SortFormer: Permutation Encoding for Transformer

Taejin Park [*]
NVIDIA, Santa Clara, CA, USA taejinp@nvidia.com

Abstract

We propose *SortFormer* architecture, which is a Transformer variant that encodes permutation information into hidden states. We address the challenge of permutation invariance in processing sequential data using Transformer architectures. Despite their effectiveness in learning and generalizing across text, speech, and image data, Transformers struggle with inputs where permutations should be disregarded. This limitation often results in significant overfitting where the trained model hardly handles permutations on on the unseen datapoints. We identify that the inherent design of the original Transformer architecture contributes to this deficiency. SortFormer operates by generating estimated labels at each layer, which are then sorted according to their arrival order. These sequentially ordered labels serve as a permutative encoding, guiding the Transformer model in recognizing the sequence order. Through ablation studies, we demonstrate that SortFormer exhibits reduced overfitting and enhanced generalization capabilities on unseen data sequences. Furthermore, we provide insights into the practical applicability of SortFormer in real-world scenarios. Our findings suggest that this innovative architecture holds significant promise for improving the robustness and versatility of Transformer models in handling permutation-sensitive sequential data.

1 Introduction

Testing autobuild 8
In recent years, the large language models (LLMs) [12, 15] gained significant popularity and most of the LLMs are based on the Transformer [17] architecture that is now being used for not only language modeling as language understanding [6, 14], GPT3 [3]. Other tasks such as image processing field also started leveraging Transformers such as image generation [11] and image recognition with ViT [7]. In speech recognition field, Conformer [8] and Whisper [13] successfully employed Transformer archietcture to speech-to-text (STT) tasks.

Transformer Variants for Efficiency The wide adoption of Transformer architecture also led to more efficient variants of the model such as Reformer [9], which introduces efficient memory handling in Transformers through locality-sensitive hashing; Transformer-XL [5], enhancing Transformer models for long sequence tasks with segment-level recurrence and a novel positional encoding; Performe [4], offering a scalable Transformer variant with the new attention mechanism for processing long sequences efficiently; Longformer [1], designed for longer texts with a modified attention mechanism that combines sliding window and global attention for tasks like document classification and question answering; SegFormer [18], an approach for semantic segmentation combining a hierarchical Transformer encoder with a lightweight MLP decoder; and RoFormer[16], which enhances Transformer models with a rotational position encoding scheme for improved performance in various natural language processing tasks. These variants showcase significant advancements in handling

[*]Use footnote for providing further information about author (webpage, alternative address)—*not* for acknowledging funding agencies.

long sequences, improving memory efficiency, and extending the applicability of Transformer models in various domains, including text and image processing.

Problem that SortFormer solves We propose a model designed for the simultaneous estimation of class presences from a sequence of input tokens. Consider a set of frame-wise F-dimensional embedding vectors $\{\mathbf{x}_t\}_{t=1}^{T}$, where $\mathbf{x}_t \in \mathbb{R}^D, t = 1, 2, \ldots, T$, representing the frame index. Given the input sequence, the model is expected to generate the estimated sequence $\{\mathbf{y}_t\}_{t=1}^{T}$, where $\mathbf{y}_t \in \mathbb{R}^K, t = 1, 2, \ldots, T$. SortFormer is tasked with estimating the class presences $\{\mathbf{y}_t\}_{t=1}^{T}$. In this context, $\mathbf{y}_t = [y_{1,t}, y_{2,t}, \ldots, y_{K,t}]^\top$ denotes the class presences of K classes at time t. SortFormer operates under the assumption that $y_{k,t}$ is conditionally independent given the embedding vectors (features). This assumption is formalized as:

$$P\left(\mathbf{y}_1, \ldots, \mathbf{y}_T \mid \mathbf{x}_1, \ldots, \mathbf{x}_T\right) = \prod_{k=1}^{K} \prod_{t=1}^{T} P\left(y_{k,t} \mid \mathbf{x}_1, \ldots, \mathbf{x}_T\right). \tag{1}$$

Under this framework, the task at hand is construed as a multi-label classification problem, which is amenable to modeling via a neural network, denoted as f_Θ. The model is defined as:

$$\left(\mathbf{p}_1, \ldots, \mathbf{p}_T\right) = f_\Theta\left(\mathbf{x}_1, \ldots, \mathbf{x}_T\right), \tag{2}$$

where $\mathbf{p}_t = \left[p_{1,t}, \ldots, p_{K,t}\right]^\top \in (0,1)^S$ represents the posterior probabilities of the presences of S classes at frame index t and f represents the Sortformer model with parameter. Each $y_{k,t}$ is defined as follows depending on the value of $p_{k,t}$:

$$y_{k,t} = \begin{cases} 1 & \text{if } p_{k,t} > 0.5 \\ 0 & \text{if } p_{k,t} \leq 0.5 \end{cases} \tag{3}$$

The estimation of class presences $\{\hat{\mathbf{y}}_t\}_{t=1}^{T}$ is given by:

$$\left(\hat{\mathbf{y}}_1, \ldots, \hat{\mathbf{y}}_T\right) = \underset{\mathbf{y}_k \in \mathbf{Y}}{\arg\max} \left\{ P\left(\mathbf{y}_1, \ldots, \mathbf{y}_T \mid \mathbf{x}_1, \ldots, \mathbf{x}_T\right) \right\}, \tag{4}$$

The equation 4 can be rewritten in matrix form by concatenating the vectors $\mathbf{X} = [\mathbf{x}_1 \mathbf{x}_2 \ldots, \mathbf{x}_T]$, where each column of $\mathbf{Y}$ represents the class presences at time t, $\mathbf{Y} = [\mathbf{y}_1\, \mathbf{y}_2\, \ldots\, \mathbf{y}_T]$. In the simplest form, the model can be represented as:

$$\mathbf{Y} = f_\Theta\left(\mathbf{X}\right), \tag{5}$$

where $\mathbf{X} \in \mathbb{R}^{D \times T}$ is the input sequence and $\mathbf{Y} \in \mathbb{R}^{K \times T}$ is a binary matrix contains an output sequence.

Task Description In this problem, the core challenge lies in the output sequence $\mathbf{Y}$, where the number of frames have non-zero $\mathbf{y}_t$ is determined by the model's estimation based on the input sequence $\mathbf{X}$. Here, $\mathbf{X}$ represents an unseen type of data. The model's task is to count the number of distinct classes within the input sequence and generate corresponding class labels for each identified class.

It is crucial to make a clearn distinction between the problem that SortFormer is tackling and the well known tasks such as clustering, identification tasks. This problem shares similarities with clustering tasks, open-set recognition tasks where the objective is to count the number of clusters and assign labels to them. However, there are follwing key characteristics that distinguish this problem from the aforementioned tasks:

- **Infer Unseen Data Class** The task is not a typical classification task, where the model needs to classify the unseen type of class. For example, in open-set face indexing, the model needs to identify the face of people that have not been seen during training time.

- **Multi-labels** Each token into SortFormer can have multiple labels. For example, one picture can contain multiple people's faces. Most clustering systems are not designed to assign multiple labels to a single data point.
- **Non-Event Classification** The system should detect "class absence", which requires the model to learn the characteristics of null-class characteristics of the feature input. Clustering methods typically do not accommodate cases with 'void class', where no discernible patterns exist within a given frame.
- **Set-based Label Assignment Loss** The model should count the number of sets and label each frame accordingly to get generic identifiers. Followingly, the loss should be calculated based on the set-based label assignment. This aspect is often handled with permutation invariant loss (PIL).
- **Small-scale Set Count** The task is expecting $K < 10$ classes, which is a small-scale set count. Most of the neural network based clustering systems are designed to handle large-scale set count where $K > 1000$. This also decreases the burden of applying sorting algorithm on the output classes.

An illustrative example is the speaker diarization problem. In speaker diarization, the input sequence is an audio signal, and the output sequence comprises generic identifiers (e.g., speaker1, speaker2, ..., speakerN) for each frame. However, challenges arise in frames where no speaker is present, resulting in an output $\mathbf{y}_t$ that consists entirely of zeros. Additionally, complications occur in frames with multiple speakers, where $\mathbf{y}_t$ becomes a binary vector containing multiple ones.

This framework is not only relevant to the speaker diarization task but is also applicable to a variety of other problems, such as anomaly detection in time series data, open-set recognition in image processing, open-set facial entity indexation in video streams, unknown language detection in natural language processing, and new genre detection in music classification. the following table summarizes the tasks suitable for SortFormer and their corresponding representations.

Task	**Feature Representation**	**Input Data Type**	**Output Description**
Speaker Diarization	Speaker embeddings	Audio streams	Identifiers for each speaker; 'no label' for silent frames
Anomaly Detection	Signal feature embeddings	Time series data	Anomaly identifiers; 'no label' for normal data points
Open-set Face Indexation	Face embeddings	Video streams	Anonymized identifiers for each face; 'no label' for non-face frames
Unknown Language Detection	Text embeddings	Text data	Language identifiers; 'no label' for unrecognizable languages
New Genre Detection	Music feature embeddings	Audio tracks	Music genre identifiers; 'no label' for unconventional genres

Table 1: Data Analysis Tasks and their Representations

Permutation Invariance in Neural Network Problem

2 Prior Works

clustering There have been numerous studies on making Transformer to perceive longer context, model compressing for memory efficiency and sparse models. However, there have been limited number of published work regarding general-purpose transformer variants which can handle permutation

The most closest previous study to our work is Set-Transformer [10],

permutation invairant loss

clustering

3 Permutation Invariance for Transformers

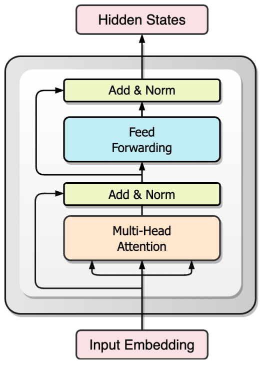

Figure 1: Caption1

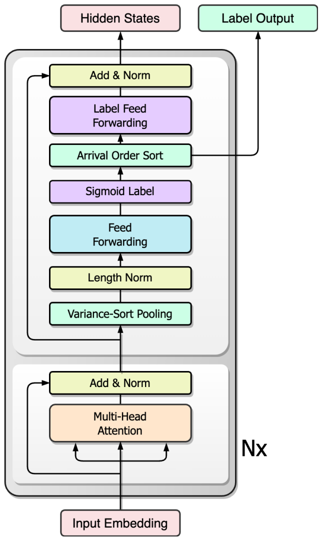

Figure 2: Caption 2

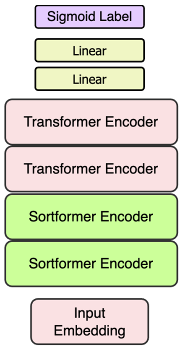

Figure 3: Caption 3

3.1 Copied and Pasted from a Paper

[2]. This section is fo Let $Y \in \mathcal{Y}$ be some random variable of interest, to be predicted from $\mathbf{X}_n$. Permutation invariance of the conditional distribution $P_{Y|\mathbf{x}_n}$ of Y given $\mathbf{X}_n$ is defined as follows.

Definition 1. The conditional distribution $P_{Y|\mathbf{X}_n}$ of Y given $\mathbf{X}_n$ is $\mathbb{S}_n$-invariant if for all $B \in \mathcal{B}(\mathcal{Y})$ and $\pi \in \mathbb{S}_n$,

$$P_{Y|\mathbf{X}_n}[Y \in B \mid \pi \cdot \mathbf{X}_n] = P_{Y|\mathbf{X}_n}[Y \in B \mid \mathbf{X}_n] \quad \text{a.s.} \ - P_{\mathbf{X}_n}. \tag{6}$$

Together with exchangeability of $P_{\mathbf{X}_n}$, conditional invariance implies the joint equality $(\pi \cdot \mathbf{X}_n, Y) \overset{\text{d}}{=} (\mathbf{X}_n, Y)$ for all $\pi \in \mathbb{S}_n$.

In the neural networks literature, equivariance plays a key role in the hidden layers of invariant deterministic neural networks [2, 8, 6]. As we show in Section 2, the following adaptation to random variables plays the same role in $\mathbb{S}_n$-invariant stochastic neural networks.

Definition 2. Let $\mathbf{Y}_n \in \mathcal{Y}^n$ be a random $\mathcal{Y}$-valued sequence. The conditional distribution $P_{\mathbf{Y}_n|\mathbf{x}_n}$ of $\mathbf{Y}_n$ given $\mathbf{X}_n$ is $\mathbb{S}_n$-equivariant if for all $B \in \mathcal{B}(\mathcal{Y}^n)$ and $\pi \in \mathbb{S}_n$,

$$P_{\mathbf{Y}_n|\mathbf{X}_n}[\pi \cdot \mathbf{Y}_n \in B \mid \pi \cdot \mathbf{X}_n] = P_{\mathbf{Y}_n|\mathbf{X}_n}[\mathbf{Y}_n \in B \mid \mathbf{X}_n] \quad \text{a.s.} \ - P_{\mathbf{X}_n}. \tag{7}$$

Together with exchangeability of $P_{\mathbf{X}_n}$, conditional invariance implies the joint equality $(\pi \cdot \mathbf{X}_n, \pi \cdot \mathbf{Y}_n) \overset{\mathrm{d}}{=} (\mathbf{X}_n, \mathbf{Y}_n)$ for all $\pi \in \mathbb{S}_n$

4 Submission of papers to NeurIPS 2023

Please [17] read the instructions below carefully and follow them faithfully. **Important:** This year the checklist will be submitted separately from the main paper in OpenReview, please review it well ahead of the submission deadline: `https://neurips.cc/public/guides/PaperChecklist`.

4.1 Style

Papers to be submitted to NeurIPS 2023 must be prepared according to the instructions presented here. Papers may only be up to **nine** pages long, including figures. Additional pages *containing only acknowledgments and references* are allowed. Papers that exceed the page limit will not be reviewed, or in any other way considered for presentation at the conference.

The margins in 2023 are the same as those in previous years.

Authors are required to use the NeurIPS LaTeX style files obtainable at the NeurIPS website as indicated below. Please make sure you use the current files and not previous versions. Tweaking the style files may be grounds for rejection.

4.2 Retrieval of style files

The style files for NeurIPS and other conference information are available on the website at

`http://www.neurips.cc/`

The file `neurips_2023.pdf` contains these instructions and illustrates the various formatting requirements your NeurIPS paper must satisfy.

The only supported style file for NeurIPS 2023 is `neurips_2023.sty`, rewritten for LaTeX 2_{ε}. **Previous style files for LaTeX 2.09, Microsoft Word, and RTF are no longer supported!**

The LaTeX style file contains three optional arguments: `final`, which creates a camera-ready copy, `preprint`, which creates a preprint for submission to, e.g., arXiv, and `nonatbib`, which will not load the `natbib` package for you in case of package clash.

Preprint option If you wish to post a preprint of your work online, e.g., on arXiv, using the NeurIPS style, please use the `preprint` option. This will create a nonanonymized version of your work with the text "Preprint. Work in progress." in the footer. This version may be distributed as you see fit, as long as you do not say which conference it was submitted to. Please **do not** use the `final` option, which should **only** be used for papers accepted to NeurIPS.

At submission time, please omit the `final` and `preprint` options. This will anonymize your submission and add line numbers to aid review. Please do *not* refer to these line numbers in your paper as they will be removed during the generation of camera-ready copies.

The file `neurips_2023.tex` may be used as a "shell" for writing your paper. All you have to do is replace the author, title, abstract, and text of the paper with your own.

The formatting instructions contained in these style files are summarized in Sections 5, 6, and 7 below.

5 General formatting instructions

The text must be confined within a rectangle 5.5 inches (33 picas) wide and 9 inches (54 picas) long. The left margin is 1.5 inch (9 picas). Use 10 point type with a vertical spacing (leading) of 11 points. Times New Roman is the preferred typeface throughout, and will be selected for you by default. Paragraphs are separated by ½ line space (5.5 points), with no indentation.

The paper title should be 17 point, initial caps/lower case, bold, centered between two horizontal rules. The top rule should be 4 points thick and the bottom rule should be 1 point thick. Allow ¼ inch space above and below the title to rules. All pages should start at 1 inch (6 picas) from the top of the page.

For the final version, authors' names are set in boldface, and each name is centered above the corresponding address. The lead author's name is to be listed first (left-most), and the co-authors' names (if different address) are set to follow. If there is only one co-author, list both author and co-author side by side.

Please pay special attention to the instructions in Section 7 regarding figures, tables, acknowledgments, and references.

6 Headings: first level

All headings should be lower case (except for first word and proper nouns), flush left, and bold.

First-level headings should be in 12-point type.

6.1 Headings: second level

Second-level headings should be in 10-point type.

6.1.1 Headings: third level

Third-level headings should be in 10-point type.

Paragraphs There is also a `\paragraph` command available, which sets the heading in bold, flush left, and inline with the text, with the heading followed by 1 em of space.

7 Citations, figures, tables, references

These instructions apply to everyone.

7.1 Citations within the text

The `natbib` package will be loaded for you by default. Citations may be author/year or numeric, as long as you maintain internal consistency. As to the format of the references themselves, any style is acceptable as long as it is used consistently.

The documentation for `natbib` may be found at

```
http://mirrors.ctan.org/macros/latex/contrib/natbib/natnotes.pdf
```

Of note is the command `\citet`, which produces citations appropriate for use in inline text. For example,

```
\citet{hasselmo} investigated\dots
```

produces

Hasselmo, et al. (1995) investigated...

If you wish to load the `natbib` package with options, you may add the following before loading the `neurips_2023` package:

```
\PassOptionsToPackage{options}{natbib}
```

If `natbib` clashes with another package you load, you can add the optional argument `nonatbib` when loading the style file:

```
\usepackage[nonatbib]{neurips_2023}
```

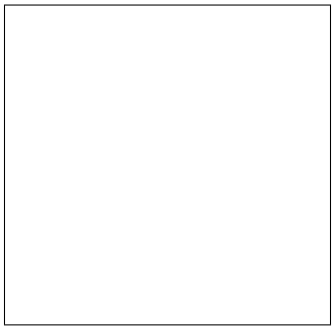

Figure 4: Sample figure caption.

As submission is double blind, refer to your own published work in the third person. That is, use "In the previous work of Jones et al. [4]," not "In our previous work [4]." If you cite your other papers that are not widely available (e.g., a journal paper under review), use anonymous author names in the citation, e.g., an author of the form "A. Anonymous" and include a copy of the anonymized paper in the supplementary material.

7.2 Footnotes

Footnotes should be used sparingly. If you do require a footnote, indicate footnotes with a number[2] in the text. Place the footnotes at the bottom of the page on which they appear. Precede the footnote with a horizontal rule of 2 inches (12 picas).

Note that footnotes are properly typeset *after* punctuation marks.[3]

7.3 Figures

All artwork must be neat, clean, and legible. Lines should be dark enough for purposes of reproduction. The figure number and caption always appear after the figure. Place one line space before the figure caption and one line space after the figure. The figure caption should be lower case (except for first word and proper nouns); figures are numbered consecutively.

You may use color figures. However, it is best for the figure captions and the paper body to be legible if the paper is printed in either black/white or in color.

7.4 Tables

All tables must be centered, neat, clean and legible. The table number and title always appear before the table. See Table 2.

Place one line space before the table title, one line space after the table title, and one line space after the table. The table title must be lower case (except for first word and proper nouns); tables are numbered consecutively.

Note that publication-quality tables *do not contain vertical rules.* We strongly suggest the use of the `booktabs` package, which allows for typesetting high-quality, professional tables:

`https://www.ctan.org/pkg/booktabs`

This package was used to typeset Table 2.

7.5 Math

Note that display math in bare TeX commands will not create correct line numbers for submission. Please use LaTeX (or AMSTeX) commands for unnumbered display math. (You

[2]Sample of the first footnote.

[3]As in this example.

Table 2: Sample table title

Part		
Name	Description	Size (μm)
Dendrite	Input terminal	$\sim$100
Axon	Output terminal	$\sim$10
Soma	Cell body	up to 10^6

really shouldn't be using $$ anyway; see `https://tex.stackexchange.com/questions/503/why-is-preferable-to` and `https://tex.stackexchange.com/questions/40492/what-are-the-differences-between-align-equation-and-displaymath` for more information.)

7.6 Final instructions

Do not change any aspects of the formatting parameters in the style files. In particular, do not modify the width or length of the rectangle the text should fit into, and do not change font sizes (except perhaps in the **References** section; see below). Please note that pages should be numbered.

8 Preparing PDF files

Please prepare submission files with paper size "US Letter," and not, for example, "A4."

Fonts were the main cause of problems in the past years. Your PDF file must only contain Type 1 or Embedded TrueType fonts. Here are a few instructions to achieve this.

- You should directly generate PDF files using `pdflatex`.
- You can check which fonts a PDF files uses. In Acrobat Reader, select the menu Files>Document Properties>Fonts and select Show All Fonts. You can also use the program `pdffonts` which comes with `xpdf` and is available out-of-the-box on most Linux machines.
- `xfig` "patterned" shapes are implemented with bitmap fonts. Use "solid" shapes instead.
- The `\bbold` package almost always uses bitmap fonts. You should use the equivalent AMS Fonts:

  ```
  \usepackage{amsfonts}
  ```

  followed by, e.g., `\mathbb{R}`, `\mathbb{N}`, or `\mathbb{C}` for $\mathbb{R}$, $\mathbb{N}$ or $\mathbb{C}$. You can also use the following workaround for reals, natural and complex:

  ```
  \newcommand{\RR}{I\!\!R} %real numbers
  \newcommand{\Nat}{I\!\!N} %natural numbers
  \newcommand{\CC}{I\!\!\!\!C} %complex numbers
  ```

  Note that `amsfonts` is automatically loaded by the `amssymb` package.

If your file contains type 3 fonts or non embedded TrueType fonts, we will ask you to fix it.

8.1 Margins in LaTeX

Most of the margin problems come from figures positioned by hand using `\special` or other commands. We suggest using the command `\includegraphics` from the `graphicx` package. Always specify the figure width as a multiple of the line width as in the example below:

```
\usepackage[pdftex]{graphicx} ...
\includegraphics[width=0.8\linewidth]{myfile.pdf}
```

See Section 4.4 in the graphics bundle documentation (`http://mirrors.ctan.org/macros/latex/required/graphics/grfguide.pdf`)

A number of width problems arise when LaTeX cannot properly hyphenate a line. Please give LaTeX hyphenation hints using the `\-` command when necessary.

Acknowledgments and Disclosure of Funding

Use unnumbered first level headings for the acknowledgments. All acknowledgments go at the end of the paper before the list of references. Moreover, you are required to declare funding (financial activities supporting the submitted work) and competing interests (related financial activities outside the submitted work). More information about this disclosure can be found at: `https://neurips.cc/Conferences/2023/PaperInformation/FundingDisclosure`.

Do **not** include this section in the anonymized submission, only in the final paper. You can use the `ack` environment provided in the style file to autmoatically hide this section in the anonymized submission.

9 Supplementary Material

Authors may wish to optionally include extra information (complete proofs, additional experiments and plots) in the appendix. All such materials should be part of the supplemental material (submitted separately) and should NOT be included in the main submission.

References

References follow the acknowledgments in the camera-ready paper. Use unnumbered first-level heading for the references. Any choice of citation style is acceptable as long as you are consistent. It is permissible to reduce the font size to `small` (9 point) when listing the references. Note that the Reference section does not count towards the page limit.

References

[1] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

[2] Benjamin Bloem-Reddy and Yee Whye Teh. Neural network models of exchangeable sequences. In *NeurIPS Workshop on Bayesian Deep Learning*, 2018.

[3] Tom B Brown, Benjamin Mann, Nick Ryder, Mike Subramanian, Jared Kaplan, Praveen Dhariwal, Dario Neel, Pranav Shyam, Adam Mishkin, and Alec Radford. Gpt-3: Language models are few-shot learners. *arXiv preprint arXiv:2001.08237*, 2020.

[4] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

[5] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Niu, Kristina Toutanova, Yonatan M. Belinkov, Michael Peters, Vamsi Nair, and Paul Poliakoff. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Advances in neural information processing systems*, pages 9385–9404, 2018.

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[8] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, et al. Conformer: Convolution-augmented transformer for speech recognition. *arXiv preprint arXiv:2005.08100*, 2020.

[9] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *International Conference on Learning Representations*, 2019.

[10] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*, pages 3744–3753. PMLR, 2019.

[11] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Llion Jones, Aidan N Gómez, Illia Polosukhin, Lukasz Kaiser, and Illia Polosukhin. Image transformer. *arXiv preprint arXiv:1802.05720*, 2018.

[12] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[13] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. In *International Conference on Machine Learning*, pages 28492–28518. PMLR, 2023.

[14] Colin Raffel, Noam Shazeer, Adam Roberts, Kim Lee, Nikita Parekh, and Karthik Narasimhan. Learning to transfer the effectiveness of pretraining to downstream tasks. In *International Conference on Learning Representations*, 2019.

[15] Rajkrishna Rao, Chenglong Chen, Marcus Hendricks, Mathias Kreutzer, Rasa Li, Michael Mishkin, Noah Price, Shafiq Puri, Muhammed Qadir, Siva Reddi, Vikas Shobhana, Yash Talwar, Yen-Ling Tay, Tong Wang, Ming Wei, Eric Wilcox, Yu Wu, Yuan Yuan, Pengcheng Zhang, and Yu Zhou. Megatron-turing nlg: Training an extensively labeled 530b parameter language model. *arXiv preprint arXiv:2201.11991*, 2022.

[16] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, page 127063, 2023.

[17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[18] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. *Advances in Neural Information Processing Systems*, 34:12077–12090, 2021.