



# TRANSFORMERS AND NLP

Trevor Yu

November 20, 2023



# Learning Objectives

- Describe the high-level architectural blocks of transformer models
- Describe the process of transfer learning and creating foundation models
- Identify NLP tasks which are appropriate to use with Encoder, Decoder, and Encoder-Decoder architectures
- List limitations of applying transformers to NLP problems
- List options of how to use transformer models for practical problems

# Questions

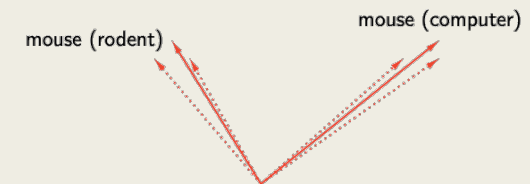
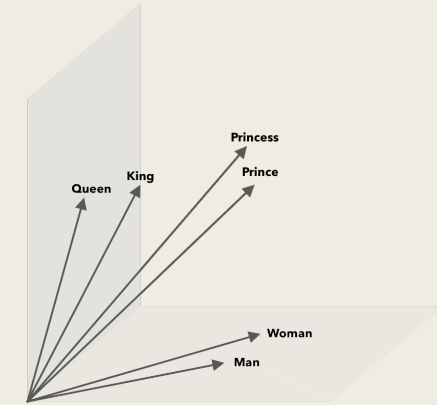
- If you're ever unsure of what a term means or have any questions, please stop me and we can discuss further

# Background: NLP

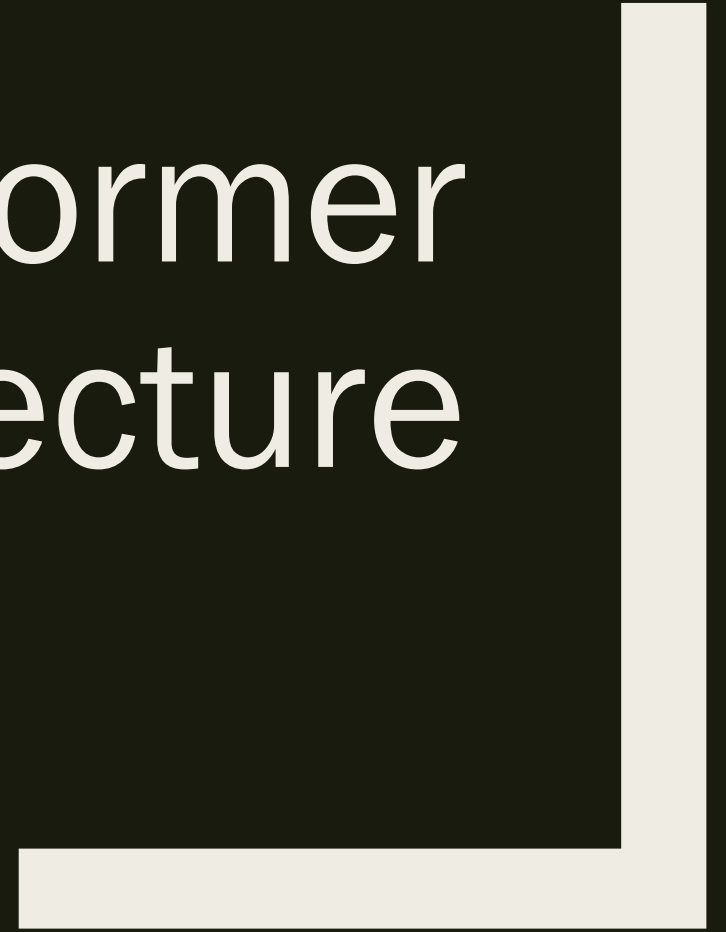
- Natural Language Processing (NLP): Giving computers the ability to process human language
  - Includes data such as speech, pictures of written language, and text encoded as strings
  - Tasks can include speech recognition, optical character recognition, text classification, and language generation
- Before machine learning (ML), NLP techniques focused on applying symbolic, rule-based approaches, such as grammar rules and stemming
- Early statistical and ML approaches (before 2015) required extensive feature engineering inspired by linguistics (e.g. part-of-speech tagging)

# Background: Neural Network Approaches

- Word2Vec (2013) is a shallow neural network trained to produce semantic vector embeddings for individual words
  - Demonstrated that some language semantics could be learned and encoded in vectors, such as binary gender roles, country capitals, and tenses
  - “Ottawa” - “Canada” + “France” = “Paris”
  - “Bag of words” approach assumes no contextual relationships between words
- ELMo (2018) is a LSTM recurrent network trained to produce contextual vector embeddings based on sequences of characters
  - Produced contextual embeddings for words based on the entire sequence of inputs
  - E.g. “Swing a **bat** at the ballpark” versus “Swarmed by **bats** in a cave”

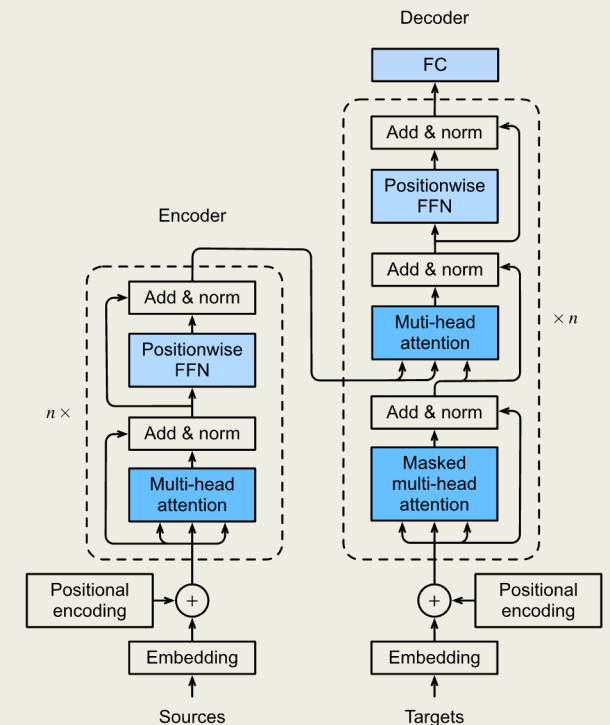


# Transformer Architecture



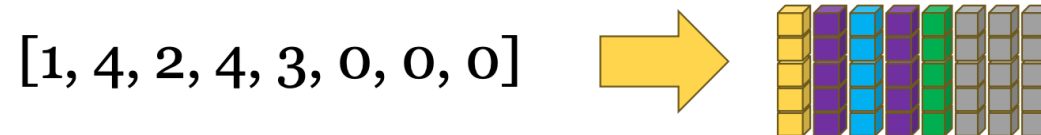
# Features of Transformer Models

- Transformers are class of neural network models that process sequences of vectors
- Architecture introduced in “Attention Is All You Need” (Vaswani et al, 2017)
- Three main stages
  - Input block: Transform raw input into sequences of vectors
  - **Processing blocks: Alternating blocks of attention and multi-layer perceptron blocks**
  - Output blocks: Task-specific layers to transform sequence representation into useful outputs



# Inputs

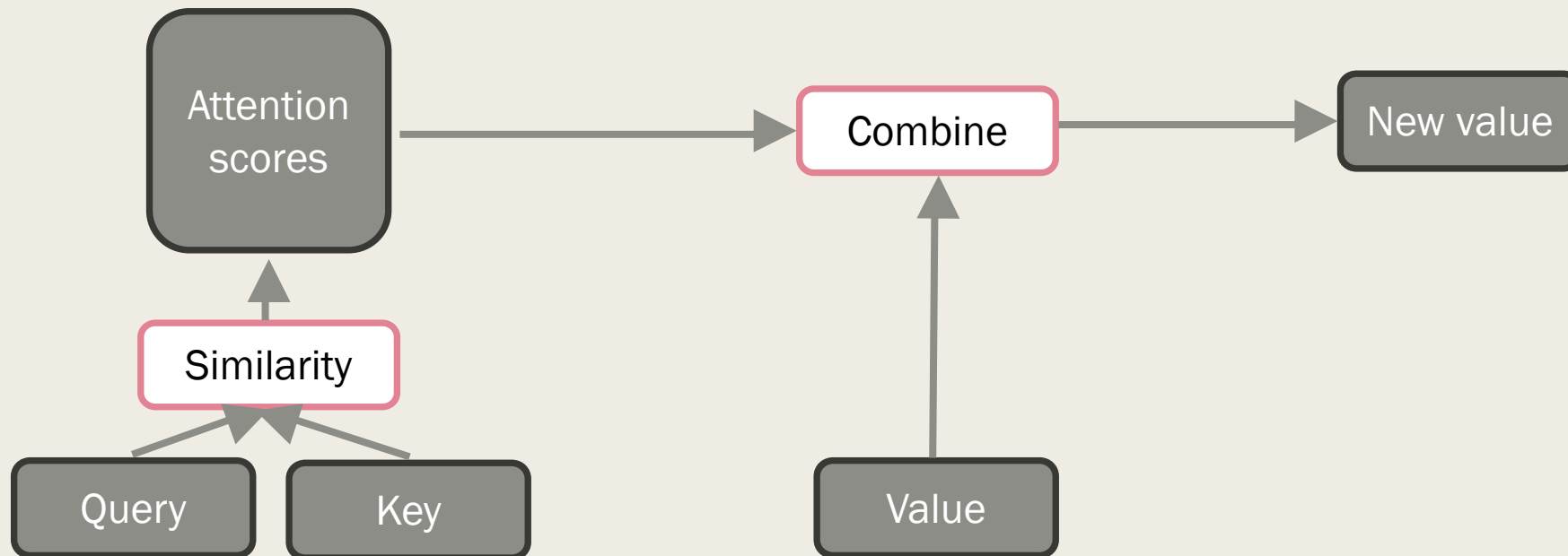
- For NLP, it's not obvious how to transform text into vectors for ML
- Today, we use tokenization to create a finite vocabulary of sub-word “tokens” where each token is assigned an integer
- We use an “embedding layer” to lookup a unique vector for each integer in the input sequence
- “WAT.ai Design Team” → [W][AT][.ai][ Design][ Team] → [54, 835, 41483, 7127, 8068]
  - Each of the integer IDs would correspond to a high-dimensional vector
  - <https://platform.openai.com/tokenizer>



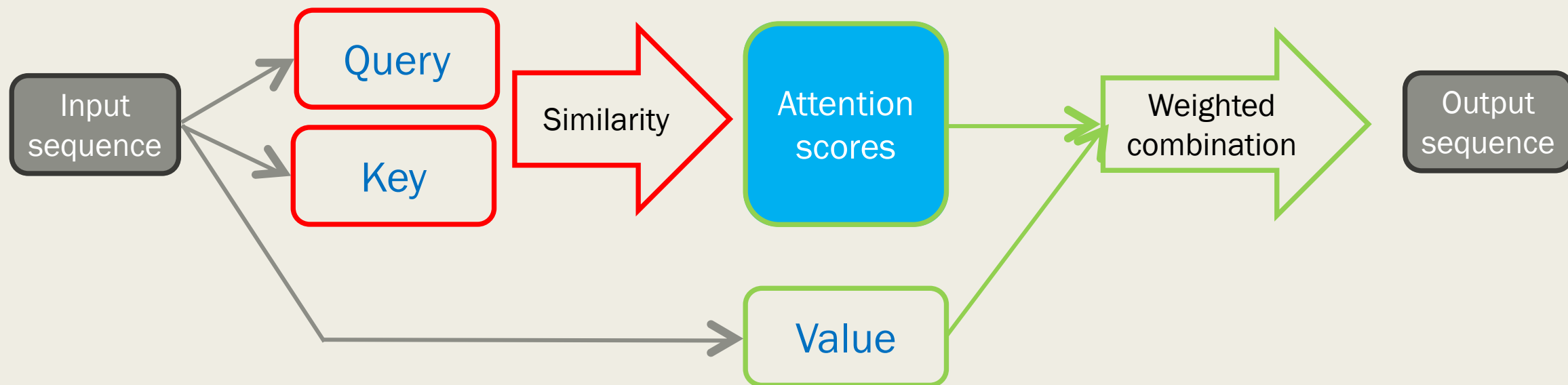


# Attention

- Attention is a mechanism that combines information from different elements in the sequence based on how “similar” they are to each other



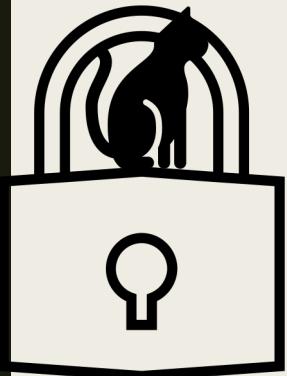
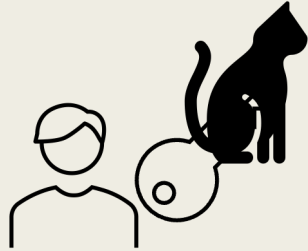
# Attention



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

# How similar is this Query?

Query: Cat



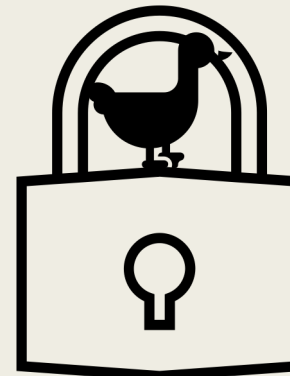
Cat



Lion



Giraffe



Goose



Tiger

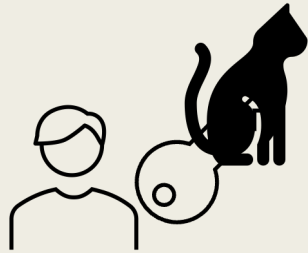
Key



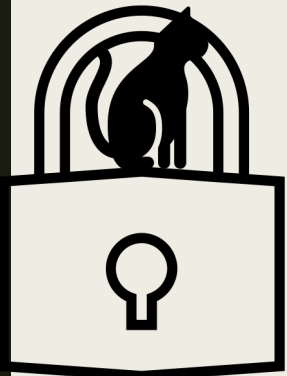
Value

# How similar is this Query?

Query: Cat



4



Cat



2



Lion

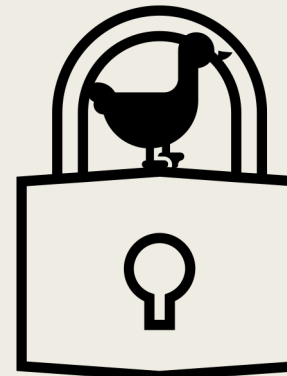


0



Giraffe

0



Goose

2



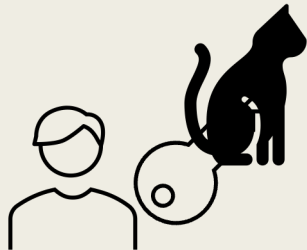
Tiger



Key

Value

Query: Cat



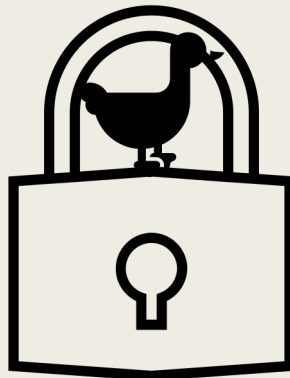
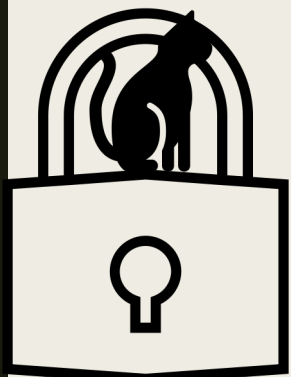
4

2

0

0

2



Key

Cat

Lion

Giraffe

Goose

Tiger



Value



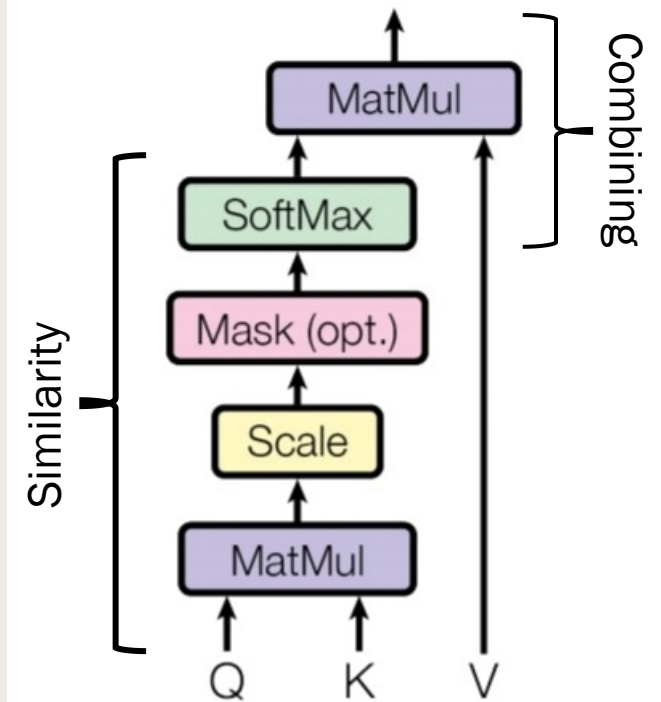
New representation for  
"Cat"

# Attention Equation

- In practice, attention is implemented as scaled dot-product attention
- Multi-head attention, multi-query attention, and grouped-query attention are all extensions of this basic formula that compute parallel attention similarities, which allows for richer representations
- Attention memory usage scales with  $O(n^2)$  for sequence length
- Attention can be masked (causal masks, short sequences)

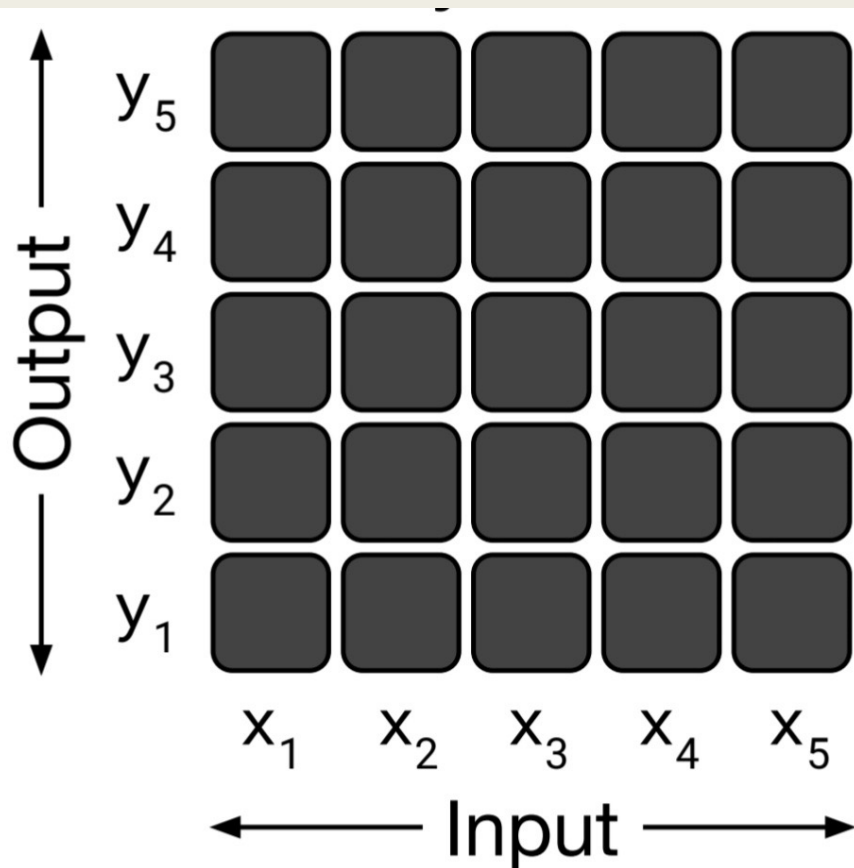
$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

Scaled Dot-Product Attention

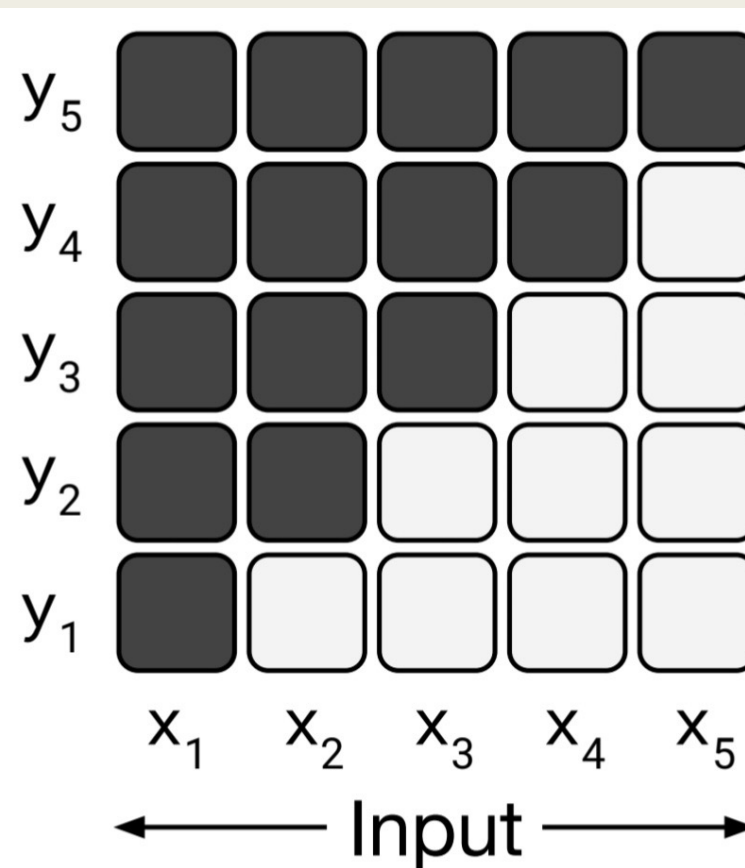


# Attention Masks

Bi-directional, nothing  
masked out

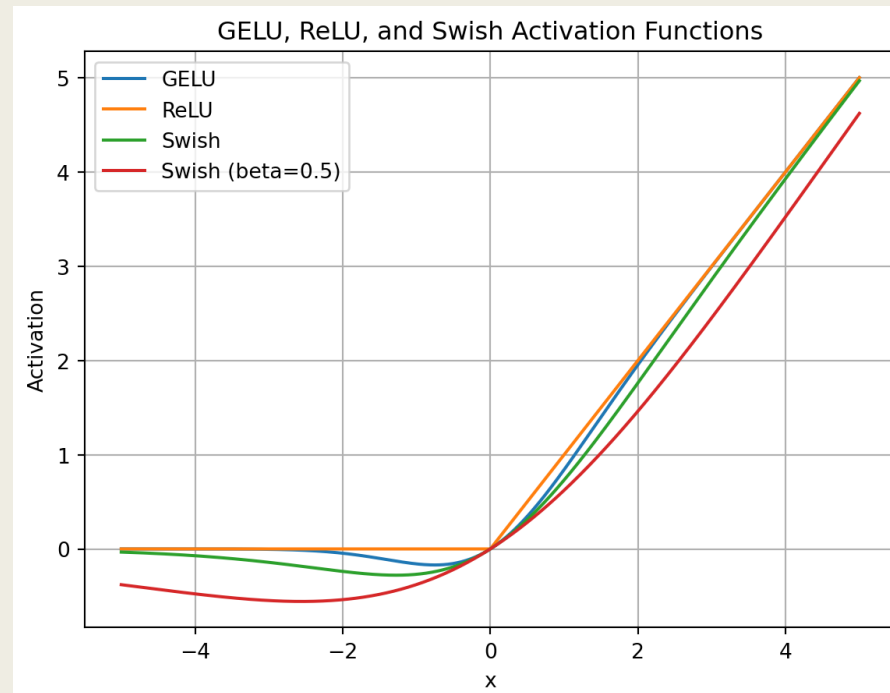


Causal, future tokens  
masked out (white)



# Feed-forward Network (FFN)

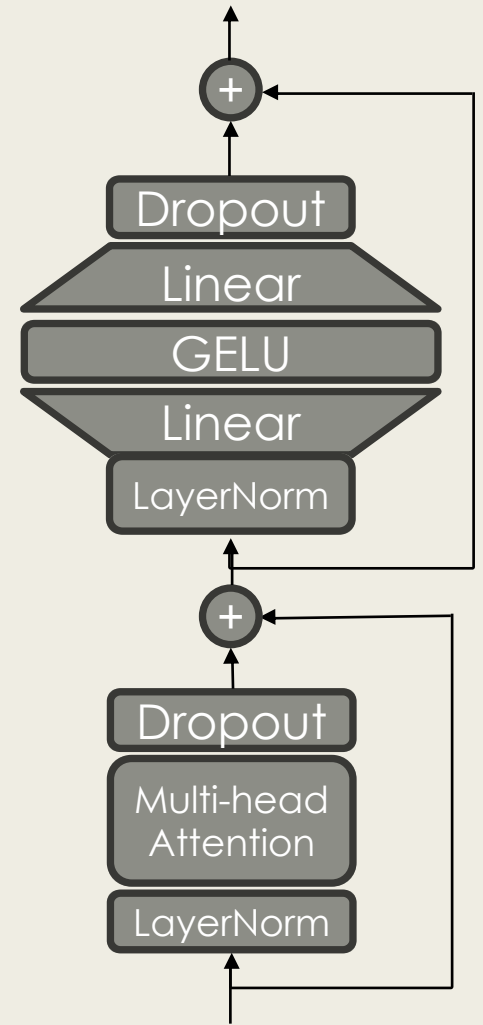
- Multi-layer perceptron with two linear layers and a non-linear activation in between
- First layer is an expansion ( $D \rightarrow 4D$ ) and second layer is a contraction ( $4D \rightarrow D$ )
- The original transformer used ReLU, but modern networks use GeLU or swish



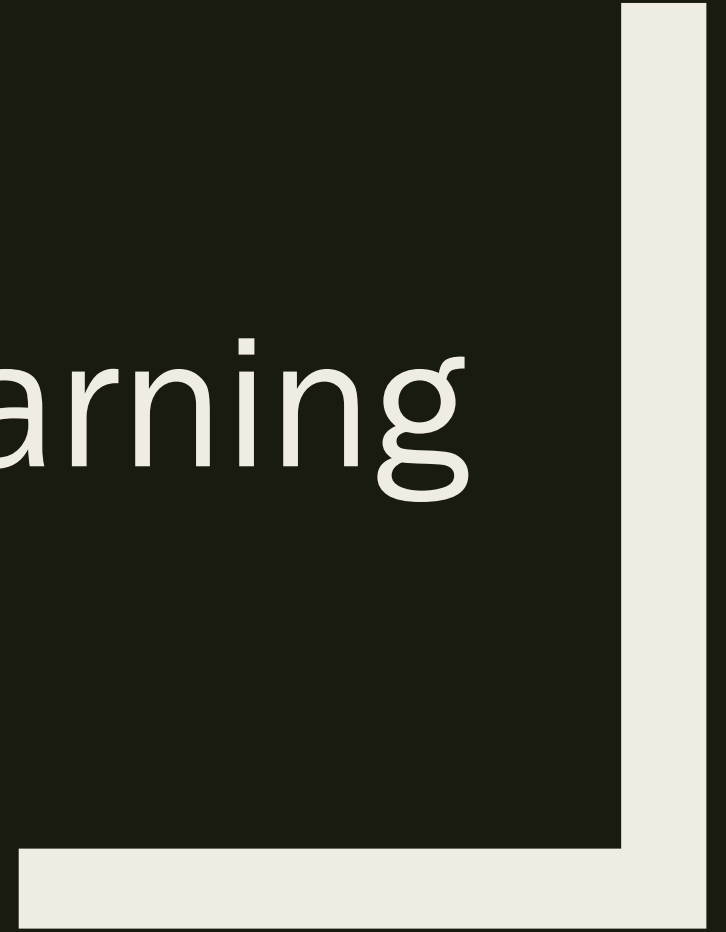


# Other Features

- Dropout and LayerNorm for regularization
- Modern transformer networks use “pre-LayerNorm” residual connections (similar to ResNetV2)



# Transfer Learning



# Transfer Learning

- Transfer learning the process of applying a "pre-trained" model on a different task by further "fine-tuning" it
- The pre-training task generally uses a lot of data and helps the model learn useful information about the general class of problems it will be applied to
- Examples:
  - Pre-training vision models on ImageNet and applying it to classifying pictures of animals on a farm
  - Pre-training language models on internet text corpora and applying them to generate poetry



# Foundation Models

- “Large Foundation Models” have been pre-trained on extremely large datasets with massive amounts of compute
  - This is expensive, but only needs to be done once
  - We can apply the same foundation model to many problems
  - Examples: GPT-4, LLaMA, BERT, DALL-E, Whisper, etc.
- Language models are almost all foundation models pre-trained on large amounts of text
- As an AI developer, we don’t build foundation models, but apply existing ones to solve specific problems

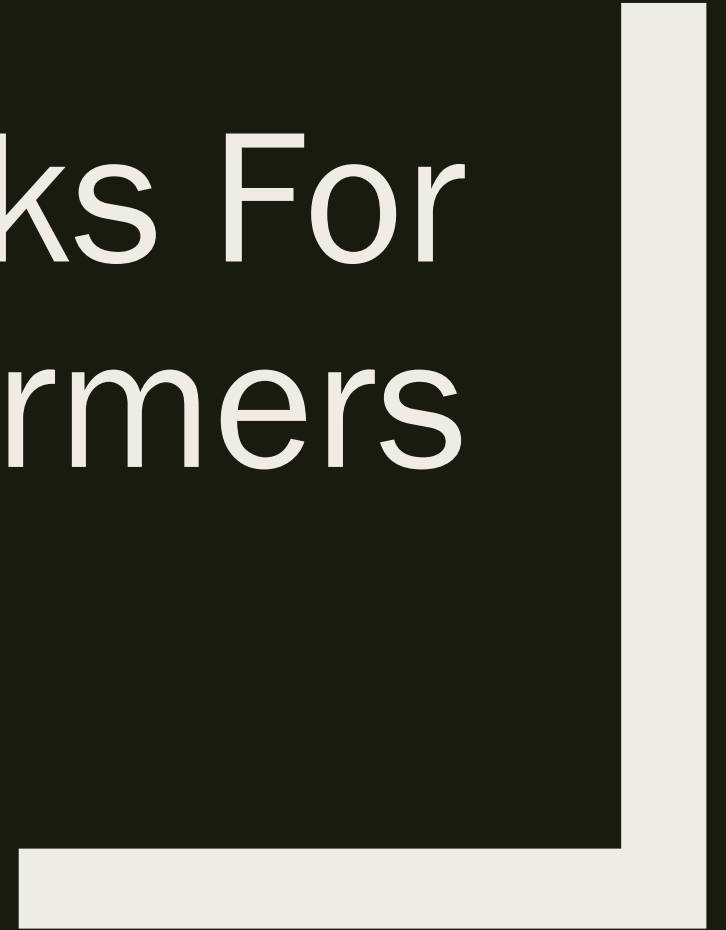
# Pre-training Tasks

- Causal Language Modelling (CLM)
  - Most popular at the moment since it can produce generative models
  - Causal = Only the past affects the future
  - Statistical approach that assumes we can estimate the conditional probability of the next word given the previous words
  - Framed as a classification task over the vocabulary of tokens
- Example
  - “The quick brown fox jumped over the lazy [...]”

# Pre-training Tasks

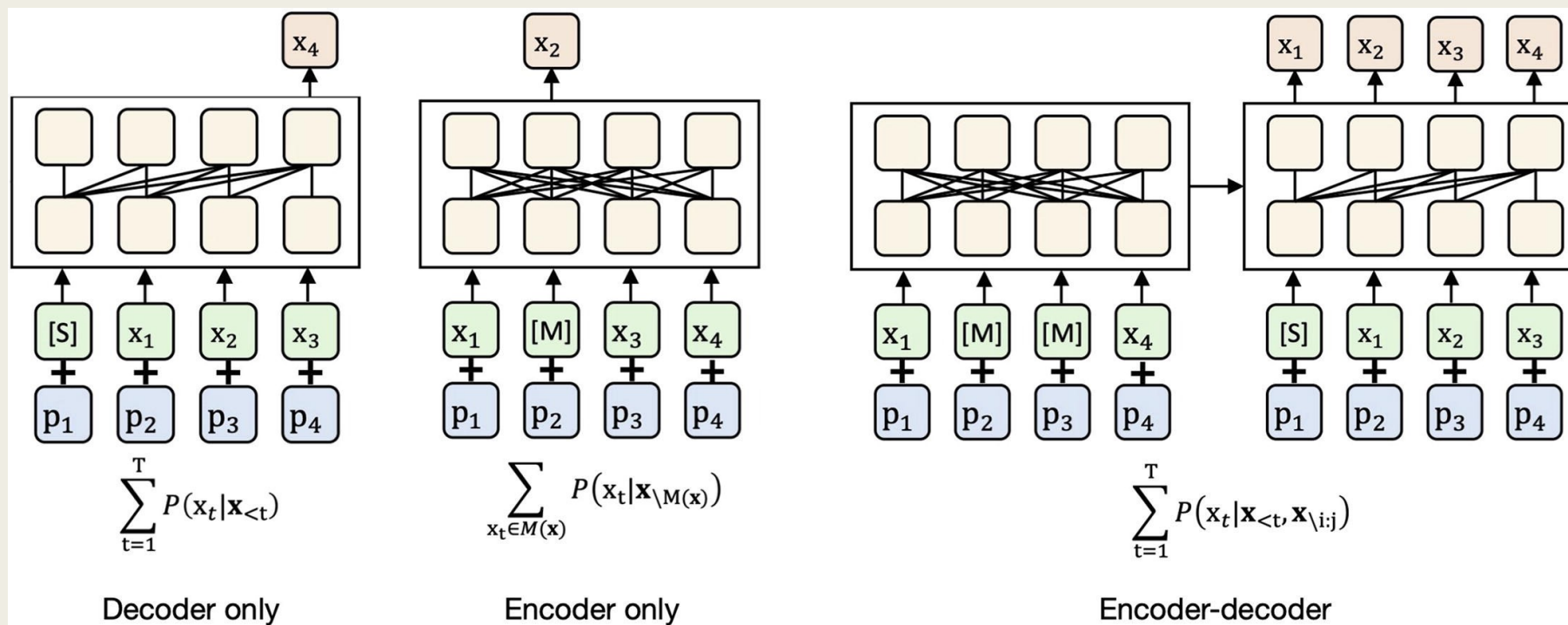
- Masked Language Modelling (MLM)
  - Popularized by the BERT (Devlin et al., 2018) model for pre-training encoders
  - Estimates the probability of predicting a masked token given the context of the entire sequence (past and future)
  - Non-causal task
  - Classification task to predict which vocabulary item should replace the mask token
- Example:
  - “The [MASK] brown fox [MASK] the lazy dog”

# NLP Tasks For Transformers



# Three Architectures

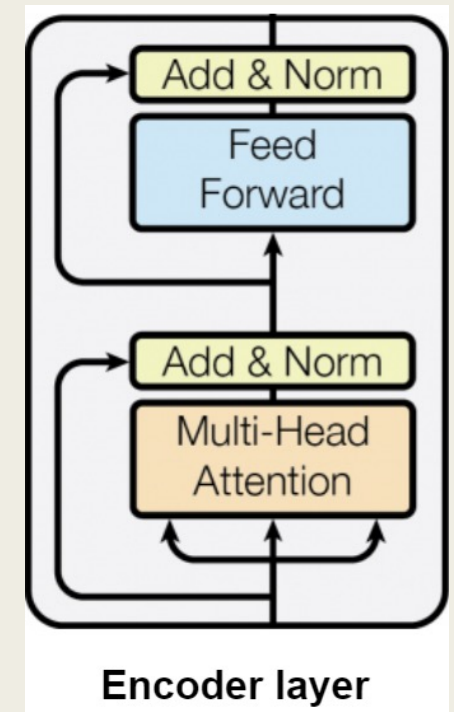
- Vaswani et al. (2017) introduces the “Encoder-Decoder” architecture, but Encoder-only (like BERT) and Decoder-only (like GPT) architectures are more commonly used





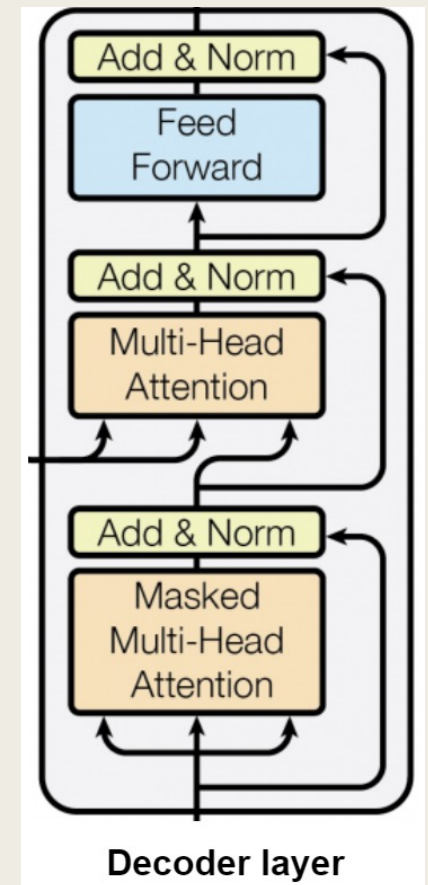
# Encoder-only

- The encoder was popularized by BERT (Devlin et al, 2018)
- Used for sequence classification, token classification, and vector embeddings
- Assumes there is a bi-directional relationship between the sequence, so are non-causal models
  - No generative capabilities, since current inputs have access to future context



# Decoder-only

- The decoder was popularized by GPT (Radford et al., 2018)
- Used mainly for text generation, though it can do sequence classification
  - Bi-directional encoder models often perform better and representation tasks
- Assumes a causal relationship between sequence elements, so an input only has access to past information for attention.
- Auto-regressive generation
- With the CLM assumption, we can feed in sequence elements one at a time to produce a new prediction. We can take that prediction and feed it back into the input, which produces a new output.



# Auto-regressive Generation

- Predict the most likely word given the previous words, repeat with new inputs
  - The quick brown [horse]
  - The quick brown horse [trotted]
  - The quick brown horse trotted [down]
  - The quick brown horse trotted down [the]
  - The quick brown horse trotted down the [road]



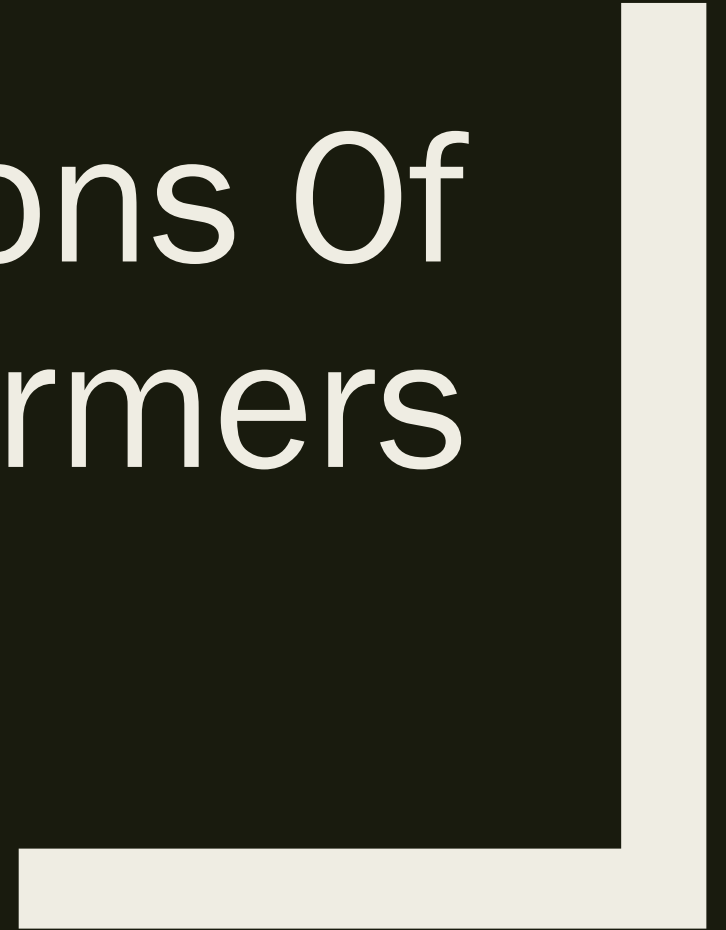
# Encoder-Decoder

- The original transformer architecture is encoder-decoder. Another popular architecture is T5 (Raffel et al., 2020)
- This model can process two different sequences. The encoder sequence is processed for rich representations (encoding). The decoder sequence combines information from the encoder for generation.
- Used for both sequence representation and generation. It can also be used for sequence-to-sequence tasks, like translation.
- Encoder is conditioning the output sequence
  - With advances of LLMs, we can also achieve a similar kind of conditioning through prompting

# Summary

- Encoder-only
  - Vector embeddings, token classification, sequence classification
- Decoder-only
  - Generative tasks
- Encoder-Decoder
  - All of the above, translation

# Limitations Of Transformers



# Statistical Language Modelling

- Transformers assume language is a statistical process. As such, their training inherently assumes that outputs should have the highest "likelihood" and does not enforce "correctness"
- Results in generated content that is false/misleading but presented as a fact (hallucinations)
- When outputs could be ambiguous, predicting what is "likely" may result in false outputs. This might be addressed by including further information in the input sequence



**You**

How many n's are in the word "mayonnaise"?



**ChatGPT**

There are three "n's" in the word "mayonnaise."



**You**

That is incorrect



**ChatGPT**

Apologies for the mistake! You're right; there are actually two "n's" in the word "mayonnaise."



**You**

Can you show me where they are?



**ChatGPT**

Of course! The two "n's" in "mayonnaise" are located after the "y" and after the "a." So, the word "mayonnaise" has two "n's."



# Input Data

- Foundation models are only as good as their input data. When this data contains bias, this can result in biased outputs.
- It's challenging to generate sensible outputs outside the training data distribution.
  - Results in incorrect factual inferences about the world
  - Works best on languages the model has been trained on
  - Limited use of models in specialized domains

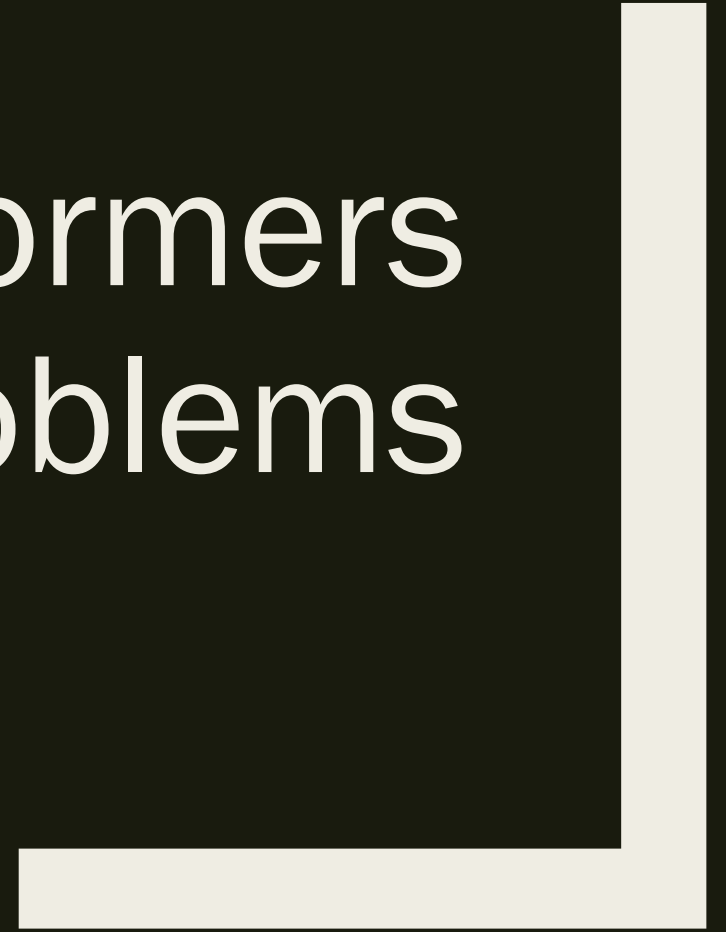
# Sequence Length

- Most models require inputs to be less than the maximum sequence length of training data
  - Early models capped out at 512, recent models tend to be around 4096
- Challenge: Model is trained on a certain distribution of position encodings, going outside that range is challenging to extrapolate effectively
  - Models using ALiBi or RoPE encodings may have interpolation methods that increase effective maximum sequence length in inference
- Memory required to compute attention scales with square of sequence length
- FlashAttention algorithm is used in modern networks to reduce memory burden

# Compute

- Larger models increase performance, but require much more compute to use
- Massive amounts of compute to do pre-training
  - Falcon-40B was pre-trained on 384 A100 40GB GPUs in AWS Sagemaker for several months on 1 trillion tokens
- Often inaccessible amounts of compute for inference
  - 1B: 12 GB GPU
  - 7B: 24 GB GPU
  - 13B: 4x24 GB GPUs
  - 70B: 8x40 GB GPUs
- To get GPU resources on cloud platforms requires approval and is expensive!

# How To Apply Transformers To Practical Problems



# Availability of Transformers Models

- With the availability of so many pre-trained models on e.g. HuggingFace, as AI developers, we spend less time hand-crafting an architecture for the problem, and more time doing engineering and evaluation
- HuggingFace has access to pre-trained models, respective tokenizers, and training/evaluation pipelines in Torch, TensorFlow, and Jax

# Step by Step

1. Identify the NLP task at hand
  1. What data is required?
  2. What are the input sequences?
  3. Does it require generative capabilities?
  4. What budget of compute / time / data do you have?
  5. What is metric for success?
2. Select a transformer architecture
  1. Sequence / token classification / embeddings: Encoder
  2. Generation: Decoder
  3. Translation / conditional generation: Encoder-decoder

# Step by Step

3. Choose a pre-trained foundation model with that architecture that fits limitations
4. Gather examples of labelled data for the task
5. Fine-tune the model or attempt few-shot inference
6. Evaluate outputs based on metric for success

# Other Considerations

- Large models may be able to be trained on smaller compute using PEFT (e.g. QLoRA, prompt tuning, etc.)
- Quantization methods may help run inference on smaller hardware
- Knowledge distillation methods are also possible to transfer knowledge from very large models to smaller models. Very useful for deployment.
- Training large models requires several nodes of 8x GPUs and will require distributed training frameworks like DeepSpeed
- If applying transformers to non-NLP problems, think about how inputs will be turned into sequences of vectors. Does tokenization work? Are they already vectors?



# Bibliography

- [1] A. Vaswani *et al.*, “Attention Is All You Need.” arXiv, Aug. 01, 2023. Accessed: Nov. 23, 2023. [Online]. Available: <http://arxiv.org/abs/1706.03762>
- [2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.” arXiv, Oct. 10, 2018. Accessed: Nov. 23, 2023. [Online]. Available: <http://arxiv.org/abs/1810.04805>
- [3] M. E. Peters *et al.*, “Deep contextualized word representations.” arXiv, Mar. 22, 2018. doi: [10.48550/arXiv.1802.05365](https://doi.org/10.48550/arXiv.1802.05365).
- [4] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, “DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter.” arXiv, Feb. 29, 2020. doi: [10.48550/arXiv.1910.01108](https://doi.org/10.48550/arXiv.1910.01108).
- [5] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient Estimation of Word Representations in Vector Space.” arXiv, Sep. 06, 2013. Accessed: Nov. 23, 2023. [Online]. Available: <http://arxiv.org/abs/1301.3781>
- [6] C. Raffel *et al.*, “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer.” arXiv, Sep. 19, 2023. Accessed: Nov. 23, 2023. [Online]. Available: <http://arxiv.org/abs/1910.10683>
- [7] T. Dao, D. Y. Fu, S. Ermon, A. Rudra, and C. Ré, “FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness.” arXiv, Jun. 23, 2022. Accessed: Nov. 23, 2023. [Online]. Available: <http://arxiv.org/abs/2205.14135>

# Bibliography

- [8] D. Hendrycks and K. Gimpel, “Gaussian Error Linear Units (GELUs).” arXiv, Jun. 05, 2023. Accessed: Nov. 23, 2023. [Online]. Available: <http://arxiv.org/abs/1606.08415>
- [9] K. He, X. Zhang, S. Ren, and J. Sun, “Identity Mappings in Deep Residual Networks.” arXiv, Jul. 25, 2016. doi: [10.48550/arXiv.1603.05027](https://doi.org/10.48550/arXiv.1603.05027).
- [10] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, “Improving Language Understanding by Generative Pre-Training”.
- [11] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, “Language Models are Unsupervised Multitask Learners”.
- [12] H. Touvron *et al.*, “Llama 2: Open Foundation and Fine-Tuned Chat Models.” arXiv, Jul. 19, 2023. doi: [10.48550/arXiv.2307.09288](https://doi.org/10.48550/arXiv.2307.09288).
- [13] R. Sennrich, B. Haddow, and A. Birch, “Neural Machine Translation of Rare Words with Subword Units.” arXiv, Jun. 10, 2016. Accessed: Nov. 23, 2023. [Online]. Available: <http://arxiv.org/abs/1508.07909>
- [14] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, “QLoRA: Efficient Finetuning of Quantized LLMs.” arXiv, May 23, 2023. Accessed: Nov. 23, 2023. [Online]. Available: <http://arxiv.org/abs/2305.14314>

# Bibliography

- [15] J. Su, Y. Lu, S. Pan, A. Murtadha, B. Wen, and Y. Liu, “RoFormer: Enhanced Transformer with Rotary Position Embedding.” arXiv, Nov. 08, 2023. Accessed: Nov. 23, 2023. [Online]. Available: <http://arxiv.org/abs/2104.09864>
- [16] P. Ramachandran, B. Zoph, and Q. V. Le, “Searching for Activation Functions.” arXiv, Oct. 27, 2017. doi: [10.48550/arXiv.1710.05941](https://doi.org/10.48550/arXiv.1710.05941).
- [17] B. Lester, R. Al-Rfou, and N. Constant, “The Power of Scale for Parameter-Efficient Prompt Tuning.” arXiv, Sep. 02, 2021. Accessed: Nov. 23, 2023. [Online]. Available: <http://arxiv.org/abs/2104.08691>
- [18] G. Penedo *et al.*, “The RefinedWeb Dataset for Falcon LLM: Outperforming Curated Corpora with Web Data, and Web Data Only.” arXiv, Jun. 01, 2023. Accessed: Nov. 23, 2023. [Online]. Available: <http://arxiv.org/abs/2306.01116>
- [19] O. Press, N. A. Smith, and M. Lewis, “Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation.” arXiv, Apr. 22, 2022. Accessed: Nov. 23, 2023. [Online]. Available: <http://arxiv.org/abs/2108.12409>
- [20] S. Rajbhandari, J. Rasley, O. Ruwase, and Y. He, “ZeRO: Memory Optimizations Toward Training Trillion Parameter Models.” arXiv, May 13, 2020. doi: [10.48550/arXiv.1910.02054](https://doi.org/10.48550/arXiv.1910.02054).