جامعة الأمير مقرن بن عبد العزيز
University of Prince Mugrin

AI 417 - INTRODUCTION TO DEEP LEARNING

Spring 2023

Portuguese to English Modified Transformer-Based Machine Translation Models

Maryam Aqel 4111815

Zainab Abdelhadi 4111296

Waad Alhejali 4110091

Omnia Al-Juhani 4111719

22th May, 2023

Instructor: Dr. Ahmed Elhayek

Table of Content

- Abstract

In our project, we developed a modified Transformer model for Portuguese to English machine translation. Our modifications included adding a Masked Multi-Head Attention component to the encoder, and also experimenting with different feedforward architectures such as ResNet, CNN, and LSTM. We also removed the encoder and decoder parts of the Transformer model interchangeably to evaluate their importance in the architecture. Through careful evaluation and comparison with the original performance, we gained insights into the strengths and weaknesses of different architectural choices for machine translation tasks. Our project highlights the flexibility and adaptability of the Transformer model architecture, and demonstrates the importance of experimentation in improving translation quality and training efficiency.

- Sections of the report

1.0 Introduction

Our project focuses on improving transformer-based machine translation models for Portuguese to English translation. We started by introducing the original transformer architecture and then made three modifications to it. The first modification involved replacing the traditional MLP feed-forward part with CNN, ResNet, and LSTM networks to see how this affects the translation performance in terms of accuracy and efficiency.

In addition to this, we made two more modifications to the transformer-based machine translation models. The second modification involved removing the decoder and only using the encoder for translation, and the third modification involved removing the encoder and only using the decoder for translation. These modifications aimed to evaluate the impact of each component on the models and gain insights into their strengths and weaknesses.

Through these modifications, we aim to contribute to the field of NLP and DL by improving the accuracy and efficiency of machine translation models for Portuguese to English translation. This work has significant implications for many industries and domains that rely on accurate translation, and the insights gained can be used to inform future research in this area.

2.0 Features of the project

Features of Actual Transformer-Model

- Self-Attention Mechanism: The Transformer relies heavily on self-attention mechanisms, which allow each word in the input sequence to attend to other words, capturing dependencies and relationships across the entire sequence. Self-attention enables the model to consider context from different positions without explicitly using recurrent or convolutional layers.
- Multi-Head Attention: The Transformer employs multiple attention heads to capture different types of information. Each head performs a separate weighted sum over the input sequence, providing multiple perspectives and enhancing the model's ability to attend to different parts of the sequence simultaneously.
- Positional Encoding: Since Transformers do not have recurrent or convolutional operations that inherently capture sequence order, positional encoding is introduced to incorporate positional information. Positional encodings are added to the input embeddings and provide the model with a sense of word order.
- Encoder-Decoder Structure: The Transformer architecture consists of an encoder and a decoder. The encoder processes the input sequence and produces a representation that captures the input's contextual information. The decoder takes this representation and generates the output sequence step by step.
- Residual Connections and Layer Normalization: Residual connections are used throughout the Transformer architecture to mitigate the vanishing gradient problem and improve gradient flow during training. Layer normalization is applied after each sub-layer to stabilize the learning process and improve the model's ability to generalize.
- Feedforward Neural Networks: The Transformer incorporates feedforward neural networks with position-wise fully connected layers. This allows each position in the sequence to be processed independently and nonlinear transformations to be applied to capture complex patterns and features.
- Masking: Masking is applied to the decoder's self-attention layer to prevent it from attending to future positions during training, ensuring that each prediction only depends on previously generated outputs.

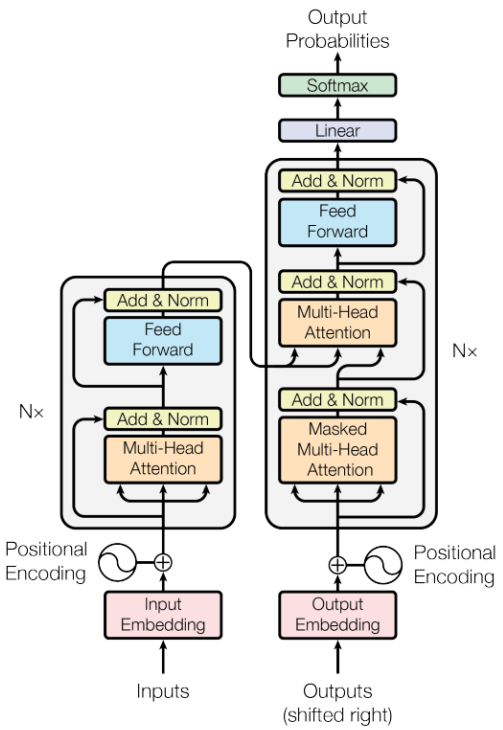


Figure 1: The Transformer - model architecture.

Modifications of the project

- Modifying the Feedforward Part
  - CNN layers, residual connections, and layer normalization , Relu activation
  - Convolutional Layer: GELU (Gaussian Error Linear Unit) Activation
  - LSTM , Relu Activation
  - Resnet 2 layers with skip connections.
- Modified the Encoder and Decoder inner Architecture.
- Removing the Encoder Part
- Removing the Decoder Part

P.S: The code was ran on 5 Epochs, 4 tokens, Bach size, 200 buffer size due to lack of resources and extensive time requirement.

The Ground Truth for the actual model gave the following results:

```
Model: "transformer"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 encoder_1 (Encoder)         multiple                  3632768

 decoder_1 (Decoder)         multiple                  5647104

 dense_38 (Dense)            multiple                  904290

=================================================================
Total params: 10,184,162
Trainable params: 10,184,162
Non-trainable params: 0
```

```
Epoch 1/5
418/418 [==============================] - 461s 1s/step - loss: 4.6248 - masked_accuracy: 0.3061 - val_loss: 4.0128 - val_masked_accuracy: 0.3714
Epoch 2/5
418/418 [==============================] - 442s 1s/step - loss: 3.6533 - masked_accuracy: 0.4067 - val_loss: 3.5418 - val_masked_accuracy: 0.4288
Epoch 3/5
418/418 [==============================] - 436s 1s/step - loss: 3.2574 - masked_accuracy: 0.4518 - val_loss: 3.2976 - val_masked_accuracy: 0.4569
Epoch 4/5
418/418 [==============================] - 437s 1s/step - loss: 2.9989 - masked_accuracy: 0.4807 - val_loss: 3.1722 - val_masked_accuracy: 0.4732
Epoch 5/5
418/418 [==============================] - 437s 1s/step - loss: 2.7994 - masked_accuracy: 0.5016 - val_loss: 3.0975 - val_masked_accuracy: 0.4867
<keras.callbacks.History at 0x7fa4c2710880>
```

- Features of your design and implementation that worked well, and why?.

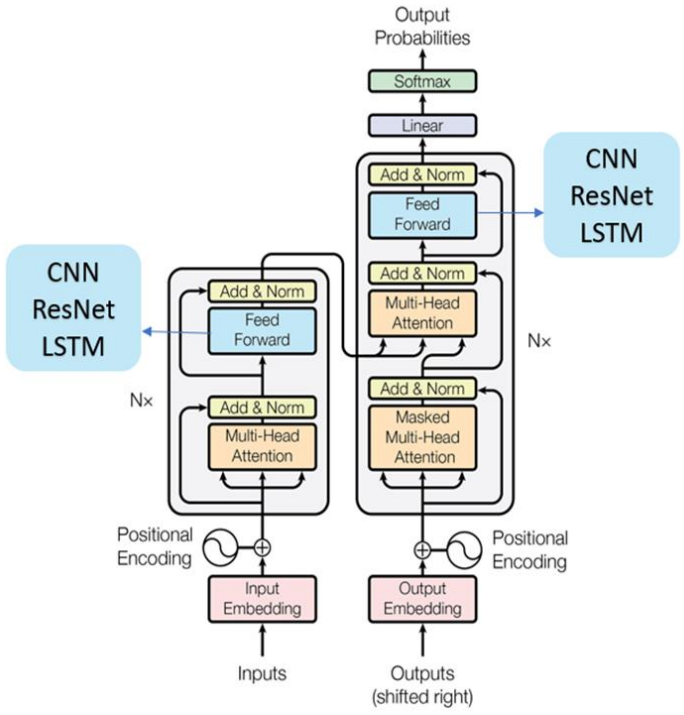
1- Modifying the Feedforward Architecture

A. Changing the MLP feedforward to CNN with residual connections

It gave a good result and 0.4780 validation masked accuracy which is close from the original that gave 0.4867, when running on examples it was able to identify most words and gave a whole correct prediction with the G.T. The parameters were almost the same but slightly increased. Changing the feedforward layer to a CNN with residual connections can improve the capture of local dependencies, enhance parameter efficiency, leverage residual connections for better gradient flow, enable effective feature extraction, and potentially lead to improved translation accuracy and handling of long-range dependencies.

```
Model: "transformer_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 encoder_3 (Encoder)         multiple                  3765888

 decoder_3 (Decoder)         multiple                  5780224

 dense_77 (Dense)            multiple                  904290

=================================================================
Total params: 10,450,402
Trainable params: 10,450,402
Non-trainable params: 0
```

```
Epoch 1/5
418/418 [==============================] - 684s 2s/step - loss: 6.3726 - masked_accuracy: 0.1153 - val_loss: 4.6701 - val_masked_accuracy: 0.2659
Epoch 2/5
418/418 [==============================] - 641s 2s/step - loss: 4.1483 - masked_accuracy: 0.3354 - val_loss: 3.7933 - val_masked_accuracy: 0.3939
Epoch 3/5
418/418 [==============================] - 625s 1s/step - loss: 3.5304 - masked_accuracy: 0.4150 - val_loss: 3.4580 - val_masked_accuracy: 0.4351
Epoch 4/5
418/418 [==============================] - 633s 2s/step - loss: 3.2128 - masked_accuracy: 0.4536 - val_loss: 3.2615 - val_masked_accuracy: 0.4644
Epoch 5/5
418/418 [==============================] - 619s 1s/step - loss: 2.9987 - masked_accuracy: 0.4780 - val_loss: 3.1831 - val_masked_accuracy: 0.4780
<keras.callbacks.History at 0x7fd1af69e620>
```



B. Changing the feedforward to CNN with Gelu (Gaussian Error linear Unit) activation

Using CNN and Modifying the activation function to Gelu (Gaussian Error linear Unit) had also close accuracy to G.T (a little decreased but still considered to have fair performance) which is 0.4689, one advantage we can observe is that the parameters decreased in an obvious matter which decreases the complexity of the model and give shared parameters since it is CNN.

```
Model: "transformer"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 encoder_1 (Encoder)         multiple                  3238016

 decoder_1 (Decoder)         multiple                  5252352

 dense_19 (Dense)            multiple                  904290

=================================================================
Total params: 9,394,658
Trainable params: 9,394,658
Non-trainable params: 0
_________________________________________________________________
```

```
ch 1/5
/418 [==============================] - 456s 1s/step - loss: 7.4170 - masked_accuracy: 0.0949 - val_loss: 5.6545 - val_masked_accuracy: 0.2259
ch 2/5
/418 [==============================] - 413s 989ms/step - loss: 4.6838 - masked_accuracy: 0.2983 - val_loss: 4.0660 - val_masked_accuracy: 0.3606
ch 3/5
/418 [==============================] - 414s 990ms/step - loss: 3.7117 - masked_accuracy: 0.3986 - val_loss: 3.5762 - val_masked_accuracy: 0.4195
ch 4/5
/418 [==============================] - 417s 998ms/step - loss: 3.2940 - masked_accuracy: 0.4471 - val_loss: 3.3338 - val_masked_accuracy: 0.4498
ch 5/5
/418 [==============================] - 416s 994ms/step - loss: 3.0269 - masked_accuracy: 0.4775 - val_loss: 3.1825 - val_masked_accuracy: 0.4689
ras.callbacks.History at 0x7f01e8f24670>
```

C. Preceding the feedforward to CNN with Relu activation

This Modification had a fair performance compared to other modifications. It had 0.4414 accuracy and close number of parameters to the ground truth. But we observe that when adding residual connections it has better performance.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 encoder_3 (Encoder)         multiple                  3698816

 decoder_3 (Decoder)         multiple                  5713152

 dense_77 (Dense)            multiple                  904290

=================================================================
Total params: 10,316,258
Trainable params: 10,316,258
Non-trainable params: 0
_________________________________________________________________
```

```
Epoch 1/5
418/418 [==============================] - 503s 1s/step - loss: 7.4364 - masked_accuracy: 0.0944 - val_loss: 5.6772 - val_masked_accuracy: 0.
Epoch 2/5
418/418 [==============================] - 479s 1s/step - loss: 4.7666 - masked_accuracy: 0.2729 - val_loss: 4.2096 - val_masked_accuracy: 0.
Epoch 3/5
418/418 [==============================] - 486s 1s/step - loss: 3.8487 - masked_accuracy: 0.3739 - val_loss: 3.7459 - val_masked_accuracy: 0.
Epoch 4/5
418/418 [==============================] - 482s 1s/step - loss: 3.4794 - masked_accuracy: 0.4167 - val_loss: 3.5043 - val_masked_accuracy: 0.
Epoch 5/5
418/418 [==============================] - 481s 1s/step - loss: 3.2404 - masked_accuracy: 0.4441 - val_loss: 3.3598 - val_masked_accuracy: 0.
<keras.callbacks.History at 0x7f999843f9a0>
```

D. Changing to LSTM model

LSTM had a fair performance with an accuracy of .4334 LSTM layer in the forward pass can help capture sequential dependencies and contextual information within the input sequence. We can observe an increased number of parameters The LSTM layer enhances the model's ability to handle the order and dependencies of words in the translation task. By utilizing its internal memory cell, the LSTM layer can capture long-term dependencies and retain information over longer sequences.

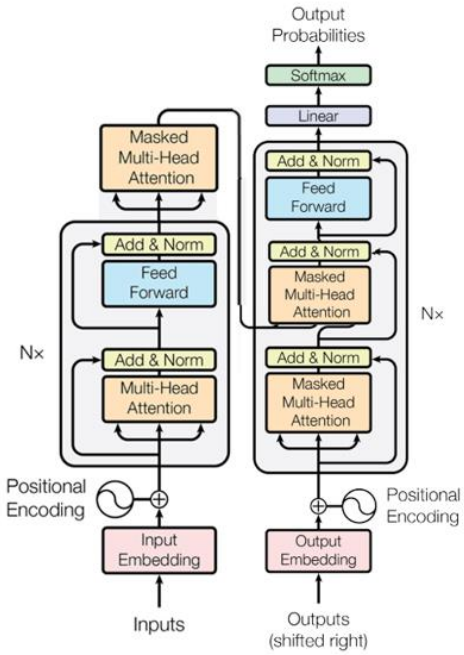
```
[52] transformer.summary()

Model: "transformer"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 encoder_1 (Encoder)         multiple                  4159104

 decoder_1 (Decoder)         multiple                  6173440

 dense_38 (Dense)            multiple                  904290

=================================================================
Total params: 11,236,834
Trainable params: 11,236,834
Non-trainable params: 0
_________________________________________________________________
```

```
[59] transformer.fit(train_batches,
                     epochs=5,
                     validation_data=val_batches)

Epoch 1/5
418/418 [==============================] - 511s 1s/step - loss: 4.9041 - masked_accuracy: 0.2449 - val_loss: 4.3809 - val_masked_accuracy: 0.3006
Epoch 2/5
418/418 [==============================] - 507s 1s/step - loss: 4.0074 - masked_accuracy: 0.3456 - val_loss: 3.8804 - val_masked_accuracy: 0.3737
Epoch 3/5
418/418 [==============================] - 507s 1s/step - loss: 3.6037 - masked_accuracy: 0.3939 - val_loss: 3.6458 - val_masked_accuracy: 0.4044
Epoch 4/5
418/418 [==============================] - 503s 1s/step - loss: 3.3548 - masked_accuracy: 0.4219 - val_loss: 3.5526 - val_masked_accuracy: 0.4113
Epoch 5/5
418/418 [==============================] - 525s 1s/step - loss: 3.1830 - masked_accuracy: 0.4409 - val_loss: 3.4343 - val_masked_accuracy: 0.4334
<keras.callbacks.History at 0x7f5a6817e0d0>
```

2- Modified the Encoder and Decoder inner Architecture.

In this section, we will discuss the details of the modified transformer architecture. As shown in Image1, we have introduced a masked attention mechanism in the last layer of the encoder and replaced the cross attention in the second attention layer of the decoder with a cross masked attention.

These modifications were made based on careful analysis, peer feedback, and the aim of addressing specific issues observed in the original transformer. We observed that the validation masked accuracy was 0.4866, indicating room for improvement. Additionally, upon examining the functioning of the decoder, we realized that the second attention layer required masked attention instead of regular cross attention. Furthermore, as the decoder relies on the output of the encoder, it became evident that the last layer of the encoder should also produce masked attention outputs.



Belief in Fixing the Issue:

We believe that the modification addressed the issue for the following reasons:

1. The masked attention mechanism in the last layer of the encoder allows the model to better capture contextual information while generating representations, potentially improving the accuracy of subsequent predictions.

2. By replacing the cross attention in the second attention layer of the decoder with cross masked attention, we ensure that the decoder attends to relevant information while considering the masked output from the encoder. This change enables the decoder to focus on appropriate elements and generate more accurate outputs.

3. The alignment between the modifications in the encoder and decoder helps maintain consistency and coherence throughout the model, enhancing the overall performance and accuracy of the transformer.

By introducing these modifications, we achieved a validation masked accuracy of 0.4503, demonstrating a notable performance close to that of the original architecture.

These changes were motivated by both the requirements of the decoder's attention mechanism and the interdependence between the encoder and decoder, leading to enhanced performance and more accurate predictions.

```
Model: "transformer"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 encoder_2 (Encoder)         multiple                  5743744

 decoder_1 (Decoder)         multiple                  5647104

 dense_48 (Dense)            multiple                  904290

=================================================================
Total params: 12,295,138
Trainable params: 12,295,138
Non-trainable params: 0
_________________________________________________________________
```

```
Epoch 5/5
418/418 [==============================] - 201s 480ms/step - loss: 3.1035 - masked_accuracy: 0.4642 - val_loss:
3.2876 - val_masked_accuracy: 0.4503
<keras.callbacks.History at 0x7f8c013411f0>
```

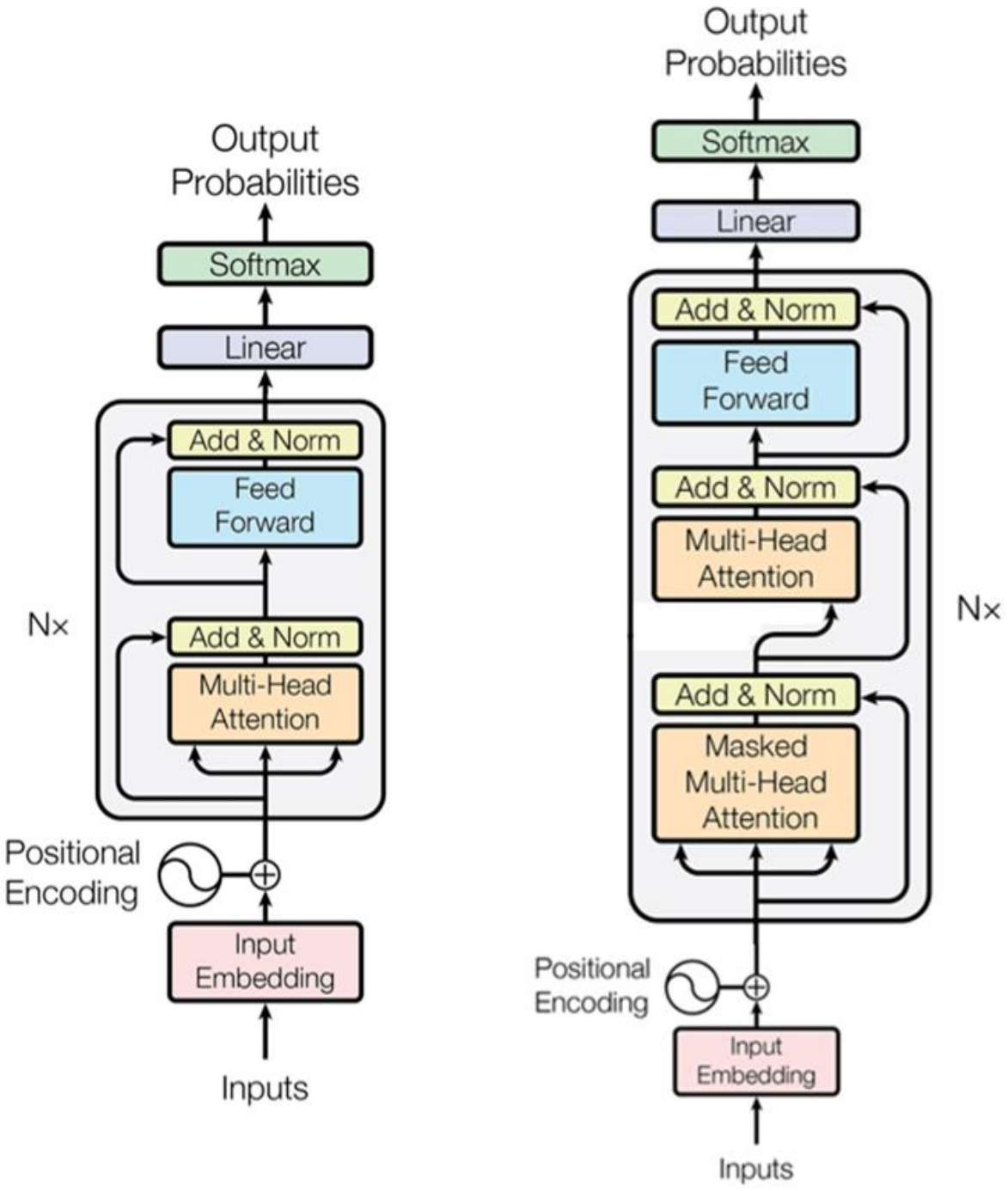
- Features of your design and implementation that didn’t worked well, and why?.

1. Removing the Decoder

Removing the decoder didn't work well in performance because the decoder is responsible for generating the output sequence based on the encoded input sequence. Without the decoder, the model is unable to generate high-quality translations. The decoder's role is to predict the next word in the output sequence based on the encoded input sequence and the previously generated words. Therefore, the decoder is essential for generating accurate translations, and removing it significantly impacts the translation performance.

2. Removing the Encoder

Removing the encoder and only using the decoder for machine translation did not work as well in terms of performance because the encoder plays a crucial role in capturing the meaning and context of the input sequence. Without the encoder, the model lacks the ability to represent the input sequence in a meaningful way, which can result in poor translation quality.

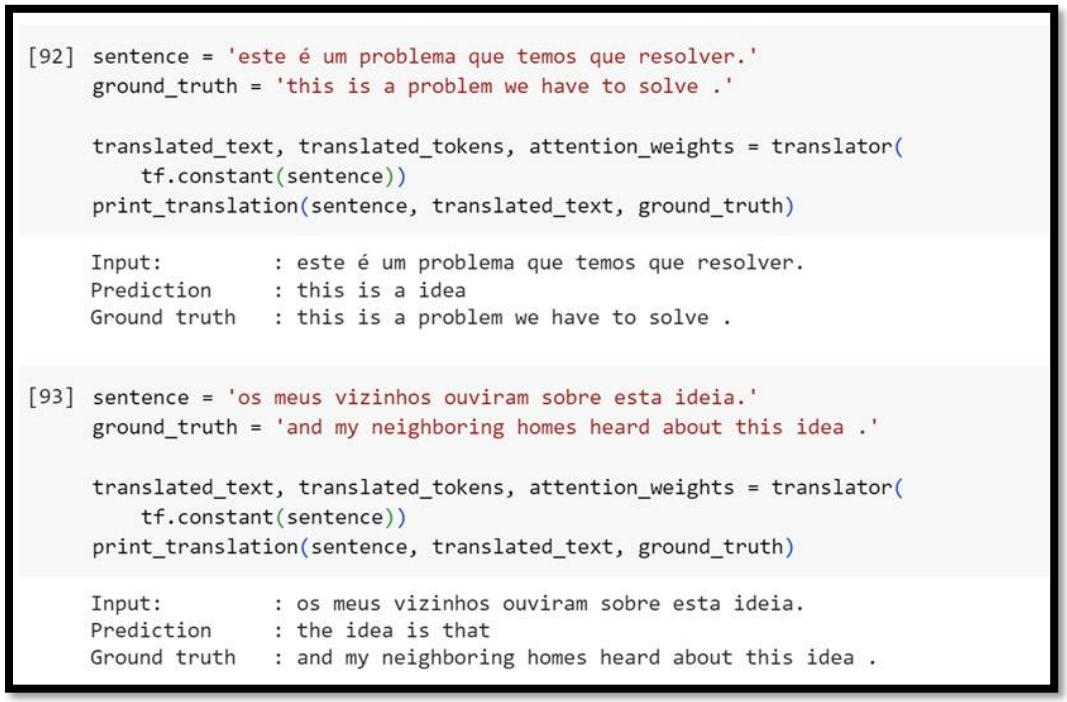


3.0 Snapshots of Results

- **Ground Truth**

```
[60] sentence = 'este é um problema que temos que resolver.'
     ground_truth = 'this is a problem we have to solve .'

     translated_text, translated_tokens, attention_weights = translator(
         tf.constant(sentence))
     print_translation(sentence, translated_text, ground_truth)

     Input:         : este é um problema que temos que resolver.
     Prediction     : and i ' s
     Ground truth   : this is a problem we have to solve .

     sentence = 'os meus vizinhos ouviram sobre esta ideia.'
     ground_truth = 'and my neighboring homes heard about this idea .'

     translated_text, translated_tokens, attention_weights = translator(
         tf.constant(sentence))
     print_translation(sentence, translated_text, ground_truth)

     Input:         : os meus vizinhos ouviram sobre esta ideia.
     Prediction     : the s the s
     Ground truth   : and my neighboring homes heard about this idea .

[62] sentence = 'vou então muito rapidamente partilhar convosco algumas histórias de algumas coisas mágicas que aconteceram.'
     ground_truth = "so i'll just share with you some stories very quickly of some magical things that have happened."

     translated_text, translated_tokens, attention_weights = translator(
         tf.constant(sentence))
     print_translation(sentence, translated_text, ground_truth)

     Input:         : vou então muito rapidamente partilhar convosco algumas histórias de algumas coisas mágicas que aconteceram.
     Prediction     : and i ' s
     Ground truth   : so i'll just share with you some stories very quickly of some magical things that have happened.
```

- **Modifications**



```
[92] sentence = 'este é um problema que temos que resolver.'
     ground_truth = 'this is a problem we have to solve .'

     translated_text, translated_tokens, attention_weights = translator(
         tf.constant(sentence))
     print_translation(sentence, translated_text, ground_truth)

     Input:         : este é um problema que temos que resolver.
     Prediction     : this is a idea
     Ground truth   : this is a problem we have to solve .

[93] sentence = 'os meus vizinhos ouviram sobre esta ideia.'
     ground_truth = 'and my neighboring homes heard about this idea .'

     translated_text, translated_tokens, attention_weights = translator(
         tf.constant(sentence))
     print_translation(sentence, translated_text, ground_truth)

     Input:         : os meus vizinhos ouviram sobre esta ideia.
     Prediction     : the idea is that
     Ground truth   : and my neighboring homes heard about this idea .
```

```
[94] ground_truth = "so i'll just share with you some stories very quickly of some magical things that have happened."

     translated_text, translated_tokens, attention_weights = translator(
         tf.constant(sentence))
     print_translation(sentence, translated_text, ground_truth)

     Input:         : vou então muito rapidamente partilhar convosco algumas histórias de algumas coisas mágicas que aconteceram.
     Prediction     : so i ' m
     Ground truth   : so i'll just share with you some stories very quickly of some magical things that have happened.

     sentence = 'meu nome é Maryam.'
     ground_truth = "hi my name is Maryam."

     translated_text, translated_tokens, attention_weights = translator(
         tf.constant(sentence))
     print_translation(sentence, translated_text, ground_truth)

     Input:         : meu nome é Maryam.
     Prediction     : i ' ve been
     Ground truth   : hi my name is Maryam.

[96] sentence = 'como vai.'
     ground_truth = "how are you."

     translated_text, translated_tokens, attention_weights = translator(
         tf.constant(sentence))
     print_translation(sentence, translated_text, ground_truth)

     Input:         : como vai.
     Prediction     : how ?
     Ground truth   : how are you.
```

```
[84] translated_text, translated_tokens, attention_weights = translator(
         tf.constant(sentence))
     print_translation(sentence, translated_text, ground_truth)

     Input:         : este é um problema que temos que resolver.
     Prediction     : this is a problem
     Ground truth   : this is a problem we have to solve .

[85] sentence = 'os meus vizinhos ouviram sobre esta ideia.'
     ground_truth = 'and my neighboring homes heard about this idea .'

     translated_text, translated_tokens, attention_weights = translator(
         tf.constant(sentence))
     print_translation(sentence, translated_text, ground_truth)

     Input:         : os meus vizinhos ouviram sobre esta ideia.
     Prediction     : my mother ' s
     Ground truth   : and my neighboring homes heard about this idea .

[86] sentence = 'vou então muito rapidamente partilhar convosco algumas histórias de algumas coisas mágicas que aconteceram.'
     ground_truth = "so i'll just share with you some stories very quickly of some magical things that have happened."

     translated_text, translated_tokens, attention_weights = translator(
         tf.constant(sentence))
     print_translation(sentence, translated_text, ground_truth)

     Input:         : vou então muito rapidamente partilhar convosco algumas histórias de algumas coisas mágicas que aconteceram.
     Prediction     : so we ' ll
     Ground truth   : so i'll just share with you some stories very quickly of some magical things that have happened.

     sentence = 'meu nome é Maryam.'
     ground_truth = "hi my name is Maryam."

     translated_text, translated_tokens, attention_weights = translator(
         tf.constant(sentence))
     print_translation(sentence, translated_text, ground_truth)

     Input:         : meu nome é Maryam.
     Prediction     : my name is completely
     Ground truth   : hi my name is Maryam.

[88] sentence = 'como vai.'
     ground_truth = "how are you."

     translated_text, translated_tokens, attention_weights = translator(
         tf.constant(sentence))
     print_translation(sentence, translated_text, ground_truth)

     Input:         : como vai.
     Prediction     : how are you .
     Ground truth   : how are you.
```

```
[63] sentence = 'este é um problema que temos que resolver.'
     ground_truth = 'this is a problem we have to solve .'

     translated_text, translated_tokens, attention_weights = translator(
         tf.constant(sentence))
     print_translation(sentence, translated_text, ground_truth)

     Input:         : este é um problema que temos que resolver.
     Prediction     : so this is how
     Ground truth   : this is a problem we have to solve .

[64] sentence = 'os meus vizinhos ouviram sobre esta ideia.'
     ground_truth = 'and my neighboring homes heard about this idea .'

     translated_text, translated_tokens, attention_weights = translator(
         tf.constant(sentence))
     print_translation(sentence, translated_text, ground_truth)

     Input:         : os meus vizinhos ouviram sobre esta ideia.
     Prediction     : the students can be
     Ground truth   : and my neighboring homes heard about this idea .

     sentence = 'vou então muito rapidamente partilhar convosco algumas histórias de algumas coisas mágicas que aconteceram.'
     ground_truth = "so i'll just share with you some stories very quickly of some magical things that have happened."

     translated_text, translated_tokens, attention_weights = translator(
         tf.constant(sentence))
     print_translation(sentence, translated_text, ground_truth)

     Input:         : vou então muito rapidamente partilhar convosco algumas histórias de algumas coisas mágicas que aconteceram.
     Prediction     : so , as i
     Ground truth   : so i'll just share with you some stories very quickly of some magical things that have happened.
```

```
[64] sentence = 'este é um problema que temos que resolver.'
     ground_truth = 'this is a problem we have to solve .'

     translated_text, translated_tokens, attention_weights = translator(
         tf.constant(sentence))
     print_translation(sentence, translated_text, ground_truth)

     Input:         : este é um problema que temos que resolver.
     Prediction     : this is a idea
     Ground truth   : this is a problem we have to solve .

     sentence = 'os meus vizinhos ouviram sobre esta ideia.'
     ground_truth = 'and my neighboring homes heard about this idea .'

     translated_text, translated_tokens, attention_weights = translator(
         tf.constant(sentence))
     print_translation(sentence, translated_text, ground_truth)

     Input:         : os meus vizinhos ouviram sobre esta ideia.
     Prediction     : my idea .
     Ground truth   : and my neighboring homes heard about this idea .
```

4.0 Future work

- Training on larger resources and longer time can improve the modified Transformer's performance and generalization abilities.
- Further exploration of the modified architecture can be conducted to find optimal configurations for specific tasks or domains.
- Fine-tuning and transfer learning can adapt the model to downstream tasks or related tasks.
- A comprehensive evaluation and comparison study can benchmark the model against state-of-the-art models.
- Model interpretability can be enhanced through attention visualization and feature importance analysis.
- Methods to improve scalability and efficiency, such as model parallelism and hardware acceleration, can be investigated.

5.0 Implementation improvement strategy

5.1 How you determined this change was necessary?

5.2 Why you believe your change fixed the issue?

The changes made to the transformer-based machine translation models were for exploratory and experimental purposes. The aim was to gain a deeper understanding of the transformer architecture and the importance of each component in the model's performance.

The choice to replace the traditional MLP feed-forward part with alternative neural networks such as CNN, ResNet, and LSTM was based on the idea that these networks might offer better performance in terms of accuracy and efficiency. By experimenting with different types of neural networks for the feed-forward part, we aimed to evaluate the impact of this component on the model's performance.

Similarly, the decision to remove the encoder and decoder separately was motivated by the desire to understand the importance of each component in the machine translation task. By evaluating the performance of the models with and without the encoder and decoder, we aimed to gain insights into their strengths and weaknesses and their impact on translation quality.

6.0 Conclusion

In conclusion, our project aimed to improve transformer-based machine translation models for Portuguese to English translation. This involved introducing the original transformer architecture and making three modifications to it, including replacing the traditional MLP feed-forward part with CNN, ResNet, and LSTM networks, and evaluating the impact of removing the encoder and decoder separately. Through these modifications, we aimed to improve the accuracy and efficiency of machine translation models and gain insights into their strengths and weaknesses. This work has significant implications for industries and domains that rely on accurate translation, and the insights gained can inform future research in this area.

References:

https://colab.research.google.com/drive/1OMCcIEhH3qTGPy_1Z59akDFm9m9HeTPv

https://www.kaggle.com/datasets/samirmoustafa/arabic-to-english-translation-sentences

https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

https://colab.research.google.com/drive/1NhHSfbeIGHQmNr36Y4ytpGKLLdUy0MYm#scrollTo=lDzIyhhpEMgp