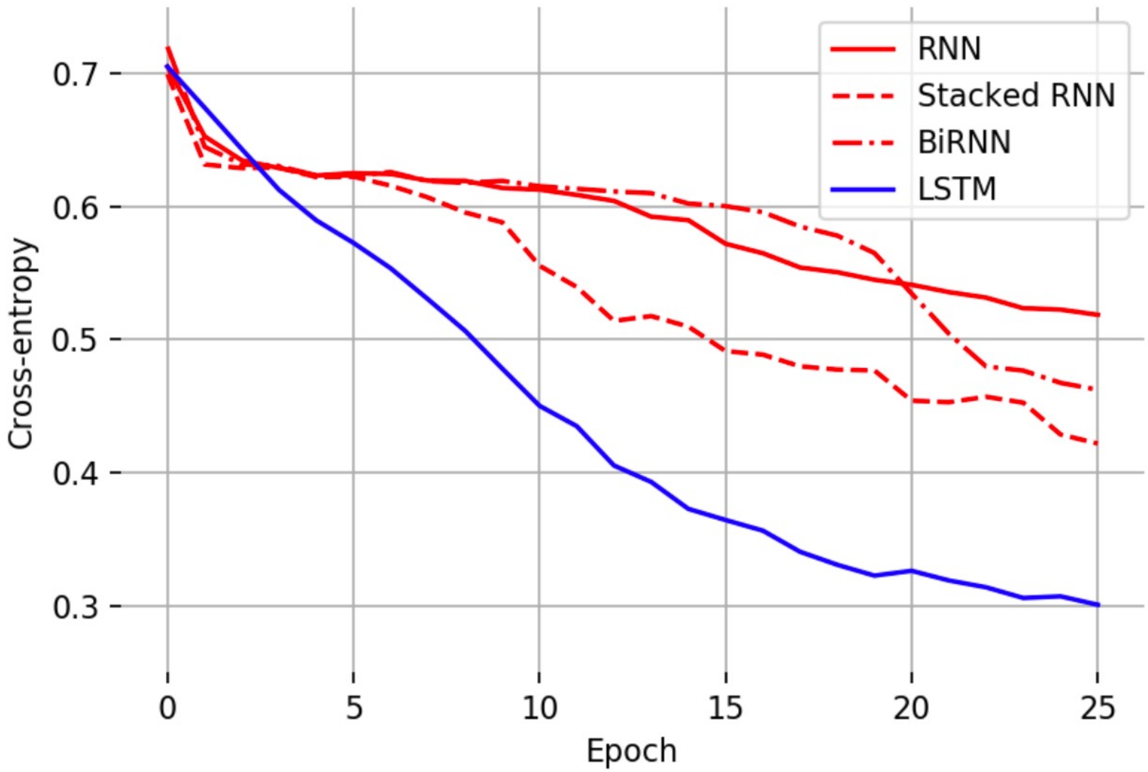


Credit: Gilles Louppe

Comparison on Toy Problem

Learn to recognize palindrome
Sequence size between 1 to 10

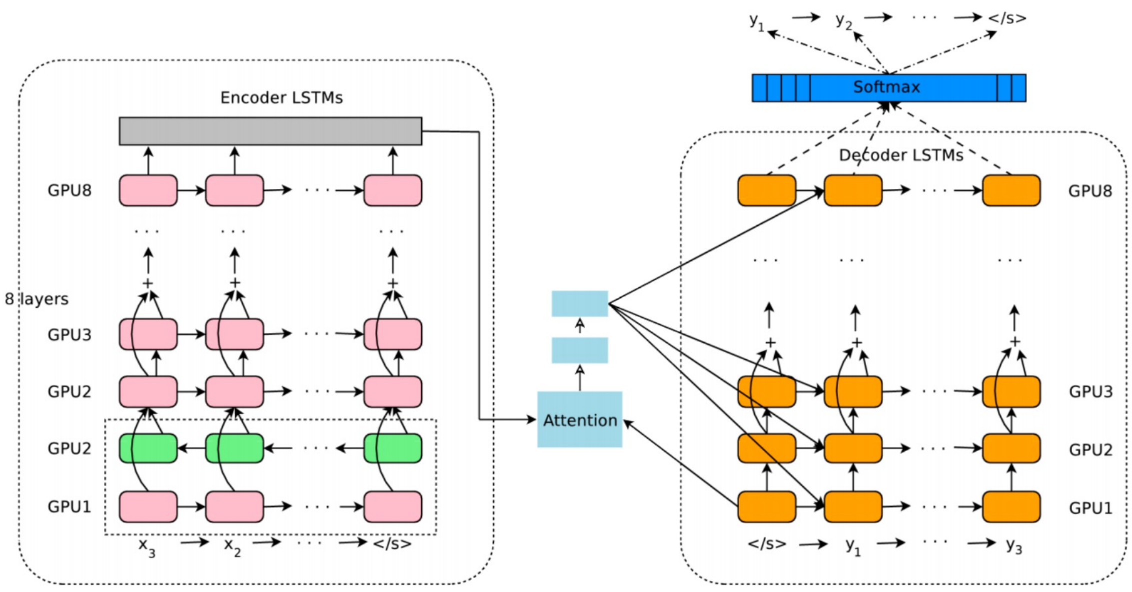
$\mathbf{x}$	y
$(1, 2, 3, 2, 1)$	1
$(2, 1, 2)$	1
$(3, 4, 1, 2)$	0
(0)	1
$(1, 4)$	0

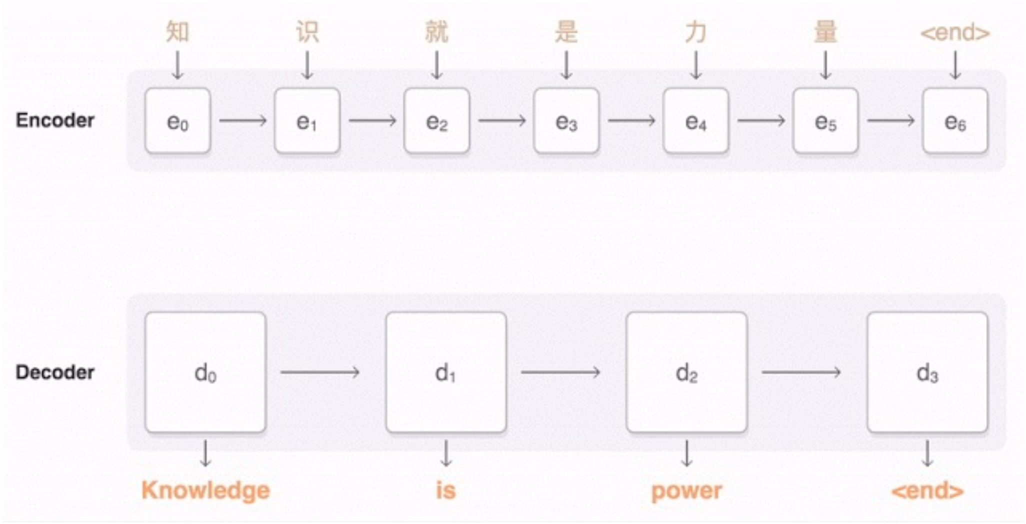


Slide credit: G. Louppe

Examples

Neural machine translation

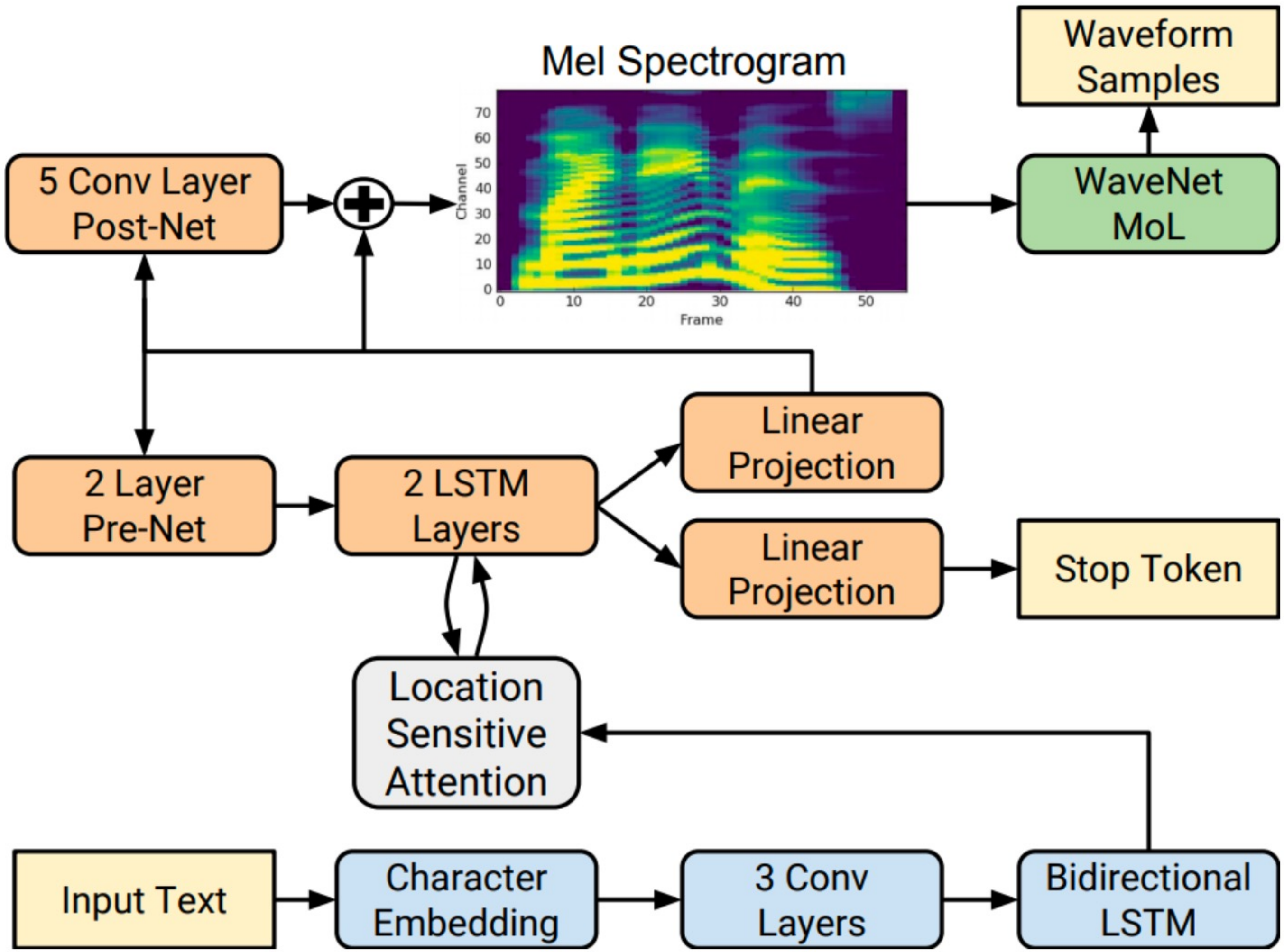




Y. Wu et al, 2016

Examples

Text-to-speech synthesis

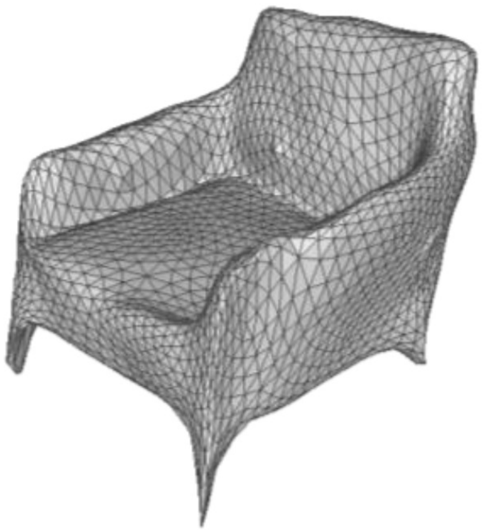


Shen et al., 2017

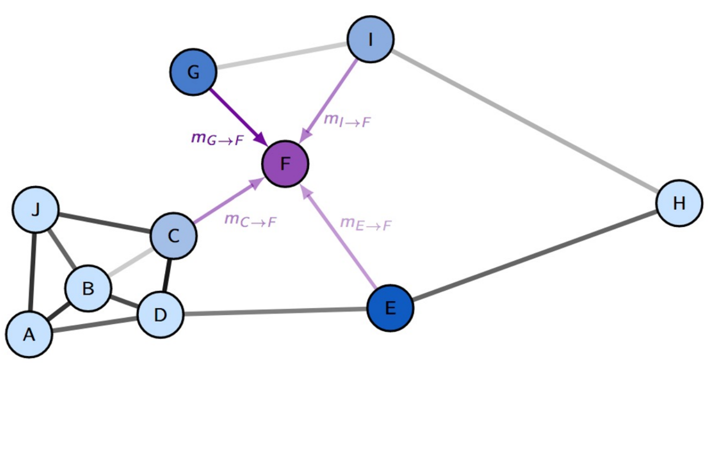
Many Other Architecture Choices

Graph Neural Networks

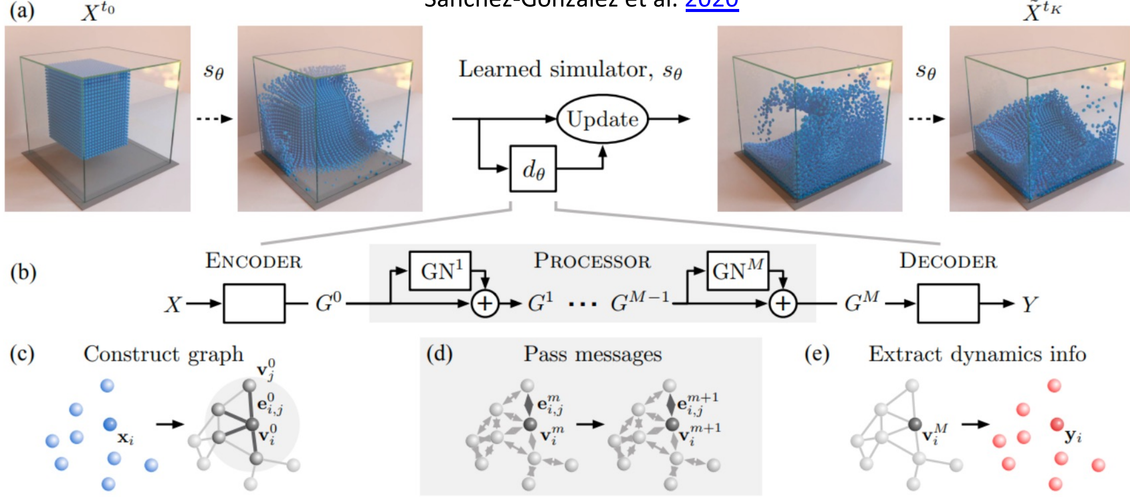
- Permutation invariant data with geometric relationships
  – Features can be local on graph, but meaningful anywhere on graph



- Graph layers can encode these relationships on nodes & edges



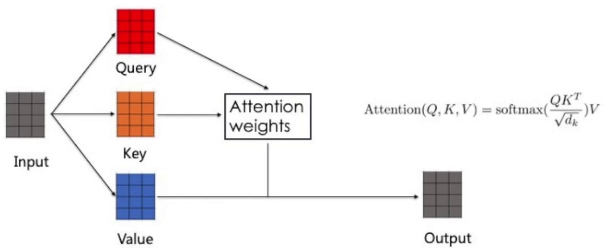
Sanchez-Gonzalez et al. 2020

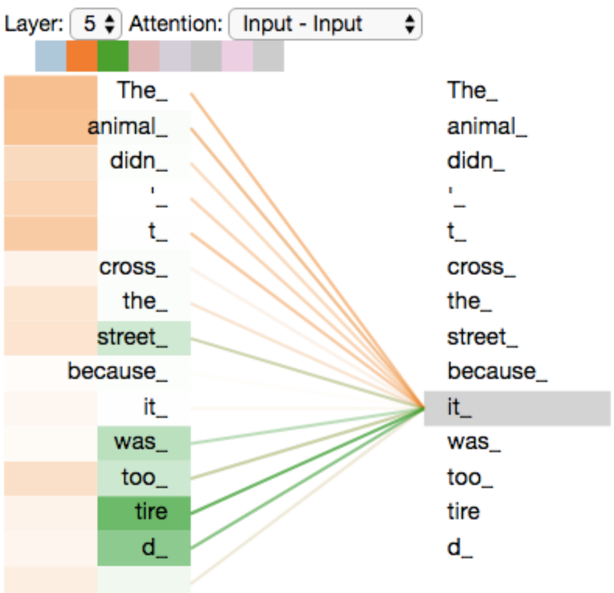


Transformers & Deep Sets

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

- *Transformers are very adaptable*:
  Built using layers of ***attention***,
  Excellent at process sequences,
  but also 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

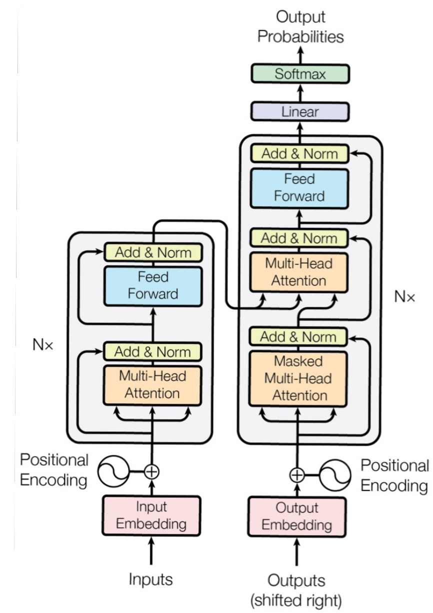
Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

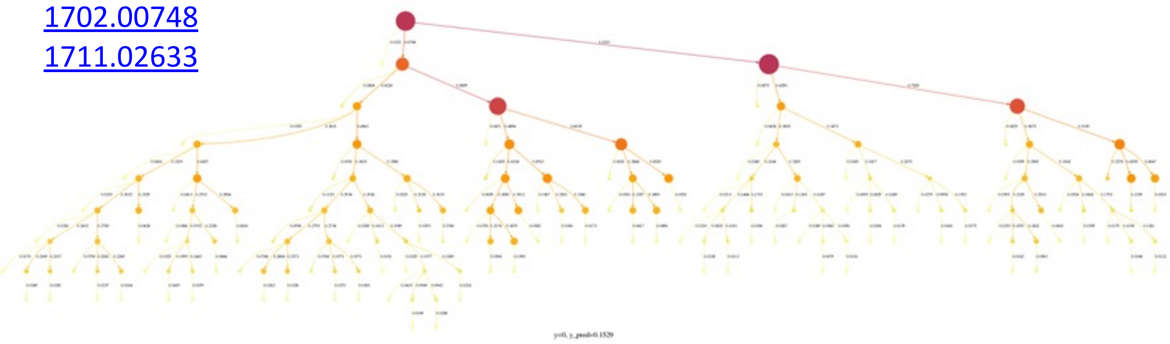
Illia Polosukhin* ‡ illia.polosukhin@gmail.com



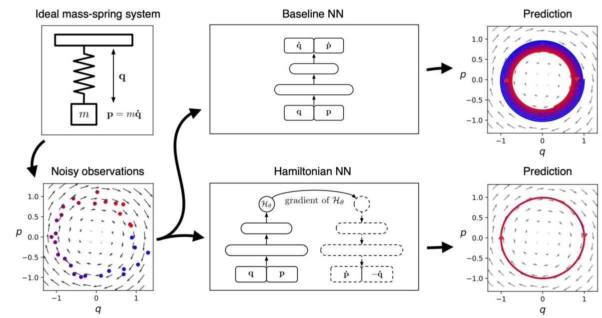
Physics Inspired Models

QCD Structured Neural Nets

[1702.00748](1702.00748)
[1711.02633](1711.02633)

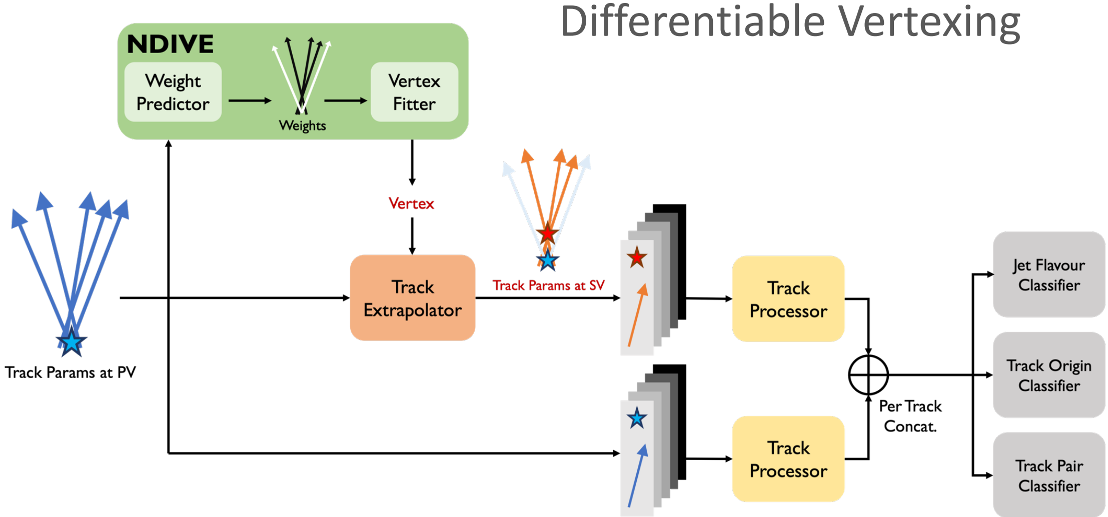


Hamiltonian Neural Nets



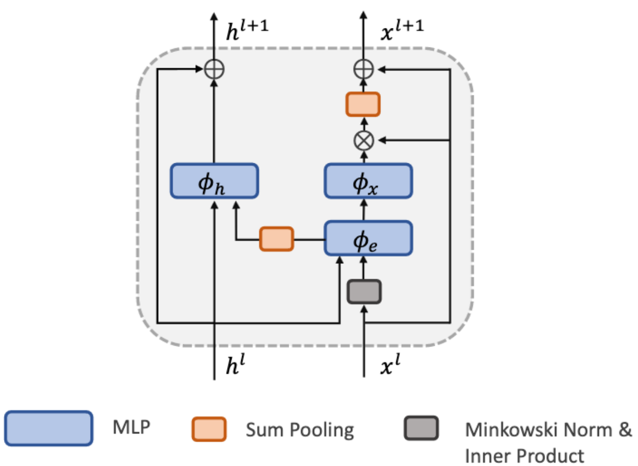
1906.01563

Differentiable Vertexing



Smith, Ochoa, Inacio, Shoemaker, **MK**, 2310.12804

Lorentz Equivariance



Lorentz Group Equivariant Block (LGEB)

2201.08187

Summary

- Deep neural networks allow learning complex function by hierarchically structuring the feature learning

- We can use our inductive bias (knowledge) to define models that are well adapted to our problem

- Many neural networks structures are available for training models on a wide array of data types.

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

Modern Neural Networks

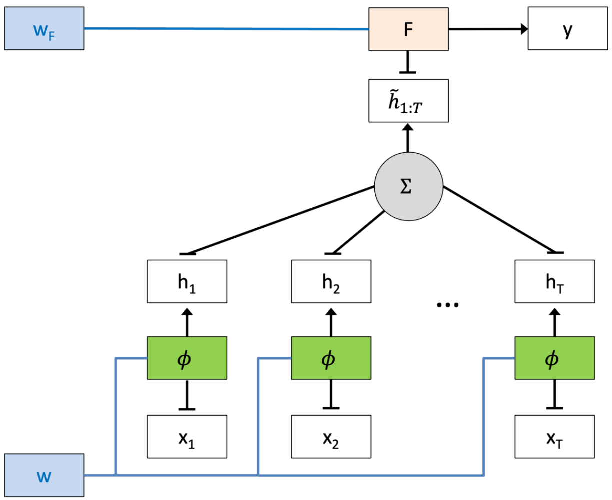
People are now building a **new kind of software** by assembling networks of **parameterized functional blocks** and by **training them from examples using some form of gradient-based optimization**.

- Yann LeCun, 2018

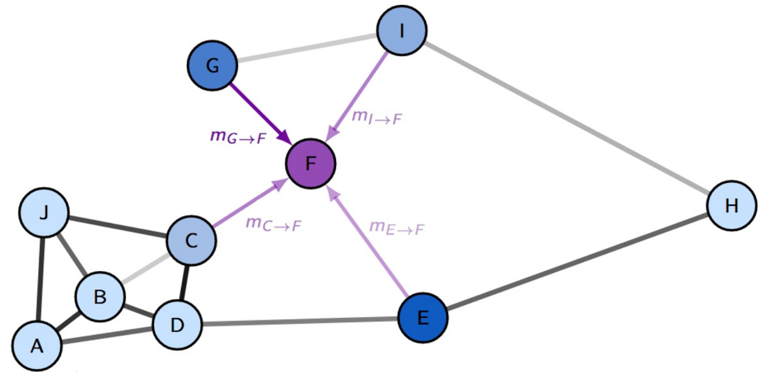
- Non-linear operations of data with parameters
- Layers (set of operations) designed to perform specific mathematical operations
- Chain together layers to perform desired computation
- Train system (with examples) for desired computation using gradient descent

Many Other Architecture Choices

Deep Sets



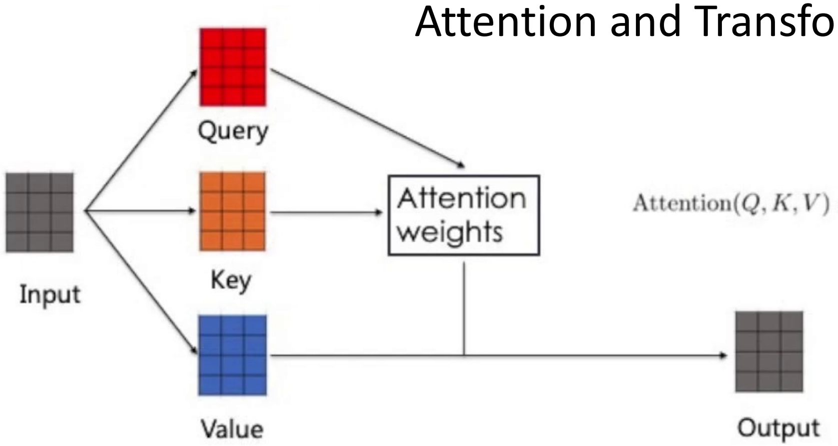
Graph Neural Networks



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

Image Credit: [I. Henrion]

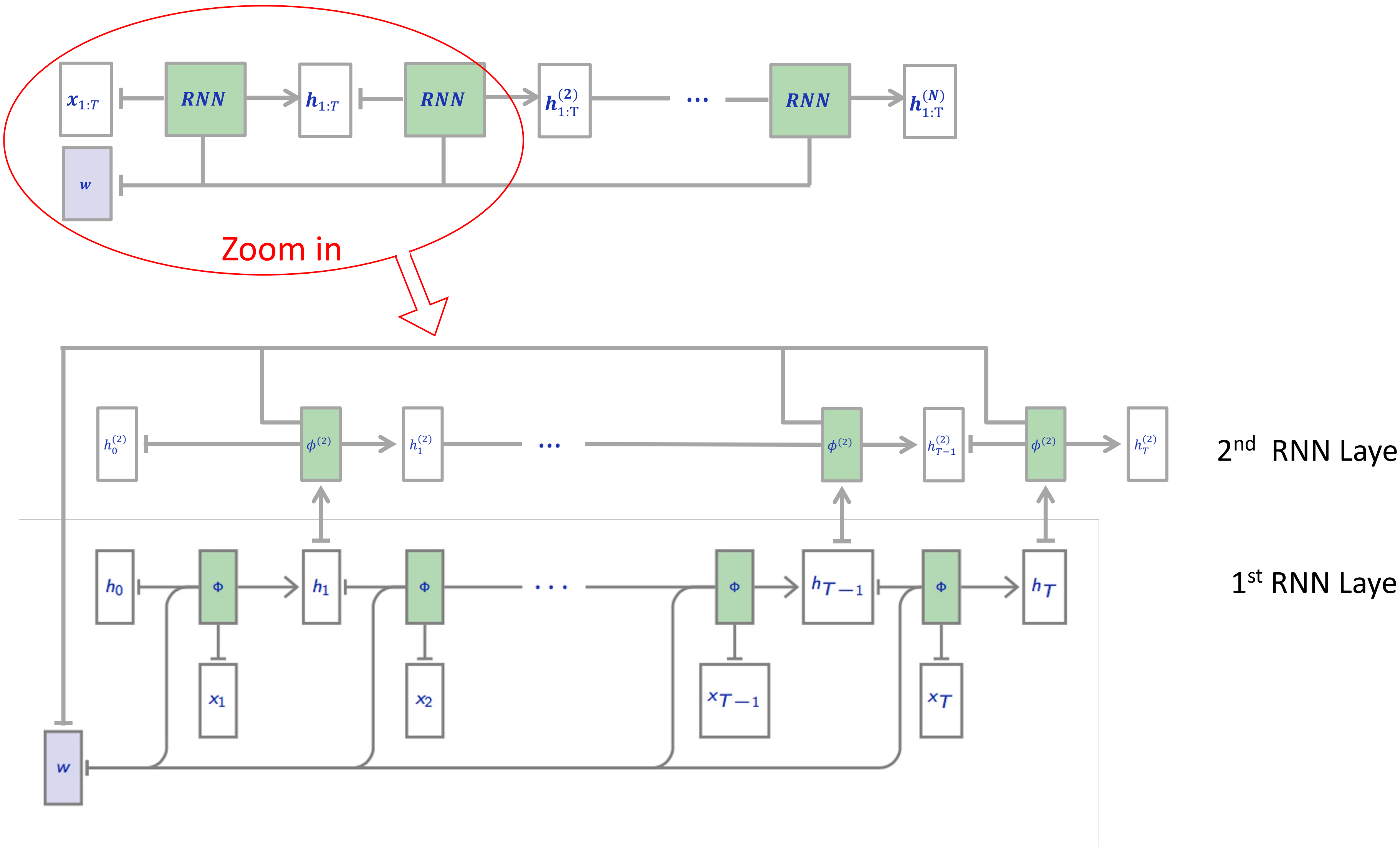
Attention and Transformers



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

+ More…

Stacked RNN



Two Stacked LSTM Layers

Deep Sets

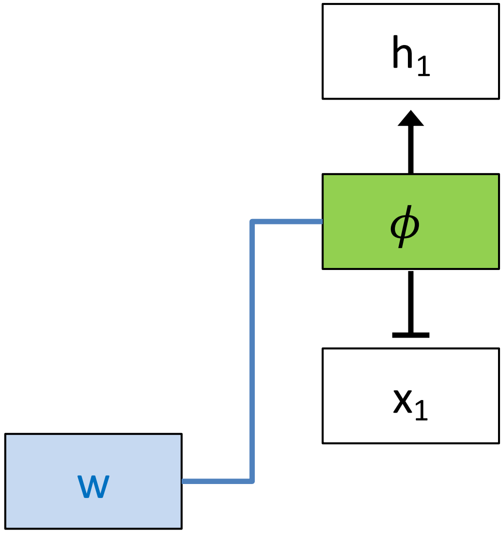
What if our data has no time structure?

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

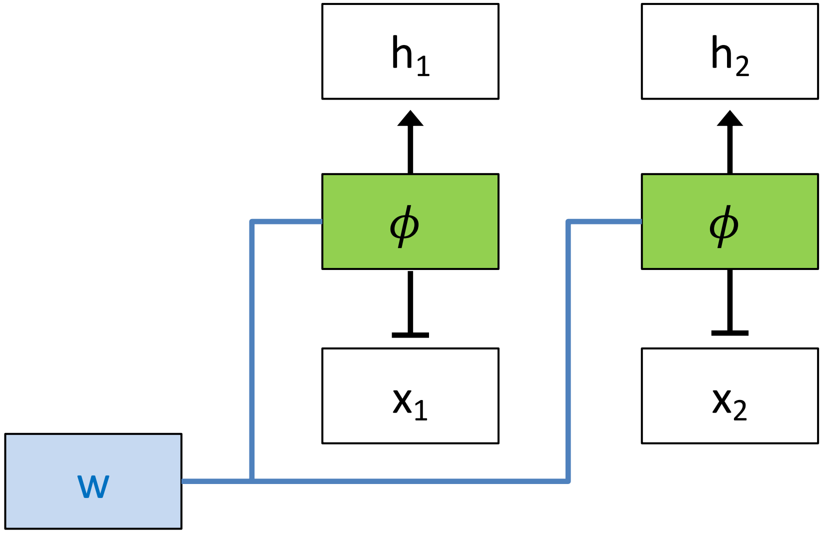
- *One option*: If we know about the data domain, could try to impose an ordering, then use RNN

- *Better option*: use system that can operate on variable length sets in permutation invariant way
  - Why permutation invariant → so order doesn't matter

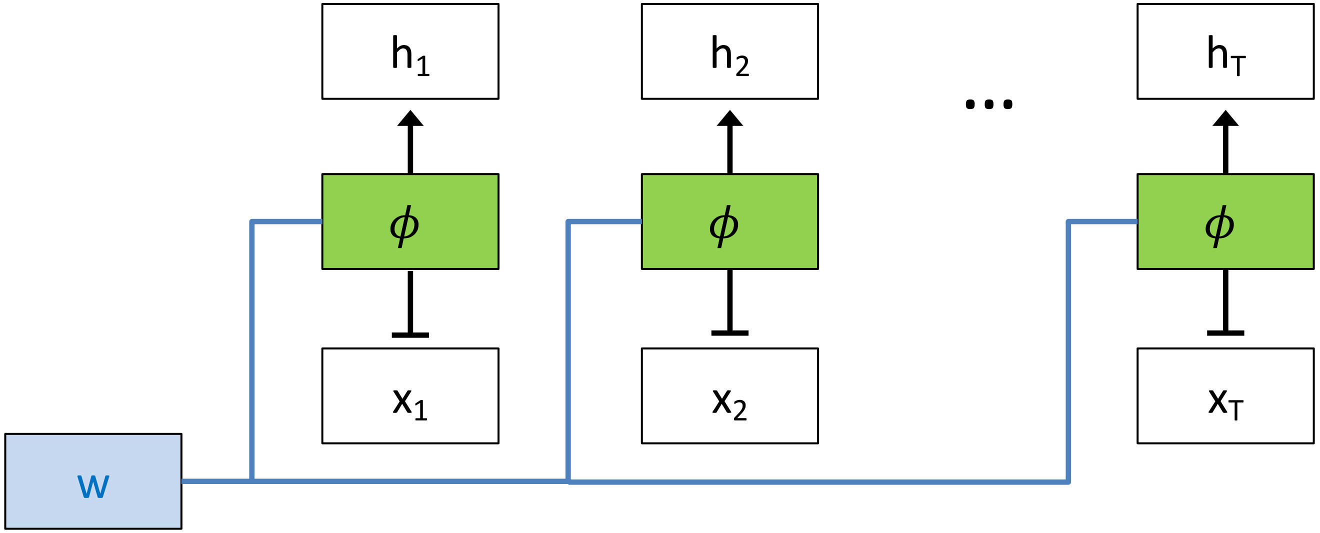
Deep Sets



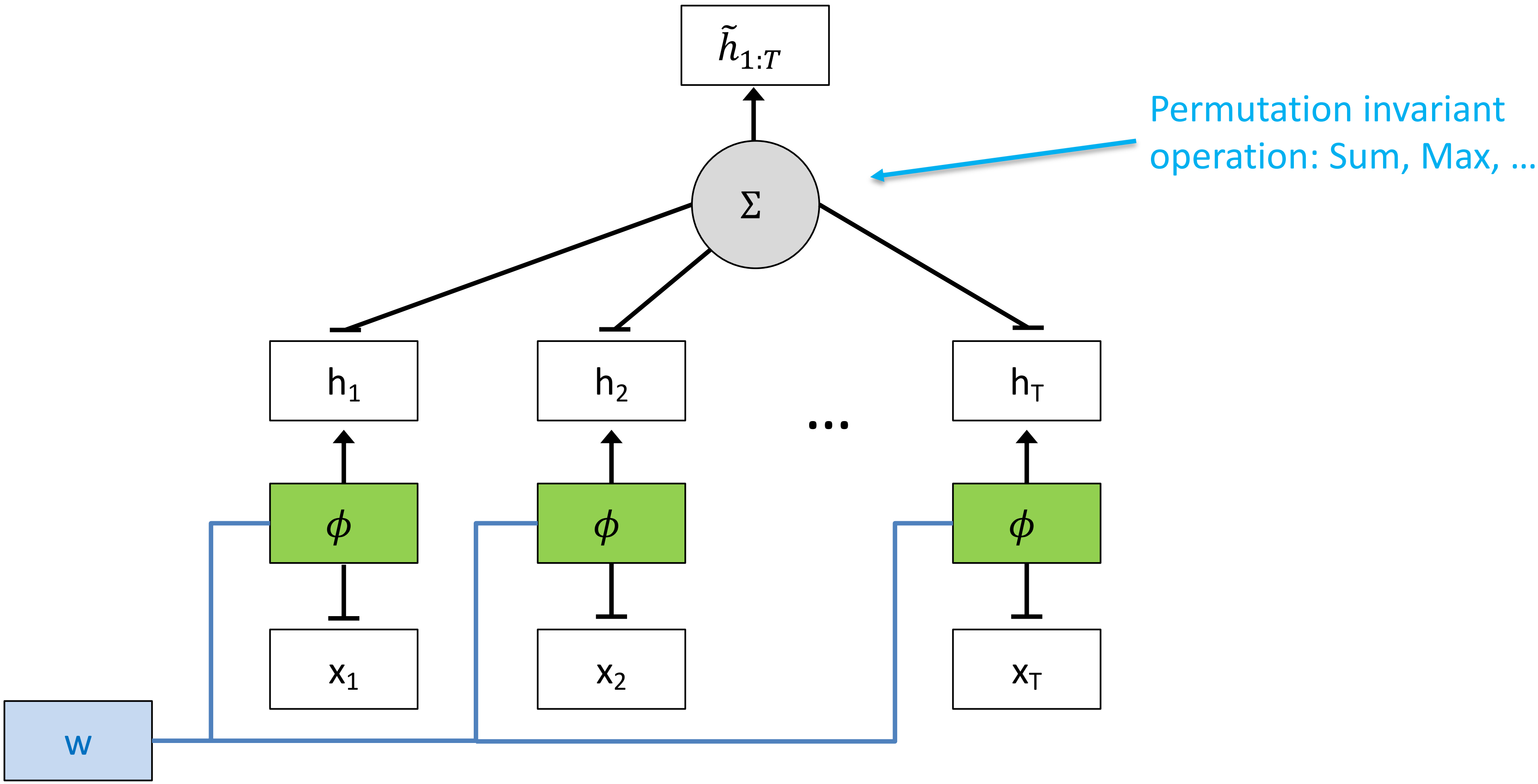
Deep Sets



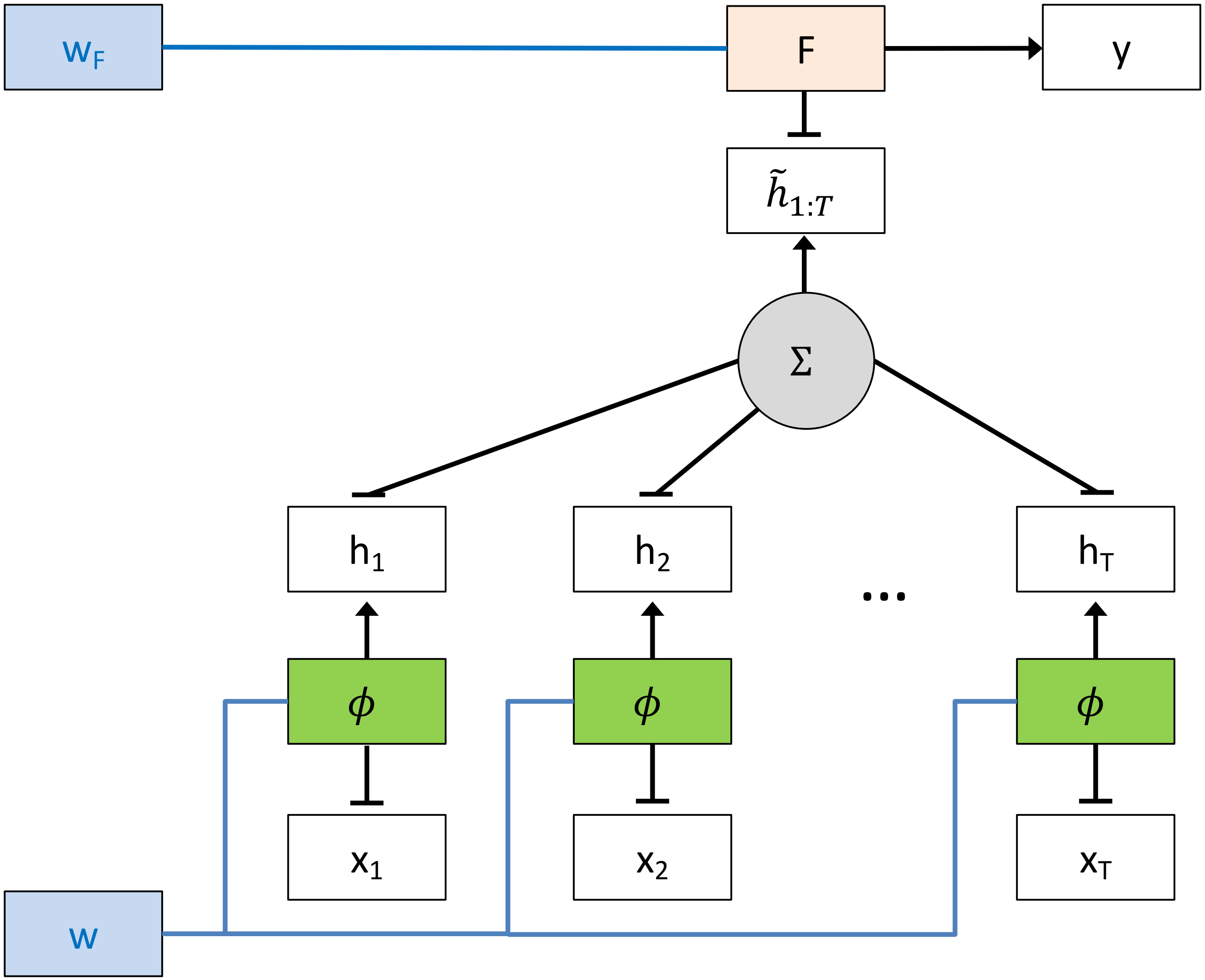
Deep Sets



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

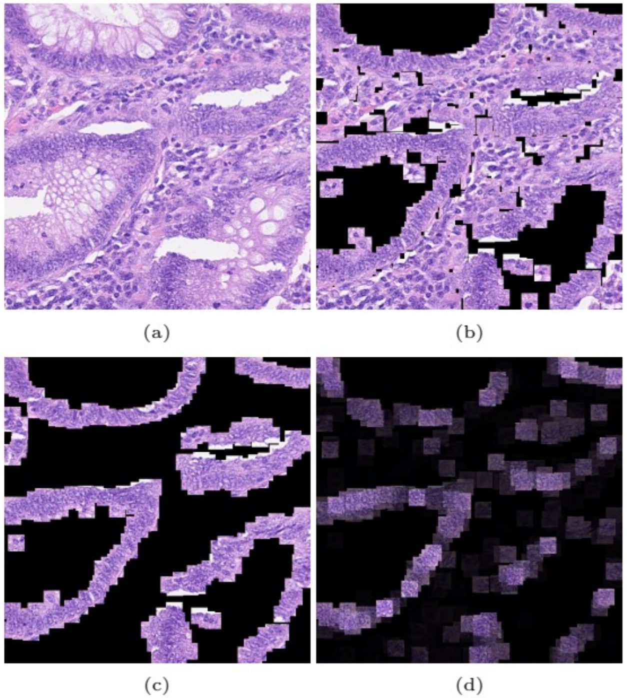
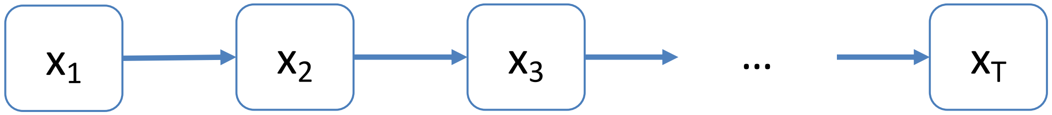


Figure 5. (a) H&E stained histology image. (b) 27×27 patches centered around all marked nuclei. (c) Ground truth: Patches that belong to the class epithelial. (d) Heatmap: Every patch from (b) multiplied by its corresponding attention weight, we rescaled the attention weights using $a'_k = (a_k - \min(\mathbf{a}))/(\max(\mathbf{a})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

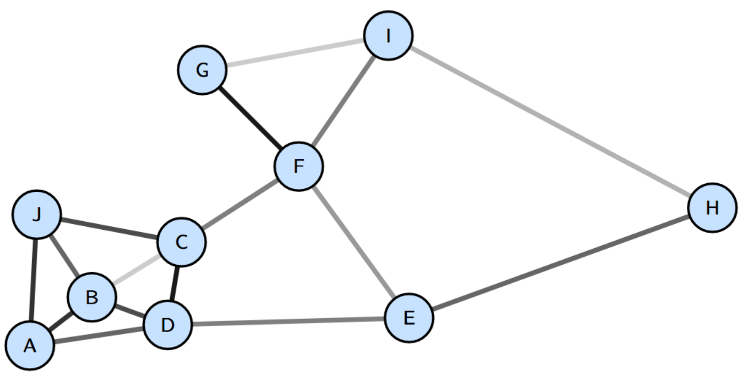
M. Ilse et al., 2018

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



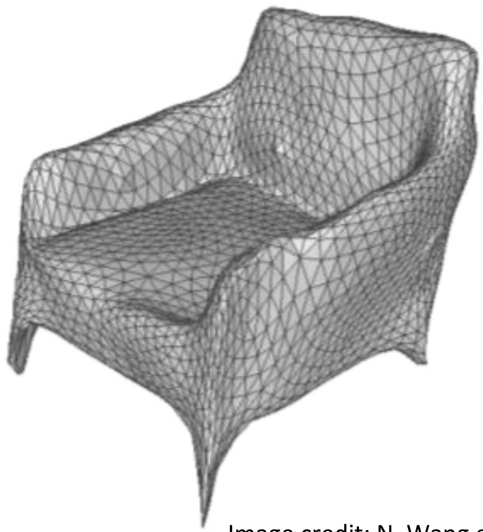
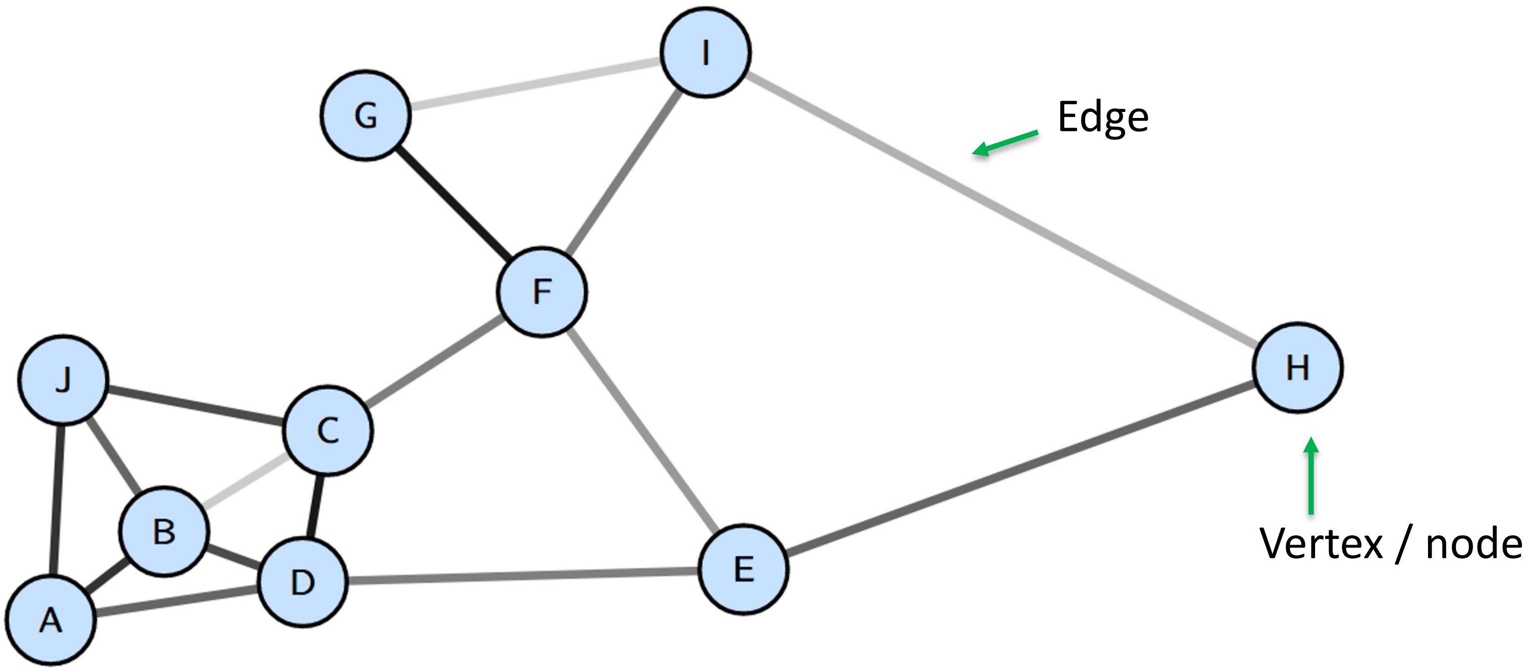


Image Credit: I. Henrion

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

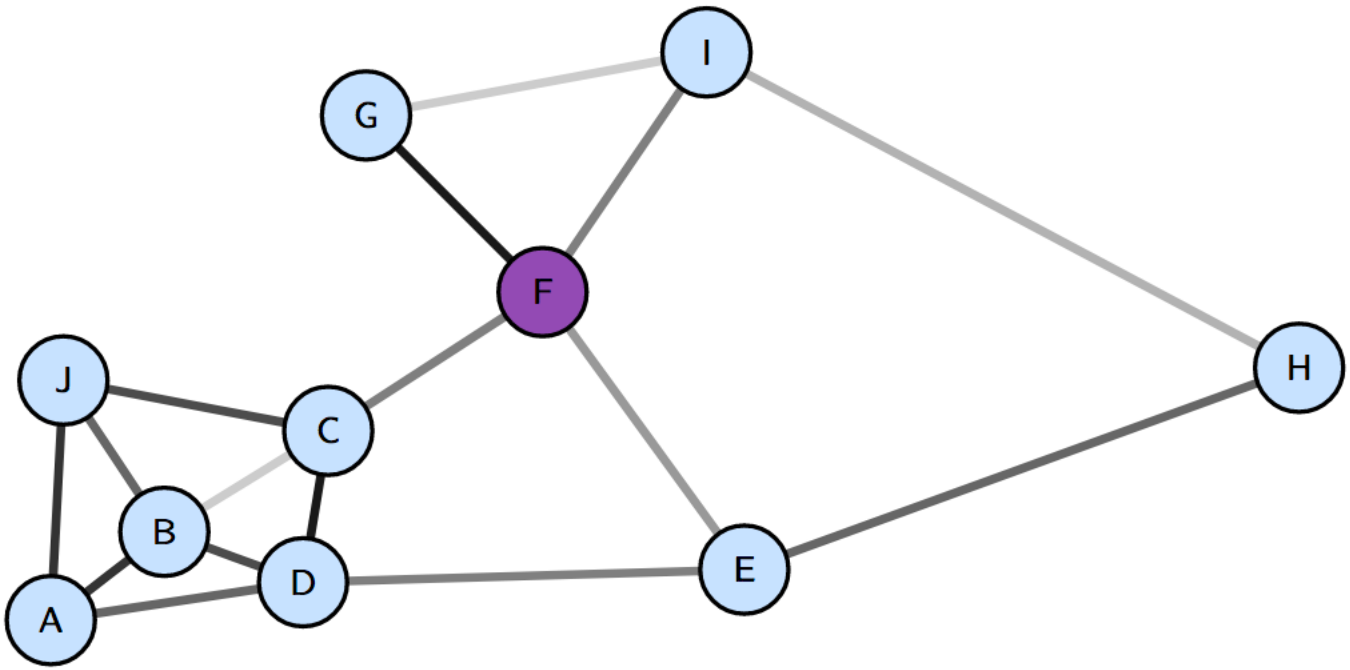
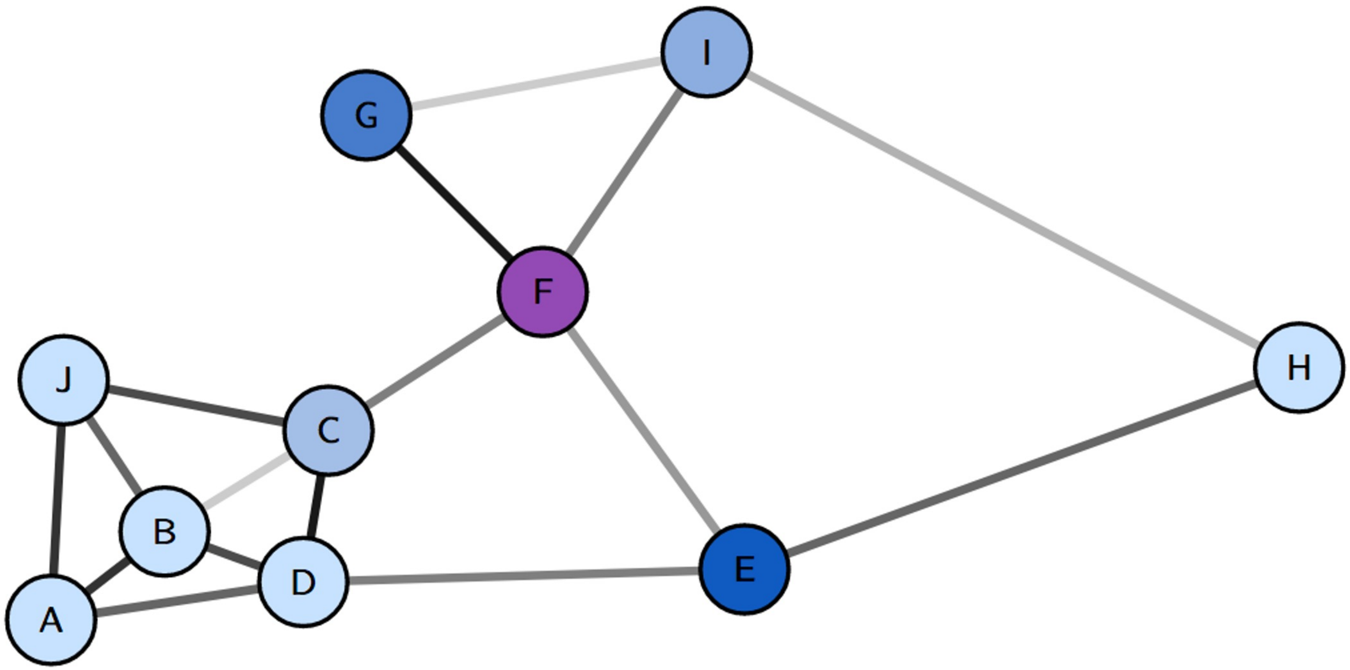


Image Credit: I. Henrion

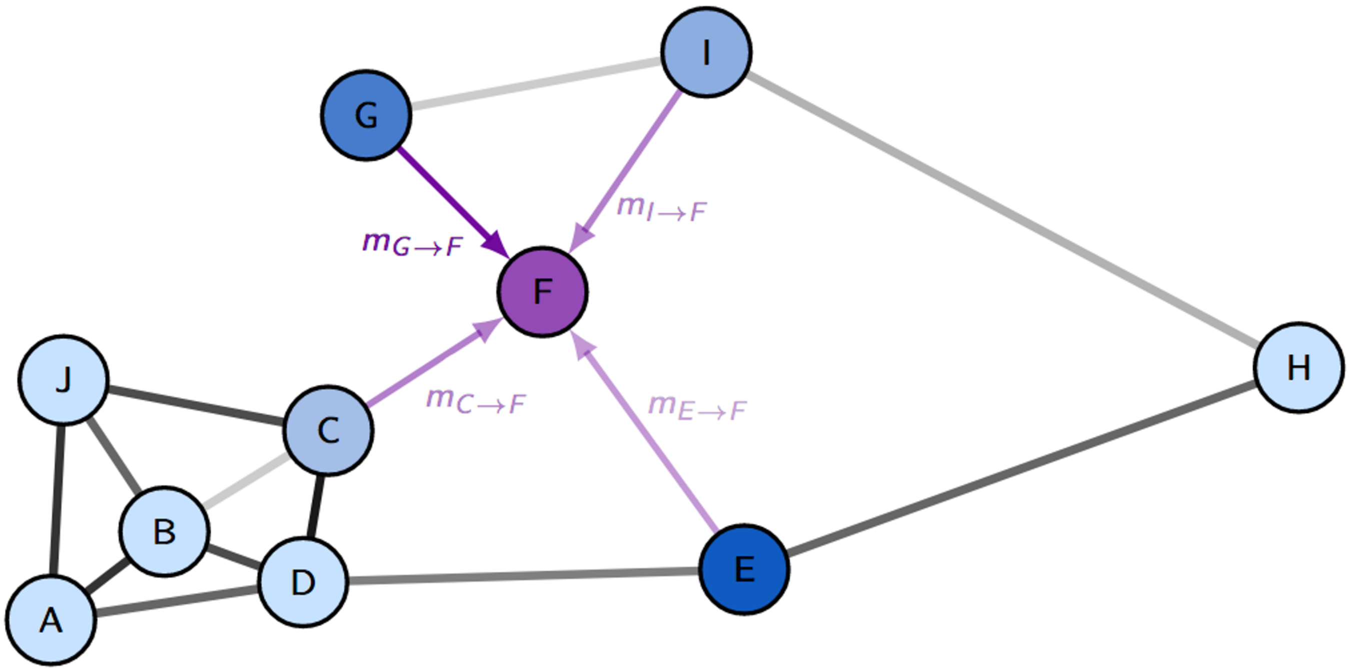
Neural Message Passing



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

Image Credit: I. Henrion

Neural Message Passing

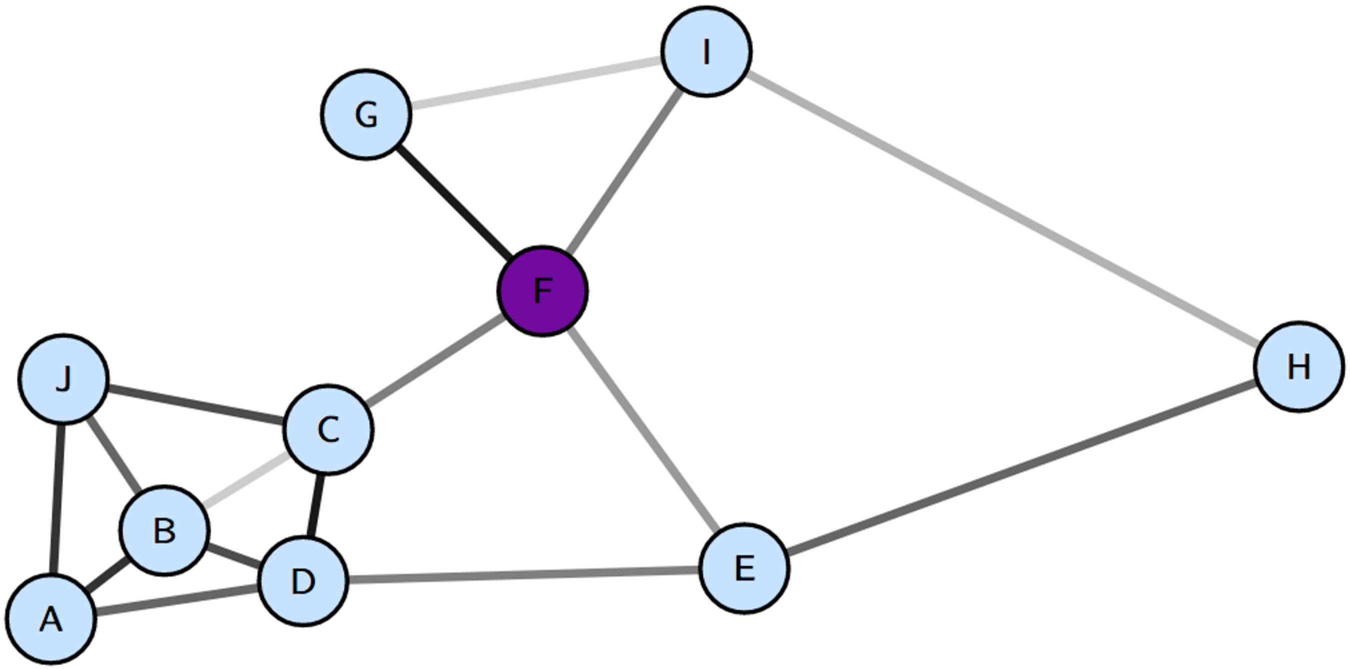


$$\tilde{m}_j^t = f(h_j^{t-1})$$

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

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

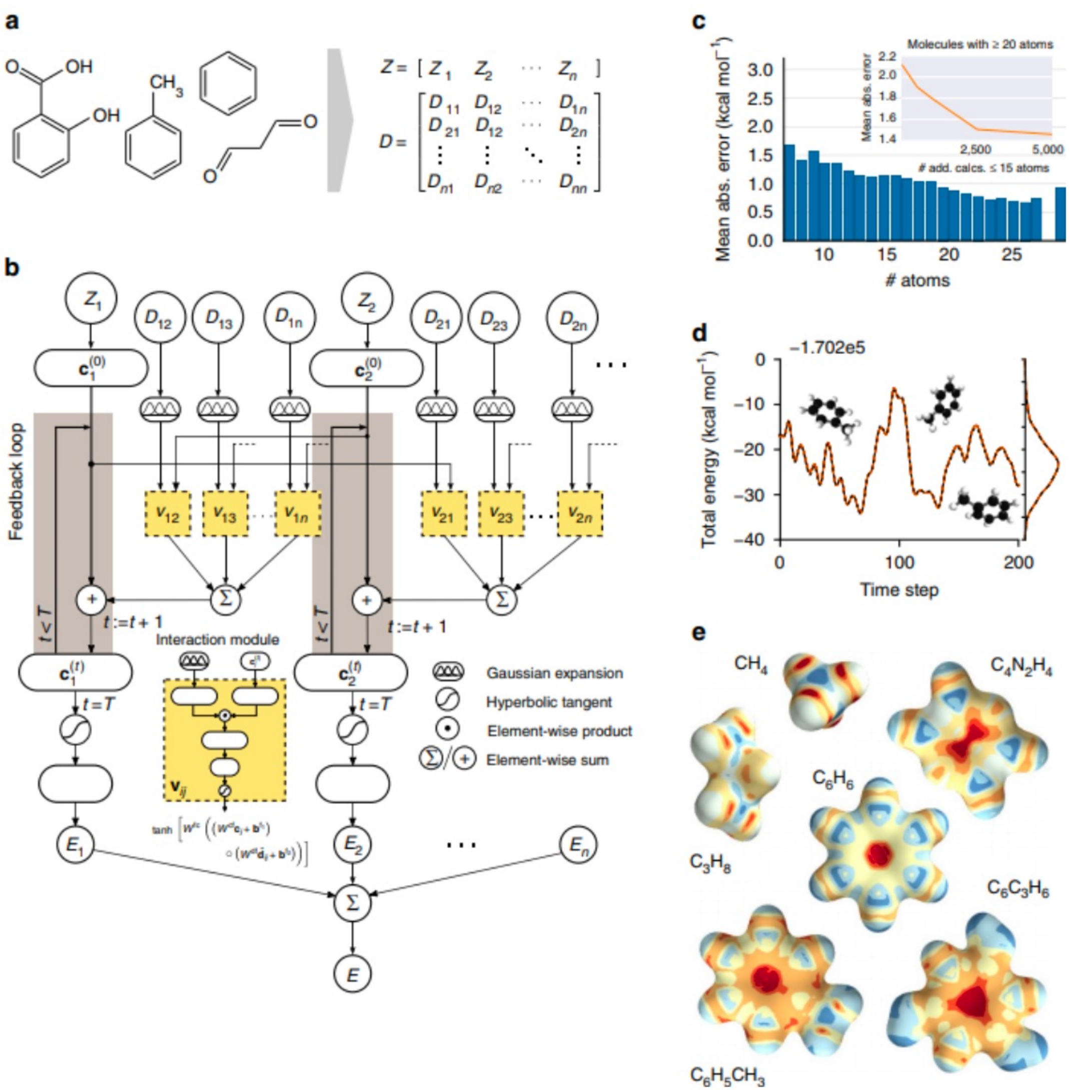
$\mathbf{r} = \text{Readout}(\mathbf{h})$

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

Image Credit: I. Henrion

Examples

Quantum chemistry with graph networks



Schutt et al. 2017

Learning to simulate physics with graph networks

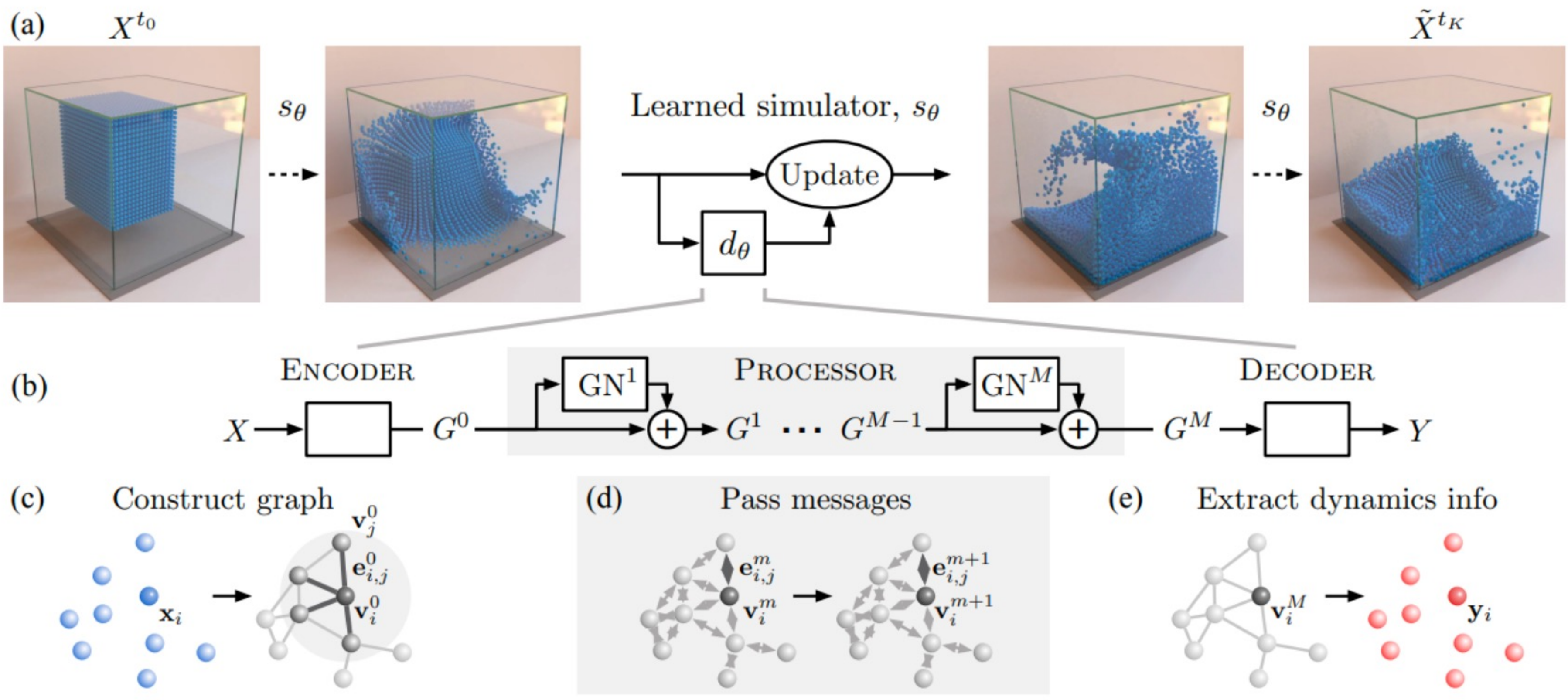


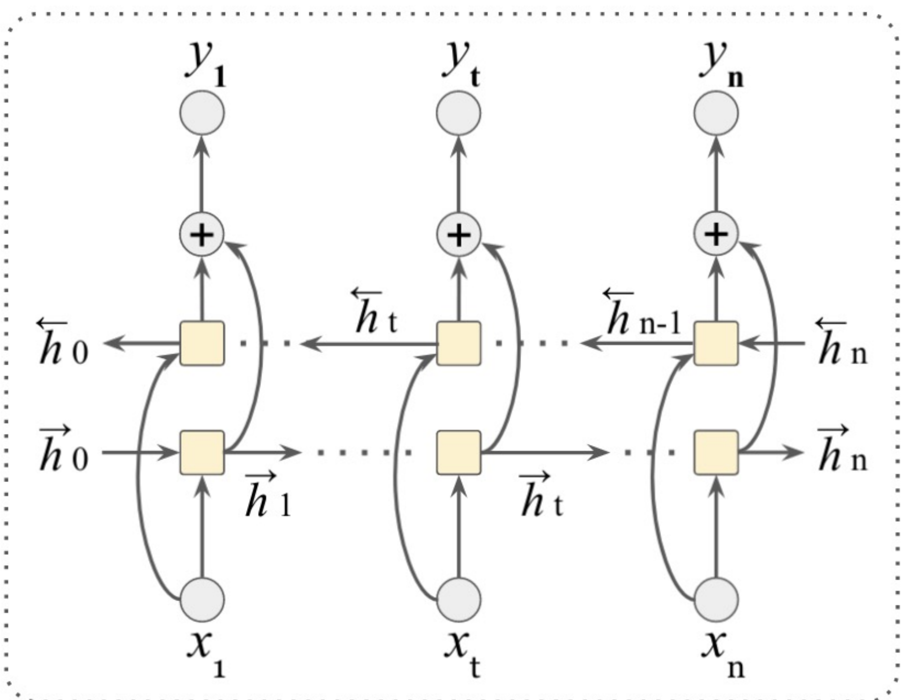
Figure 2. **(a)** Our GNS predicts future states represented as particles using its learned dynamics model, d_θ, and a fixed update procedure. **(b)** The d_θ uses an "encode-process-decode" scheme, which computes dynamics information, Y, from input state, X. **(c)** The ENCODER constructs latent graph,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

Sanchez-Gonzalez et al. 2020

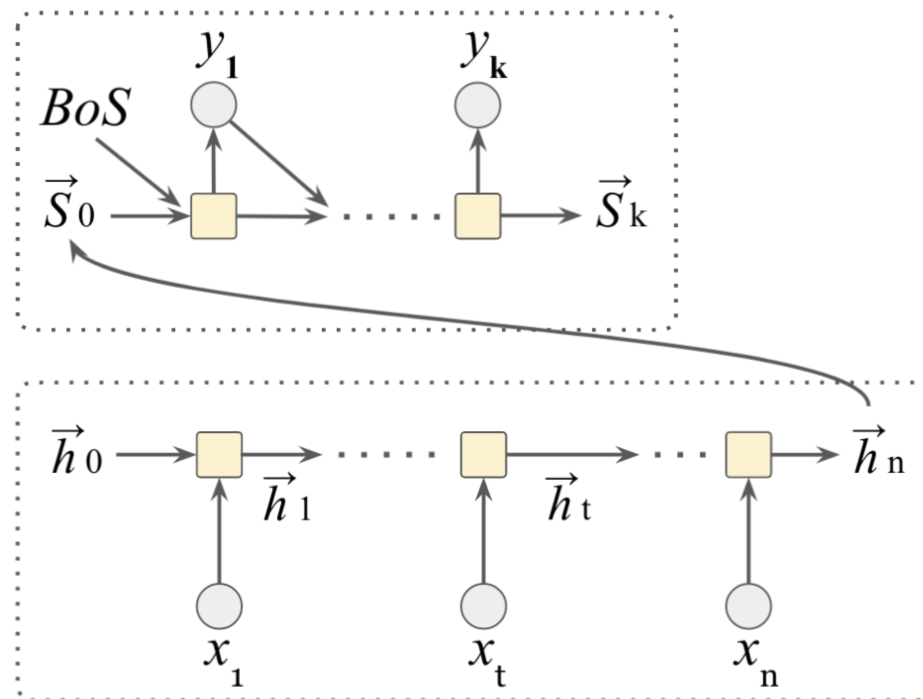
Transformers

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



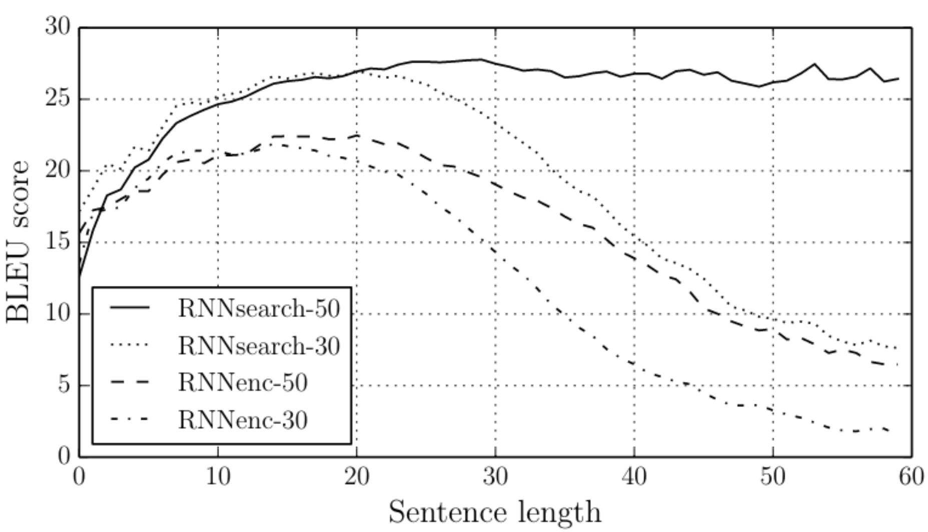
RNN Encoder-Decoder

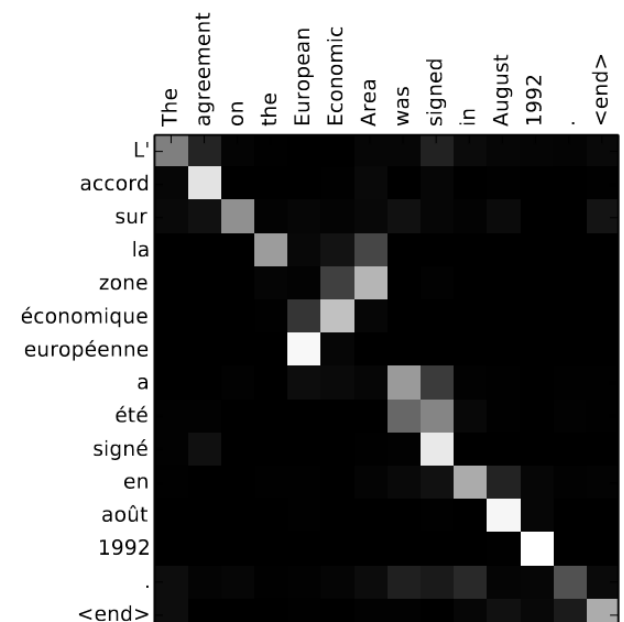
Additive Attention Mechanism

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

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

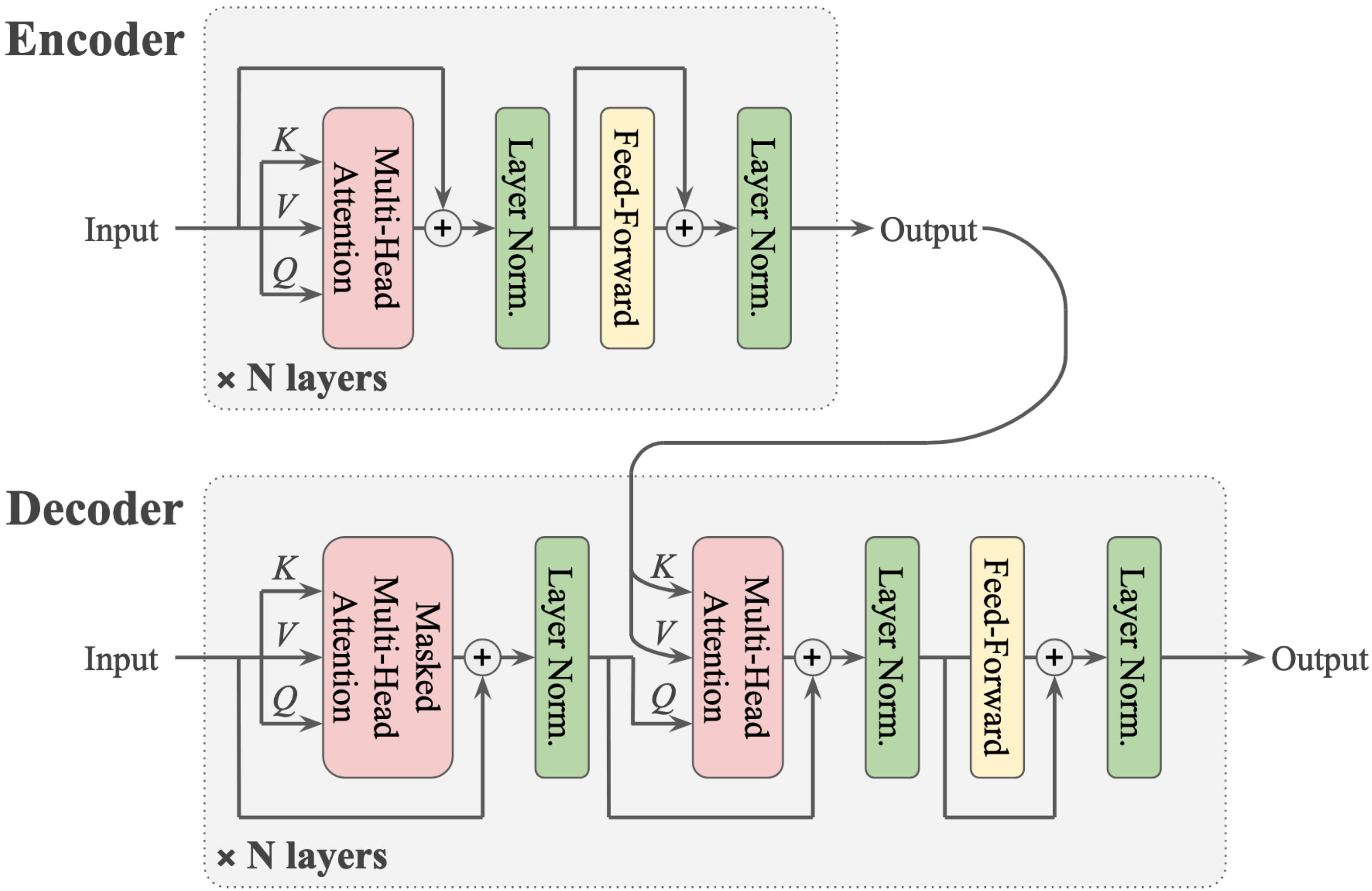
$$\text{and} \quad \beta_{ij} = U \tanh(W s_{i-1} + \widetilde{W} h_j + b_i)$$





1409.0473

- 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

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

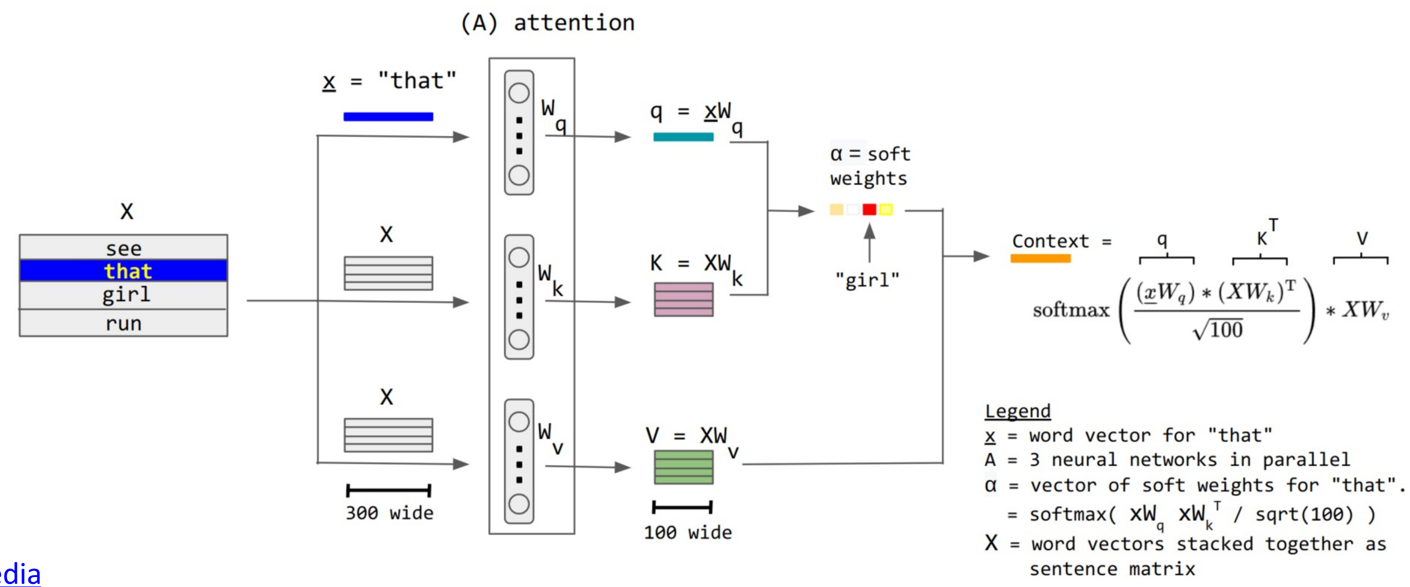


Image credit: [Wikipedia]

Attention Computations

- Lets look at a single query

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

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

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

- Generalize input to length *n*

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