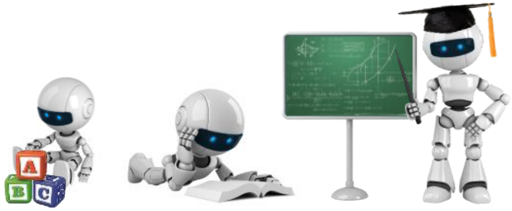
USC Viterbi
School of Engineering
*Department of Computer Science*

# CSCI 544
# Applied Natural Language Processing

Mohammad Rostami
USC Computer Science Department

Portions from Slides by Wang Yue and Giuseppe Attardi



# Recurrent Neural Networks

- We can consider NL data as sequential data points, where the current word depends on the previous words in the sequence:

1 2 3 4 5 6 7 8 9 10 11

- Ex: Today, I want to play football and then watch a movie.
- Learning representations by back-propagating errors, 1986

- Core Idea: the function approximator can receive the input word by word such that its output depend on the history, i.e., relying on a notion of memory

FNN $y = f(x)$

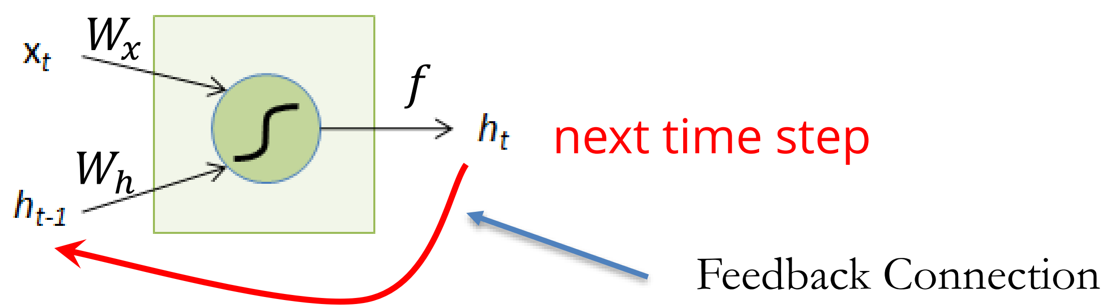
RNN $y_t = f(x_t, h_{t-1})$

output input memory

# Recurrent Neural Networks

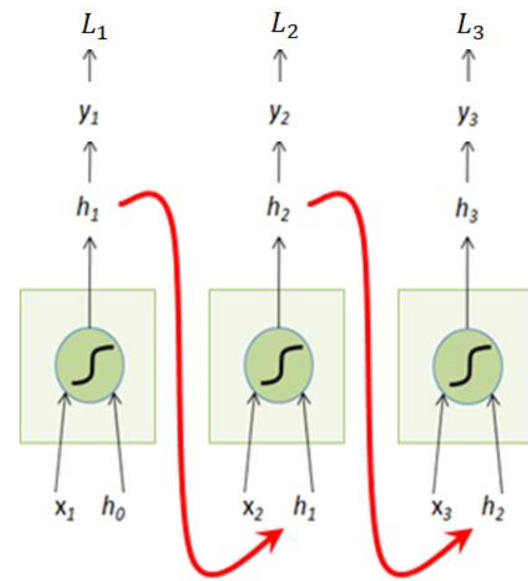
- Equipping perceptron with memory

> - $x_t$: Input at time t
> - $h_{t-1}$: State at time t-1




$$h_t = f(W_x x_t + W_h h_{t-1}), W_x \in \mathbb{R}^{M*N}, W_h \in \mathbb{R}^{H*H}$$

- Unfolding RNN
- We can make the unit multi-layer




Figures adopted from slides by Arun Mallya

# Recurrent Neural Networks

- The weight matrices are shared across time

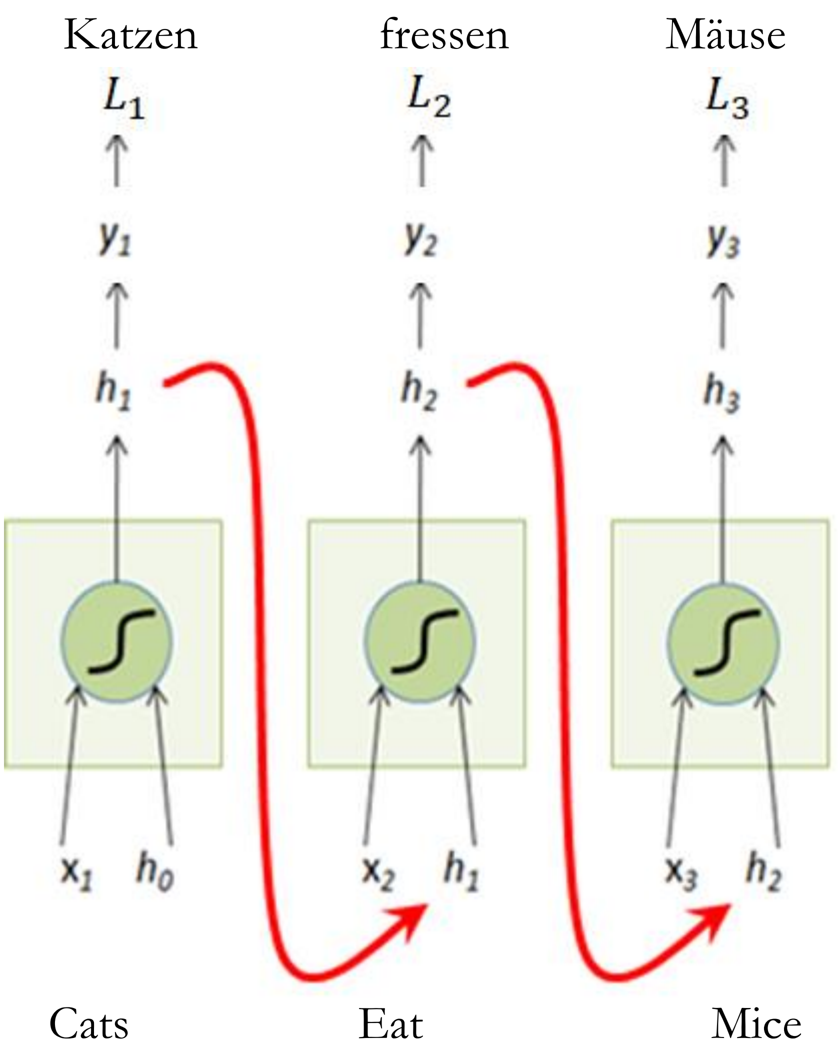
- multi-output

$$L = L_1 + L_2 + L_3$$

$$L = \sum_{t=1}^{T} L_t$$

$$y_t = f(x_t, h_{t-1})$$
$$L_t = l(y, \hat{y}_t)$$

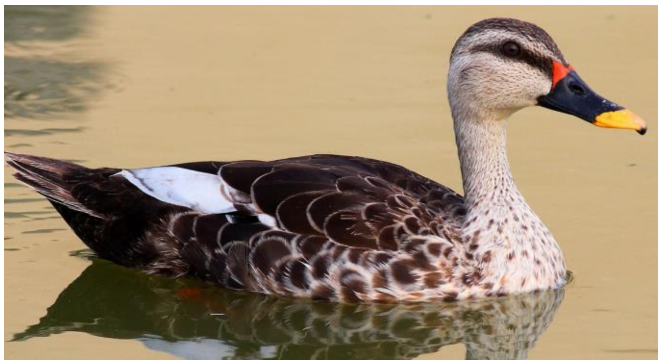
- Sequential processing
- Hard to train
- Long range dependencies




Figures adopted from slides by Arun Mallya

# Attention Mechanism

## Images



Indian Spot-billed Duck

Mandarin Duck

## Text

### Deep Learning

- Table of Contents
- Acknowledgements
- Notation
- 1 Introduction
- Part I: Applied Math and Machine Learning Basics
  - 2 Linear Algebra
  - 3 Probability and Information Theory
  - 4 Numerical Computation
  - 5 Machine Learning Basics
- Part II: Modern Practical Deep Networks
  - 6 Deep Feedforward Networks
  - 7 Regularization for Deep Learning
  - 8 Optimization for Training Deep Models
  - 9 Convolutional Networks
  - 10 Sequence Modeling: Recurrent and Recursive Nets
  - 11 Practical Methodology
  - 12 Applications
- Part III: Deep Learning Research
  - 13 Linear Factor Models
  - 14 Autoencoders
  - 15 Representation Learning
  - 16 Structured Probabilistic Models for Deep Learning
  - 17 Monte Carlo Methods
  - 18 Confronting the Partition Function
  - 19 Approximate Inference
  - 20 Deep Generative Models

# Convolutional Neural Networks: Motivation

- Some differentiating patterns are local

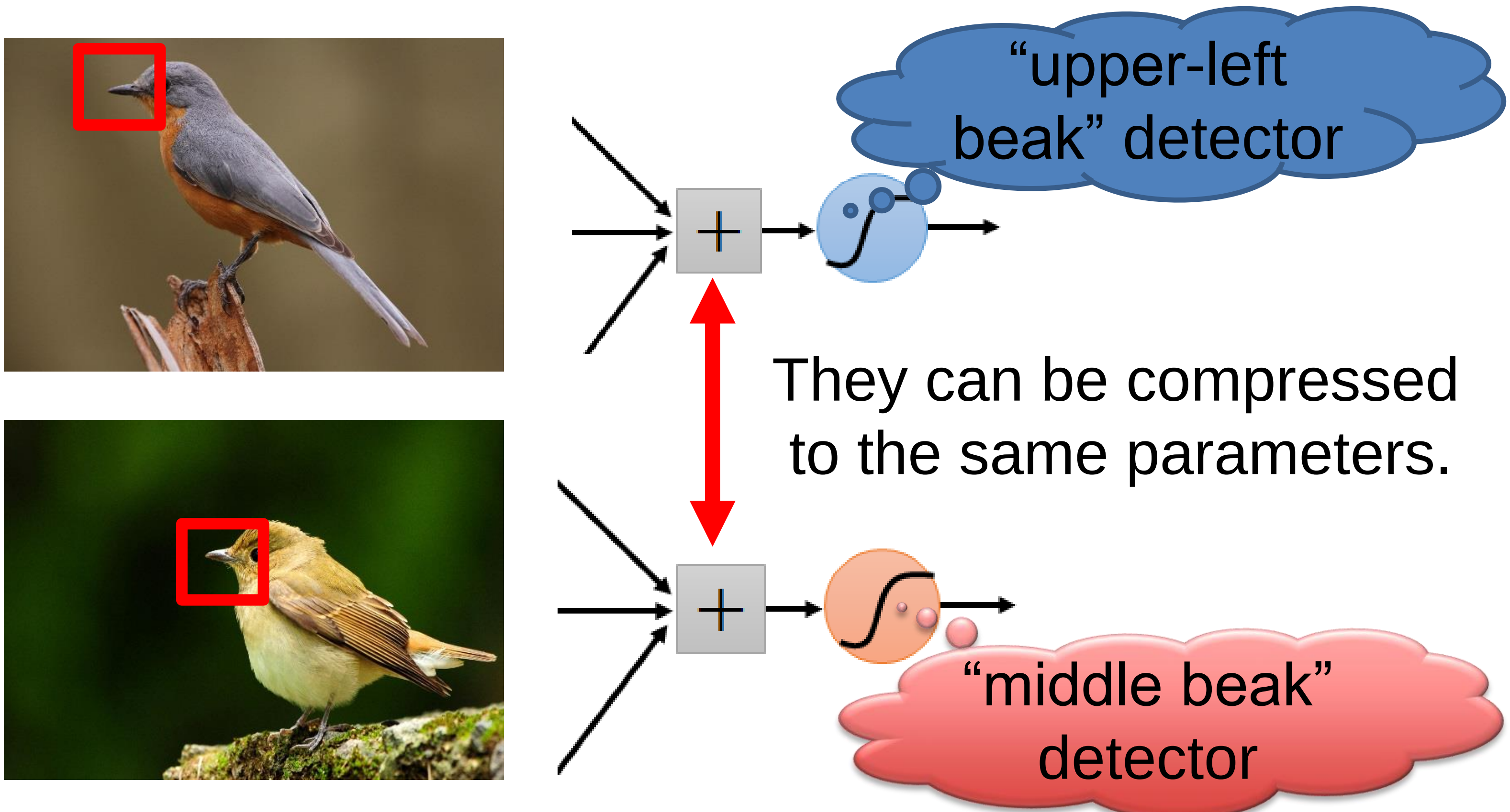
 - What is a bird?

- We should attend to smaller locations to perform a task

# Convolutional Neural Networks: Motivation

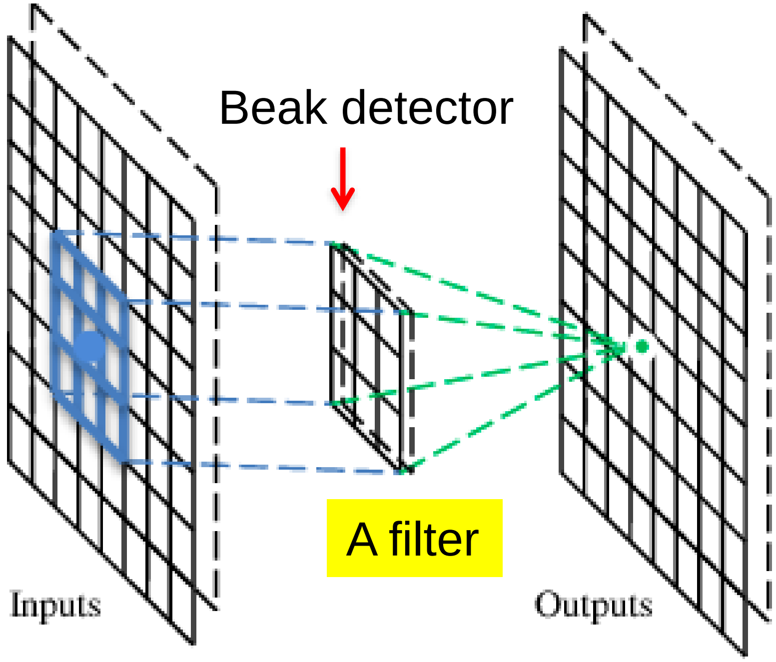
Local pattern appears in different places.

Solution: training a lot of “local” detectors and move around each detector to make the model spatially invariant.



# Convolutional Layer

A CNN is a neural network with some convolutional layers (and some other layers). A convolutional layer has a number of filters that does convolutional operation.



# Convolutional Layer

**These are the network parameters to be learned.**

| | | | | | |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 |

6 x 6 image

Filter 1

| | | |
|---|---|---|
| 1 | -1 | -1 |
| -1 | 1 | -1 |
| -1 | -1 | 1 |

Filter 2

| | | |
|---|---|---|
| -1 | 1 | -1 |
| -1 | 1 | -1 |
| -1 | 1 | -1 |

⋮ ⋮

$$F \circ I(x,y) = \sum_{j=-N}^{N} \sum_{i=-N}^{N} F(i,j) I(x+i, y+j)$$

Each filter detects a small pattern (3 x 3).

# Convolutional Layer

stride=1

| | | | | | |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 |

6 x 6 image

| | | |
|---|---|---|
| 1 | -1 | -1 |
| -1 | 1 | -1 |
| -1 | -1 | 1 |

Filter 1

| | | | |
|---|---|---|---|
| 3 | -1 | -3 | -1 |
| -3 | 1 | 0 | -3 |
| -3 | -3 | 0 | 1 |
| 3 | -2 | -2 | -1 |

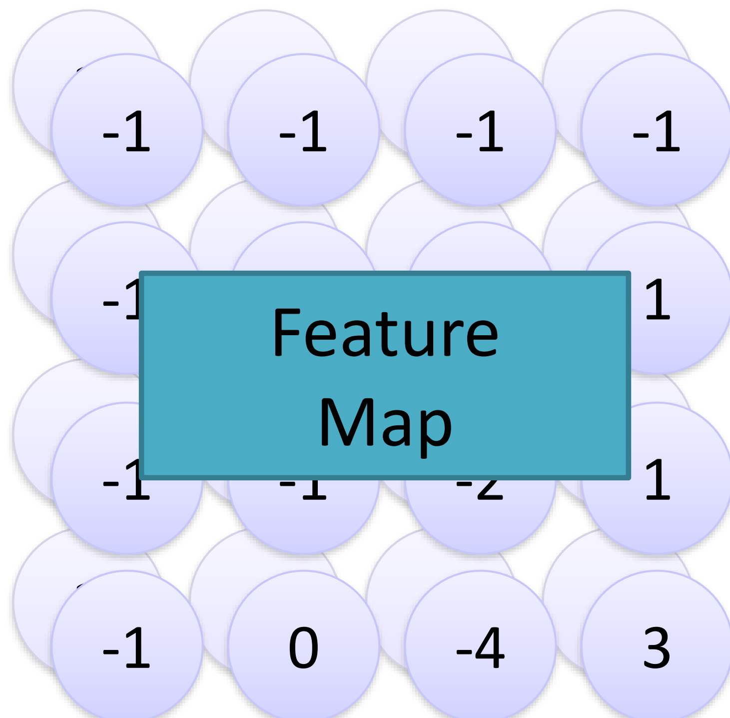
# Convolutional Layer

stride=1

| | | | | | |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 |

6 x 6 image

Repeat this for each filter




Two 4 x 4 images
Forming 2 x 4 x 4 matrix

# Convolutional Layer

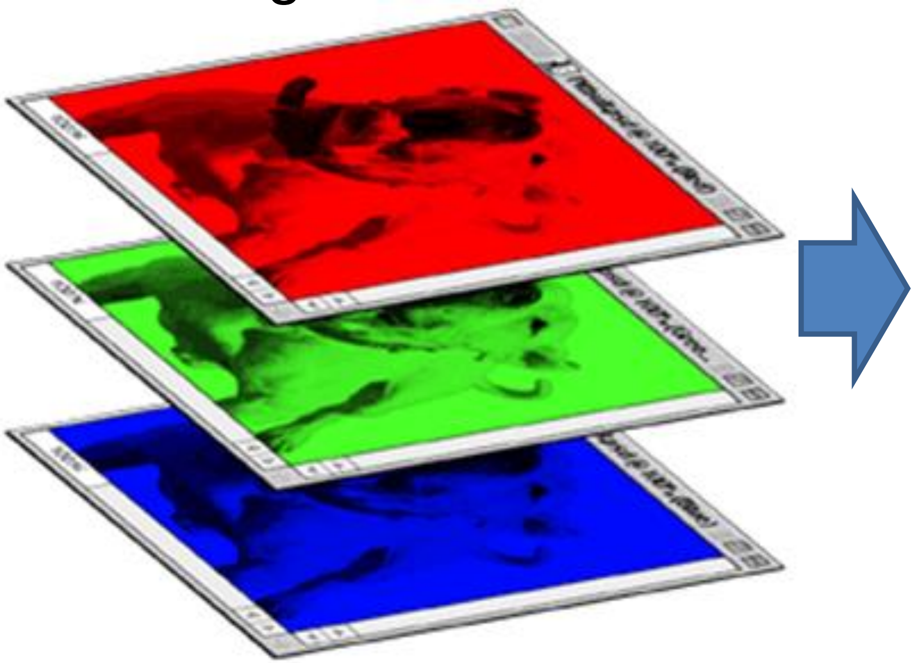
| | | |
|---|---|---|
| 1 | -1 | -1 |
| -1 | 1 | -1 |
| -1 | -1 | 1 |

Filter 1

| | | |
|---|---|---|
| -1 | 1 | -1 |
| -1 | 1 | -1 |
| -1 | 1 | -1 |

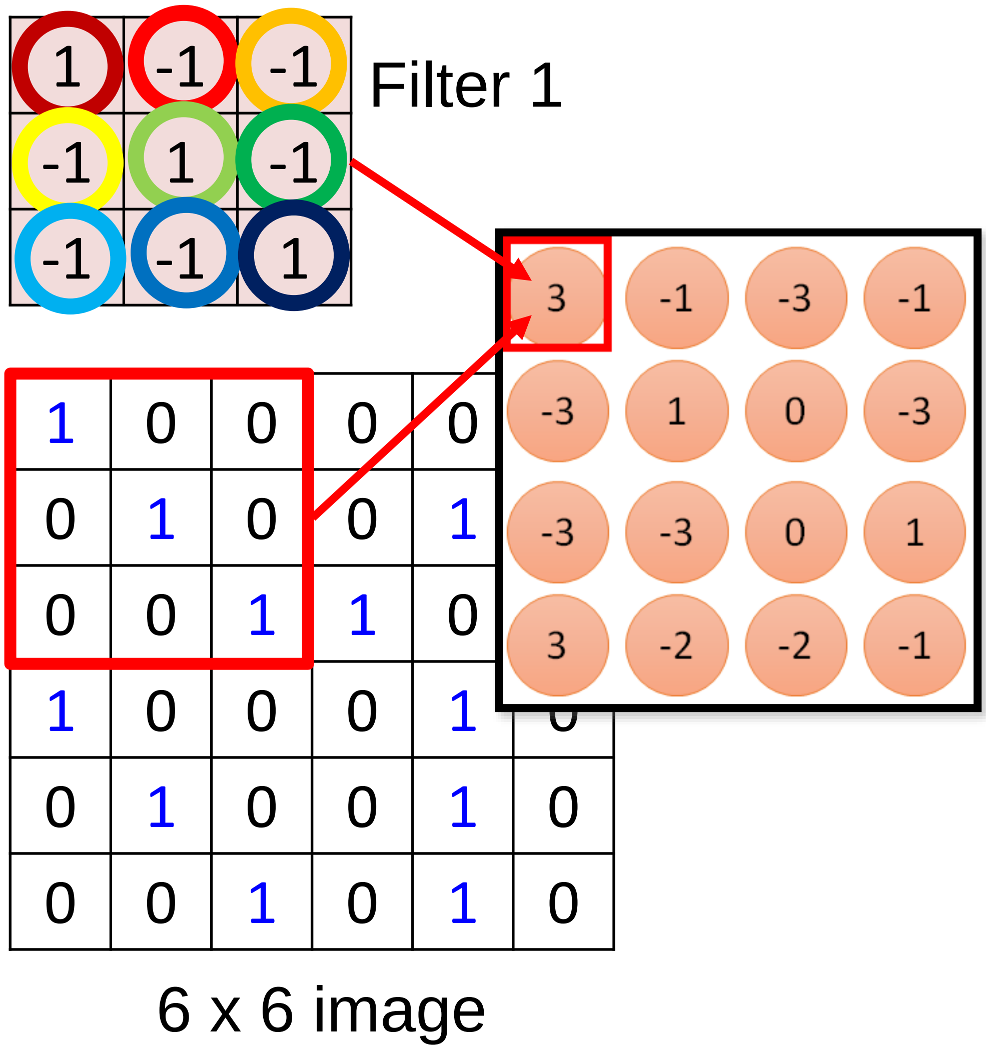
Filter 2

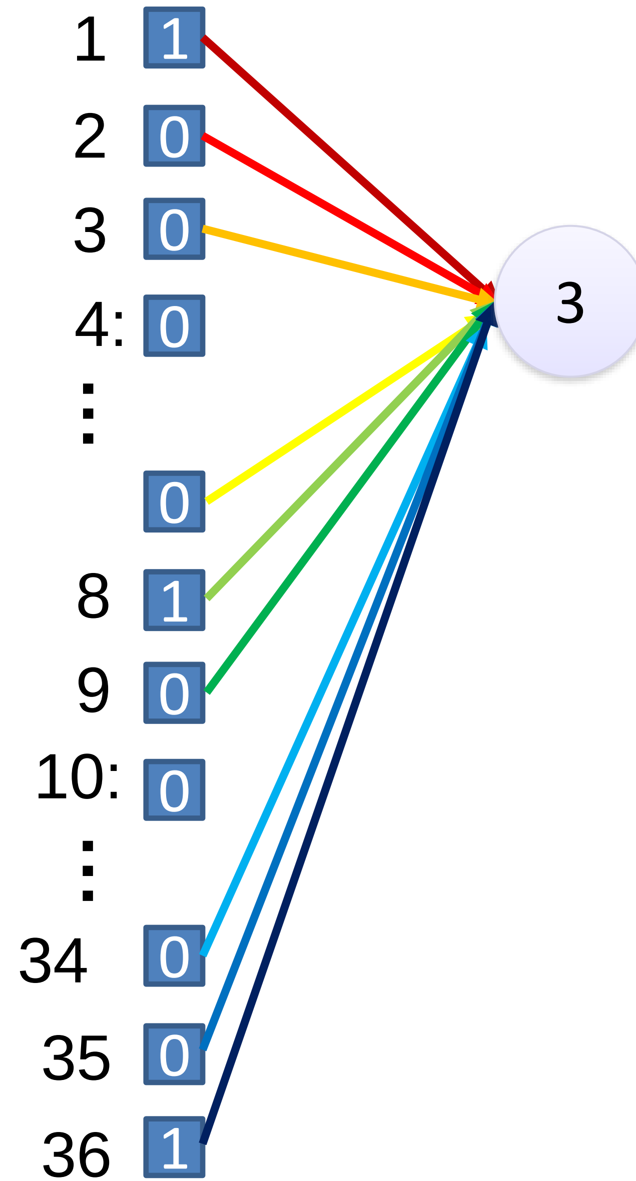
Color image



| | | | | | |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 |

# Convolutional vs FCL




fewer parameters!

# Max Pooling

- Subsampling pixels will not change the object

bird

Subsampling

bird

- fewer parameters to characterize the image

# Max Pooling Layer

| | | |
|---|---|---|
| 1 | -1 | -1 |
| -1 | 1 | -1 |
| -1 | -1 | 1 |

Filter 1

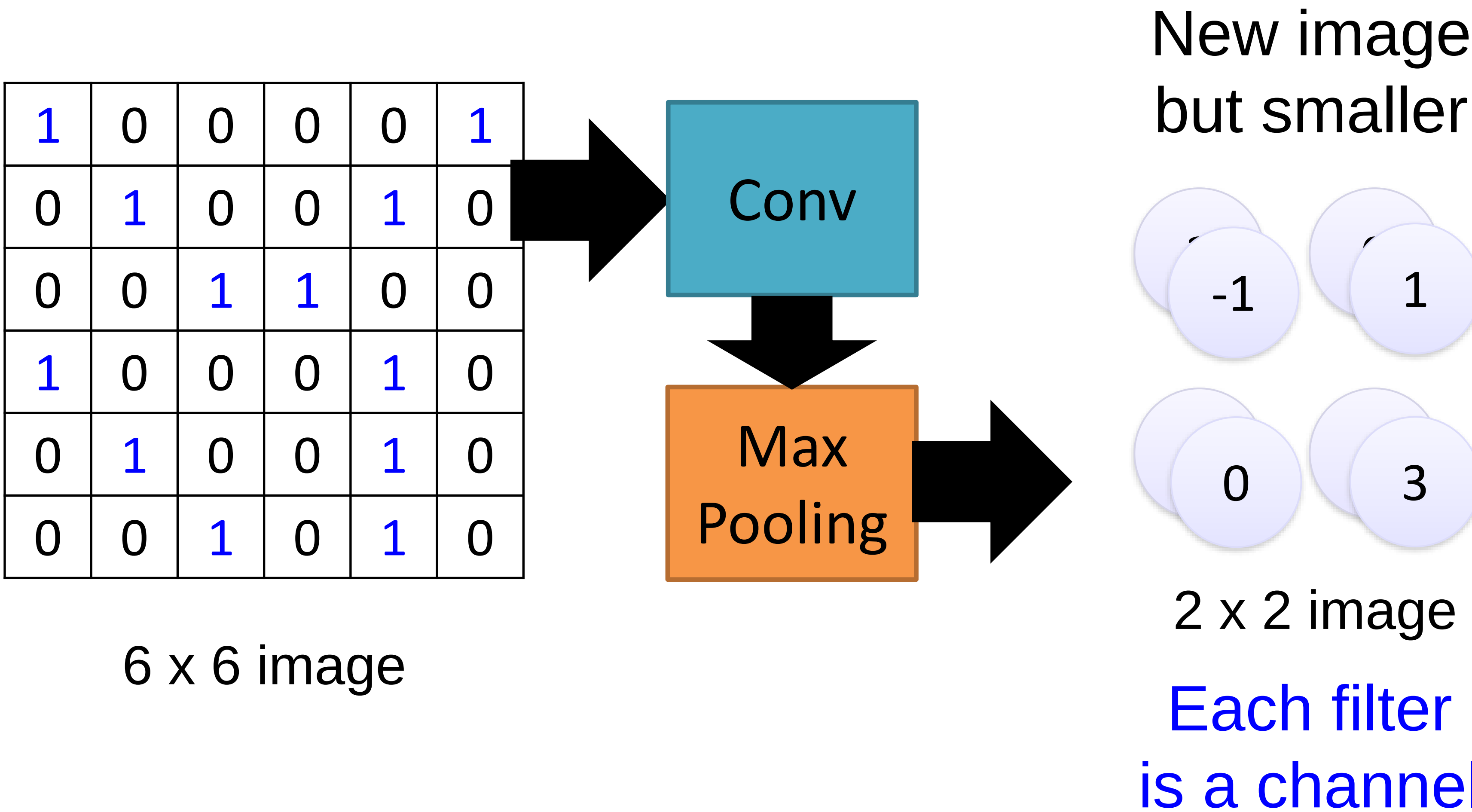
| | | |
|---|---|---|
| -1 | 1 | -1 |
| -1 | 1 | -1 |
| -1 | 1 | -1 |

Filter 2

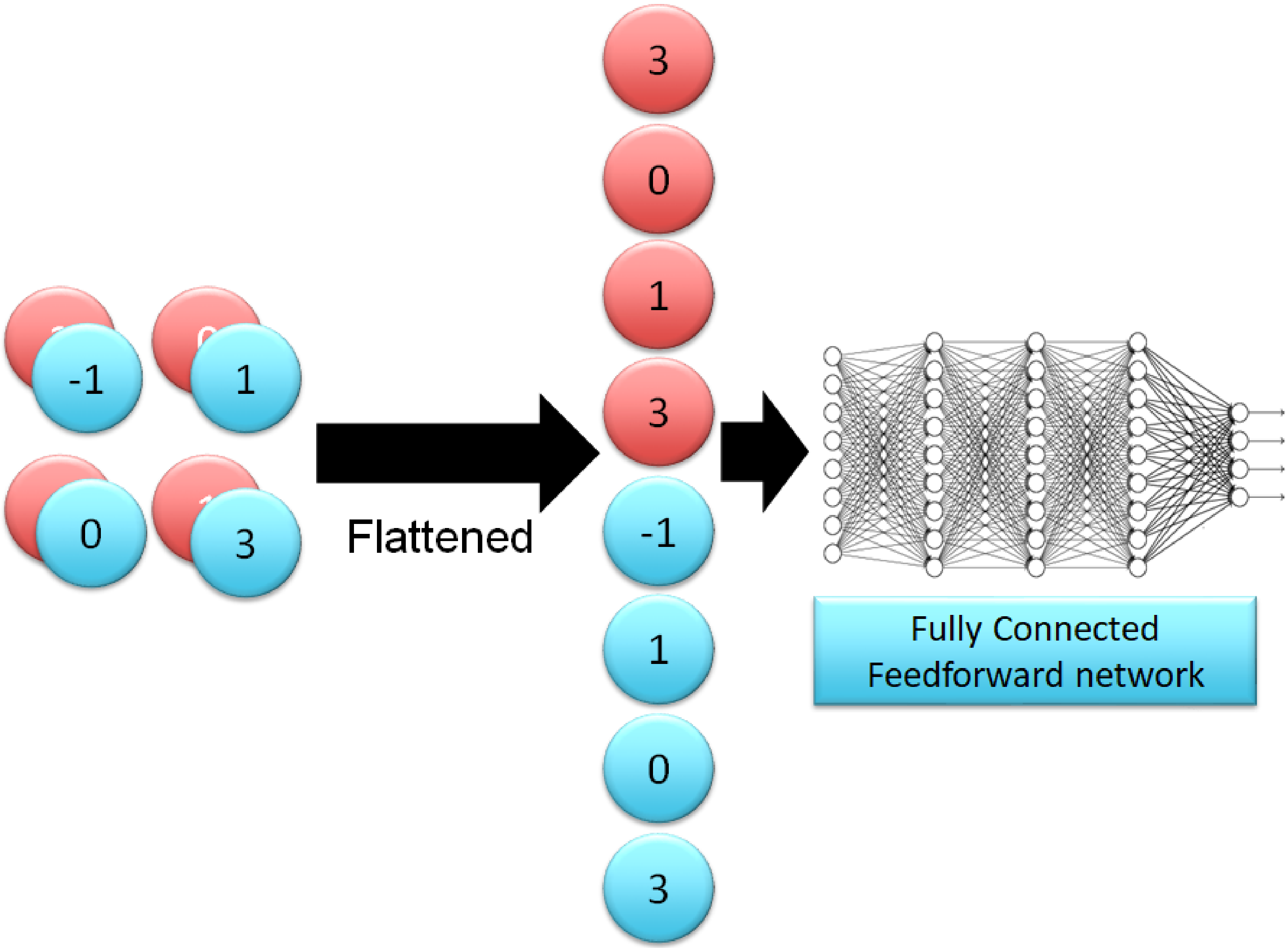
| | | | |
|---|---|---|---|
| 3 | -1 | -3 | -1 |
| -3 | 1 | 0 | -3 |
| -3 | -3 | 0 | 1 |
| 3 | -2 | -2 | -1 |

| | | | |
|---|---|---|---|
| -1 | -1 | -1 | -1 |
| -1 | -1 | -2 | 1 |
| -1 | -1 | -2 | 1 |
| -1 | 0 | -4 | 3 |

# Max Pooling Layer

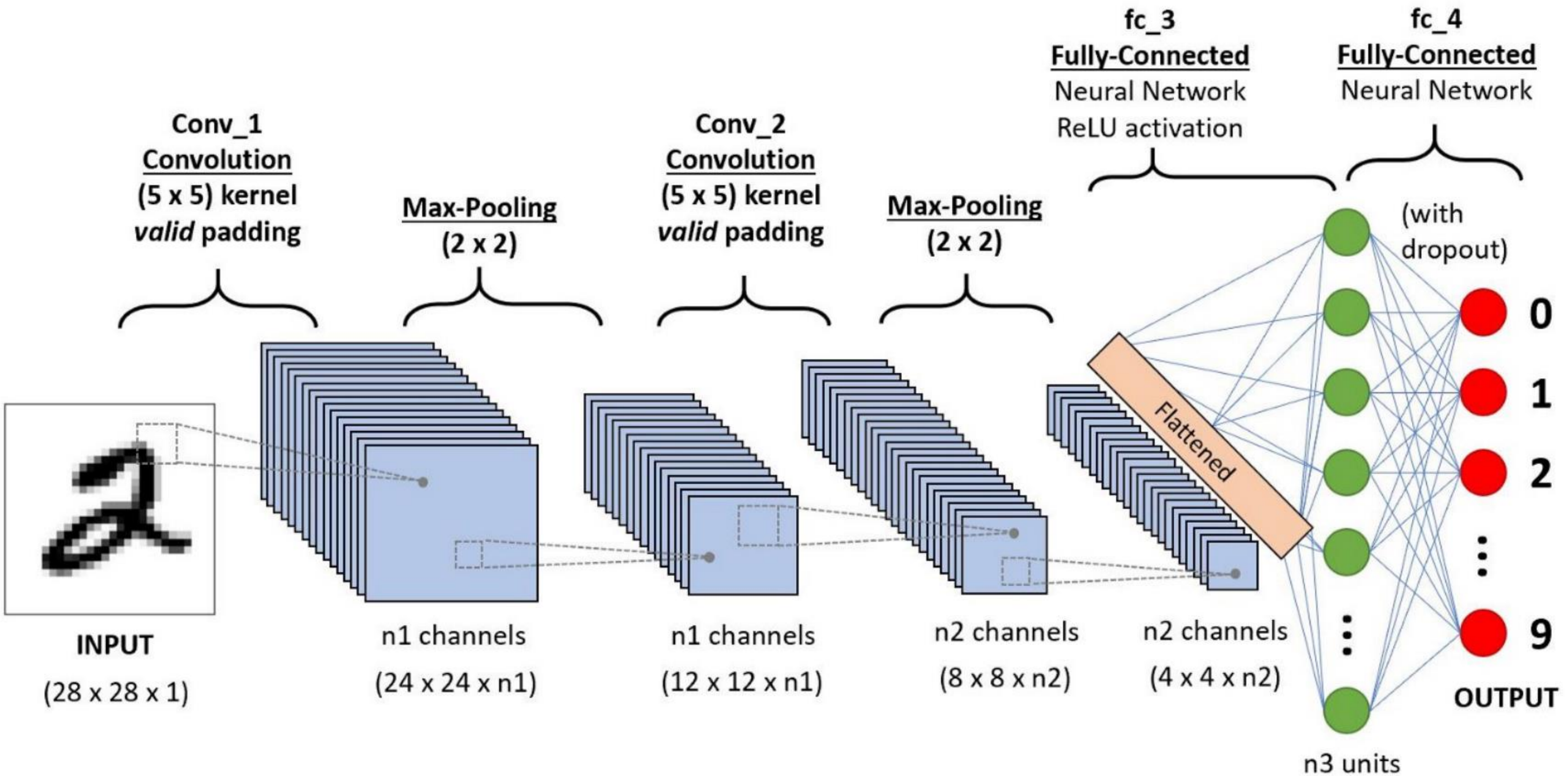


# Flattening




Revolutionized Computer Vision!

# Convolutional Neural Networks

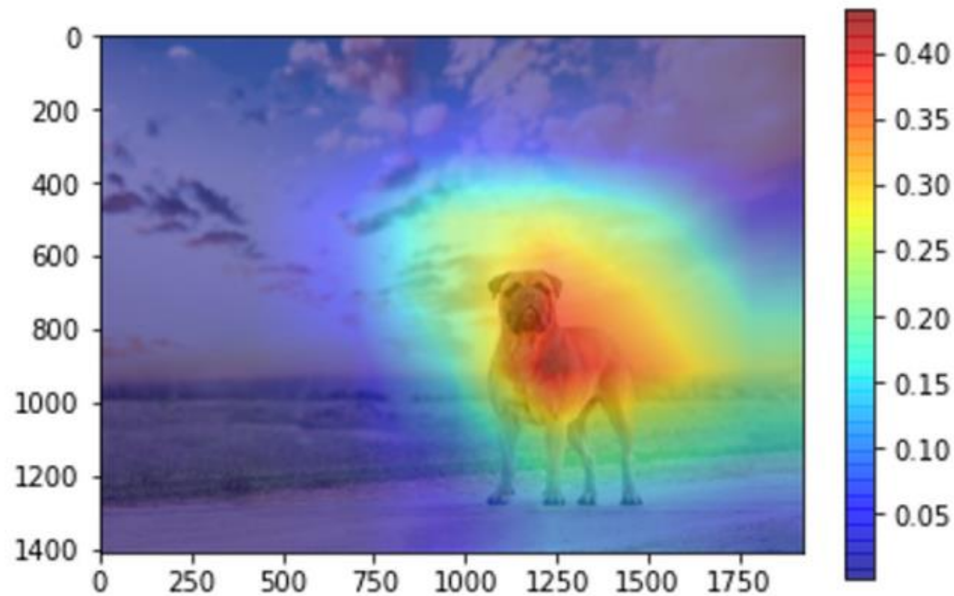



Revolutionized Computer Vision!

# Attention in CNNs

- The network learns to attend to the proper spatial location to perform a downstream task

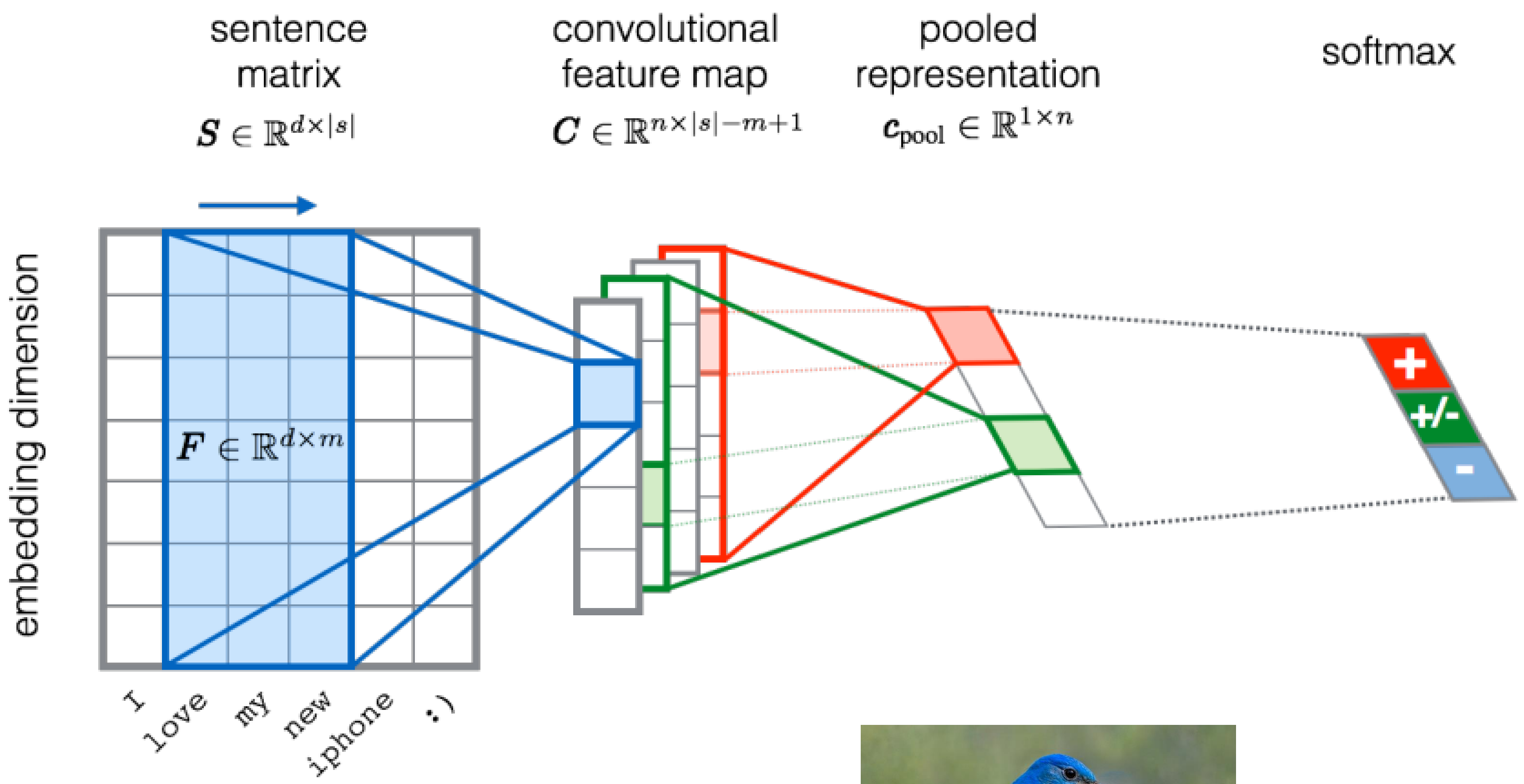
(a) Original image




(b) Attention map

# CNNs for Text Classification

- Sentence Sizes are different



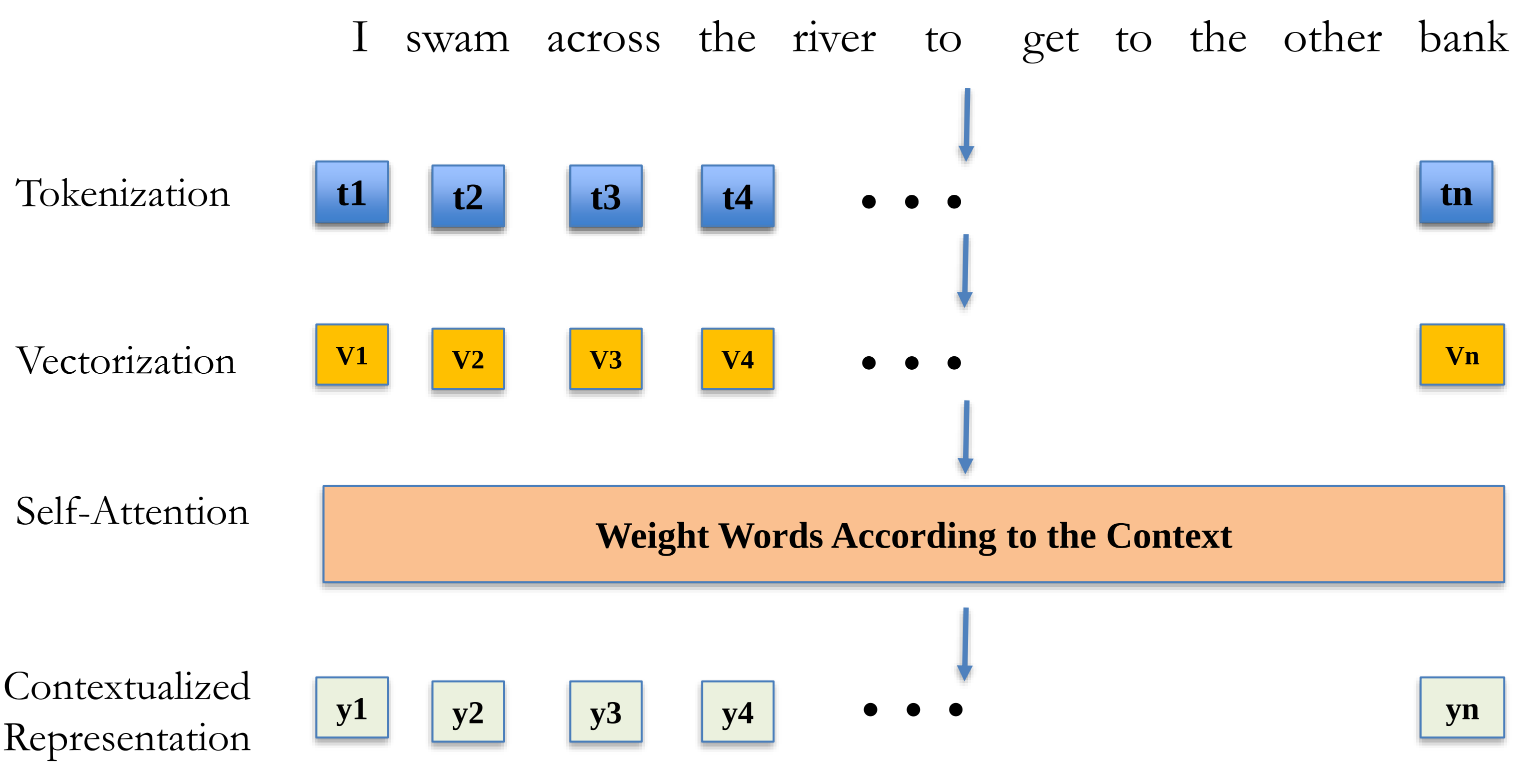

- We can use 1D convolution

# Transformers: intuition

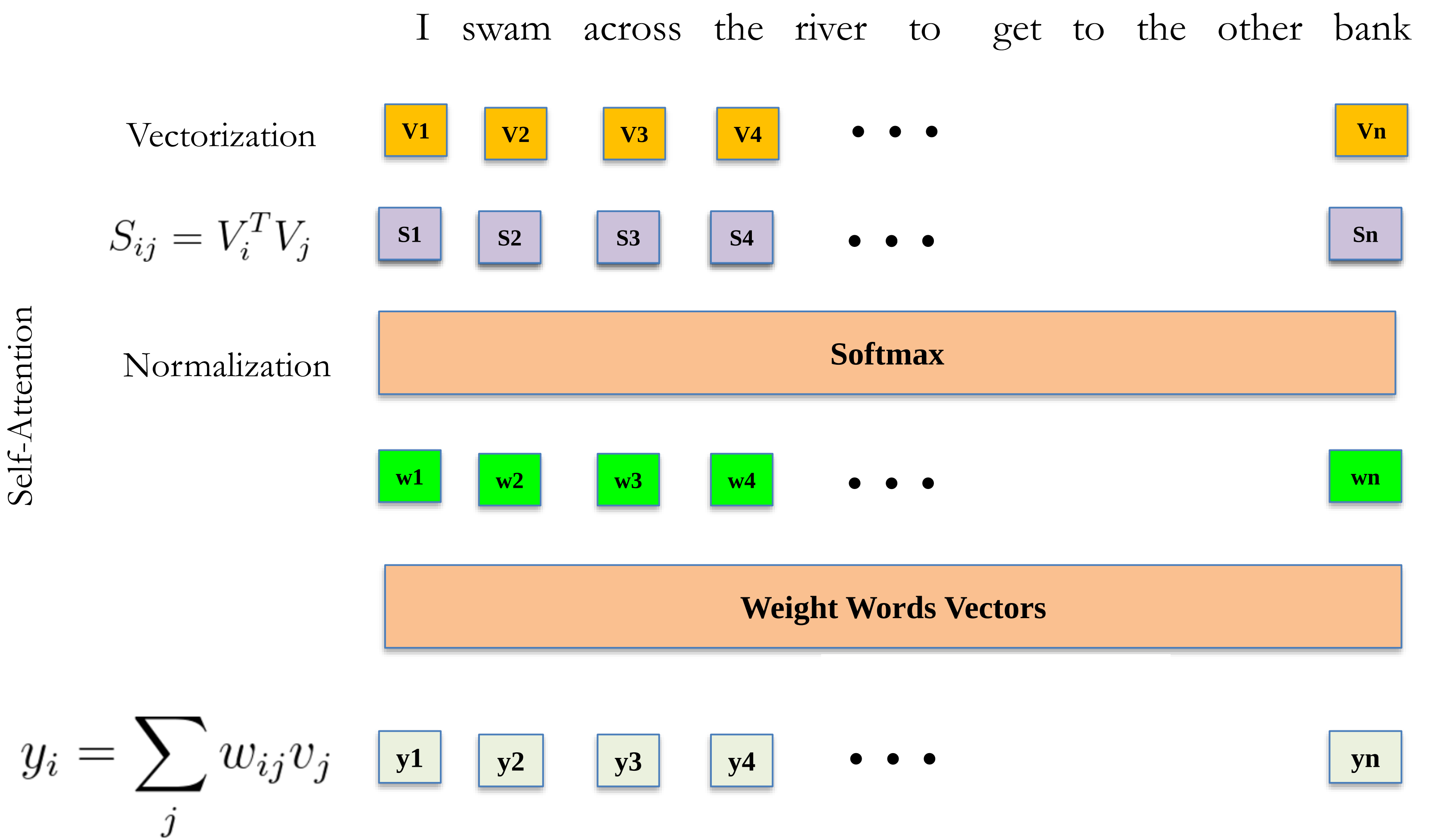
- Example:
  - I swam across the river to get to the other **bank**
  - I drove across the road to get to the other **bank**

→ Context Matters

- Can we build an architecture that weighs neighboring words to understand meaning of words?



# Transformers: Attention Mechanism Basics

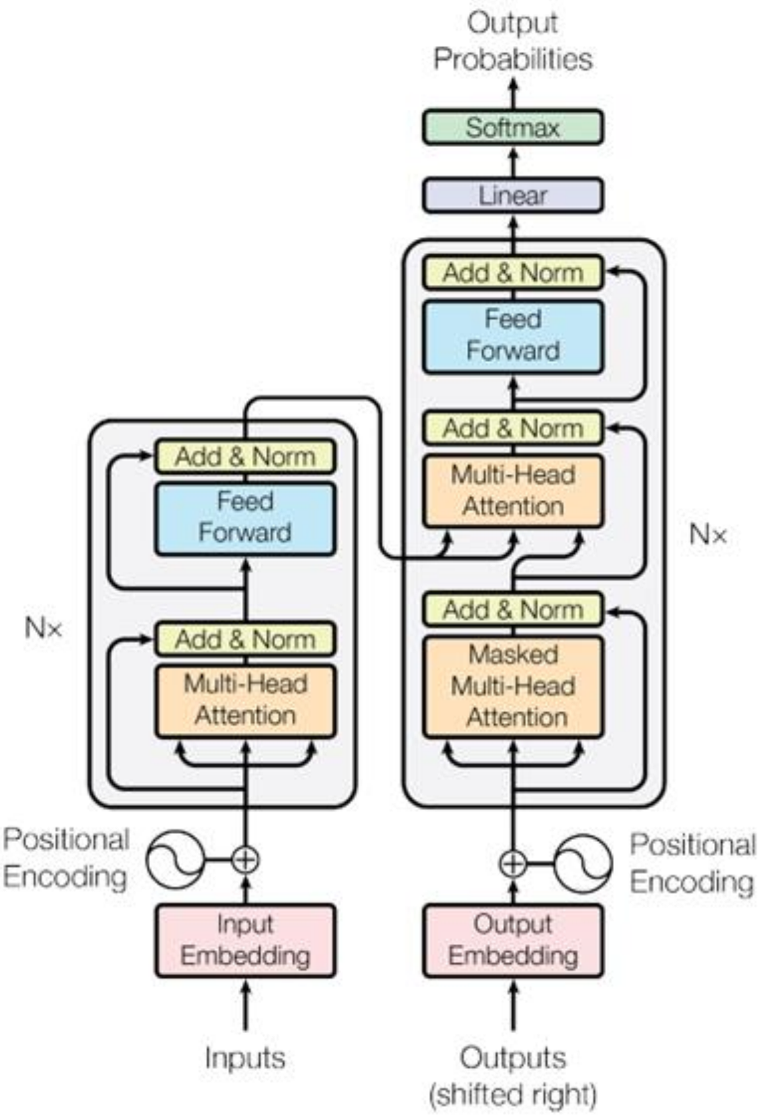


# Transformers

- Attention over entire input: path lengths between words become constant
- Parallelization

- Inputs: word vectors
  - Source sentence
  - Input sentence produced so far
- Output: word vectors
  - Probability distribution for the next word
- Position Encoding

Encoder Decoder




Cats Eat Mice Katzen fressen

# Transformers

## Embeddings and Softmax

We maximize the probability of the next output token

Share embedding weights and the pre-softmax linear transformation

Repetitive Blocks

Attention Blocks

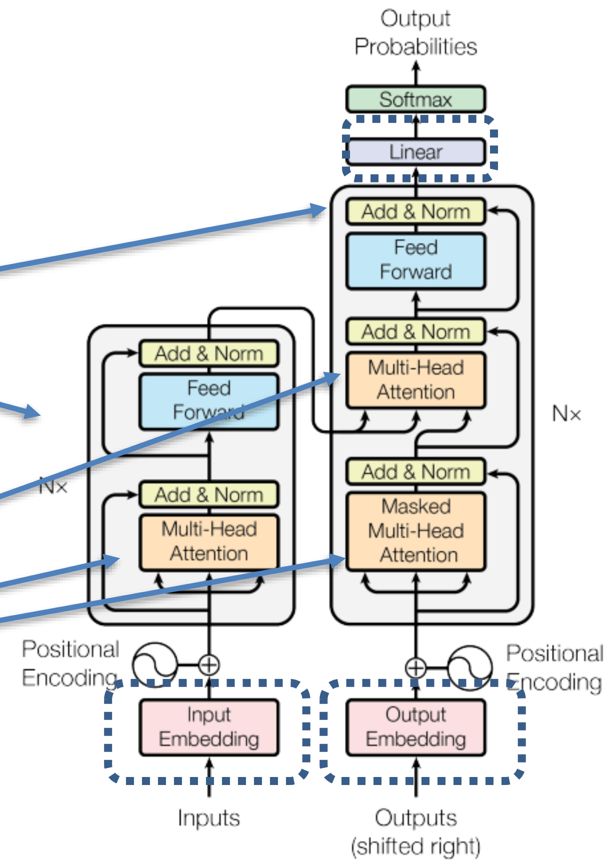
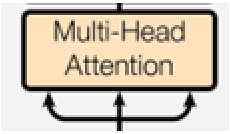



Figure 1: The Transformer - model architecture.

# Transformers: Attention Mechanism

Which word in the source sentence we should pay more attention at each step?

Queries: Target Sentence
Keys: Source Sentence
Values: Source Sentence




Three inputs

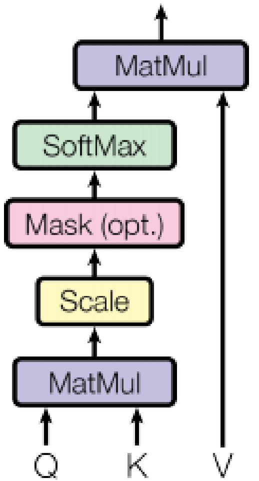
Scaled Dot-Product Attention: it selects the most important word in the source

Similarity

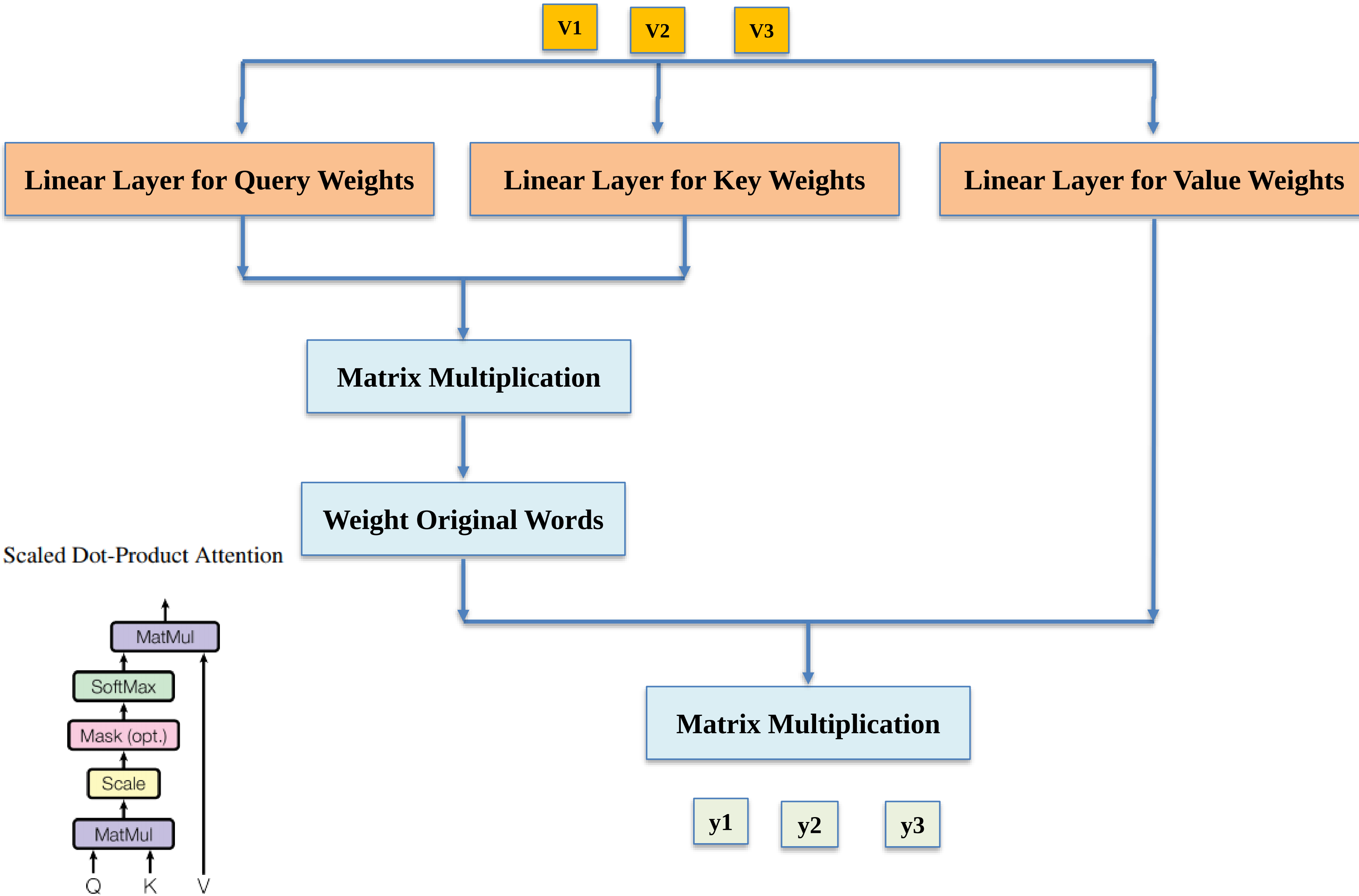
Q: queries
K: keys
V: values

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

Scaled Dot-Product Attention

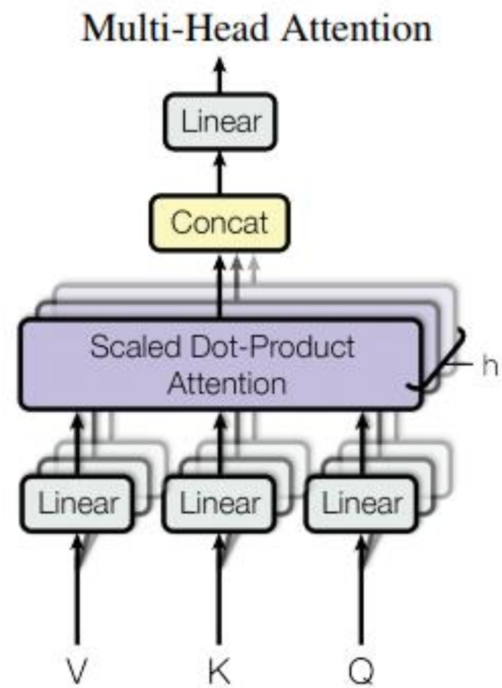


# Transformers: Attention Mechanism



# Transformers: Attention Mechanism

Multi-Head Attention: Learn **h** attention heads to learn different representation simultaneously so each head pays to different types of things




For projecting back to the input dimension

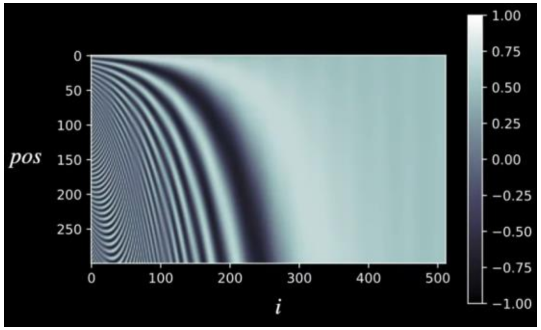
$$\mathrm{MultiHead}(Q,K,V) = \mathrm{Concat}(\mathrm{head_1}, \ldots, \mathrm{head_h})W^O$$
$$\text{where } \mathrm{head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Where the projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$.

# Transformers: Positional Encoding

How do we take words orders into  accounts?

- Augment the embeddings with a position identifier




Reason: no RNN to model the sequence position

Two types:

- learned positional embeddings Sinusoid:

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

- Bounded, Periodic, Unique

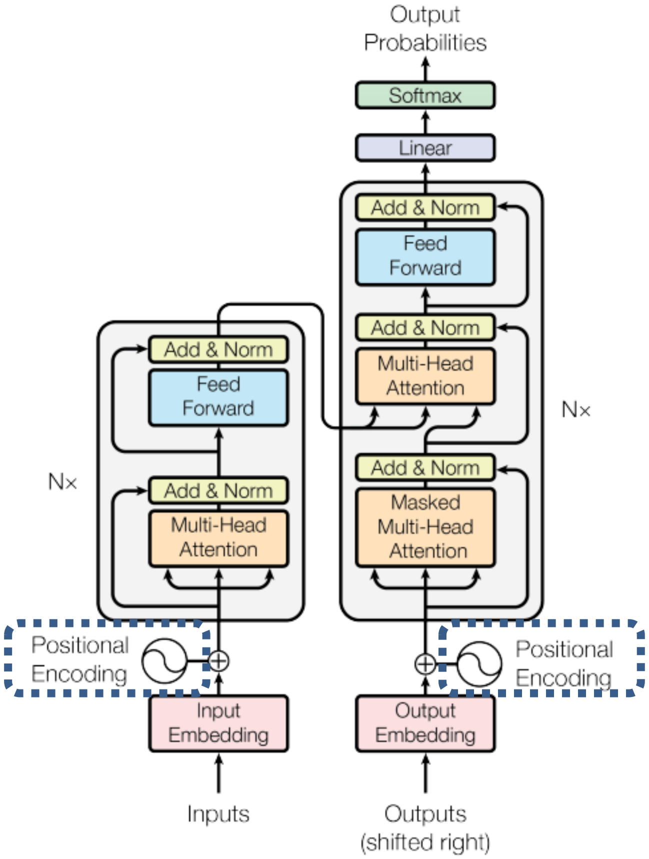



Figure 1: The Transformer - model architecture.

# Transformers

Position-Wise Feed-Forward Networks

$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

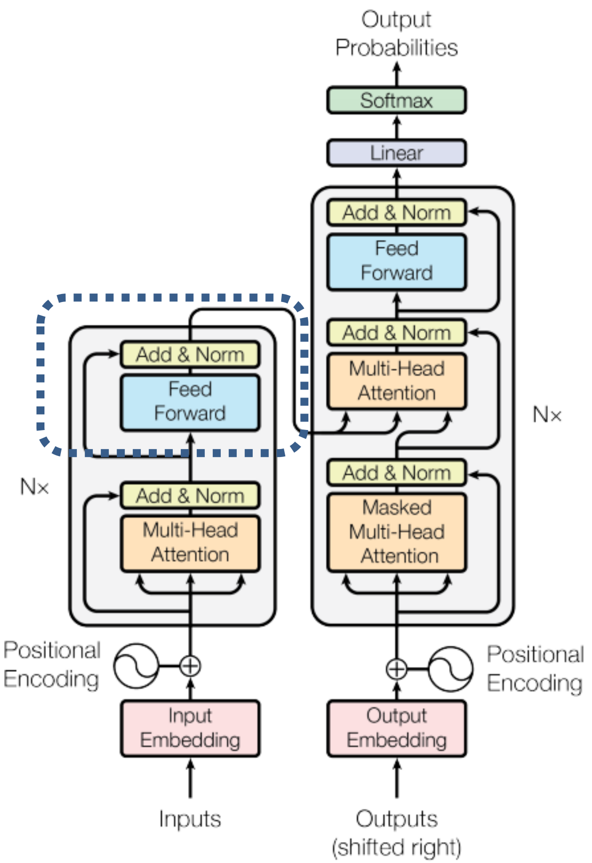



Figure 1: The Transformer - model architecture.

# Transformers: Decoder

Masked Multihead Attention: limits decencies only to prior words

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

Improvements over LSTM:

- The source languages and their semantics and grammars are analyzed
- Mapping to the source language is learned separately

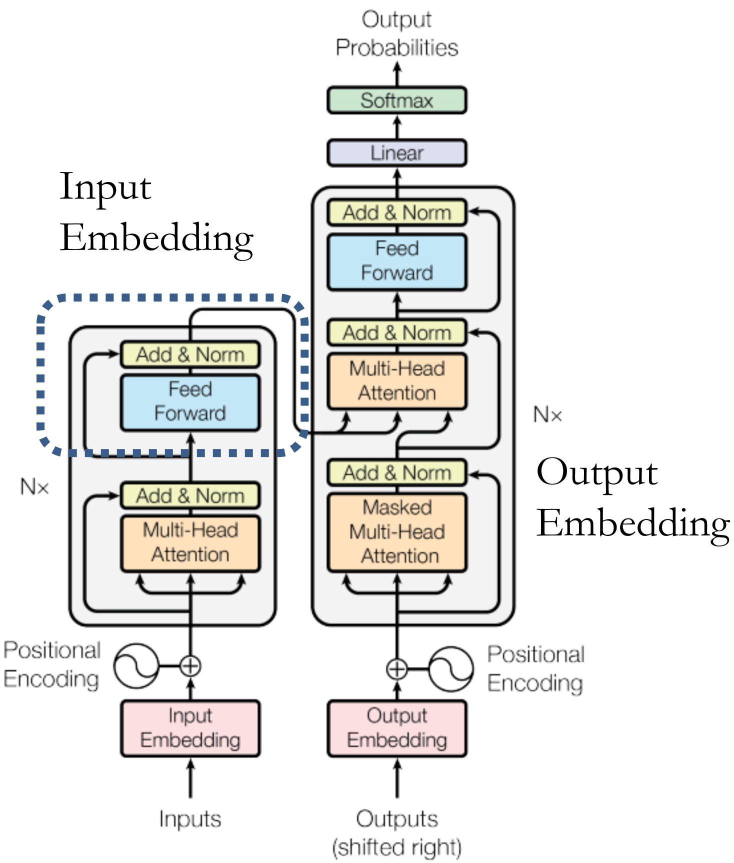



Figure 1: The Transformer - model architecture.

# Performance Comparison

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

| Model | BLEU | | Training Cost (FLOPs) | |
|---|---|---|---|---|
| | EN-DE | EN-FR | EN-DE | EN-FR |
| ByteNet [17] | 23.75 | | | |
| Deep-Att + PosUnk [37] | | 39.2 | | $1.0 \cdot 10^{20}$ |
| GNMT + RL [36] | 24.6 | 39.92 | $2.3 \cdot 10^{19}$ | $1.4 \cdot 10^{20}$ |
| ConvS2S [9] | 25.16 | 40.46 | $9.6 \cdot 10^{18}$ | $1.5 \cdot 10^{20}$ |
| MoE [31] | 26.03 | 40.56 | $2.0 \cdot 10^{19}$ | $1.2 \cdot 10^{20}$ |
| Deep-Att + PosUnk Ensemble [37] | | 40.4 | | $8.0 \cdot 10^{20}$ |
| GNMT + RL Ensemble [36] | 26.30 | 41.16 | $1.8 \cdot 10^{20}$ | $1.1 \cdot 10^{21}$ |
| ConvS2S Ensemble [9] | 26.36 | **41.29** | $7.7 \cdot 10^{19}$ | $1.2 \cdot 10^{21}$ |
| Transformer (base model) | 27.3 | 38.1 | $\mathbf{3.3 \cdot 10^{18}}$ | |
| Transformer (big) | **28.4** | **41.0** | $2.3 \cdot 10^{19}$ | |

# GPTs and BERT

- A transformer uses Encoder stack to model input, and uses Decoder stack to model output (using input information from encoder side).
- If we do not have input and just want to model the “next word”, we can get rid of the Encoder side of a transformer and output “next word” one by one. This gives us GPTs.

  - Application: question answering, summarization

- If we are only interested in training a model for the input for some other tasks, then we do not need the Decoder of the transformer, that gives us BERT.

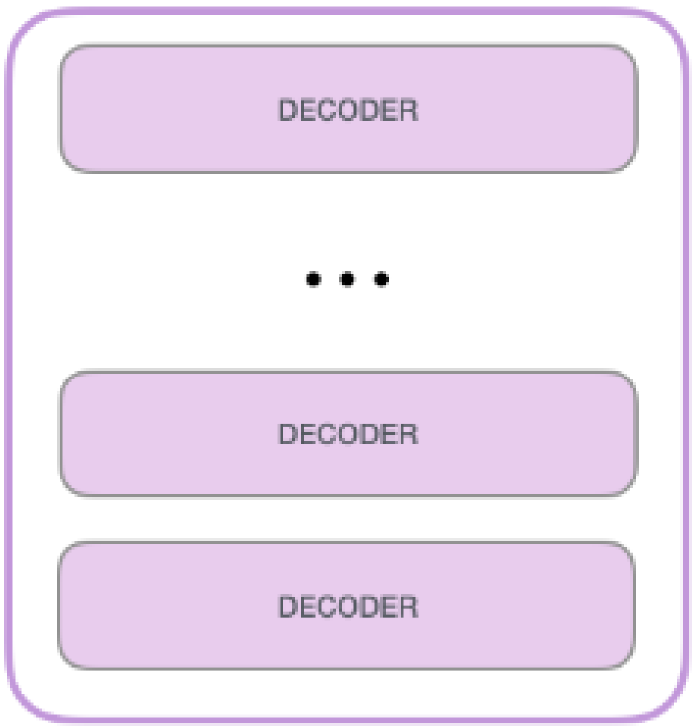
  - Application: classification

# GPT and BERT

Two steps:

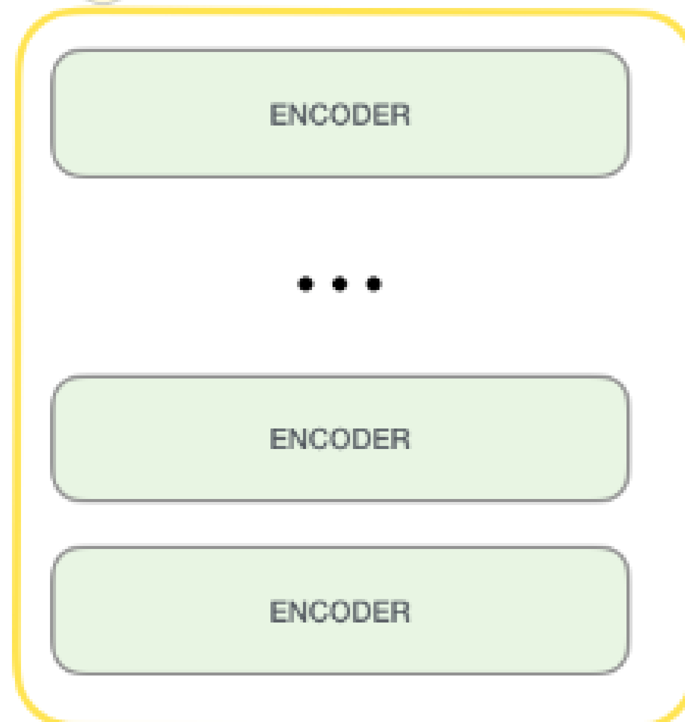
- Pretraining using self-supervised learning
- Finetuning

GPT-2




Tasks: classification, similarity, multiple choice, next word

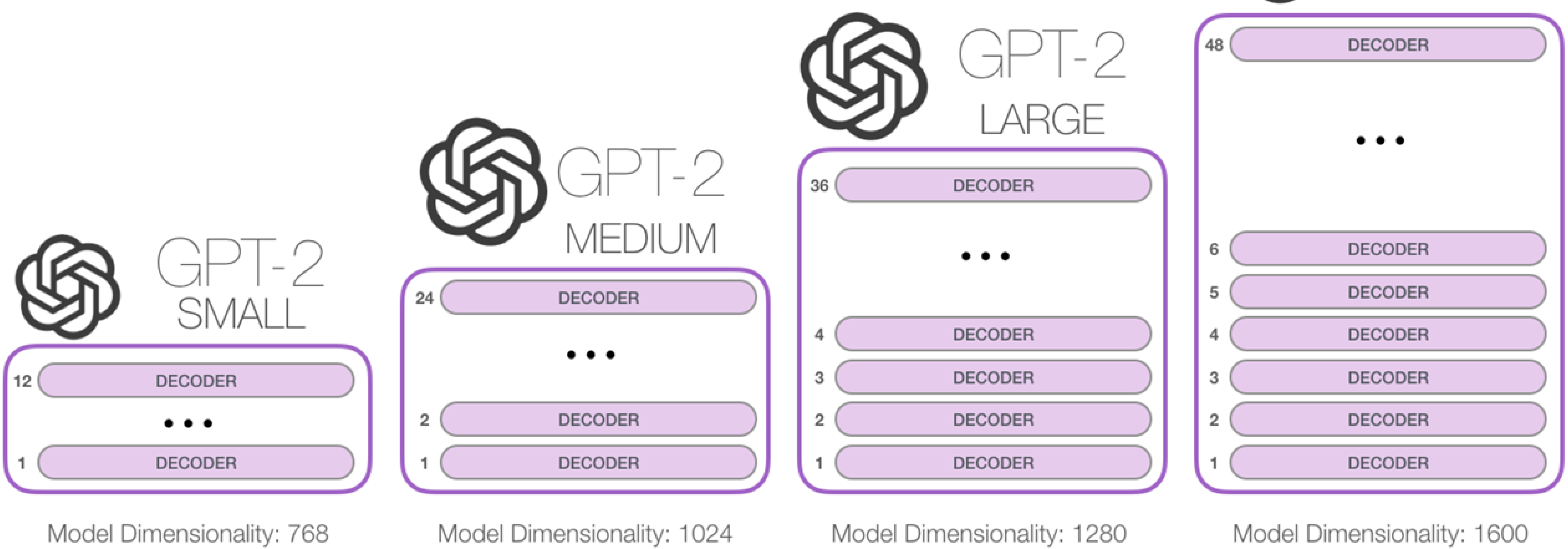
BERT




Tasks: masking, next sentence prediction

# GPT Versions

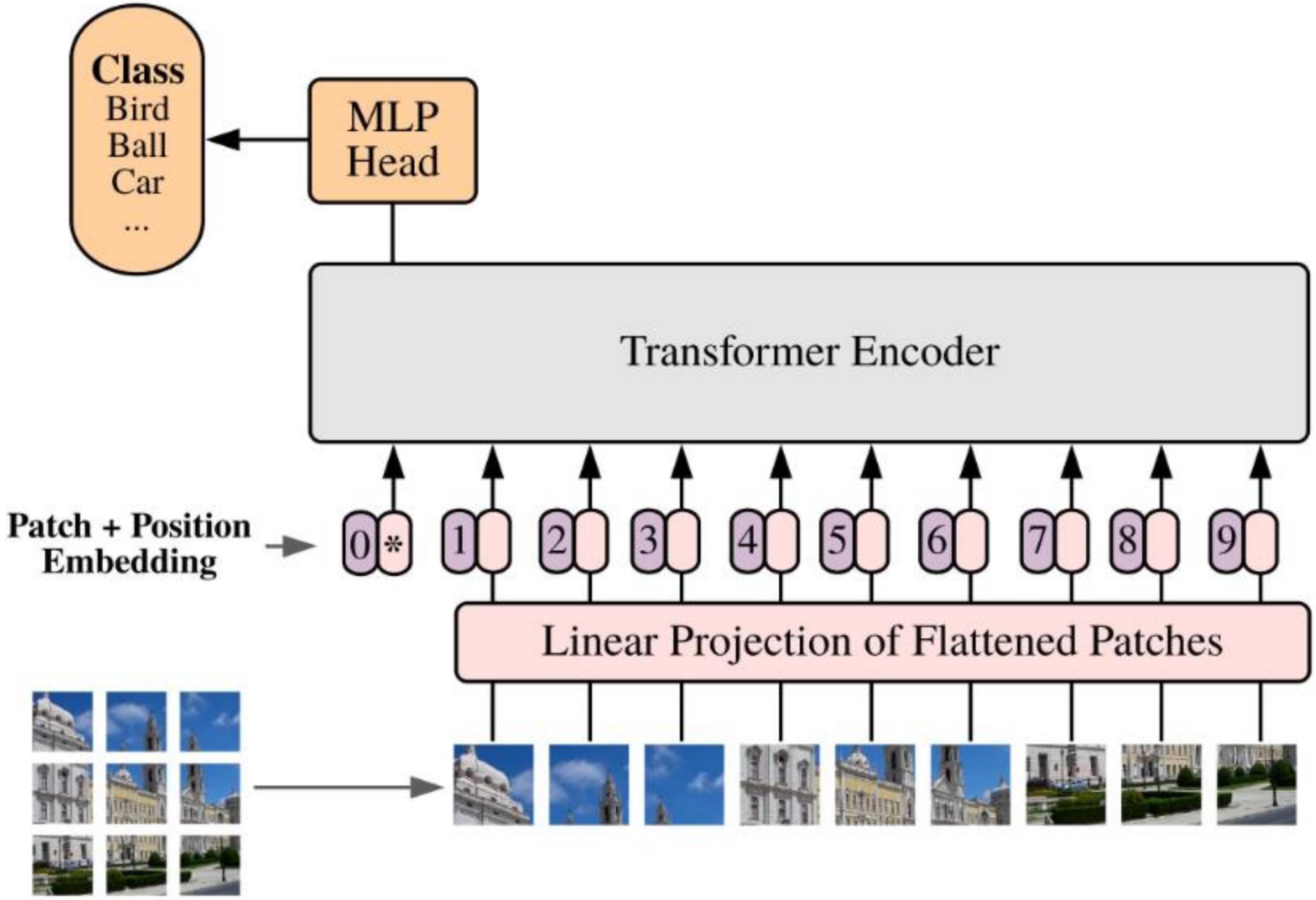
GPT : released Jun 2018
GPT-2: released Nov 2019 with 1.5B parameters
GPT-3: released Jun 2020 with 175B parameters trained on 45TB texts




117M parameters | 345M | 762M | 1542M

# Visual Transformer

Idea: Representing an image as a sequence




Similar performance to CNNs with less computational load

# Visual Language Transformer

Representing coupled vision-language data with a single integrated embedding

- Application: question answering, caption generation

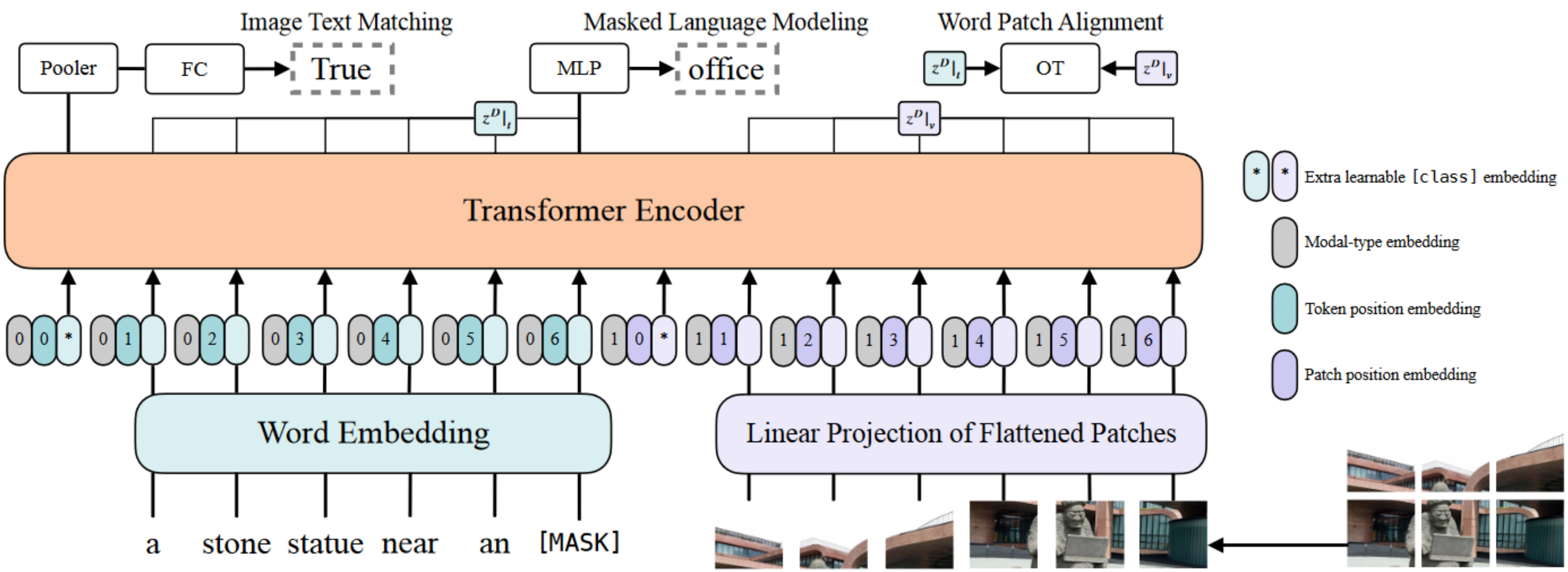



*Figure 3.* Model overview. Illustration inspired by Dosovitskiy et al. (2020).