Ministry of Higher Education and Research
Higher School of Computer Science 08 May 1945 - Sidi Bel Abbes

Second Year Second Cycle - Artificial Intelligence and Data Science

Project Report

AUTOMATIC IMAGE CAPTIONING WITH TRANSFORMERS

Students:

• Abdelnour FELLAH
• Abderrahmene BENOUNENE
• Adel Abdelkader MOKADEM
• Meriem MEKKI
• Yacine Lazreg BENYAMINA

Supervisor:

Dr. Rabab Bousmaha

Release date: May 20, 2024

Table of contents

List of Figures

List of Tables

1 Introduction

Automatic image captioning is a multidisciplinary task that combines computer vision and natural language processing techniques to construct deep learning systems capable of generating textual descriptions for images. The significance of solving this task lies in its potential to cater to individuals with impaired vision. An effective AI Image Captioning System can be -for example- integrated into websites and mobile applications to enhance accessibility.

The problem of automatically generating a caption for a given image is considered an end-to-end sequence-to-sequence problem. This is because it takes an image, which is a sequence of pixels, and produces a caption, which is a sequence of tokens, without the need for manual feature extraction. The model processes the entire image to produce the entire output sequence, with minimal pre- or post-processing required.

In this report, we will elaborate on our approach using an Encoder-Decoder architecture based on transformers and vision transformers to construct a model capable of generating captions for images. The source code for the project can be accessed at: https://github.com/Devnetly/image-captioning.

2 Encoder-Decoder Architecture

An Encoder-Decoder is a type of neural network architecture used in sequence-to-sequence learning. It comprises two main components: an **encoder** and a **decoder**. This architecture enables various tasks, including machine translation, text-to-speech, speech-to-text, and image captioning.

The encoder part takes the input sequence,process it and produces a context vector/or a set of context vectors which are then fed to **The decoder** part that which takes the the encoder's output and tries to construct the output sequence, the encoder and decoder parts are task-dependent for example in the case of machine translation both the encoder and decoder parts are RNN-based networks.

Attention mechanisms are usually used with Encoder-Decoder architectures to enhance the model's ability to capture complex relationships between the input and output sequences.

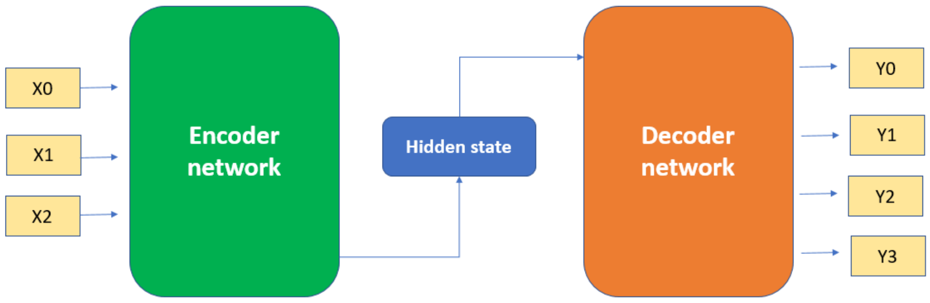


Figure 1: Encoder-Decoder Architecture, source.

3 Transformers

Transformers, introduced in 2017 by Google in their paper Attention is All You Need [3] are a **encoder-decoder** architecture designed for sequence-to-sequence tasks. They leverage multi-head attention mechanisms to handle long-range dependencies more effectively than previous models. Transformers have become foundational in natural language processing, excelling in tasks such as machine translation, text generation, and language understanding.

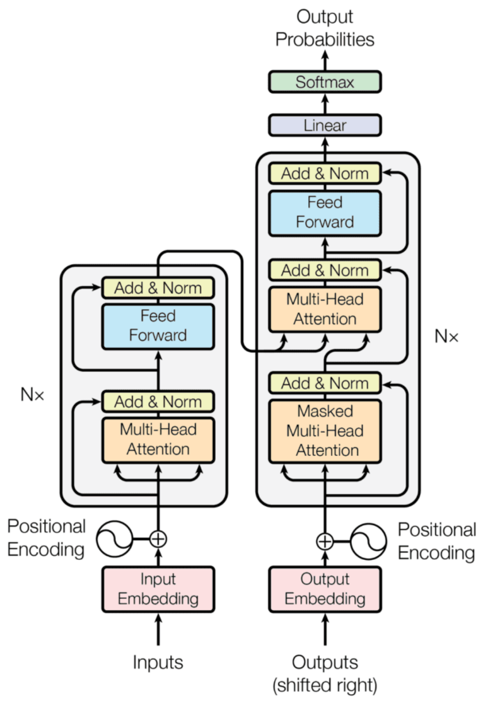


Figure 2: The transformer architecture, source.

A transformer consists of N encoder and decoder layers. The first encoder layer takes as input the token embeddings with positional encoding, while subsequent encoder layers operate on the output of the previous encoder.

The first decoder layer takes as input the token embeddings with positional encoding of the output sequence as well as the output of the last encoder layer, while subsequent decoder layers operate on the output of the previous decoder layer as well as the output of the last encoder layer.

The outputs of the last decoder layer are then passed through a linear layer to map them to the dimension of the output sequence (the size of the vocabulary of the output sequences). Finally, softmax is applied to generate probabilities.

3.1 Embedding Layer

An Embedding Layer is essentially used to represent categorical/discrete features as continuous vectors. In NLP, embedding layers are used to represent each token in the vocabulary with a vector. The weight of an embedding layer is a matrix of dimension $V \times D$, where V is the size of the vocabulary and D is the dimension of the vector. Therefore, an embedding layer takes a sequence of length S and produces a matrix of dimensions $S \times D$.

The weight matrix is either initialized randomly and updated through backpropagation, or initialized with precomputed vectors such as GloVe, or a combination of both.

3.2 Positional Encoding

Since transformers process the entire input sequence at the same time unlike recurrent units a form of ordering is needed in the architecture as it is clear that meaning of the sequences can change by changing the order of its elements.

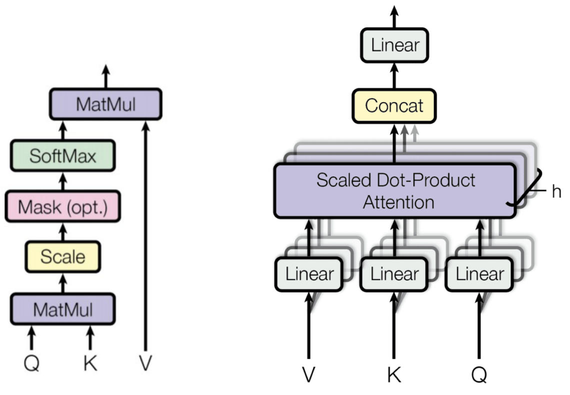
A positional encoding layer maps each unique index to a vector of dimension D,so the weight of a positional encoding layer is a $S \times D$ matrix W where S is the size length of the input sequence and D is the dimension of the embedding vectors,it takes an input x of dimension $S \times D$,and produces the output $y = x + W$.

The weights of the positional encoding layer are not learnable and remain fixed; they are not updated during backpropagation, and they are initialized using the following formula :

$$\begin{cases} W_{i,2*j} = sin(\frac{i}{N^{\frac{2*j}{D}}}) \\ W_{i,2*j+1} = cos(\frac{i}{N^{\frac{2*j}{D}}}) \end{cases}$$

The value of N is usually set to 10000.

3.3 Multi-Head Attention & Masked Multi-Head Attention



(a) Scaled-Dot-Product (b) Multi-Head-Attention

Figure 3: Multi-Head-Attention Mechanism

3.3.1 Self-Attention

Value,Key and Query :

A self-attention (or single-headed attention) layer is mainly composed of three linear layers called Value, Key, and Query, respectively. Their weights are all of dimension $D \times D_{\text{att}}$. Each layer is used to map the values of the input matrix (token embeddings + positional encoding) and/or change its dimension, meaning that each layer outputs a matrix of dimensions $S \times D_{\text{att}}$,lets call the resulted matrices V_{output}, K_{output} and Q_{output} respectively .

Scaled Dot-Product :

Then in **MatMul** stage the matrices V_{output}, K_{output} are multiplied,$y = Q_{output} \times V_{output}$, resulting in a $S \times S$ matrix that is often called **Attention Filter** which can be viewed as the attention that each token in the sequence is paying to other words in the sequence.

The attention scores are then scaled by dividing them by $\sqrt{S}$. This scaling is applied to prevent the issue of vanishing gradients. The scaled scores are then passed through a softmax layer to ensure that the values in each row of the attention matrix sum up to 1.

The attention Filter is then multiplied by the V_{output} matrix to produce a matrix of dimension $S \times D_{att}$.

$$ScaledDotProduct(Q, K, V) = softmax(\frac{Q \times K^T}{\sqrt{S}}) \times V$$

3.3.2 Multi-Head Attention

Unlike self-attention,multi-head attention learns multiple filters instead of one using the same mechanism described previously,so it can be viewed to a h self-attention layers with an output dimension D_{att}/h where h is the number of heads.

The outputs of the Scaled Dot-Product stage of each head are then concatenated resulting in an $S \times D\_att$ matrix,this matrix is then finally passed to fully connected layer which maps the values and changes input dimension back to $S \times D$.

3.3.3 Masked Multi-Head Attention

You may have noticed that in the figure above, a mask stage can be optionally performed after the Scale stage. The role of this stage is to prevent the model from looking into the future (i.e., paying attention to next tokens in the sequence) by masking (setting to zero) the upper triangular part of the attention filter.

3.4 Add & Norm Layer

3.4.1 Add

The add layer (also known ad skip-connection or residual-connection) preforms element-wise addition between the input and the output of the previous layer,which helps preventing the issue of vanishing gradient descent and allows for a better information flow.

3.4.2 Norm

After the addition operation, layer normalization is applied. Layer normalization normalizes the values across the feature dimension for each example in the mini-batch. This helps in stabilizing the learning process and improving generalization.

4 Vision Transformers

Vision transformers are encoder-only transformers adapted for computer vision tasks. The idea is to break down an image of size $W \times H$ into a series of patches of size $P_w \times P_h$. These patches are then flattened, resulting in a two-dimensional matrix of dimensions $S \times D$, where $S = \frac{W}{P_w} \times \frac{H}{P_h}$ and $D = P_w \times P_h$. A linear projection layer is used to transform the individual flattened patches to a lower-dimensional vector, resulting in a matrix of dimension $S \times D'$. Finally, positional embedding is applied.

The output of patching and positional embedding is then fed to a regular transformer encoder,the outputs of the encoder are then passed to a Multi-Layer Perception to output the classes probabilities.

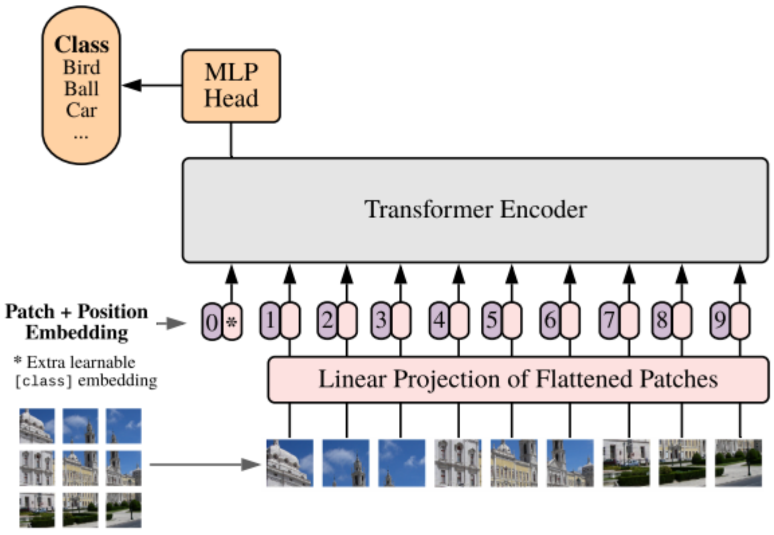

Class
Bird
Ball
Car
...
MLP Head
Transformer Encoder
Patch + Position Embedding
* Extra learnable [class] embedding
0 * 1 2 3 4 5 6 7 8 9
Linear Projection of Flattened Patches

Figure 4: Vision Transformer, source.

5 DIET : DATA EFFICIENT IMAGE TRANSFORMER

DIET, introduced in 2021 by Facebook AI, stands for Data Efficient Image Transformer. It enhances the classic Vision Transformer to reduce the requirement for vast amounts of training data and the computational requirements of the model using A **teacher-student** with **distillation** token, the student model is first trained to mimic the teacher's behaviour then further fine-tuned on the original task using the ground truth labels.

5.1 SOFT DISTILLATION

$$\mathcal{L} = (1-\lambda) * \mathcal{L}_{ce}(f(Z_s), y) + \lambda * \tau^2 * KL(f(\frac{Z_t}{\tau}), f(\frac{Z_s}{\tau}))$$

where :

- Z_s : the student's logits.
- Z_t : the teacher's logits.
- λ : coefficient balancing KL divergence and cross entropy loss,between 0 and 1.
- τ : The temperature (Distillation).
- $\mathcal{L}_{ce}$: The cross entropy loss function.
- f : the student.

5.2 HARD DISTILLATION

$$\mathcal{L} = \frac{1}{2}\mathcal{L}_{ce}(f(Z_s), y) + \frac{1}{2}\mathcal{L}_{ce}(f(Z_s), y_t)$$

where :

- y_t : the tachers's predictions.
- y : the real labels.

5.3 DISTILLATION TOKEN

A special token called distillation token is added,and it is learnable through back propagation,it represents and capture the knowledge transferred from the larger teacher model during the knowledge distillation process.

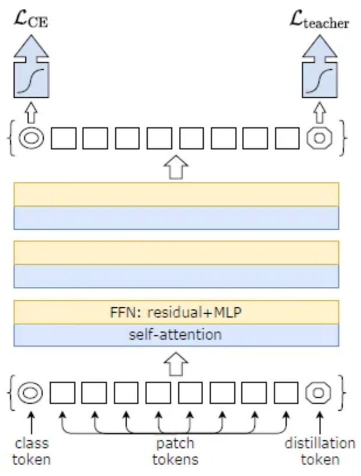


Figure 5: Distillation Process, source.

6 Architecture

For this task, we chose an Encoder-Decoder architecture with a **DeiT-III** (with the last classification layer) in the encoder part and a Transformer decoder in the decoder part, with an additional fully connected layer to output class probabilities.

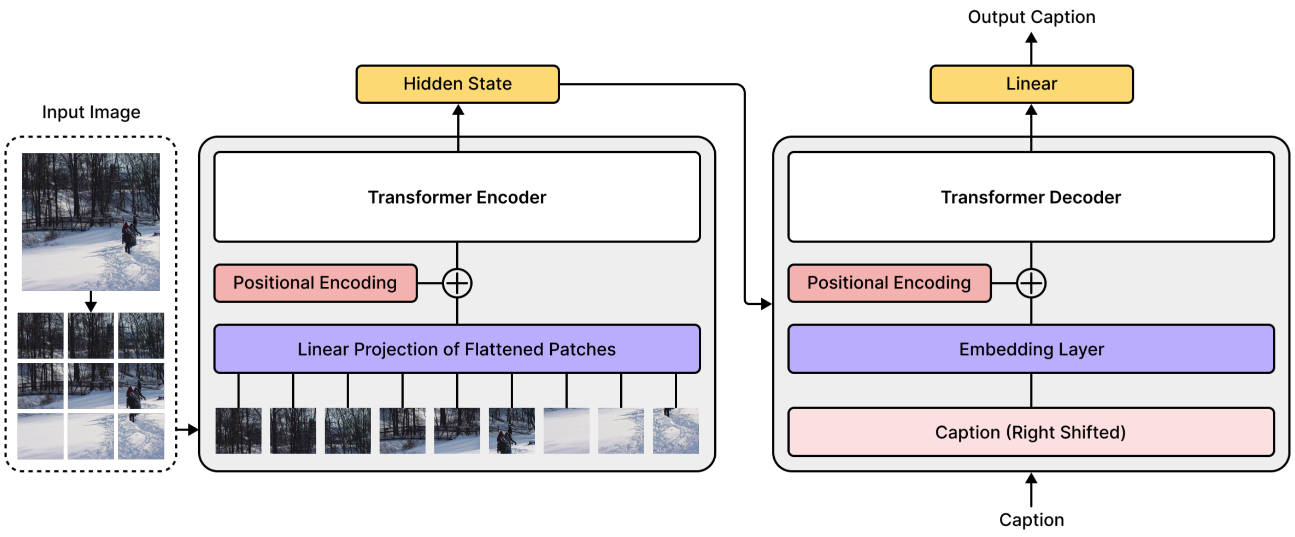


Figure 6: The model's architecture.

7 Dataset

The dataset used to train the model is the Flickr30k dataset with **31,783** images of various resolutions and five captions per image, totaling **158,915** image-caption pairs. The dataset size is **4.39GB** and primarily consists of images depicting people involved in everyday activities and events,the dataset does not provide any predefined split, so we divided it into a training and a test set with ratios of 0.8 and 0.2, respectively. while ensuring that all captions for the same image are in the same set.

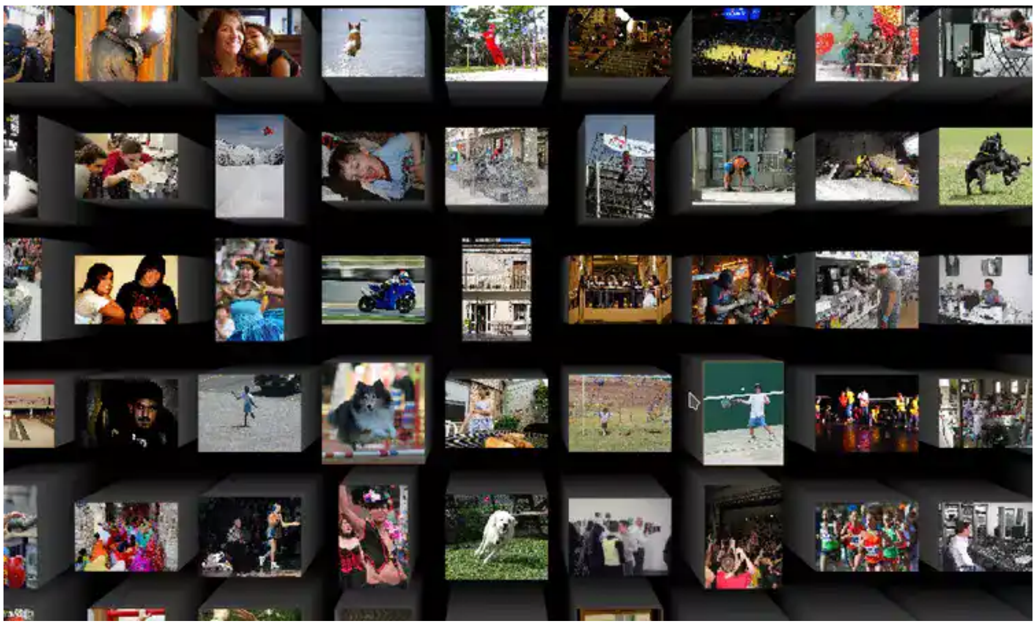


Figure 7: The Flickr30k Dataset, source.

8 Preprocessing & Data Augmentation

8.1 Images Preprocessing & Augmentation

At training, the input images were randomly horizontally flipped, and then a random sub-image of size 384×348 pixels was randomly selected. The pixel values were divided by 255 as a form of normalization. At inference time, the images were all resized to a uniform resolution of 384×384 pixels, and the pixel values were divided by 255.

8.2 Captions Preprocessing

Before feeding the captions to the transformer, three steps of text normalization are applied: lowercasing, punctuation removal, and replacing multiple spaces with one space. The normalized captions are then tokenized using space-and-punctuation-based tokenization. Each token is replaced by its index in the vocabulary. To reduce the size of the vocabulary, a token has to appear in at least n documents to be added to the vocabulary (in this case, we chose n to be equal to 8). If a token is not found in the vocabulary, it will be replaced by the index of a special token called the unknown token. Additionally, special tokens representing the start and end of each sentence are appended to the beginning and end of each caption. Finally, the sequences are padded to all have the same length of 80.

The input sequences fed to the model are right-shifted (the last token in the sequence is removed). Meanwhile, the model is expected to predict the left-shifted caption (without the first token). This approach encourages the model to predict the next word based on the preceding words and the input image.

9 Metrics & Evaluation

9.1 Automatic evaluation

9.1.1 Perplexity

the perplexity metric in NLP quantifies the level of uncertainty a model exhibits when predicting or assigning probabilities to text, the lower the perplexity, the better the model is, The perplexity of the model q on a test sample $x_1, x_2, ..., x_N$ is given.

$$PPL(X) = (\prod_{i=1}^{N} q(x_i))^{-\frac{1}{N}}$$

9.1.2 BLUE score (Bilingual Evaluation Understudy)

BLUE (bilingual evaluation understudy) is a metric used to evaluate the generated text of machine translation models or models that are trying to solve similary task ,it was invented in 2001 at IBM,it is known for having a high correlation with human annotations.

The BLUE score compares separate parts of a text with a set of reference texts, and assigns a score to each part then these scores are averaged to give the final score, the higher the score, the better the model.

$$BLUEScore = BrevityPenalty * GeometricAveragePrecisionScore$$

Brevity Penalty :

The Brevity Penalty is included to prevent the score from giving a false impression of the model's performance by favoring very short sequences.

$$BrevityPenalty = \begin{cases} 1 \text{ if } c > r \\ e^{1-\frac{r}{c}} \text{ otherwise.} \end{cases}$$

Where c is the length of the predicted text and r is the length of the target text.

N-GRAM PRECISION SCORE :

$$precision_i = \frac{\sum\limits_{snt \in Condidates} \sum\limits_{i \in snt} min(m^i_{cand}, m^i_{ref})}{w^i_t = \sum\limits_{snt \in Condidates} \sum\limits_{i' \in snt} m^{i'}_{cand}}$$

where :

- m^i_{cand} : This number of i-grams in the translation corresponds to the translation of the reference.
- m^i_{ref} : the number of i-grams in the reference translation.
- w^i_t : The total number of i-grams in the automatic translation.

GEOMETRIC AVERAGE PRECISION SCORE :

$$GeometricAveragePrecisionScore = \prod_{i=1}^{N} p_i^{w_i}$$

Where p_i is the precision score of the predicted text with $N = i$,and w_i is its weight usually equal to $1/N$.

9.1.3 ROUGE score (Recall-Oriented Understudy for Gisting Evaluation)

ROUGE, or Recall-Oriented Understudy for Gisting Evaluation, is a metric used to measure the similarity between a machine-generated summary and reference summaries. It does so by comparing overlapping n-grams. ROUGE scores range from 0 to 1, with higher scores indicating a greater similarity between the automatically generated summary and the reference summaries.
Mainly three variants of the ROUGE Score exists : **ROUGE-1** which uses unigrams,**ROUGE-2** which uses bigrams and **ROUGE-L** which uses longest common subsequence (LCS) : for each score a precision,recall and an f1-score can be calculated using the following formulas :

$$\begin{cases} ROUGE - N_{recall} = \frac{|N-grams_{cand} \cap N-grams_{ref}|}{|N*grams_{ref}|} \\ ROUGE - N_{precision} = \frac{|N-grams_{cand} \cap N-grams_{ref}|}{|N-grams_{cand}|} \\ ROUGE - N_{F1} = 2 * \frac{ROUGE-N_{recall} * ROUGE-N_{precision}}{ROUGE-N_{recall} + ROUGE-N_{precision}} \end{cases}$$

$$\begin{cases} ROUGE - L_{recall} = \frac{LCS(cand,ref)}{|unigrams_{ref}|} \\ ROUGE - L_{recall} = \frac{LCS(cand,ref)}{|unigrams_{cand}|} \\ ROUGE - L_{F1} = 2 * \frac{ROUGE-N_{recall} * ROUGE-N_{precision}}{ROUGE-N_{recall} + ROUGE-N_{precision}} \end{cases}$$

9.2 Human Evaluation

In evaluation of translations, judges use a predefined scale, such as a 1 to 5 scale, where 1 represents the lowest and 5 the highest quality. However, defining clear descriptions for each score and distinguishing the levels of quality precisely can be challenging. Even with explicit guidelines, judges struggle to assign numerical values to translation quality. The evaluation focuses on two key metrics: adequacy and fluency.

10 Training & Results

The model's encoder was initialized using weights of pre-trained **DeiT-III** on the **ImageNet-22k** dataset,and we further fine-tuned it on our task instead of freezing it, while the decoder's weights were initialized randomly.

The model was trained for 11 epochs across three different sessions using the **AdamW** optimizer with learning rate of 10^{-4} and a weight decay of 10^{-4}, a batch size of 8 and a linear learning rate decay.

The training was done a NVIDIA GeForce RTX 3050 Ti Laptop GPU,and it took an average of 115 minutes per-epoch making the total training time 21 hours.

Hyper-parameter	**Value**
Optimizer	AdamW
Learning rate	10^{-4}
Weight Decay	10^{-4}
Learning rate deacy	Linear Learning rate decay
Batch size	8

Table 1: Hyper-parameters

Parameter	**Value**
Features Diemenions	256
Patch size	16
Heads count	8
Decoder Layer's count	6
Maximum Caption length	80

Table 2: The architecture's parameters.

10.1 Learning graphs

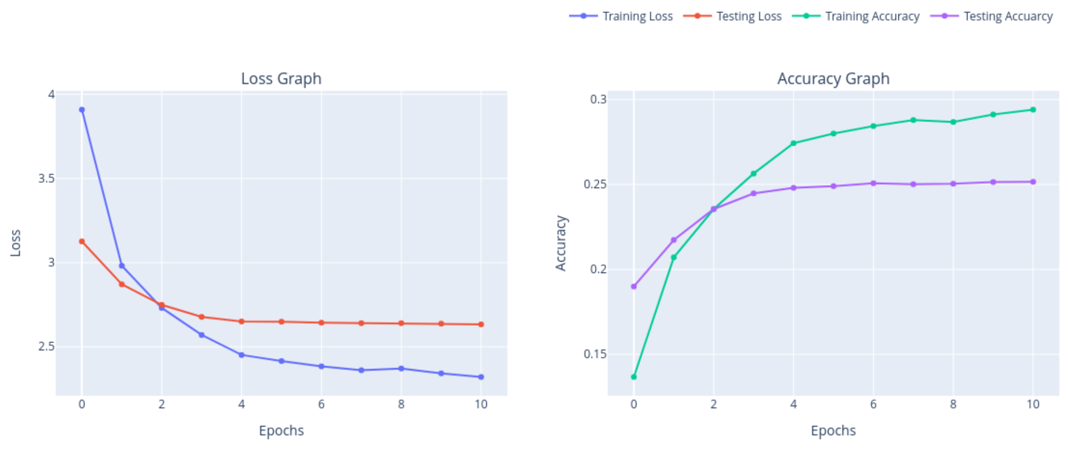


Figure 8: Loss Accuracy

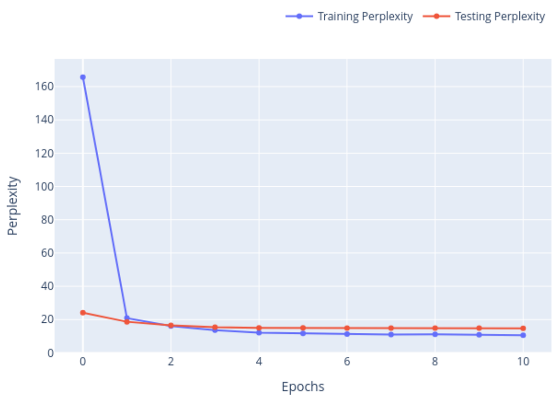


Figure 9: Perplexity

10.2 Automatic Evaluation Results

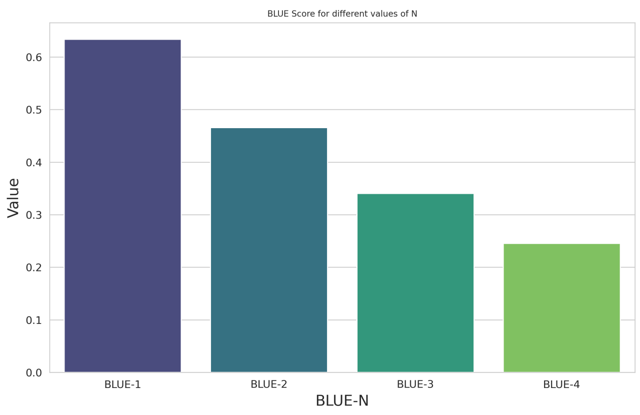
BLUE-1	BLUE-2	BLUE-3	BLUE-4
0.6335	0.4656	0.3403	0.2451

Table 3: BLUE Score

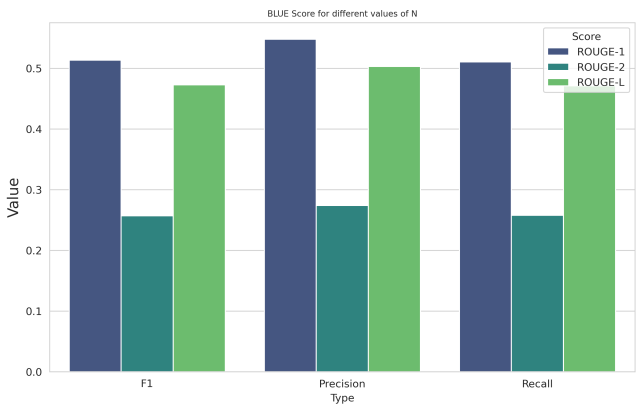
	ROUGE-1	ROUGE-2	ROUGE-L
F1	0.5135	0.2569	0.4728
Precision	0.5477	0.274	0.5029
Recall	0.5106	0.2576	0.4711

Table 4: ROUGE-Score

Table 5: Automatic Evaluation Results



(a) BLUE Score



(b) ROUGE Score

Figure 10: Automatic Evaluation Results

10.3 Testing on images outside of the dataset

To further test our model's ability to generalize on unseen data, we handpicked 20 images from the internet. These images contained similar objects to those in the training set but had different resolution ranges. Below are the results for some of them :

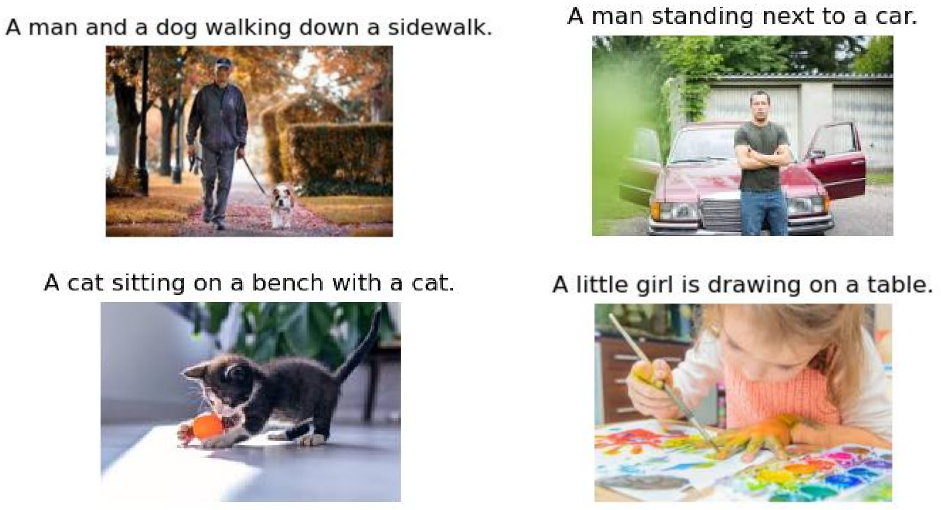

A man and a dog walking down a sidewalk.
A man standing next to a car.
A cat sitting on a bench with a cat.
A little girl is drawing on a table.

Figure 11: The Generated captions

11 Conclusion

In this project, we aimed to develop a model capable of generating captions for images using a transformer-based architecture. Our approach incorporated a Data-efficient Image Transformer (DeiT) as the encoder and a standard transformer decoder. We utilized the Flickr30k dataset for training and performed extensive text preprocessing alongside image resizing and augmentation to enhance the model's robustness.

Post-training evaluation was conducted using BLEU and ROUGE scores, and we also qualitatively assessed the model by generating captions for a variety of images randomly sourced from the internet. The results indicated that our model performed well on images similar to those in the training dataset, demonstrating its effectiveness in generating relevant captions. However, its performance declined on images that were significantly different from the training set, highlighting an area for improvement.

To further enhance the model's performance, several strategies can be considered. Expanding the dataset to include a more diverse set of images would likely improve the model's generalization capabilities. Experimenting with different model configurations and hyperparameters could also yield better results. Additionally, exploring advanced architectures, such as incorporating attention mechanisms or utilizing pre-trained models, may offer further improvements in caption generation quality.

Overall, while the current model shows promise, these suggested enhancements could significantly bolster its ability to generate accurate and contextually appropriate captions across a wider range of images.

12 References

[1] Taraneh Ghandi, Hamidreza Pourreza, and Hamidreza Mahyar. "Deep Learning Approaches on Image Captioning: A Review". In: *CoRR* abs/2201.12944 (2022). arXiv: `2201.12944`. URL: `https://arxiv.org/abs/2201.12944`.

[2] Hugo Touvron et al. "Training data-efficient image transformers & distillation through attention". In: *CoRR* abs/2012.12877 (2020). arXiv: `2012.12877`. URL: `https://arxiv.org/abs/2012.12877`.

[3] Ashish Vaswani et al. "Attention Is All You Need". In: *CoRR* abs/1706.03762 (2017). arXiv: `1706.03762`. URL: `http://arxiv.org/abs/1706.03762`.

[4] Peter Young et al. "From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions". In: *Transactions of the Association for Computational Linguistics* 2 (2014). Ed. by Dekang Lin, Michael Collins, and Lillian Lee, pp. 67–78. DOI: `10.1162/tacl_a_00166`. URL: `https://aclanthology.org/Q14-1006`.