Data Science Club - PSUT | Lecture Series

TRANSFORMERS: A VERY GENTLE INTRODUCTION

The only deep learning primer you'll ever need.

Introduction

- Definition of Deep Learning
- Brief Overview of Neural Networks

Artificial Neural Networks

- Perceptrons and Feedforward Neural Networks
- Backpropagation
- Convolutional Neural Networks
- Recurrent Neural Networks

Residual Layers

- Intuition
- Applications

Encoders and Decoders

- Intuition
- Purpose
- Encoder-only, Decoder-only and Encoder-Decoder models

Attention Mechanisms

- Intuition
- Applications

Attention Mechanisms

- Intuition
- Applications

Self-Attention Mechanisms

- Intuition
- How Does it Work
- Self-attention in Sentences
- Self-attention in Images and Videos
- Positional Encoding

Transformers

- New Concepts
- Architecture
- Learning Attention in Sentences
- Learning Attention in Images and Video

Concepts Associated with Transformers

- Pre-training and Fine-tuning (Transfer Learning)
- Reinforcement Learning by Human Feedback
- In-Context Learning
- Knowledge Distillation
- Special Tokens

Key Applications of Transformers

- Natural Language Processing (BERT, GPT)
- Computer Vision (ViT)
- Visual Language Models (Flamingo)
- Tabular Data (TabNet)

Additional Reading ("Fan Service")

- LLaMA and Alpaca
- Informer (Transformers for Sequence to Sequence Prediction)
- AI Hallucinations
- Domain Adaptation (BioMedML, BioGPT, BloombergGPT)

References

Preambles:

Almost every concept is inspired by
the human brain and one of its functions.

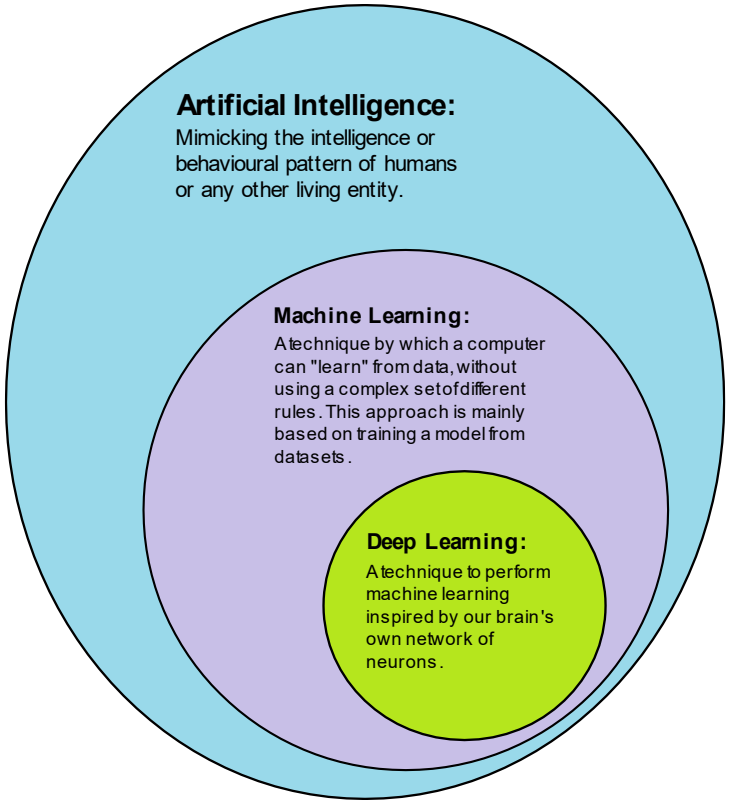
We start with a very simple model, and end up with a model
smart enough to figure out what it wants and how it wants it.

If you can turn anything into a matrix, someone can come up
with a formula for another matrix that represents something you want.

INTRODUCTION

DEFINITION OF DEEP LEARNING

- Deep learning uses **artificial neural networks** to learn from data and make predictions, by building complex hierarchical representations of the data.
- It is inspired by the **human brain**.
- A neural network is typically composed of **multiple layers that are stacked** in a way that allows the network to learn increasingly complex representations of the input data.
- The availability of large datasets and **powerful hardware** has been the major drive behind the recent breakthroughs in deep learning.



Adapted from *Hands On GPU Computing with Python* by Avimanyu Bandyopadhyay

"INTERESTING" NUMBERS

"At its Build developer conference, **Microsoft** today announced that it has teamed up with **OpenAI,** the startup trying to build a general artificial intelligence, with — among other things — a $1 billion investment from Microsoft, to create one of the world's fastest supercomputers on top of Azure's infrastructure. Microsoft says that the 285,000-core machine would have ranked in the top five of the TOP500 supercomputer rankings.

Because Microsoft doesn't actually tell us much more than that, except for a few more specs that say it had 10,000 GPUs and 400 gigabits per second of network connectivity per server, we'll just have to take Microsoft's and OpenAI's word for this."

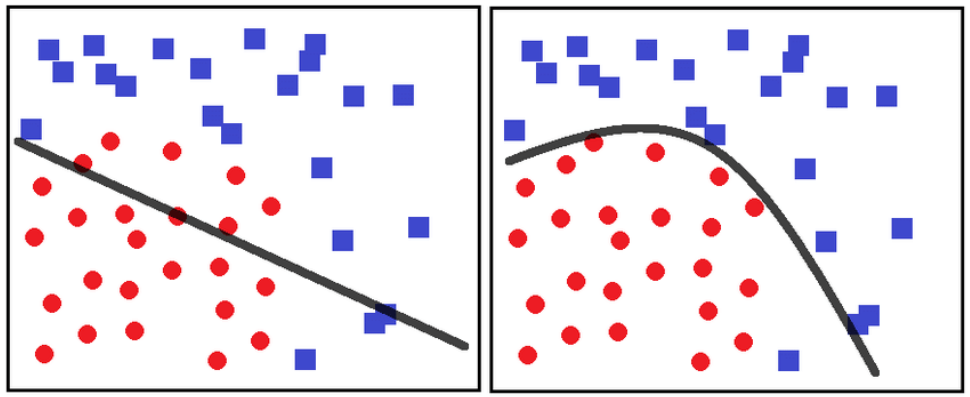
Source: TechCrunch

The cost to train GPT-3 (the model behind ChatGPT) has been estimated between 4.6$ million and 12$ million.

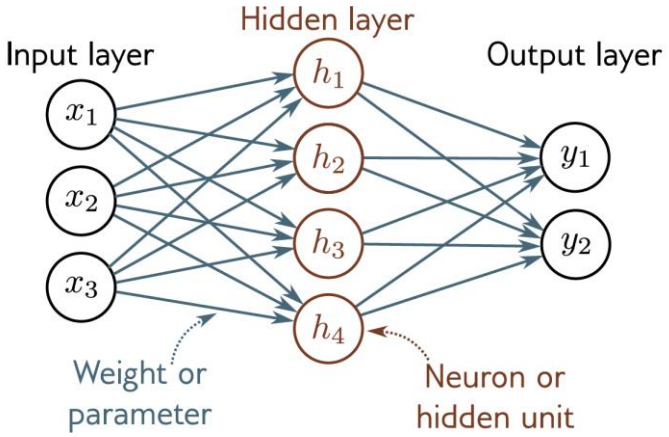
Sources: VentureBeat Lambda Labs

BRIEF OVERVIEW OF NEURAL NETWORKS

- Neural networks consist of interconnected nodes, called **neurons**, that process information.
- Each neuron takes inputs, **performs a mathematical operation** on them, and produces an output.
- The connections between neurons are **weighted**, meaning that some inputs are more important than others.
- These weights are **adjusted** during training to optimize the model's performance.



Adapted from Unboxing the "Black Box": Learning Interpretable Deep Learning Features of Brain Aging by Luis Alberto Souto Maior Neto



Adapted from *Understanding Deep Learning* by Simon J.D. Prince

ARTIFICIAL NEURAL NETWORKS

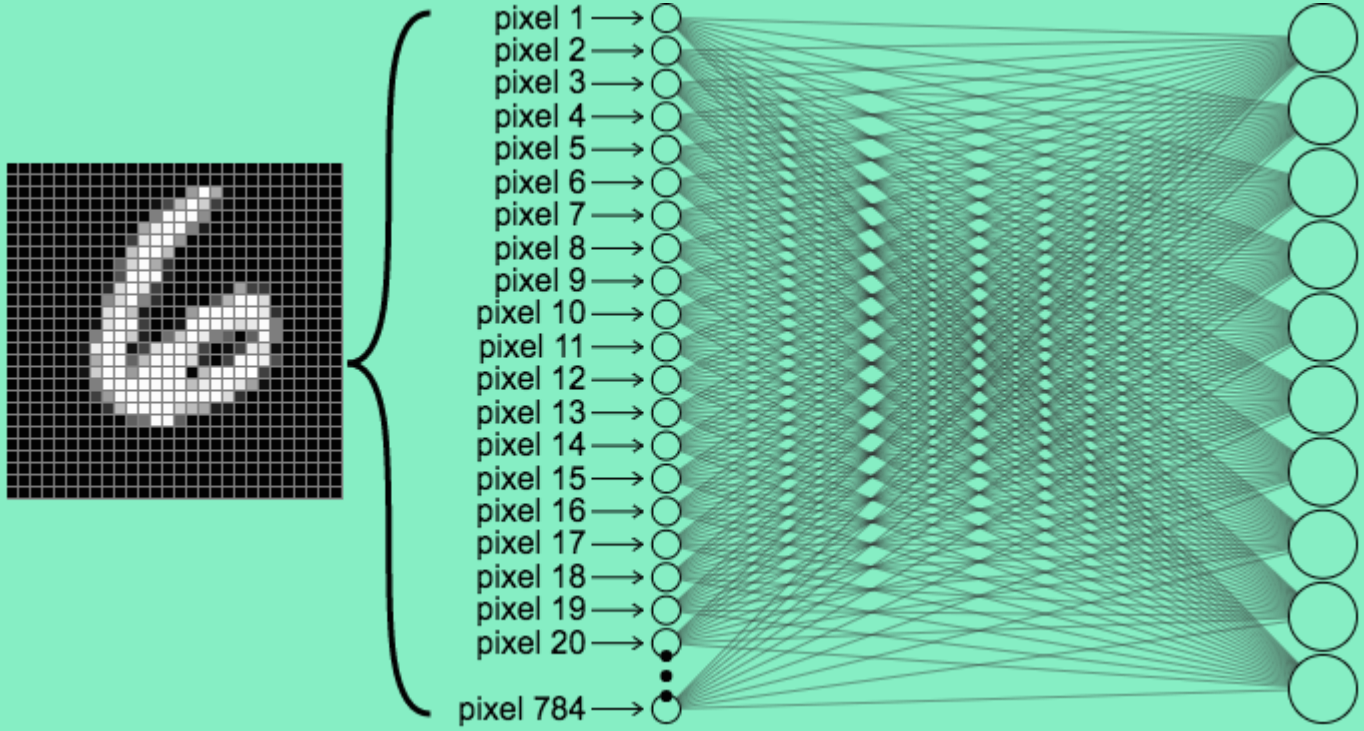
PERCEPTRONS AND FEEDFORWARD NEURAL NETWORKS (FFNS)

- Neural networks have three types of **layers:** input, hidden and output.
- The neurons within the layer are **never connected to each other,** they are connected to the neurons of the adjacent layers.
- The simplest network has **two input cells and one output cell**, which can be used to model logic gates.
- Given that the network has enough hidden layers, it can theoretically always **model the relationship** between the input and output.

Adapted from *The Neural Network Zoo* by The Asimov Institute

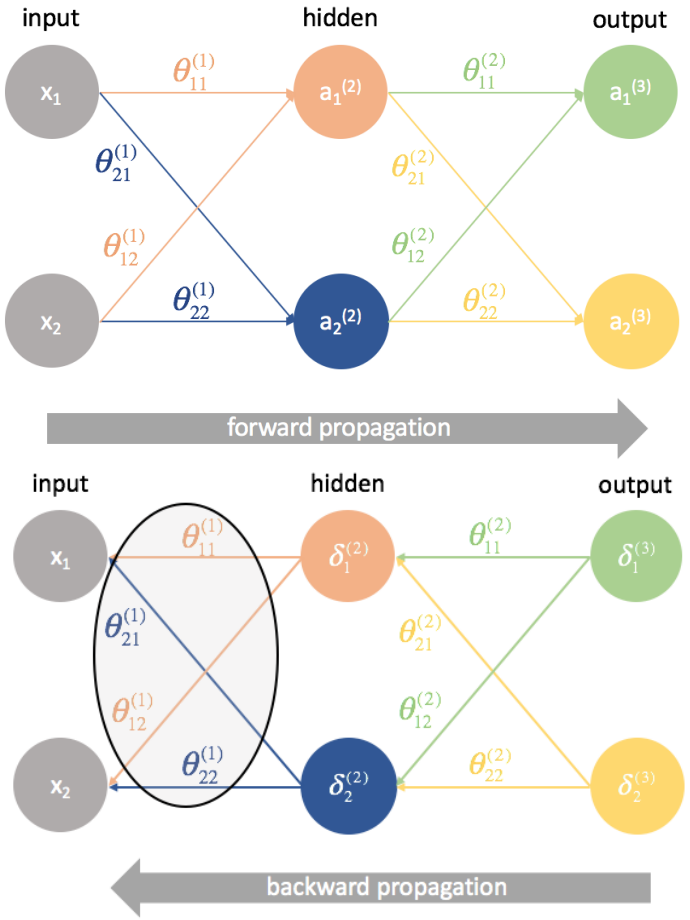
VISUALIZATION OF A FULLY CONNECTED NEURAL NETWORK

Image is Clickable | Credit: Marijn van Vliet



BACKPROPAGATION

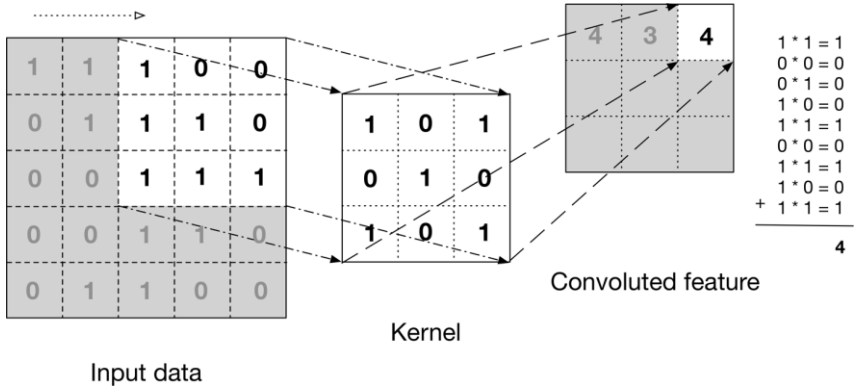
- Feedforward neural networks are trained through back-propagation, giving the network paired datasets of input and output, this is called **supervised learning**, as opposed to **unsupervised learning** where we only give it input and let the network fill in the blanks.
- The error being back-propagated is often some variation of the **difference between the input and the output** (like Mean Squared Error 'MSE' or just the linear difference).
- It works by **going backwards** through the network, comparing the output to the expected output, and adjusting the weights to reduce the difference.
- By **repeating this process** many times, the neural network can improve its ability to make accurate predictions on new, unseen data.



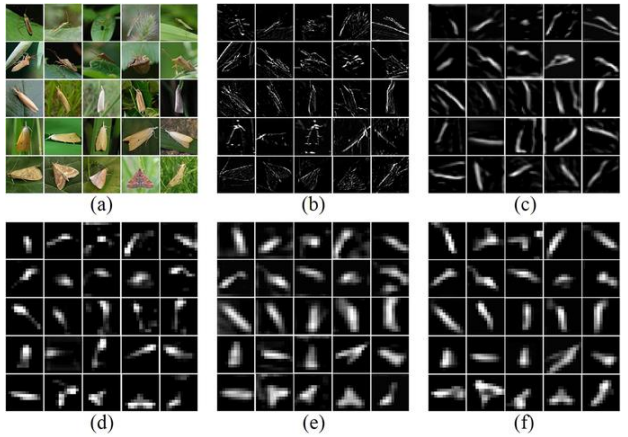
Adapted from *Jeremy Jordan's Neural networks: training with backpropagation*

CONVOLUTIONAL NEURAL NETWORKS (CNNS)

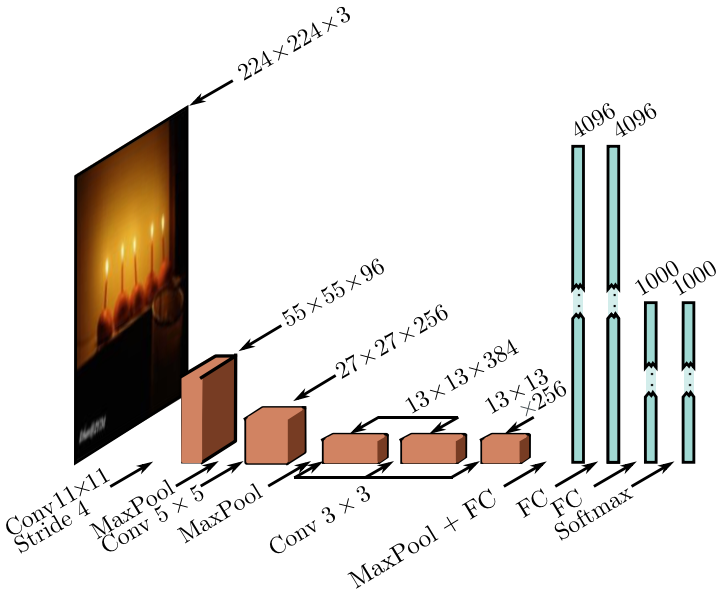
- A type of neural networks that are commonly used for **image recognition,** they are inspired by the way our brain processes visual information.
- CNNs consist of several layers that perform various operations on the input image, such as **convolution and pooling,** these operations extract **features** from the image in the form of **feature maps**.
- The final layer of the CNN is typically a **fully connected layer** that **classifies the image** into one of several categories, CNNs are trained with labeled images.



Adapted from "Deep Learning" by Adam Gibson, Josh Patterson



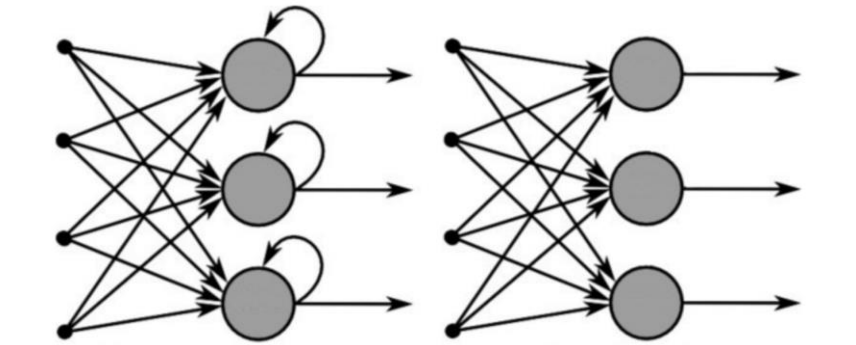
Adapted from Localization and Classification of Paddy Field Pests using a Saliency Map and Deep Convolutional Neural Network



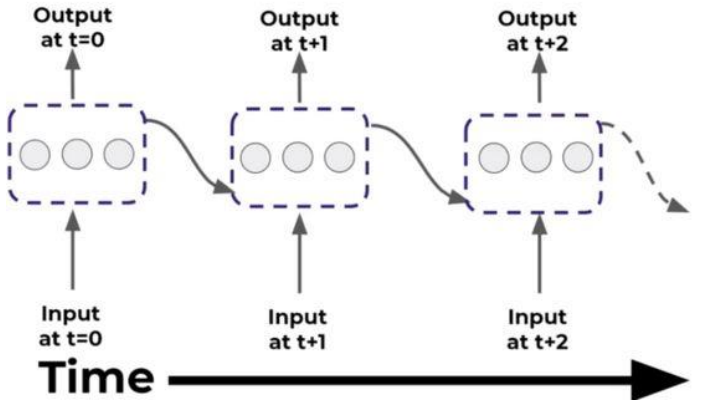
Adapted from Understanding Deep Learning by Simon J.D. Prince

RECURRENT NEURAL NETWORKS (RNNS)

- RNNs are used for processing **sequential data** and have loops that allow information to persist over time, unlike traditional neural networks, they do not process each input independently, this is necessary in allowing the network to capture long-term dependencies in the data such as predicting a word based on the previous word.
- The **vanishing gradient problem** occurs when **too many steps** are involved in the training process, causing the retained information to degrade over time and impairing proper network training.
- **Long Short-Term Memory (LSTM)** and **Gated Recurrent Unit (GRU)** are two types of RNN cells designed to address the vanishing gradient problem which occurs when training deep networks.
- **LSTM** cells use a complex system of gates that allow the network to **selectively "remember" or "forget"** information over time.



Adapted from *nerdcoder*

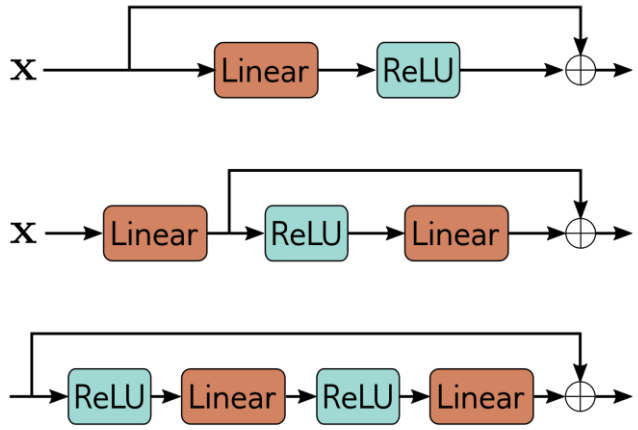


Adapted from *Pinterest*

RESIDUAL LAYERS

INTUITION

- Residual layers, also known as residual blocks or skip connections, are a key component in deep learning architectures that enable the construction of **very deep** neural networks.
- They allow a neural network to learn and **model the residual (or difference)** between the input and output of a layer, instead of learning the direct mapping between them, by doing so, the network can effectively leverage the information from previous layers during training.
- Residual layers enable the network to **shortcut through a few layers** this helps alleviate the vanishing gradient problem and enables the network to learn more complex functions.
- Residual layers have been shown to **significantly improve** the performance of deep neural networks.



Adapted from *Understanding Deep Learning* by Simon J.D. Prince

APPLICATIONS

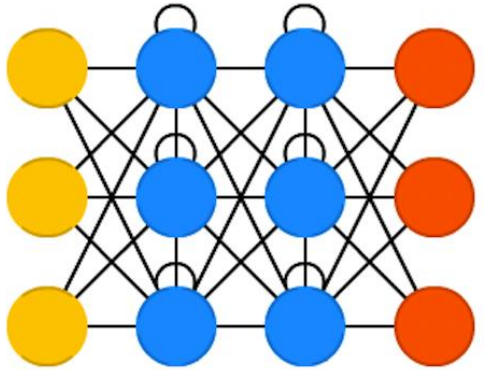
- In image recognition tasks, residual layers have proven to be very effective at improving the accuracy and efficiency of deep convolutional neural networks, the model can effectively learn the complex features of an image and **capture the subtle differences** between different classes of objects.
- In natural language processing, residual layers have been successfully applied to a wide range of tasks, by incorporating residual connections into the model so it can better capture the **long-term dependencies** in sequential data and **prevent the vanishing/exploding gradient problem.**

ENCODERS AND DECODERS

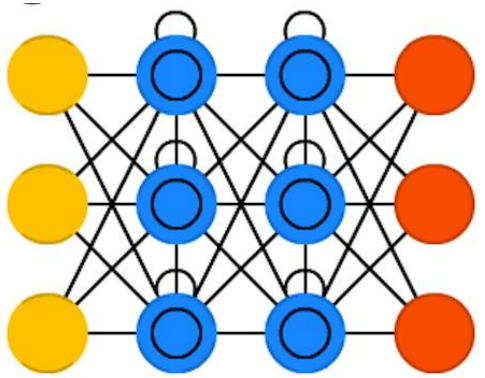
INTUITION

- Although DNNs work well whenever large labeled training sets are available, they cannot be used to map **sequences to sequences.**
- Despite their flexibility and power, DNNs can only be applied to problems whose inputs and outputs can be sensibly encoded with **vectors of fixed dimensionality.**
- It is a significant limitation, since many important problems are best expressed with sequences whose **lengths are not known beforehand.**
- For example, speech recognition and machine translation are **sequential problems.**
- The first proposed encoder-decoder model was introduced **in machine translation using LSTMs.**

RNN



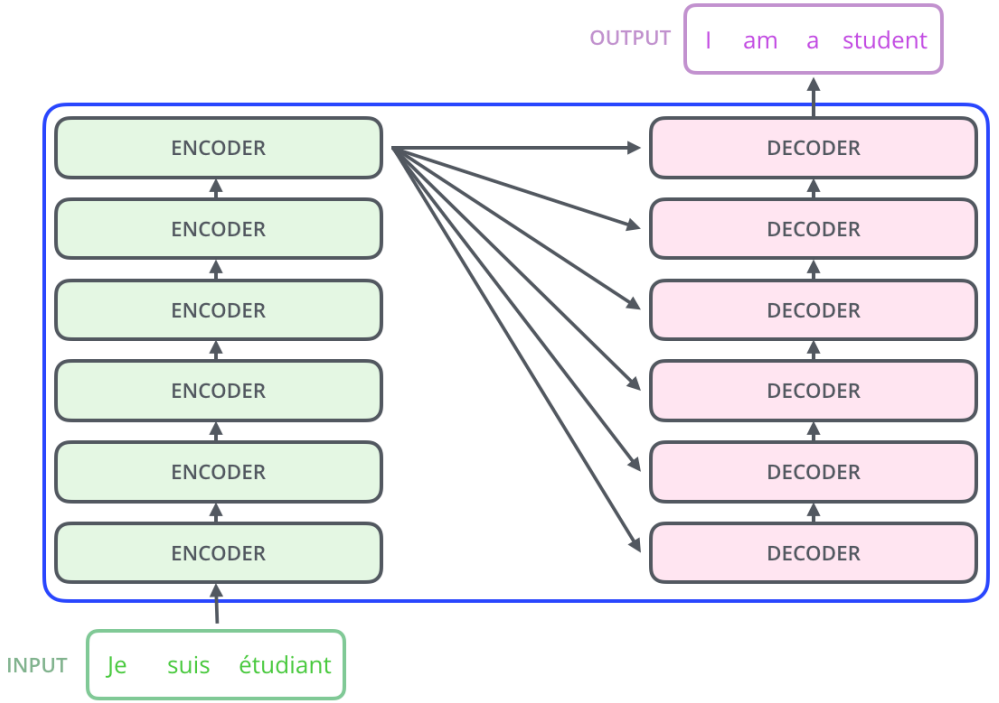
LSTM



Adapted from *The Neural Network Zoo* by The Asimov Institute

PURPOSE

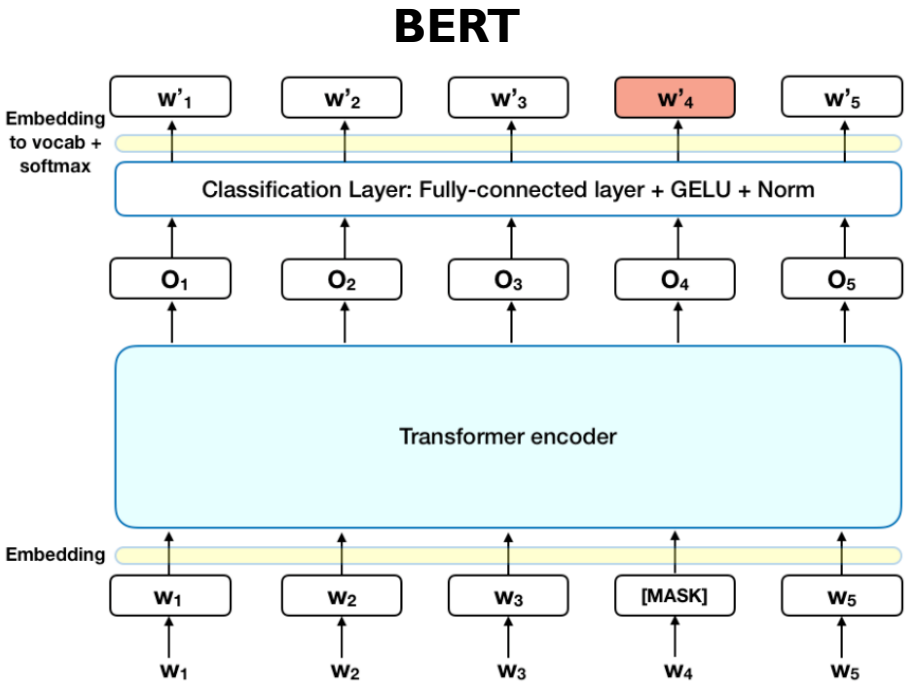
- Word embeddings are a popular way of representing words as **low-dimensional vectors** that capture their **semantic and syntactic relationships,** they have been shown to improve performance compared to traditional one-hot encoding approaches.
- Word embeddings can be fed **into the encoder.**
- The **encoder** is typically used to convert input data into **a compressed representation** that can be used for various downstream tasks, such as classification or generation, this is especially useful when dealing with **high-dimensional input data,** such as images or text.
- **A decoder** takes a fixed-size representation of input data, produced by an encoder, and **generates output data,** such as a sequence of words or an image.



Adapted from *The Illustrated Transformer* by Jay Alammar

ENCODER-ONLY, DECODER-ONLY AND ENCODER-DECODER MODELS

- In **Encoder-Decoder** models, an encoder **takes the input sequence and maps it to an output sequence** that is produced by the decoder. The original Transformer is an example and was proposed to solve a machine translation problem.
- **Encoder-only models** are used to produce a representation that is used for various applications, particularly in unsupervised learning (pre-training).
- **Decoder-only models** take some form of input, such as a sequence of words or an image, and generates a corresponding output sequence.

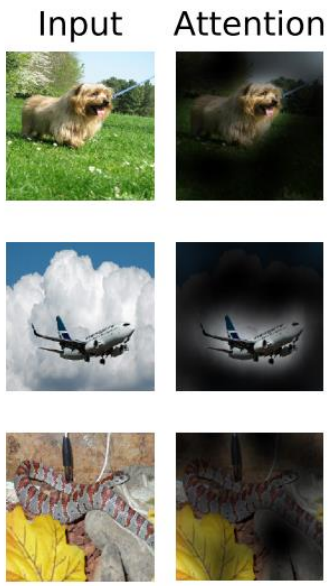


Adapted from *Towards Data Science*

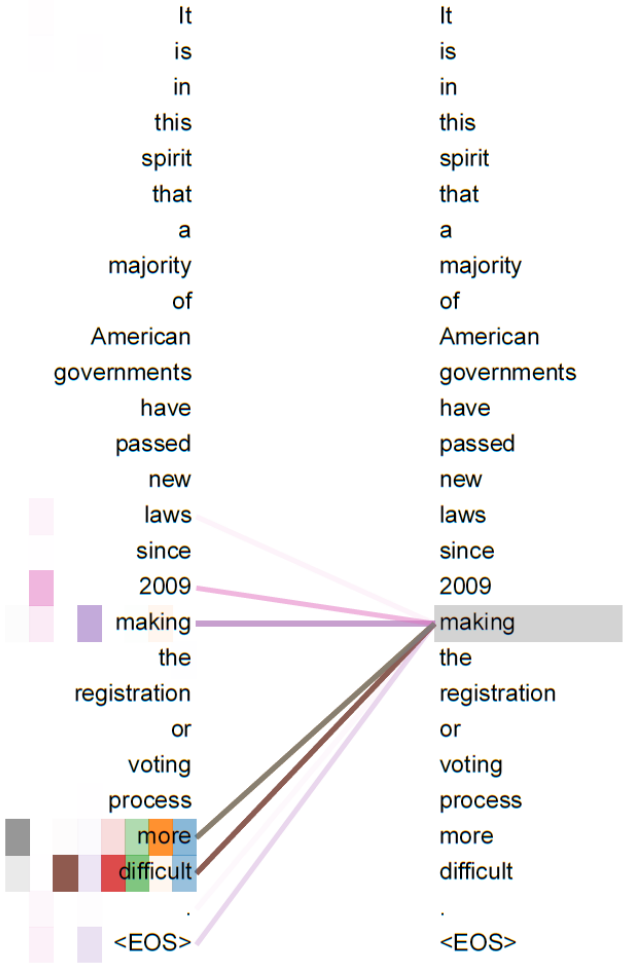
ATTENTION

INTUITION

- The use of a **fixed-length vector in encoder-decoders** has proven to be a **bottleneck,** the performance of encoder-decoders has been shown to deteriorate drastically when the input size (e.g. sentence length) increases.
- Attention was inspired by **cognitive attention**.
- Attention allows the model to look for a **sub vector within the input vector** to pass into the encoder. Imagine it as **"soft weights"** calculated by the model as it runs.
- Using attention the network is **freed from having to squash** the entire input into a fixed-length vector regardless of input size.
- Attention has been shown to **significantly improve** the performance of deep learning models.



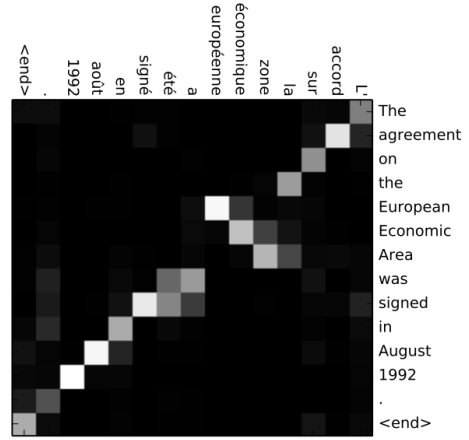
Adapted from "An Image Is Worth 16x16 Words: Transformers For Image Recognition At Scale"

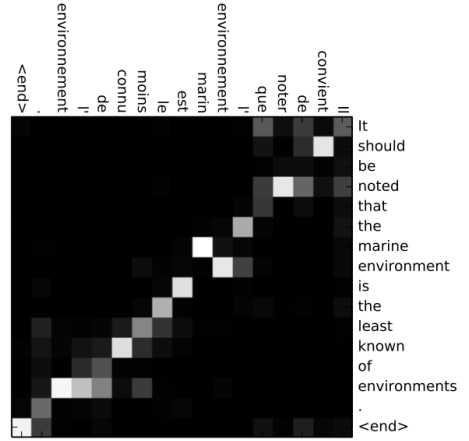


Adapted from "Attention Is All You Need"

APPLICATIONS

- Attention was known as **alignment** at some point, it was used to align different words in the input sentence to the output sentence's words.
- The first proposed attention mechanism used a **Bidirectional RNN (BiRNN)** to generate two vectors, one by traversing the input sequence **forwards,** and one by traversing it **backwards,** concatenating the resulting vector would contain **a "summary"** of the preceding and following words, this vector is known as an **annotation.**
- Due to the tendency of RNNs to better represent recent inputs, **the annotation of word X will be focused on the vicinity of X**, this annotation will be used later on to produce a context vector that will be used for aligning for e.g. different words in a translated sentence.

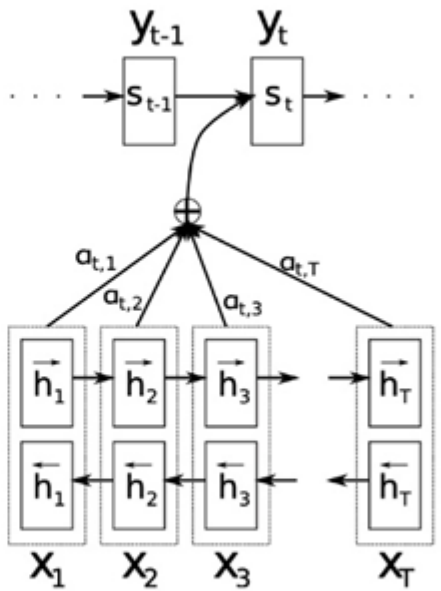




Adapted from "Neural Machine Translation By Jointly Learning To Align And Translate"

Bahdanau (2014) attention used bidirectional LSTMs

To compute attention scores, RNN processes sequence in both directions, then concatenates hidden layers



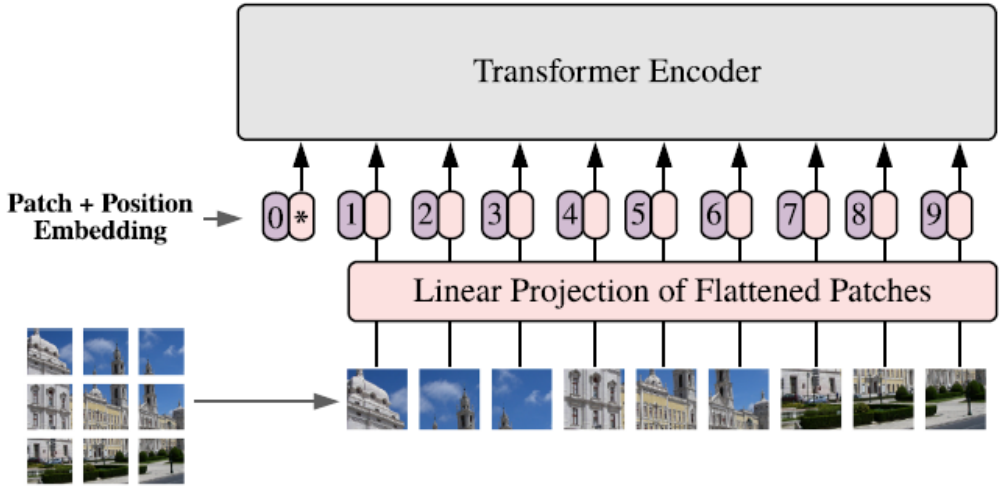
Source: https://arxiv.org/abs/1409.0473

Adapted from https://twitter.com/rasbt/status/1638538494887821313

SELF-ATTENTION

POSITIONAL ENCODING

- Positional encoding is a technique that adds **position-specific information** to the input embeddings since self-attention maps dependencies but does not take position into account.
- This allows the model to distinguish between elements of the input sequence and take their **relative positions** into account.
- There are two main types of positional encoding: absolute positional encoding and relative positional encoding.
- Absolute encoding adds a **fixed offset** to the input embeddings, while relative encoding **uses learned relative position embeddings** to encode the positional information.



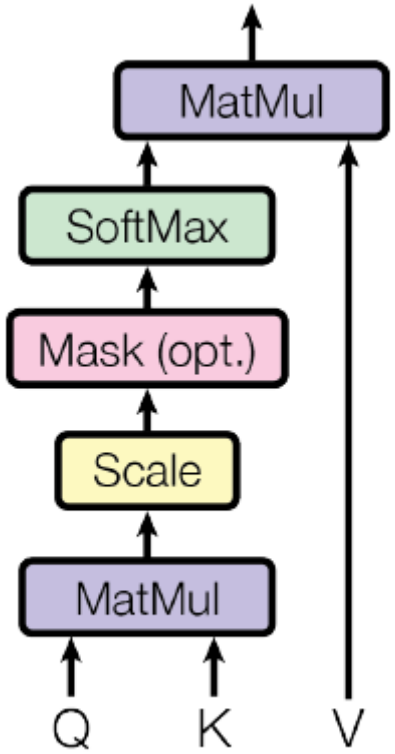
Adapted from *An Image Is Worth 16x16 Words: Transformers For Image Recognition At Scale*

INTUITION

- The combination of alignment (regular attention) and encoder-decoders allowed **the mapping of sequences to sequences** which was a breakthrough.
- A limitation arose in the form of the encoder not being able to **selectively attend to different parts of its input,** since regular attention computes a fixed-weighted sum of the entire encoded input.
- The **sequential nature of alignment** inhibits parallelization, the need for parallelization becomes critical at longer sequence lengths as memory constraints limit batching across examples.
- Self-attention introduces the idea that the model can pick and choose what part of its input to use in its encoder or decoder, this allows for **parallelization** and gives the model great flexibility in size.

HOW DOES IT WORK

- The self-attention mechanism allows the model to understand the relationships between all the words in the input sequence by **assigning weights to each sequence based on its relevance** to the current context, this enables the model to capture complex patterns and dependencies within the sequence, which can lead to more accurate and effective predictions.
- Dot-product is used as a **measure of similarity**, given two vectors, the dot product measures the extent to which they point in the same direction. If the vectors are very similar, the dot product will be large, and if they are dissimilar, the dot product will be small..
- In self-attention, each input position is mapped to a **query vector, a key vector, and a value vector,** which are then used to compute an **attention score** between the input position and all other positions in the sequence, the attention scores are used to compute a weighted sum of the value vectors, which is then used to update the representation of each input position.



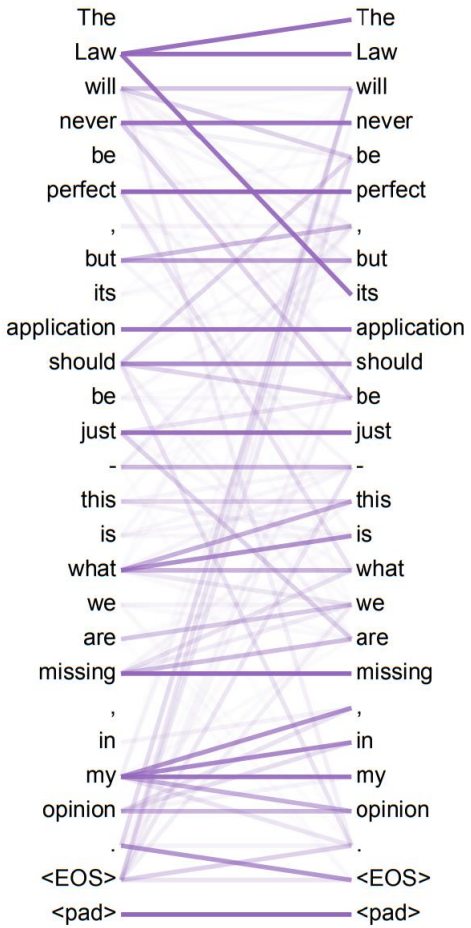
Adapted from *Attention Is All You Need*

SELF-ATTENTION IN SENTENCES

In Natural Language Processing, the input sequence can be a sequence of words or tokens, where each word or token is represented by a vector (word embeddings):

The query, key, and value vectors are computed from these word vectors as follows:

- The **query** would be the current word and the query vectors would be used to compute the similarity between the current word and all the other words.
- The **key** would represent all the other words in the sentence, and its vectors would be used to compute the similarity between each word and the current word.
- The **value** vectors contain information about each word such as semantic meaning or contextual information.



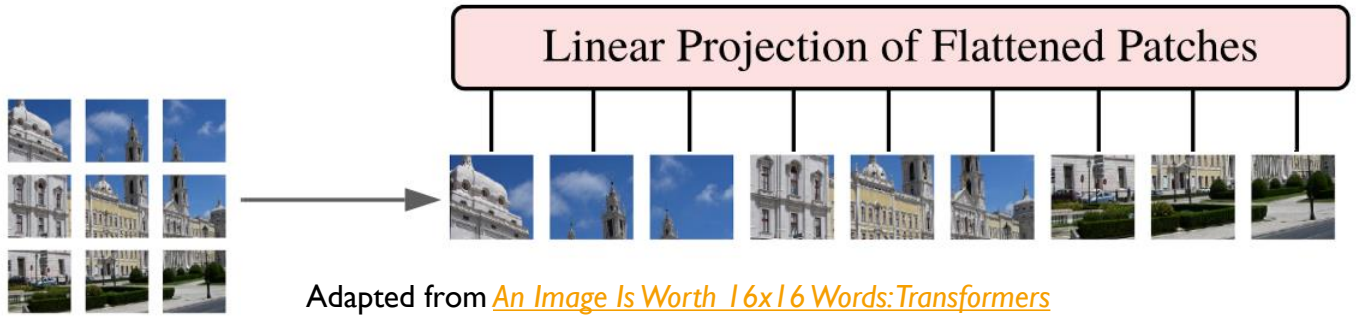
Adapted from *Attention Is All You Need*

SELF-ATTENTION IN IMAGES AND VIDEO

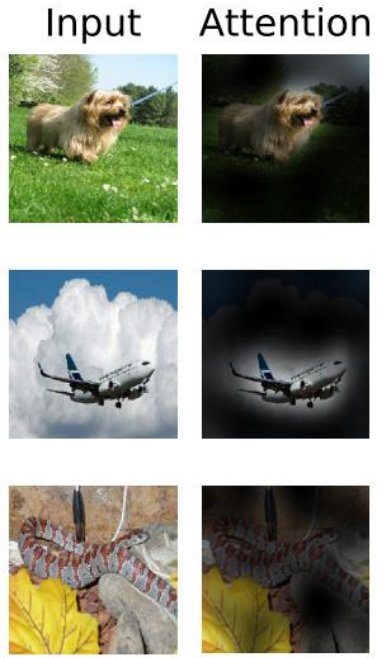
In computer vision, the input sequence can be a sequence of image features or patches.

The query, key, and value vectors are computed from these image features as follows:

- **Query** vectors represent a specific patch or feature and are used to compute the similarity between this patch and all other patches in the sequence.
- **Key** vectors represent all the patches or features in the sequence and are used to compare the similarity between each patch and the current patch or feature.
- **Value** vectors contain information about each patch or feature in the sequence, such as its color or texture information.



Adapted from *An Image Is Worth 16x16 Words: Transformers For Image Recognition At Scale*



Adapted from *An Image Is Worth 16x16 Words: Transformers For Image Recognition At Scale*

TRANSFORMERS

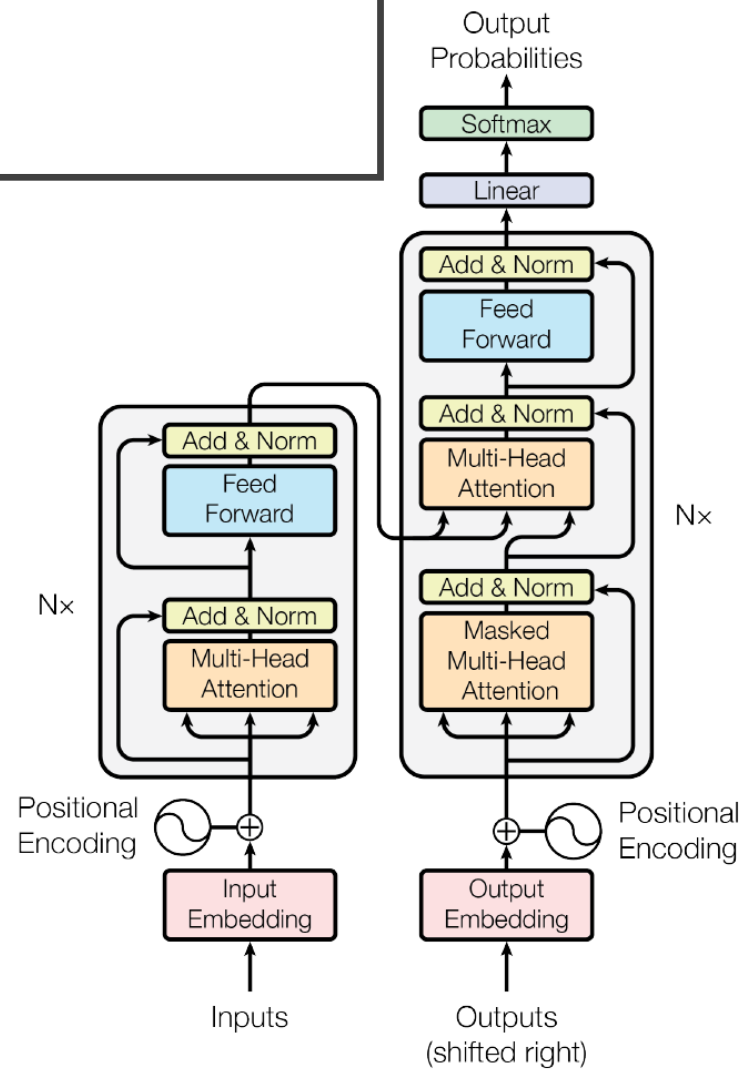
Sam Altman (OpenAI, CEO) in reference to *Attention Is All You Need*

NEW CONCEPTS

- Transformers are arguably the biggest breakthrough in deep learning in the past decade, the original proposal was to **rely entirely on self-attention,** ridding models of both recurrency and convolution, in the aptly named paper Attention is All You Need.
- **Parallelization,** the architecture of a Transformer ultimately solves the constraint of sequential computation, allowing for significantly less training time.
- **Scaled Dot Product Attention,** which calculates the relevance of each element in a sequence to every other element, and scales the results to prevent numerical instability.
- **Multi-head Attention,** which allows the model to focus on multiple aspects of input data in parallel, with a single attention head, the averaging prevents this.
- **Positional Encoding,** which allows the model to inject information about the position of an input in a sequence.

TRANSFORMER ARCHITECTURE

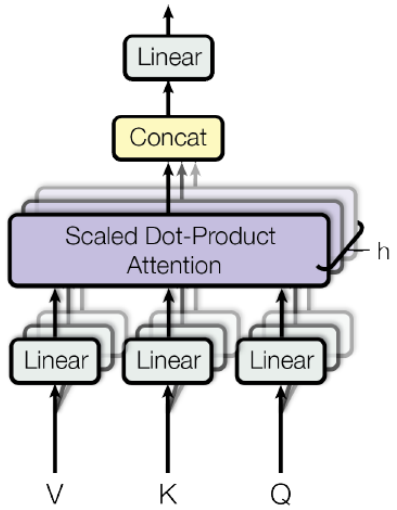
- The architecture consists of an encoder and a decoder, both of which are composed of several layers of multi-head self-attention and feed-forward neural networks.
- In the **encoder,** the input sequence is first **embedded** into a vector space using an embedding layer. The resulting embeddings are then **passed through a series of self-attention layers,** where each token attends to every other token in the sequence to compute an attention-weighted representation of the sequence. The output of each self-attention layer is then **passed through a feed-forward neural network.**
- In the decoder, the output sequence is **generated autoregressively,** one token at a time. Similar to the encoder, the input sequence is first **embedded** into a vector space using an embedding layer. The resulting embeddings are then **passed through a series of self-attention layers,** where each token attends to the previous tokens in the output sequence to generate the next token. The decoder also attends to the output of the encoder through a series of **cross-attention layers,** which allows it to incorporate information from the input sequence into the output sequence, it is then **passed through a feed-forward neural network**.
- The purpose of shifting the target sequence to the right is to ensure that the decoder can **only attend to the previous positions** in the target sequence, and **not to future positions.**



Adapted from *Attention Is All You Need*

LEARNING ATTENTION IN SENTENCES

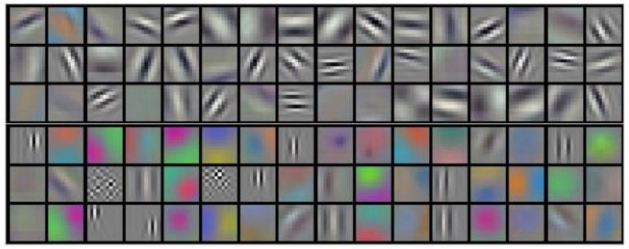
- The input sequence is first transformed into a set of queries, keys, and values using three separate linear transformations.
- Then, for each query token, the self-attention mechanism computes a weight for each key token in the sequence, based on the **similarity between the query and the key**. These weights are used to compute a weighted sum of the values, which represents the attention-weighted representation of the input sequence for that query token.
- In NLP, the query vector represents the **current token** in the input sequence, and the key vectors represent all the **other tokens** in the sequence. The value vectors are the same as the key vectors, and represent the embedded input text. The dot product of the query vector with each key vector produces an attention weight, which is then used to compute a weighted sum of the value vectors, resulting in the context vector.
- The parameters of the self-attention mechanism are **learned during training,** using backpropagation through the neural network, to optimize a given objective function such as minimizing the difference between the predicted and actual outputs.



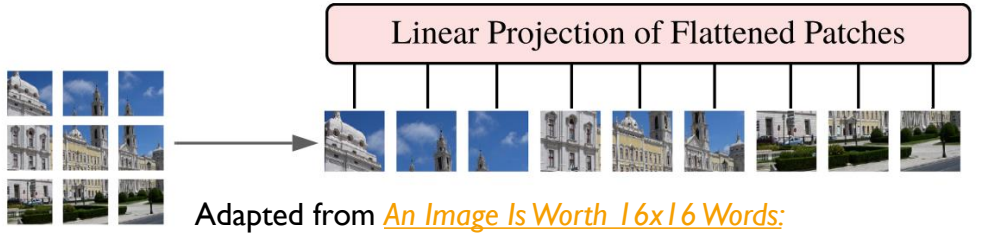
Adapted from *Attention Is All You Need*

LEARNING ATTENTION IN IMAGES AND VIDEO

- In a Vision Transformer (ViT), the key, query, and value are typically obtained by applying three separate linear transformations to the **feature maps** output by the convolutional neural network (CNN) backbone.
- The feature map is first **flattened into a sequence of 2D patches**, each of which is represented as a vector. The key, query, and value matrices are then obtained by applying separate linear transformations to the patch embeddings.
- The query matrix represents the **current patch** being attended to, and the key matrix represents all **other patches** in the sequence. The value matrix is the same as the key matrix and represents the embedded features of the input image. The dot product of the query matrix with each key matrix produces an attention weight, which is used to compute a weighted sum of the value matrix, resulting in the attended representation of the image.
- The parameters of the self-attention mechanism are **learned during training**, using backpropagation through the network, to optimize a given objective function such as minimizing the difference between the predicted and actual outputs.

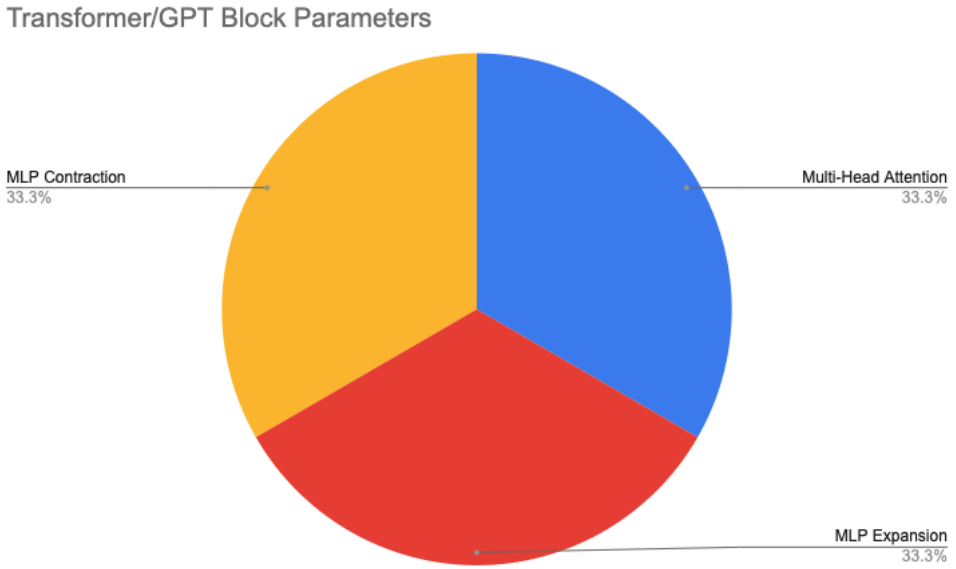


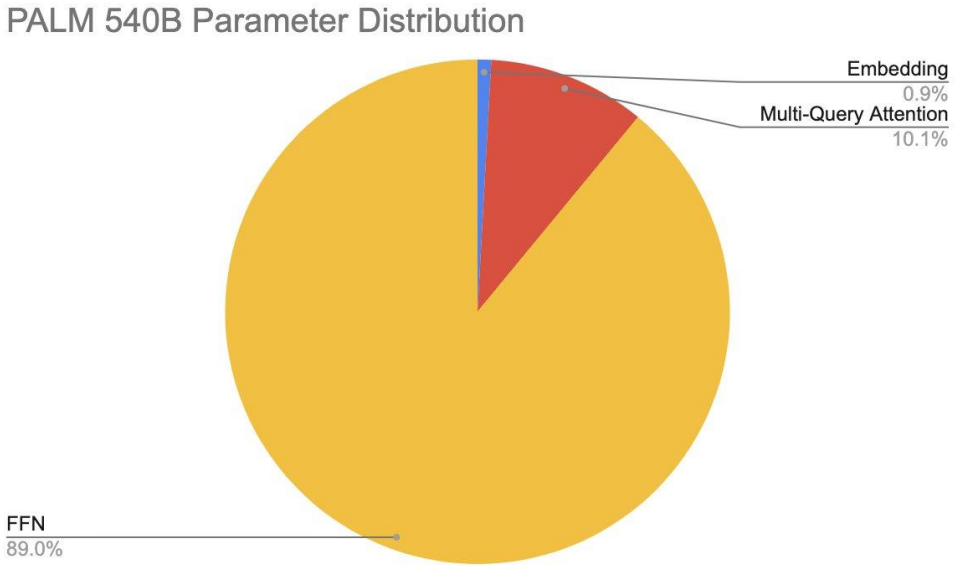
Adapted from *Convolutional neural networks (CNN) tutorial* by Jonathan Hui



Adapted from *An Image Is Worth 16x16 Words: Transformers For Image Recognition At Scale*

TRANSFORMERS PARAMETERS





Adapted from https://twitter.com/O42nl/status/1632469700402618369

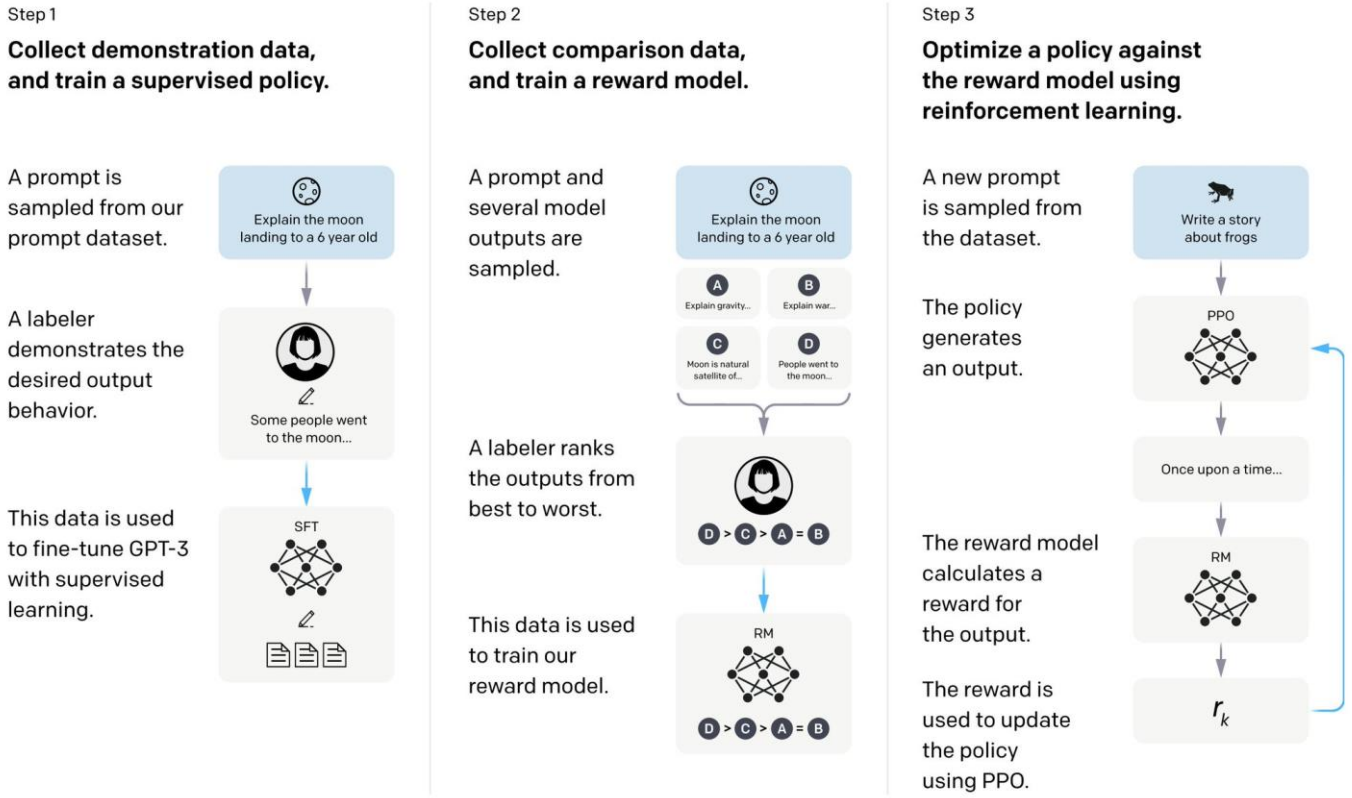
CONCEPTS ASSOCIATED WITH TRANSFORMERS

PRETRAINING AND FINETUNING (TRANSFER LEARNING)

- **Pre-training**, a type of self-supervised learning that involves training a model on large amounts of unlabeled data to learn a **general representation of the input,** which can then be fine-tuned for specific downstream tasks.
- **Masked language modeling** is a popular approach used in pre-training transformers, where a portion of the input sequence is **masked** and the model is trained to **predict the missing tokens** based on the context provided by the surrounding tokens.
- Other pre-training approaches include next **sentence prediction,** where the model is trained to **predict whether two sentences follow each other** in a given text, and **contrastive learning,** which trains the model to **differentiate between similar and dissimilar inputs.**
- **Fine-tuning** for downstream tasks involves using a **pre-trained model as a starting point** and training it further **on a smaller labeled dataset for a specific task,** such as sentiment analysis or machine translation. Fine-tuning allows the model to adapt its pre-learned representation to the specific characteristics of the task, making it more effective at the task at hand.
- **Downstream tasks** are **specific tasks** that the pre-trained model is fine-tuned for, where the goal is to **use the pre-trained knowledge to achieve good performance** on the downstream task with minimal additional training.

REINFORCEMENT LEARNING BY HUMAN FEEDBACK

- Reinforcement learning from human feedback is a method in which an AI agent learns to perform a task by **receiving feedback from humans** on its actions. The agent adjusts its behavior based on the feedback it receives.
- The agent takes actions and receives feedback based on the quality of its actions and **a subsequent reward.**
- The goal of the agent is to learn to make decisions that **maximize its cumulative reward** over time.



Adapted from Ryan Lowe, OpenAI

IN-CONTEXT LEARNING

- Used in fine-tuning and encompasses **Zero-shot** Learning, **One-shot** Learning and **Few-shot** Learning.

Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

```
1  Translate English to French:        ← task description
2  cheese =>                           ← prompt
```

One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.

```
1  Translate English to French:        ← task description
2  sea otter => loutre de mer          ← example
3  cheese =>                           ← prompt
```

Few-shot

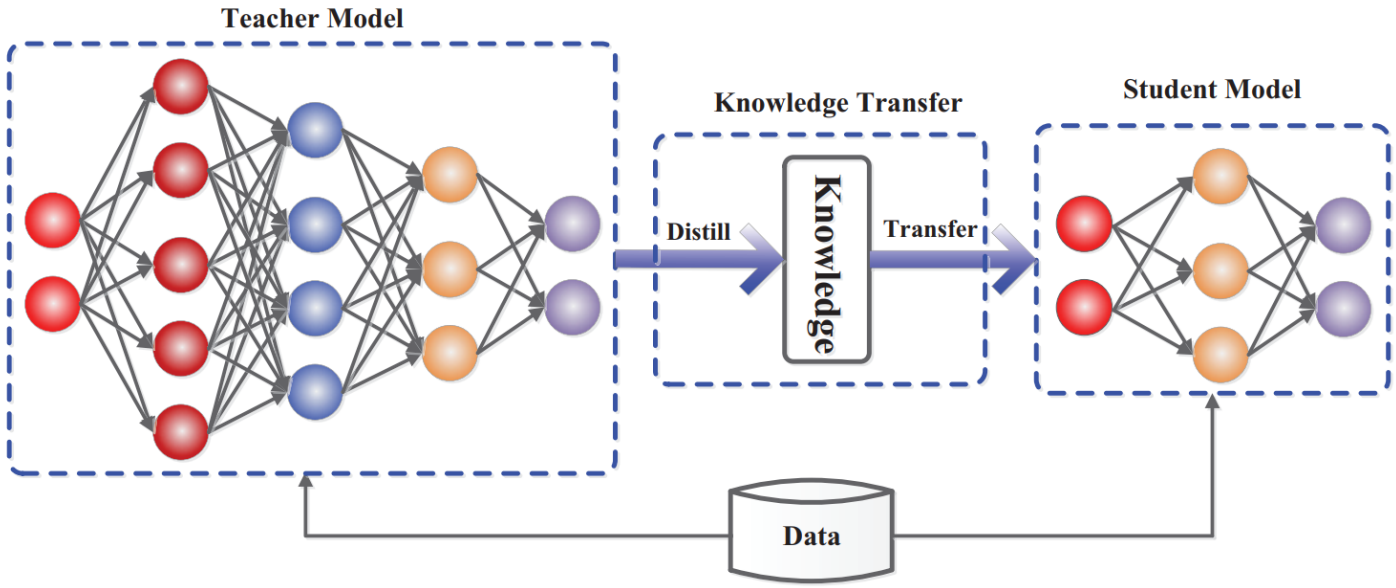
In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

```
1  Translate English to French:        ← task description
2  sea otter => loutre de mer          ← examples
3  peppermint => menthe poivrée        ←
4  plush girafe => girafe peluche      ←
5  cheese =>                           ← prompt
```

Adapted from *Language Models are Few-Shot Learners*

KNOWLEDGE DISTILLATION

- A technique to compress a large and computationally expensive model into a smaller and faster model, while maintaining the accuracy of the original model.
- The process involves training a smaller student model to **mimic the behavior of the larger teacher model,** by using the outputs of the teacher model as targets for the student model.
- The student model learns to **approximate the predictions of the teacher model,** and can then be used for inference on smaller devices.



Adapted from *Knowledge Distillation: A Survey*

SPECIAL TOKENS

Natural Language Processing Tokens:

- [CLS]: Denotes the beginning of a sequence and is often used for classification tasks.
- [SEP]: Separates two sequences in a sequence pair task.
- [MASK]: Represents a masked token, used during pre-training for masked language modeling.
- [PAD]: Used for padding sequences to ensure they all have the same length.
- [UNK]: Represents an unknown or out-of-vocabulary token.

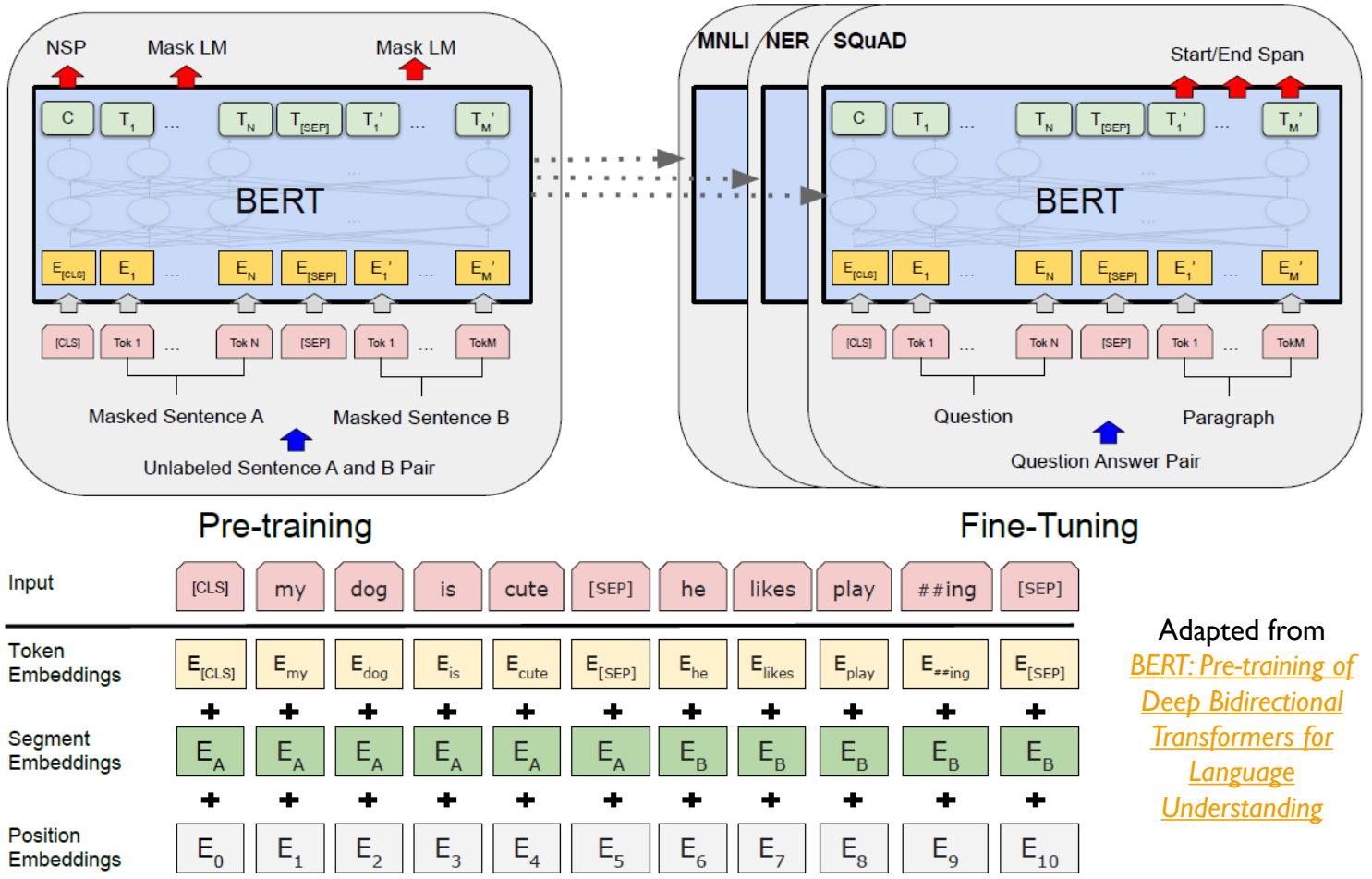
In Computer Vision:

- [CLS]: Denotes the beginning of an image sequence and is used in classification tasks.
- [SEP]: Separates two image sequences in a sequence pair task.
- [MASK]: Represents a masked region in the input image, used during pre-training for masked image modeling.
- [PAD]: Used for padding image sequences to ensure they all have the same length.

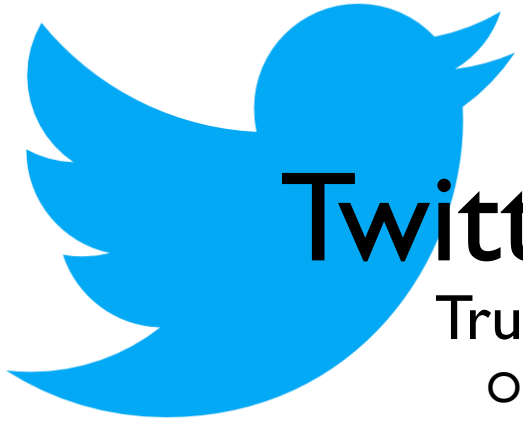
APPLICATIONS OF TRANSFORMERS

BIDIRECTIONAL ENCODER REPRESENTATIONS FROM TRANSFORMERS (BERT)

- **Encoder-only** model.
- Bidirectional means the model can learn from **the left or the right** of the input.
- Encoder representation means the model is able to **generate a representation** of the input that can be used for specific downstream tasks.
- **BERT** is pre-trained then fine-tuned for specific tasks.



Adapted from [BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding]



Twitter Algorithm

Trust and Safety Models

Open Sourced in March, 2023

A **BERT** model is trained and used in-house
for **all** of these tasks (classification).

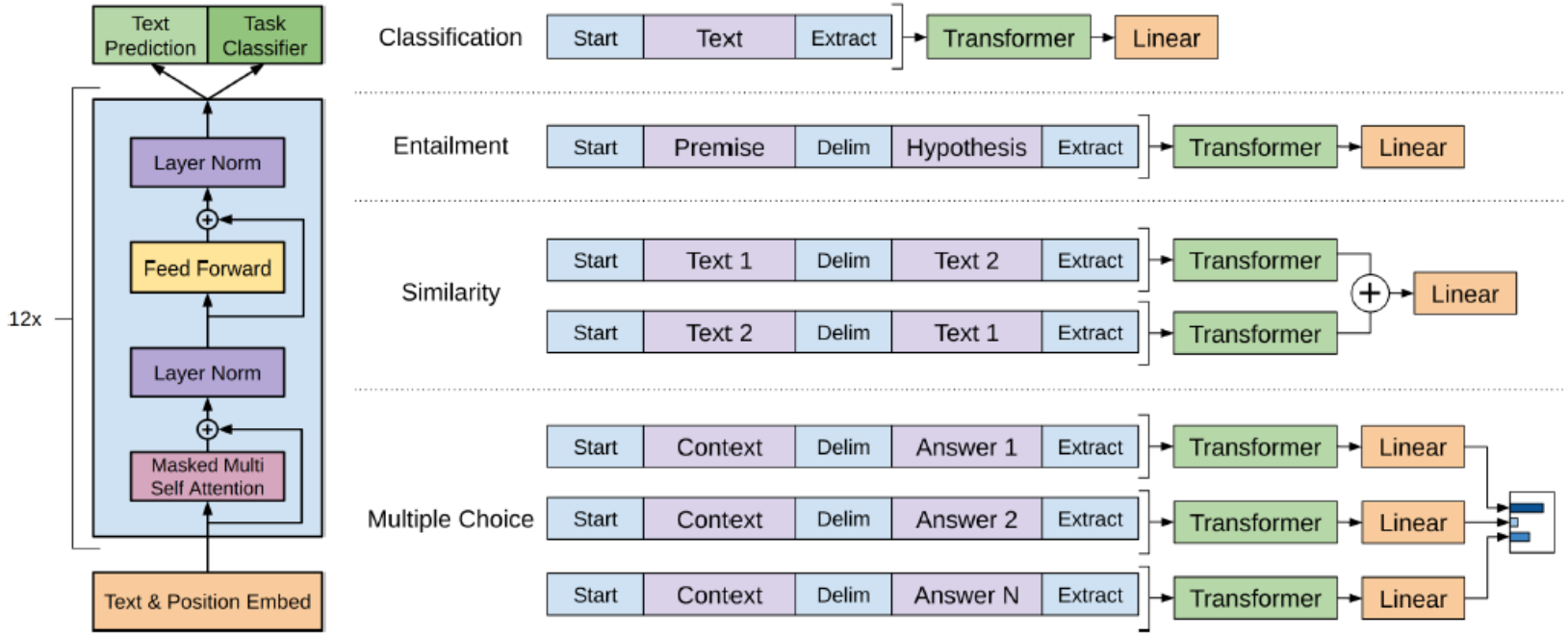
Toxicity

Abusive Language

NSFW Language

GENERATIVE PRETRAINED TRANSFORMER (GPT)

- Autoregressive **decoder-only** language model which means it uses the word before to generate the next word.
- Pre-trained with **masked language modelling** then fine-tuned.
- Fine-tuned with **instructions.**
- Fine-tuned using Reinforcement Learning from Human Feedback.



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

Encoder, BERT-style

- “Bidirectional”
- Masked language model
- learns better representations
- better for classification etc.

Input sentence: *The curious kitten deftly climbed the bookshelf*

Pick 15% of the words randomly

The curious kitten deftly climbed the bookshelf

- **80% of the time, replace with [MASK] token**
- **10% of the time, replace with random token (e.g. ate)**
- **10% of the time, keep unchanged**

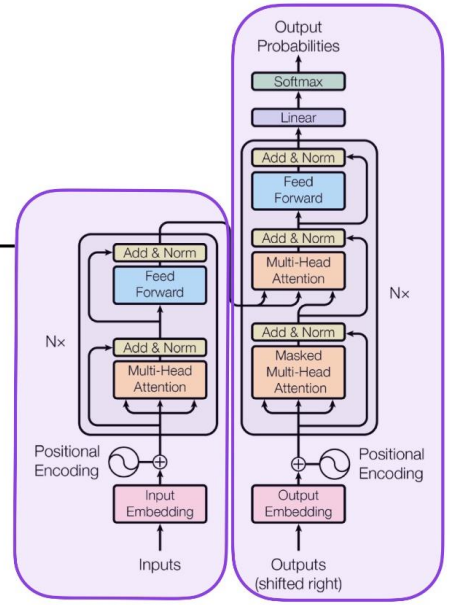


Figure 1: The Transformer - model architecture.

Source: https://arxiv.org/abs/1706.03762

Decoder, GPT-style

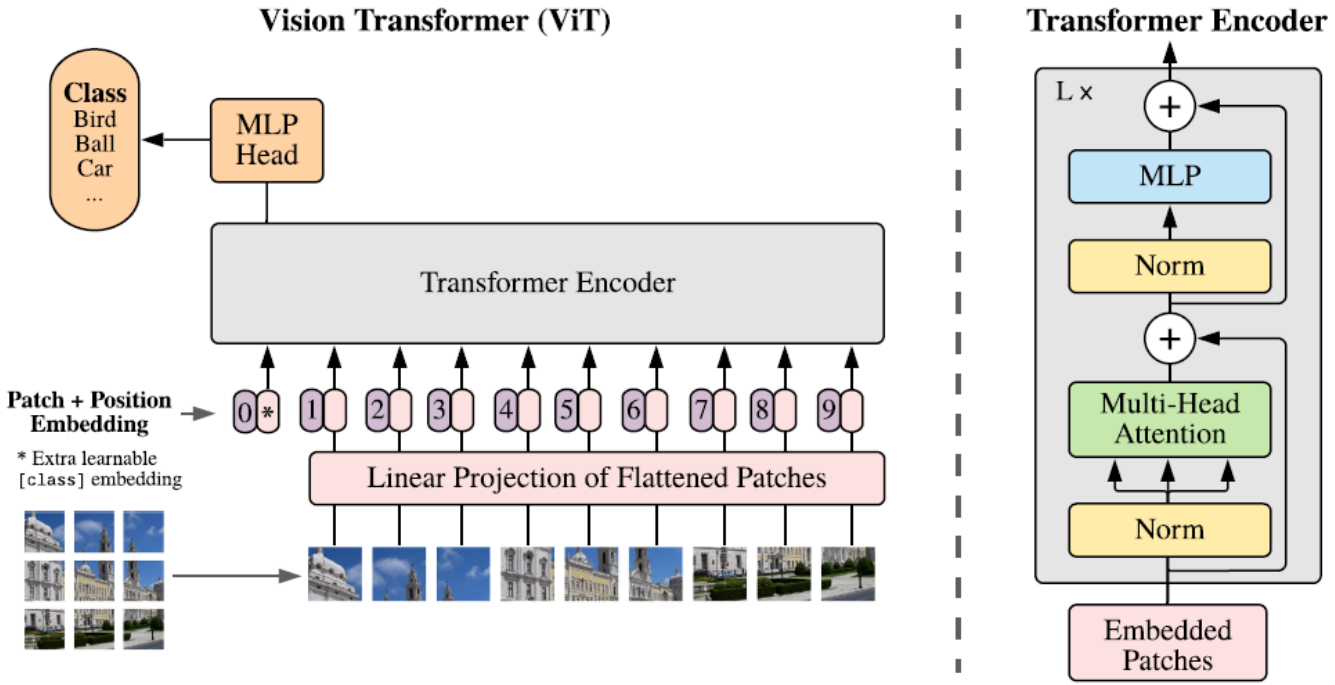
- Unidirectional
- Casual multi-head self-attention
- next-word prediction
- better for text generation

Feed model text from left to right, and it learns to predict the next word.

Adapted from https://twitter.com/rasbt/status/1638538494887821313

VISION TRANSFORMER (VIT)

- Tried to replicate the **exact architecture** of the first transformer.
- Broke images down **to smaller patches** to emulate how transformers deal with words.
- Uses self-attention instead of **convolution**.
- Embeds a **class token** for image classification
- Only performed well when trained on huge datasets, due to the absence of the **inductive biases** that convolution had (features are located spatially) and pooling (translation invariance), training on large amounts of data helped the model learn these inductive biases.

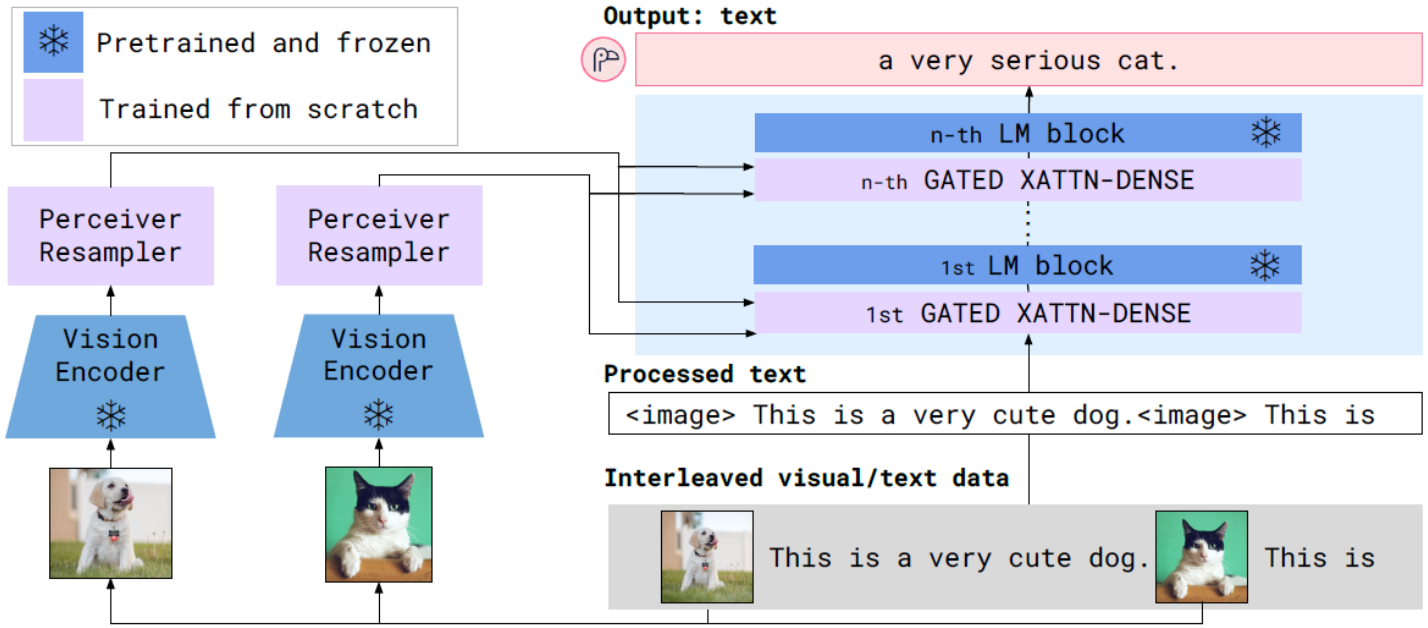


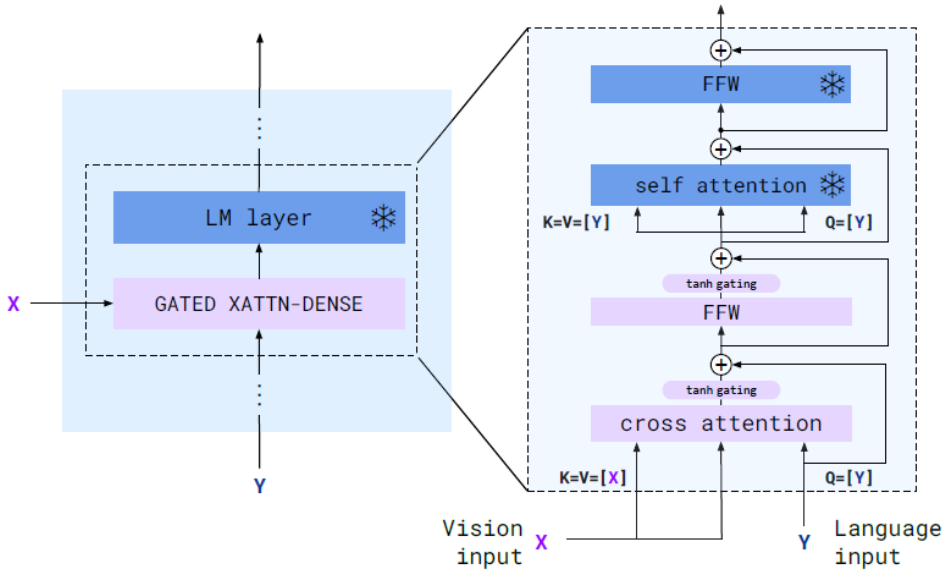
Adapted from *"An Image Is Worth 16x16 Words: Transformers For Image Recognition At Scale"*

VISUAL LANGUAGE MODELS

- Visual language models work by first processing the visual input (such as an image) using a **Vision Transformer**.
- This transformer breaks the image down into smaller features, which are then fed into an encoder.
- The encoder processes the features and produces a compact representation of the image.
- Next, the textual input (such as a caption or question) is processed using a **language model,** which produces a sequence of encoded tokens.
- These encoded tokens are then fed into a decoder, which generates a textual output.
- To combine the visual and textual information, **attention mechanisms** are used.
- These mechanisms allow the model to attend to specific parts of the image and specific words in the textual input, depending on the task at hand.

FLAMINGO

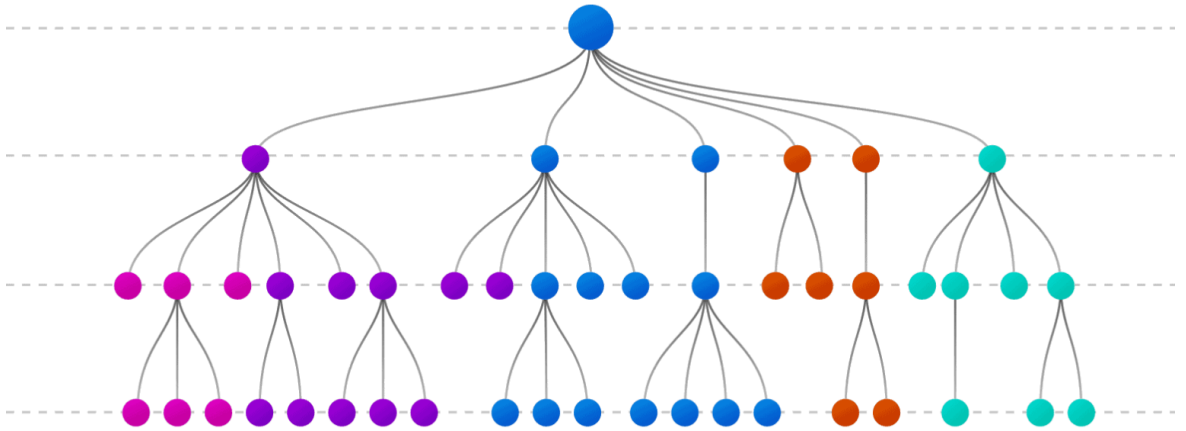




Adapted from *Flamingo: a Visual Language Model for Few-Shot Learning*

TABNET PREDECESSOR: DECISION TREES AND GRADIENT BOOSTED MODELS

- Gradient Boosted Models are **ensemble models** that use decision trees as base models, they used to (and still) **outperform** other models on tabular data.
- Require extensive **feature engineering** and meticulous **hyperparameter tuning**.



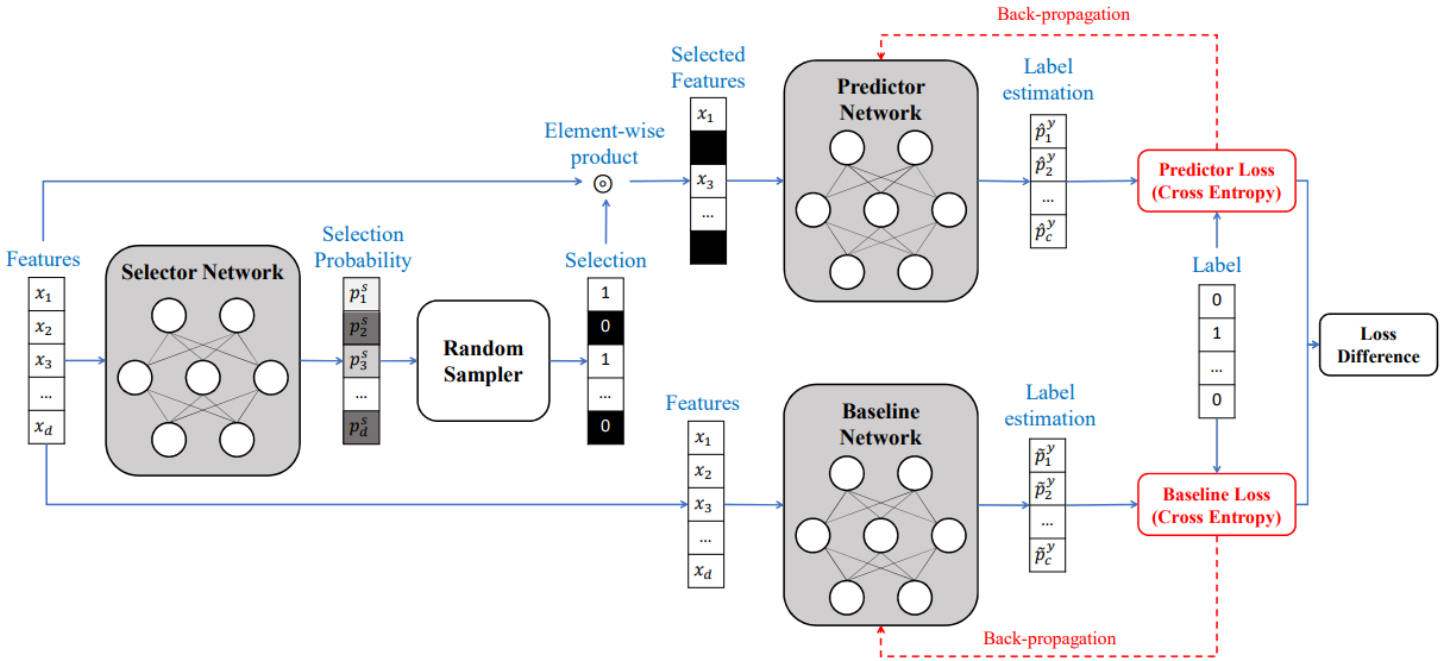
dmlc XGBoost

LightGBM

Yandex CatBoost

TABNET PREDECESSOR: INSTANCE-WISE VARIABLE SELECTION (INVASE)

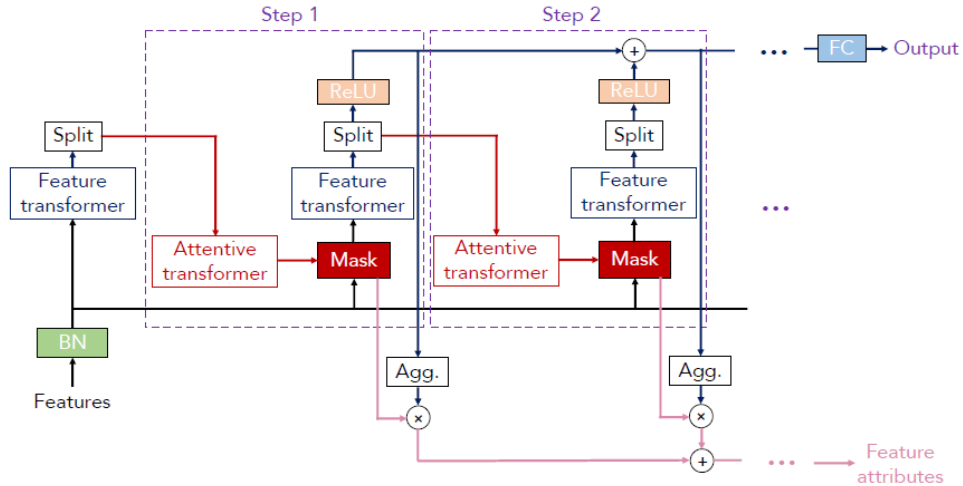
- A model that chooses what features it needs to predict **every instance**.
- Consists of 3 neural networks, a **selector network**, a **predictor network** and a **baseline network** which are used to train the selector network using the **actor-critic methodology**.



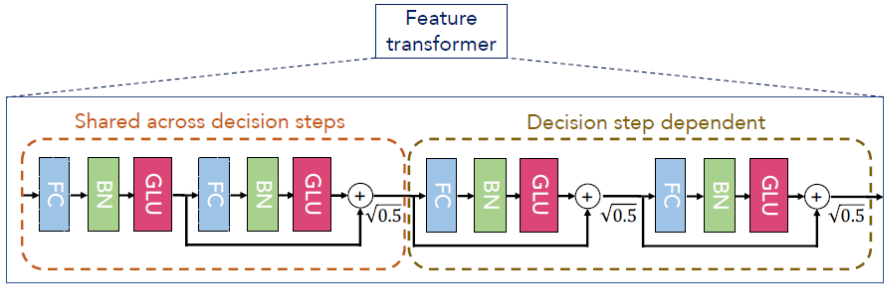
Adapted from *INVASE: Instance-wise Variable Selection Using Neural Networks*

TABNET

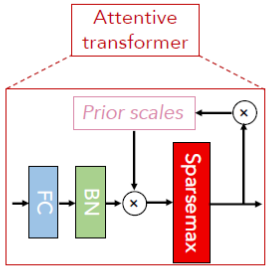
- Mimics ensembling through **decision steps.**
- Uses sequential attention for **instance-wise feature selection** by **masking features** in every step, the masks can be aggregated to obtain global feature importance.
- The masks are **learned** in training where categorical features can also be **embedded.**



(a) TabNet encoder architecture

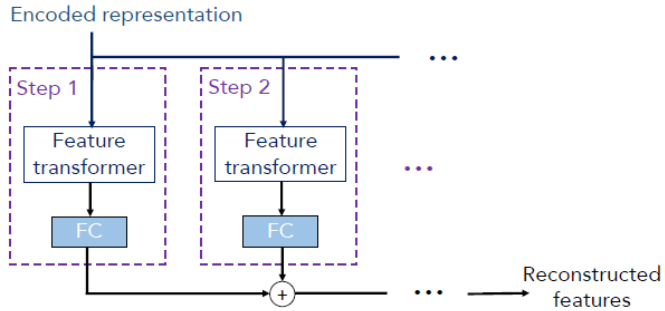


(c)



(d)

Adapted from *TabNet: Attentive Interpretable Tabular Learning*



(b) TabNet decoder architecture

Adapted from *TabNet: Attentive Interpretable Tabular Learning*

LLAMA:
LARGE LANGUAGE MODEL META AI

Did you know that Yann LeCun dropped a rap album last year? We listened to it and here's what we thought:

Dr. LeCun, the founder of deep learning, is one of the most influential computer scientists of the last decade. LeCun's latest album, titled Deep Learning, is a mix of rock, punk and rap. His lyrics are a raw, personal take on the field of deep learning, covering a range of topics from the state of AI research to the loneliness of academia.

Yann LeCun, the rap game's newest emcee.

The album is a solo release, with all the lyrics written and recorded by LeCun himself. In fact, he also composed and produced every song. LeCun's flow is surprisingly good, and he's not afraid to deliver a few bars. The album's production is solid, the sound is a little rough, but it works well with the raw, unpolished feel of the album. Lyrically, LeCun is a wordsmith, with a knack for clever rhymes. Let's take a closer look at some of the songs.

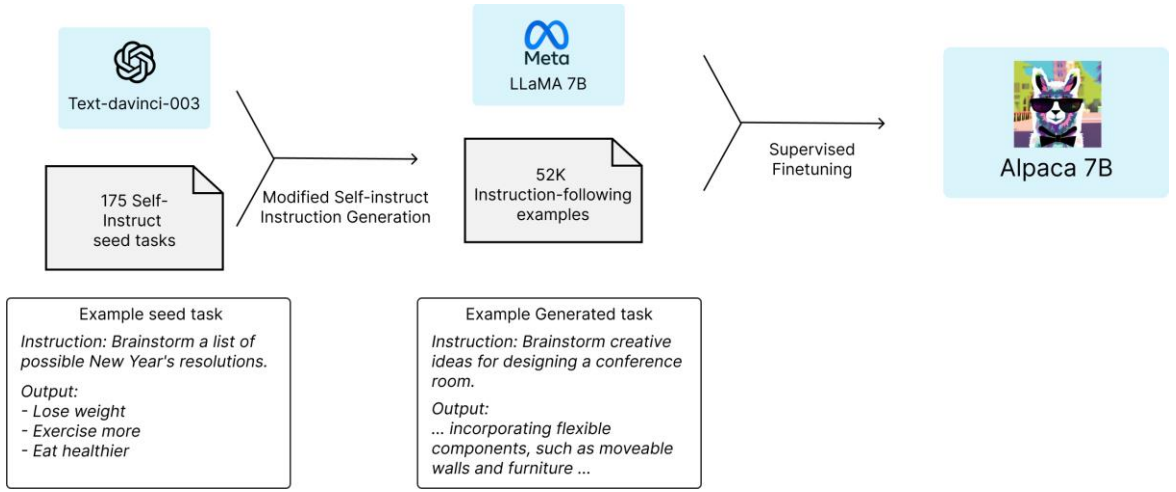
Deep Learning is the title track of the album. In the song, LeCun talks about his vision for the future of AI. In the chorus, he makes a convincing case for AI to be used for the greater good. He sings:

We gotta think about the future, it's gonna be here soon / Maybe we can even put some AI in the moon / Think about the children, think about the next generation / Let's make sure we put the right systems in their foundation.

Adapted from *LLaMA: Open and Efficient Foundation Language Models*

LLAMA

- A collection of **foundational language models** by Meta, pretrained on **public data**, to make the point that it is possible to train state-of-the-art models using publicly available datasets exclusively.
- Ranging from **7B to 65B** parameters.
- LLaMA-13B **outperforms** GPT-3.
- LLaMA gave birth to Alpaca; an **instructions fine-tuned** model.
- If LLaMA was GPT, Alpaca would be ChatGPT.
- Both models and training data are **open source.**



Adapted from *Stanford*

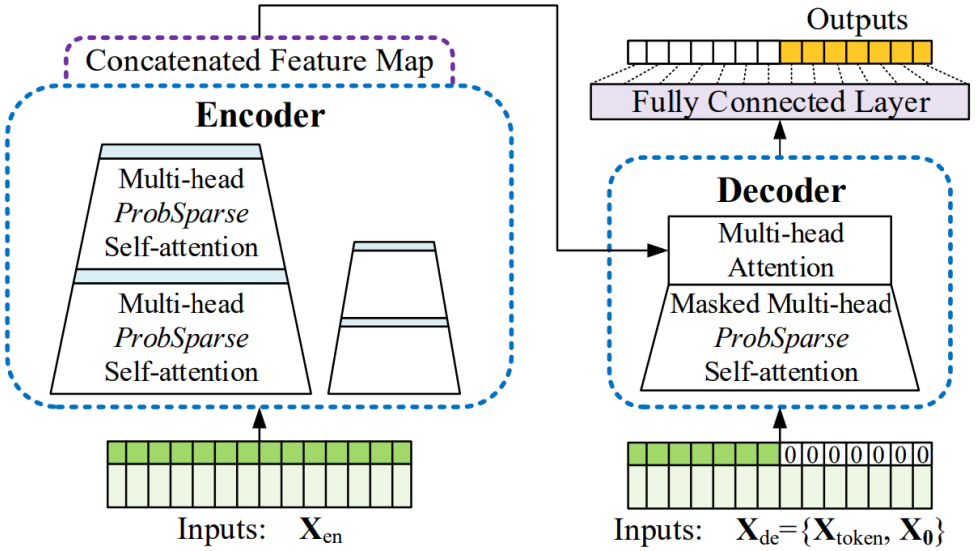
For reference:

GPT-3: 175B / **GPT-4** (Rumors): 1.5 Trillion
PaLM: 540B
LaMDA: 137B

LONG SEQUENCE TIMESERIES FORECASTING

LONG SEQUENCE TIMESERIES FORECASTING

- Timeseries is treated as a **sequence of inputs**.
- The encoder generates a representation of the input that the decoder uses to **generate future time steps**.
- The attention mechanism allows the model to selectively focus on certain parts of the input sequence, enabling it to capture **long-range dependencies.**
- For **long sequence forecasting**, the input sequence can be split into smaller subsequences and fed to the model one at a time, the encoder's output from the previous subsequence can be used as input to the next subsequence, allowing the model to incorporate information from past time steps.



Adapted from *Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting*

DOMAIN ADAPTATION

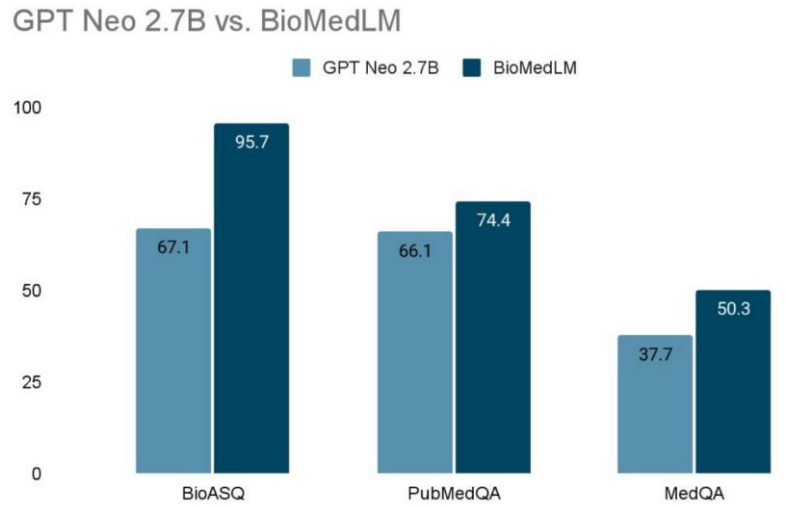
APPROACH

1. Preprocess data with a **domain specific approach**, e.g. use a domain specific tokenizer, or even a subdomain specific tokenizer!
2. Choose the **type of task** you'll be using your model for:
   - BERT (Encoder models): Twitter Safety & Trust Models, tasks are centered around language understanding and classification.
   - GPT (Decoder models): BioGPT, BioMedLM, BloombergGPT, the model must be able to generate natural language i.e. answer questions.
3. Pretrain a foundational model on **domain specific data**
4. Test **beyond** accuracy:
   - Robustness: does one type break your entire model?
   - Bias: does your model think old people don't deserve to live?
   - Fairness: does your model consider women subhuman?
   - Toxicity: will your model tell you to kill yourself?
   - Efficiency: does your model cost 3$ million to train and 5$ million each time you use it?
   - Safety: if you trained it on sensitive data, can the model leak training data during inference?

MODEL SHOWCASE

BioMedLM	BioGPT	BloombergGPT
GPT-style models: have the ability to generate text		
Open source and on HuggingFace		Closed source (BOOOOOO)
Pretrained and finetuned on domain specific data Data preprocessed with domain specific methods e.g. tokenizers for each subdomain		
Trained exclusively on large scale biomedical data such as PubMed abstracts		Trained exclusively on financial data and augmented with general purpose data

Word	Tokenization
chromatography	chrom/atography
cytotoxicity	cyt/ot/oxicity
Immunohistochemistry	Immun/oh/ist/ochemistry
photosynthesis	photos/ynthesis
probiotic	prob/iotic



TRIUMPHS & DOWNFALLS

AB

I vaguely heard a conversation on a bus the other day, it sounded like they were talking about something statistics related, the topic was interesting but I couldn't really tell what they were saying it was noisy, sounded something like linear depression

It's possible that the topic they were discussing was linear regression, which is a statistical method used to model the relationship between two variables.

In linear regression, a linear equation is used to predict the value of one variable (the dependent variable) based on the value of another variable (the independent variable). The goal is to find the line of best fit that minimizes the distance between the predicted values and the actual values.

Depression, on the other hand, is a mood disorder that affects a person's thoughts, feelings, and behavior. It is not directly related to statistics or linear regression. However, there may be research studies that use statistical methods such as linear regression to analyze data related to depression.

AI HALLUCINATION

- "Despite GPT-4's capabilities, it maintains a tendency to **make up facts**, to **double-down on incorrect information**, and to perform tasks incorrectly. Further, it often exhibits these tendencies in **ways that are more convincing and believable than earlier GPT models** (e.g., due to authoritative tone or to being presented in the context of highly detailed information that is accurate), increasing the risk of overreliance."
- "Despite its capabilities, GPT-4 has similar limitations to earlier GPT models, it is **not fully reliable** (e.g. can suffer from "hallucinations"), has a **limited context window**, and **does not learn from experience**. Care should be taken when using the outputs of GPT-4, particularly in contexts where reliability is important."

GPT-4 Technical Report

Slides and Notebook

https://github.com/a-tabaza/dsc-lectures

More Key Applications (Advanced):

- Multivariate Timeseries Forecasting: Informer
- Automatic Speech Recognition: Conformer
- Recommender Systems: BERT4Rec
- Reinforcement Learning: Decision Transformer

Textbooks:

- Understanding Deep Learning by Simon J.D. Prince (Beginner)
- Multimodal Deep Learning (Advanced)

Domain Specific Models:

BioMedLM | BioGPT | BloombergGPT

Coding Resources:

- PyTorch Fundamentals from Microsoft (Beginner)
- Understanding and Coding the Self-Attention Mechanism of Large Language Models From Scratch by Sebastian Raschka (Intermediate)
- Annotated PyTorch Paper Implementations (Advanced)

Illustrated Guides to Transformers (and other Models) by Jay Alammar | This resources was cited in MIT Lectures)

Papers:

- Deep Residual Learning for Image Recognition
- Attention Is All You Need
- Neural Machine Translation By Jointly Learning To Align And Translate
- *An Image Is Worth 16x16 Words: Transformers For Image Recognition At Scale*
- *Language Models are Few-Shot Learners*
- *Distilling the Knowledge in a Neural Network*
- *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*
- *Improving Language Understanding by Generative Pre-Training*
- *Flamingo: a Visual Language Model for Few-Shot Learning*
- *INVASE: Instance-wise Variable Selection Using Neural Networks*
- *TabNet: Attentive Interpretable Tabular Learning*
- *LLaMA: Open and Efficient Foundation Language Models*