Transformer Review

BY DEQUAN ER

Outline

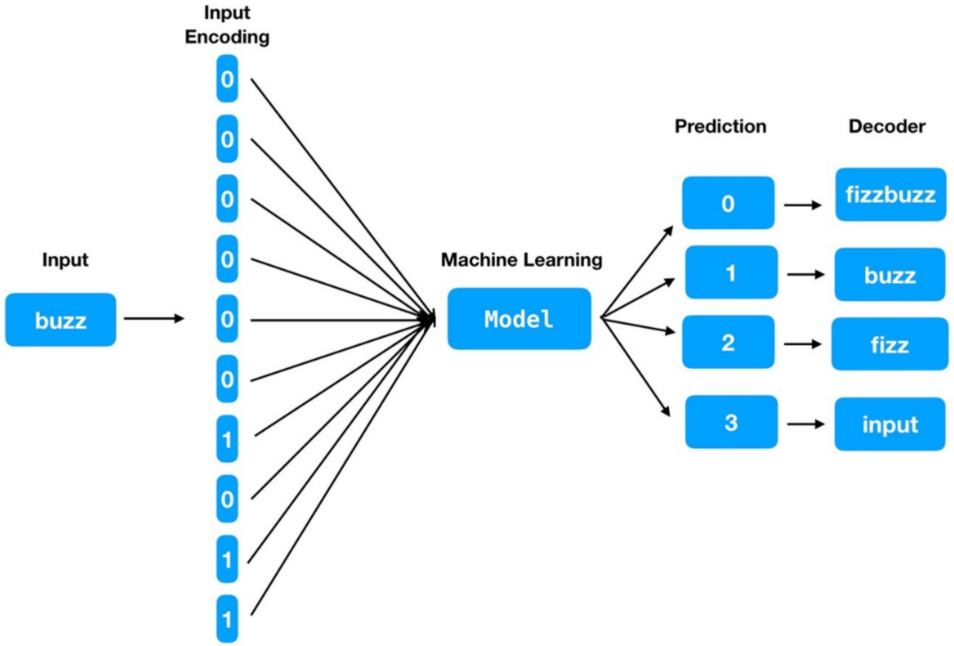
- Introduction
- Model architecture
- Result
- Derived models
- Applications

Introduction

- From Sequence Modeling to Attention (2017)
  - From NLP problem
  - RNN, LSTM, GRU
  - From recurrent language models to encoder-decoder model
    - H_{t-1} to h_t
    - Hard to parallelize
    - Long sequence memory lose, large H_t
  - Attention used from encoder to decoder

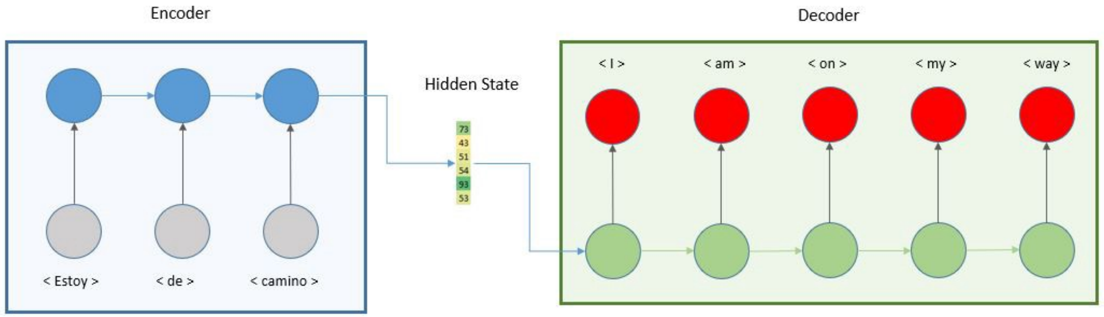
- Derivatives of Transformers
  - BERT (2018), GPT3 (2020)
  - ChatGPT(2022)

- From NLP to audio, video, etc.

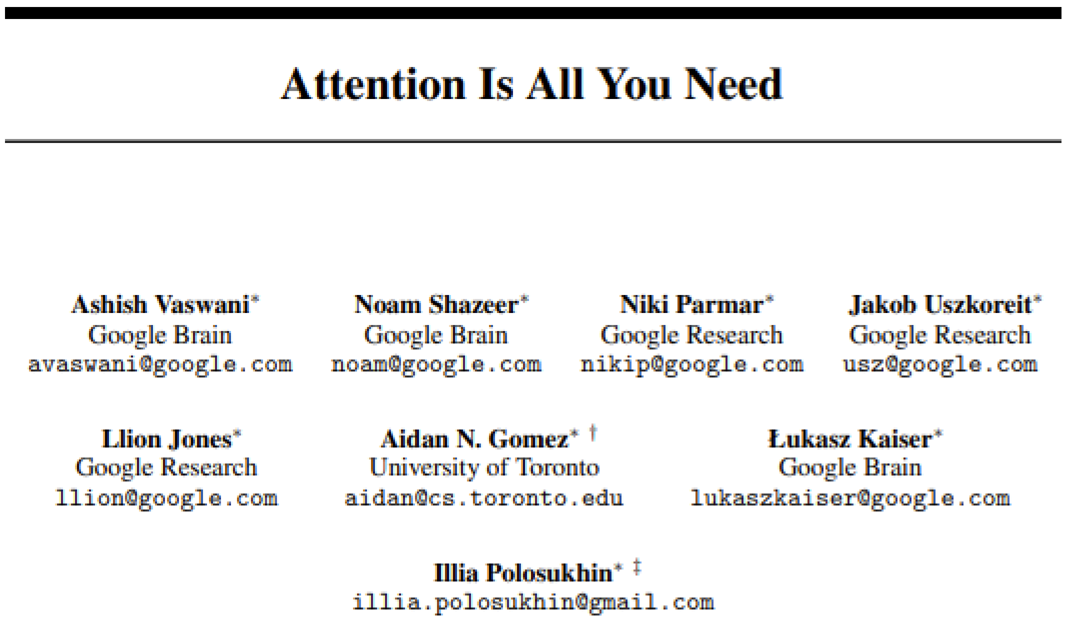


Model Architecture

- Structure of encoder-decoder



Model Architecture



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

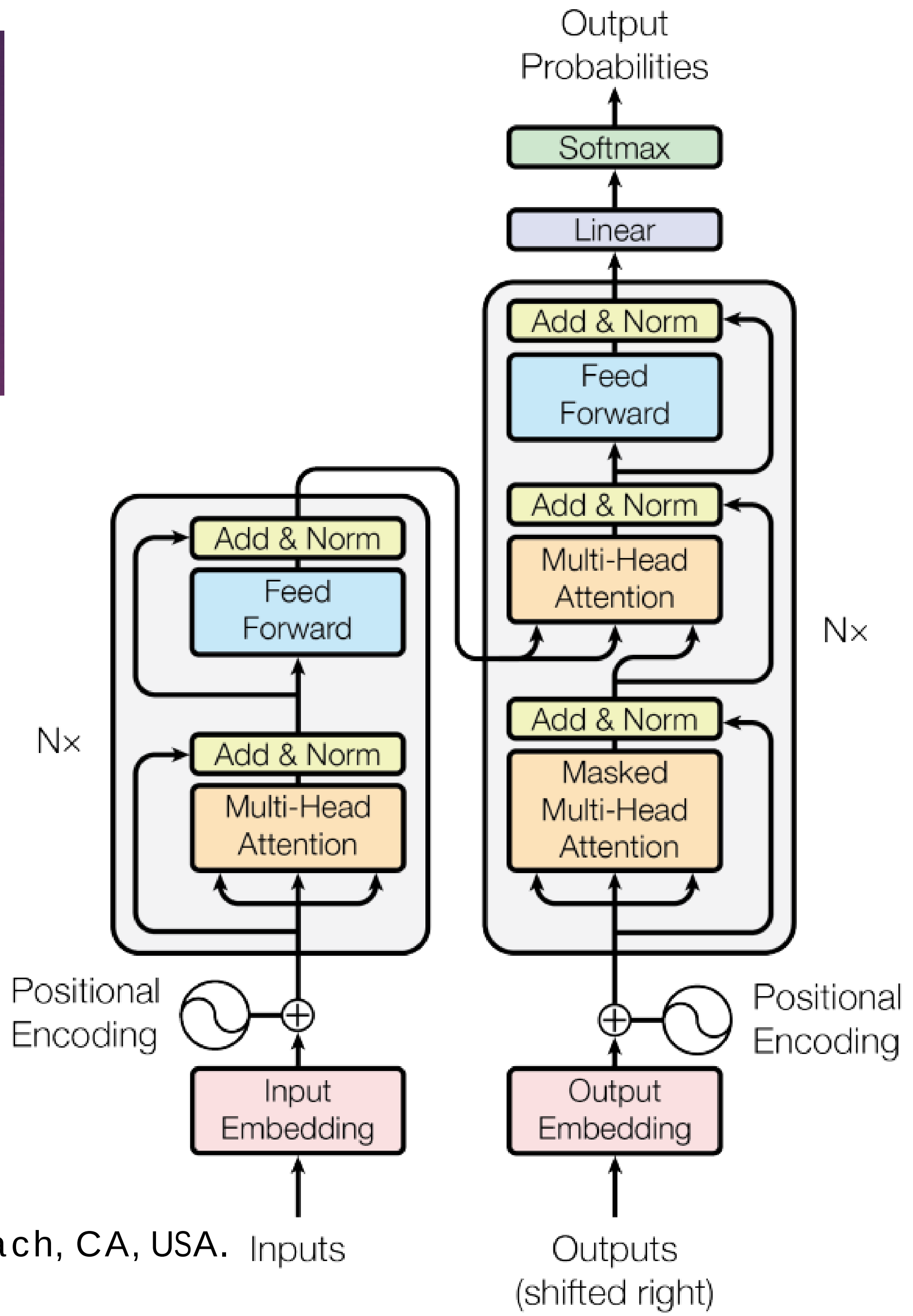
Jakob Uszkoreit*
Google Research
usz@google.com

Llion Jones*
Google Research
llion@google.com

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

Łukasz Kaiser*
Google Brain
lukaszkaiser@google.com

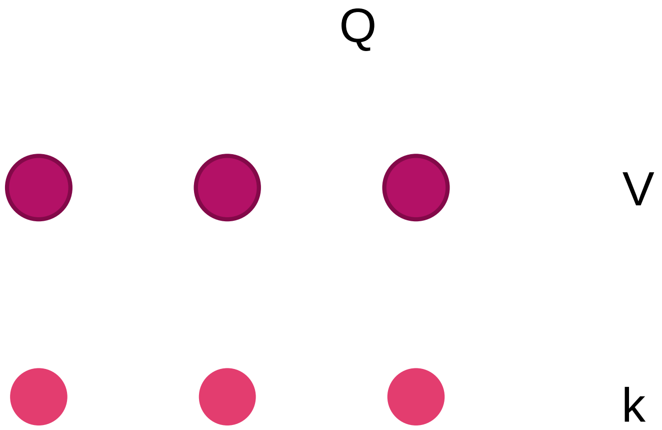
Illia Polosukhin* ‡
illia.polosukhin@gmail.com



31st Conference on Neural Information Processing Systems (**NIPS 2017**), Long Beach, CA, USA.

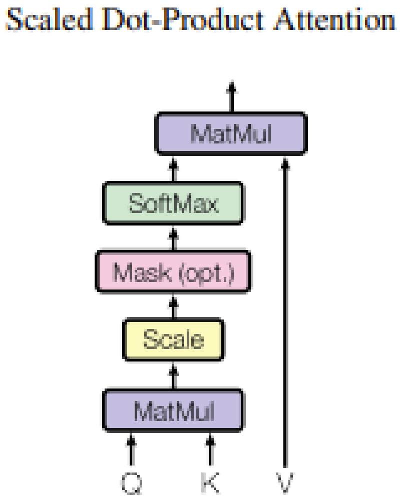
Model Architecture

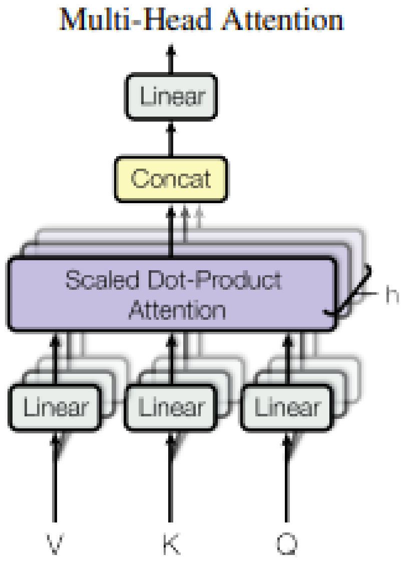
- Attention layer: mapping a query and a set of key-value pairs to an output in vector form.



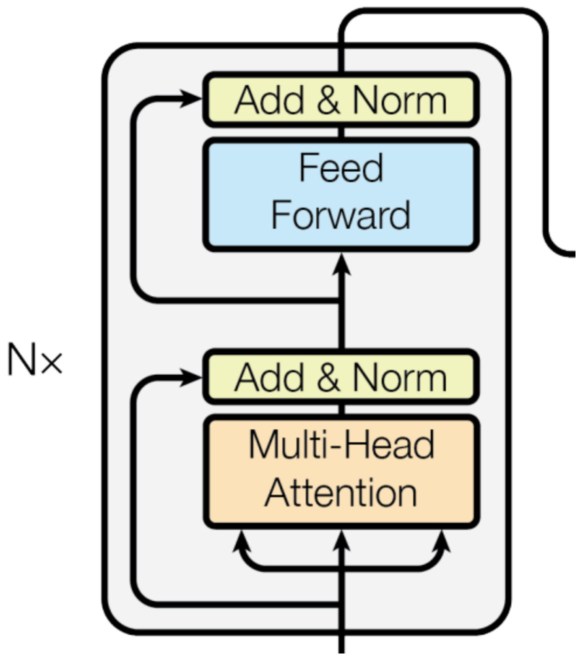
Model Architecture

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


Scaled Dot-Product Attention


Multi-Head Attention

Model Architecture

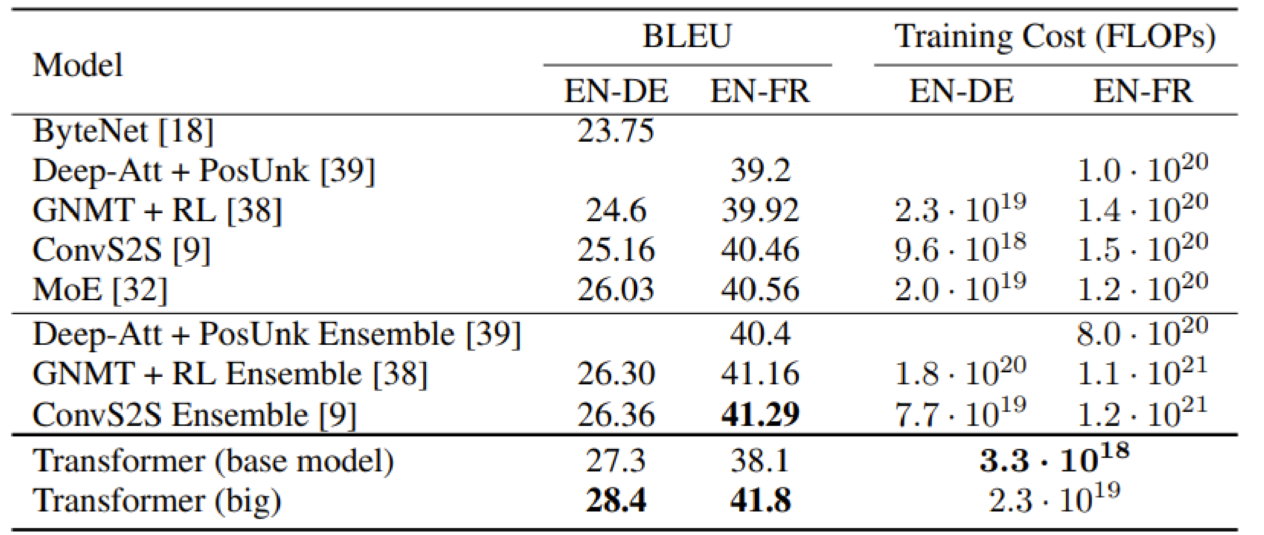


- Position-wise Feed-Forward Networks

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

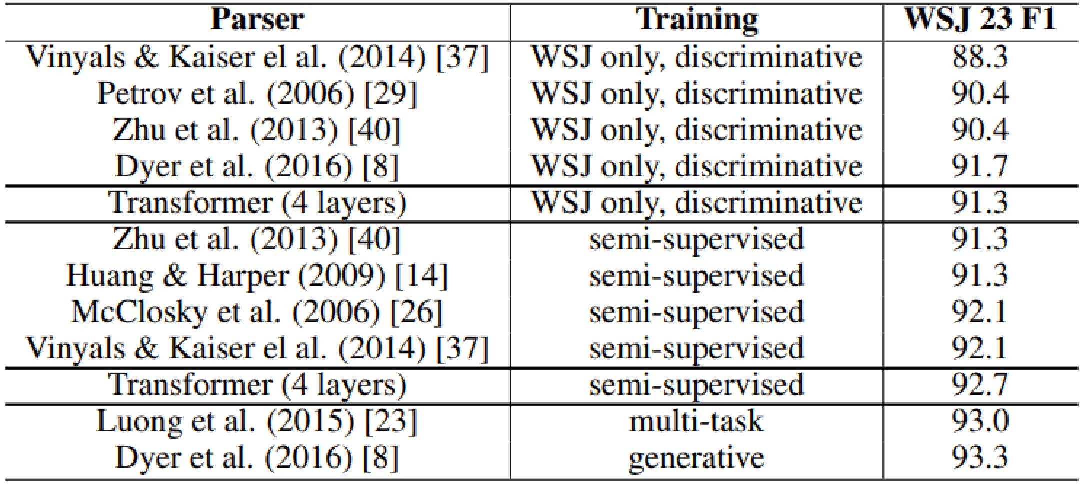
- Positional Encoding
  - Ensure sequence of tokens

Result



Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

Table 4: The Transformer generalizes well to English constituency parsing (Results are on Section 23 of WSJ)



Parser	**Training**	**WSJ 23 F1**
Vinyals & Kaiser el al. (2014) [37]	WSJ only, discriminative	88.3
Petrov et al. (2006) [29]	WSJ only, discriminative	90.4
Zhu et al. (2013) [40]	WSJ only, discriminative	90.4
Dyer et al. (2016) [8]	WSJ only, discriminative	91.7
Transformer (4 layers)	WSJ only, discriminative	91.3
Zhu et al. (2013) [40]	semi-supervised	91.3
Huang & Harper (2009) [14]	semi-supervised	91.3
McClosky et al. (2006) [26]	semi-supervised	92.1
Vinyals & Kaiser el al. (2014) [37]	semi-supervised	92.1
Transformer (4 layers)	semi-supervised	92.7
Luong et al. (2015) [23]	multi-task	93.0
Dyer et al. (2016) [8]	generative	93.3

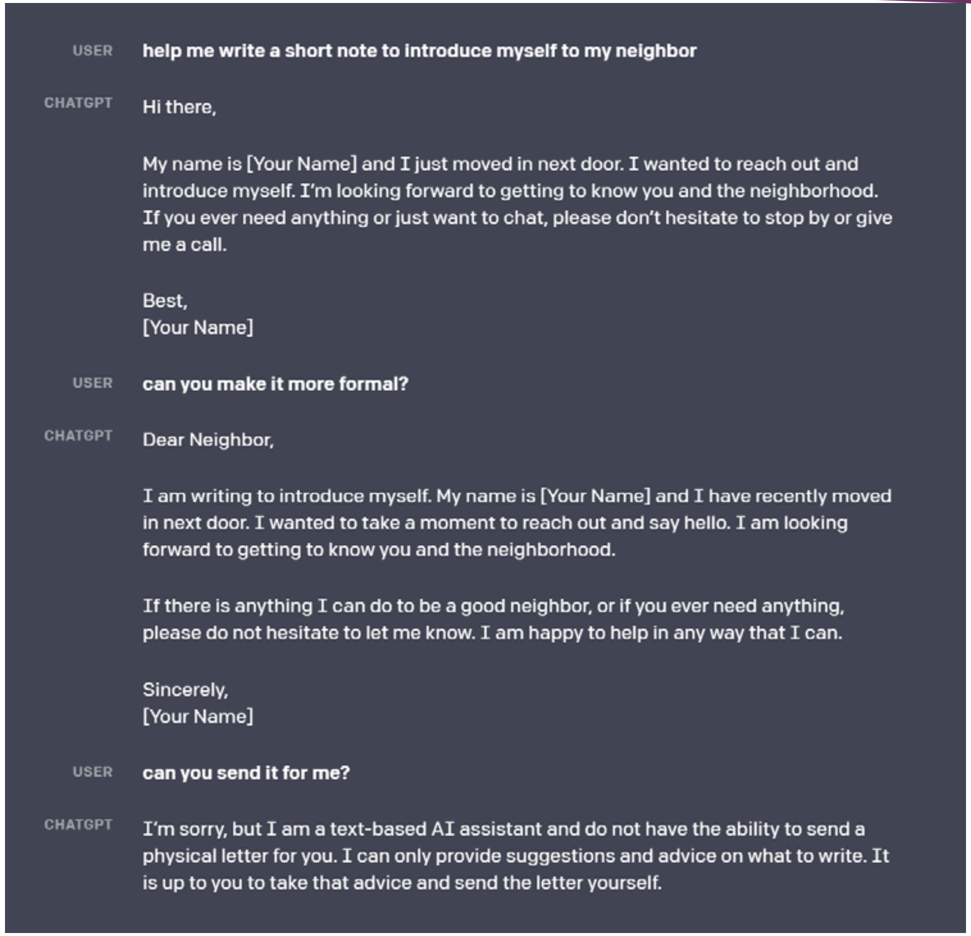
Derived models

- BERT: https://www.codemotion.com/magazine/ai-ml/bert-how-google-changed-nlp-and-how-to-benefit-from-this/

- GPT: https://huggingface.co/docs/transformers/model_doc/openai-gpt
- GPT3: https://github.com/openai/gpt-3

- ChatGPT: https://openai.com/blog/chatgpt/
  - It has 175 Bn parameters

ChatGPT



USER **help me write a short note to introduce myself to my neighbor**

CHATGPT Hi there,

My name is [Your Name] and I just moved in next door. I wanted to reach out and introduce myself. I'm looking forward to getting to know you and the neighborhood. If you ever need anything or just want to chat, please don't hesitate to stop by or give me a call.

Best,
[Your Name]

USER **can you make it more formal?**

CHATGPT Dear Neighbor,

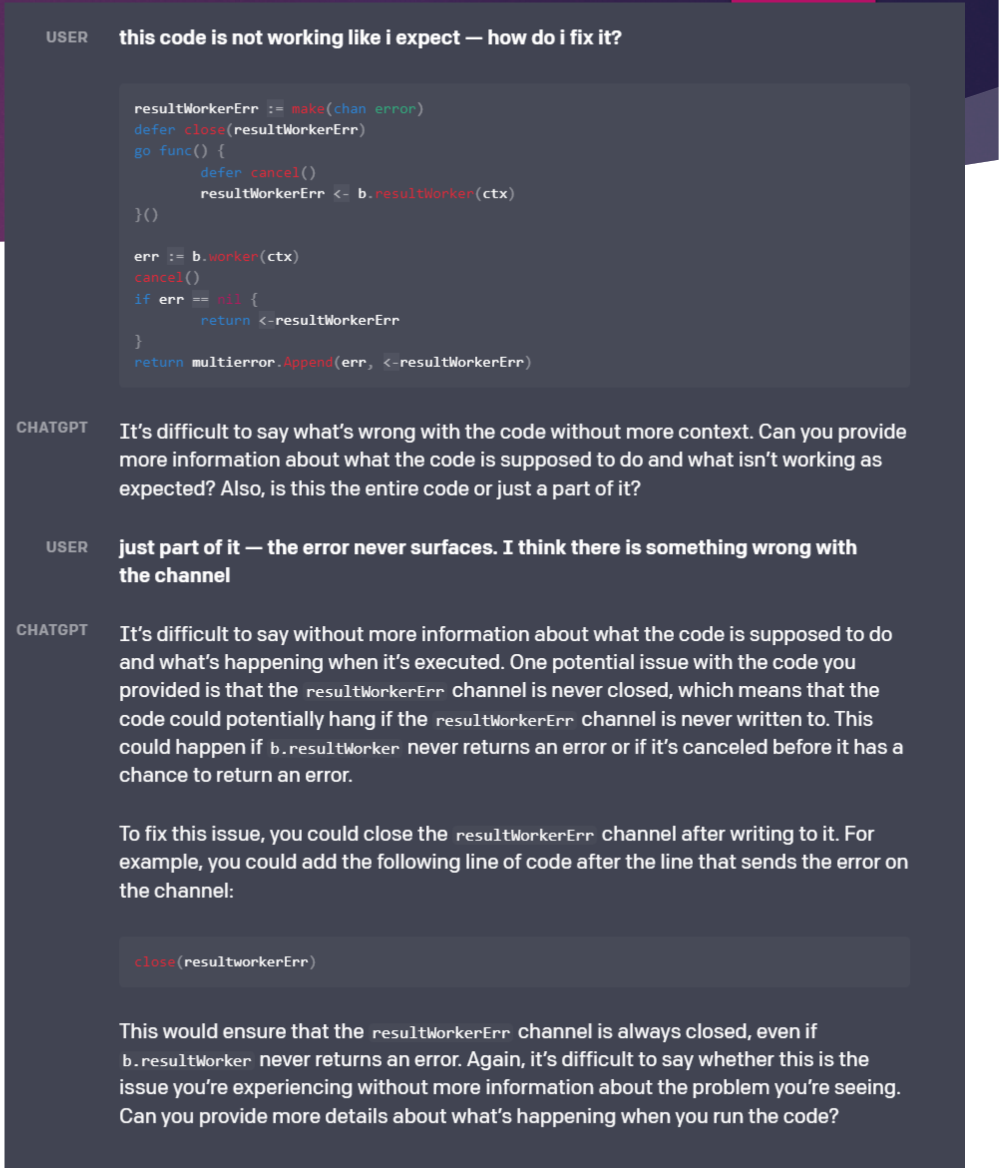
I am writing to introduce myself. My name is [Your Name] and I have recently moved in next door. I wanted to take a moment to reach out and say hello. I am looking forward to getting to know you and the neighborhood.

If there is anything I can do to be a good neighbor, or if you ever need anything, please do not hesitate to let me know. I am happy to help in any way that I can.

Sincerely,
[Your Name]

USER **can you send it for me?**

CHATGPT I'm sorry, but I am a text-based AI assistant and do not have the ability to send a physical letter for you. I can only provide suggestions and advice on what to write. It is up to you to take that advice and send the letter yourself.



USER **this code is not working like i expect — how do i fix it?**

```
resultWorkerErr := make(chan error)
defer close(resultWorkerErr)
go func() {
        defer cancel()
        resultWorkerErr <- b.resultWorker(ctx)
}()

err := b.worker(ctx)
cancel()
if err == nil {
        return <-resultWorkerErr
}
return multierror.Append(err, <-resultWorkerErr)
```

CHATGPT It's difficult to say what's wrong with the code without more context. Can you provide more information about what the code is supposed to do and what isn't working as expected? Also, is this the entire code or just a part of it?

USER **just part of it — the error never surfaces. I think there is something wrong with the channel**

CHATGPT It's difficult to say without more information about what the code is supposed to do and what's happening when it's executed. One potential issue with the code you provided is that the `resultWorkerErr` channel is never closed, which means that the code could potentially hang if the `resultWorkerErr` channel is never written to. This could happen if `b.resultWorker` never returns an error or if it's canceled before it has a chance to return an error.

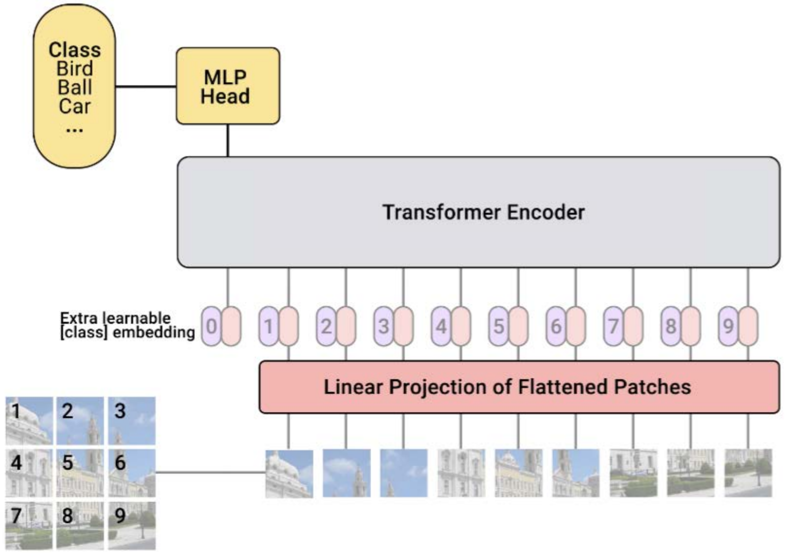
To fix this issue, you could close the `resultWorkerErr` channel after writing to it. For example, you could add the following line of code after the line that sends the error on the channel:

```
close(resultworkerErr)
```

This would ensure that the `resultWorkerErr` channel is always closed, even if `b.resultWorker` never returns an error. Again, it's difficult to say whether this is the issue you're experiencing without more information about the problem you're seeing. Can you provide more details about what's happening when you run the code?

Applications

- NLP
  - Sentiment analysis
- Vision Transformers
  - ViT (2020)
  - Swin Transformer (2021)
  - DINO, DETR …
- Audio


Class
Bird
Ball
Car
...
MLP Head
Transformer Encoder
Extra learnable [class] embedding
0
1
2
3
4
5
6
7
8
9
Linear Projection of Flattened Patches
1
2
3
4
5
6
7
8
9

Vision Transformer model architecture in action, gif from Goole AI blog

Q&A