Lecture Notes V – Residual and Convolutional Neural Networks, Transformers and attention

Marina Meilă
mmp@uwaterloo.ca

With Thanks to Pascal Poupart & Gautam Kamath
Cheriton School of Computer Science
University of Waterloo

February 8, 2026

Convolutional networks (Convnets)

Residual networks (Resnets)

Some (convolutional) neural network breakthroughs

Transformer networks

Reading HTF Ch.: 11.3 Neural networks, Murphy Ch.: (16.5 neural nets), Bach Ch.: –, Deep Learning Book (Goodfellow, Bengio, Courville) 6.1-4, ResNet 7.6, ConvNet 9., Autoencoders 14.1, Dive Into Deep Learning 4.1-4.3.

ConvNets – Convolutional Networks

- **discrete convolution** let $f, g : \mathbb{Z} \to \mathbb{R}$
  $\mathbb{Z} =$ all integers

$$(f * g)(t) = \sum_{i \in \mathbb{Z}} f(t-i)g(i) \tag{1}$$

- convolution as **Toeplitz** matrix vector multiplication
- in ConvNets, $\mathbb{Z}$ is replaced by $1 : m$, f is **padded with 0's**
  - g is a (smoothing) kernel
  - i.e. $g(i) = g(-i) > 0$ and $|\operatorname{supp} g| = 2s + 1 \ll m$, $\sum_i g(i) = 1$
- Convolutional layer $f \leftarrow x$ input, $g \leftarrow w$ weights, s output

$$s(t) = \sum_{i=t-s}^{t+s} w_i s(t-i) \tag{2}$$

- Pooling
  a symmetric function like max, $\sum$, ...

Discrete convolution

-
- Discrete convolution

$$y(i) = \sum_{t=-\infty}^{\infty} x(t)w(i-t)$$

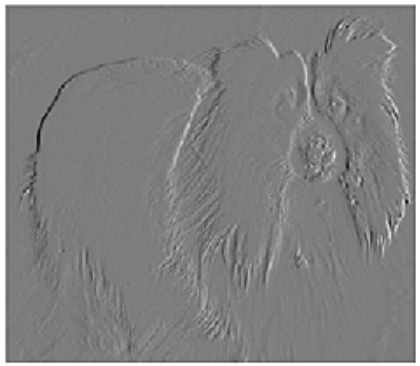
- Multidimensional convolution

$$y(i,j) = \sum_{t_1=-\infty}^{\infty} \sum_{t_2=-\infty}^{\infty} x(t_1,t_2)w(i-t_1, j-t_2)$$

Example: Edge Detection

- Consider a grey scale image
- Detect vertical edges: $y(i,j) = x(i,j) - x(i-1,j)$

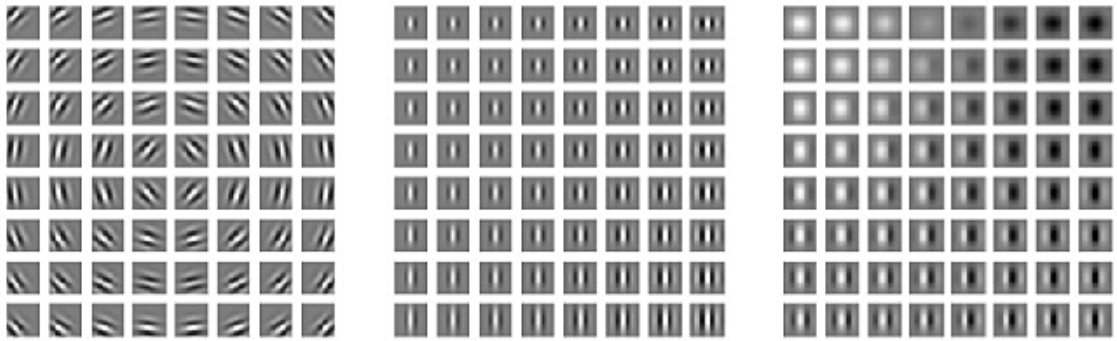
$$w(i-t_1, j-t_2) = \begin{cases} 1 & t_1 = i, t_2 = j \\ -1 & t_1 = i-1, t_2 = j \\ 0 & \text{otherwise} \end{cases}$$



Convolution for feature extraction

Gabor filters

- Gabor filters: common feature maps inspired by the human vision system



- Weights:

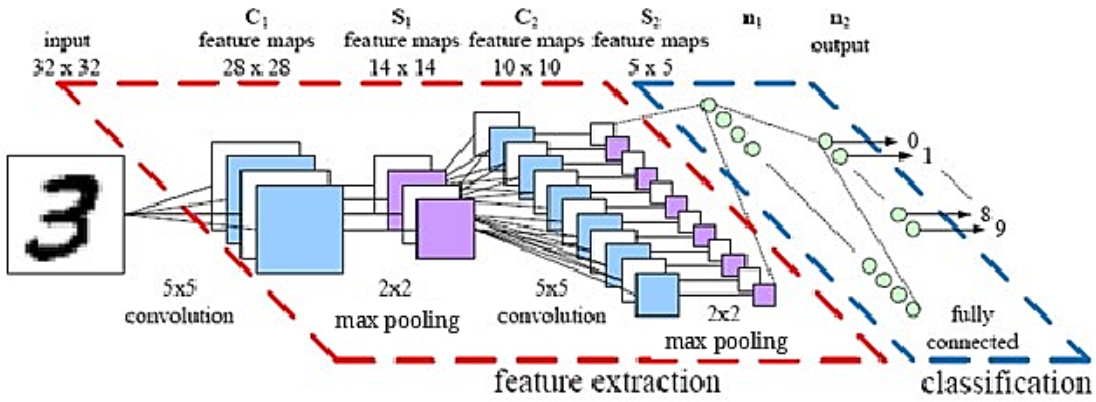
  Grey: zero White: positive Black: negative

Pooling

- Pooling: **commutative** mathematical operation that combines several units
- Examples:
  - max, sum, product, average, Euclidean norm, etc.
- Commutative property (order does not matter):

$$\text{Ex.: } \max(a, b) = \max(b, a)$$

Example: Digit Recognition

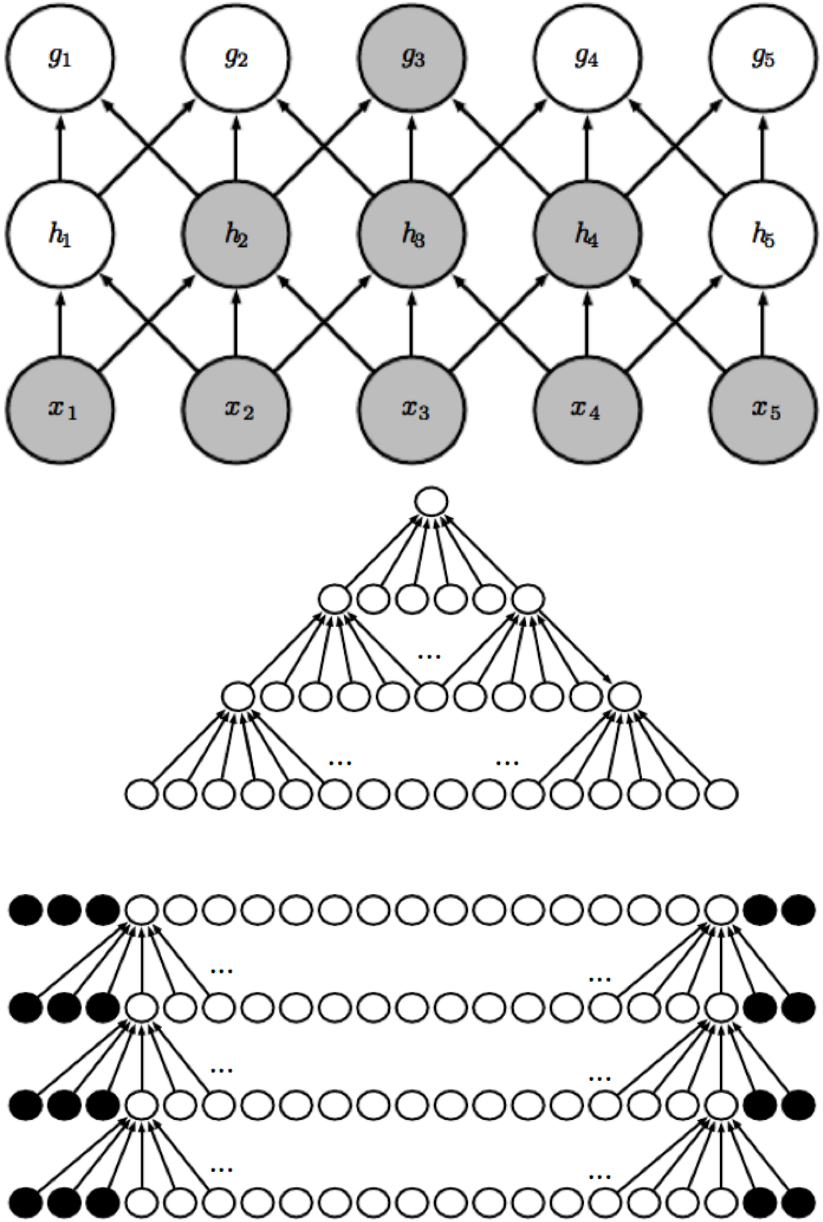


Parameters

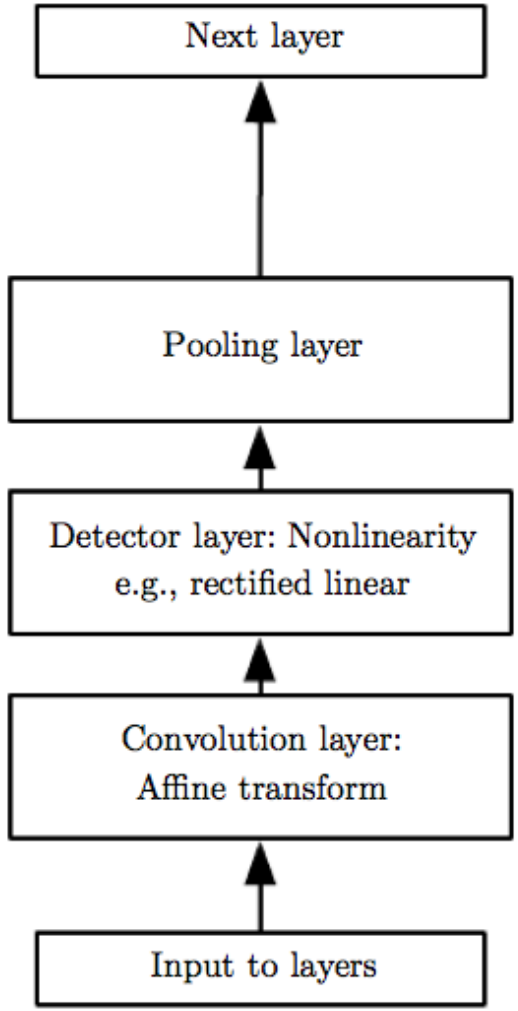
- **# of filters**: integer indicating the # of filters applied to each window.
- **kernel size:** tuple (width, height) indicating the size of the window.
- **Stride:** tuple (horizontal, vertical) indicating the horizontal and vertical shift between each window.
- **Padding:** “valid” or ”same”. Valid indicates no input padding. Same indicates that the input is padded with a border of zeros to ensure that the output has the same size as the input.

Benefits

- Sparse interactions
  - Fewer connections
- Parameter sharing
  - Fewer weights
- Locally equivariant representation
  - Locally invariant to translations
  - Handle inputs of varying length



from www.deeplearningbook.org Chapter 9



Training

- Convolutional neural networks are trained in the same way as other neural networks
  - E.g., backpropagation
- Weight sharing:
  - Combine gradients of shared weights into a single gradient

Architecture design

- What is the preferred filter size?
- VGG (Visual Geometry Group at Oxford, 2014): stack of small filters is often preferred to a single large filter
  - Fewer parameters
  - Deeper network
- Picture

Resnets – Residual networks

Idea What is the "simplest" input-output function? $f_0(x) = x$

- Hence, a NN layer should learn the difference w.r.t. identity f_0

$$x^{(l+1)} = B^{(l)}\phi(W^{(l)}x^{(l)}) + x^{(l)} \tag{3}$$

Here we must have $m_{l+1} = m_l$
Generalization DenseNet

- Layer l gets inputs from $l-1, l-2, \ldots$. These inputs are in parallel hence $m_l, m_{l-1}, \ldots$ need not be equal

Residual Networks

- Problem: very deep networks can perform worse than shallower networks (due to local optima & other stationary non-optimal points)
- Solution [He et al., 2015]: introduce **residual connections** (a.k.a. skip connections) to make blocks optional
- Picture:

Acoustic Modeling in Speech Recognition

TABLE III

A comparison of the Percentage Word Error Rates using DNN-HMMs and GMM-HMMs on five different large vocabulary tasks.

task	hours of training data	DNN-HMM	GMM-HMM with same data	GMM-HMM with more data
Switchboard (test set 1)	309	18.5	27.4	18.6 (2000 hrs)
Switchboard (test set 2)	309	16.1	23.6	17.1 (2000 hrs)
English Broadcast News	50	17.5	18.8	
Bing Voice Search (Sentence error rates)	24	30.4	36.2	
Google Voice Input	5,870	12.3		16.0 (>>5,870hrs)
Youtube	1,400	47.6	52.3	

Architecture of a DNN-HMM hybrid system

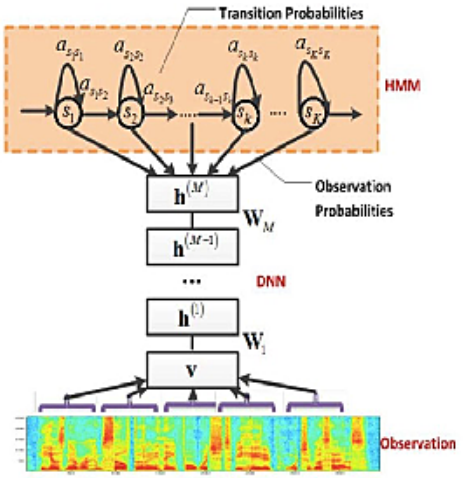
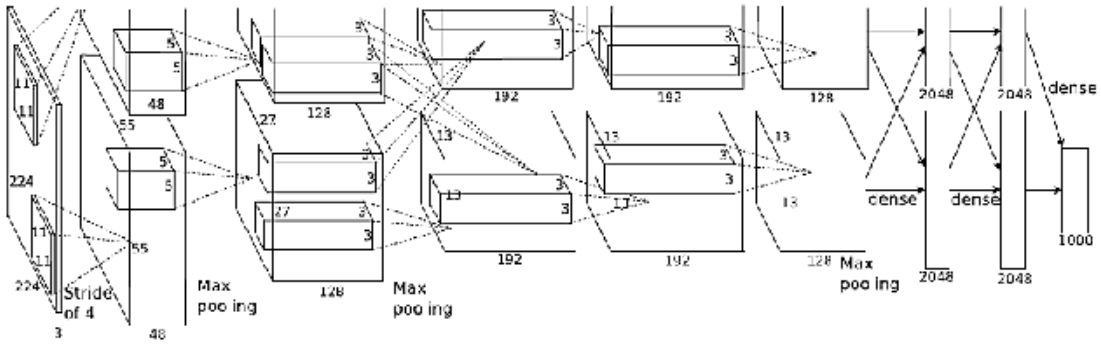


Image Recognition

- Convolutional Neural Network
  - With rectified linear units and dropout
  - Data augmentation for transformation invariance



ImageNet Breakthrough

- Results: ILSVRC-2012
- From Krizhevsky, Sutskever, Hinton

Model	Top-1 (val)	Top-5 (val)	Top-5 (test)
SIFT + FVs [7]	—	—	*26.2%*
1 CNN	40.7%	18.2%	—
5 CNNs	38.1%	16.4%	**16.4%**
1 CNN*	39.0%	16.6%	—
7 CNNs*	36.7%	15.4%	**15.3%**

Table 2: Comparison of error rates on ILSVRC-2012 validation and test sets. In *italics* are best results achieved by others. Models with an asterisk* were "pre-trained" to classify the entire ImageNet 2011 Fall release. See Section 6 for details.

ImageNet Breakthrough

- From Krizhevsky, Sutskever, Hinton

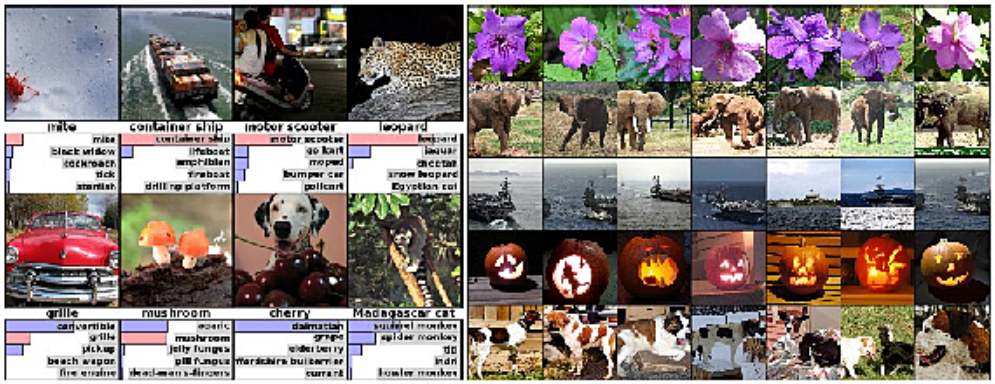


Figure 4: (**Left**) Eight ILSVRC-2010 test images and the five labels considered most probable by our model. The correct label is written under each image, and the probability assigned to the correct label is also shown with a red bar (if it happens to be in the top 5). (**Right**) Five ILSVRC-2010 test images in the first column. The remaining columns show the six training images that produce feature vectors in the last hidden layer with the smallest Euclidean distance from the feature vector for the test image.

Transformer networks: Why we need attention

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

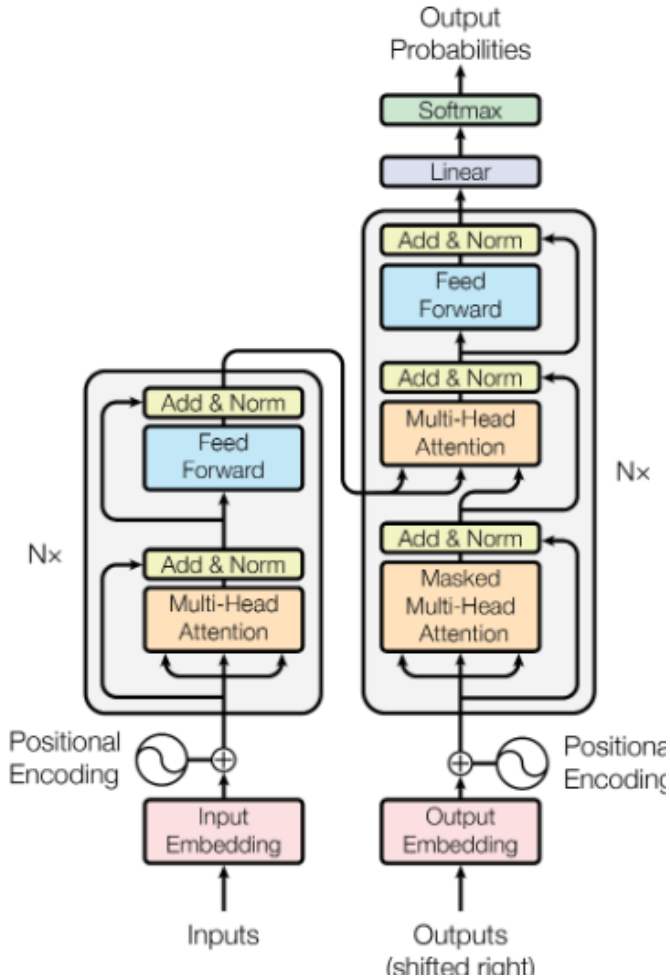
We propose a new simple network architecture, the Transformer,based solely on attention mechanisms, dispensing with recurrence and convolutionsentirely.

Wir schlagen eine neue einfache Netzwerkarchitektur, den Transformer, vor, die ausschließlich auf Aufmerksamkeitsmechanismen basiert und auf Wiederholung und Faltung vollständig verzichtet.

我们提出了一种新的简单的网络架构--Transformer，完全基于注意力机制，摒弃了递归和卷积。

- mapping sequences to sequences (structured prediction)
- both long and short range dependencies
- range depends on input sequence

Basic architecture



- inputs $x_1, x_2, \ldots$, outputs $y_1, y_2, \ldots$ from discrete set (e.g. words in English, Chinese)
- continuous internal representations
- **embedding** modules map input or output space to continuous representations (prelearned)

- recurrence/auto-regression y_t depends on $x_{1:t+k}$ and $y_{1:t-1}$

- **encoder, decoder, encoder-decoder** modules (which use attention)

How to implement attention

- **queries, keys** and **values**
- all **learned**
- Idea: query q matches key k results in selecting the corresponding value v
- q depends on current context, k depends on v
- $q, k \in \mathbb{R}^{d_k}$

- Q, K, V matrices of queries, keys, values

$$A(Q, K) = \text{softmax}(\frac{1}{\sqrt{d_k}} QK^T) \tag{4}$$

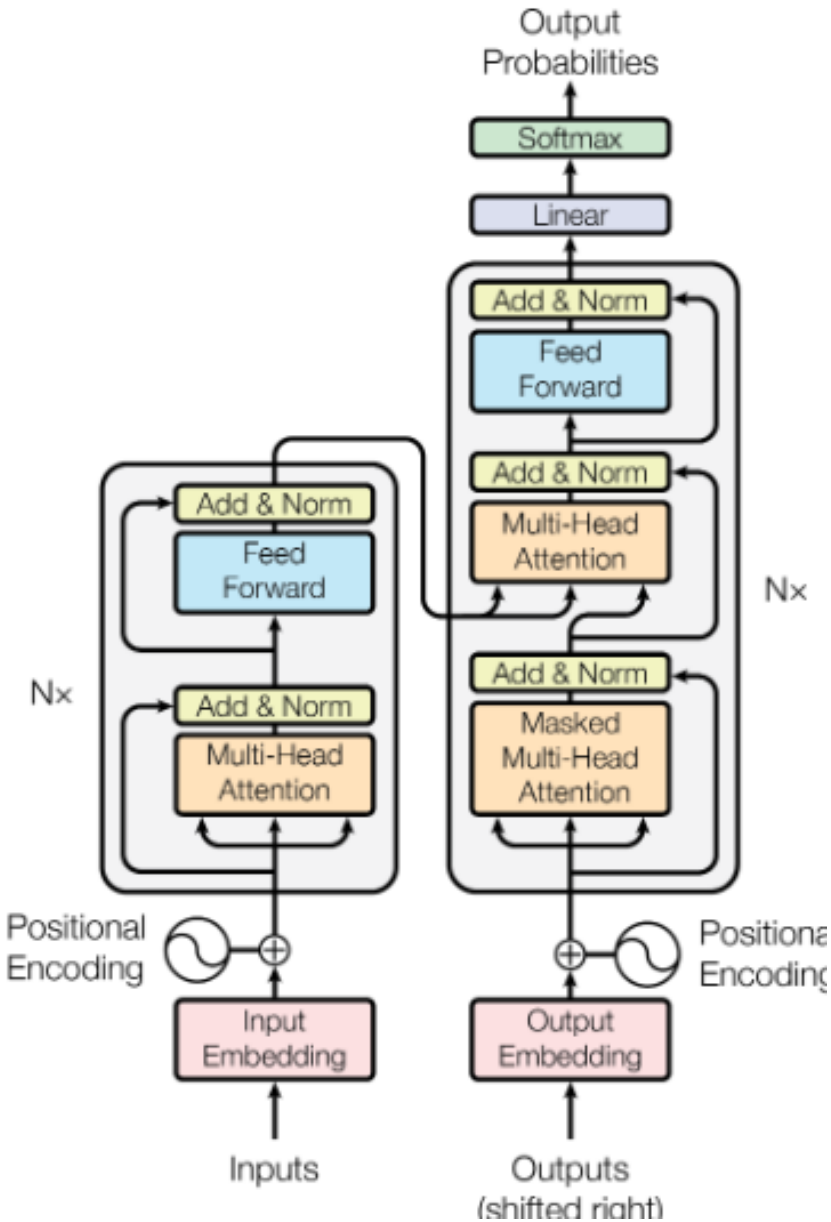
- $A_{q:}$ selects value v for the best matching key for each q

Transformer architecture

- D,E,DE modules: each have $N = 6$ layers of Attention + Feed-forward (FFW) networks of same $d = 512$
- FFW, A are ResNets
- FFW is $W_2 \max(W_1 x, 0)$, $W_{1,2}$ with identical rows
- A is **multihead attention**
  - $h = 8$ parallel attention layers, concatenated

- advantages – implements long distance dependencies with fixed (small) number layers, and parallel computations

Attention mechanism in Transformer



- encoder-decoder
  - queries from previous decode layer
  - keys, values from current encoder output
- encoder – self-attention (=previous encoder layer)
- decoder
  - self-attention, **masked**
  - only from outputs before current step