



CHULA ENGINEERING COMPUTER  
Foundation toward Innovation



## Advanced Deep Learning (DL)

Prof. Peerapon Vateekul, Ph.D.

Department of Computer Engineering,  
Faculty of Engineering, Chulalongkorn University  
[Peerapon.v@chula.ac.th](mailto:Peerapon.v@chula.ac.th)

[www.cp.eng.chula.ac.th/~peerapon/](http://www.cp.eng.chula.ac.th/~peerapon/)



# Outline

- Introduction to Deep Learning ([recap](#))
- Convolutional Neural Networks (CNN) [Imaging Task] ([recap](#))
  
- Recurrent Neural Networks (RNN) [Time Series & NLP Tasks]
- Transformer models
- Hugging Face
- Note: Today's lecture introduces several DL models so that you can understand how to select the right one for your problem. You'll learn more in depth in the AI course next semester.

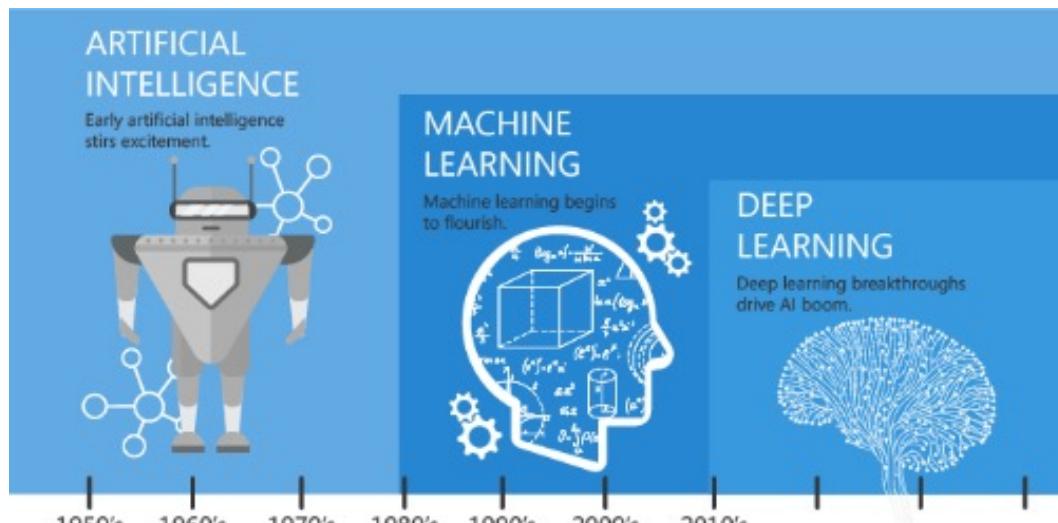


## Introduction to Deep Learning (recap)

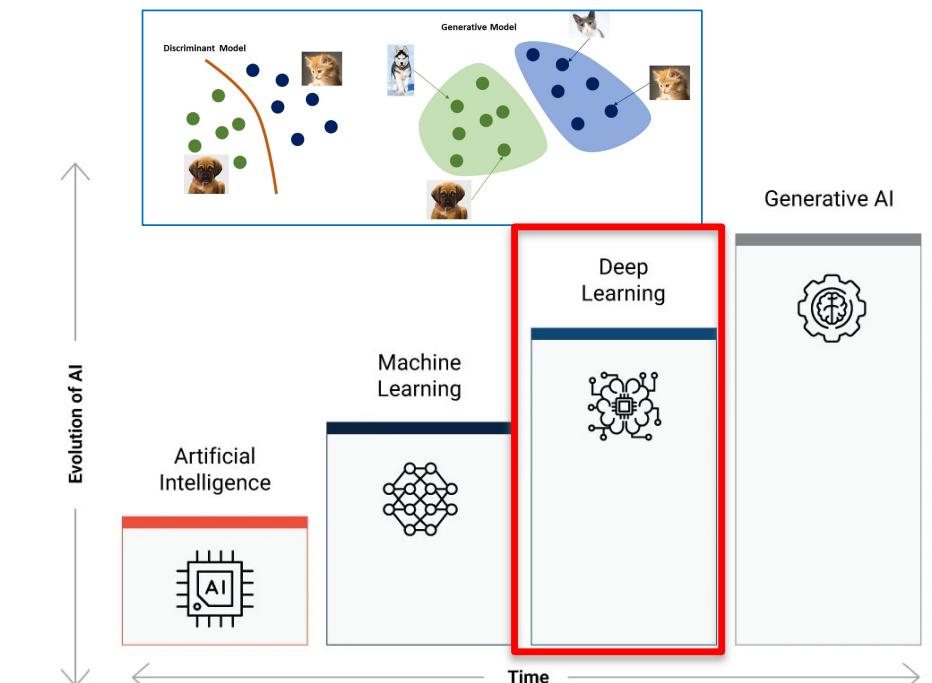
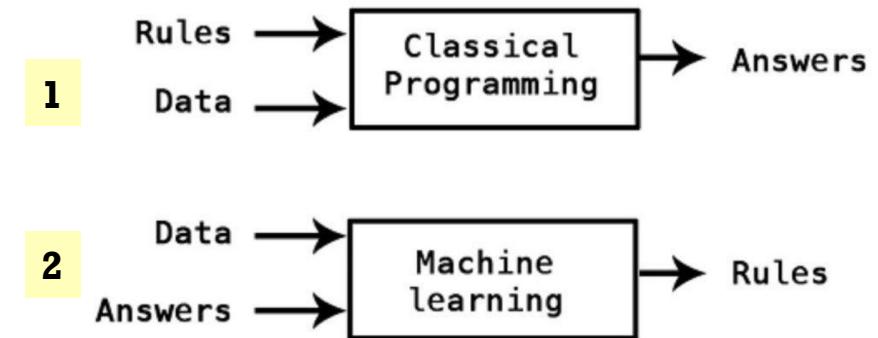


# AI = Automation

- 1) Rule-based AI
- 2) Machine Learning (ML)

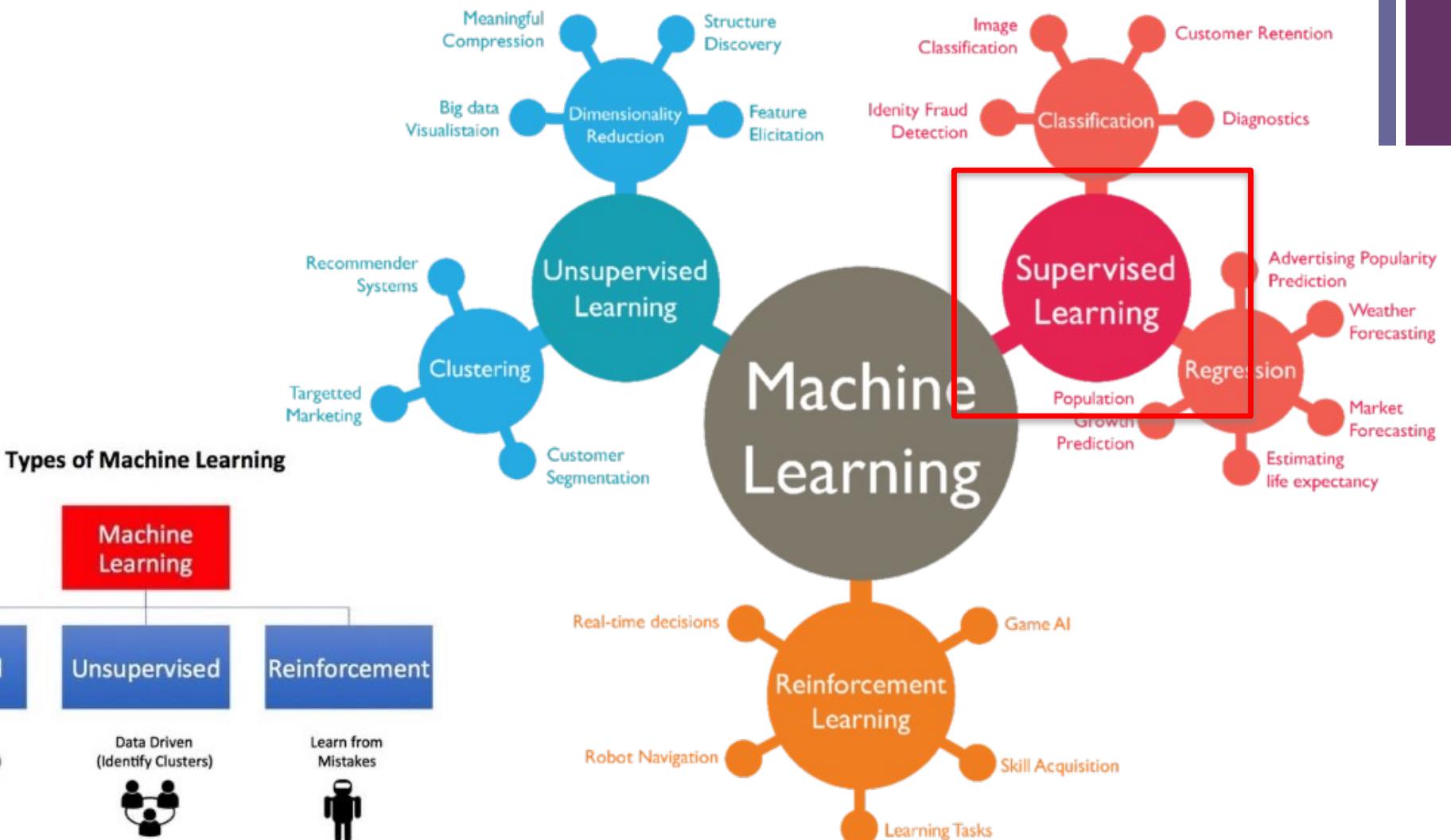


Since an early flush of optimism in the 1950's, smaller subsets of artificial intelligence - first machine learning, then deep learning, a subset of machine learning - have created ever larger disruptions.



<https://mc.ai/machine-learning-basics-artificial-intelligence-machine-learning-and-deep-learning/>

# + Machine Learning (ML)





# Task1: Supervised learning

## Handcrafted features

Training Data



inputs					target
Age	Gender	BodyTemp	Cough	Corona	
12	Female	37	Yes	Yes	
35	Female	39	No	Yes	
32	Male	38	Yes	No	

Testing Data

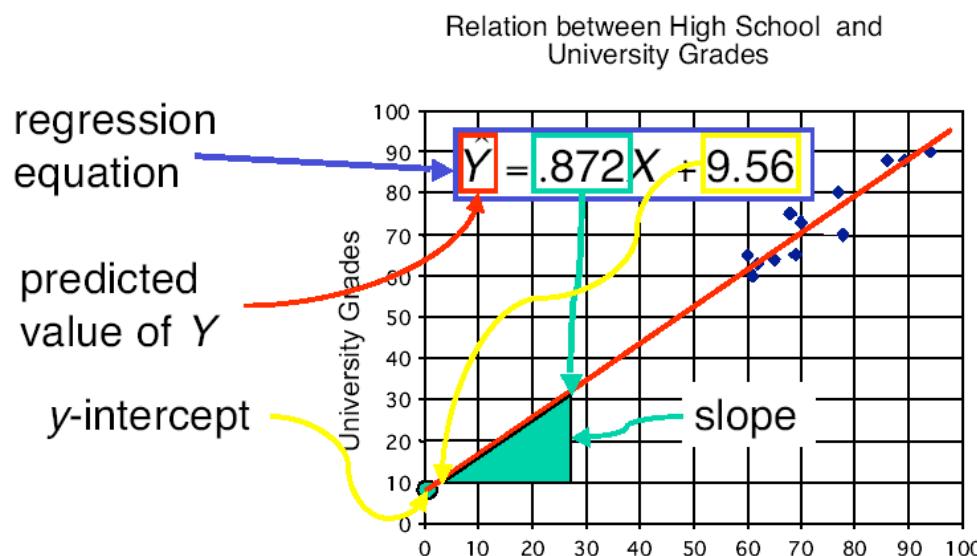


Age	Gender	BodyTemp	Cough	Corona
25	Male	40	No	?

Application: Corona Prediction



# Regression – Linear Relationship



$$\hat{y} = \hat{w}_0 + \hat{w}_1 x_1 + \hat{w}_2 x_2$$

target      intercept      input

weight, coefficient

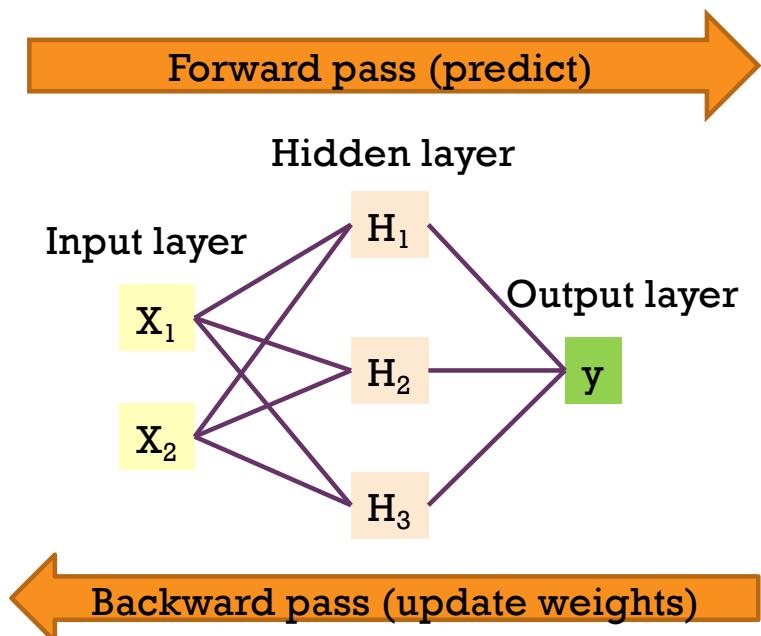
- The least square method aims to minimize the following term

$$\sum_{\text{training data}} (y_i - \hat{y}_i)^2$$



# Neural Networks (universal approximator)

## Non-linear relationship



Age	Income	Gender	Province	Corona
25	25,000	Female	Bangkok	Yes
35	50,000	Female	Nontaburi	Yes
32	35,000	Male	Bangkok	No

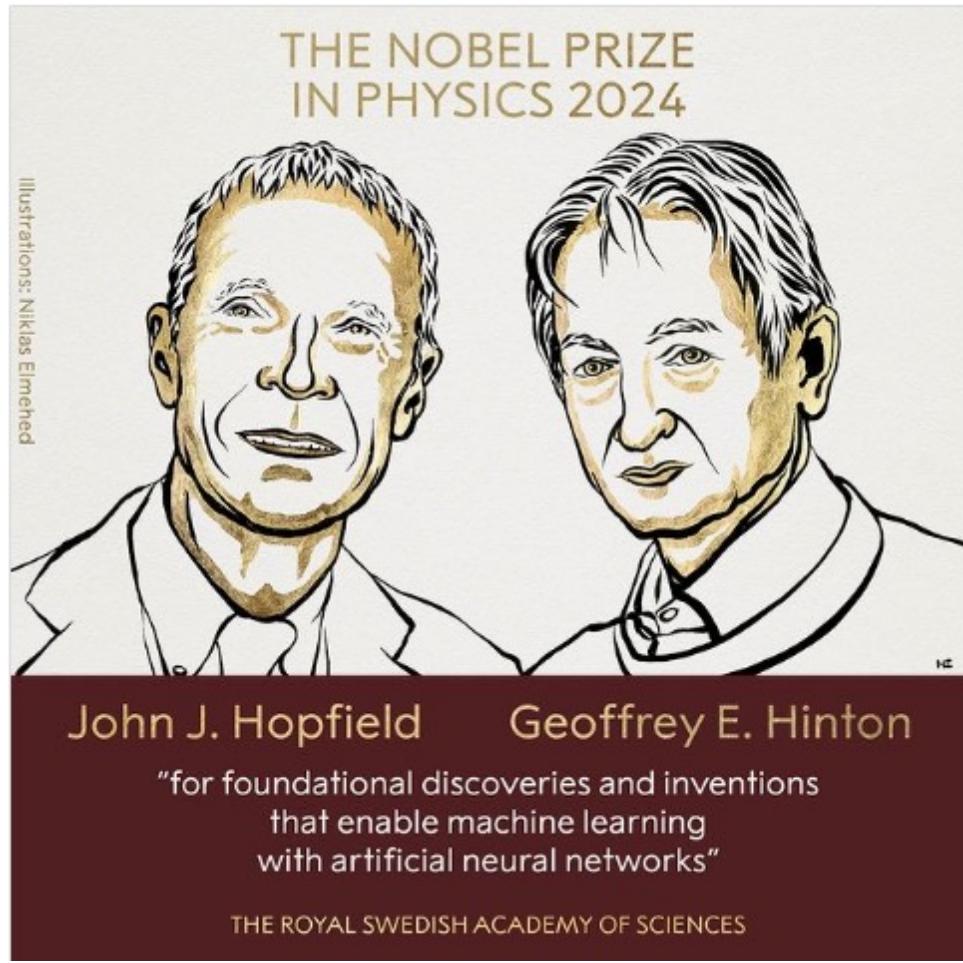
$$\hat{y} = \hat{w}_0 + \hat{w}_1 H_1 + \hat{w}_2 H_2 + \hat{w}_3 H_3$$

$$H_1 = \tanh(\hat{w}_{10} + \hat{w}_{11}x_1 + \hat{w}_{12}x_2)$$

$$H_2 = \tanh(\hat{w}_{20} + \hat{w}_{21}x_1 + \hat{w}_{22}x_2)$$

$$H_3 = \tanh(\hat{w}_{30} + \hat{w}_{31}x_1 + \hat{w}_{32}x_2)$$

# Nobel Prize to Backpropagation's Inventors



8 October 2024

The Royal Swedish Academy of Sciences has decided to award the Nobel Prize in Physics 2024 to

**John J. Hopfield**  
Princeton University, NJ, USA

**Geoffrey E. Hinton**  
University of Toronto, Canada

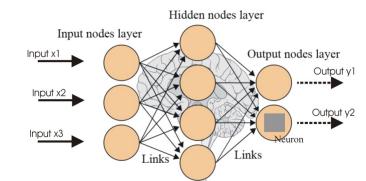
*"for foundational discoveries and inventions that enable machine learning with artificial neural networks"*

**They trained artificial neural networks using physics**

**John Hopfield** invented a network that uses a method for saving and recreating patterns. We can imagine the nodes as pixels.

The *Hopfield network* utilises physics that describes a material's characteristics due to its atomic spin – a property that makes each atom a tiny magnet.

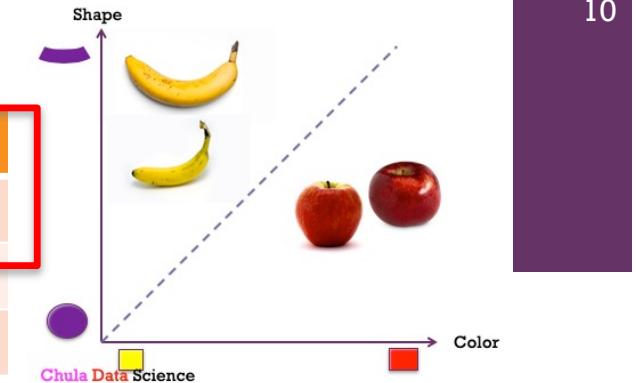
**Geoffrey Hinton** used the Hopfield network as the foundation for a new network that uses a different method: the *Boltzmann machine*. This can learn to recognise characteristic elements in a given type of data.





## Handcrafted features

Age	Income	Gender	Province	Corona
25	25,000	Female	Bangkok	Yes
35	50,000	Female	Nontaburi	Yes
32	35,000	Male	Bangkok	No



Can we still tell the features (columns)?



shutterstock.com - 451802557



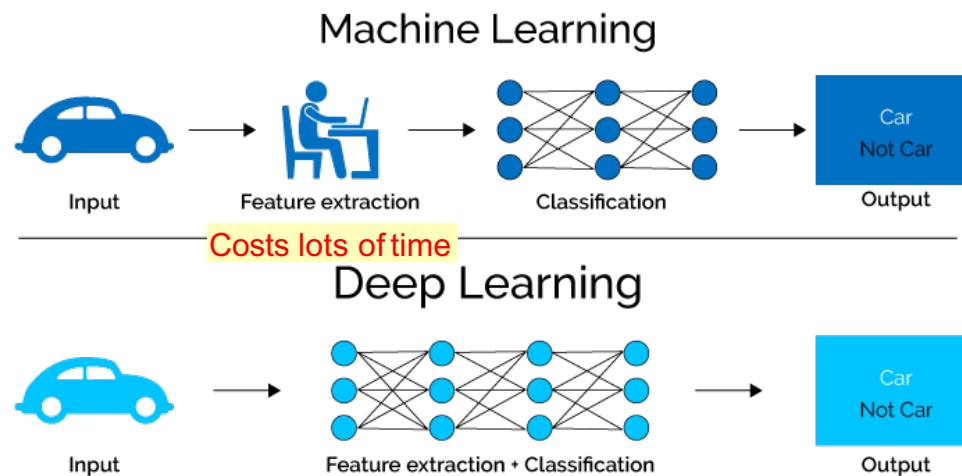
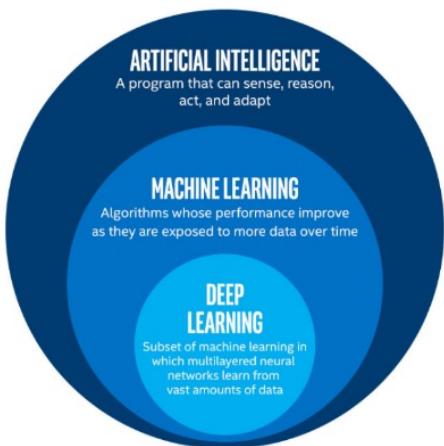
# What is Deep Learning (DL)?



Part of the machine learning field of learning representations of data. Exceptional effective at learning patterns.



Utilizes learning algorithms that derive meaning out of data by using a hierarchy of multiple layers that mimic the neural networks of our brain.

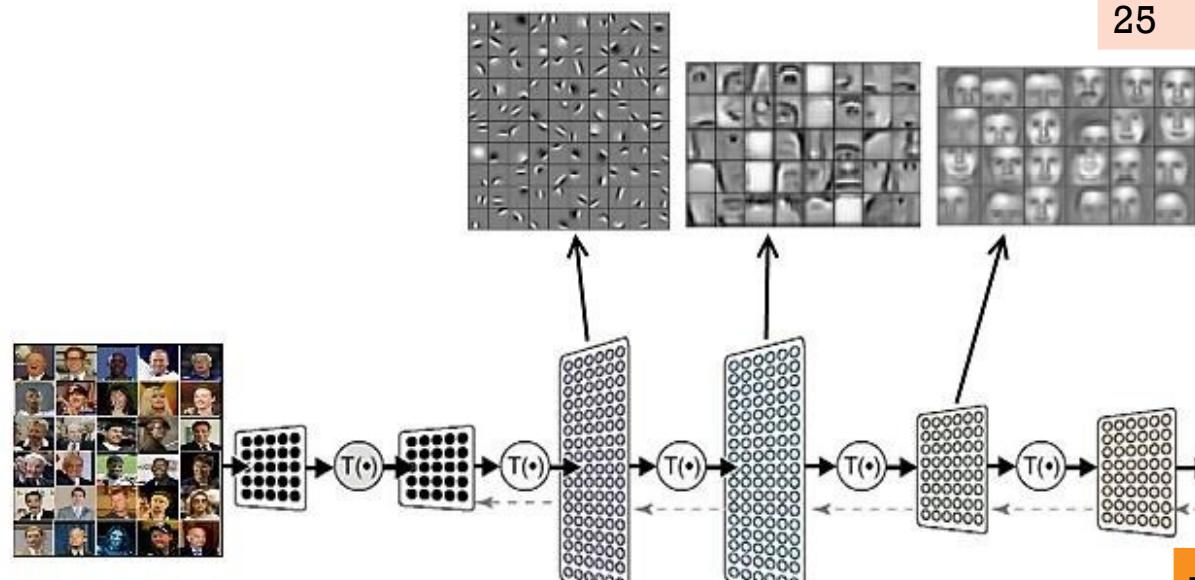




# Deep Learning – Basics (cont.)

What did it learn?

A deep neural network consists of a **hierarchy of layers**, whereby each layer **transforms the input data** into more abstract representations (e.g., edge -> nose -> face). The output layer combines those features to make predictions.



Age	Income	Gender	Province	Corona
25	25,000	Female	Bangkok	Yes

x1	x2	x3	x4	Corona
0.7	0.2	-0.5	-0.1	Yes

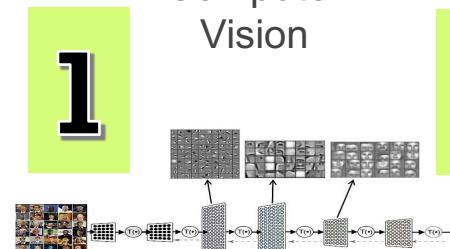
# Deep Learning Application



Speech  
Recognition



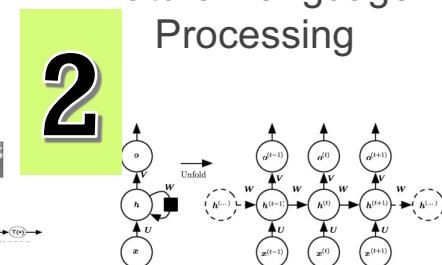
Computer  
Vision



CNN



Natural Language  
Processing

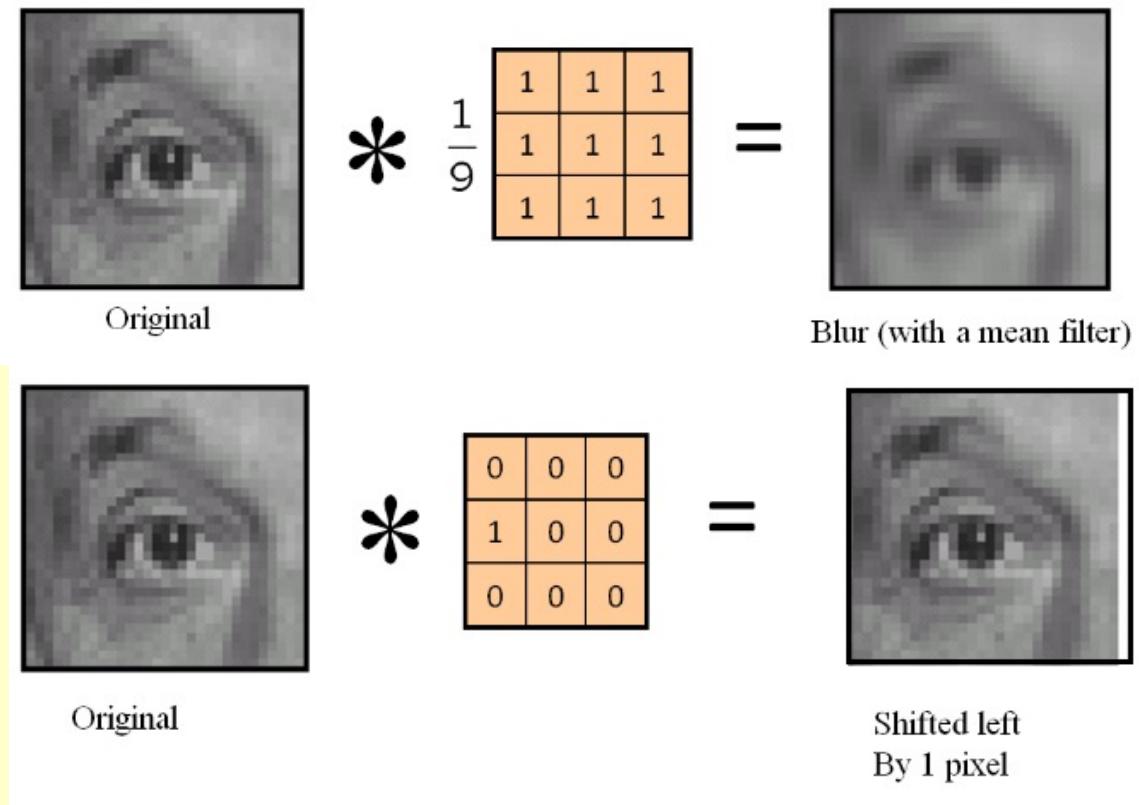
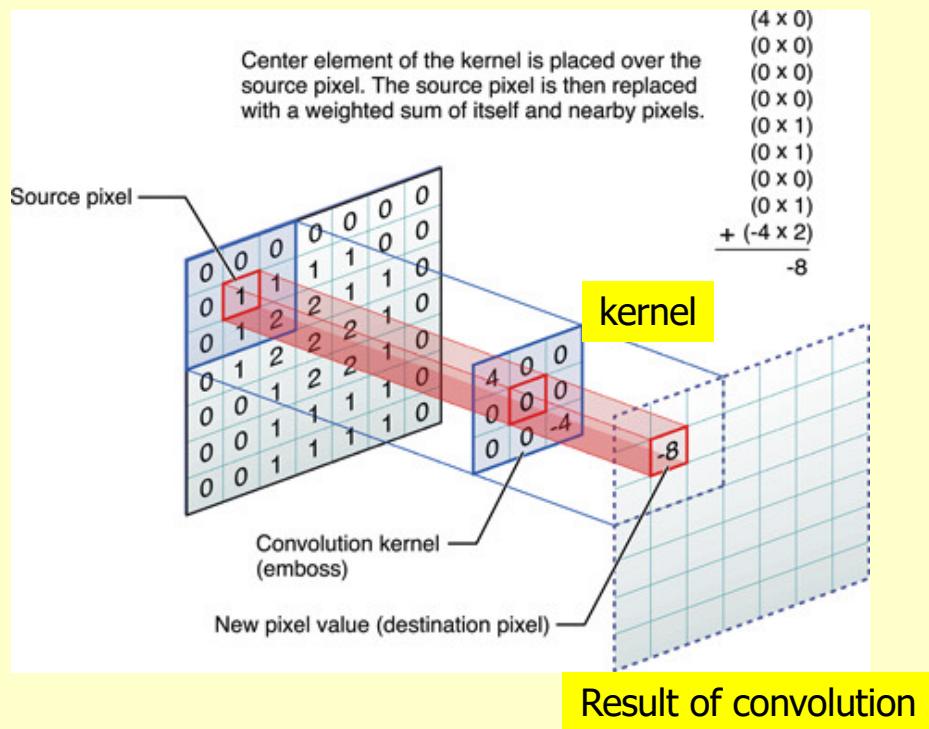


RNN



## Convolutional Neural Networks (CNN) (recap)

# Convolution



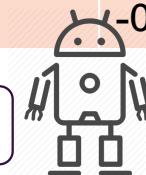
Changing the kernel (filter) results in different image processing outcomes



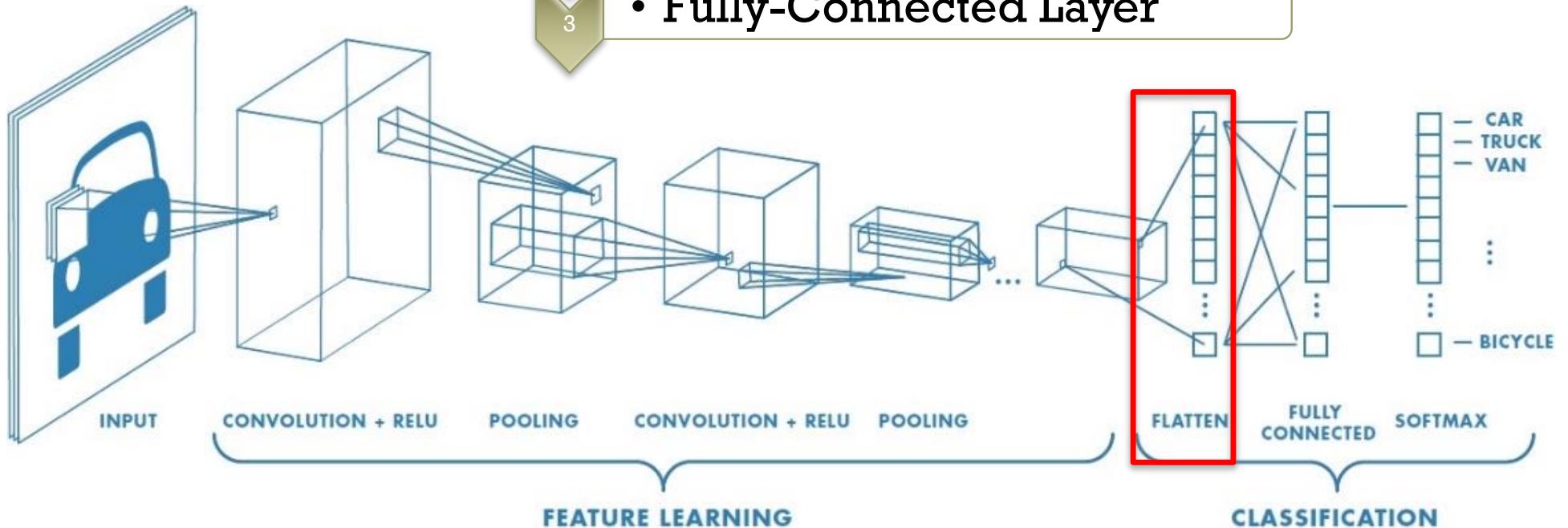
# Convolutional Neural Networks

## Embedding Vector

x1	x2	x3	x4	Corona
0.7	0.2	-0.5	-0.1	Yes



- Convolutional Layer
- Pooling Layer
- Fully-Connected Layer

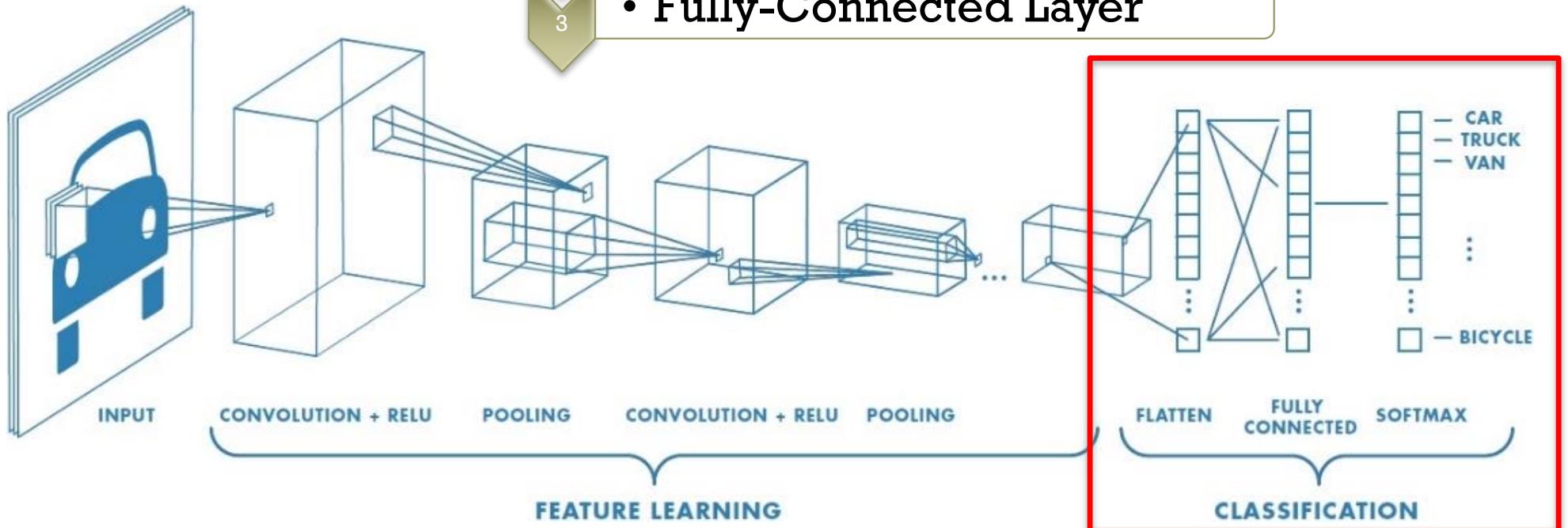




# Convolutional Neural Networks (CNN)

## 3) Fully-Connected Layer (FC) = Neural Networks

- Convolutional Layer
- Pooling Layer
- Fully-Connected Layer



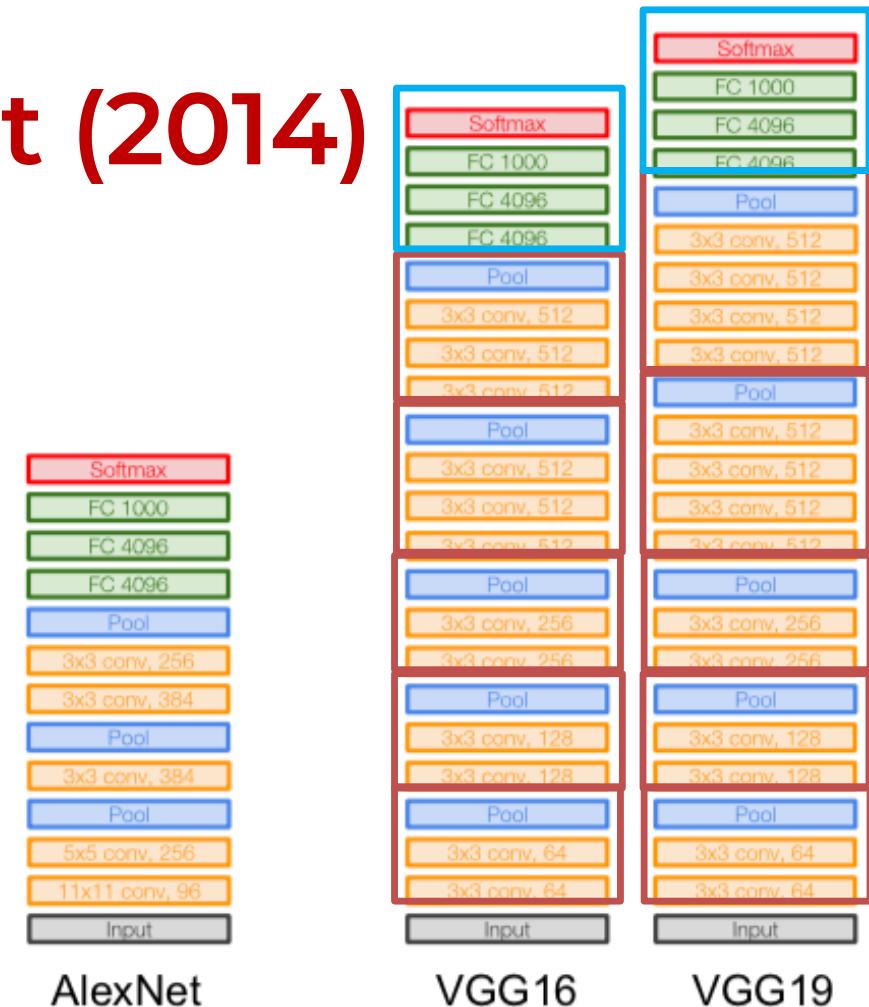
# CNN Example: VGGNet (2014)

[Simonyan and Zisserman, 2014]

Small filters, Deeper networks

8 layers (AlexNet) → 16-19 layers (VGG16Net)

Only 3x3 CONV Stride 1, pad 1  
And 2x2 MAX POOL stride 2



Network	A	B	C	D	E
Number of parameters (Millions)	133	133	134	138	144

# TORCHVISION.MODELS

The models subpackage contains definitions of models for addressing different tasks, including: image classification, pixelwise semantic segmentation, object detection, instance segmentation, person keypoint detection and video classification.

## Classification

The models subpackage contains definitions for the following model architectures for image classification:

- [AlexNet](#)
- [VGG](#)
- [ResNet](#)
- [SqueezeNet](#)
- [DenseNet](#)
- [Inception v3](#)
- [GoogLeNet](#)
- [ShuffleNet v2](#)
- [MobileNetV2](#)
- [MobileNetV3](#)
- [ResNeXt](#)
- [Wide ResNet](#)
- [MNASNet](#)

We provide pre-trained models, using the PyTorch `torch.utils.model_zoo`. These can be constructed by passing `pretrained=True`:

```
import torchvision.models as models
resnet18 = models.resnet18(pretrained=True)
alexnet = models.alexnet(pretrained=True)
squeezenet = models.squeeze1_0(pretrained=True)
vgg16 = models.vgg16(pretrained=True)
densenet = models.densenet161(pretrained=True)
inception = models.inception_v3(pretrained=True)
googlenet = models.googlenet(pretrained=True)
shufflenet = models.shufflenet_v2_x1_0(pretrained=True)
mobilenet_v2 = models.mobilenet_v2(pretrained=True)
mobilenet_v3_large = models.mobilenet_v3_large(pretrained=True)
mobilenet_v3_small = models.mobilenet_v3_small(pretrained=True)
resnext50_32x4d = models.resnext50_32x4d(pretrained=True)
wide_resnet50_2 = models.wide_resnet50_2(pretrained=True)
mnasnet = models.mnasnet1_0(pretrained=True)
```

# Image Classification Tasks

**Models**

YOLOv8 Detect, Segment and Pose models pretrained on the COCO dataset are available here, as well as YOLOv8 Classify models pretrained on the ImageNet dataset. Track mode is available for all Detect, Segment and Pose models.

All Models download automatically from the latest Ultralytics release on first use.

## Classification



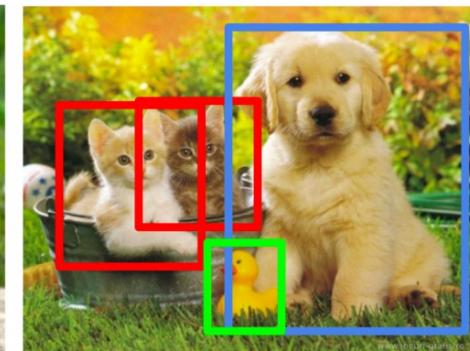
CAT

## Classification + Localization



CAT

## Object Detection



CAT, DOG, DUCK

## Instance Segmentation



CAT, DOG, DUCK

Single object

Multiple objects

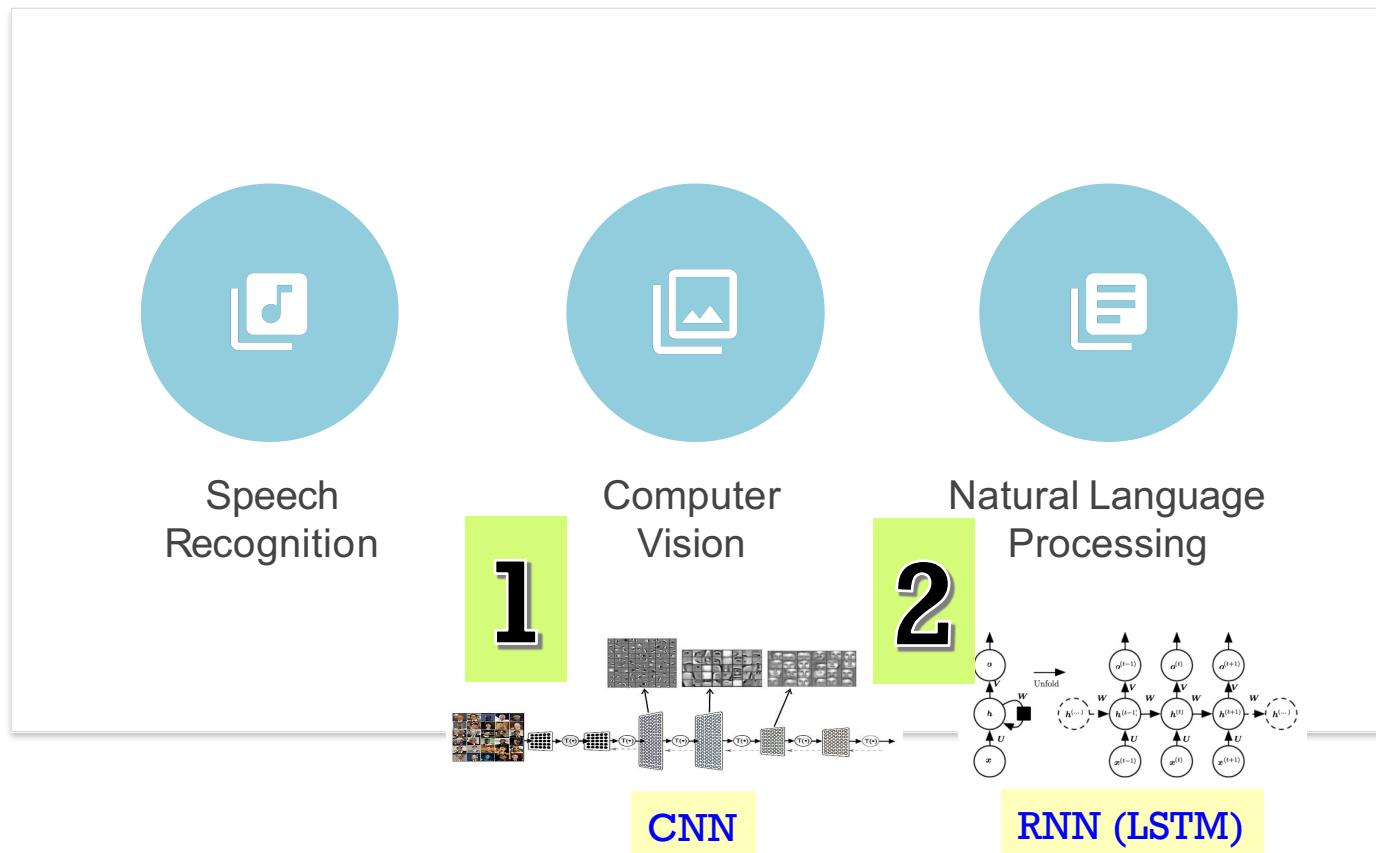
<https://analyticsindiamag.com/top-5-image-classification-research-papers-every-data-scientist-should-know/>



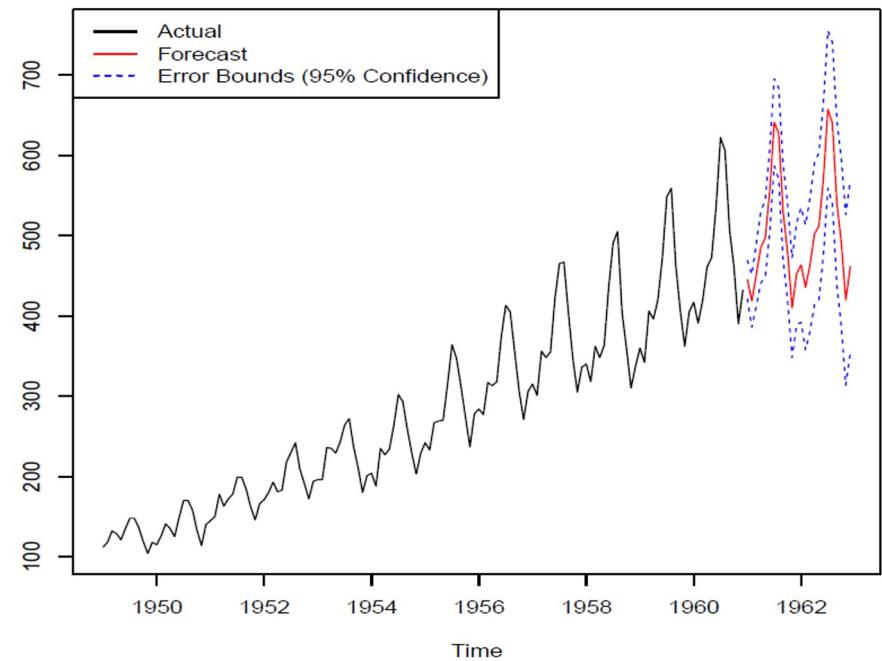
## Recurrent Neural Networks (RNN)

Note: Today's lecture introduces several DL models so that you can understand how to select the right one for your problem. You'll learn more in depth in the AI course next semester.

# Deep Learning Application



# Time Series Forecasting



## Top 10 Strategic Technology Trends for 2020

People-centric      Smart spaces



[gartner.com/SmarterWithGartner](http://gartner.com/SmarterWithGartner)

Source: Gartner  
© 2019 Gartner, Inc. and/or its affiliates. All rights reserved. GTRKT\_364600

Gartner

## Gartner Top Strategic Technology Trends for 2021

### People centricity

### Internet of Behaviors

### Location independence

### Distributed cloud

### Resilient delivery

### Intelligent composable business

### Total experience strategy

### Anywhere operations

### AI engineering

### Privacy-enhancing computing

### Cybersecurity mesh

### Hyperautomation

## Combinatorial innovation

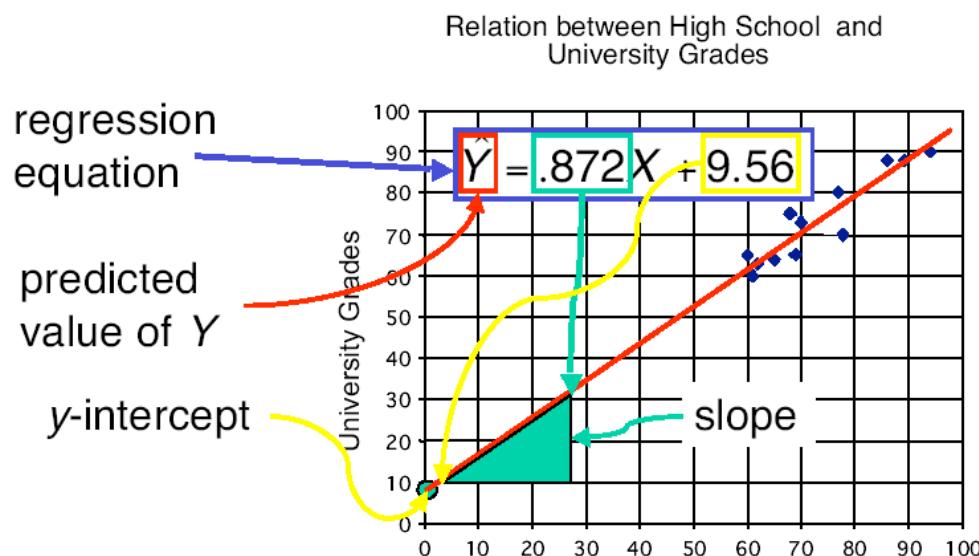
[gartner.com/SmarterWithGartner](http://gartner.com/SmarterWithGartner)

Source: Gartner  
© 2020 Gartner, Inc. and/or its affiliates. All rights reserved. CTMKT\_3026461

Gartner



# 1) Regression – Linear Relationship



weight, coefficient

$$\hat{y} = \hat{w}_0 + \hat{w}_1 x_1 + \hat{w}_2 x_2$$

target      intercept      input

- The least square method aims to minimize the following term

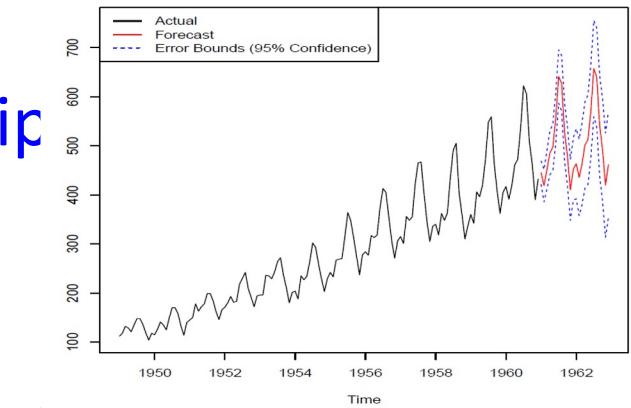
$$\sum_{\text{training data}} (y_i - \hat{y}_i)^2$$

## 2) Autoregressive Model – Linear Relationship

- Based on linear regression, but using previous timestep data

$$X_t = c + \sum_{i=1}^p \varphi_i X_{t-i}$$

- Where  $X_t$  is data at timestep  $t$ ,  $c$  is constant, and each timestep data



are parameters for  
 $\varphi_1, \varphi_2, \varphi_3, \dots, \varphi_p$

Example: When  $p = 2$  (looking back two steps), the equation will be

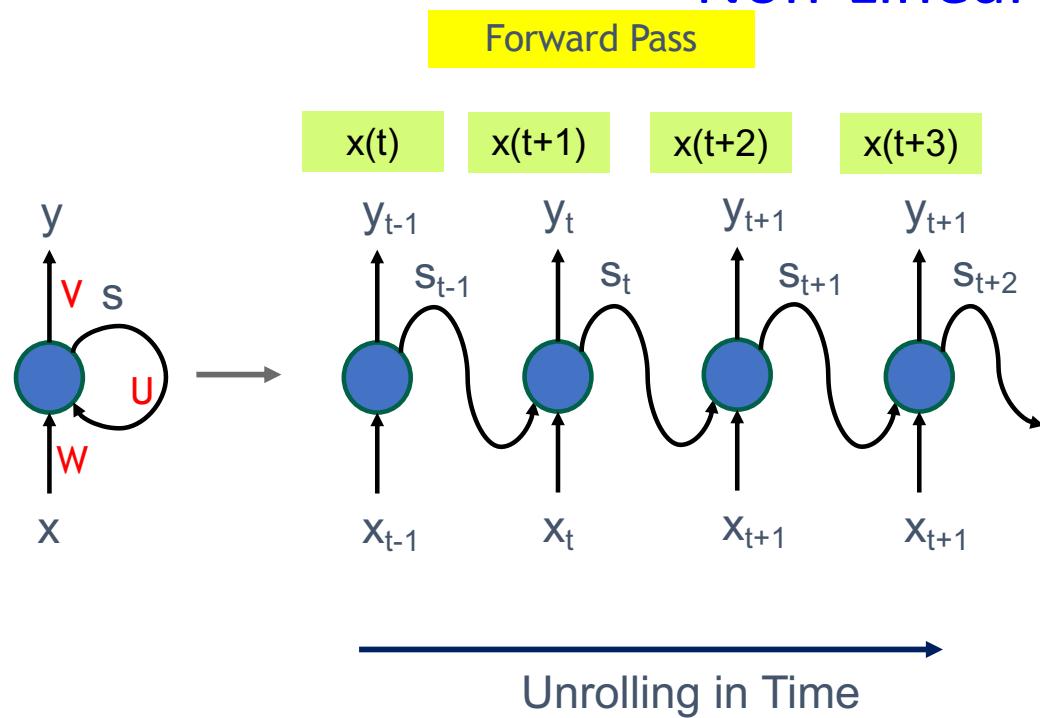
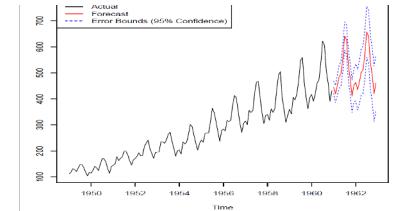
$$X_t = c + \varphi_1 X_{t-1} + \varphi_2 X_{t-2}$$

- Parameters are able to calculate using various method, such as ordinary least square procedure or Yule–Walker equations

$$x(t+1) = \hat{w}_0 + \hat{w}_1 x(t) + \hat{w}_2 x(t-1)$$

### 3) Recurrent Neural Networks (RNN)

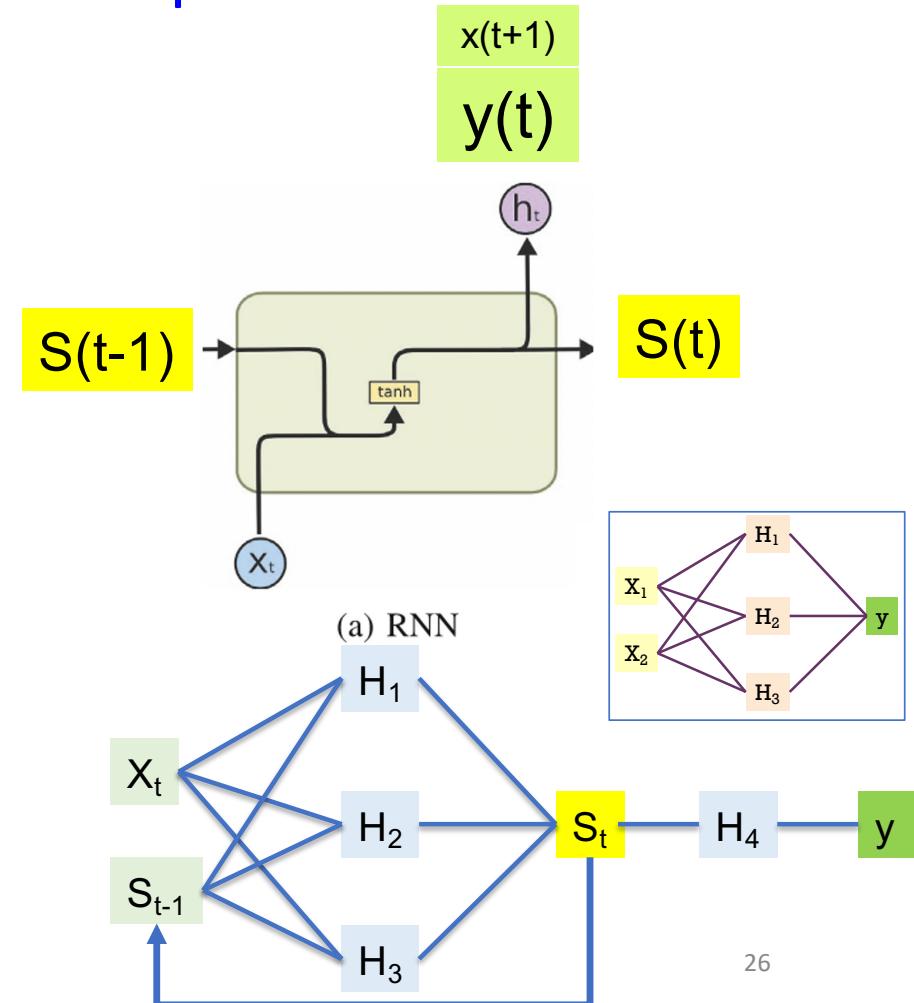
#### Non-Linear Relationship



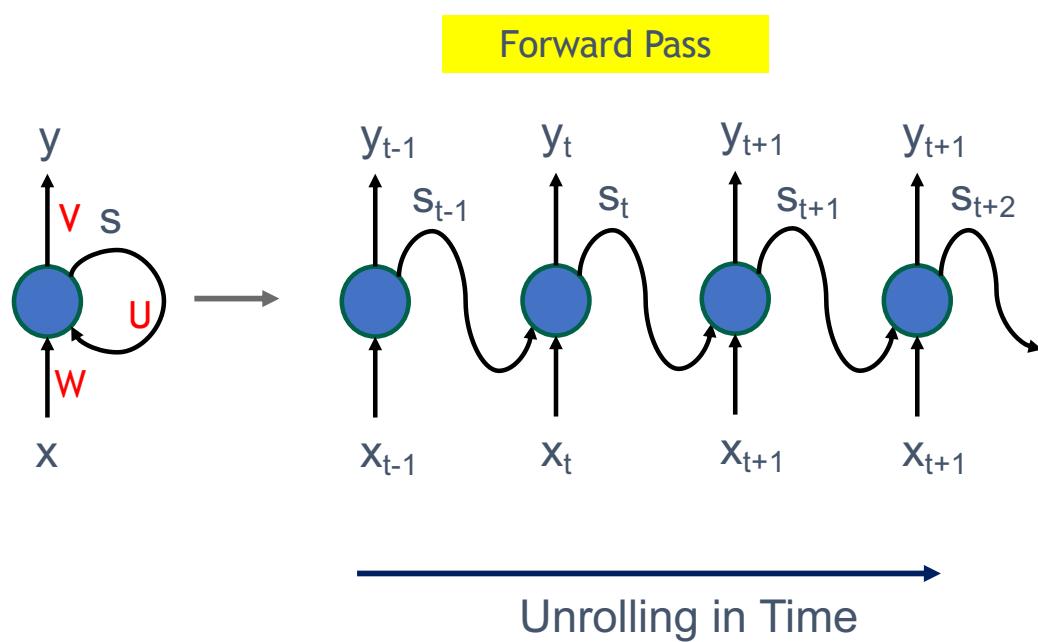
$$s_t = \tanh(Ux_t + Vs_{t-1})$$

$$x(t+1) \hat{y}_t = \text{softmax}(Vs_t)$$

Reference: <https://www.cse.iitk.ac.in/users/sigml/lec/Slides/LSTM.pdf>

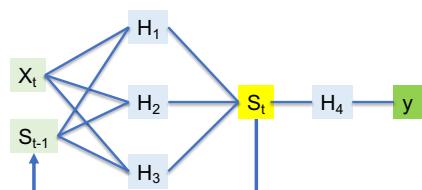


### 3) Recurrent Neural Networks (RNN)

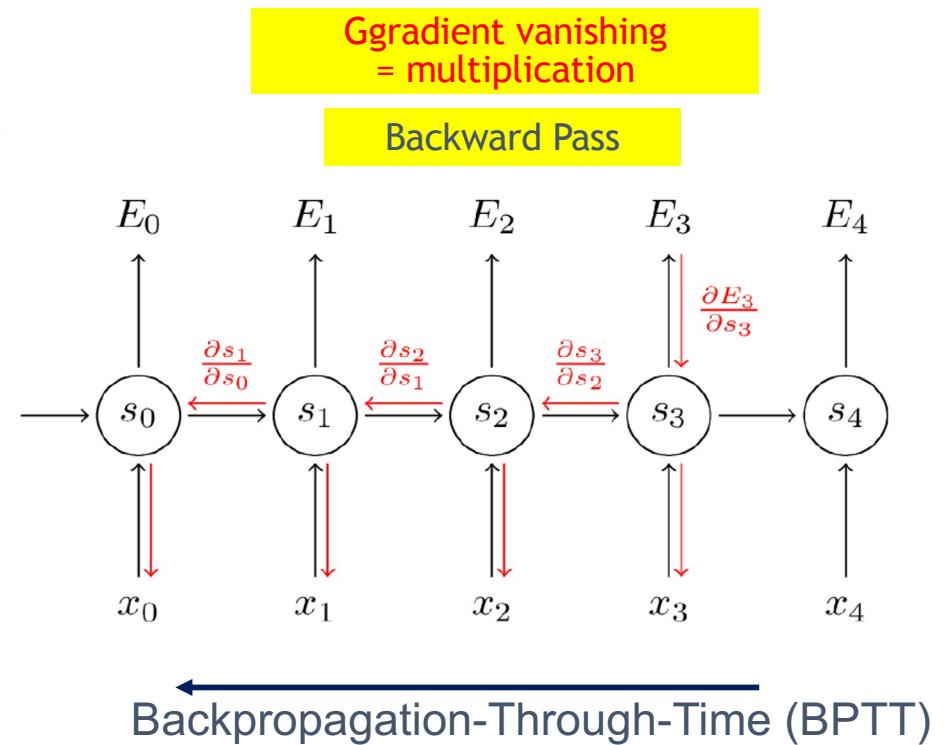


$$s_t = \tanh(Ux_t + Vs_{t-1})$$

$$\hat{y}_t = \text{softmax}(Vs_t)$$



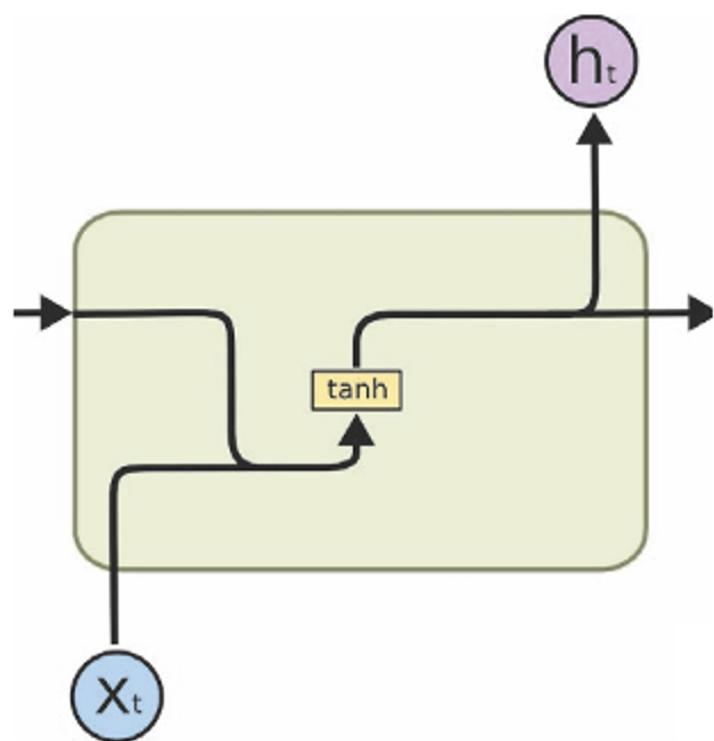
Reference: <https://www.cse.iitk.ac.in/users/sigml/lec/Slides/LSTM.pdf>



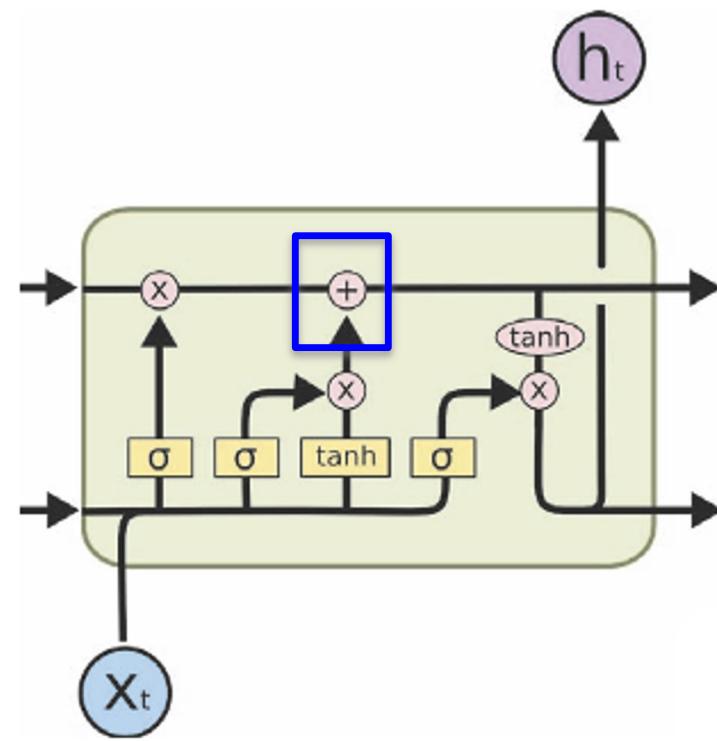
$$\frac{\partial E}{\partial W} = \sum_t \frac{\partial E_t}{\partial W}$$

$$\frac{\partial E_3}{\partial W} = \sum_{k=0}^3 \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \frac{\partial s_3}{\partial s_k} \frac{\partial s_k}{\partial W}$$

## 4) Long-Short Term Memory (LSTM)



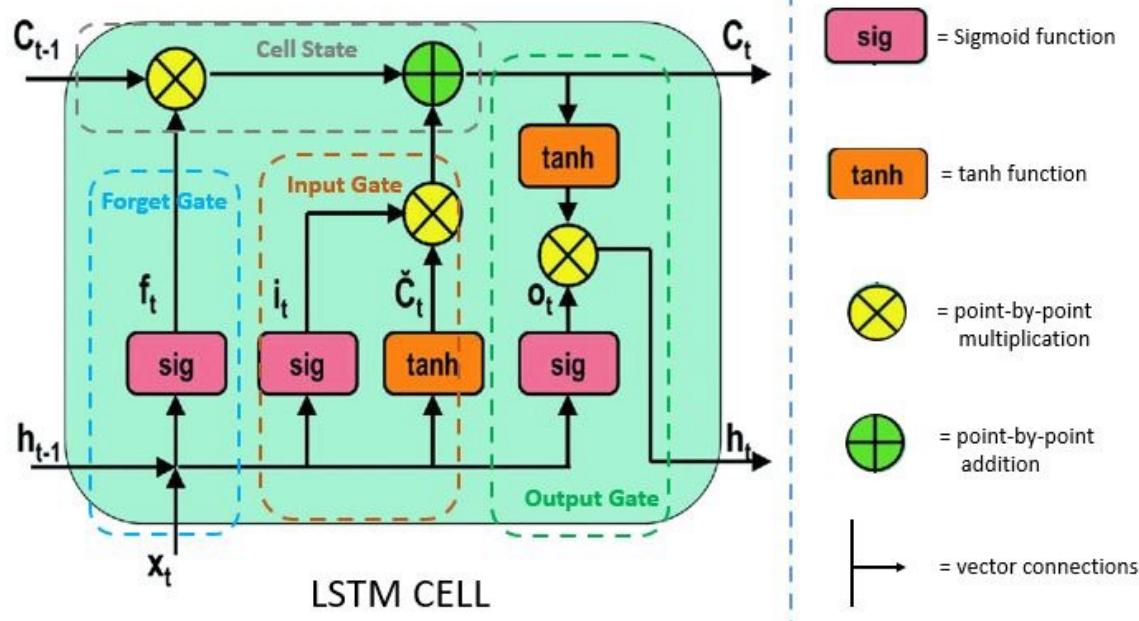
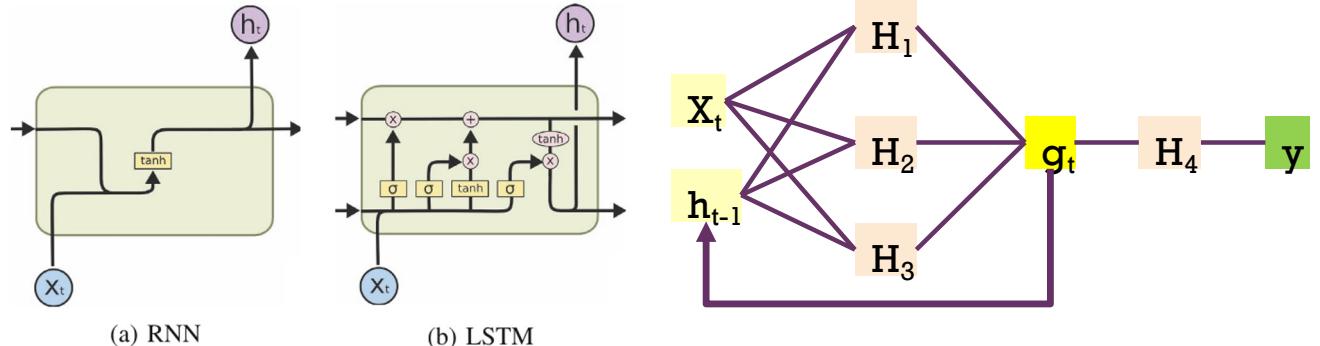
(a) RNN



(b) LSTM

Reference: <https://cs.uwaterloo.ca/~mli/Deep-Learning-2017-Lecture6RNN.ppt>

# Inside LSTM



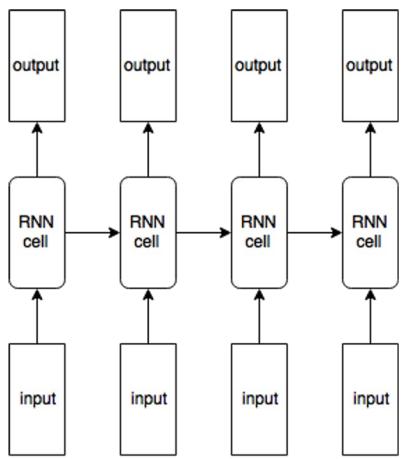
There are 3 gates with 2 outputs.

$$\begin{aligned}
 \mathbf{i}_{(t)} &= \sigma(\mathbf{W}_{xi}^T \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hi}^T \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_i) \\
 \mathbf{f}_{(t)} &= \sigma(\mathbf{W}_{xf}^T \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hf}^T \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_f) \\
 \mathbf{o}_{(t)} &= \sigma(\mathbf{W}_{xo}^T \cdot \mathbf{x}_{(t)} + \mathbf{W}_{ho}^T \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_o) \\
 \mathbf{g}_{(t)} &= \tanh(\mathbf{W}_{xg}^T \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hg}^T \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_g) \\
 \mathbf{c}_{(t)} &= \mathbf{f}_{(t)} \otimes \mathbf{c}_{(t-1)} + \mathbf{i}_{(t)} \otimes \mathbf{g}_{(t)} \\
 \mathbf{y}_{(t)} &= \mathbf{h}_{(t)} = \mathbf{o}_{(t)} \otimes \tanh(\mathbf{c}_{(t)})
 \end{aligned}$$

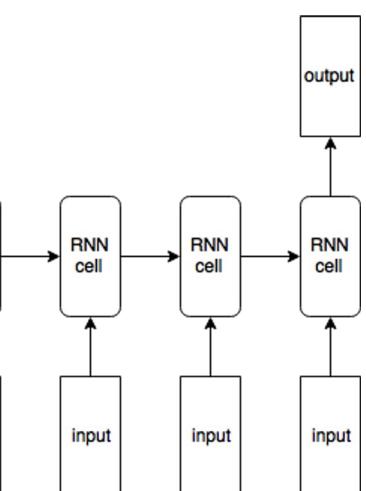


## RNN in NLP Application

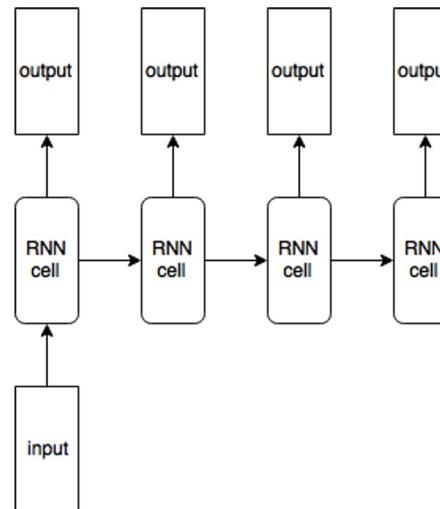
**many-to-many**



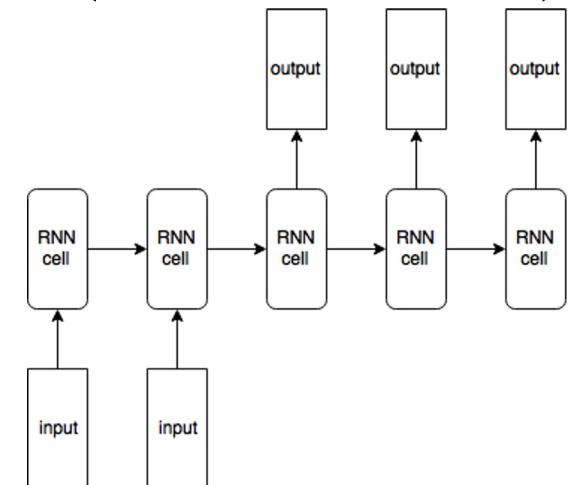
**many-to-one**



**one-to-many**



**many-to-many  
(encoder-decoder)**

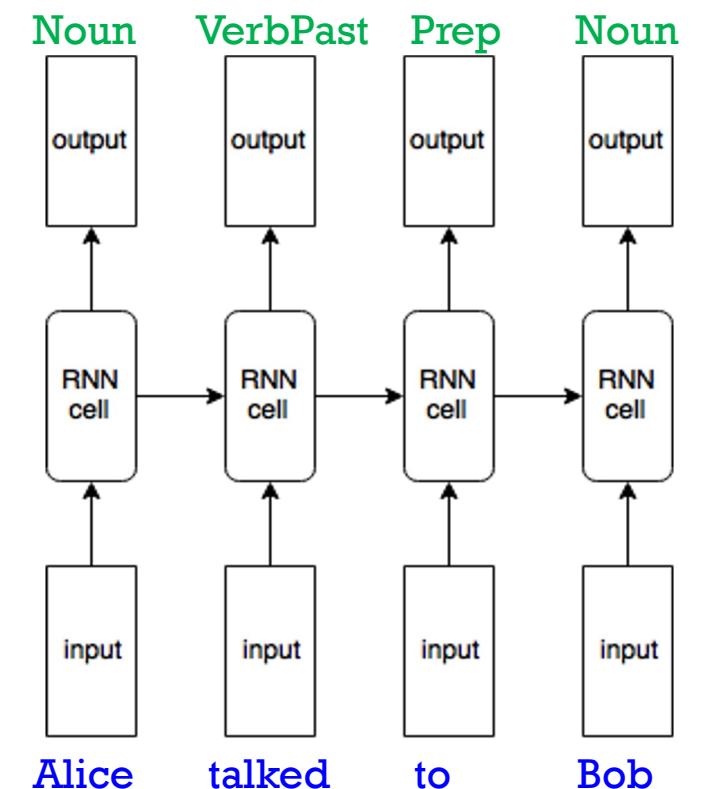
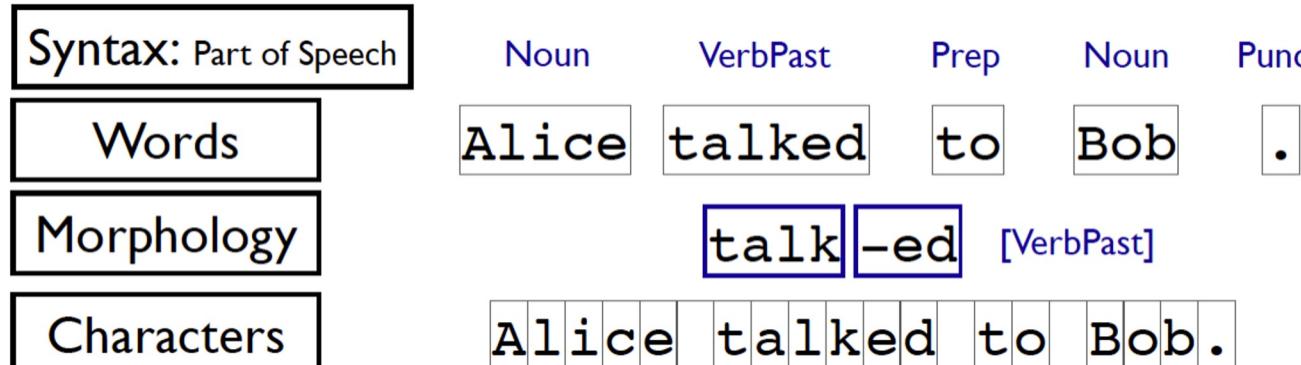


Reference: The Unreasonable Effectiveness of Recurrent Neural Networks, <http://karpathy.github.io/2015/05/21/rnn-effectiveness/>



## 1) Many-to-Many (e.g., PoS Tagging)

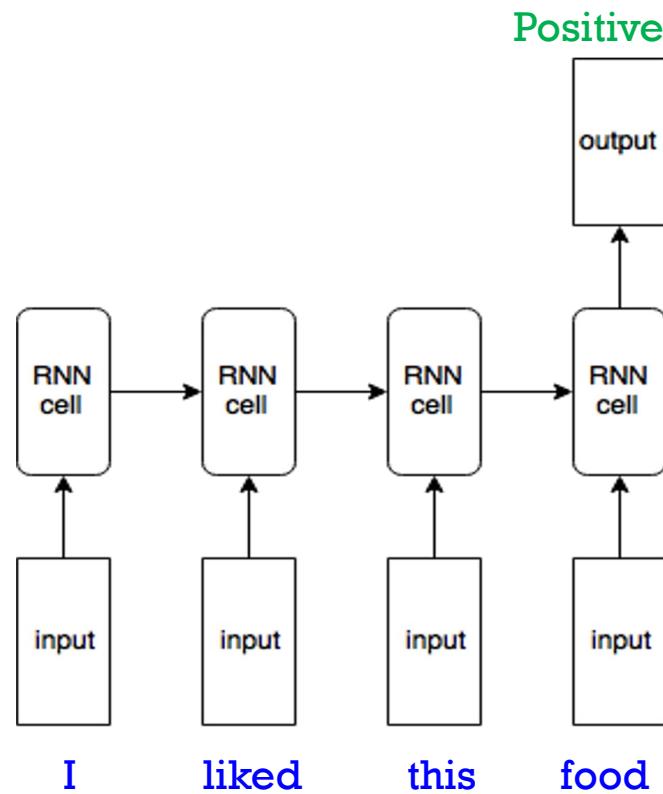
- You have seen and implemented this type of RNN architecture in your homework already.
- E.g., Tokenization, POS tagging
- Sequence Input, Sequence Output





## 2) Many-to-One (e.g., Text Classification)

- E.g., sentiment analysis, text classification
- Sequence input, scalar output

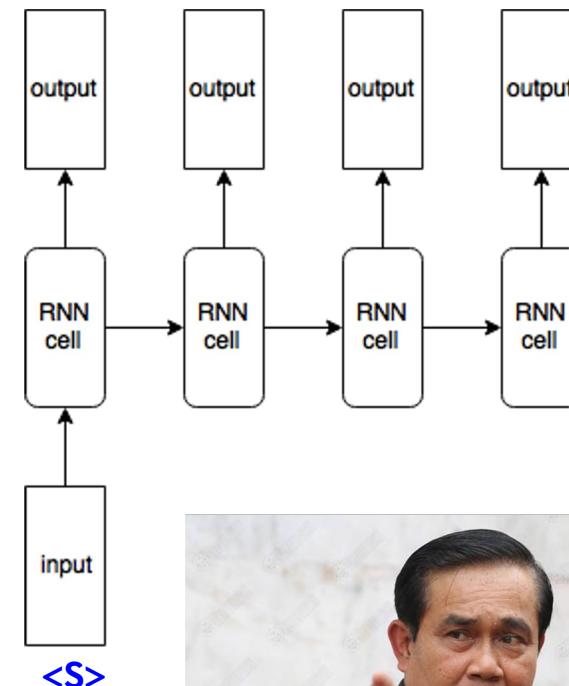




### 3) One-to-Many (Text Generation)

- Scalar input & Sequence output
- E.g., Music Generation, Image caption generation
- Music **generation**
  - Input: Initial seed
  - Output: Sequence of music notes
- Image caption **generation**
  - Input: Image features extracted by CNN
  - Output: Sequence of text

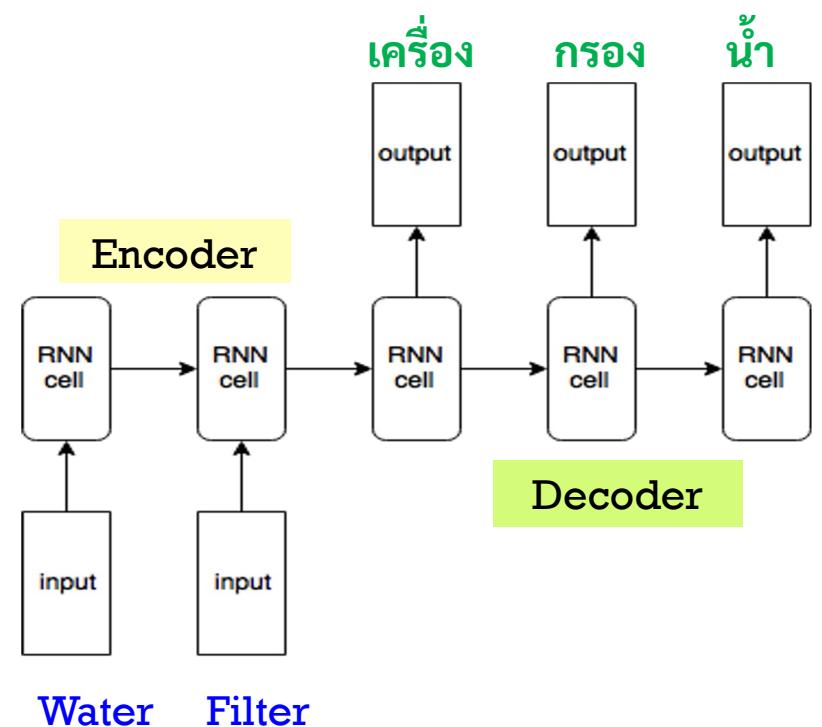
Moving Thailand Forward </S>



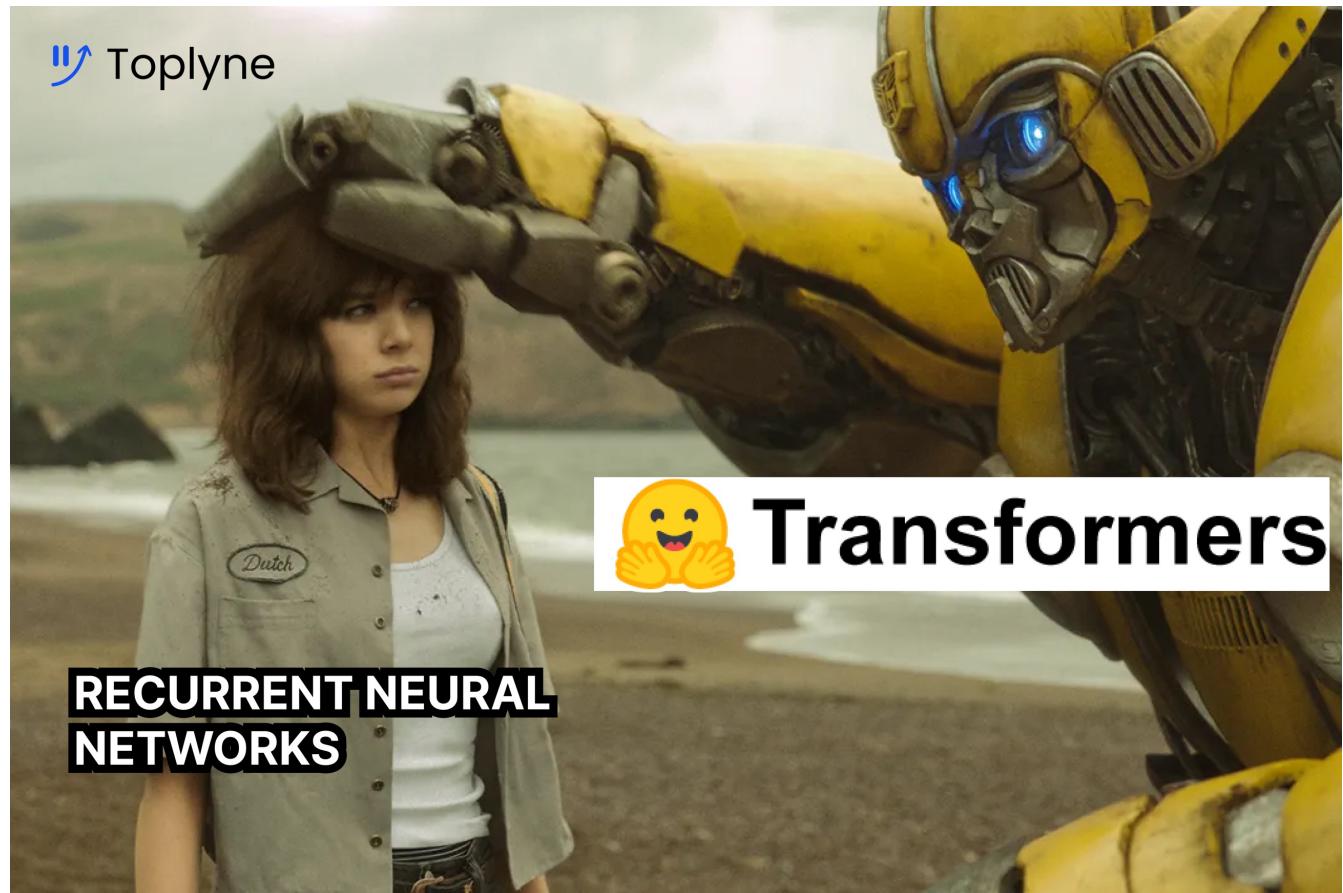
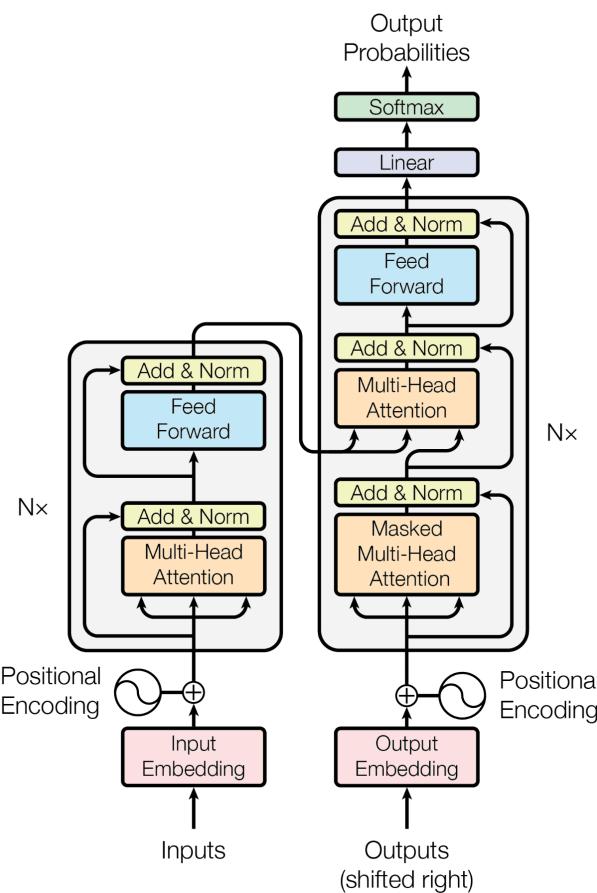


## 4) Many-to-Many (encoder-decoder) (e.g., Machine Translation)

- Sequence Input, Sequence output
- These two sequences can be of different length
- E.g., **Machine Translation**
  - Input: English Sentence
  - Output: Thai Sentence
- Machine Translation is also a **text generation task**



# Rise of Transformer (2017)





## Transformer (TF)

Note: Today's lecture introduces several DL models so that you can understand how to select the right one for your problem. You'll learn more in depth in the AI course next semester.

# Attention mechanism

Published as a conference paper at ICLR 2015

## NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

Dzmitry Bahdanau  
Jacobs University Bremen, Germany

KyungHyun Cho    Yoshua Bengio\*  
Université de Montréal

### ABSTRACT

Neural machine translation is a recently proposed approach to machine translation. Unlike the traditional statistical machine translation, the neural machine translation aims at building a single neural network that can be jointly tuned to maximize the translation performance. The models proposed recently for neural machine translation often belong to a family of encoder-decoders and encode a source sentence into a fixed-length vector from which a decoder generates a translation. In this paper, we conjecture that the use of a fixed-length vector is a bottleneck in improving the performance of this basic encoder-decoder architecture, and propose to extend this by allowing a model to automatically (soft-)search for parts of a source sentence that are relevant to predicting a target word, without having to form these parts as a hard segment explicitly. With this new approach, we achieve a translation performance comparable to the existing state-of-the-art phrase-based system on the task of English-to-French translation. Furthermore, qualitative analysis reveals that the (soft-)alignments found by the model agree well with our intuition.

### 1 INTRODUCTION

Reference: Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." ICLR(2015).

6 Dec 2017

## 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

Lukasz Kaiser\*  
Google Brain  
lukaszkaiser@google.com

Illia Polosukhin\* ‡  
illia.polosukhin@gmail.com

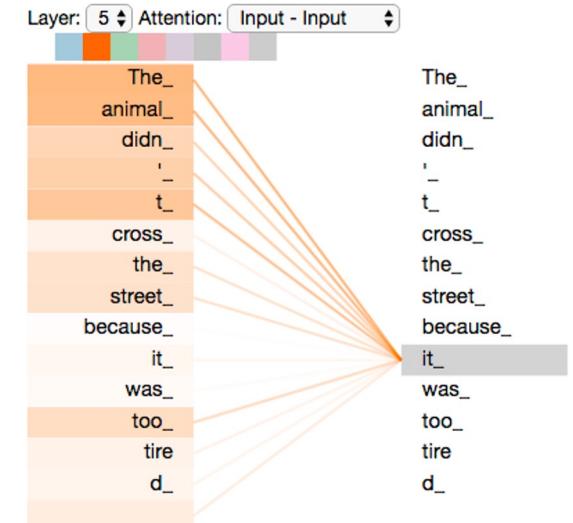
Attention Is All You Need. 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

# + Scaled Dot-Product Attention (1)

- Introduced in Attention is all you need (Vishwani et al., 2017)
- NO** recurrence nor convolution
- Widely used today in all Transformer-based model
- “Relating different positions of a single sequence in order to compute a representation of the sequence”

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

where the query, keys, values, and output are all vectors and  $d_k$  is a number.





## Scaled Dot-Product Attention (2)

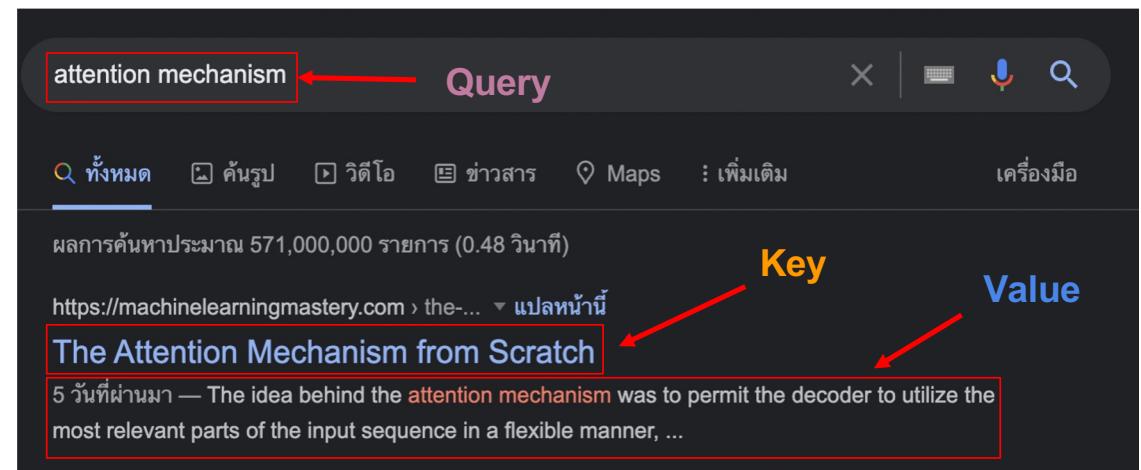
What are the “query”, “key”, and “value” vectors?

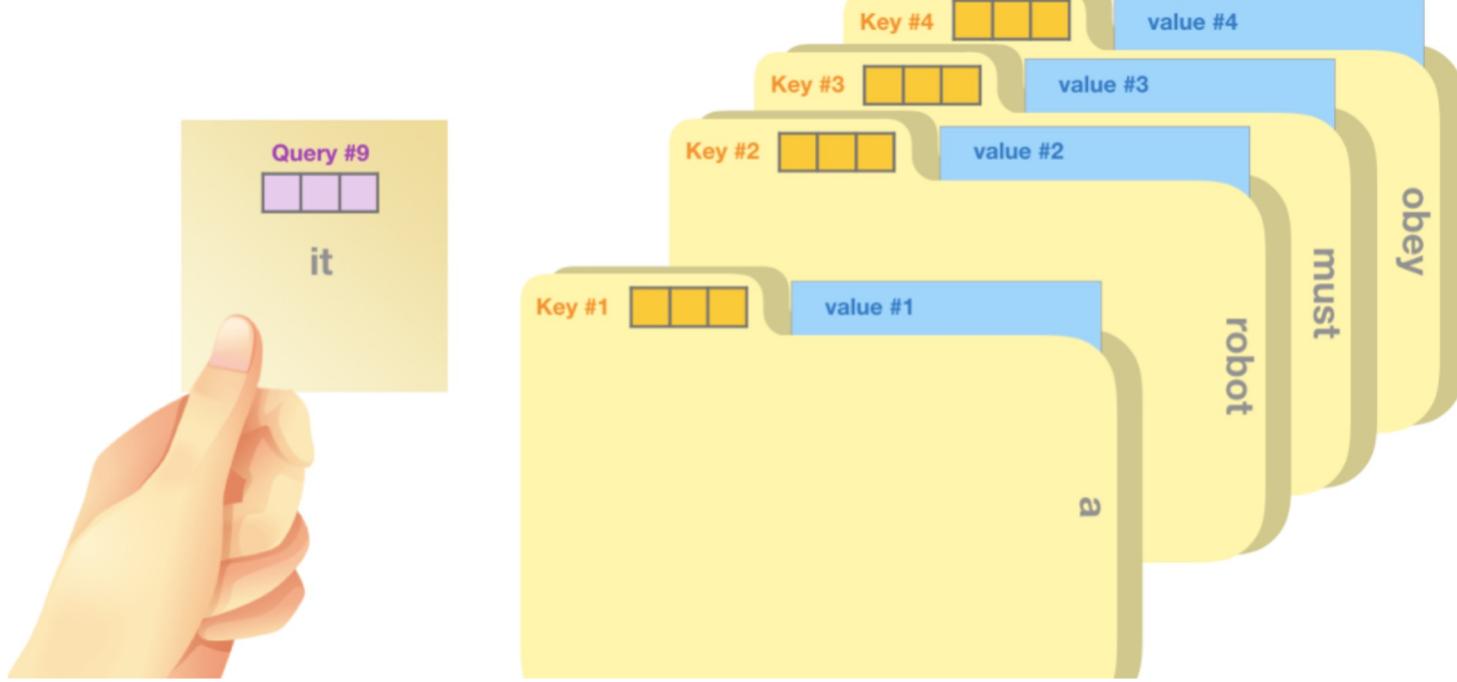
As an analogy, think of Google Search.

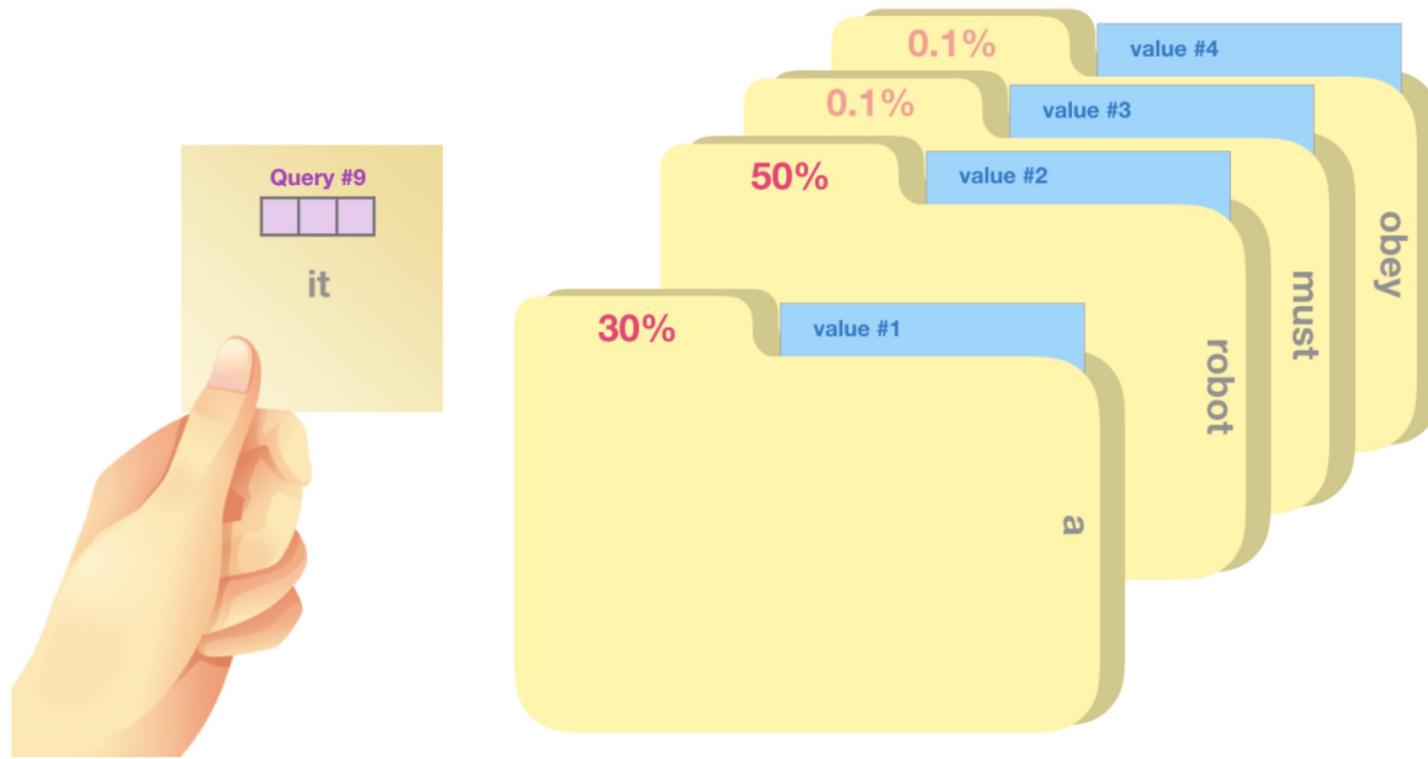
Query - what we want to know

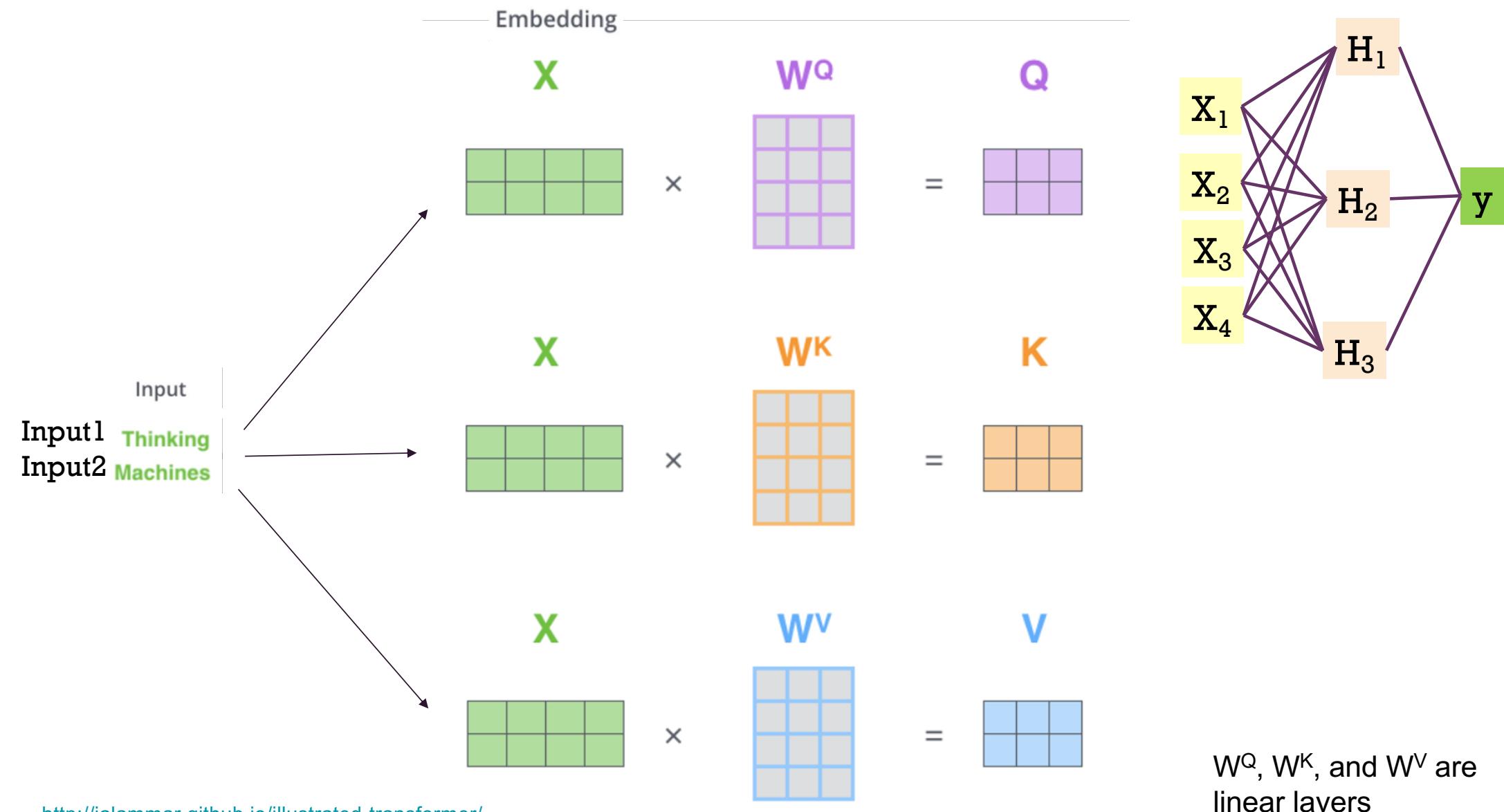
Key - how to index information

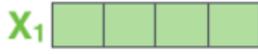
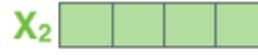
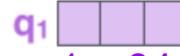
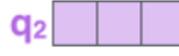
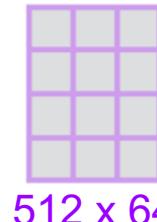
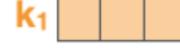
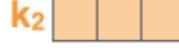
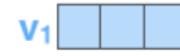
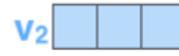
Value - what kind of info is in each website









Input	<b>Thinking</b>	<b>Machines</b>	
Embedding	$x_1$ 	$x_2$ 	
	$1 \times 512$	$1 \times 512$	
Queries	$q_1$ 	$q_2$ 	$W^Q$
	$1 \times 64$	$1 \times 64$	
Keys	$k_1$ 	$k_2$ 	$W^K$
	$1 \times 64$	$1 \times 64$	
Values	$v_1$ 	$v_2$ 	$W^V$
	$1 \times 64$	$1 \times 64$	

Multiplying  $x_1$  by the  $W^Q$  weight matrix produces  $q_1$ , the "query" vector associated with that word. We end up creating a "query", a "key", and a "value" projection of each word in the input sentence.

Input

Embedding

Queries

Keys

Values

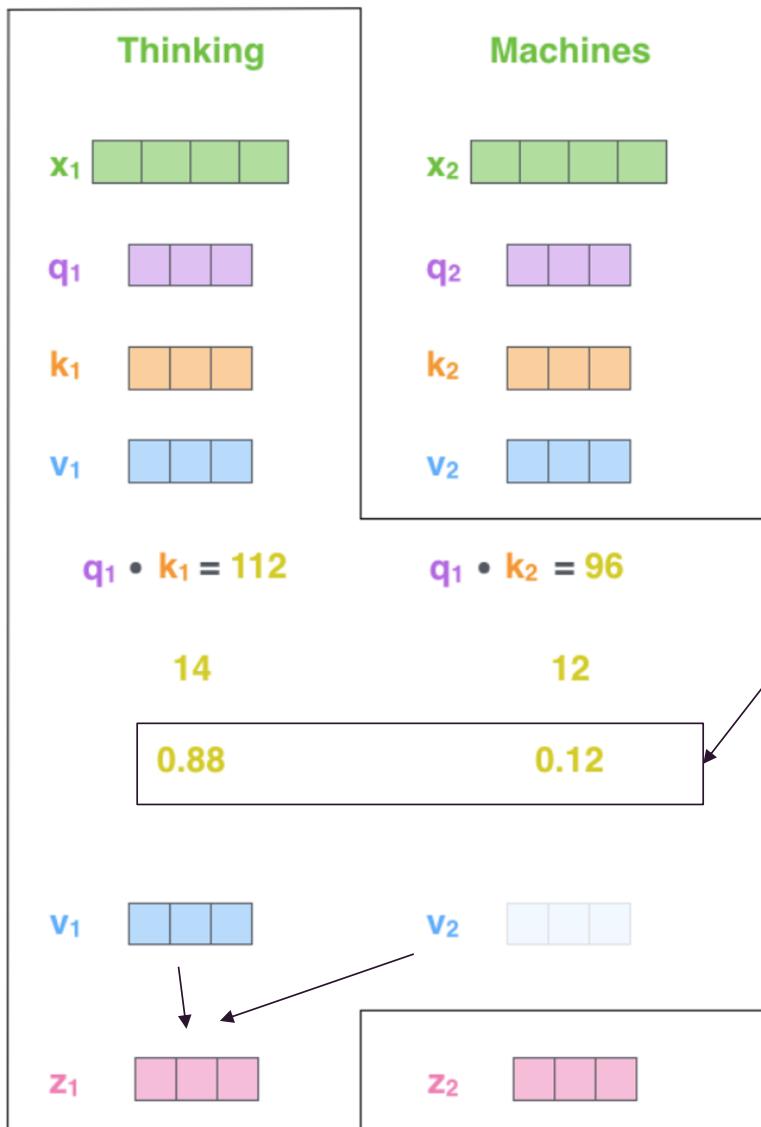
Score

Divide by 8 ( $\sqrt{d_k}$ )

Softmax

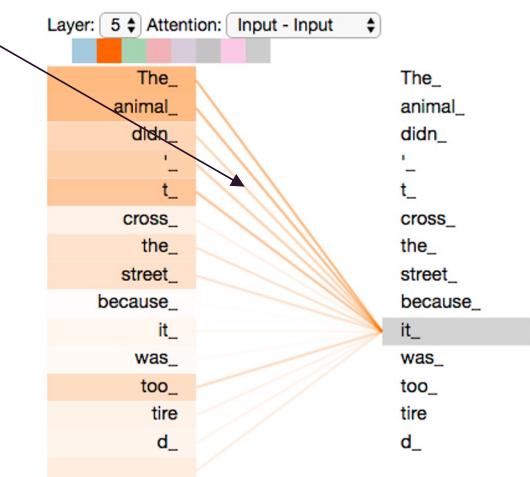
Softmax  
X  
Value

Sum



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

Attention scores

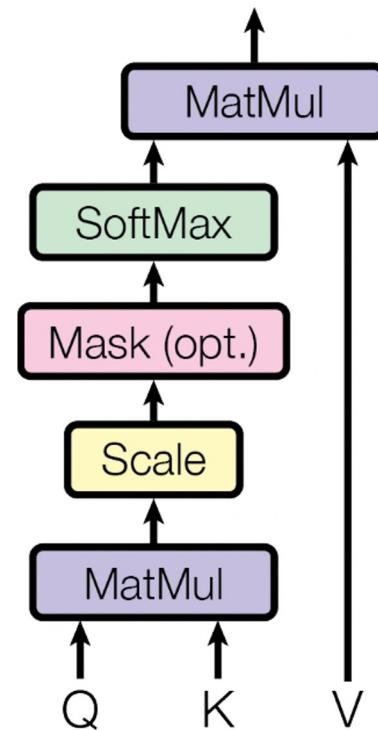


$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$$\text{softmax}\left( \frac{\begin{array}{c} \text{Q} \\ \left[ \begin{array}{cccc} \quad & \quad & \quad & \quad \end{array} \right] \\ \times \\ \left[ \begin{array}{cc} \quad & \quad \end{array} \right] \text{K}^T \end{array}}{\sqrt{d_k}} \right) \text{V}$$

$\text{Z}$   
 $=$   
 $\left[ \begin{array}{cccc} \quad & \quad & \quad & \quad \end{array} \right]$

## Scaled Dot-Product 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

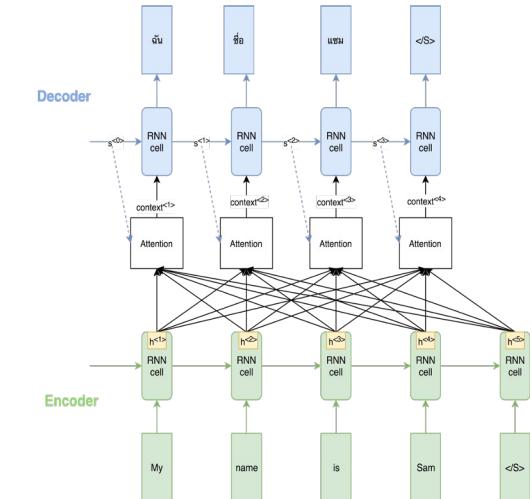
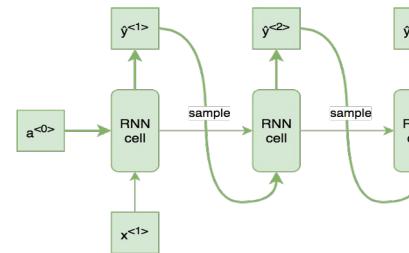
**Illia Polosukhin\*** ‡

illia.polosukhin@gmail.com

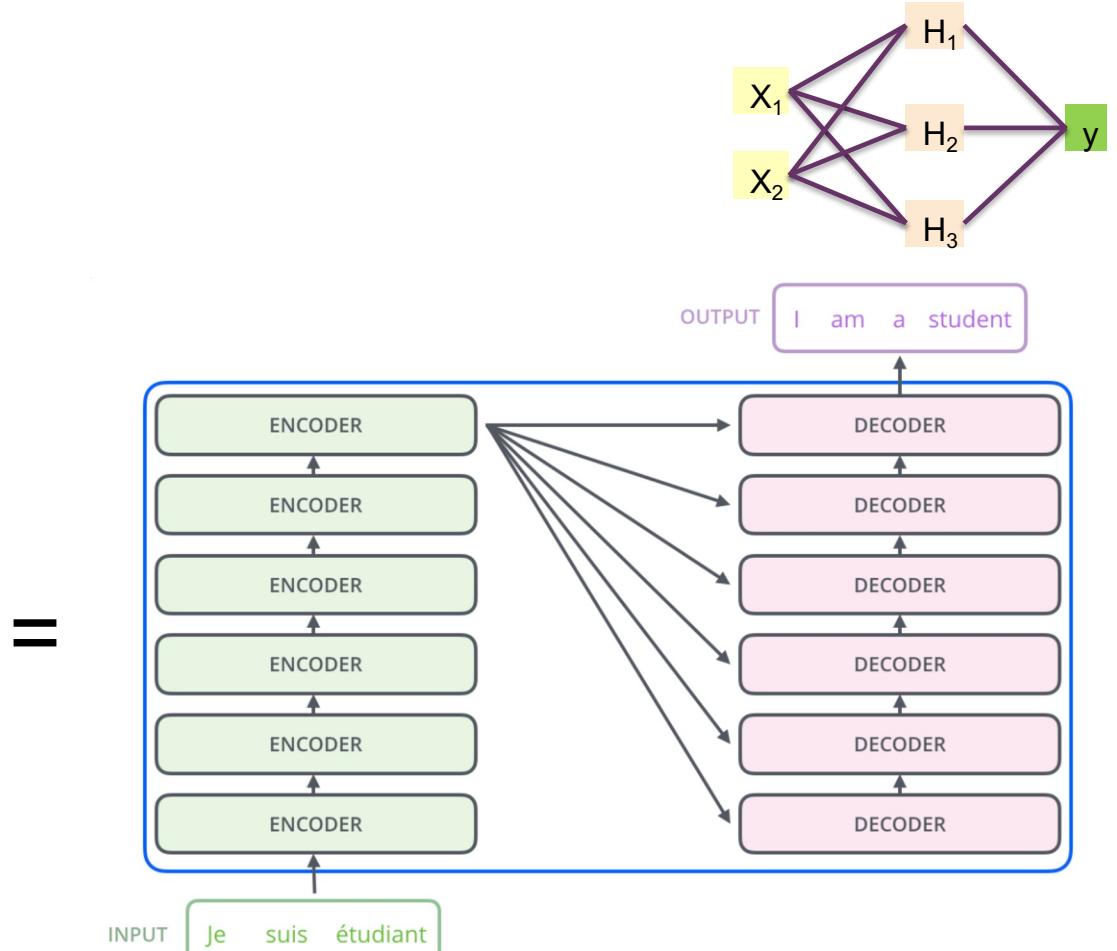
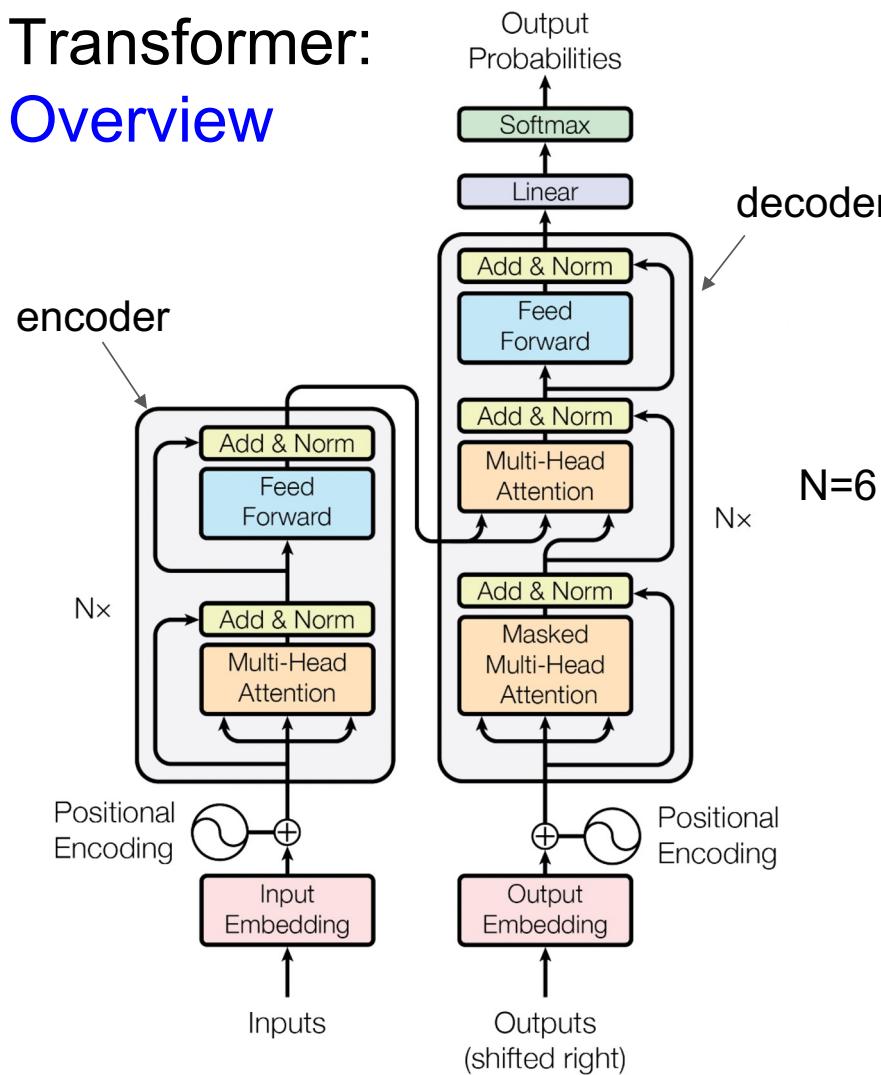
# Transformer

Only rely on the attention mechanism - **No RNN!**

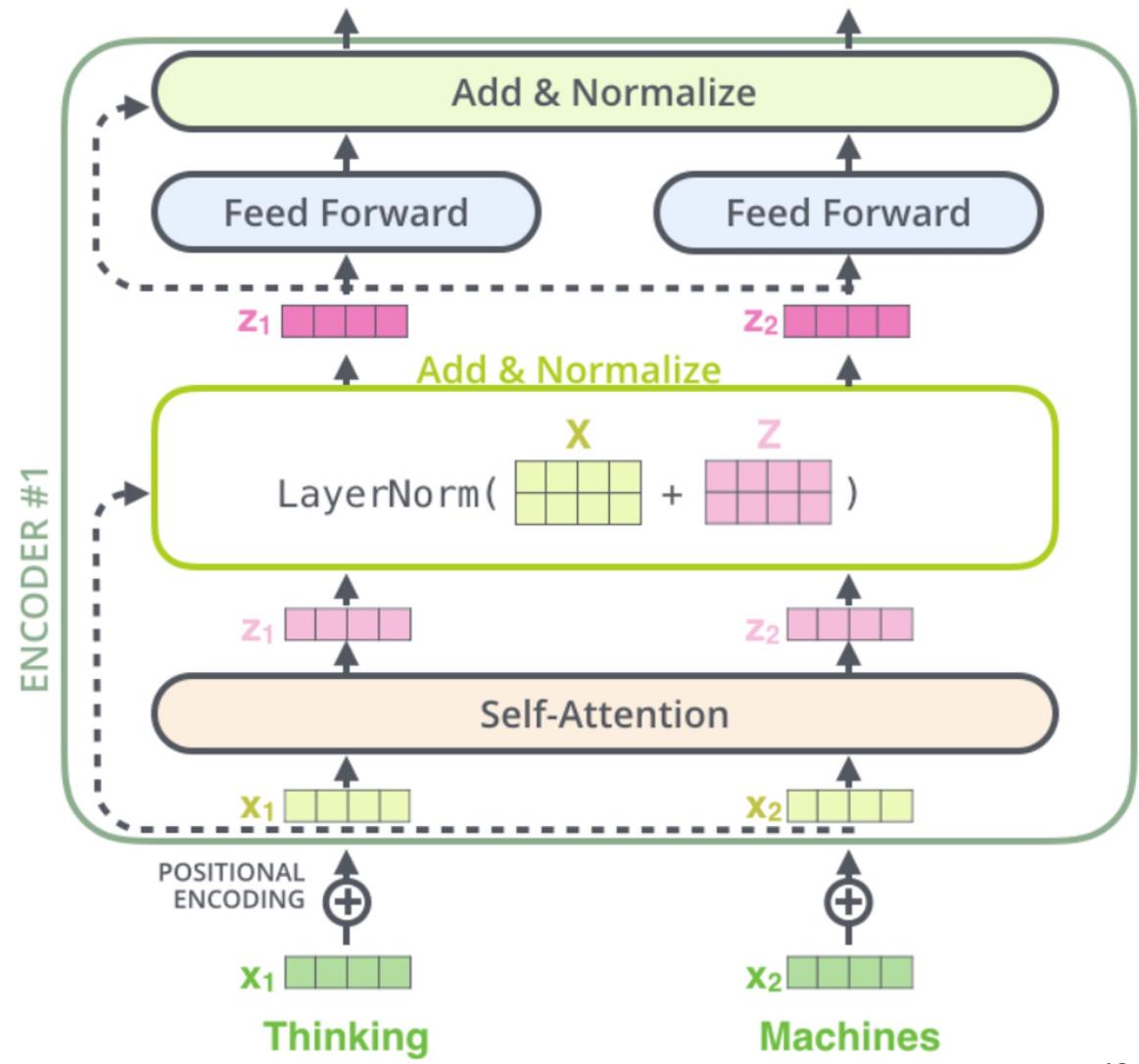
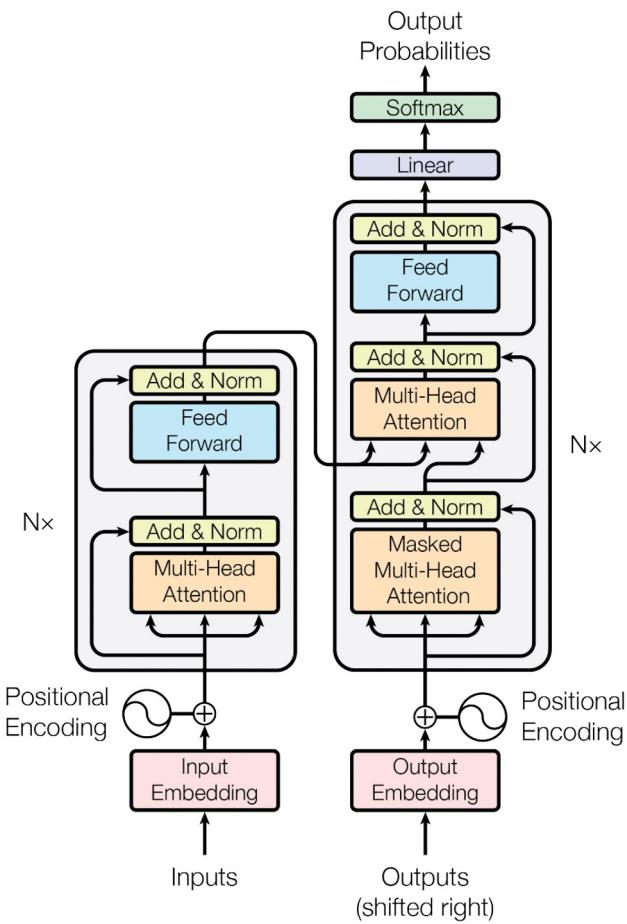
- More parallelizable -> Faster training time
- Better at longer sequence



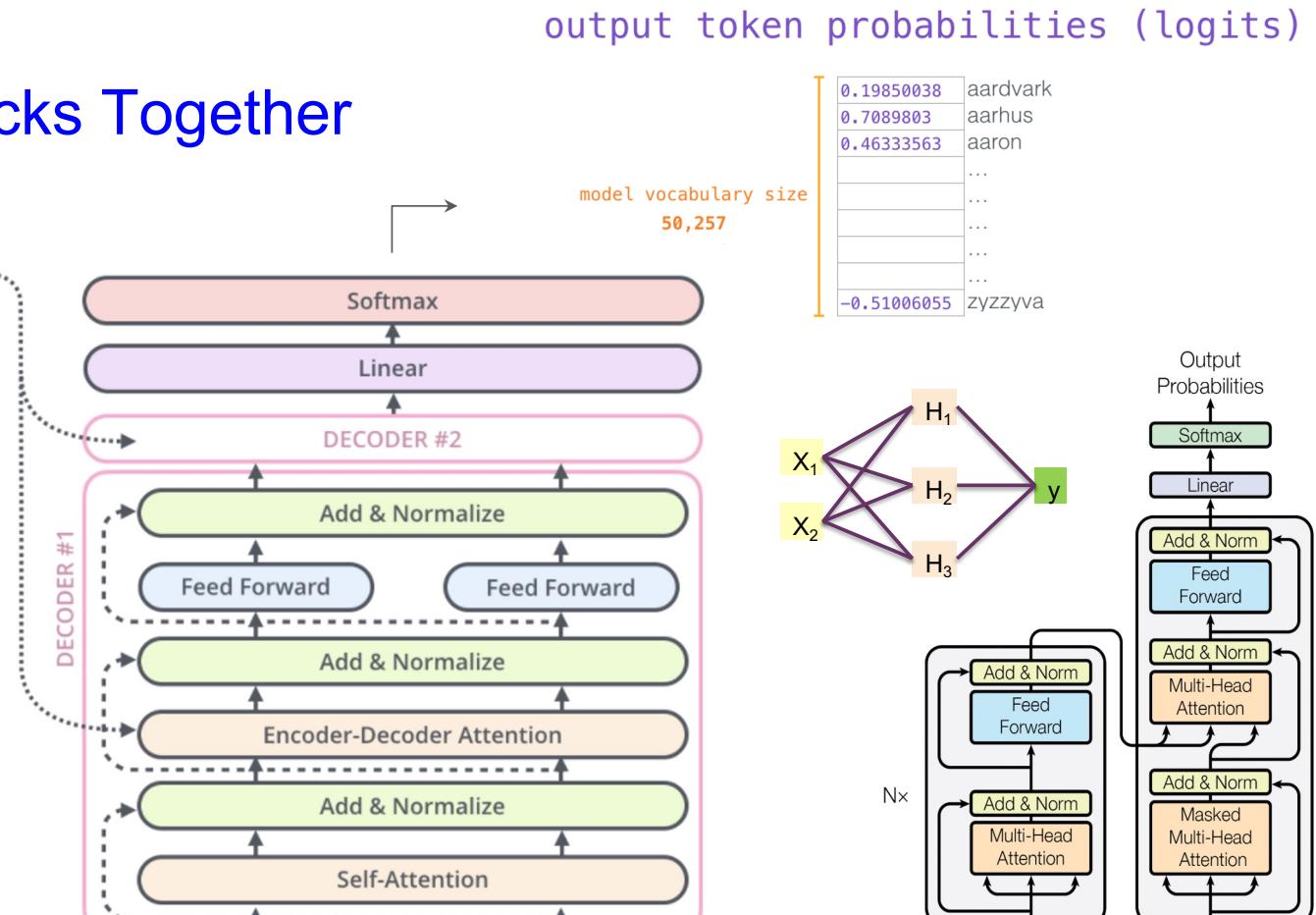
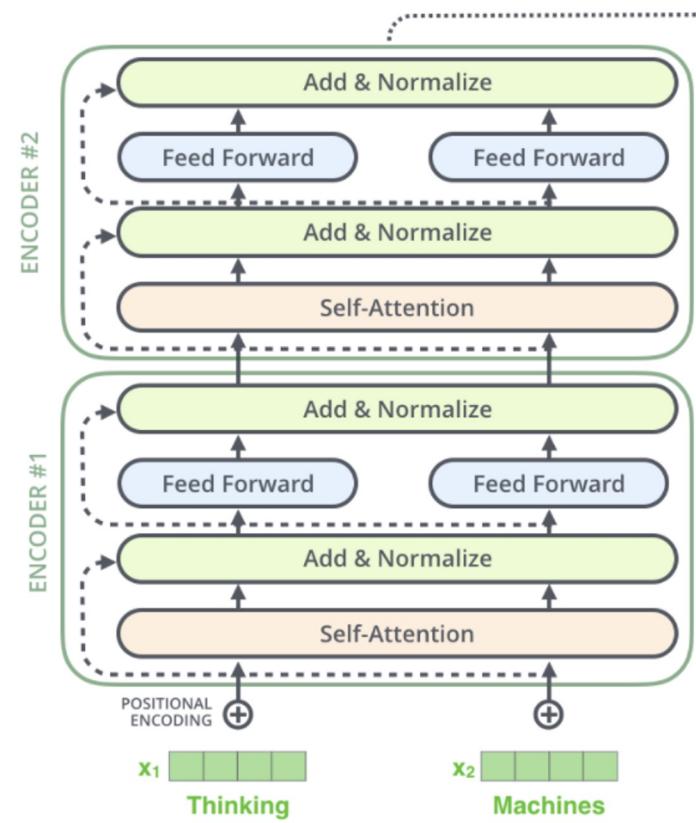
# Transformer: Overview



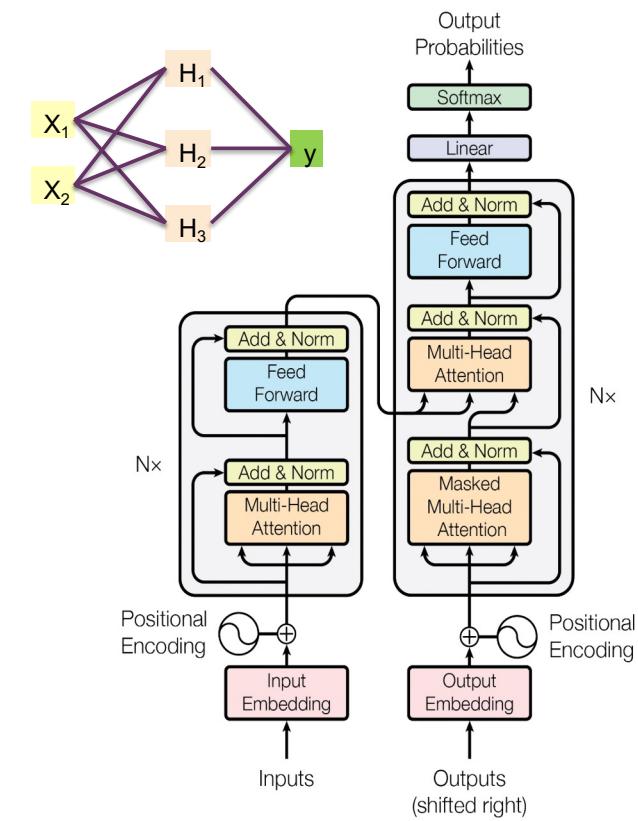
# Transformer Block



# Transformer: Put All Blocks Together



0.19850038	aardvark
0.7089803	aarhus
0.46333563	aaron
...	...
...	...
...	...
-0.51006055	zyzzyva



# Transformer-Based Models

1. Decoder-based model: OpenAI GPT (Generative Pre-Training) [Radford, 2018]
2. Encoder-based model: BERT [Devlin, et al, 2018]:
  - o Bidirectional Encoder Representation from Transformers
3. Encoder and Decoder: BART [2019]

**Improving Language Understanding by Generative Pre-Training**

---

Alec Radford      Karthik Narasimhan      Tim Salimans      Ilya Sutskever  
OpenAI      OpenAI      OpenAI      OpenAI  
alec@openai.com      karthikn@openai.com      tim@openai.com      ilyasu@openai.com

**Abstract**

Natural language understanding comprises a wide range of diverse tasks such as textual entailment, question answering, semantic similarity assessment, and document classification. Although large unlabeled text corpora are abundant, labeled data for learning these specific tasks is scarce, making it challenging for discriminatively trained models to perform adequately. We demonstrate that large gains on these tasks can be realized by *generative pre-training* of a language model on a diverse corpus of unlabeled text, followed by *discriminative fine-tuning* on each specific task. In contrast to previous approaches, we make use of task-aware input transformations during fine-tuning to achieve effective transfer while requiring minimal changes to the model architecture. We demonstrate the effectiveness of our approach on a wide range of benchmarks for natural language understanding. Our general task-agnostic model outperforms discriminatively trained models that

**BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding**

Jacob Devlin    Ming-Wei Chang    Kenton Lee    Kristina Toutanova  
Google AI Language  
{jacobdevlin, mingweichang, kentonl, kristout}@google.com

**Abstract**

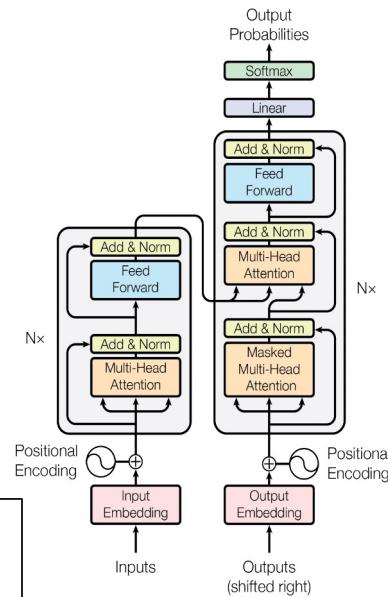
We introduce a new pre-training model for natural language processing, called Bidirectional Encoder Representations from Transformers (BERT). BERT is a deep bidirectional transformer model that takes unlabeled text as input and generates representations for every word in the sequence, capturing both left and right context. We train deep BERT on a large unlabeled dataset and show that it can be fine-tuned on a variety of downstream NLP tasks, including classification, question answering, and reading comprehension. BERT outperforms state-of-the-art models on several benchmarks, including GLUE, SQuAD, and CoQA. We also show that BERT can be used for generating text, which matches the performance of state-of-the-art generative models like GPT-2. BERT is a general-purpose model that can be used for a wide range of NLP tasks, and its pre-training scheme can be applied to other models and datasets.

**BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension**

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, Luke Zettlemoyer

We present BART, a denoising autoencoder for pretraining sequence-to-sequence models. BART is trained by (1) corrupting text with an arbitrary noising function, and (2) learning a model to reconstruct the original text. It uses a standard Transformer-based neural machine translation architecture which, despite its simplicity, can be seen as generalizing BERT (due to the bidirectional encoder), GPT (with the left-to-right decoder), and many other more recent pretraining schemes. We evaluate a number of noising approaches, finding the best performance by both randomly shuffling the order of the original sentences and using a novel in-filling scheme, where spans of text are replaced with a single mask token. BART is particularly effective when fine tuned for text generation but also works well for comprehension tasks. It matches the performance of RoBERTa with comparable training resources on GLUE and SQuAD, achieves new state-of-the-art results on a range of abstractive dialogue, question answering, and summarization tasks, with gains of up to 6 ROUGE. BART also provides a 1.1 BLEU increase over a back-translation system for machine translation, with only target language pretraining. We also report ablation experiments that replicate other pretraining schemes within the BART framework, to better measure which factors most influence end-task performance.

Subjects: Computation and Language (cs.CL); Machine Learning (cs.LG); Machine Learning (stat.ML)  
Cite as: arXiv:1910.13461 [cs.CL]  
(or arXiv:1910.13461v1 [cs.CL] for this version)  
<https://doi.org/10.48550/arXiv.1910.13461>



# Why Transformer models are so powerful?

- 1. Contextual Understanding, e.g., Thinking Machine
- 2. Pretraining on Massive Data (Transfer Learning)
- 3. Scalable Architecture
- 4. Powerful ecosystem, e.g., Hugging Face

Model	Parameters	Pretraining Data
BERT-Base	110M	Wiki + Books (3.3B words)
BERT-Large	340M	Same as Base
DistilBERT	66M	Same as Base
RoBERTa	125–355M	160 GB (CommonCrawl + Books + Wiki)
ALBERT	12–235M	Same as Base

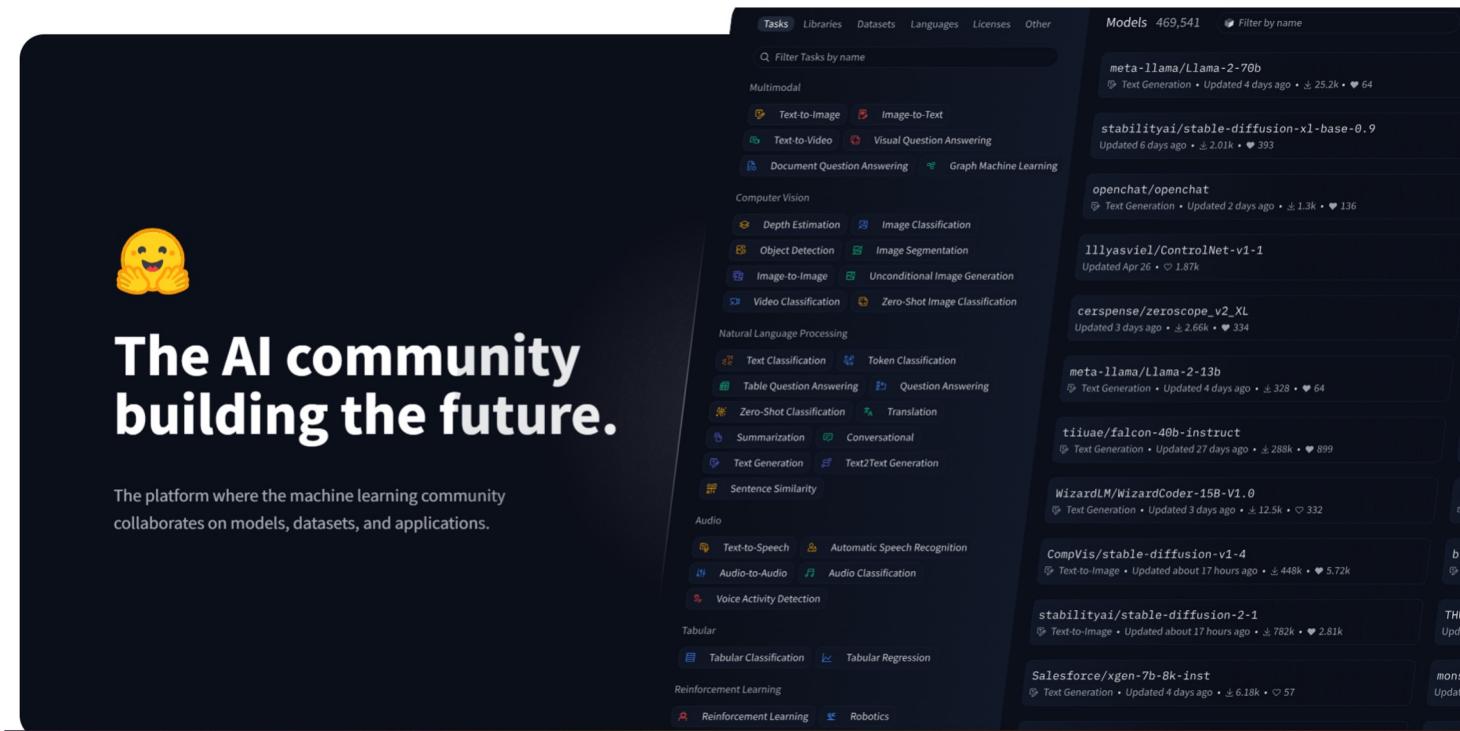


## Hugging Face

Note: Today's lecture introduces several DL models so that you can understand how to select the right one for your problem. You'll learn more in depth in the AI course next semester.

# What is Huggingface?

- HuggingFace is an AI community that promotes open source contributions. It is a hub of open source models for Natural Language Processing, computer vision, and other fields where AI plays its role. Even the tech giants like Google, Facebook, AWS, Microsoft, and others use the models, datasets, and libraries.



# Datasets in Huggingface

The screenshot shows the Huggingface Datasets homepage. On the left, there's a sidebar with categories like Tasks, Sizes, Sub-tasks, Languages, Licenses, and Other. Below these are sections for Multimodal tasks (Feature Extraction, Text-to-Image, Image-to-Text, Text-to-Video, Visual Question Answering, Graph Machine Learning), Computer Vision tasks (Depth Estimation, Image Classification, Object Detection, Image Segmentation, Image-to-Image, Unconditional Image Generation, Video Classification, Zero-Shot Image Classification), and Natural Language Processing tasks (Text Classification, Token Classification, Table Question Answering, Question Answering, Zero-Shot Classification, Translation, Summarization, Conversational, Text Generation, Text2Text Generation). The main area displays a list of 56,783 datasets, with the first few listed below:

- allenai/dolma: Preview • Updated 4 days ago • ↓ 49 • 179
- garage-bAInd/Open-Platypus: Viewer • Updated 8 days ago • ↓ 4.09k • 125
- BAAI/COIG-PC: Preview • Updated 12 days ago • ↓ 373 • 166
- Open-Orca/OpenOrca: Viewer • Updated 3 days ago • ↓ 20.7k • 601
- timdettmers/openassistant-guanaco: Viewer • Updated May 28 • ↓ 37.1k • 184
- OpenAssistant/oasst1: Viewer • Updated May 2 • ↓ 20.9k • 978
- ehartford/dolphin: Preview • Updated 23 days ago • ↓ 2.27k • 152
- fka/awesome-chatgpt-prompts: Viewer • Updated Mar 7 • ↓ 3.96k • 3.09k
- PygmalionAI/PIPPA: Preview • Updated 9 days ago • ↓ 271 • 69
- google/dreambooth: Viewer • Updated 8 days ago • ↓ 8 • 26
- b-mc2/sql-create-context: Viewer • Updated Apr 21 • ↓ 1.38k • 68
- databricks/databricks-dolly-15k: Viewer • Updated Jul 1 • ↓ 42.8k • 325
- togethercomputer/RedPajama-Data-1T: Viewer • Updated Jul 1 • ↓ 17k • 828
- mlabonne/guanaco-llama2-1k: Viewer • Updated 28 days ago • ↓ 11.3k • 34

Ref: <https://huggingface.co/datasets?sort=trending>

# Models in Huggingface

The screenshot shows the Huggingface Model Hub interface. On the left, there's a sidebar with categories like Tasks, Libraries, Datasets, Languages, Licenses, and Other. Below these are sections for Multimodal tasks (Feature Extraction, Text-to-Image, Image-to-Text, Text-to-Video, Visual Question Answering), Computer Vision tasks (Depth Estimation, Image Classification, Object Detection, Image Segmentation, Image-to-Image, Unconditional Image Generation, Video Classification, Zero-Shot Image Classification), and Natural Language Processing tasks (Text Classification, Token Classification, Table Question Answering, Question Answering, Zero-Shot Classification, Translation). The main area displays a grid of model cards. A red box highlights the 'Models 301,886' count at the top center. Other visible model cards include 'stabilityai/control-lora', 'meta-llama/Llama-2-7b', 'Open-Orca/OpenOrca-Platypus2-13B', 'diffusers/controlnet-canny-sdxl-1.0', 'facebook/seamless-m4t-large', 'stabilityai/stablecode-instruct-alpha-3b', 'TheBloke/Llama-2-7B-Chat-GGML', 'stabilityai/stable-diffusion-xl-base-1.0', 'Deci/DeciCoder-1b', 'meta-llama/Llama-2-7b-chat-hf', 'meta-llama/Llama-2-70b-chat-hf', 'garage-bAInd/Platypus2-70B-instruct', 'runwayml/stable-diffusion-v1-5', and 'defog/sqlcoder'. The interface includes a search bar, a 'Full-text search' button, and a 'Sort: Trending' button.

Ref: <https://huggingface.co/models>

# Examples notebook

## PyTorch Examples

### Natural Language Processing

Notebook	Description	Open in Colab	Open Studio Lab
<a href="#">Train your tokenizer</a>	How to train and use your very own tokenizer		
<a href="#">Train your language model</a>	How to easily start using transformers		
<a href="#">How to fine-tune a model on text classification</a>	Show how to preprocess the data and fine-tune a pretrained model on any GLUE task.		
<a href="#">How to fine-tune a model on language modeling</a>	Show how to preprocess the data and fine-tune a pretrained model on a causal or masked LM task.		
<a href="#">How to fine-tune a model on token classification</a>	Show how to preprocess the data and fine-tune a pretrained model on a token classification task (NER, PoS).		
<a href="#">How to fine-tune a model on question answering</a>	Show how to preprocess the data and fine-tune a pretrained model on SQuAD.		
<a href="#">How to fine-tune a model on multiple choice</a>	Show how to preprocess the data and fine-tune a pretrained model on SWAG.		

### Computer Vision

Notebook	Description	Open in Colab	Open Studio Lab
<a href="#">How to fine-tune a model on image classification (Torchvision)</a>	Show how to preprocess the data using Torchvision and fine-tune any pretrained Vision model on Image Classification		
<a href="#">How to fine-tune a model on image classification (Albumentations)</a>	Show how to preprocess the data using Albumentations and fine-tune any pretrained Vision model on Image Classification		
<a href="#">How to fine-tune a model on image classification (Kornia)</a>	Show how to preprocess the data using Kornia and fine-tune any pretrained Vision model on Image Classification		
<a href="#">How to perform zero-shot object detection with OWL-VIT</a>	Show how to perform zero-shot object detection on images with text queries		
<a href="#">How to fine-tune an image captioning model</a>	Show how to fine-tune BLIP for image captioning on a custom dataset		
<a href="#">How to build an image similarity system with Transformers</a>	Show how to build an image similarity system		
<a href="#">How to fine-tune a SegFormer model on semantic segmentation</a>	Show how to preprocess the data and fine-tune a pretrained SegFormer model on Semantic Segmentation		
<a href="#">How to fine-tune a VideoMAE model on video classification</a>	Show how to preprocess the data and fine-tune a pretrained VideoMAE model on Video Classification		

Fine-tune a pretrained model: <https://huggingface.co/docs/transformers/training>

Ref: <https://huggingface.co/docs/transformers/notebooks>

+

## Appendix