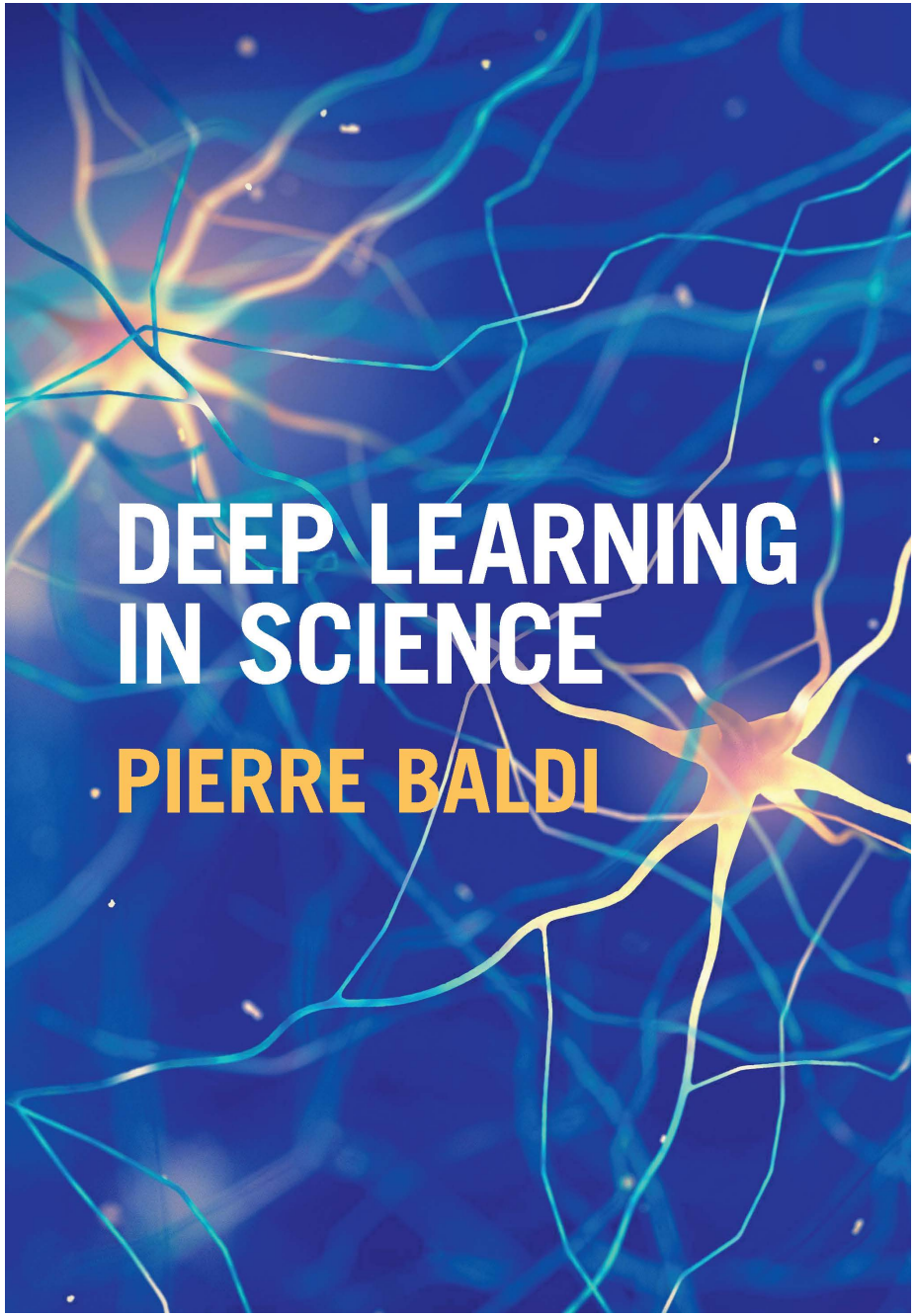
Attention in Deep Learning and Physics Applications

P.Baldi

Department of Computer Science
Institute for Genomics and Bioinformatics
Center for Machine Learning and Intelligent Systems
University of California, Irvine

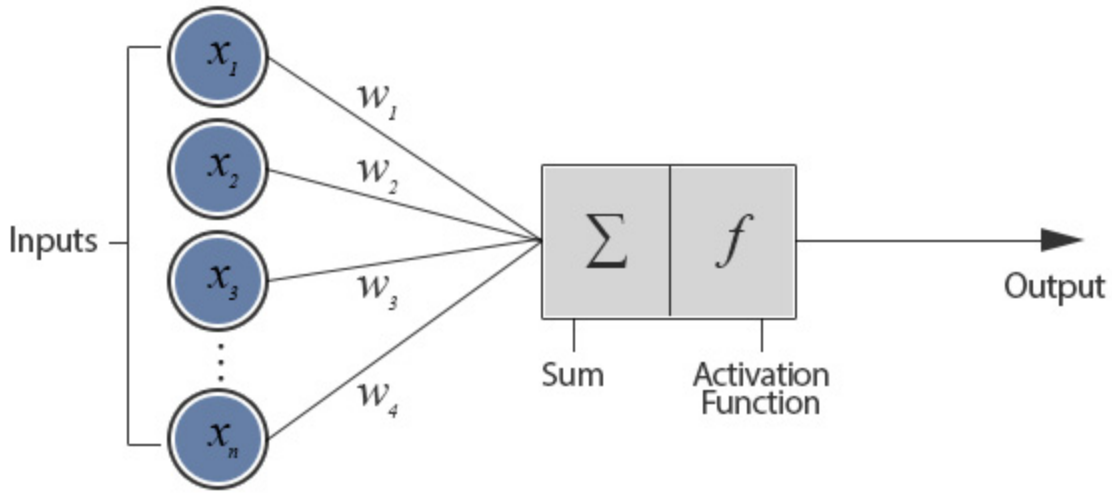


Cambridge University Press
TOC and sample chapters on
my web site.

Basic Facts

- Attention mechanisms are widely used in modern deep learning (e.g. transformers)
- Attention is a modern rebranding of “gating”, in the same way that deep learning is a modern rebranding of neural networks.
- Attention expands the repertoire of differentiable operations available in neural networks.
- It renders the roles of synaptic weights and neuronal activities more symmetric.
- Attention expands the repertoire of functions that can be implemented in an efficient way by enabling modulation.
- It allows the function implemented by a neuron/layer/network to be modulated by another neuron/layer/network.

Simple Neuronal Model



$O=f(\sum w_i x_i)$

Two basic elementary operations:

1) Dot product $x.w$ (activation)

2) Application of linear or non-linear activation function

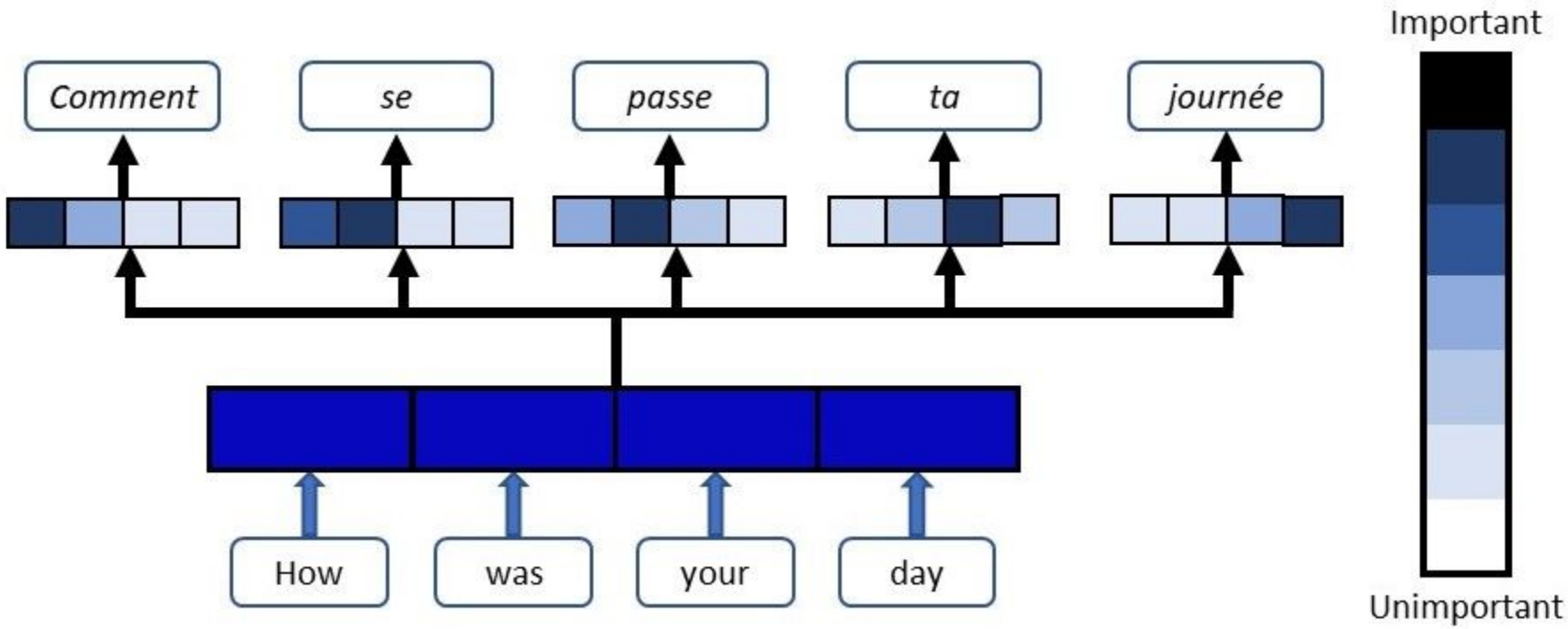
Attention Mechanisms

Motivation:

- the brain pays different amount of attention to regions of an image, or locations in a sequence.

Various formulations:

- Content-base attention Graves et al., 2014
- Dot-Product attention Luong et al., 2015
- Additive attention Bahdanau et al., 2015
- …



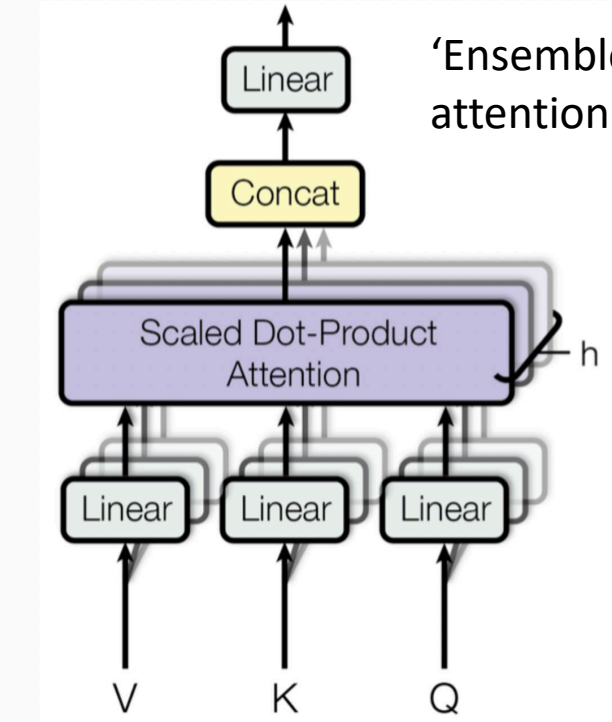
Sequence to sequence models

Transformer Model & (self)-attention

The Transformer Model is **entirely** built on the self-attention mechanisms, **without** using sequence-aligned recurrent architectures.

Every input element has three learnable vectors: Query (Q), Key (K), and Value (V)

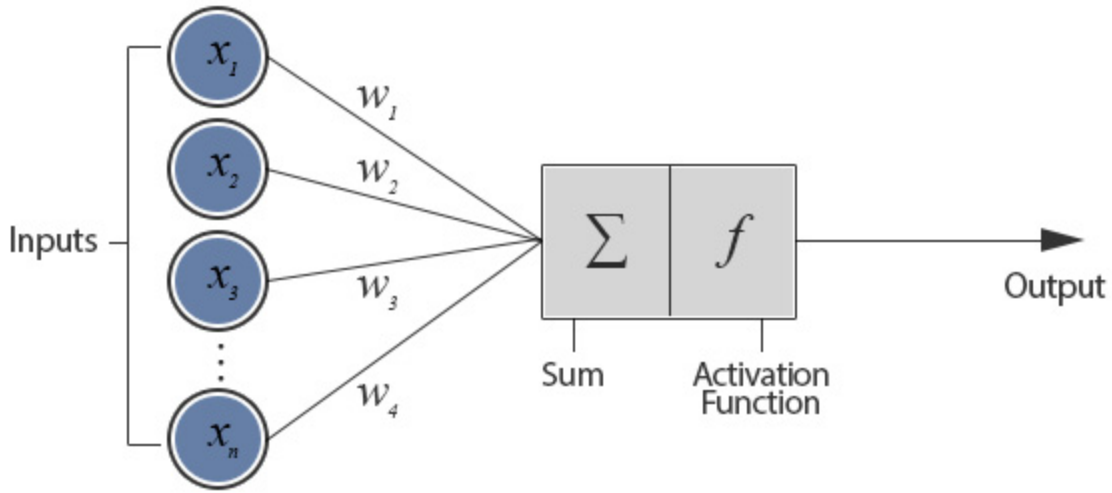
$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{n}})\mathbf{V}$$



'Ensemble' multiple attention in parallel

Rather than only computing the attention once, the multi-head mechanism runs through the scaled dot-product attention multiple times in parallel.

Simple Neuronal Model



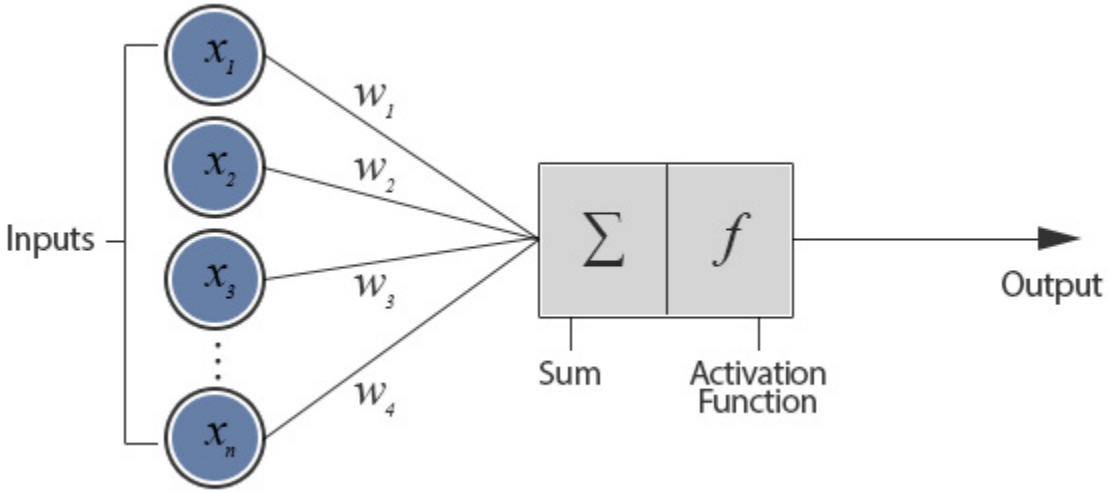
$O=f(\sum w_i x_i)$

Two basic elementary operations:

1) Dot product $x.w$ (activation)

2) Application of linear or non-linear activation function

More Complex Neuronal Models



O=f ($\sum w_{ij} x_i \, x_j$ + $\sum w_i x_i$)

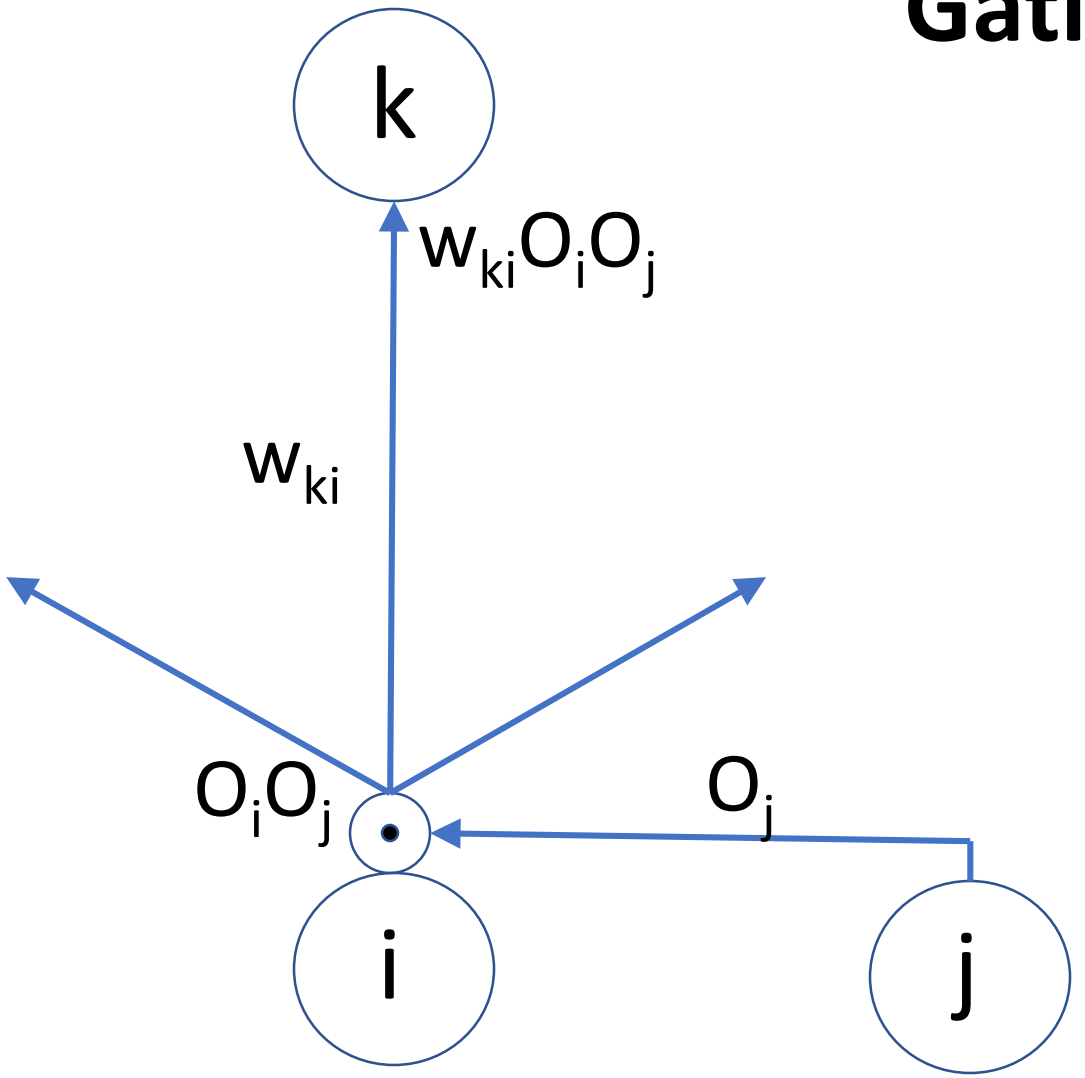
Two basic elementary operations:

1) Polynomial of degree d (e.g. d=2) (activation)
2) Application of linear or non-linear activation function

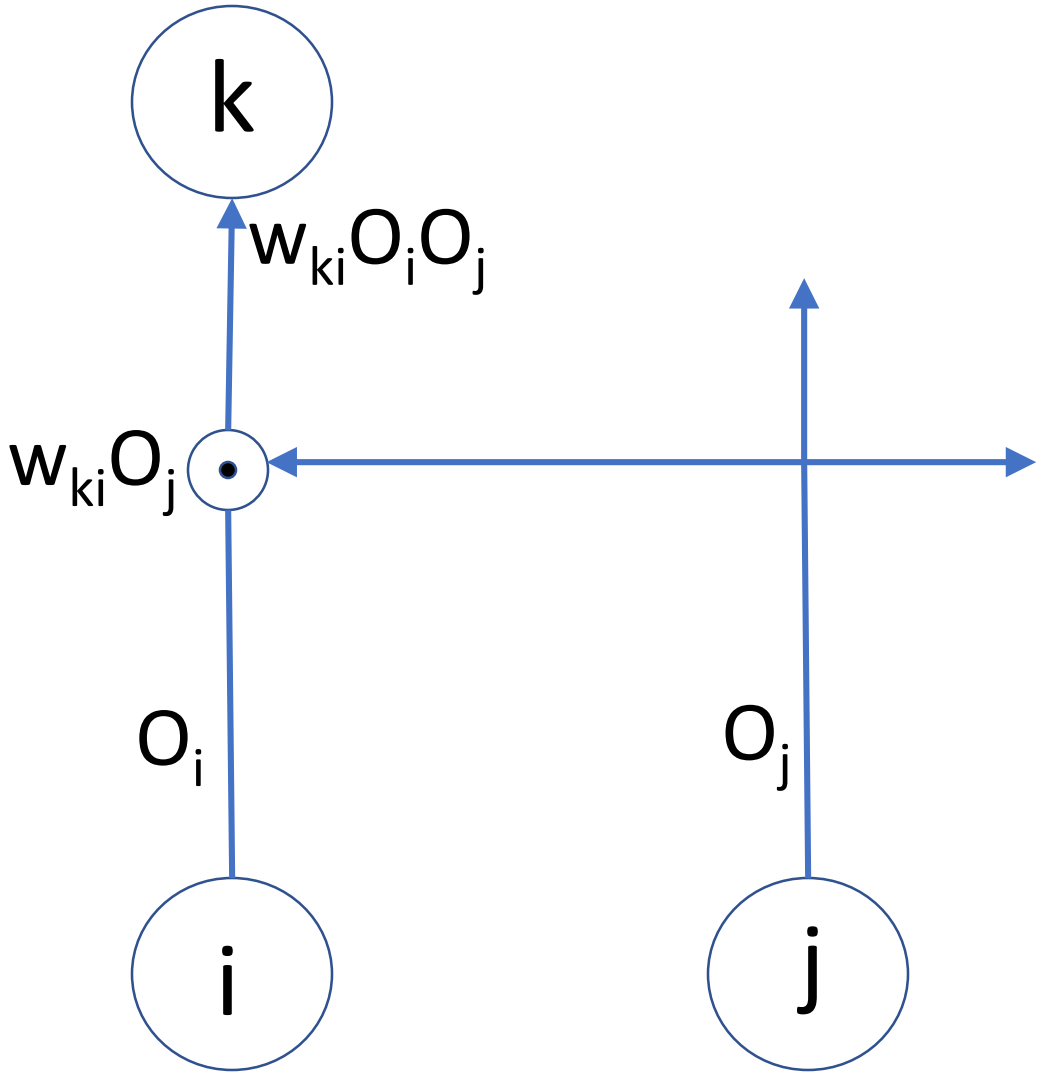
More powerful neuronal model.

However it requires $O(n^d)$ parameters and multi-way synapses, as well as the ability to multiply neuronal outputs (violates locality principle in a physical system).

Gating



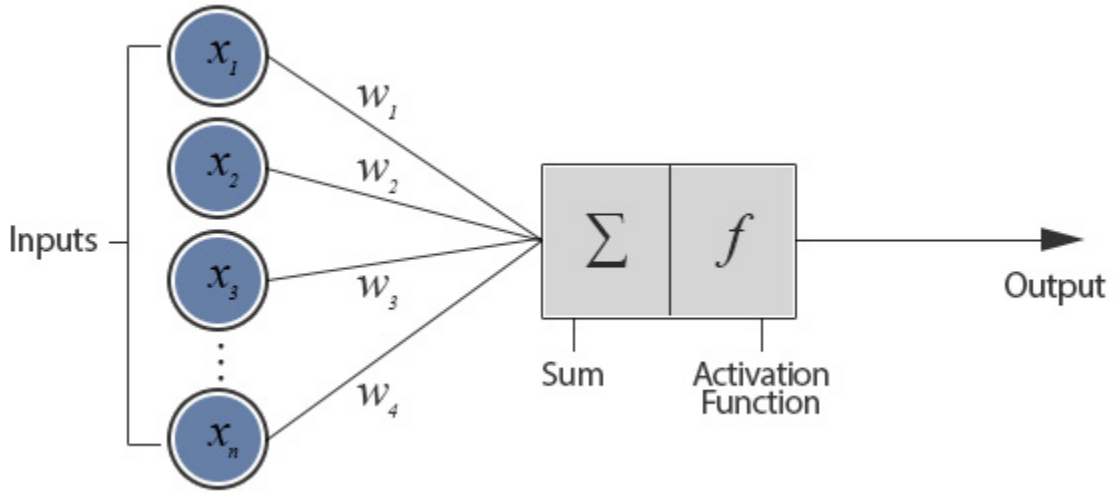
Output Gating



Synaptic Gating

When neuron i has a single outgoing connection, output gating and synaptic gating are equivalent.

Simple Neuronal Model with Attention

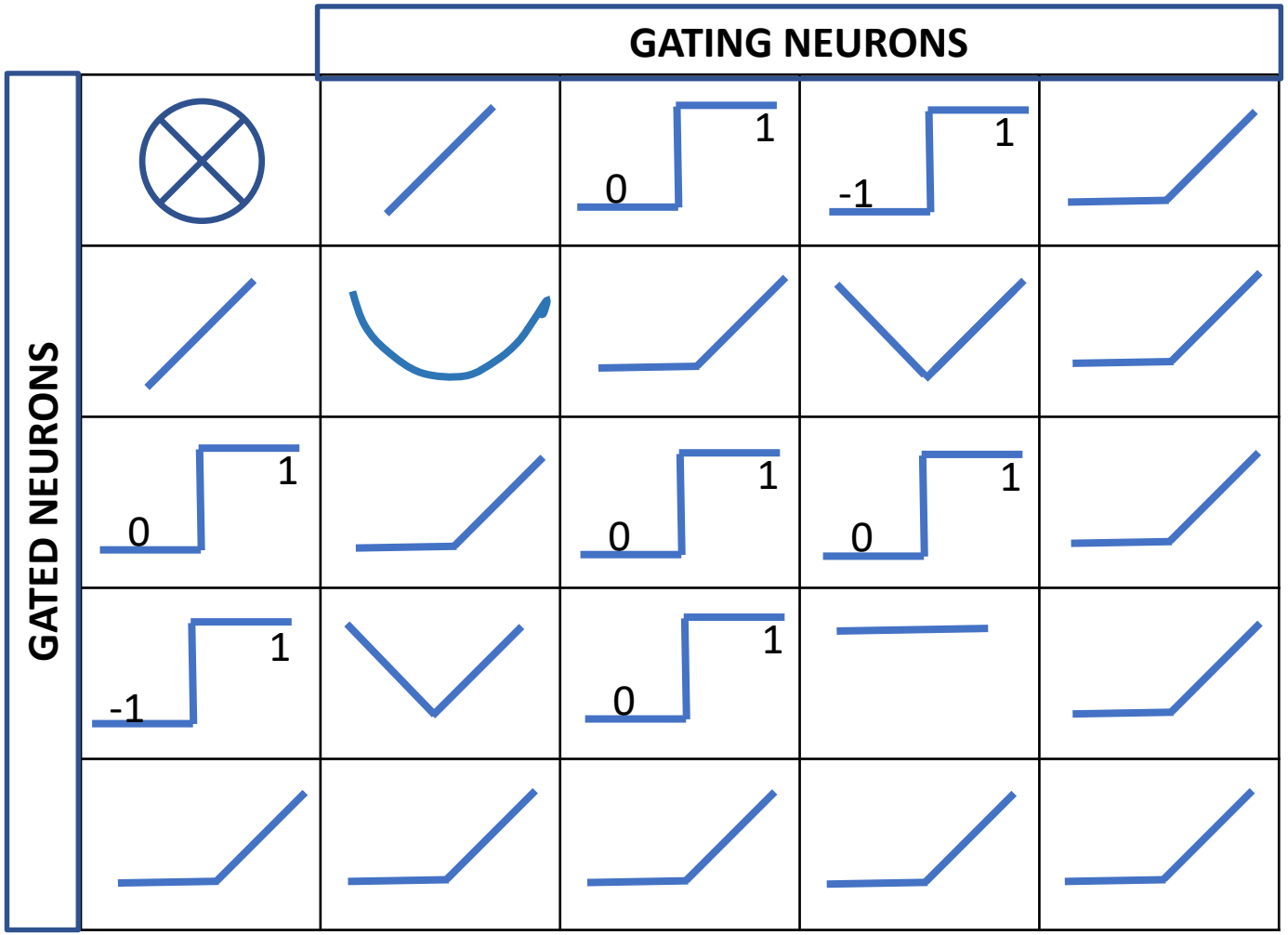


$O=f(\sum w_i x_i)$

Three basic elementary operations:

1) Dot product x.w (activation)
2) Application of linear or non-linear activation function
3) Products of neuronal outputs.

Effect of Gating on Transfer Functions

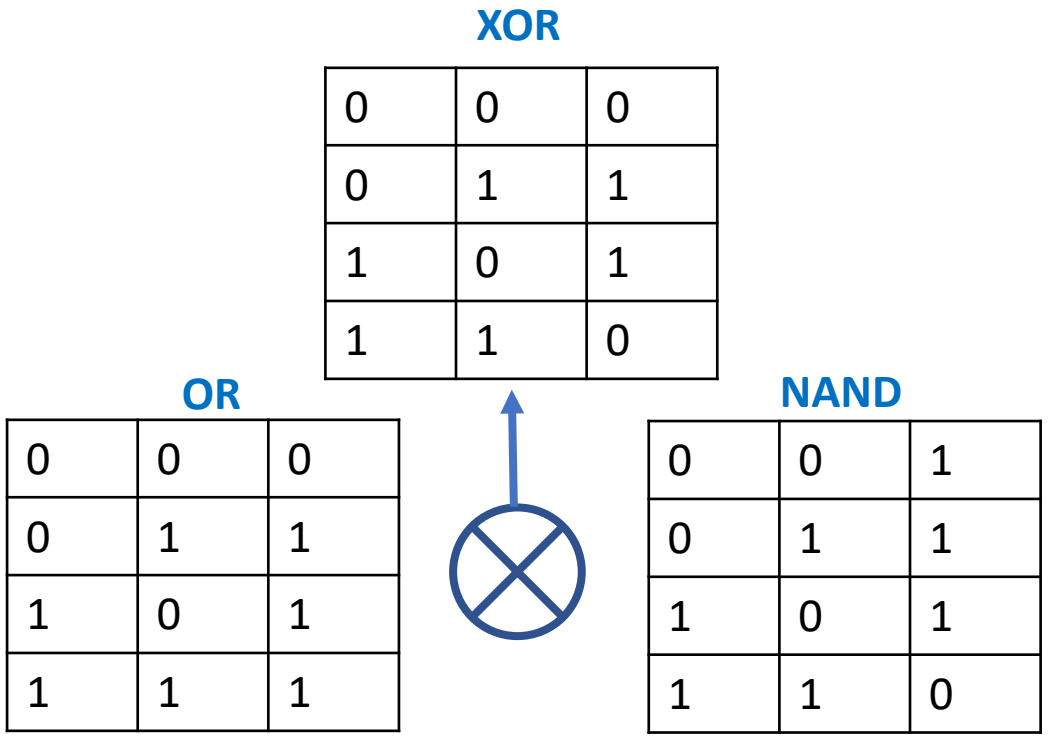


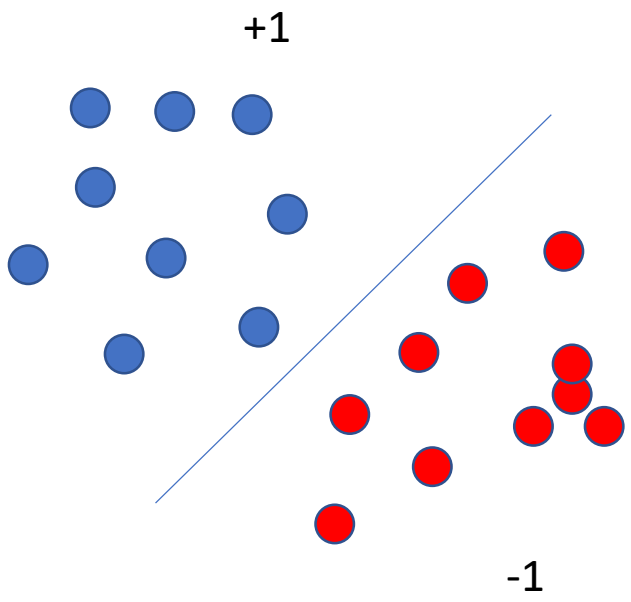
Assuming the gated and gating function have the same sign and are centered at 0.

Table is symmetric with respect to the diagonal since gating is a product.

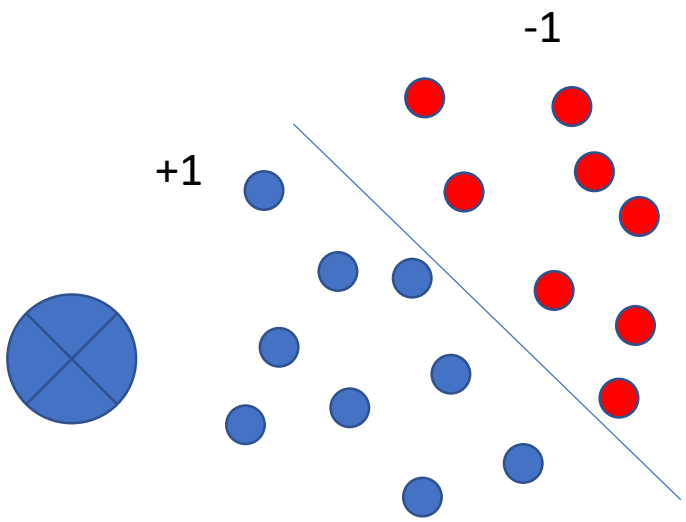
XOR Computed in a Single Layer with Gating

- AND, OR, NOT, NAND, NOR can easily be computed by a single neuron.
- However XOR cannot be computed by a single neuron without gating (shallow). It requires at least one hidden layer (deep).
- XOR = NAND gating OR

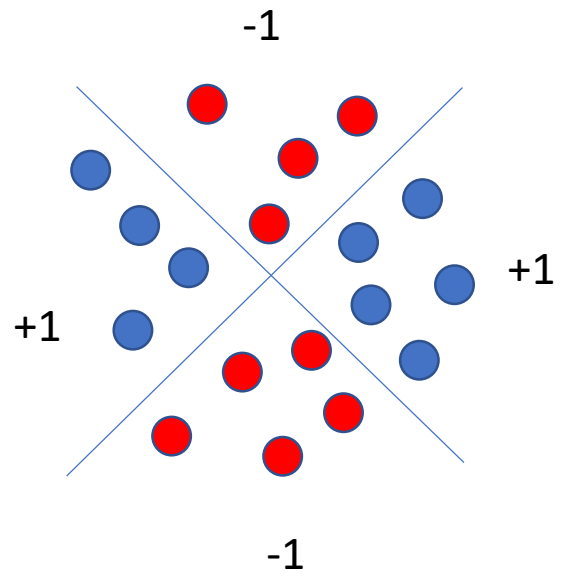




Random Linear
Threshold Function

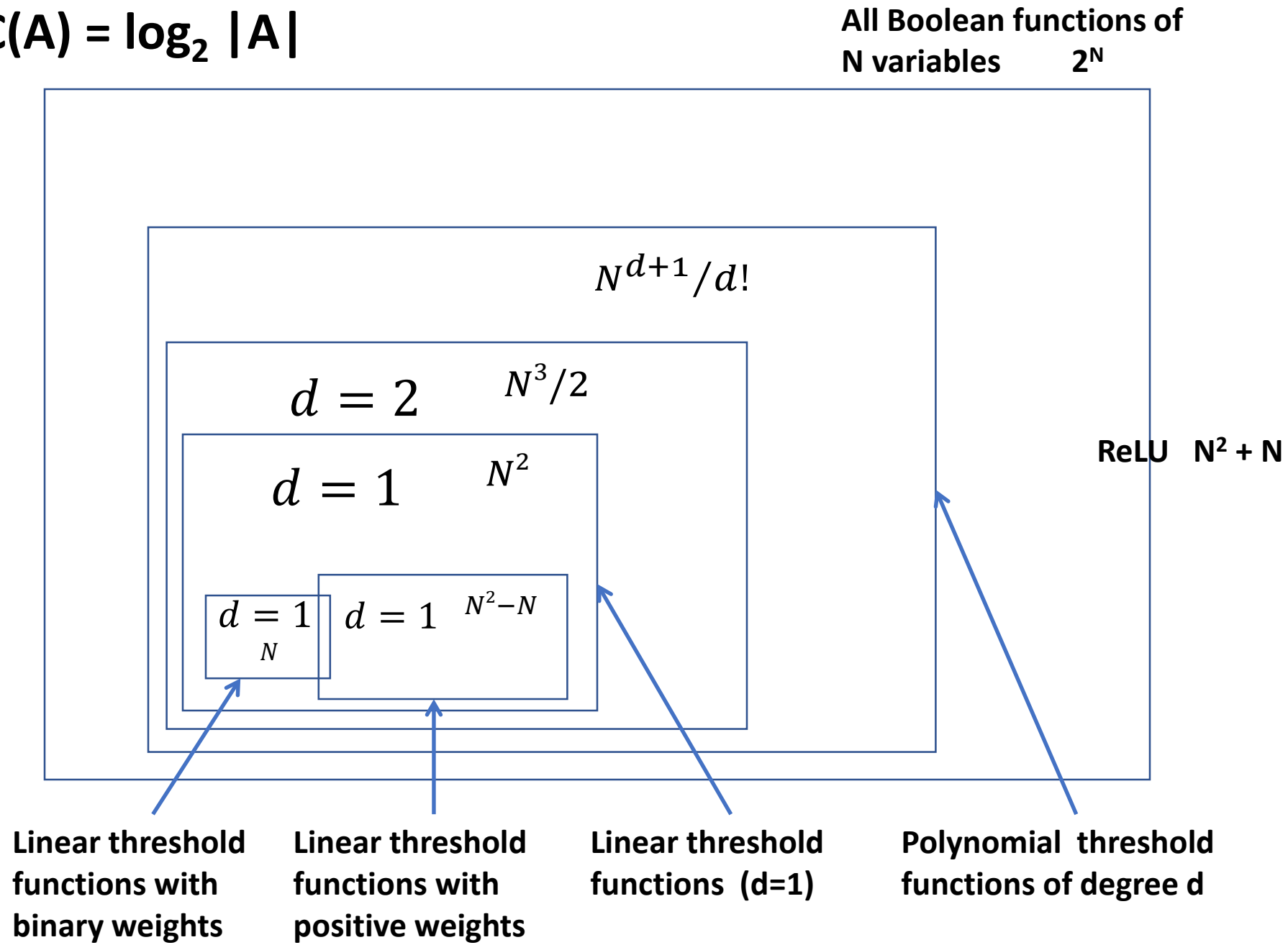


Random Linear
Threshold Function



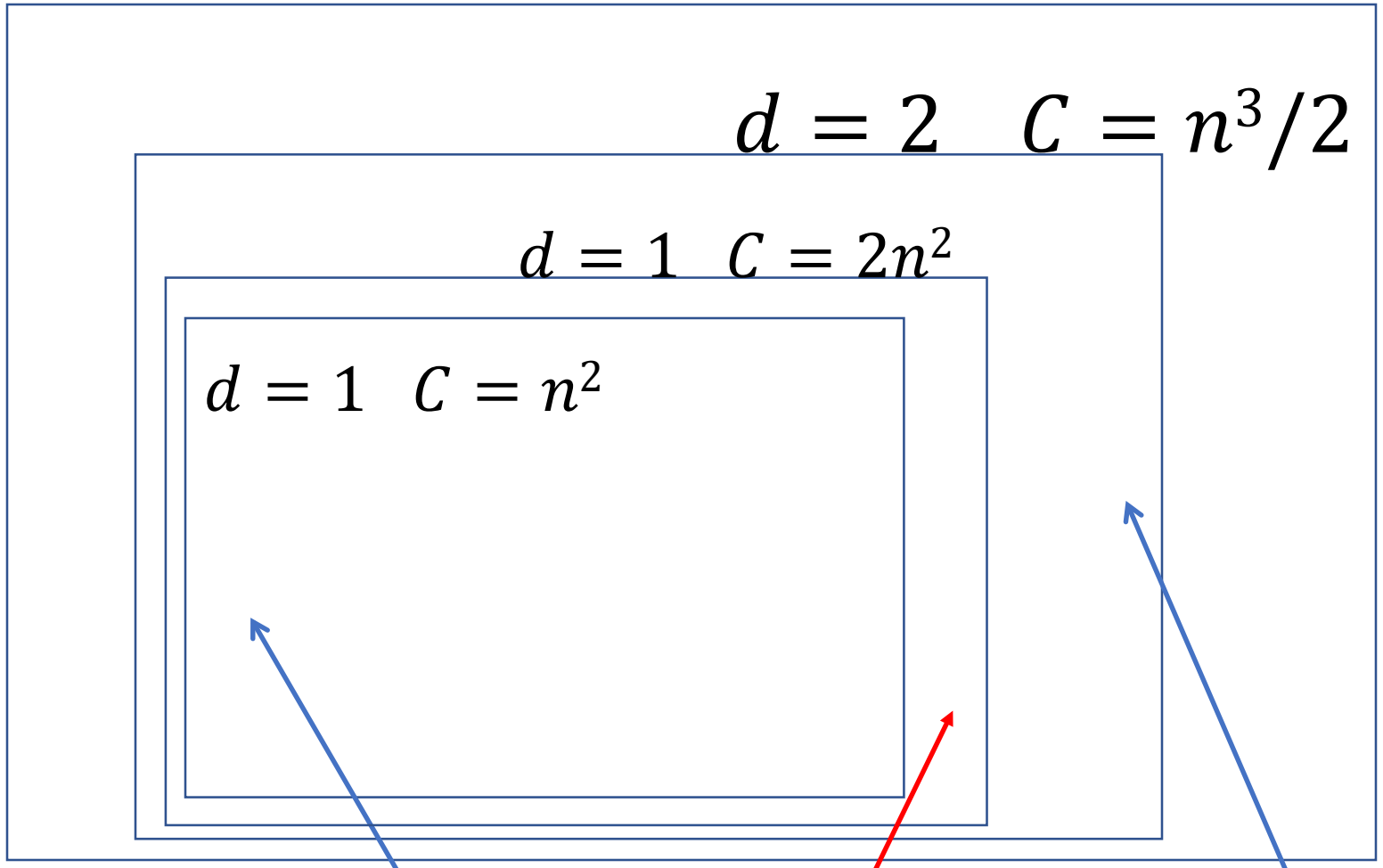
New Function
Non-linearly Separable

Capacity: C(A) = $\log_2$ |A|



Capacity: $C(A) = \log_2 |A|$

All Boolean functions of n variables 2^n



Linear threshold functions (d=1) n parameters

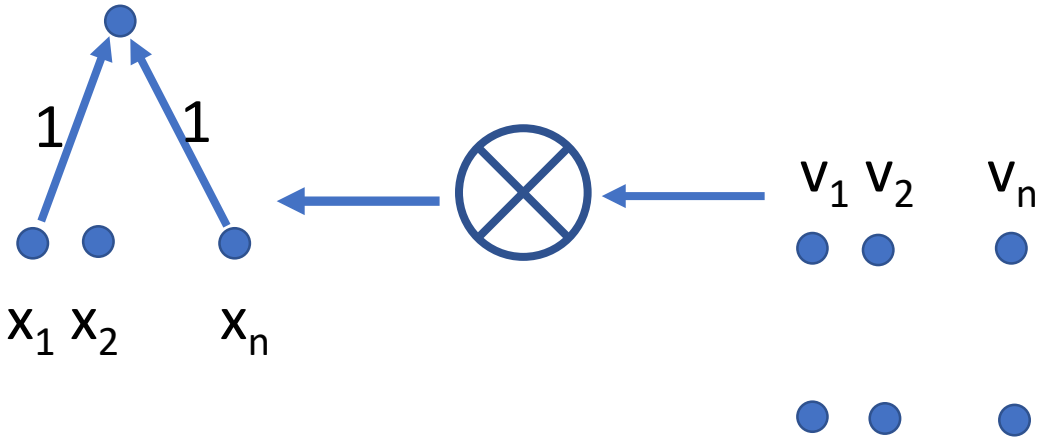
Linear threshold functions (d=1) with gating 2n parameters

Polynomial threshold functions of degree d=2 n^2 parameters

Attention Enables Computing the Dot Product of the Activities of Two Layers of the Same Size

gated output

O= $\sum x_i v_i$

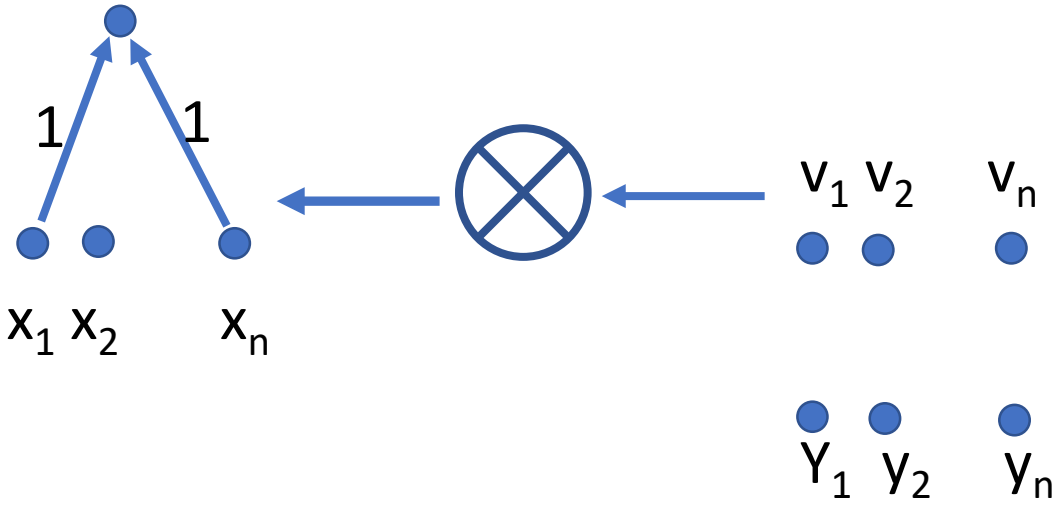


pairwise gating layer

Softmax Attention=Dot Product with Softmax

gated output

$O=\text{sum}_i\, v_i\, x_i$

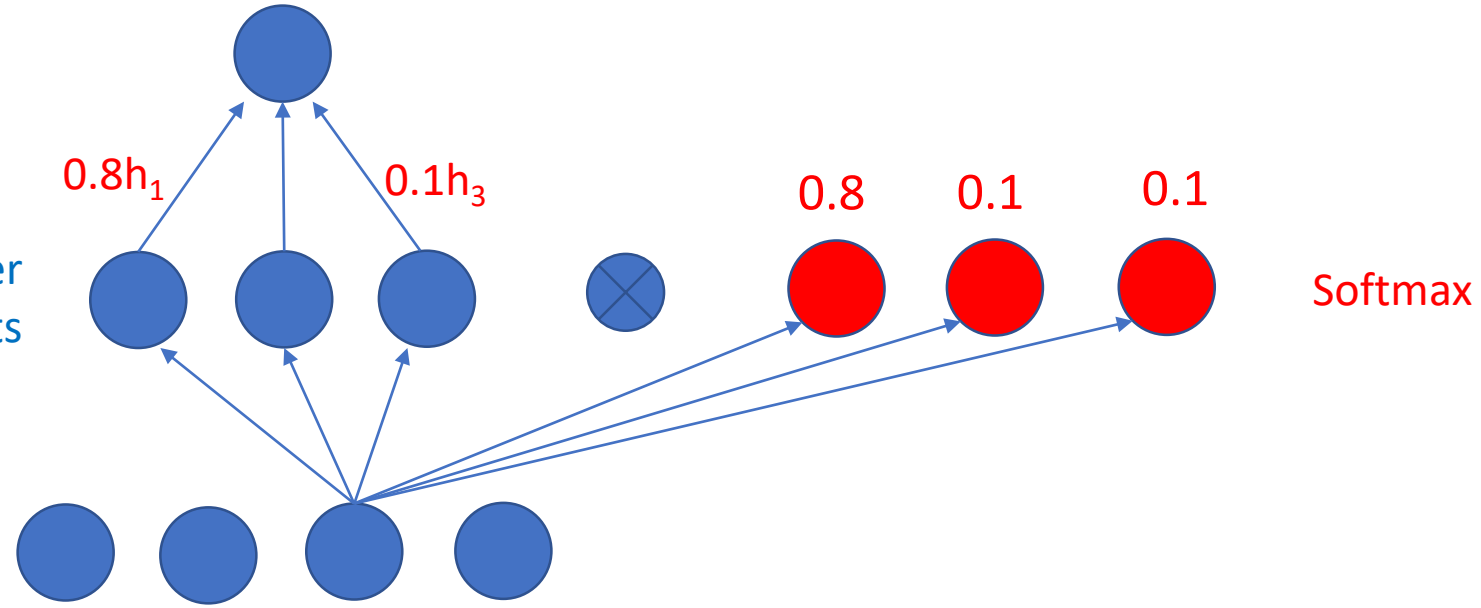


gating layer:

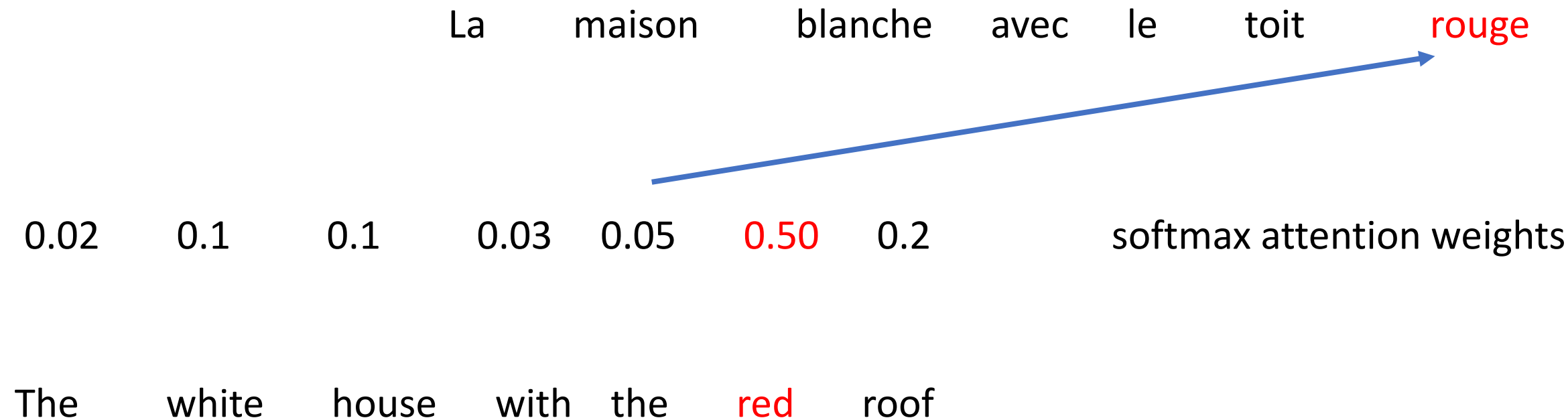
softmax unit

$vi=\exp y_i / \text{sum}_j \exp v_j$

Typical Softmax Attention Layer

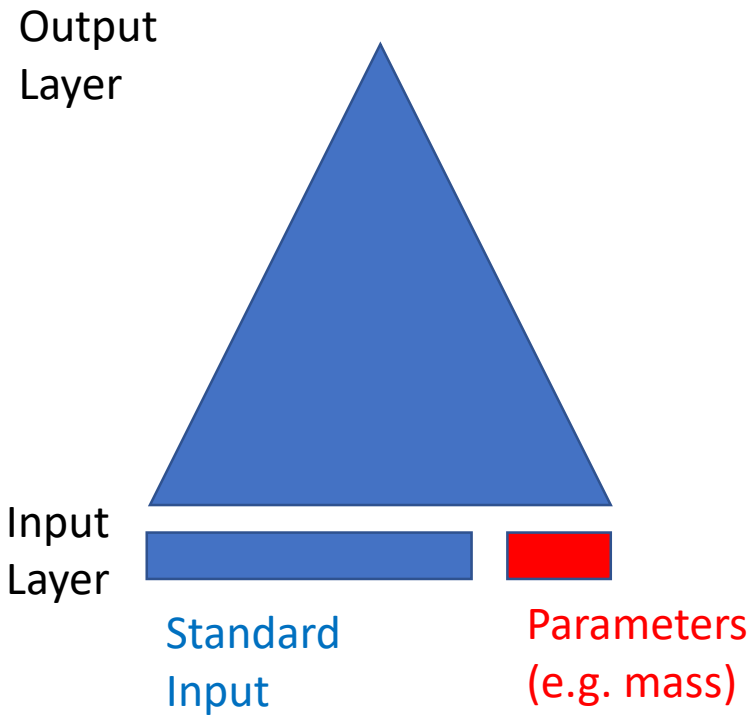


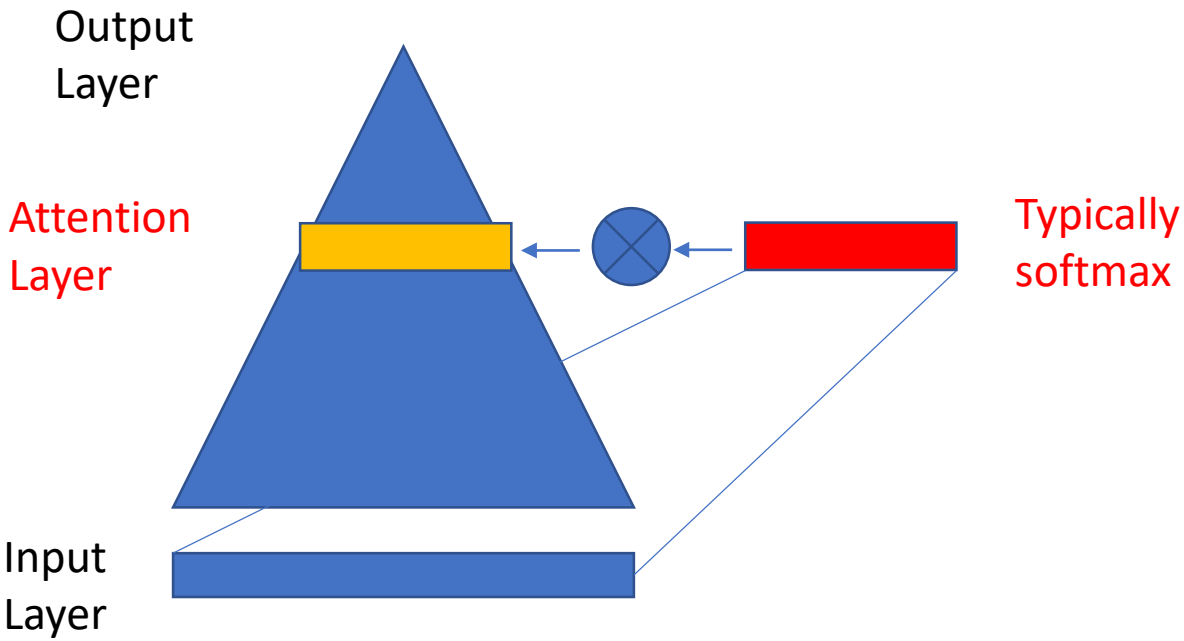
Attention in Natural Language Processing



Parameterized Networks and Attention

- Create neural networks that can be modulated and compute more than one function





Attention Mechanisms

Motivation:

- the brain pays different amount of attention to regions of an image, or locations in a sequence.

Various formulations:

- Content-base attention Graves et al., 2014
- Dot-Product attention Luong et al., 2015
- Additive attention Bahdanau et al., 2015
- …

- **Basic mechanism:** product of neuronal outputs (gating) which, in combination with softmax can produce attention modulating layers

- Google's BERT, OpenAI's GPT and the more recent XLNet are the more popular NLP models today and are largely based on self-attention and the Transformer architecture.

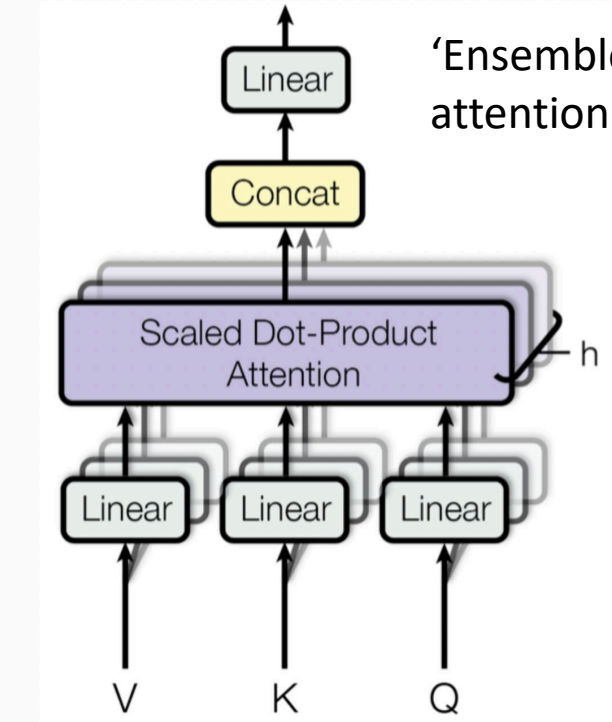
- Standard modules in DL packages (TensorFlow, PyTorch).

Transformer Model & (self)-attention

The Transformer Model is **entirely** built on the self-attention mechanisms, **without** using sequence-aligned recurrent architectures.

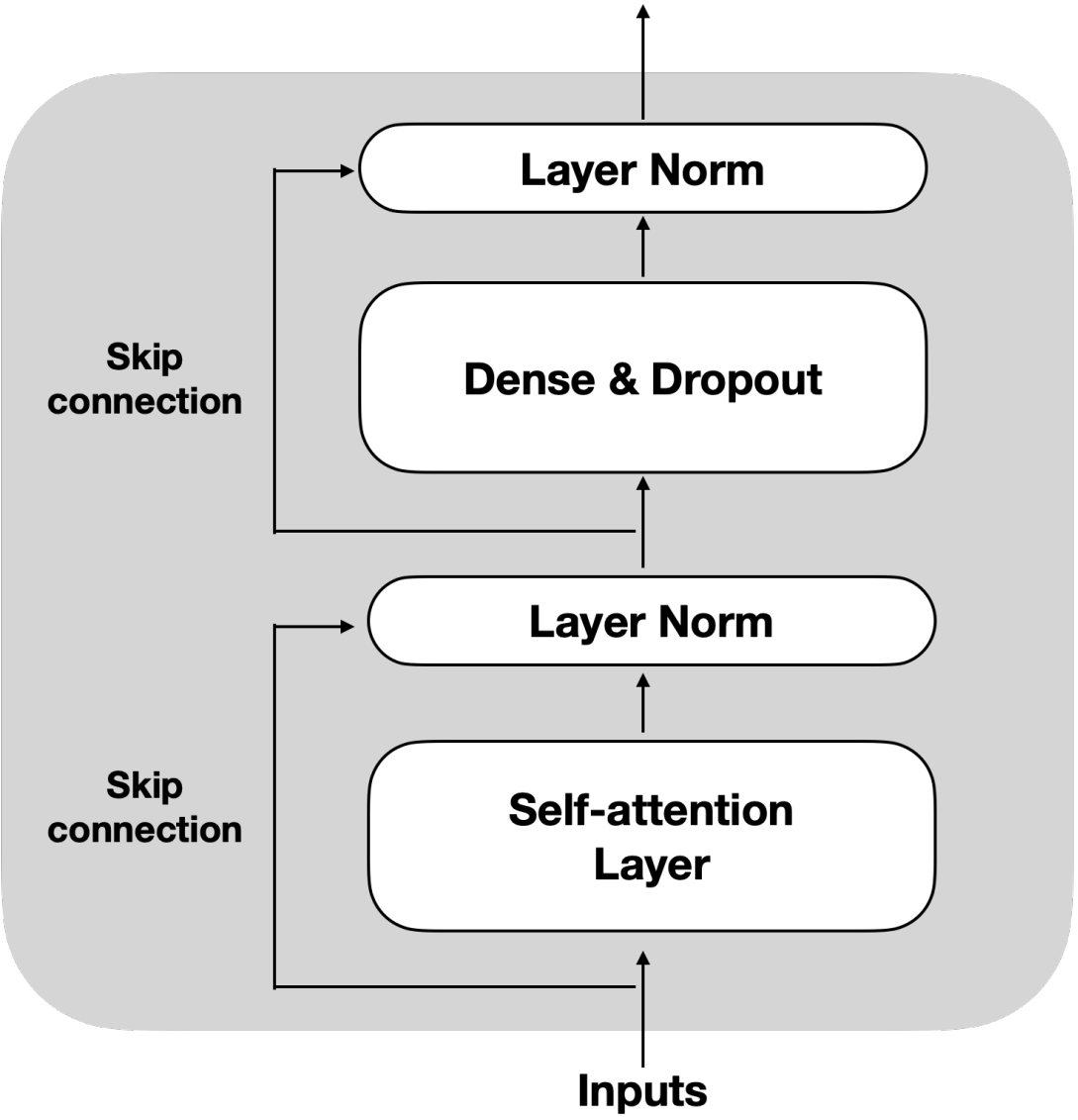
Every input element has three learnable vectors: Query (Q), Key (K), and Value (V)

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{n}})\mathbf{V}$$

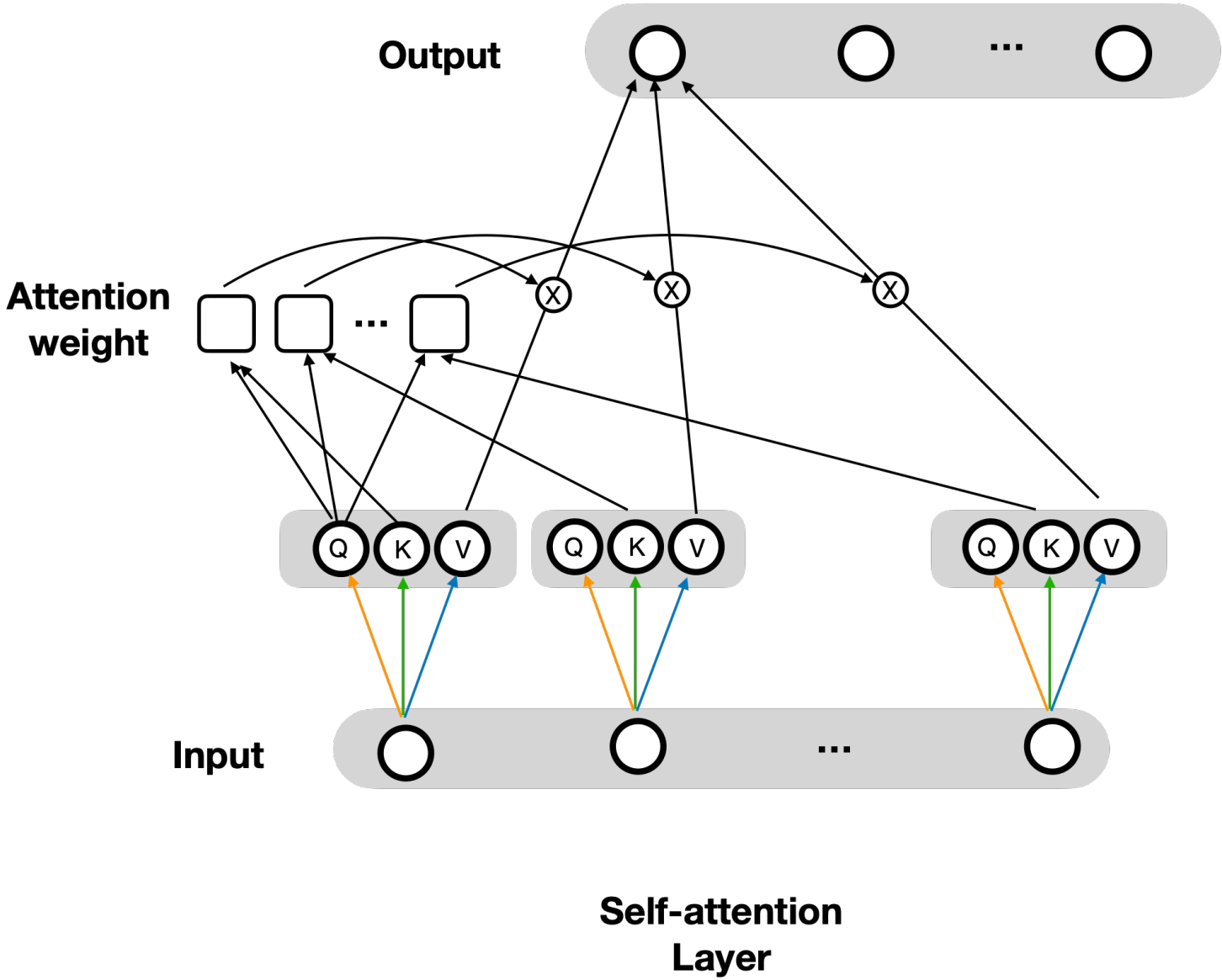


'Ensemble' multiple attention in parallel

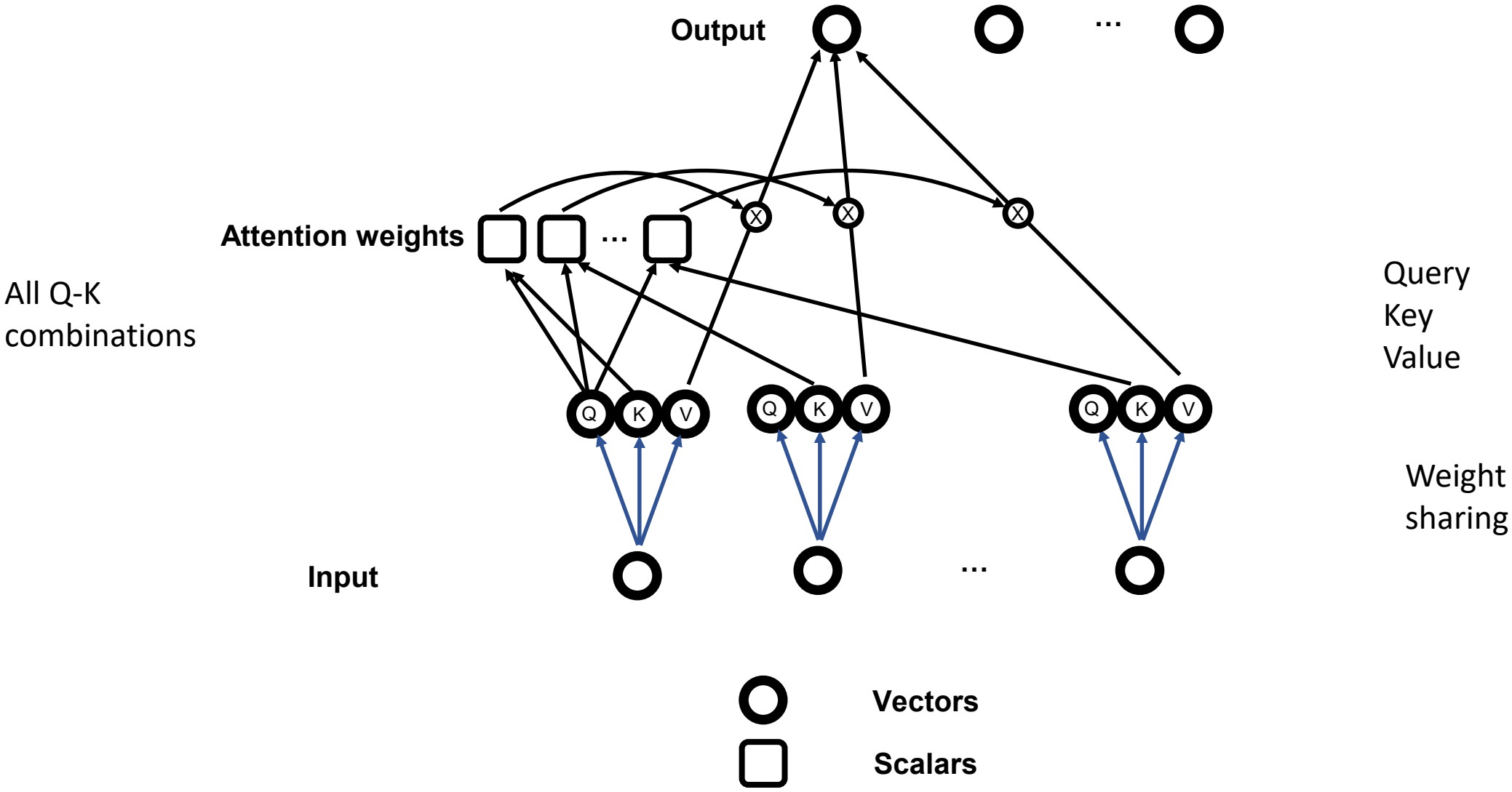
Rather than only computing the attention once, the multi-head mechanism runs through the scaled dot-product attention multiple times in parallel.

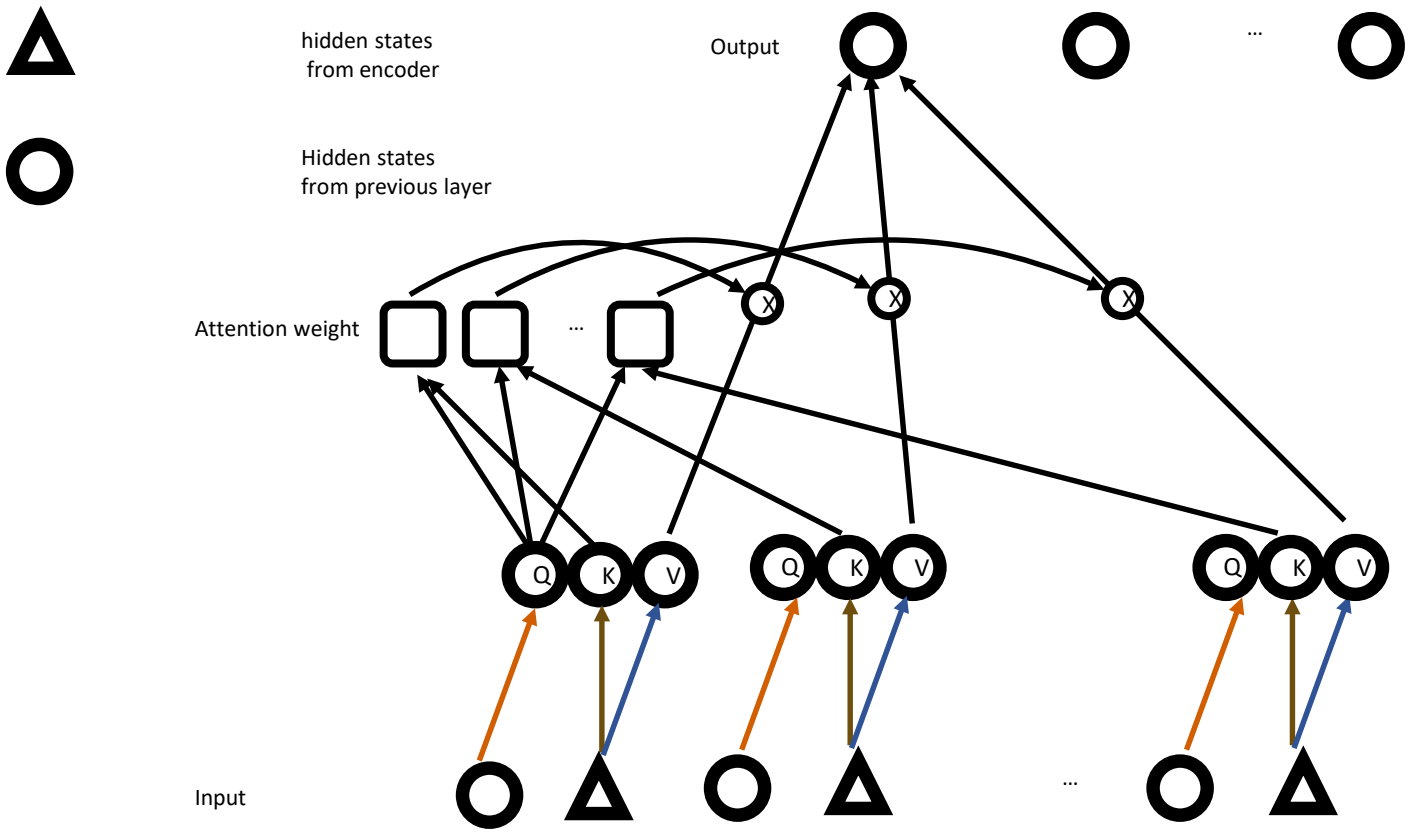


Transformer Block



Self-attention Layer


Output
Attention weights
All Q-K combinations
Query
Key
Value
Weight sharing
Input
Vectors
Scalars


hidden states
from encoder
Hidden states
from previous layer
Output
Attention weight
Input
Q
K
V
X

Remarks

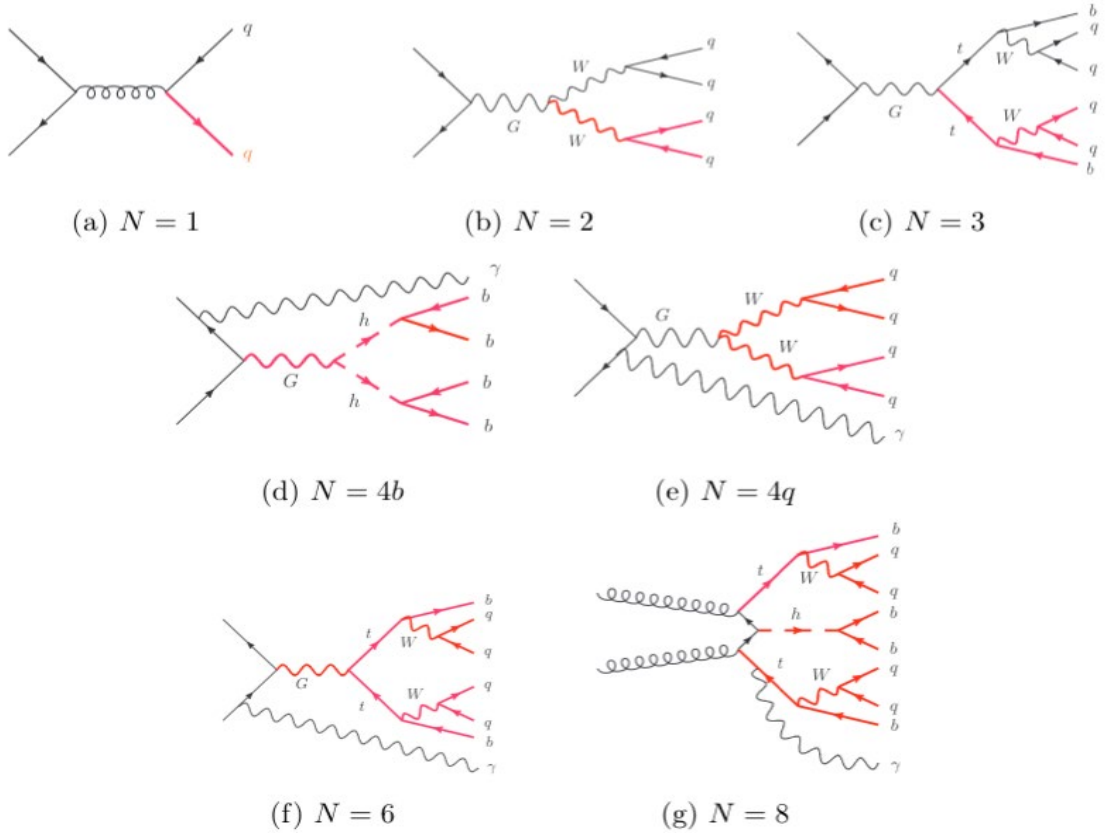
- Permutation invariant
- Surprising since originated with sequence processing

Physics Applications

- Tagging Extreme Jets
- Jet Parton Matching
- Neutrino Classification
- Neutron Star Analysis

Tagging Extreme Jets

Goal: Classifying jets in different prong classes (N=1, 2, 3, ,4b, 4q, 6, 8: in total 7 classes).



Tagging Extreme Jets

Goal: Classifying jets in different prong classes (N=1, 2, 3, ,4b, 4q, 6, 8, in total 7 classes).

Data Source:

- Simulator: MG5+pythia8+delphes (ATLAS)
- Conditions:
  - Jet pT 1000-1200 GeV, jet Mass 300-700 GeV
  - Truth match dR cone of 1.2 for q and b
  - Samples have uniform Mass and pT distribution

Data Format:

- List of towers associated with the jets.
- Maximum length of the lists is 230.
- Each tower contains three variables: pT, eta, and phi.
- From the list of towers→ vector of HL variables of size 16.

```
Tower pt,eta,phi= 0.3987 -1.8823 4.91169
Tower pt,eta,phi= 0.5077 -1.7792 4.79809
Tower pt,eta,phi= 1.7711 -1.7441 0.0002
Tower pt,eta,phi= 3.0643 -1.6887 4.84469
Tower pt,eta,phi= 0.6896 -1.682 5.06559
Tower pt,eta,phi= 4.406 -1.6716 4.81589
Tower pt,eta,phi= 5.2555 -1.6513 4.62029
Tower pt,eta,phi= 1.6739 -1.6364 4.97079
Tower pt,eta,phi= 0.4357 -1.5764 4.96129
Tower pt,eta,phi= 0.6812 -1.5609 4.57889
```

Baselines:

- **Mass + HL (tower)** network: fully connected network taking Mass and 16 high level features (HL) computed from the towers as input.

- Particle Flow Networks (**PFN**) *(Komiske et al., 2019)*: Neural networks with no attention mechanism taking towers as input.

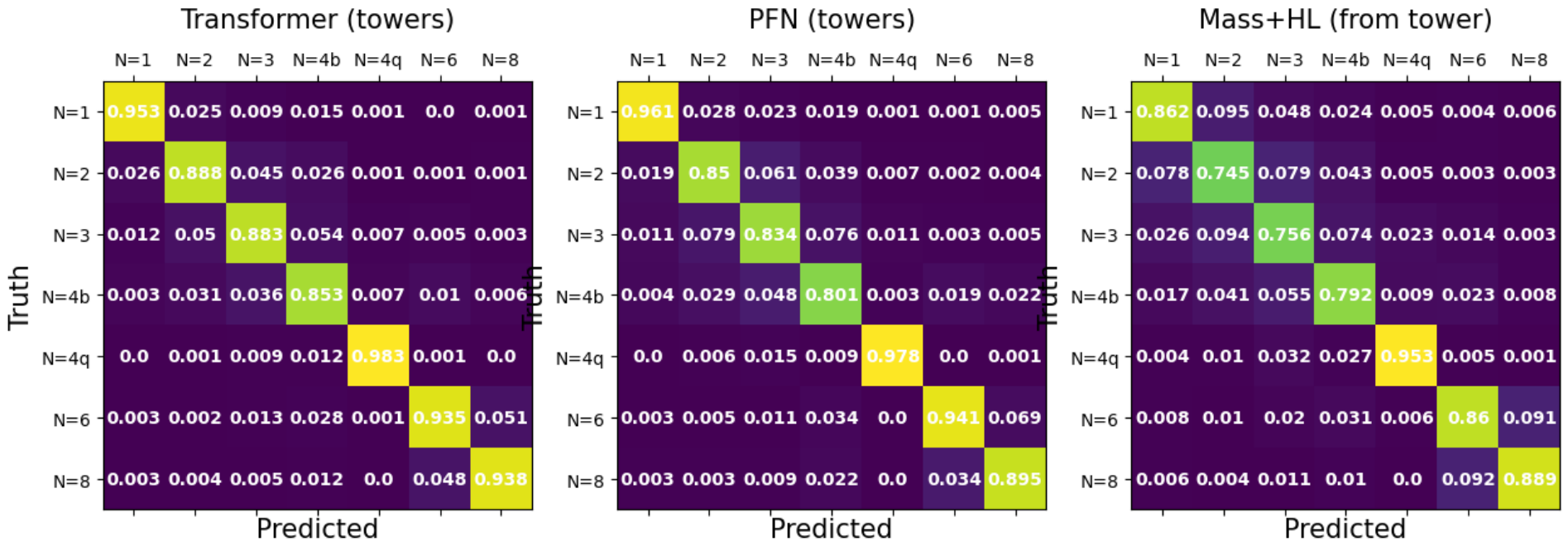
Transformer model is more effective in distinguishing different prong classes compared to other deep neural network models.

	Parameter count	Top 1 acc
Mass+HL (Tower)	3,270,119	$84.14 \pm 0.45\%$
PFN (Tower)	1,205,895	$89.42 \pm 0.49\ \%$
Transformer (Tower)	1,388,807	$91.25 \pm 0.65\%$

Transformer model is more effective in distinguishing different prong classes compared to other deep neural network model.

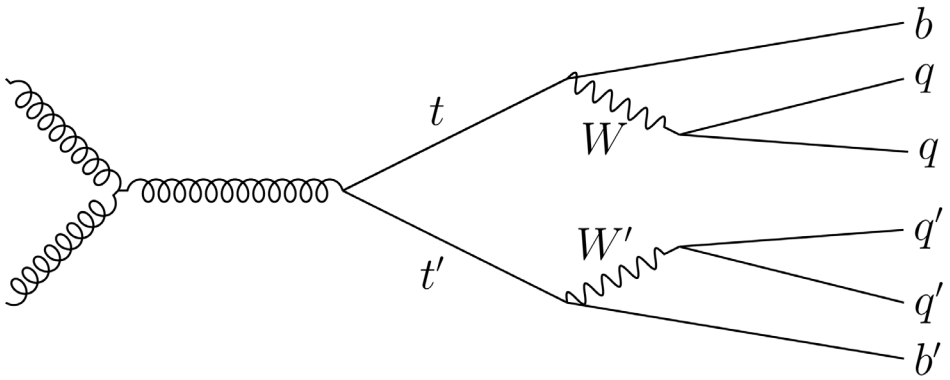
Confusion matrices

Transformer model is more effective in distinguishing different prong classes compared to other deep neural network model.



Spatter Jet-Parton Matching in LHC Top Quark Decays

- Primary (all-hadronic) decay channel produces six particles: two qqb triplets with opposite charge.
- After these particles are produced, they are propagated and measured by the detector as *jets*.
- Along with the jets from each of the particles, there may be additional jets from other decay products.



$$\{j_1, j_2, j_3, j_4, j_5, j_6, j_7, j_8\}$$

Match Jets to Particle Labels

$$\{b, q', \emptyset, q', b', \emptyset, q, q\}$$

Garbage Jets

This is a difficult matching problem: Observing the jets from the detector, can you determine which jets belong to which particles?
Effective matching requires exploiting the symmetries in this problem!

Symmetry Top Quark Symmetries

We notice an opportunity to exploit symmetries to simplify the matching task.
The following particle targets are equivalent.

$$\boldsymbol{q_1 q_2} b q_1' q_2' b' \leftrightarrow \boldsymbol{q_2 q_1} b q_1' q_2' b'$$
$$q_1 q_2 b \boldsymbol{q_1' q_2'} b' \leftrightarrow q_1 q_2 b \boldsymbol{q_2' q_1'} b'$$

We call these light quark symmetries – the q terms can be freely rearranged. We will handle these later.

$$\underset{\mathcal{T}_1}{q_1 q_2 b} \underset{\mathcal{T}_2}{\boldsymbol{q_1' q_2' b'}} \leftrightarrow \underset{\mathcal{T}_2}{\boldsymbol{q_1' q_2' b'}} \underset{\mathcal{T}_1}{q_1 q_2 b}$$

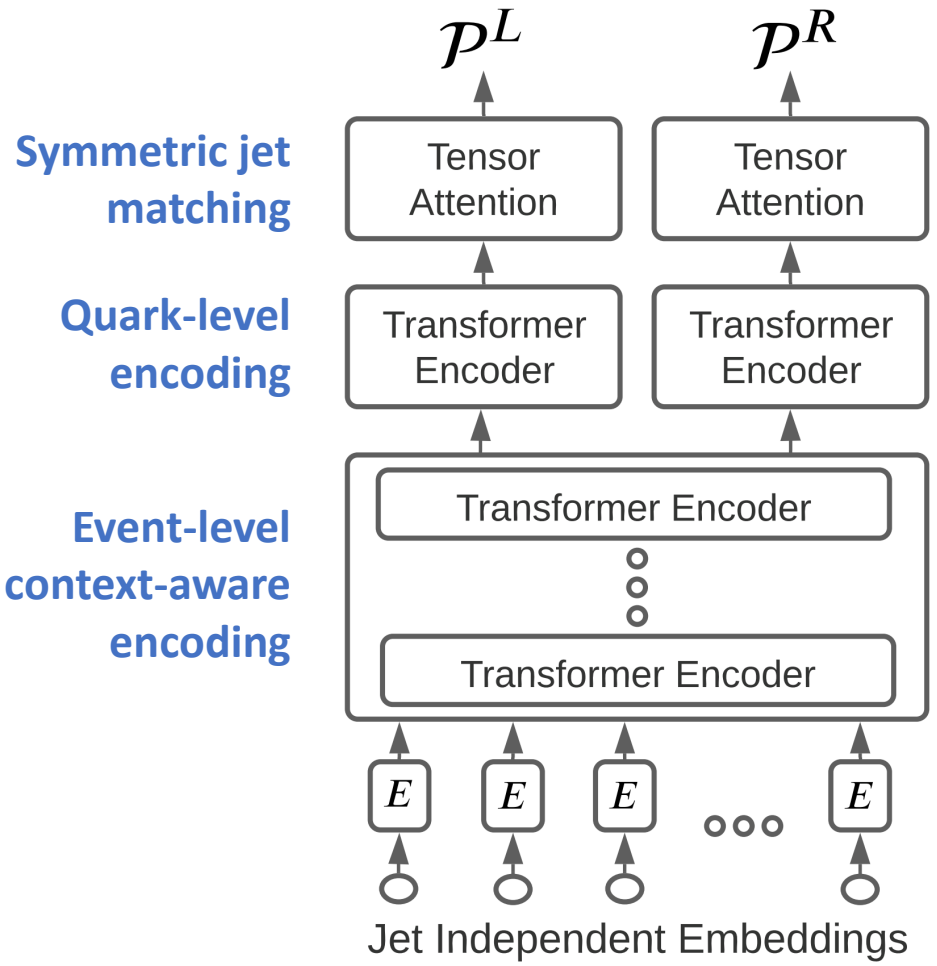
We call this top symmetry – the two top triplets cannot be separated due to charge-symmetry.

We encode the top-symmetries during training by allowing the network to fit to either of the two possibilities.

$$\min\left\{\mathcal{L}(\mathcal{P}^L, \mathcal{T}_1, \mathcal{P}^R, \mathcal{T}_2), \mathcal{L}(\mathcal{P}^L, \mathcal{T}_2, \mathcal{P}^R, \mathcal{T}_1)\right\}$$

Spatter Architecture

To avoid a combinatorial explosion by trying to match all 6 particles at the same time, we instead try and match each top quark (triplet) individually.



We output distributions predicting the likely triplets associated with each particle.

We employ attention in several sections within our network for context-aware learning.

We accept as input an unsorted list of jet 4-momentum vectors.

Symmetry Input Permutation Invariance

- First challenge is to match arbitrary sets (unordered lists) of jets to a varying number of targets.
- Any architecture we choose must be invariant to the order of jets.
- Attention (Transformers) lends itself naturally to encoding sets of objects.

Pairwise similarity provides context-aware permutation invariant weights.

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

The transformer encoder is invariant!

We can use transformers as **general permutation invariant encoders** for sets of objects.

Symmetry Light Quark Symmetries

We need to predict two three-way joint distributions that abide by our light-quark symmetry.

Introduce generalization of dot-product attention: **Symmetric Tensor Attention**

1. We store a trainable order-3 tensor $\boldsymbol{\theta}$ which does not conform to any symmetries.
2. We then symmetrize it into tensor $\boldsymbol{S}$ which enforces index symmetry along the first two indices.
3. We perform generalized dot-product on some list of input vectors $\boldsymbol{X}$ to generate a predictive output $\mathbf{O}$.
4. Finally, we create a predictive distribution $\boldsymbol{P}$ by normalizing $\boldsymbol{O}$.

$$S^{ijk} = \frac{1}{2}\left(\theta^{ijk} + \theta^{jik}\right)$$

$$O^{ijk} = X_n^i X_m^j X_l^k S^{nml}$$

$$\mathcal{P}(i,j,k) = \frac{\exp O^{ijk}}{\sum_{i,j,k} \exp O^{ijk}}$$

Spatter Results

- We compare *Spatter* to a classical method for jet-parton matching based on the χ^2 probability of assignments.
- *Spatter* only needs to match the top-quarks while the χ^2 method needs to match entire events.
- *Spatter* **reduces the runtime** from $O(N^6)$ to $O(N^3)$ while **increasing accuracy** by $\sim 30\%$.

	χ^2 Method			SPA$t\bar{t}$ER		
N_{jets}	ϵ^{event}	ϵ_2^{top}	ϵ_1^{top}	ϵ^{event}	ϵ_2^{top}	ϵ_1^{top}
6	61.8%	65.0%	24.2%	80.7%	84.1%	56.7%
7	40.8%	50.4%	24.6%	66.8%	75.7%	56.2%
≥ 8	23.2%	35.5%	20.2%	52.3%	66.2%	52.9%
Inclusive	**37.7%**	**47.0%**	**23.0%**	**63.7%**	**73.5%**	**55.2%**

Runtime on 8 jet events

χ^2 : 369 *ms* per event

Spatter : 4.4 *ms* per event

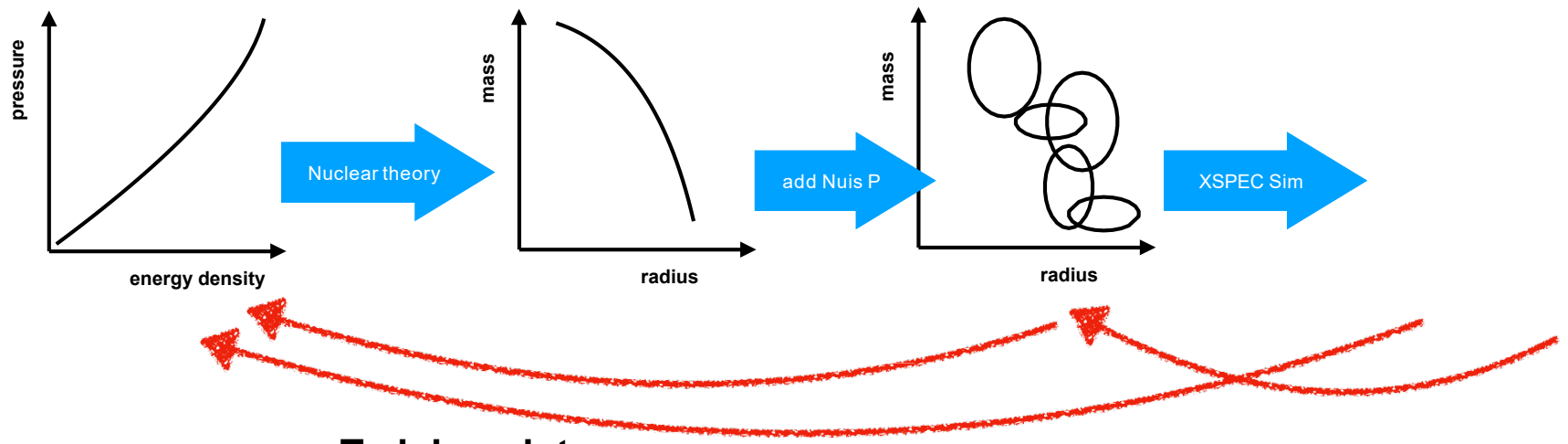
The problem

Equation of State (4 params)

Mass-Radius relation (2 params: M,R)

Star parameters
2 params: (mass, radius)
3 nuis params (dist, temp, dust)

Neutron Star X-ray data
(1024 chan,80% empty)



Training data

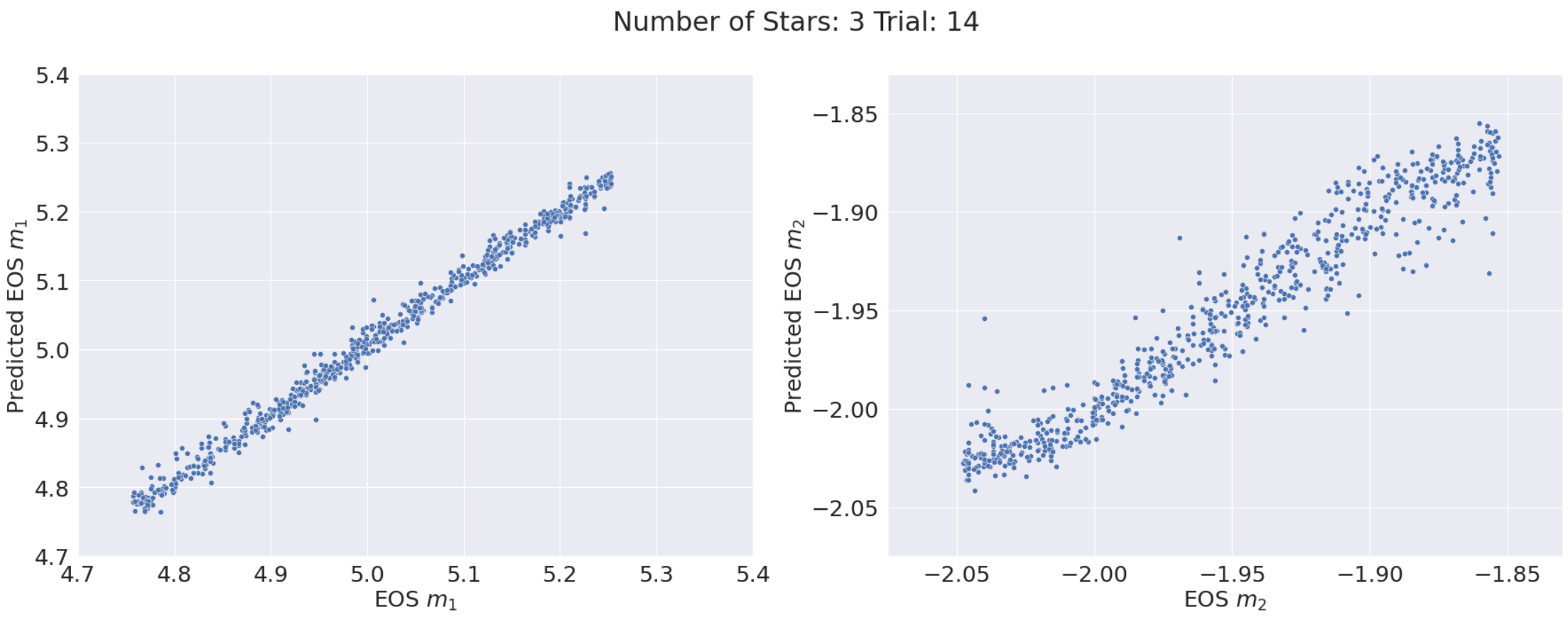
Fixed EOS, sample of (M,R) pairs
For each M,R pair, add 3 nuisance param
generate sample spectra

Inference

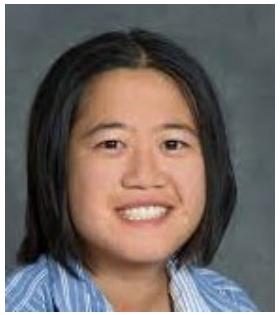
End-to-end: spectra -> EOS
Also might try: spectra-> star
star -> EOS

MLLR method: ttps://arxiv.org/abs/2002.04699

Prediction of EOS coefficients from 3 Stars (M,R)



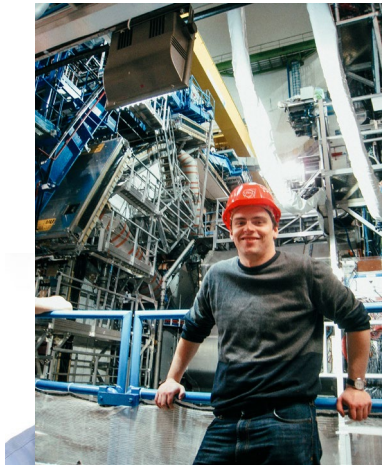


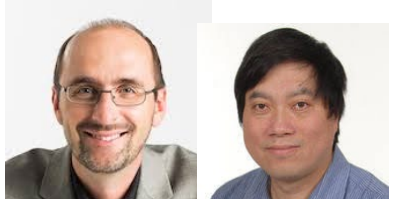








THANK YOU