PUNJAB ENGINEERING COLLEGE

(Deemed to be University) CHANDIGARH

ADVANCED AI PROJECT

A BRIEF REVIEW ON

TRANSFORMERS

&

LARGE LANGUAGE MODELS

BY-

DIVYAM AGRAWAL (20103088)

MUNISH KUMAR (20103064)

RAVEESH GOYAL (20103077)

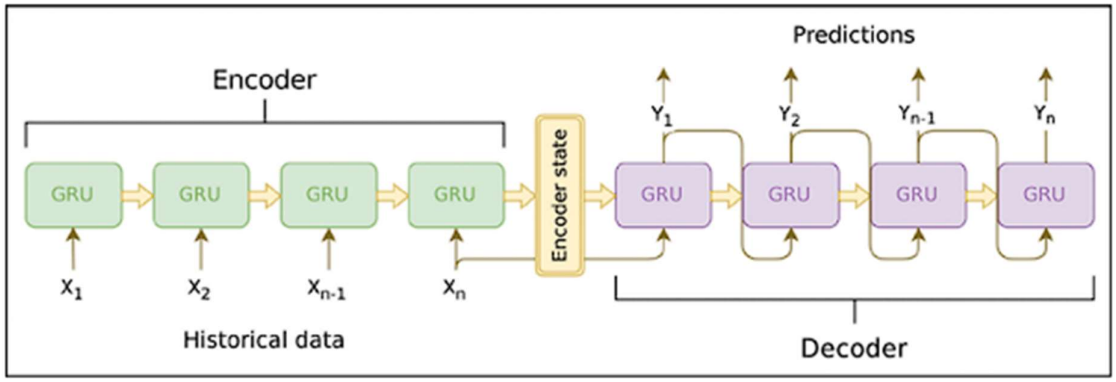
GOURAV SINGLA (20103094)

INDEX

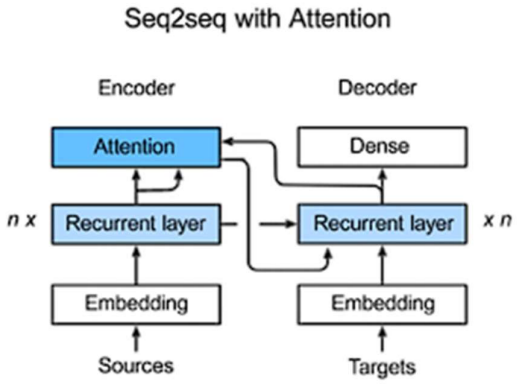
INTRODUCTION TO ATTENTION

1.1 INTRODUCTION

Neural machine translation models belong to a family of encoders and decoders in which the encoders encode the source sentence into a fixed length vector from which decoder generates the translation. This makes it difficult for the encoder to compress all the necessary information from long sentences especially those that are longer than the sentences in the training corpus thus reducing the performance of encoder-decoder models.



Then extension of encoder-decoder models is proposed, in which both learn to align and translate jointly by searching the set of positions of most relevant information in source sentence during translation and then predicts a target word based on context vectors associated with these source positions and the previous generated target words. Here encoder does not encode the input sentence into a single fixed length vector but into sequence of vectors and chooses a sequence of vectors adaptively while decoding the translation.



1.2 RNN STRUCTURE

In RNN encoder-decoder architecture, encoder reads the input sequence of vectors $\mathbf{x} = (x_1, \ldots, x_{Tx})$ into a vector c^2, and processed further as

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

$$c = q(\{h_1, \cdots, h_{T_x}\}),$$

Where h_t is a hidden state at time t and c is a vector generated from the sequence of the hidden states, f and q are some nonlinear functions.

The decoder is trained to predict the next word y_t, given the context vector c and all the previously predicted words $\{y_1, y_2, \ldots, y_{t'-1}\}$, thus decoder defines a probability over the translation $\mathbf{y}$ by decomposing the joint probability into ordered conditionals ,

$$p(\mathbf{y}) = \prod_{t=1}^{T} p(y_t \mid \{y_1, \cdots, y_{t-1}\}, c),$$

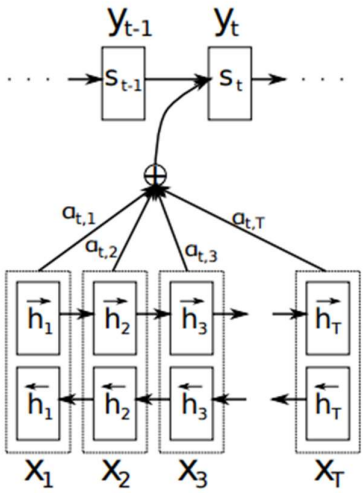
Where $\mathbf{y} = (y_1, \ldots, y_{Ty})$, and each conditional probability is modelled as:

$$p(y_t \mid \{y_1, \cdots, y_{t-1}\}, c) = g(y_{t-1}, s_t, c),$$

Where g is a nonliner multilayer function that outputs the probability of y_t, and s_t is hidden state of the RNN.

1.3 LEARNING TO ALIGN AND TRANSLATE

The new architecture consists of bidirectional RNN as an encoder and a decoder that performs searching through a source sentence during a translation.



1.3.1 DECODER

In the new model, study define each conditional probability as:

$$p(y_i|y_1, \ldots, y_{i-1}, \mathbf{x}) = g(y_{i-1}, s_i, c_i),$$

Where s_i is an RNN hidden state for time i, computed by

$$s_i = f(s_{i-1}, y_{i-1}, c_i).$$

Here, the probability is conditioned on a distinct context vector c_i for each target word y_i, the context vector c_i depends on a sequence of annotations $(h_1, \ldots, h_{Tx})$ to which encoder maps the input sequence. Each annotation h_i contains information of the whole input sequence which strong attention to parts surrounding i^{th} word of the input sequence and then context vector ci is calculated as,

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j.$$

Where weight α_{ij} of each annotation h_j is calculated by,

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})},$$

Where ,

$$e_{ij} = a(s_{i-1}, h_j)$$

Is an *alignment model* which scores the similarity of inputs at position j and outputs at position i. The alignment model a is a feedforward neural network which is jointly trained with the other components of the network. The alignment model computes a soft alignment which makes gradient descent of the cost function to be backpropagated through.

The weighed sum of all annotations is computing a *expected annotation*, where the expectation is over possible alignments. Let α_{ij} be a probability that target word y_i is aligned to source word x_j , then the i^{th} context vector ci is the expected annotation over all annotations.

The probability α_{ij} or its associated energy e_{ij} reflects the significance of annotation h_j with respect to previous hidden state s_{i-1} in deciding the next stage s_i and generating y_i, thus this implements a mechanism of attention in the decoder and relieves the encoder to fit all the information of input sequence in a fixed length vector.

1.3.2 ENCODER

In the above procedure they annotated each word with not only its preceding words, but also the following words, they use a bidirectional RNN (BiRNN).

A BiRNN consists of forward and backward RNN. The forward RNN $\overrightarrow{f}$ reads the input sequence as it is ordered (from x_i to x_{Tx}) and calculates a sequence of forward hidden states $(\overrightarrow{h}_1, \cdots, \overrightarrow{h}_{T_x})$. The backward RNN $\overleftarrow{f}$ reads the sequence in reverse order (from x_{Tx} to x_1) , resulting in a sequence of backward hidden states $(\overleftarrow{h}_1, \cdots, \overleftarrow{h}_{T_x})$.

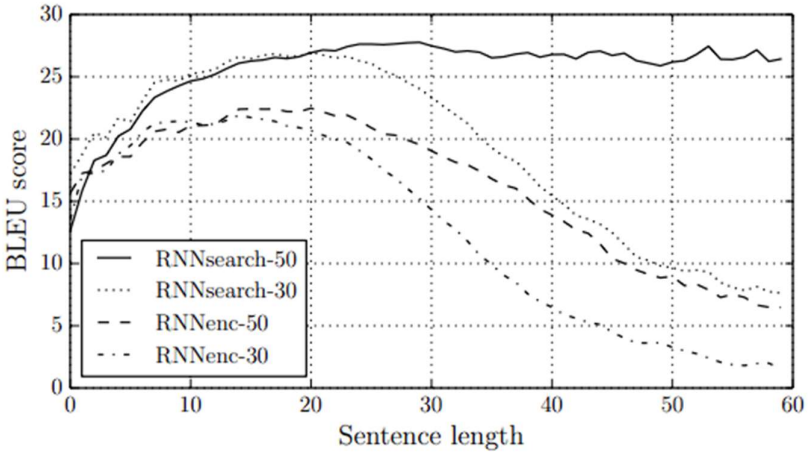
The annotation for each word xj is calculated by concatenating forward hidden state and backward hidden state, i.e. $h_j = \left[\overrightarrow{h}_j^\top; \overleftarrow{h}_j^\top\right]^\top$ that RNNs to better represent recent inputs where annotation h_j is focused on words around x_j which would be used by decoder and alignment model to compute the context vector.

1.4 EXPERIMENT AND ANALYSIS

The above approach was tested on bilingual parallel corpora for the task of English to French translation totalling overall 850 M words.

Two models of the above were trained, first being encoder-decoder RNN model (RNNencdec) and second being the proposed models (RNNsearch). Both models respectively had two variations, one model trained with sentences of length up to 30 words and other with sentence length up to 50 words.

1.4.1 QUANTITATIVE ANALYSIS



We observe that RNNencdec models fail miserably during increase in length of the input sequences whereas RNNsearch models are more robust in comparison to RNNencdec models. We can observe that RNNsearch-50 model is performing well with input sequences longer than 50 words. Also, RNNsearch-30 also performs better than RNNencdec-50.

1.4.2 QUALITATIVE ANALYSIS(ALIGNEMENT)

Here they introduced the concept of soft-alignment which is evidently powerful than hard alignment approach which allows alignment in both directions from the current input sequence and it naturally deals with source and target phrases of different lengths.

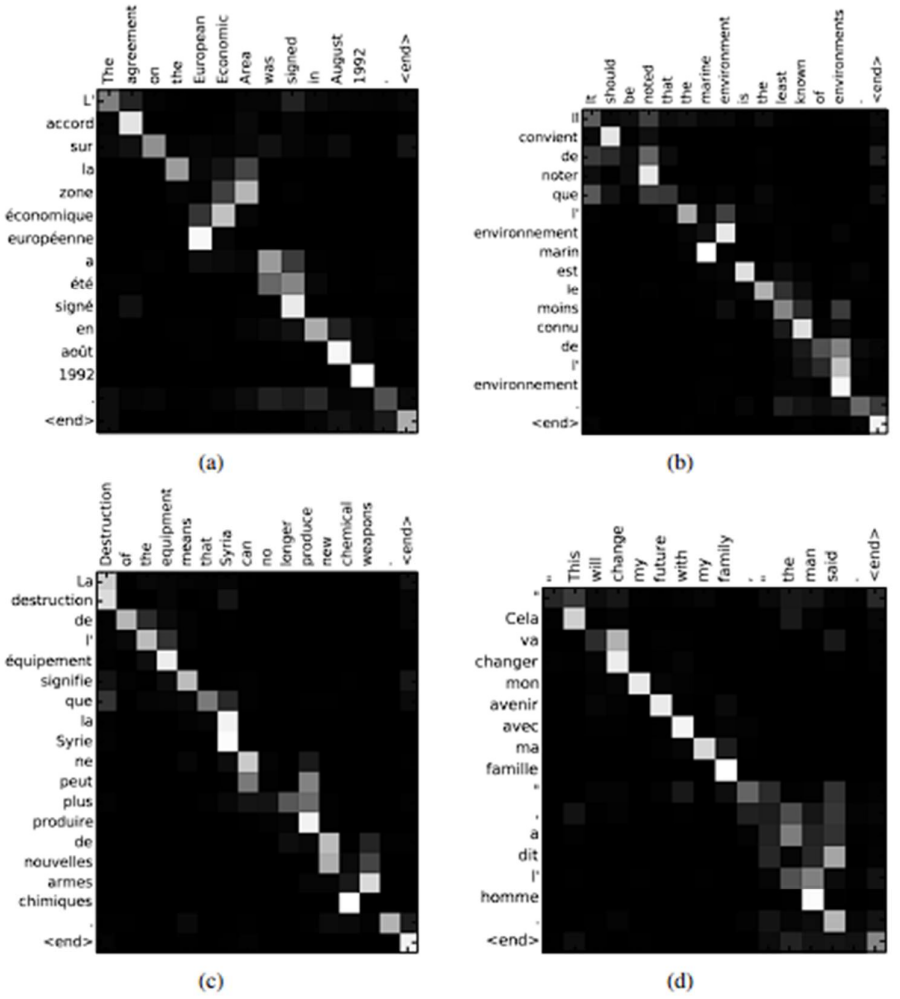


Figure 3: Four sample alignments found by RNNsearch-50. The x-axis and y-axis of each plot correspond to the words in the source sentence (English) and the generated translation (French), respectively. Each pixel shows the weight α_{ij} of the annotation of the j-th source word for the i-th target word (see Eq. (6)), in grayscale (0: black, 1: white). (a) an arbitrary sentence. (b–d) three randomly selected samples among the sentences without any unknown words and of length between 10 and 20 words from the test set.

INTRODUCTION TO TRANSFORMERS

'ATTENTION IS ALL YOU NEED'

2.1 INTRODUCTION

Reinforcement networks, such as LSTM and GatedRNNs, are dominant in the context of sequence modelling: they perform particularly well at tasks like language modelling. The Transformer, a new model architecture which uses only attentional mechanisms to disaggregate from timing computations, is introduced in this paper. It allows for an effective parallelisation that results in significant performance improvements and marks a major step forward with sequence modelling and transduction.

2.2 BACKGROUND

In order to deliberately reduce sequential computing, the Extended Neural GPU, ByteNet, and ConvS2S models employ parallel convolutional neural networks in sequence-oriented applications. The Transformer introduces a novel paradigm that uses self-attention mechanisms, departing from conventional approaches and enabling efficient learning of long-range relationships. Self-attention differs from conventional sequence-aligned RNNs or convolutions in that its adaptability is clear across a variety of activities. The Transformer's skill in transduction is highlighted by this transformative method, which also broadens the Transformer's use to a variety of linguistic tasks while amplifying its effectiveness in the field of sequence modelling.

2.3 MODEL ARCHITECTURE

Encoder-decoder structures are utilised by the majority of competitive neural sequence transduction models. The encoder converts continuous representations $(z_1,\ldots,z_n)$ to input symbols $(x_1,\ldots,x_n)$. The encoder and decoder in the Transformer are fully connected layers with stacked self-attention.

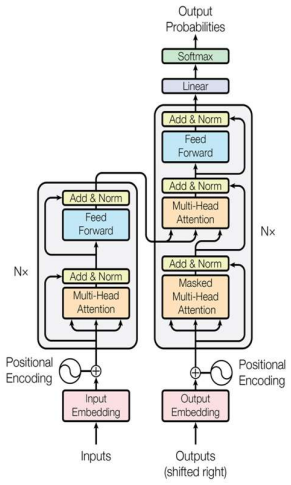


Figure 1: The Transformer - model architecture.

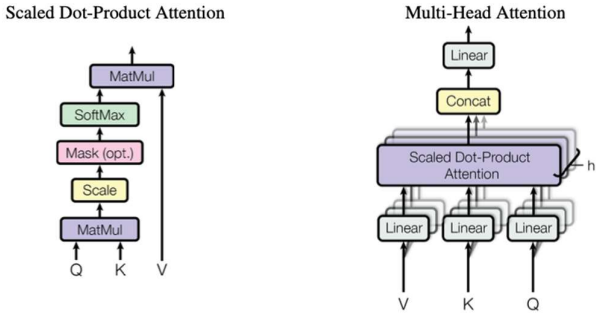
2.3.1 ENCODER AND DECODER STACKS

The encoder consists of N = 6 layers, each of which has a position-wise feed-forward network and a multi-head self-attention sub-layer. Layer normalisation and residual connections are employed. The outputs from each sublayer have the dimension d model = 512.

A third sub-layer is added to the decoder, which likewise has N = 6 layers, to allow for multi-head attention to encoder outputs. Predictions are aided by the self-attention sub-layer in the decoder, which inhibits paying attention to following places.

2.3.2 ATTENTION

A query, a set of key-value pairs, and an output, all of which are vectors, can be mapped to one another by an attention function. The result is calculated as a weighted sum of the values, with each value's weight determined by the query's compatibility function with its corresponding key.



- Scaled Dot-Product Attention: Our "Scaled Dot Product Attention" deals with d-dimensional queries and keys, as well as values. The dot product of queries and keys is divided by a value of √d, and then softmax is used to calculate value weights. This is a much more efficient approach than additive attention, particularly when dealing with large dk values.

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

- Multi-Head Attention: Instead of using a single D model-dimensional attention function they used h = 8 multidimensional attention layers. Questions, keys and values are projected h times to Dk, Dk, and Dv dimensions. The outputs are combined and projected to get the final values. This allows joint attention to different information subspaces.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

- Applications of Attention in our Model: The Transformer employs multi-head attention in three ways:

- In "encoder-decoder attention" layers, queries are from the previous decoder layer, memory keys/values from the encoder output.
- decoder layer, memory keys/values from the encoder output.
- Encoders use self-attention, where keys/queries/values come from the previous encoder layer.
- Decoders have self-attention, with masking to prevent leftward flow.

2.3.3 POSITION-WISE FEED-FORWARD NETWORKS

In addition to the attention sub-layer(s), each of the layers(s) in our encoder(s) and decoder(s) includes a fully integrated feed-forward network(s), which is applied independently and simultaneously to each position(s). This network(s) consist of two linear transformations, with ReLU activation(s) in between.

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

2.3.4 EMBEDDINGS AND SOFTMAX

Like other sequence transduction models, they utilised learned embeddings to convert input and output tokens into dmodel-dimensional vectors. They are using the standard linear transformation and softmax processes to produce decoder outputs. Weight matrix sharing is applied to both embedding layers and pre-softmax linear transformation. dModel scales the embedded layer weights.

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

2.3.5 POSITIONAL ENCODING

For sequence order, our non-recurrent, non-convolutional model needs positional information. With "positional encodings," they improved the input embeddings in the encoder and decoder. The sine and cosine encodings aid the model's comprehension of relative positions. As an alternative, learned positional embeddings produce comparable outcomes.

2.4 WHY SELF-ATTENTION

Convolutional, recurrent, and self-attentional layers are contrasted in this section's mapping of variable-length sequences (xi) to equivalent sequences (zi). For long-distance dependencies, they evaluate computational complexity, parallelizability, and

path lengths. Self-attention outperforms recurrent layers' O(n) sequential operations by delivering constant operations between all positions. Self-attention performs better for n d (which is typically true for sentence representations in machine translation). Self-attention may take into account a small input neighbourhood to accommodate longer sequences. Convolutional layers are more expensive than self-attention because they require O(n/k) stacks for full connection. Convolutions with separable components are simpler. Self-attention may also produce more interpretable models; examples are provided in the appendix.

2.5 RESULTS

They tweaked the newstest2013 base model for English-to-German translation in order to evaluate the Transformer's components. The findings in the table illustrate quality trade-offs with various attention heads. Quality is negatively impacted by key dimension reduction in (B), indicating the need for a more sophisticated compatibility function. Performance is enhanced by larger models and dropout (C). Similar results are obtained when learned embeddings (E) are substituted for sinusoidal positional encoding.

	N	d_{model}	d_{ff}	h	d_k	d_v	P_{drop}	ϵ_{ls}	train steps	PPL (dev)	BLEU (dev)	params $\times 10^6$
base	6	512	2048	8	64	64	0.1	0.1	100K	4.92	25.8	65
(A)				1	512	512				5.29	24.9	
				4	128	128				5.00	25.5	
				16	32	32				4.91	25.8	
				32	16	16				5.01	25.4	
(B)					16					5.16	25.1	58
					32					5.01	25.4	60
(C)	2									6.11	23.7	36
	4									5.19	25.3	50
	8									4.88	25.5	80
		256			32	32				5.75	24.5	28
		1024			128	128				4.66	26.0	168
			1024							5.12	25.4	53
			4096							4.75	26.2	90
(D)							0.0			5.77	24.6	
							0.2			4.95	25.5	
								0.0		4.67	25.3	
								0.2		5.47	25.7	
(E)	positional embedding instead of sinusoids									4.92	25.7	
big	6	1024	4096	16			0.3		300K	**4.33**	**26.4**	213

2.6 CONCLUSION

This study replaces recurrent layers with the Transformer, the first all-attention sequence model. The Transformer excels at translation tasks in terms of performance and speed, producing cutting-edge outcomes. The code for future plans is available on GitHub and includes a variety of tasks and modalities.

Attention Visualizations

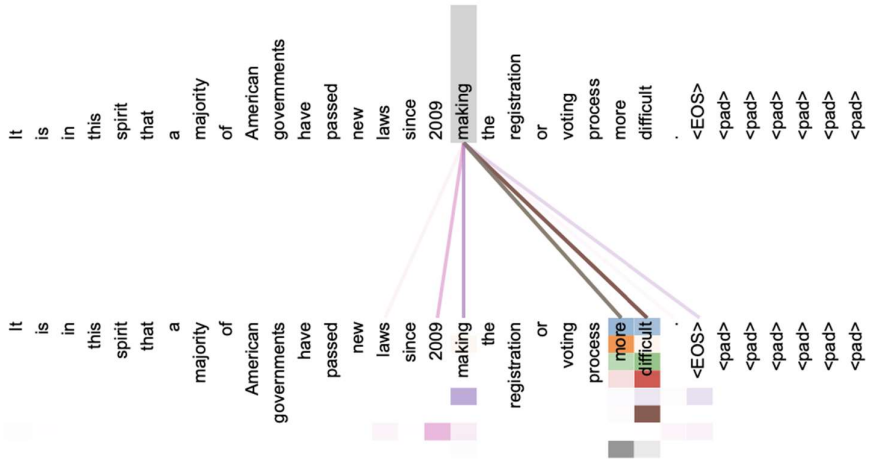
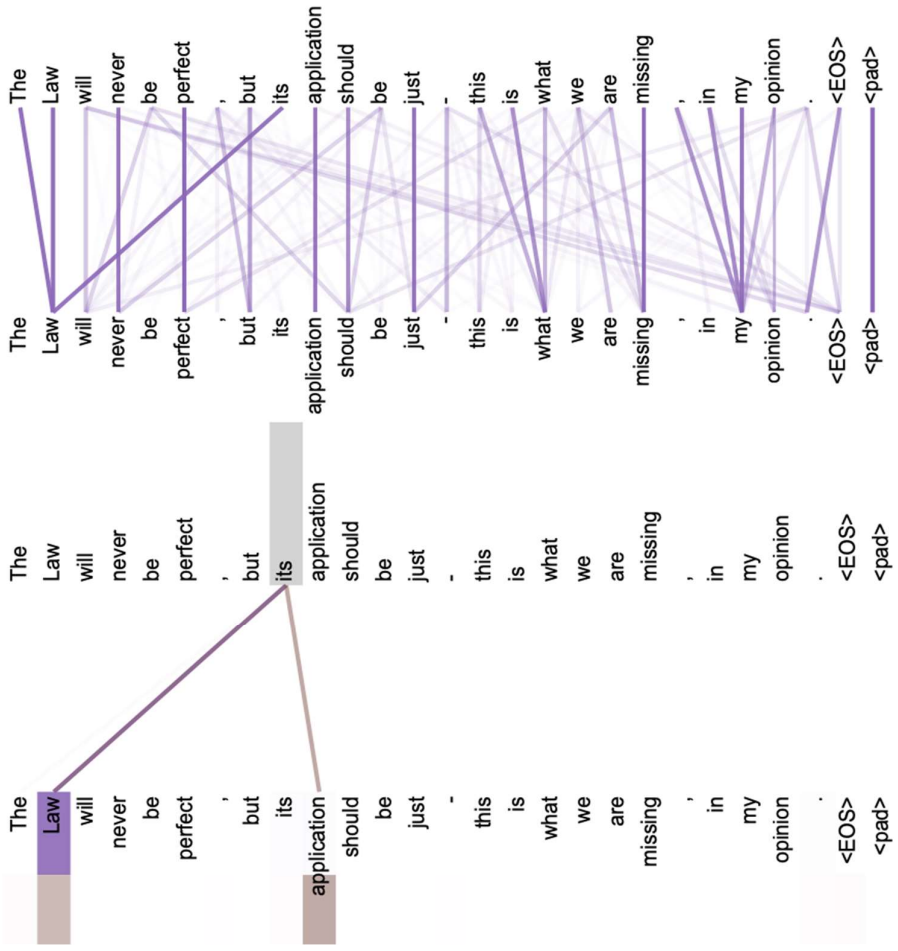


Figure 3: An example of the attention mechanism following long-distance dependencies in the encoder self-attention in layer 5 of 6. Many of the attention heads attend to a distant dependency of the verb 'making', completing the phrase 'making...more difficult'. Attentions here shown only for the word 'making'. Different colors represent different heads. Best viewed in color.



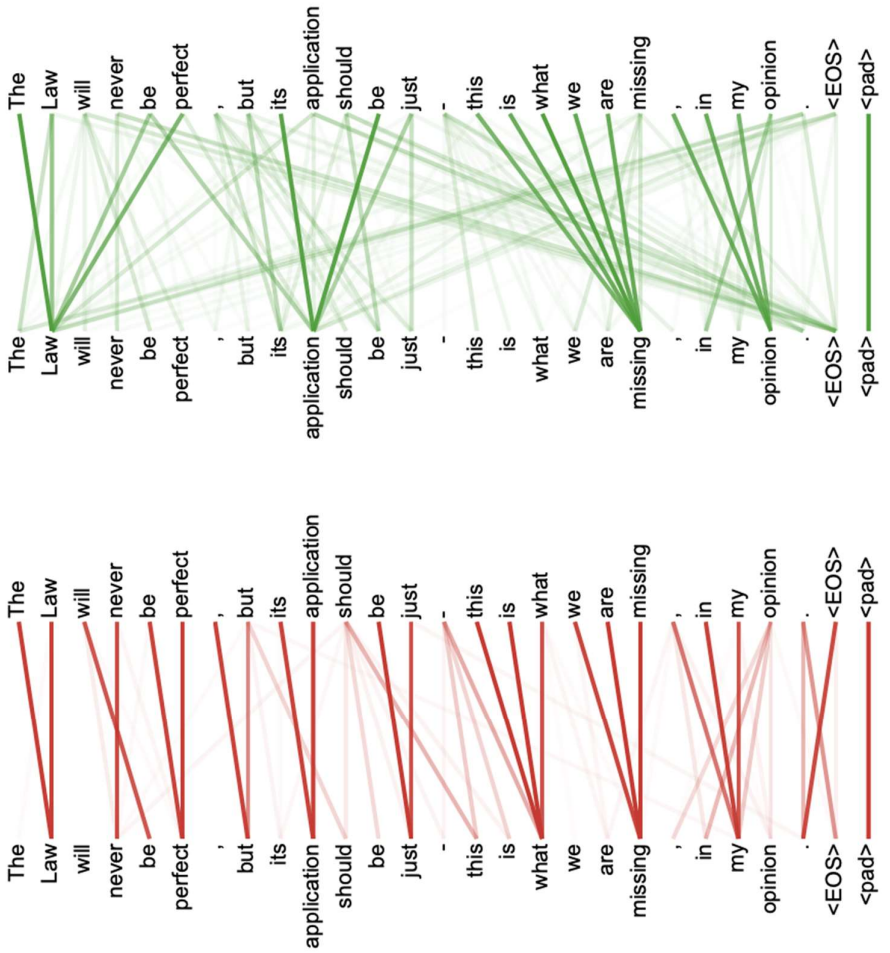


Figure 5: Many of the attention heads exhibit behaviour that seems related to the structure of the sentence. We give two such examples above, from two different heads from the encoder self-attention at layer 5 of 6. The heads clearly learned to perform different tasks.

IMPROVING LANGUAGE UNDERSTANDING BY GENERATIVE PRE-TRAINING

3.1 INTRODUCTION

Natural Language Processing is predominantly a supervised learning task but due to scarcity of manually labelled datasets, there application has been restricted in domains with less labelled data. We can leverage the linguistic information from unlabeled information from the unlabeled dataset in these conditions. Significant performance boost has been observed in learning good representations through unsupervised fashion.

But, utilizing linguistic information is challenging from unlabeled text for two reasons:

- It is unclear what optimization objectives are most effective at learning text representations useful for transfer
- There is no effective way to transfer the representations to the target task.

A semi-supervised approach for language understanding combines a unsupervised pre-training and supervised fine-tuning and the goal is to learn universal representation that adapts to wide range of tasks where target tasks are not required to be of same domain from the source unlabeled corpus.

Transformer model architecture is used to perform strongly on various NLP tasks. During fine-tuning, the task specific input adaptations are derived from traversal style approaches where structured text input is processed as a single contiguous sequence of tokens.

3.2 MODEL FRAMEWORK

3.2.1 UNSUPERVISED PRE-TRAINING

They use a standard language modelling objective over a unsupervised corpus of tokens $U = \{u_1, u_2, \ldots, u_n\}$ to maximise the likelihood:

$$L_1(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \ldots, u_{i-1}; \Theta)$$

Where k is the size of the context window and the conditional probability P is modelled using a neural network with parameters Θ trained using stochastic gradient descent.

They use a Transformer Decoder which applies multi-headed self-attention operation over a input context tokens followed by position wise feedforwaed layers to produce an output distribution over target tokens:

$$h_0 = UW_e + W_p$$
$$h_l = \texttt{transformer\_block}(h_{l-1}) \forall i \in [1, n]$$
$$P(u) = \texttt{softmax}(h_n W_e^T)$$

Where $U = \{u_{-k}, \ldots, u_{-1}\}$ is the context vector of tokens, n is the number of layers, W_e is the token embedding matrix, W_p is the position embedding matrix.

3.2.2 SUPERVISED FINE-TUNING

After training the model with the objective, they fine-tuned the parameters of the target task using a labelled dataset C, with a sequence of input tokens, $\{x^1, \ldots, x^m\}$ along with label y. These inputs are fed into the transformer and the activations of the final transformer block are obtained and are fed into added linear output layer with parameters W_y to predict y:

$$P(y|x^1, \ldots, x^m) = \texttt{softmax}(h_l^m W_y).$$

Which gives the objective to maximise :

$$L_2(\mathcal{C}) = \sum_{(x,y)} \log P(y|x^1, \ldots, x^m).$$

Including language modelling as an auxillary objective to fine-tuning helped learning:

- Improving generalization of the model
- Accelerating convergence

Also, they optimised the objective (with weight λ) :

$$L_3(\mathcal{C}) = L_2(\mathcal{C}) + \lambda * L_1(\mathcal{C})$$

The only extra parameters we require during fine-tuning are W_y and embeddings for delimiter tokens.

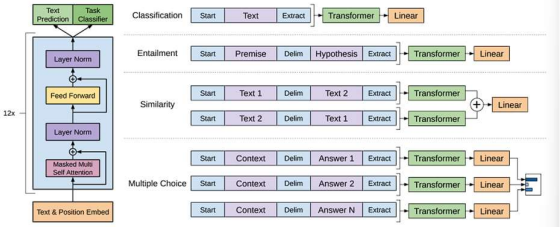


Figure 1: **(left)** Transformer architecture and training objectives used in this work. **(right)** Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer.

3.3 EXPERIMENTS

Various experiments were performed over a variety of NLP supervised tasks to benchmark the pre-trained model.

3.3.1 NATURAL LANGUAGE INFERENCE

This task included recognizing textual entailment, involved reading a pair of sentences and judging relationship between them. The pre-trained model was evaluated on 5 datasets which contain image captions(SNLI), Wikipedia articles(QNLI), government reports(MNLI), science exams(SciTail) or news articles(RTE) and compared with other state-of-the-art models.

Method	MNLI-m	MNLI-mm	SNLI	SciTail	QNLI	RTE
ESIM + ELMo [44] (5x)	-	-	89.3	-	-	-
CAFE [58] (5x)	80.2	79.0	89.3	-	-	-
Stochastic Answer Network [35] (3x)	80.6	80.1	-	-	-	-
CAFE [58]	78.7	77.9	88.5	83.3		
GenSen [64]	71.4	71.3	-	-	82.3	59.2
Multi-task BiLSTM + Attn [64]	72.2	72.1	-	-	82.1	**61.7**
Finetuned Transformer LM [ours]	**82.1**	**81.4**	**89.9**	**88.3**	**88.1**	56.0

3.3.2 QUESTION ANSWERING AND COMMON-SENSE REASONING

This task requires single and multi-sentence reasoning. The datasets include English passages from middle and high school (RACE) and a dataset that involves selecting correct ending to multi-sentence stories from two options (Story Cloze Test) and compare with other state-of-the-art models.

Method	Story Cloze	RACE-m	RACE-h	RACE
val-LS-skip [55]	76.5	-	-	-
Hidden Coherence Model [7]	77.6	-	-	-
Dynamic Fusion Net [67] (9x)	-	55.6	49.4	51.2
BiAttention MRU [59] (9x)	-	60.2	50.3	53.3
Finetuned Transformer LM [ours]	**86.5**	**62.9**	**57.4**	**59.0**

3.3.3 SEMANTIC SIMILARITY

This task involves predicting whether two sentences are semantically equivalent or not where challenges lei in recognizing rephrasing of concepts understanding negation and handling syntactic ambiguity. They used Microsoft Paraphrase corpus (MRPC), Quora Question Pairs (QQP) and the Semantic Textual Similarity benchmark (STS-B) and compare with other state-of-the-art models.

3.3.4 CLASSIFICATION

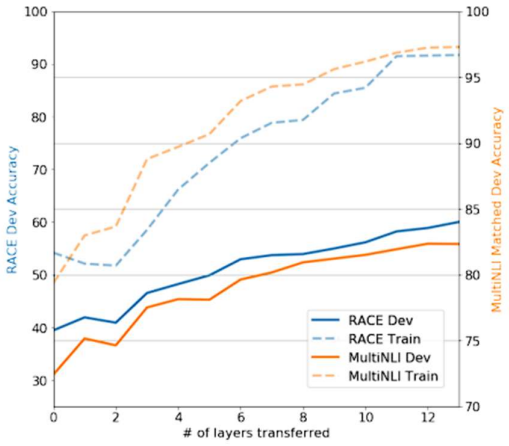
Classification tasks were performed over Corpus of Linguistic Acceptability (CoLA) whih contains expert judgements on whether a sentence is grammatical correct or not and Standford Sentiment Treebank (SST-2) and compare with other state-of-the-art models.

Method	Classification		Semantic Similarity		
	CoLA (mc)	SST2 (acc)	MRPC (F1)	STSB (pc)	QQP (F1)
Sparse byte mLSTM [16]	-	**93.2**	-	-	-
TF-KLD [23]	-	-	**86.0**	-	-
ECNU (mixed ensemble) [60]	-	-	-	81.0	-
Single-task BiLSTM + ELMo + Attn [64]	35.0	90.2	80.2	55.5	66.1
Multi-task BiLSTM + ELMo + Attn [64]	18.9	91.6	83.5	72.8	63.3
Finetuned Transformer LM (ours)	**45.4**	91.3	82.3	**82.0**	**70.3**

3.4 ANALYSIS

3.4.1 IMPACT OF NUMBER OF LAYERS TRANSFERRED

The performance in various supervised tasks increased with increase in number of layers pre-trained models. This indicates that each layer during pre-training learns useful functionality for solving target tasks.



3.4.2 ZERO-SHOT BEHAVIOURS

The underlying generative model learns many of the tasks They evaluate on in order to improve its language modelling and the attentional memory of the transformer assists in transfer compared to LSTMs. Also, LSTMs exhibits higher variance in its zero-shot performance suggesting that the inductive bias of the Transformer architecture assists in transfer.

TRAINING COMPUTE OPTIMAL LARGE LANGUAGE MODELS

4.1 INTRODUCTION

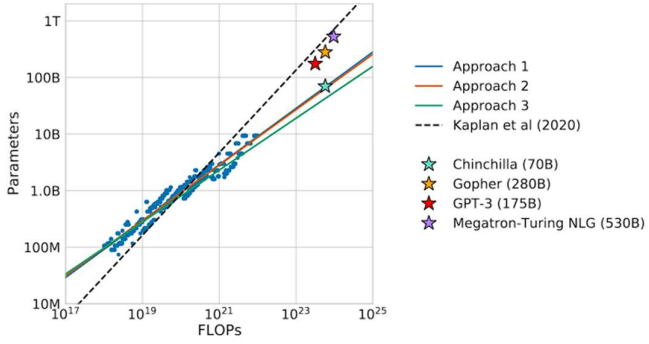
Recently various large language models are introduced having more than 500B parameters. These autoregressive transformers have showed great performance but their training and inference costs are rapidly increasing with their increase in size.

There is power law relationship between number of parameters in an autoregressive large language model and its performance. Given a 10× increase in computational budget, models size increases by 5.5× and performance by 1.8× that is disproportionate. There is a conclusion that large models should not be trained to their lowest possible loss to be compute optimal. Many of recent LLMs have been trained on 300B tokens in line with increase in budget to increase model size.

The question arises given a fixed number of compute capacity, what is the trade-off in model size and number of training tokens? To study this the study make a final pre-training loss $L(N, D)$, where N is number of parameter and D is number of training tokens, Since compute budget C is a function of FLOPs (N,D) , they minimize L under the condition FLOPs $(N,D) = C$.

$$N_{opt}(C), D_{opt}(C) = \underset{N,D \text{ s.t. } \mathrm{FLOPs}(N,D)=C}{\operatorname{argmin}} L(N, D).$$

Various models were estimated based on different number of parameters size and training tokens. The study estimated that based on the compute budget of LLM *Gopher*, a 280B parameter model trained on 300 B tokens, the optimal model should be four times smaller and be trained on 4 times more tokens.



Chinchilla, is a 70B model trained on 1.4 trillion tokens and outperforms Gopher and reduce the inference cost.

4.2 ESTIMATING OPTIMAL PARAMETER/TOKEN ALLOCATION

To find out the trade-off estimate between parameter size and training tokens, three approaches were studied and all of their suggestions contrast the previous works.

4.2.1 FIX MODEL SIZES AND VARY TRAINING TOKENS

They trained different size models with 4 different sizes of tokens and then find which model size gets least loss at a required compute power with required token size. Finally, they fit power laws to estimate the optimal model size and number of training tokens by given amount of compute, obtaining the relationship $N_{opt} \propto C^{a}$ and $D_{opt} \propto C^{b}$ and $a = 0.5$ and $b = 0.5$ are found.

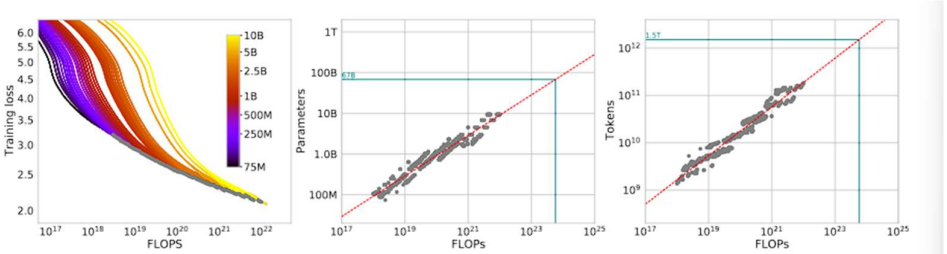


Figure 2 | **Training curve envelope.** On the **left** we show all of our different runs. We launched a range of model sizes going from 70M to 10B, each for four different cosine cycle lengths. From these curves, we extracted the envelope of minimal loss per FLOP, and we used these points to estimate the optimal model size (**center**) for a given compute budget and the optimal number of training tokens (**right**). In green, we show projections of optimal model size and training token count based on the number of FLOPs used to train *Gopher* (5.76×10^{23}).

4.2.2 IsoFLOP PROFILES

In this study, they vary the model size on a fixed set on 9 FLOP counts to find out best model size for each compute budget. They estimate for each FLOP count where there is least training loss due to model size and then fit a power law between FLOPs and loss-optimal model-size and number of training tokens, in the relationship, $N_{opt} \propto C^{a}$ and $D_{opt} \propto C^{b}$ and $a = 0.49$ and $b = 0.51$ are found.

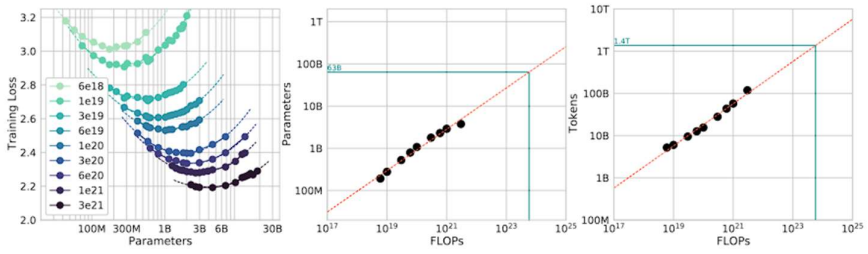


Figure 3 | **IsoFLOP curves.** For various model sizes, we choose the number of training tokens such that the final FLOPs is a constant. The cosine cycle length is set to match the target FLOP count. We find a clear valley in loss, meaning that for a given FLOP budget there is an optimal model to train (**left**). Using the location of these valleys, we project optimal model size and number of tokens for larger models (**center** and **right**). In green, we show the estimated number of parameters and tokens for an *optimal* model trained with the compute budget of *Gopher*.

4.3.3 FITTING A PARAMETRIC LOSS FUNCTION

In this approach, they model all final losses from Approaches 1 and 2 as a parametric function of model parameter count and number of seen tokens.

$$\hat{L}(N, D) \triangleq E + \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}}.$$

In which takes first term takes account loss for generative data distribution, second takes account that a perfectly trained transformer with N parameters underperforms and third term takes account for number of limited optimisation steps.

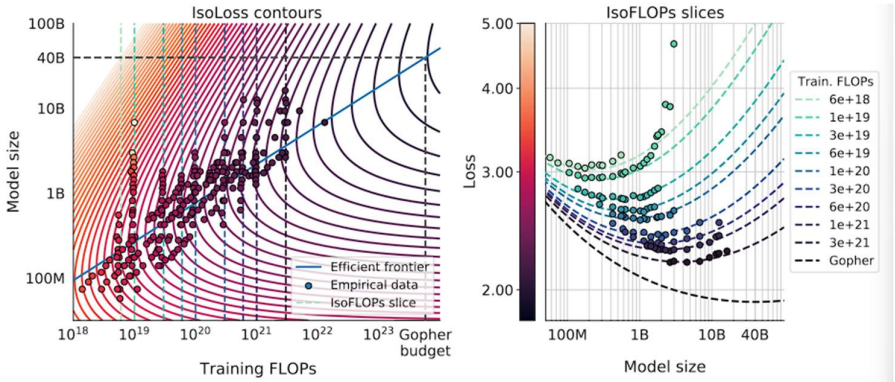


Figure 4 | **Parametric fit.** We fit a parametric modelling of the loss $\hat{L}(N, D)$ and display contour (**left**) and isoFLOP slices (**right**). For each isoFLOP slice, we include a corresponding dashed line in the left plot. In the left plot, we show the efficient frontier in blue, which is a line in log-log space. Specifically, the curve goes through each iso-loss contour at the point with the fewest FLOPs. We project the optimal model size given the *Gopher* FLOP budget to be 40B parameters.

From this approach, they found that $a = 0.46$ and $b = 0.54$.

4.4 CHINCHILLA

Due to these optimisation approaches Chinchilla LLM is trained with 70B parameters and 1.4 trillion tokens.

Model	Size (# Parameters)	Training Tokens
LaMDA (Thoppilan et al., 2022)	137 Billion	168 Billion
GPT-3 (Brown et al., 2020)	175 Billion	300 Billion
Jurassic (Lieber et al., 2021)	178 Billion	300 Billion
Gopher (Rae et al., 2021)	280 Billion	300 Billion
MT-NLG 530B (Smith et al., 2022)	530 Billion	270 Billion
Chinchilla	70 Billion	1.4 Trillion

BERT: PRE-TRAINING OF DEEP BIDIRECTIONAL TRANSFORMERS FOR LANGUAGE UNDERSTANDING

5.1 ABSTRACT

- BERT stands for Bidirectional Encoder Representations from Transformers.
- designed to pretrain deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers.
- Fine tune by only training only layer to create state of the art models
- Obtain state of the art results on GLUE score

5.2 INTRODUCTION

- GPT OpenAI trained by fine tuning all parameters.
- ElMo uses task specific architectures.
- These techniques restrict the power of pre trained parameters.
- It is because standard models are unidirectional.
- BERT uses Masked Language Model pre training objective
- This randomly masks some tokens in input and output is tasked to predict those tokens and complete the text or sentence

5.3 RELATED WORK

- Unsupervised Feature Based Approaches – Word and sentence embeddings are created as vectors. Vectors are ranked based on context-sensitive features and meaning.
- Unsupervised Fine-Tuning Approaches – Apart from word embedding, now sentence and document embeddings have been created and fine-tuned on downstream tasks.
- Transfer Learning from Supervised Data – Some works show effective results when trained on large datasets on specific tasks.

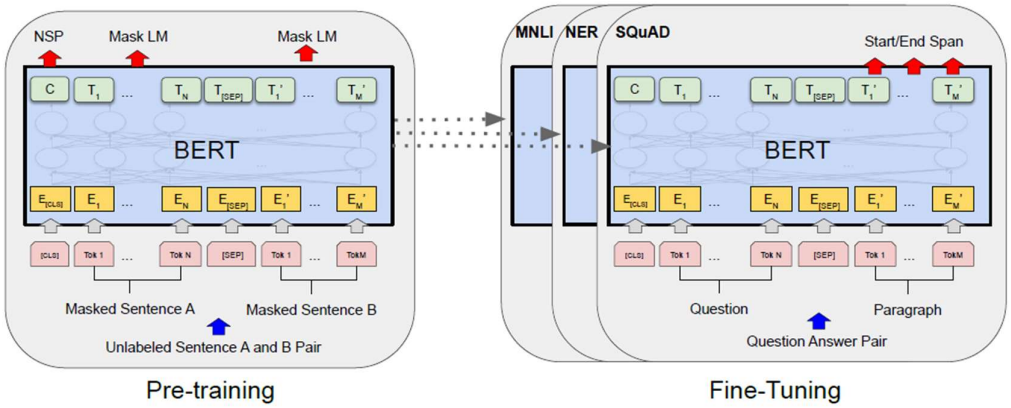


Figure 1: Overall pre-training and fine-tuning procedures for BERT. Apart from output layers, the same architectures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initialize models for different down-stream tasks. During fine-tuning, all parameters are fine-tuned. [CLS] is a special symbol added in front of every input example, and [SEP] is a special separator token (e.g. separating questions/answers).

5.4 BERT

- There are two steps in our framework: pre-training and fine-tuning. During pre-training, the model is trained on unlabeled data over different pre-training tasks. For finetuning, the BERT model is first initialized with the pre-trained parameters, and all of the parameters are fine-tuned using labeled data from the downstream tasks. Each downstream task has separate fine-tuned models, even though they are initialized with the same pre-trained parameters.
- **Model Architecture** - multi-layer bidirectional Transformer encoder based on the original implementation. BERTBASE was chosen to have the same model size as OpenAI GPT for comparison purposes.
- **Input/Output Representations** – Input can represent a single sentence or a pair of sentences in a single token sequence. WordPiece embeddings are used. They separated sentences with [SEP] token and then separate it into tokens using tokenizer models.

5.5 PRE-TRAINING BERT

- **Masked LM task** – Mask some random percent of tokens and predict outputs on those. the final hidden vectors corresponding to the mask tokens are fed into an output softmax over the vocabulary, as in a standard LM. Here, they mask 15% of the tokens. they only replace 80% of the tokens to be replaced with the [MASK] tokens.

- **Next Sentence Prediction** – Many tasks are based on understanding relation between two sentences. Model is pre trained on monolingual corpus. 50% of the time next sentence is actually next and rest if the time it is not.
- **Pre-training data** - The pre-training procedure largely follows the existing literature on language model pre-training. For the pre-training corpus they used the BooksCorpus (800M words) and English Wikipedia (2,500M words).

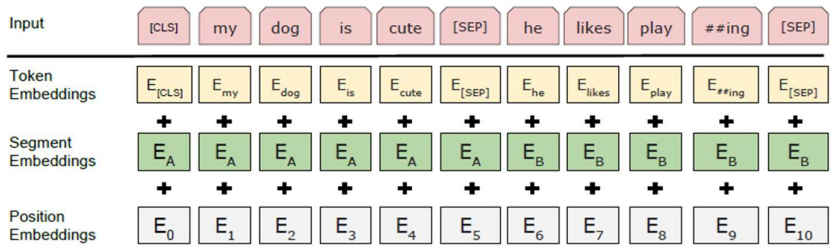


Figure 2: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings.

5.6 FINE-TUNING BERT

- Fine-tuning is straightforward since the self-attention mechanism in the Transformer allows BERT to model many downstream tasks— whether they involve single text or text pairs—by swapping out the appropriate inputs and outputs.
- For each task, they simply plug in the task-specific inputs and outputs into BERT and finetune all the parameters end-to-end.

5.7 EXPERIMENTS

- **GLUE** - The General Language Understanding Evaluation (GLUE) benchmark is a collection of diverse natural language understanding tasks. They fine tune out BERT base and large on all the tasks in GLUE dataset and get state of the art

results. BERT outperforms all other models

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	Average -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$BERT_{BASE}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$BERT_{LARGE}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

Table 1: GLUE Test results, scored by the evaluation server (https://gluebenchmark.com/leaderboard). The number below each task denotes the number of training examples. The "Average" column is slightly different than the official GLUE score, since we exclude the problematic WNLI set.[8] BERT and OpenAI GPT are single-model, single task. F1 scores are reported for QQP and MRPC, Spearman correlations are reported for STS-B, and accuracy scores are reported for the other tasks. We exclude entries that use BERT as one of their components.

- **SQuAD v1.1** - The Stanford Question Answering Dataset (SQuAD v1.1) is a collection of 100k crowdsourced question/answer pairs. Given a question and a text paragraph from Wikipedia, the task is to give an answer to the question.

System	Dev EM	Dev F1	Test EM	Test F1
Top Leaderboard Systems (Dec 10th, 2018)				
Human	-	-	82.3	91.2
#1 Ensemble - nlnet	-	-	86.0	91.7
#2 Ensemble - QANet	-	-	84.5	90.5
Published				
BiDAF+ELMo (Single)	-	85.6	-	85.8
R.M. Reader (Ensemble)	81.2	87.9	82.3	88.5
Ours				
$BERT_{BASE}$ (Single)	80.8	88.5	-	-
$BERT_{LARGE}$ (Single)	84.1	90.9	-	-
$BERT_{LARGE}$ (Ensemble)	85.8	91.8	-	-
$BERT_{LARGE}$ (Sgl.+TriviaQA)	**84.2**	**91.1**	**85.1**	**91.8**
$BERT_{LARGE}$ (Ens.+TriviaQA)	**86.2**	**92.2**	**87.4**	**93.2**

Table 2: SQuAD 1.1 results. The BERT ensemble is 7x systems which use different pre-training checkpoints and fine-tuning seeds.

- **SQuAD v 2.0** - The SQuAD 2.0 task extends the SQuAD 1.1 problem definition by allowing for the possibility that no short answer exists in the provided paragraph, making the problem more realistic. They use a simple approach to extend the SQuAD v1.1 BERT model for this task. They treat questions that do not have an answer as having an answer span with start and end at the [CLS] token.
- The probability space for the start and end answer span positions is extended to include the position of the [CLS] token.

System	Dev EM	Dev F1	Test EM	Test F1
Top Leaderboard Systems (Dec 10th, 2018)				
Human	86.3	89.0	86.9	89.5
#1 Single - MIR-MRC (F-Net)	-	-	74.8	78.0
#2 Single - nlnet	-	-	74.2	77.1
Published				
unet (Ensemble)	-	-	71.4	74.9
SLQA+ (Single)	-		71.4	74.4
Ours				
$BERT_{LARGE}$ (Single)	78.7	81.9	80.0	83.1

Table 3: SQuAD 2.0 results. We exclude entries that use BERT as one of their components.

- **SWAG** - The Situations With Adversarial Generations (SWAG) dataset contains 113k sentence-pair completion examples that evaluate grounded commonsense inference (Zellers et al., 2018). Given a sentence, the task is to choose the most plausible continuation among four choices.

System	Dev	Test
ESIM+GloVe	51.9	52.7
ESIM+ELMo	59.1	59.2
OpenAI GPT	-	78.0
$\text{BERT}_{\text{BASE}}$	81.6	-
$\text{BERT}_{\text{LARGE}}$	**86.6**	**86.3**
Human (expert)†	-	85.0
Human (5 annotations)†	-	88.0

Table 4: SWAG Dev and Test accuracies. †Human performance is measured with 100 samples, as reported in the SWAG paper.

5.8 EFFECT OF PRE-TRAINING TASKS

two models were developed: No NSP and LTR&No NSP. No NSP is a bidirectional model that is trained using the "masked LM" technique, but without the "next sentence prediction" task. LTR&No NSP, on the other hand, is a left-context-only model that is trained using a standard Left-to-Right (LTR) LM. This model was pre-trained without the NSP task and is directly comparable to OpenAI GPT, but with a larger training dataset, a unique input representation, and a fine-tuning scheme. The left-only constraint was also applied at fine-tuning to avoid a pre-train/fine-tune mismatch that could degrade downstream performance.

5.9 EFFECT OF MODEL SIZE

They examined the impact of model size on task accuracy by training various BERT models with different layers, hidden units, and attention heads. The results, as shown in Table 6, indicate that larger models lead to better accuracy on all four datasets, including the MRPC dataset with only 3,600 labeled training examples. They achieved significant improvements even on top of models that are already large relative to existing literature. This finding demonstrates that scaling to extreme model sizes can lead to improvements on very small scale tasks, as long as the model has been sufficiently pre-trained. Prior works have shown mixed results on increasing model size, but they hypothesize that fine-tuning directly on downstream tasks with a small number of initialized parameters can benefit from larger pre-trained representations

Hyperparams				Dev Set Accuracy		
#L	#H	#A	LM (ppl)	MNLI-m	MRPC	SST-2
3	768	12	5.84	77.9	79.8	88.4
6	768	3	5.24	80.6	82.2	90.7
6	768	12	4.68	81.9	84.8	91.3
12	768	12	3.99	84.4	86.7	92.9
12	1024	16	3.54	85.7	86.9	93.3
24	1024	16	3.23	86.6	87.8	93.7

Table 6: Ablation over BERT model size. #L = the number of layers; #H = hidden size; #A = number of attention heads. "LM (ppl)" is the masked LM perplexity of held-out training data.

5.10 MASKED LM AND MASKING PROCEDURE

Masked LM and the Masking Procedure Assuming the unlabeled sentence is `my dog is hairy`, and during the random masking procedure we chose the 4-th token (which corresponding to `hairy`), our masking procedure can be further illustrated by

- 80% of the time: Replace the word with the `[MASK]` token, e.g., `my dog is hairy` → `my dog is [MASK]`
- 10% of the time: Replace the word with a random word, e.g., `my dog is hairy` → `my dog is apple`
- 10% of the time: Keep the word unchanged, e.g., `my dog is hairy` → `my dog is hairy`. The purpose of this is to bias the representation towards the actual observed word.

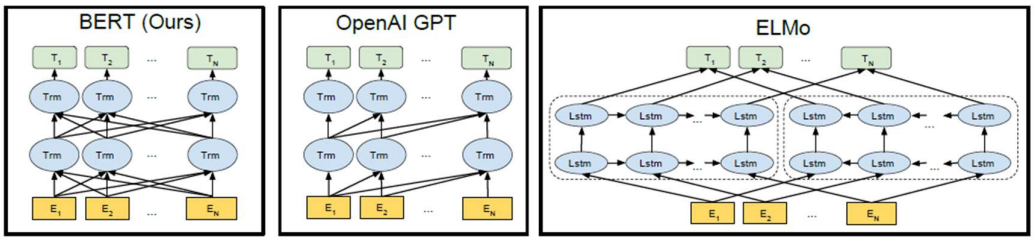
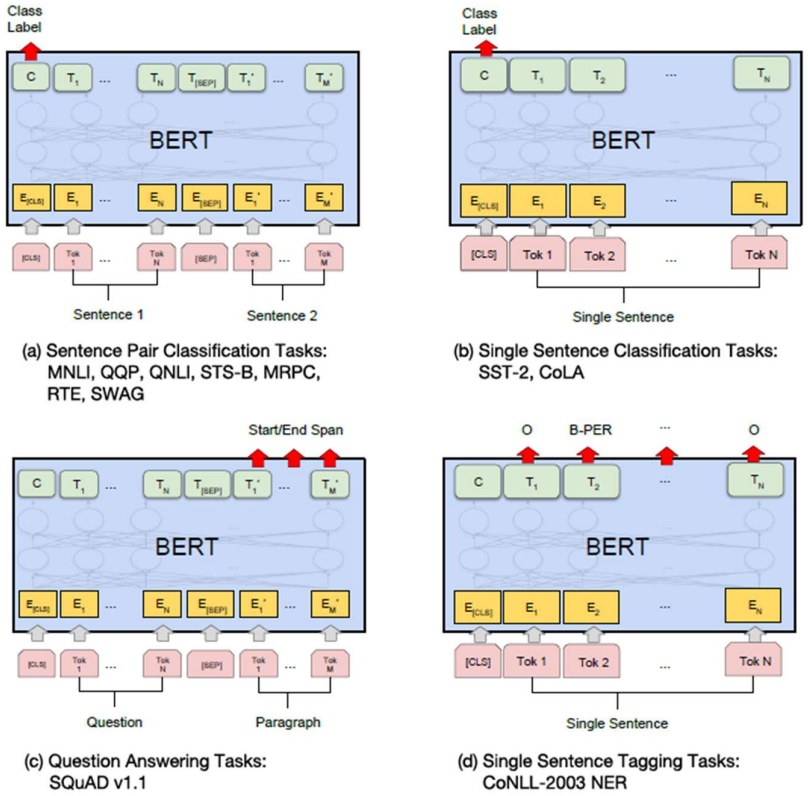


Figure 3: Differences in pre-training model architectures. BERT uses a bidirectional Transformer. OpenAI GPT uses a left-to-right Transformer. ELMo uses the concatenation of independently trained left-to-right and right-to-left LSTMs to generate features for downstream tasks. Among the three, only BERT representations are jointly conditioned on both left and right context in all layers. In addition to the architecture differences, BERT and OpenAI GPT are fine-tuning approaches, while ELMo is a feature-based approach.



(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

(b) Single Sentence Classification Tasks: SST-2, CoLA

(c) Question Answering Tasks: SQuAD v1.1

(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

Figure 4: Illustrations of Fine-tuning BERT on Different Tasks.

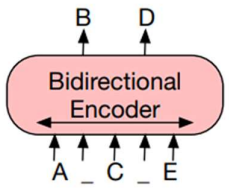
BART: DENOISING SEQUENCE-TO-SEQUENCE PRE-TRAINING FOR NATURAL LANGUAGE GENERATION, TRANSLATION, AND COMPREHENSION

6.1 INTRODUCTION

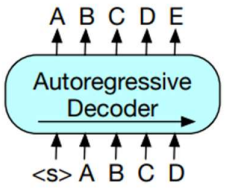
BART is a denoising autoencoder for pretraining sequence-to-sequence models. BART is trained by:-

1. Corrupting text with some random noise
2. Learning how to reconstruct the original text.

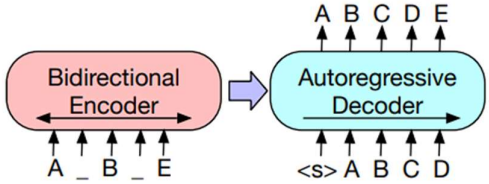
Which attempts to generalize BERT (due to the bidirectional encoder) and GPT (left-to-right decoder). BART includes noise flexibility where we can the length of the sentence too which would lead model to reason more about overall sentence.



(a) BERT: Random tokens are replaced with masks, and the document is encoded bidirectionally. Missing tokens are predicted independently, so BERT cannot easily be used for generation.



(b) GPT: Tokens are predicted auto-regressively, meaning GPT can be used for generation. However words can only condition on leftward context, so it cannot learn bidirectional interactions.



(c) BART: Inputs to the encoder need not be aligned with decoder outputs, allowing arbitary noise transformations. Here, a document has been corrupted by replacing spans of text with mask symbols. The corrupted document (left) is encoded with a bidirectional model, and then the likelihood of the original document (right) is calculated with an autoregressive decoder. For fine-tuning, an uncorrupted document is input to both the encoder and decoder, and we use representations from the final hidden state of the decoder.

6.2 MODEL ARCHITECTURE, PRE-TRAINING AND FINE TUNING

6.2.1 ARCHITECTURE

This model uses sequence to sequence Transformer architecture with 6 layers of Encoder and Decoder where each layer of decoder performs cross-attention with final hidden layer of the encoder and has an additional feedforward network before prediction thus having 10% more parameters than BERT.

6.2.2 PRE-TRAINING

BART is trained through corrupting documents and optimised through reconstruction loss of original document. BART allows any corruption technique and for empty input, it can act as language model. Some transformations used are:-

6.2.2.1 TOKEN MASKING

Random tokens are sampled and masked with masked elements.

6.2.2.2 TOKEN DELETION

Random tokens deleted from Input such that we don't know their positions

6.2.2.3 TEXT INFILLING

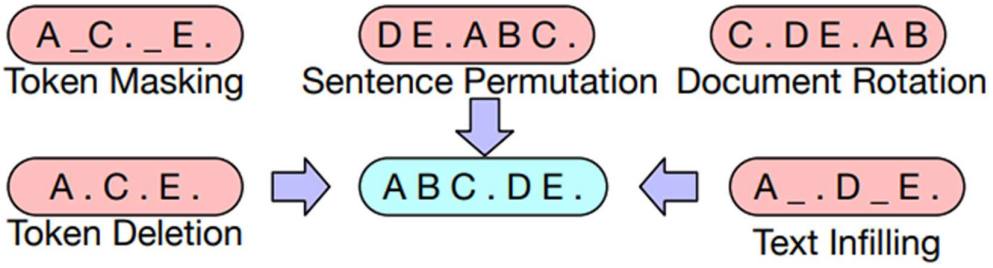
Number of text spans are sampled using Poisson distribution and replaces the span with sequence of masked elements of same length which teaches the model to predict how many tokens are missing from a span.

6.2.2.4 SENTENCE PERMUTATION

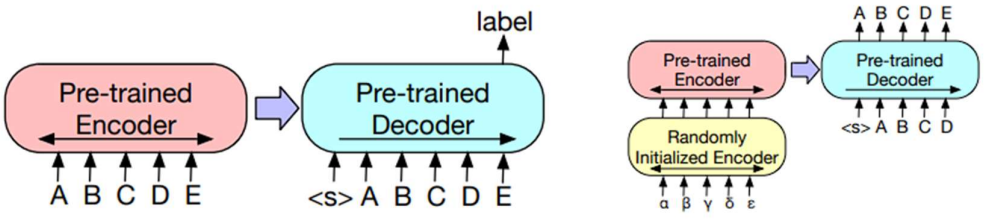
Sentences in a document are shuffles in a random order.

6.2.2.5 DOCUMENT ROTATION

Sentence chosen at random and document rotated such that document starts from the selected sentence.



6.2.3 FINE TUNING



(a) To use BART for classification problems, the same input is fed into the encoder and decoder, and the representation from the final output is used.

(b) For machine translation, we learn a small additional encoder that replaces the word embeddings in BART. The new encoder can use a disjoint vocabulary.

6.2.3.1 SEQUENCE CLASSIFICATION TASKS

Both encoder and decoder are fed same inputs and final hidden state of the final decoder is fed into a multi-class classifier.

6.2.3.2 TOKN CLASSIFICATION TASKS

Both encoder and decoder are fed the complete document and the top hidden state of the decoder is used as the representation for each word which is then used to classify the token.

6.2.3.3 SEQUENCE GENERATION TASKS

As BART uses autoregressive decoder, we can use it for question answering and summarisation where encoder inputs the information and decoder generates output autoregressively.

6.2.3.4 MACHINE TRANSLATION

They replace the encoder with a randomly initialised encoder that is trained to map foreign words into an input that BART can denoise in English. This training process takes two steps:-

1. Freeze most of BART parameters and only update randomly initialized encoder, positional embeddings and self-attention input of encoder's first layer.
2. Train all parameters for a small number of iterations.

6.3 RESULTS

BART, a pre-training approach that learns to map corrupted documents to the original. BART achieves similar performance to RoBERTa on discriminative tasks, while achieving new state-of-the art results on a number of text generation tasks. Future work should explore new methods for corrupting documents for pre-training, perhaps tailoring them to specific end tasks.

CONVOLUTIONAL XFORMERS FOR VISION

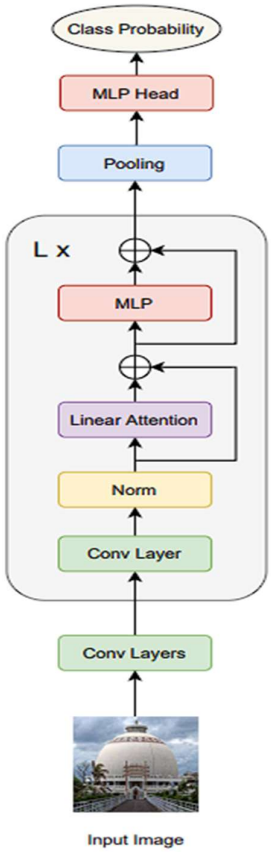
7.1 INTRODUCTION

Transformers have been state of the art mechanisms in NLP tasks but their implications in computer vision have been very limited due to the complexity self-attention mechanism and the extraordinary length of input sequence of computer vision problems. This becomes an issue for problems with less data and limit compute resources.

CNNs are very successful in pooling the information and they learn form smaller datasets and use less compute resources, but do not have ability to capture long-range dependencies like transformers.

Convolutional Xformers for Vision(CXV) are introduced with use convolutions and attention for image classification which uses linear attention mechanisms(discussed later in detail in the project) and convolutions to achieve best of both worlds.

7.2 MODEL ARCHITECTURE



7.2.1 LINEAR ATTENTION

Using Linear Attention mechanisms increase performance for low data image classification and decrease compute requirement and handle longer input image sequences.

7.2.2 CONVOLUTIONAL EMBEDDING

Convolutional layers are used to generate pixel embeddings as they provide the inductive prior which reduces the need for more training data. Also, for high resolution images, we can adjust convolution strides and kernel size to maintain GPU usage.

7.2.3 CONVOLUTIONAL SUB-LAYER

The convolution sub-layer is used to provide spatial information of the input to the linear attention and keeps the 2-D image shape till linear attention, also eliminates the requirement of positional embedding.

7.2.4 LAYER NORMALISATION

To reduce the variance of the inputs for the next sub-layer, a single layer normalisation is placed between convolutional and self-attention block.

7.3 RESULTS

The above discussed model was tested on CIFAR-10 dataset and compared with other recently developed architectures.

Model	# Params	GPU (GB)	MACs	CIFAR-10
Convolutional Models				
ResNet-18	11.2 M	0.6	0.56 G	86.29
ResNet-34	21.3 M	0.7	1.16 G	87.97
Mixing Models				
FNet-10	1.3 M	2.3	1.34 G	51.05
MLP Mixer-5	21.3 M	1.5	3.02 G	60.26
WaveMix-5	1.4 M	0.4	2.59 G	83.71
ConvMixer-16	1.3 M	3.5	1.37 G	88.46
Transformer Models				
ViT-10/4	1.3 M	14.7	1.34 G	57.53
Hybrid ViN-6/8	1.3 M	5.3	1.41 G	77.96
Hybrid ViP-6/8	1.3 M	5.9	1.31 G	77.54
Hybrid ViLT-6/8	1.3 M	2.6	0.33 G	78.34
CCT-6/4	1.3 M	13.6	1.32 G	82.66
CvT-5/4	1.3 M	9.4	1.32 G	77.90
Convolutional Xformers for Vision				
CNV-5/4	1.3 M	3.1	1.39 G	89.56
CLTV-5/4	1.3 M	1.8	1.38 G	86.99
CPV-5/4	1.3 M	3.2	1.37 G	**91.42**

A SURVEY ON EFFICIENT TRANSFORMERS

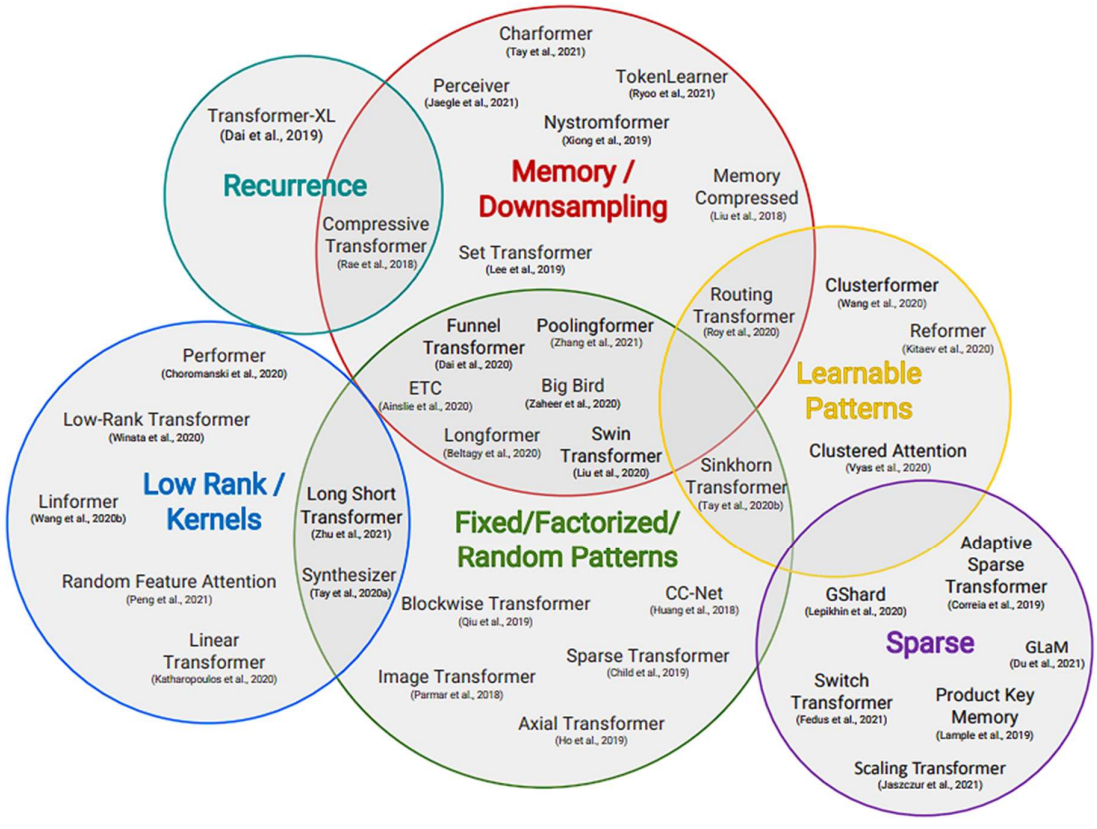
8.1 INTRODUCTION

Transformers are recent advancements in deep learning and they have been a great force in the field. There have been recent advancements on in Transformer models and it is challenging for researchers and practitioners to keep pace with innovation.

The self-attention mechanism is a key characteristic of Transformer models but the key concern with this mechanism is its quadratic time and memory complexity and many efficient Transformers have been proposed recently to address the issue.

The efficiency of a model can be interpreted as the memory footprint of the model and also direct referred to the computational costs, i.e. no. of FLOPs during training and inference. The efficiency of the transformers are necessary in circumstances with long input sequences such as documents, images and videos.

8.2 TAXONOMY OF EFFIECIENT TRANSFORMERS



The primary goal is to improve memory complexity of the self-attention mechanism, but some also improve the general efficiency of the Transformer architecture.

8.2.1 FIXED PATTERNS

This technique sparsifies the attention matrix by limiting the field of view to fixed, predefined patterns.

8.2.1.1 BLOCKWISE PATTERNS

This considers blocks of receptive fields by chinking input into fixed blocks which reduces the time complexity from N^2 to B^2 (block size) with $B<<<N$. Various complex models are based on these blocking methods.

8.2.1.2 STRIDED PATTERNS

This approach considers strided patterns and only attending at fixed intervals.

8.2.1.3 COMPRESSED PATTERNS

This uses some pooling operators to down-sample the sequence length into a form of fixed pattern and use strided convolution to reduce the sequence length.

8.2.2 COMBINATION OF PATTERNS

They improve coverage by combining two or more distinct access patterns and reduces the memory complexity like fixed pattern. The major distinguishing factor is that aggregation and combination of multiple patterns improve overall coverage of self-attention.

8.2.3 LEARNABLE PATTERNS

This approach aims to learn the access patterns in a data-driven fashion and to determine a notion of token relevance and then assign tokens to buckets or clusters and uses similarity-based functions and trained end-to-end jointly with rest of the network.

8.2.4 NEURAL MEMORY

This method is to leverage a learnable side memory module that can access multiple tokens at once. A common form is global neural memory which is able to access multiple tokens at once and a common global neural memory which is able to access the entire sequence that performs a pooling-like operation of input sequence to compress the input sequence.

8.2.5 LOW-RANK METHODS

This method improves efficiency by leveraging low-rank approximation of self-attention matrix as $N \times N$ matrix is now reduced to $N \times k$ matrix.

8.2.6 KERNELS

The use of kernels enable mathematical re-writing of self-attention mechanism to avoid the $N \times N$ matrix and also can be viewed as low-rank approach.

8.2.7 DOWNSAMPLING

This method they reduce resolution of the sequence hence reducing the computation costs by commensurate factor by memory tokens or kernel based approaches or strided based pooling.

8.2.8 SPARSE MODELS AND COMPUTATIONAL COMPUTATION

These approaches do not optimise attention mechanism itself but sparse models sparsely activate a subset of the parameters which generally improve the parameters to FLOPs ratio.

8.3 SOME EFFICIENT TRANSFORMER MODELS

8.3.1 MEMORY COMPRESSED TRANSFORMER

This approach uses following modification approaches:

- **Local Attention Span**
  This approach reduces the attention span of long sequences to a local neighbourhood and divide input into blocks of similar length so that self-attention can be calculated with each block independently keeping the cost of activation constant.
- **Memory-compressed Attention**
  This approach reduces the number of key and value pairs using a strided convolution while queries remain unchanged that leads to reduction to reduction in size of attention matrix as well as computations based on kernel size and strides of the convolution and lets the model exchange the information globally.

For a block size of b, the computational and memory cost of self-attention in each block is $O(b^2)$. Given there are n/b blocks, the computational and memory cost of local attention is $O(b.n)$. For memory-compressed attention, applying a convolution with kernel size and strides of k, the computational and memory cost of the attention mechanism reduces to $O(n \cdot n/k)$.

8.3.2 SPARSE TRANSFORMER

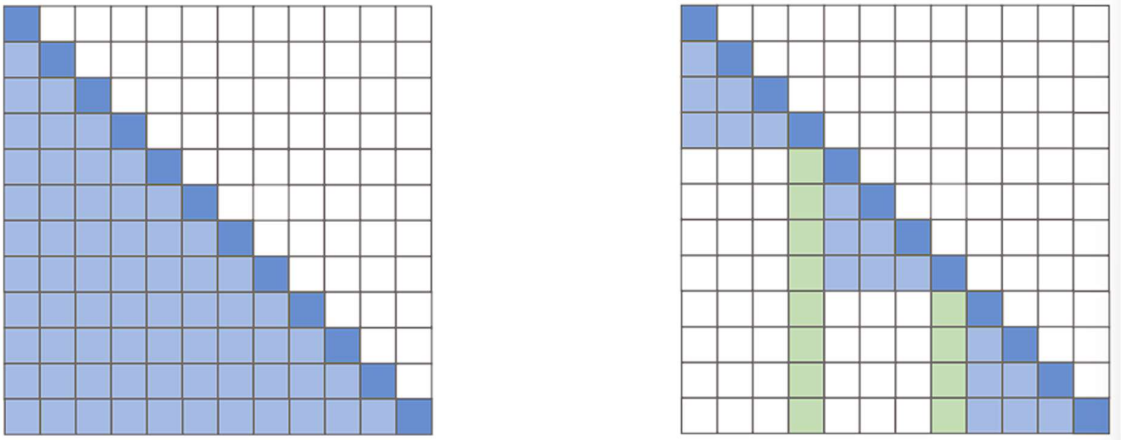
This approach reduces the dense attention matrix to a sparse version by computing attention matrix on a sparse number of q_i, k_j pairs and employs fixed attention patterns which are defined by strides and local neighbourhoods where computation is factorised, where:

- **Local Attention Heads**

$$\hat{A}_{ij} = \begin{cases} Q_i(K)_j^\top), & \text{if } \lfloor j/N \rfloor = \lfloor i/N \rfloor \\ 0 & \text{otherwise} \end{cases}$$

- **Strided Attention Heads**

$$\hat{A}_{ij} = \begin{cases} Q_i(K)_j^\top), & \text{if } (i-j) \mod N = 0 \\ 0 & \text{otherwise} \end{cases}$$



(a) Transformer (b) Sparse Transformer

Figure 4: Illustration of patterns of the attention matrix for dense self-attention in Transformers and sparse fixed attention in Sparse Transformers. Blue in the right diagram represents the local self-attention while green represents the strided component of the sparse attention.

The modification in the self-attention mechanism does not alter the parameter costs of the model since the model still retains the *Q, K, V* transforms from the original Transformer model. The memory complexity of the attention layer is reduced from $O(n^2)$ to $O(n \log n)$.

TRANFORMERS WITH LINEAR ATTENTION

'TRANSFORMERS ARE RNNs'

9.1 INTRODUCTION

Transformer models are a recent breakthrough in natural language understanding and have performed very impressively with pre-training with autoregressive and modelling objectives. Transformers naturally come up with high performance computation needs mainly caused by its self-attention mechanism which costs $O(N^2)$ memory and time complexity.

Various optimisation methods have been approached to tackle the situation and pre-training complexity has been reduced to $O(N \log N)$, but autoregressive inference is not set up.

Linear Transformers have been designed to be as performant as a Transformer but reduce the inference time by three orders of magnitude by scaling linearly with respect to the context length.

9.2 QUADRATIC ATTENTION V/S LINEAR ATTENTION

9.2.1 QUADRATIC ATTENTION

Transformer made of L Transformer Layers $T_1(.), \ldots, T_L(.)$:

$$T_l(x) = f_l(A_l(x) + x)$$

Where function $f_l(.)$ transformers each feature independently and implemented with a two-layer feedforward network. $A_l(.)$ is the self-attention mechanism, calculated as,

$$\begin{aligned} Q &= xW_Q, \\ K &= xW_K, \\ V &= xW_V, \\ A_l(x) = V' &= \text{softmax}\left(\frac{QK^T}{\sqrt{D}}\right) V. \end{aligned}$$

Generalizing,

$$V_i' = \frac{\sum_{j=1}^{N} \text{sim}\,(Q_i, K_j)\, V_j}{\sum_{j=1}^{N} \text{sim}\,(Q_i, K_j)}.$$

9.2.2 LINEAR ATTENTION

The above equation, the similarity function can be any polynomial or kernel attention, just the constraint is that its domain should be in the positive Real number, then the above was restudied as :-

$$V_i' = \frac{\sum_{j=1}^{N} \phi(Q_i)^T \phi(K_j) V_j}{\sum_{j=1}^{N} \phi(Q_i)^T \phi(K_j)},$$

By further simplifying by the associative property of matrix multiplication to :-

$$V_i' = \frac{\phi(Q_i)^T \sum_{j=1}^{N} \phi(K_j) V_j^T}{\phi(Q_i)^T \sum_{j=1}^{N} \phi(K_j)}.$$

Where feature map $\Phi(.)$ is applied row-wise to the matrices Q and K. In the above equation, we can calculate $\sum_{j=1}^{N} \phi(K_j) V_j^T$ and $\sum_{j=1}^{N} \phi(K_j)$ once and then reuse it for the rest of the queries that makes this mechanism linear.

9.2.3 FEATURE MAPS AND COMPUTATIONAL COST

Due to softmax attention, the cost in terms of addition and multiplication is in order of $O(N^2 \max(D,M))$, where D is dimensionality of Q and K, and M is dimension of V. But in linear transformer, feature maps of dimensionality C is first calculated thus making cost $O(N C M)$.

9.2.4 CAUSAL MASKING

Transformers are efficiently trained by masking the attention computation such that i–th position can only be influenced by position j if $j \leq i$, making linear attention as,

$$V_i' = \frac{\phi(Q_i)^T \sum_{j=1}^{i} \phi(K_j) V_j^T}{\phi(Q_i)^T \sum_{j=1}^{i} \phi(K_j)}.$$

And intruding S_i, Z_i

$$S_i = \sum_{j=1}^{i} \phi(K_j) V_j^T,$$

$$Z_i = \sum_{j=1}^{i} \phi(K_j),$$

They make,

$$V_i' = \frac{\phi(Q_i)^T S_i}{\phi(Q_i)^T Z_i}.$$

9.3 TRAINING AND INFERENCE

Due to availability of full ground truth sequence, parallelization of the training is possible. Also, during inference output for timestep *i* is input for timestep *i+1* which makes parallelization impossible. But with the linear transformer, both training are parallelizable as $\phi(K_j)\,V_j^T$ matrix can stored for each next step which makes it thousands more faster than older transformers.

9.4 TRANSFROMERS ARE RNNs

With above discussion and derivation of Causal Masking equation, it can be showed that-

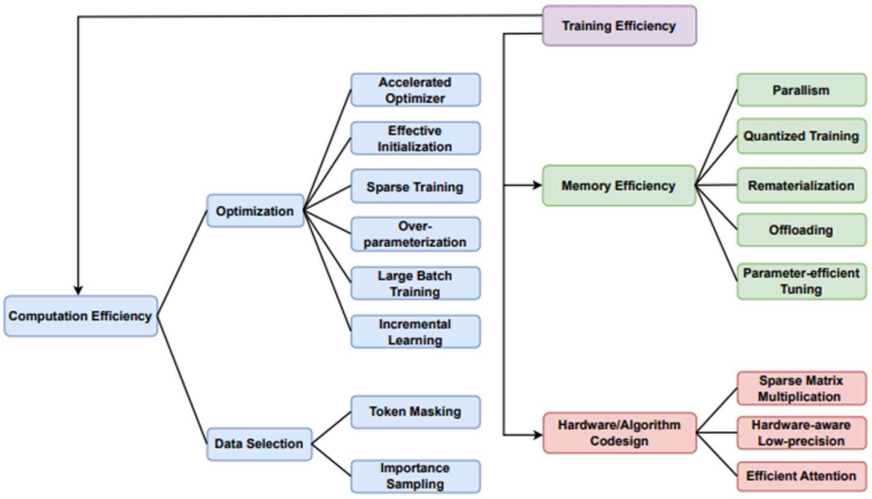
$$
\begin{aligned}
s_0 &= 0, \\
z_0 &= 0, \\
s_i &= s_{i-1} + \phi\left(x_i W_K\right)\left(x_i W_V\right)^T, \\
z_i &= z_{i-1} + \phi\left(x_i W_K\right), \\
y_i &= f_l\left(\frac{\phi\left(x_i W_Q\right)^T s_i}{\phi\left(x_i W_Q\right)^T z_i} + x_i\right).
\end{aligned}
$$

Where the RNN has two hidden states, attention memory as *s* and normalizer memory as *z*.

A SURVEY ON EFFICIENT TRAINING OF TRANSFORMERS

10.1 INTRODUCTION

- The fourth industrial revolution is being driven by deep learning, a modern and sophisticated method that has revolutionized machine learning (ML) and artificial intelligence.
- According to a widely discussed energy study of deep learning models, training a Transformer base model with neural architecture search (NAS) results in about 626,155 pounds of climate-warming carbon dioxide, or the lifetime emissions of five cars; as models get bigger, their computing requirements are outpacing advancements in hardware efficiency.
- From 345M with BERT-large in 2018, to hundreds of billions until with versions like MT-NLG equipped with 530B characteristics.
- In order to reduce the memory footprint of storing intermediate tensors and assuring high processing elements (PE) utilization across accelerators, these SOTA large models ask for memory-efficient training strategies.



10.2 OPTIMIZATION

- Greater training effectiveness falls essentially into three categories:
  (i) Measure the impact of connections on the loss at initialization to identify sparse networks, doing away with the complex iterative optimization schedule.
  (ii) Use inexpensive approaches to identify the Transformers winning tickets at a very early training stage and then simply train these early tickets till convergence.
  (iii) Use an alternate pruning and growing schedule, appropriate for broad architectures, to dynamically update model sparsity patterns throughout training.

- A major finding is that, in line with the well-known training dynamics that emphasize low-frequency structure in the early stages and high-frequency semantics in the later stages, steadily increasing the input image resolution can greatly speed up ViT training.

10.3 DATA SELECTION

- Data efficiency, in addition to model efficiency, is a key component of effective training.
- Skipping processing the masked tokens results in a significant increase in training efficiency for MLM and MIM because shortening the sequence length reduces both the computational and memory complexity quadratically.
- In order to reduce memory and processing costs, it is suggested for MLM to pre-train the encoder and decoder together for language generating jobs.
- For MIM, typical study indicates that eliminating the masked picture patches before to the encoder results in higher performance and up to three times less overall pre-training time and memory usage.
- Simplified average gradient norms or error 2-norms over many weight initializations can be used to identify significant examples at the very early stage of training, speeding up the sampling process comparable to the early-bird LTH in the data domain.
- Based on data importance sampling, it shows a viable path towards more effective neural scaling rules.

10.4 METHOD

- A significant training bottleneck is the expanding model size of huge Transformer models, which do not fit into the memory of a single device. Examples include the BERT 345M parameter model and the GPT-3 with 1.75 trillion parameters.
- Megatron-LM, which is tailored to train Transformer-based models, slices both MSA and FFN across GPUs and only needs a few more All-Reduce operations in the forward and backward pass. This enables them to train models with up to 8.3 billion parameters utilizing 512 GPUs. Regarding PP, it was first suggested in GPipe, which divides the input mini-batch into numerous smaller micro-batches, allowing various accelerators to work on various micro-batches concurrently, before performing a single synchronous gradient update for the mini-batch as a whole.
- Faster training and inference for DNNs are made possible by the invention of effective hardware accelerators, which alleviates the pressure on compute and memory. The high level of parallelism in graphics processing units (GPUs) makes them more capable of performing matrix multiplication than central processing units (CPUs).

- FlashAttention makes the suggestion that tiling be used to decrease I/O traffic between on-chip SRAM and GPU high bandwidth memory (HBM), which is evolving into a standard fast and memory-efficient attention module for speedup.

10.5 CONCLUSION AND FUTURE RESEARCH

- The following significant elements have been examined in order to enhance the training of Transformers:
  1) Useful initialization and optimization paradigms that can hasten convergence with fewer training iterations, lowering computational costs.
  2) improved data efficiency by the use of informative training samples and laws of test error with respect to dataset size for neural scaling
  3) memory-efficient methods to satisfy the memory needs for training huge Transformers, which necessitates jointly optimizing accelerator PE use, memory and communication footprints employing parallelism, low-precision arithmetic, checkpointing and offloading approaches, etc.
  4) co-designing the hardware and algorithm to increase the hardware platforms' capacity for training.
- New techniques for effectively training an elastic supertnet that supports a variety of architectural configurations after a single-shot NAS or a combination of experts, for different jobs and financial constraints, are highly desirable:-
  (i) On-device training on the edge with constrained resources, to prevent routine data transmission that causes latency and privacy problems.
  (ii) Combining effective approximation methods like token/model pruning and low-rank. To minimize the model size and computing cost in a complementary sense, (iii) factorization, lightweight architectural design, dynamic neural networks, etc. (iv) A common benchmark to assess and contrast the effective training techniques.
- The availability of such a benchmark will hasten the adoption of such techniques and result in actual cost savings in the future.

CONCLUSIONS AND ACKNOWLEDGEMENT

In this brief review of Transformers and LLMs, we covered some fundamental research elements of Transformers such as how to concept of attention emerged in seq-to-seq models, how we took that approach to build the ground breaking Transformers, exploring state-of-the research from Google(BERT) , Facebook(BART), IIT Bombay(CXVs) and went in depth in some efficient Transformers with discussing in-depth Linear self-attention mechanism with a discussion on efficient training approaches.

In the case of LLMs we first discussed why OpenAI saw a need of emergence of pre-trained models and fine-tuning them as a better idea and then discussed how Google DeepMind displayed us that its not necessary that only increase in size of LLMs would increase the performance of LLM, but also increasing the pre-training data can also lead to better performance.

We would like to thank Dr. Amandeep Kaur to provide us with such a great opportunity to learn about Transformers and having a brief understanding of the latest buzzwords ‘LLMs’, ‘ChatGPT’ and ‘AI’.

REFERENCES

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. *ArXiv*. /abs/1409.0473

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention Is All You Need. *ArXiv*. /abs/1706.03762

Radford, A., & Narasimhan, K. (2018). Improving Language Understanding by Generative Pre-Training.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D. D., Hendricks, L. A., Welbl, J., Clark, A., Hennigan, T., Noland, E., Millican, K., Driessche, G. V., Damoc, B., Guy, A., Osindero, S., Simonyan, K., Elsen, E., . . . Sifre, L. (2022). Training Compute-Optimal Large Language Models. *ArXiv*. /abs/2203.15556

Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *ArXiv*. /abs/1810.04805

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., & Zettlemoyer, L. (2019). BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. *ArXiv*. /abs/1910.13461

Jeevan, P., & Sethi, A. (2022). Convolutional Xformers for Vision. *ArXiv*. /abs/2201.10271

Tay, Y., Dehghani, M., Bahri, D., & Metzler, D. (2020). Efficient Transformers: A Survey. *ArXiv*. /abs/2009.06732

Katharopoulos, A., Vyas, A., Pappas, N., & Fleuret, F. (2020). Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention. *ArXiv*. /abs/2006.16236

Zhuang, B., Liu, J., Pan, Z., He, H., Weng, Y., & Shen, C. (2023). A Survey on Efficient Training of Transformers. *ArXiv*. /abs/2302.01107