WRITE THE TITLE OF YOUR THESIS HERE WITHIN MAXIMUM 10-14 WORDS

Submitted By:

MD ABRAR JAHIN ROLL NO. 1811035

Submitted To:

SUPERVISOR’S NAME
DESIGNATION

A thesis report submitted to the Department of Industrial Engineering and Management, Khulna University of Engineering & Technology, Khulna in partial fulfillment of the requirements for the degree of ***“Bachelor of Science in Industrial & Production Engineering”***

February 2024

DECLARATION

This is to certify that the thesis work entitled **“Write the title of your thesis here within maximum 10-14 words”** has been carried out by *Md Abrar Jahin* in the Department of *Industrial Engineering and Management*, Khulna University of Engineering & Technology, Khulna, Bangladesh. The above thesis work or any part of this work has not been submitted anywhere for the award of any degree or diploma.

SUBMITTED BY

Md Abrar Jahin
Roll: 1811035

SUPERVISED BY

Md. Saiful Islam
Assistant Professor
Department of Industrial Engineering and Management
Khulna University of Engineering & Technology

EXAMINED BY (EXTERNAL)

External member’s name
Designation
Department of Industrial Engineering and Management
Khulna University of Engineering & Technology

COUNTERSIGNED

Name of the department head
Head
Department of Industrial Engineering and Management
Khulna University of Engineering & Technology

ACKNOWLEDGMENT

I am deeply grateful to my esteemed supervisor, **Md. Saiful Islam**, Assistant Professor in the Department of Industrial Engineering and Management at Khulna University of Engineering & Technology. His unwavering guidance, exceptional support, and constant encouragement have been the cornerstones of my journey through this thesis. His expertise, remarkable patience, and insightful feedback have played a pivotal role in shaping the trajectory of this work.

[Convey gratitude here to the others who contributed to your thesis directly or passively]

Acknowledgment is also due to the faculty and staff at the Department of Industrial Engineering and Management, Khulna University of Engineering & Technology, for fostering an environment that stimulates academic exploration. The resources and support provided by this department have been instrumental in the successful completion of this thesis.

I am indebted to my family and friends for their unwavering encouragement and understanding during this challenging yet immensely rewarding academic journey.

Finally, my heartfelt appreciation goes out to all the individuals who have contributed directly or indirectly to this thesis. Their assistance and cooperation have been invaluable, shaping the collaborative spirit that defines this work.

February 2024 ***Md Abrar Jahin***

ABSTRACT

[Write the abstract here within 250-300 words]

This template was initially created to facilitate drafting the B.Sc. Eng. thesis or project on LaTeX for the next generations of KUET was further modified by **Md Abrar Jahin** (1811035). Recognizing the importance of open collaboration and the need for an alternative to the traditional B.Sc. thesis template available only in MS Word format, this initiative was taken voluntarily by Md Abrar Jahin. The modifications made by Md Abrar Jahin is hereby released under an open-source license to conserve the principles of collaboration and knowledge sharing in the academic community.

Keywords: Write 5-6 keywords primarily representing your research that might not be present in your title or abstract necessarily

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

KUET	Khulna University of Engineering & Technology
IEM	Industrial Engineering and Management
SCM	Supply Chain Management

CHAPTER I

INTRODUCTION

1.1 Section 1

1.1.1 Sub-section 1

Sub-sub-section 1

In Chapter III, we discussed the methodology followed in this thesis.

CHAPTER II

LITERATURE REVIEW

This thesis followed the IEEE style citation format. If you wanna change it to Author-Date, Harvard, MLA, Chicago, or APA style, consider modifying the parameters ”style=author-date” or ”style=apa” in the ”package-settings.tex” file.

[1], [2]
Or,
[3]–[5]

CHAPTER III

METHODOLOGY

$$C = [C_j]'_{1\times n} = \left[\sum_{n}^{j-1} t_{ij}\right]'_{1\times n} ; j = 1, ..., n \tag{3.1}$$

This is how an Algorithm 1 can be formatted.

Algorithm 1 Training a Simple Feedforward Neural Network

```
1.  **Input:** Training data $\{(x_i, y_i)\}_{i=1}^{N}$, learning rate $\eta$, number of epochs $E$
2.  **Initialize:** Neural network $f$ with parameters $\theta$
3.  **for** epoch = 1 to $E$ **do**
4.      **for** each $(x_i, y_i)$ in training data **do**
5.          Compute prediction $\hat{y}_i = f(x_i; \theta)$
6.          Compute loss $\mathcal{L}(y_i, \hat{y}_i)$
7.          Compute gradients $\nabla_\theta \mathcal{L}$
8.          Update parameters: $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}$
9.      **end for**
10. **end for**
```

Figure 1 shows a cat.

Figure 2a and 2b also display the cats.

Figure 1: Figure caption

(a) First subfigure caption

(b) Second subfigure caption

Figure 2: Master figure caption.

CHAPTER IV

RESULTS

This is how to refer to the Table 1 and 1.

Table 1: Table Style One

Model	t-statistic (metric 1)	p-value (metric 1)	t-statistic (metric 2)	p-value (metric 2)
A	71.81148356	9.95E-14	17.63726588	2.74E-08
B	63.94133809	2.82E-13	17.72554207	2.63E-08
C	57.87545827	6.91E-13	73.09019106	8.49E-14
D	63.01465217	3.22E-13	14.14082681	1.88E-07
E	68.07699279	1.61E-13	14.08814707	1.94E-07
F	65.40333765	2.31E-13	34.80494872	6.59E-11
G	71.80695901	9.96E-14	24.47760546	1.52E-09

Table 1: Table Style Two

Model	t-statistic (metric 1)	p-value (metric 1)	t-statistic (metric 2)	p-value (metric 2)
A	71.81148356	9.95E-14	17.63726588	2.74E-08
B	63.94133809	2.82E-13	17.72554207	2.63E-08
C	57.87545827	6.91E-13	73.09019106	8.49E-14
D	63.01465217	3.22E-13	14.14082681	1.88E-07
E	68.07699279	1.61E-13	14.08814707	1.94E-07
F	65.40333765	2.31E-13	34.80494872	6.59E-11
G	71.80695901	9.96E-14	24.47760546	1.52E-09

CHAPTER V

DISCUSSIONS

CHAPTER VI

CONCLUSIONS, LIMITATIONS, AND FUTURE RESEARCH DIRECTIONS

REFERENCES

[1] M. Abdel-Basset, G. Manogaran, and M. Mohamed, “Internet of things (iot) and its impact on supply chain: A framework for building smart, secure and efficient systems,” *Future generation computer systems*, vol. 86, no. 9, pp. 614–628, 2018.

[2] Z. A. Abkenar, H. F. Lajimi, M. Hamedi, and S. V. Parkouhi, “Determining the importance of barriers to iot implementation using bayesian best-worst method,” in *Advances in Best-Worst Method: Proceedings of the Second International Workshop on Best-Worst Method (BWM2021)*, Springer, 2022, pp. 144–159.

[3] H. Afreen and I. S. Bajwa, “An iot-based real-time intelligent monitoring and notification system of cold storage,” *IEEE Access*, vol. 9, pp. 38 236–38 253, 2021.

[4] A. Aryal, Y. Liao, P. Nattuthurai, and B. Li, “The emerging big data analytics and iot in supply chain management: A systematic review,” *Supply Chain Management: An International Journal*, vol. 25, no. 2, pp. 141–156, 2020.

[5] M. Babagolzadeh, A. Shrestha, B. Abbasi, Y. Zhang, A. Woodhead, and A. Zhang, “Sustainable cold supply chain management under demand uncertainty and carbon tax regulation,” *Transportation Research Part D: Transport and Environment*, vol. 80, p. 102 245, 2020.

APPENDICES

Data availability

Write about the availability of the datasets you used in your thesis.

Code availability

Indicate and link the public repository of the codes you used in your thesis.

Attention Is All You Need

Ashish Vaswani[*]
Google Brain
avaswani@google.com

Noam Shazeer[*]
Google Brain
noam@google.com

Niki Parmar[*]
Google Research
nikip@google.com

Jakob Uszkoreit[*]
Google Research
usz@google.com

Llion Jones[*]
Google Research
llion@google.com

Aidan N. Gomez[*] [†]
University of Toronto
aidan@cs.toronto.edu

Łukasz Kaiser[*]
Google Brain
lukaszkaiser@google.com

Illia Polosukhin[*] [‡]
illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.0 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature.

1 Introduction

Recurrent neural networks, long short-term memory [12] and gated recurrent [7] neural networks in particular, have been firmly established as state of the art approaches in sequence modeling and transduction problems such as language modeling and machine translation [29, 2, 5]. Numerous efforts have since continued to push the boundaries of recurrent language models and encoder-decoder architectures [31, 21, 13].

[*]Equal contribution. Listing order is random. Jakob proposed replacing RNNs with self-attention and started the effort to evaluate this idea. Ashish, with Illia, designed and implemented the first Transformer models and has been crucially involved in every aspect of this work. Noam proposed scaled dot-product attention, multi-head attention and the parameter-free position representation and became the other person involved in nearly every detail. Niki designed, implemented, tuned and evaluated countless model variants in our original codebase and tensor2tensor. Llion also experimented with novel model variants, was responsible for our initial codebase, and efficient inference and visualizations. Lukasz and Aidan spent countless long days designing various parts of and implementing tensor2tensor, replacing our earlier codebase, greatly improving results and massively accelerating our research.

[†]Work performed while at Google Brain.

[‡]Work performed while at Google Research.

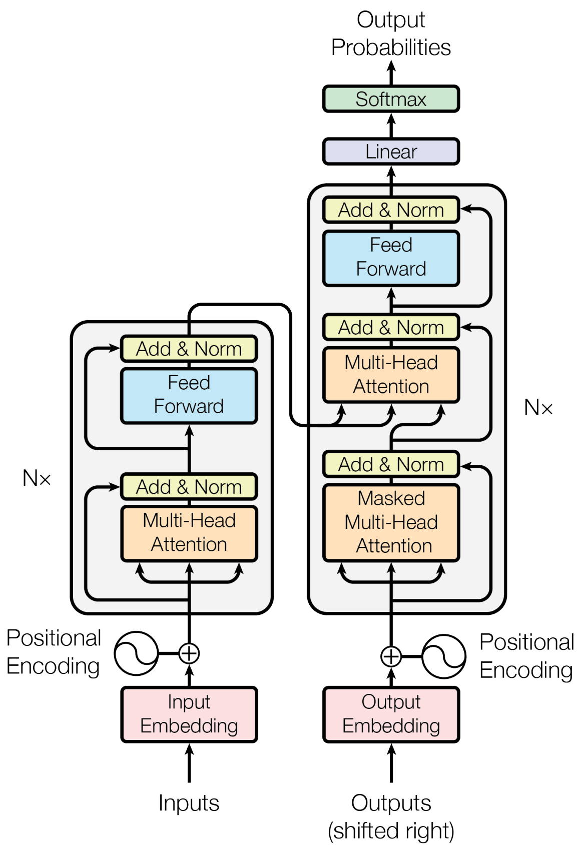


Figure 1: The Transformer - model architecture.

wise fully connected feed-forward network. We employ a residual connection [10] around each of the two sub-layers, followed by layer normalization [1]. That is, the output of each sub-layer is $\mathrm{LayerNorm}(x + \mathrm{Sublayer}(x))$, where $\mathrm{Sublayer}(x)$ is the function implemented by the sub-layer itself. To facilitate these residual connections, all sub-layers in the model, as well as the embedding layers, produce outputs of dimension $d_{\text{model}} = 512$.

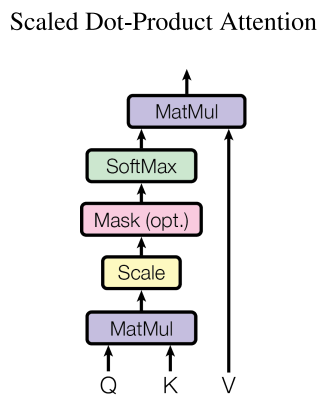
Decoder: The decoder is also composed of a stack of $N = 6$ identical layers. In addition to the two sub-layers in each encoder layer, the decoder inserts a third sub-layer, which performs multi-head attention over the output of the encoder stack. Similar to the encoder, we employ residual connections around each of the sub-layers, followed by layer normalization. We also modify the self-attention sub-layer in the decoder stack to prevent positions from attending to subsequent positions. This masking, combined with fact that the output embeddings are offset by one position, ensures that the predictions for position i can depend only on the known outputs at positions less than i.

3.2 Attention

An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.

3.2.1 Scaled Dot-Product Attention

We call our particular attention "Scaled Dot-Product Attention" (Figure 2). The input consists of queries and keys of dimension d_k, and values of dimension d_v. We compute the dot products of the



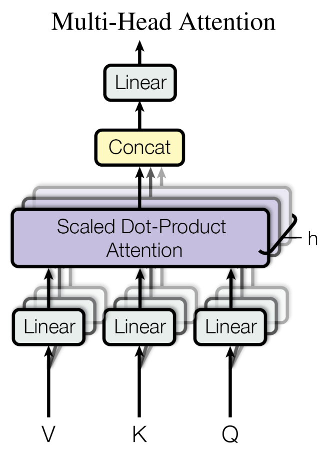


Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.

query with all keys, divide each by $\sqrt{d_k}$, and apply a softmax function to obtain the weights on the values.

In practice, we compute the attention function on a set of queries simultaneously, packed together into a matrix Q. The keys and values are also packed together into matrices K and V. We compute the matrix of outputs as:

$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{1}$$

The two most commonly used attention functions are additive attention [2], and dot-product (multi-plicative) attention. Dot-product attention is identical to our algorithm, except for the scaling factor of $\frac{1}{\sqrt{d_k}}$. Additive attention computes the compatibility function using a feed-forward network with a single hidden layer. While the two are similar in theoretical complexity, dot-product attention is much faster and more space-efficient in practice, since it can be implemented using highly optimized matrix multiplication code.

While for small values of d_k the two mechanisms perform similarly, additive attention outperforms dot product attention without scaling for larger values of d_k [3]. We suspect that for large values of d_k, the dot products grow large in magnitude, pushing the softmax function into regions where it has extremely small gradients [4]. To counteract this effect, we scale the dot products by $\frac{1}{\sqrt{d_k}}$.

3.2.2 Multi-Head Attention

Instead of performing a single attention function with d_{model}-dimensional keys, values and queries, we found it beneficial to linearly project the queries, keys and values h times with different, learned linear projections to d_k, d_k and d_v dimensions, respectively. On each of these projected versions of queries, keys and values we then perform the attention function in parallel, yielding d_v-dimensional output values. These are concatenated and once again projected, resulting in the final values, as depicted in Figure 2.

Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. With a single attention head, averaging inhibits this.

[4]To illustrate why the dot products get large, assume that the components of q and k are independent random variables with mean 0 and variance 1. Then their dot product, $q \cdot k = \sum_{i=1}^{d_k} q_i k_i$, has mean 0 and variance d_k.

$$\begin{aligned}\mathrm{MultiHead}(Q, K, V) &= \mathrm{Concat}(\mathrm{head_1}, ..., \mathrm{head_h})W^O\\ \text{where}~\mathrm{head_i} &= \mathrm{Attention}(QW^Q_i, KW^K_i, VW^V_i)\end{aligned}$$

Where the projections are parameter matrices $W^Q_i \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W^K_i \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W^V_i \in \mathbb{R}^{d_{\text{model}} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$.

In this work we employ $h = 8$ parallel attention layers, or heads. For each of these we use $d_k = d_v = d_{\text{model}}/h = 64$. Due to the reduced dimension of each head, the total computational cost is similar to that of single-head attention with full dimensionality.

3.2.3 Applications of Attention in our Model

The Transformer uses multi-head attention in three different ways:

- In "encoder-decoder attention" layers, the queries come from the previous decoder layer, and the memory keys and values come from the output of the encoder. This allows every position in the decoder to attend over all positions in the input sequence. This mimics the typical encoder-decoder attention mechanisms in sequence-to-sequence models such as [31, 2, 8].
- The encoder contains self-attention layers. In a self-attention layer all of the keys, values and queries come from the same place, in this case, the output of the previous layer in the encoder. Each position in the encoder can attend to all positions in the previous layer of the encoder.
- Similarly, self-attention layers in the decoder allow each position in the decoder to attend to all positions in the decoder up to and including that position. We need to prevent leftward information flow in the decoder to preserve the auto-regressive property. We implement this inside of scaled dot-product attention by masking out (setting to $-\infty$) all values in the input of the softmax which correspond to illegal connections. See Figure 2.

3.3 Position-wise Feed-Forward Networks

In addition to attention sub-layers, each of the layers in our encoder and decoder contains a fully connected feed-forward network, which is applied to each position separately and identically. This consists of two linear transformations with a ReLU activation in between.

$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{2}$$

While the linear transformations are the same across different positions, they use different parameters from layer to layer. Another way of describing this is as two convolutions with kernel size 1. The dimensionality of input and output is $d_{\text{model}} = 512$, and the inner-layer has dimensionality $d_{ff} = 2048$.

3.4 Embeddings and Softmax

Similarly to other sequence transduction models, we use learned embeddings to convert the input tokens and output tokens to vectors of dimension d_{model}. We also use the usual learned linear transformation and softmax function to convert the decoder output to predicted next-token probabilities. In our model, we share the same weight matrix between the two embedding layers and the pre-softmax linear transformation, similar to [24]. In the embedding layers, we multiply those weights by $\sqrt{d_{\text{model}}}$.

3.5 Positional Encoding

Since our model contains no recurrence and no convolution, in order for the model to make use of the order of the sequence, we must inject some information about the relative or absolute position of the tokens in the sequence. To this end, we add "positional encodings" to the input embeddings at the

Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. n is the sequence length, d is the representation dimension, k is the kernel size of convolutions and r the size of the neighborhood in restricted self-attention.

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)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

bottoms of the encoder and decoder stacks. The positional encodings have the same dimension d_{model} as the embeddings, so that the two can be summed. There are many choices of positional encodings, learned and fixed [8].

In this work, we use sine and cosine functions of different frequencies:

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

where pos is the position and i is the dimension. That is, each dimension of the positional encoding corresponds to a sinusoid. The wavelengths form a geometric progression from 2π to $10000 \cdot 2\pi$. We chose this function because we hypothesized it would allow the model to easily learn to attend by relative positions, since for any fixed offset k, PE_{pos+k} can be represented as a linear function of PE_{pos}.

We also experimented with using learned positional embeddings [8] instead, and found that the two versions produced nearly identical results (see Table 3 row (E)). We chose the sinusoidal version because it may allow the model to extrapolate to sequence lengths longer than the ones encountered during training.

4 Why Self-Attention

In this section we compare various aspects of self-attention layers to the recurrent and convolutional layers commonly used for mapping one variable-length sequence of symbol representations $(x_1, ..., x_n)$ to another sequence of equal length $(z_1, ..., z_n)$, with $x_i, z_i \in \mathbb{R}^d$, such as a hidden layer in a typical sequence transduction encoder or decoder. Motivating our use of self-attention we consider three desiderata.

One is the total computational complexity per layer. Another is the amount of computation that can be parallelized, as measured by the minimum number of sequential operations required.

The third is the path length between long-range dependencies in the network. Learning long-range dependencies is a key challenge in many sequence transduction tasks. One key factor affecting the ability to learn such dependencies is the length of the paths forward and backward signals have to traverse in the network. The shorter these paths between any combination of positions in the input and output sequences, the easier it is to learn long-range dependencies [11]. Hence we also compare the maximum path length between any two input and output positions in networks composed of the different layer types.

As noted in Table 1, a self-attention layer connects all positions with a constant number of sequentially executed operations, whereas a recurrent layer requires $O(n)$ sequential operations. In terms of computational complexity, self-attention layers are faster than recurrent layers when the sequence length n is smaller than the representation dimensionality d, which is most often the case with sentence representations used by state-of-the-art models in machine translations, such as word-piece [31] and byte-pair [25] representations. To improve computational performance for tasks involving very long sequences, self-attention could be restricted to considering only a neighborhood of size r in

PUBLICATION ASSOCIATED WITH THIS THESIS

[1] Jahin, M. A. & Supervisor's last name, initials (2024). Title of your article. *Journal Name*, *Volume*(Issue), Article 1. doi: doi link.