Image Captioning

Abibi Aymane
Enseirb-Matmeca
aabibi@enseirb-matmeca.fr

Ahbaiz Mouhcine
Enseirb-Matmeca
mahbaiz@enseirb-matmeca.fr

Soufary Farouk
Enseirb-Matmeca
fsoufary@enseirb-matmeca.fr

Wardi Mohamed Amine
Enseirb-Matmeca
mwardi@enseirb-matmeca.fr

Abstract

This computer vision project focuses on creating and comparing two different solutions for automatically generating captions from images. The first method combines a YOLOv8-based Convolutional pre-trained Neural Network for extracting image features with a Long Short-Term Memory network for generating descriptive text. This CNN-LSTM approach aims to capture detailed visual information and produce coherent captions for input images.

The second solution explores the use of transformers in image captioning. It integrates a Vision Transformer (VIT) for image feature extraction (encoder) and a transformer model for text generation (decoder). By leveraging self-attention mechanisms, this transformer-based approach aims to capture both global context and dependencies between visual and textual elements.

Through experiments on the Flickr8k dataset, we aim to compare the performance of the CNN-LSTM and the Transformer-based model using the BLEU metric, the loss value and qualitative examples.

In conclusion, our observations indicate that the Transformer-based model outperforms the LSTM model. The Transformer model demonstrated the ability to effectively handle the entire dataset, showcasing its capacity to learn. In contrast, the LSTM-based model struggled with datasets exceeding 128 images, highlighting its limitations in managing larger datasets. Additionally, the Transformer-based model yielded compelling results not only on the test images but also on externally sourced images from the internet.

1. Introduction

The automated generation of detailed textual descriptions of images is a difficult challenge. The text generated must not only describe the objects or individuals present in the image, but must also respect English grammar, in order to guarantee global comprehension and an accurate representation of the relationships between the elements described. Achieving this task successfully has a significant societal value, as it offers visually impaired people a means of understanding the content of an image without direct visual access.

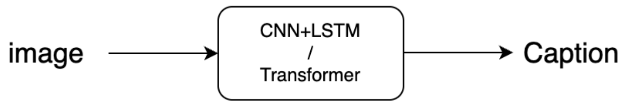


Figure 1. An end-to-end neural network that generates captions from an image

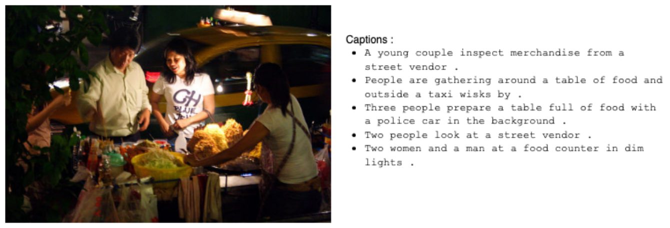
To meet this challenge, our project explores two distinct approaches as shown in Figure 1. The first is based on a neural network architecture inspired by [4], employing a combined CNN and LSTM model for end-to-end image analysis and caption generation. The second approach, influenced by [3], exploits transformers, an advanced neural network architecture recognized for providing state-of-the-art solutions to current computer vision problems.

2. Datatset

The dataset used for this approach is the `Flickr8k` dataset [1], which includes 8,000 images, each accompanied by five descriptive captions. As we can see from Figure 2, each caption points to different perspectives of the image. The diversity and completeness of the dataset make it ideally suited to the objectives of our model.

2.1. Preprocessing

In the preprocessing phase, we began by extracting the necessary features from our images. This step was accomplished using YOLOv8's backbone model. The YOLO



Captions :

- A young couple inspect merchandise from a street vendor .
- People are gathering around a table of food and outside a taxi wisks by .
- Three people prepare a table full of food with a police car in the background .
- Two people look at a street vendor .
- Two women and a man at a food counter in dim lights .

Figure 2. An image and its five captions from the Flickr8k dataset

model is divided into two parts: the feature extraction part and the classification part. In our process, we exclusively utilized the first part. Once we converted the images into tensors, we also made some adjustments to the captions. These changes involved eliminating unnecessary words and punctuation, and additionally, we added the words `startseq` and `endseq` to the beginning and end of each caption. After cleaning the captions, we constructed a vocabulary from these words.

2.2. Padding

As we used PyTorch to create our models,we leveraged the DataLoader to facilitate the loading of our dataset for model training. Given the requirement for uniform sizes, we encountered the need for padding both the inputs and targets, where the targets represent actual and predicted captions. Due to variations in caption lengths, the DataLoader necessitated consistent sizes for what it returned. To address this, we implemented padding by initializing every input and output sequence with a predetermined length, denoted as max_length, set to the maximum caption length of 35 in our case. These sequences were initially filled with zeros and subsequently populated with the actual caption data.

3. Models

In this section, we will delve into the details of the implementation of our two image captioning solutions: the Long Short-Term Memory (LSTM) model coupled with a Convolutional Neural Network (CNN), and the Transformer model. These architectures represent distinct approaches to tackling the task of generating descriptive captions for images.

3.1. LSTM Model

An LSTM unit (Figure 3) includes a cell c, an input gate i, an output gate o and a forgetting gate f. The cell holds information for variable periods of time, while the gates regulate the flow of information. Forget gates determine what to reject from the previous state based on a value between 0 and 1, where 1 retains information and 0 rejects it. Input gates decide what new information to store using a similar mechanism. Output gates control the output of information by assigning a value between 0 and 1, taking into account previous and current states. these variable are defined using the following formulas :

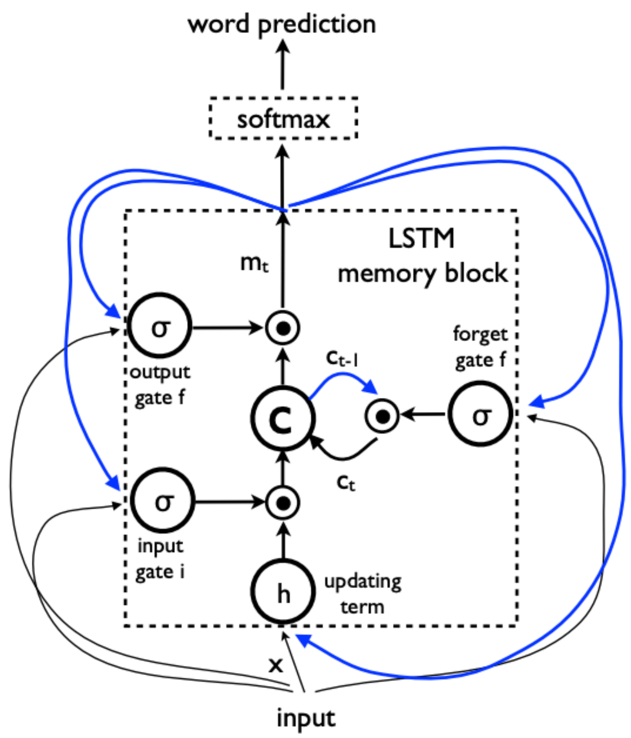


Figure 3. A LSTM cell [4]

$$i_t = \sigma(W_{ix}x_t + W_{im}m_{t-1}) \quad (1)$$
$$f_t = \sigma(W_{fx}x_t + W_{fm}m_{t-1}) \quad (2)$$
$$o_t = \sigma(W_{ox}x_t + W_{om}m_{t-1}) \quad (3)$$
$$c_t = f_t \odot c_{t-1} + it \odot \tanh(W_{cx}x_t + W_{cm}m_{t-1}) \quad (4)$$
$$m_t = o_t \odot \tanh(c_t) \quad (5)$$

where $\odot$ represents the product with a gate value, and the various W matrices are trained parameters.

Training :

In the training phase, prior to entering the LSTM, we resize the features obtained from our images to the $hidden\_size$. Then, the LSTM is used to predict each word in the caption based on image features fe, initializing with the first hidden state h_0. For each LSTM cell input, a word passes through an embedding layer to be encoded and resized to a fixed size. With h_0 and the initial input (always $startseq$ in our case), the LSTM predicts the next word. Then, each following LSTM cell receives an input x_i and the output hidden state of the previous cell, as well as the image feature fe (the necessity of which will be discussed in the results section).

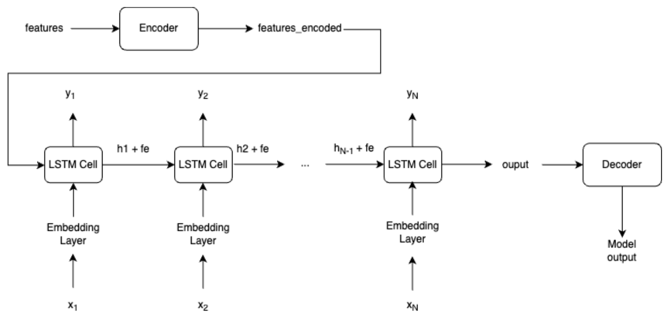


Figure 4. LSTM architecture given the image features as the first hidden state and inputs that are embedded

This procedure is repeated for a maximum of max_length times, resulting in an output (Figure 6). We then pass this output to a decoder so that every prediction the LSTM made will be resized to the vocabulary's size. This phase is performed to subsequently apply a Softmax to our predictions, generating probabilities for each word in the vocabulary as the predicted word.

The loss function employed is the CrossEntropy loss, defined by the following formula:

$$Loss = -\sum_{i}^{N} y_i \cdot \log(\hat{y}_i) \tag{6}$$

The goal of our training phase is to minimize this loss, and this is achieved by adjusting various parameters in the encoder, LSTM, and decoder neural networks.

Given our use of padding for input sequences, the model tended to predict additional elements that needed to be removed before passing it to the loss function. To solve this problem, we implemented a mask to selectively eliminate these filler elements from the target and output tensors.

The mask acted as a filter, allowing us to exclude padded elements when calculating the loss function. By applying this approach, we ensured that the evaluation and optimization processes focused only on the relevant parts of the predicted and target sequences, thus contributing to the overall learning efficiency of our model.

Inference :

During the testing phase, we provide our model fe as h_0 and $startseq$ as input. The model produces an output of size $vocab\_size$, to which we apply Softmax. Then, we sample a word from the available options (words) based on their probabilities and a multinomial distribution. This sampled word becomes the new input for our model, along with the output hidden state plus fe as the new hidden state. This process is repeated until we obtain the $endseq$ word or predict max_length words, whichever occurs first.

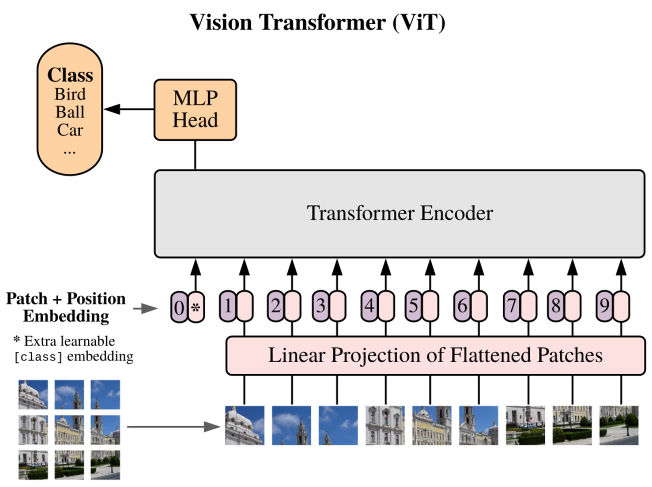

Vision Transformer (ViT)
Class
Bird
Ball
Car
...
MLP Head
Transformer Encoder
Patch + Position Embedding
* Extra learnable [class] embedding
0 * 1 2 3 4 5 6 7 8 9
Linear Projection of Flattened Patches

Figure 5. Vit Architecture

3.2. Transformer Model

The transformer model contains an encoder and a decoder as shown in figure **??**, the encoder is a pretrained ViT (Vision transformer) model that extracts meaningful features from the image while the decoder follows the structure outlined in the paper "Attention is All You Need." [3].

3.2.1 Encoder : ViT (Vision Transformer)

Our initial choice for a feature extractor in this solution was "YOLOv8," which is similar to the approach we took in the first solution. In our original plan, the transformer part was reserved for handling sequences. But after further research we discovered that "AN IMAGE IS WORTH 16X16 WORDS" [2] and that we can indeed extend the transformer part to the encoder also using a Vision Transformer (ViT) model. ViT is a specialized kind of transformer that is used for image classification. This is done by splitting the image into patches, and using the sequence of linear embeddings of this patches as input to the Transformer encoder as shown in fig **??**. The idea behind, is that patches are treated the same way as tokens in an NLP application. Vision Transformer (ViT) attains excellent results compared to state-of-the-art convolutional networks while requiring substantially fewer computational resources to train.

In order to extract meaningful features of our images, we use a pretained Vision Transformer (ViT) model, that is trained on imagenet dataset and our features are the output of the Transformer encoder **??**.

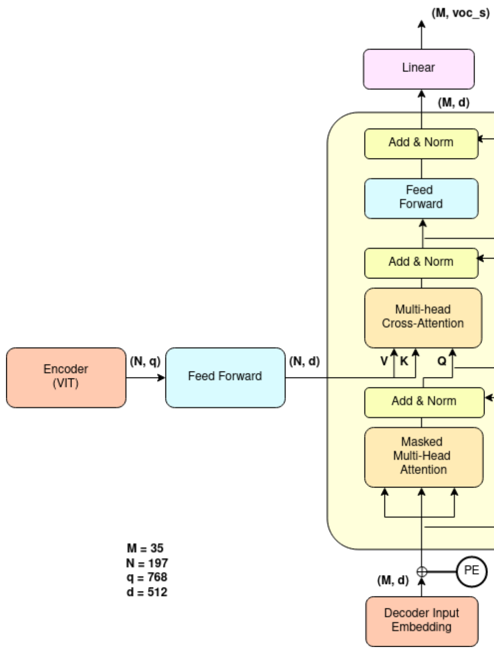


Figure 6. Our Transformer's architecture

3.2.2 Decoder :

The decoder is composed of N (n_heads) identical layers, following the architecture of the decoder in the paper 'Attention is All You Need' [3]. The decoder model starts with a positional encoding of the input sequence. The output is then fed to a masked self-attention layer to correlate tokens with each other in the sequence while preventing information flow from future tokens. This is followed by a cross-attention layer, where the learned tokens' relationships with the features of the image are correlated. The output of the cross-attention layer is then fed to a pointwise feed-forward layer, capturing complex relationships between features. A residual connection (also known as a skip connection) is used between each of the layers in the decoder, involving the addition of the input to the output of each sub-layer followed by layer normalization. The result is then passed to a final linear layer that reshapes the tokens of the sequence into one-hot encoded vectors.

Training :

During the phase of training, the decoder tries to compute the self-attention on textual data, and cross-attention between textual and visual data. The process is as follows :

The tokens of the text sequence are passed as input to the decoder, it starts by going through a positional encoding then it passes by a Masked self attention block of 8 heads. In this attention block, the inner correlations between the text tokens are computed.
However, to train the model under conditions that are similar to the inference phase, the attention of a text token is exclusively calculated with preceding tokens in the sequence. This masking is achieved by using an upper triangular matrix, where all non-zero values are set to $-\infty$:

$$Attention\ Mask = \begin{pmatrix} 0 & -\infty & \dots & -\infty \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & -\infty \\ 0 & \dots & \dots & 0 \end{pmatrix}$$

The attention bloc understands the value $-\infty$ in position $(i, j)\ with\ i > j$ as The correlation of the ith token and the jth token is forbidden.

Furthermore, a second mask is used to facilitate the self-attention block in recognizing embedded padding tokens within the input sequence, allowing for their exclusion during the self-attention process.

With both these masks, the training process focuses only on significant tokens and ignores the paddings.
Once the self attention is done, the transformed text input sequence is passed to a residual connection and a normalization layer, followed by a cross-attention block of 8 heads as a Key, while the Query and the Value tensors are represented by the image's features tensor.

These choices were made so that the dimension of the final transformed sequence matches with the dimension of the text input sequence.
Once the cross-attention is computed, the correlations map is then multiplied by the Value (Image features) in order to generate the output text tokens based on the visual information, weighted by the correlation between words and image patches. The result is then passed to a residual connection and normalization layer before heading to a Feed Forward network.

After another residual connection and normalization layer, the output's generated sequence passes through a final linear layer that transformers the generated words into one-hot encoded tokens.

Inference :

Once the training is done we get to the inference phase. In this process, we initiate the caption generation by providing the model with an image and an initial sequence containing only the token 'startseq.' Subsequently, the model predicts the next word in the sequence. This predicted word is then appended to the original 'startseq,' forming an extended sequence, which is fed back into the model to predict the subsequent word. This iterative process continues until either the model predicts the 'endseq' token or the sequence reaches the predefined maximum length.

In essence, the inference phase involves iteratively expanding the caption sequence by predicting one word at a time, each time appending the predicted word to the existing sequence. This method allows the model to dynamically generate coherent and contextually relevant captions for a given image.

4. Experiments

In this section, we present the experimental setup and results for our image captioning models: the Long Short-Term Memory (LSTM) coupled with a Convolutional Neural Network (CNN) and the Transformer. The goal of these experiments is to evaluate the effectiveness of each model in generating accurate and coherent captions for images.

4.1. Evaluation Metric

As our evaluation metric, we opted to use the BLEU score, a commonly employed metric for assessing the quality of machine-generated text, such as in our case of caption generation. It needs :

- One or more target texts that are considered as the desirable output.
- The machine-generated text that requires evaluation.

BLEU calculates the precision of n-grams (sequences of n words) in the candidate text in comparison to the reference texts. Typically, n is set to 1 (for matches of individual words), 2 (for matches of pairs of words), 3 (for matches of three-word sequences), and 4 (for matches of four-word sequences).

In our case, we specifically use the Unigram BLEU score, denoted as BLEU-1 score, and it is calculated using the following formula :

$$\text{BLEU-1 Score} = \text{BP} \times \text{BLEU-1 Prec}$$

where BLEU-1 Precision is calculated as:

$$\text{BLEU-1 Prec} = \frac{\text{num of matching Unigrams in candidate}}{\text{total num of unigrams in candidate}}$$

and BP (Brevity Penalty) is given by:

$$\text{BP} = \begin{cases} 1 & \text{if len(c)} \geq \text{len(r)} \\ \exp\left(1 - \frac{\text{Reference Length}}{\text{Candidate Length}}\right) & \text{if len(c)} < \text{len(r)} \end{cases}$$

Where c = candidate and r = reference

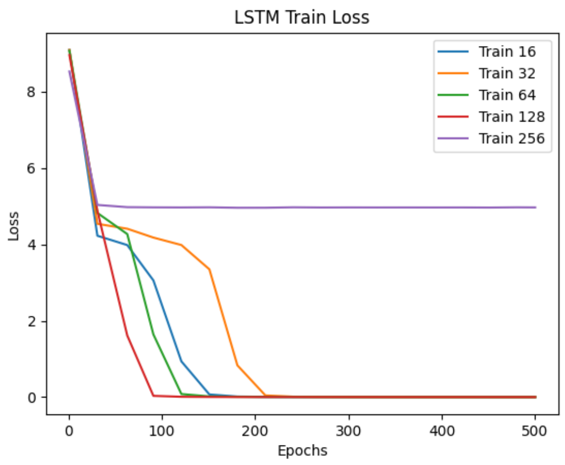


Figure 7. Cross entropy loss for the LSTM model when trained with different sized datasets (16, 32, 62, 128, 256 images)

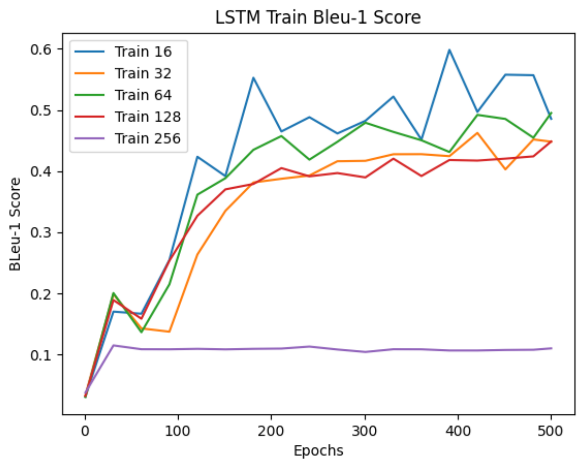
4.2. Results

4.2.1 LSTM Results

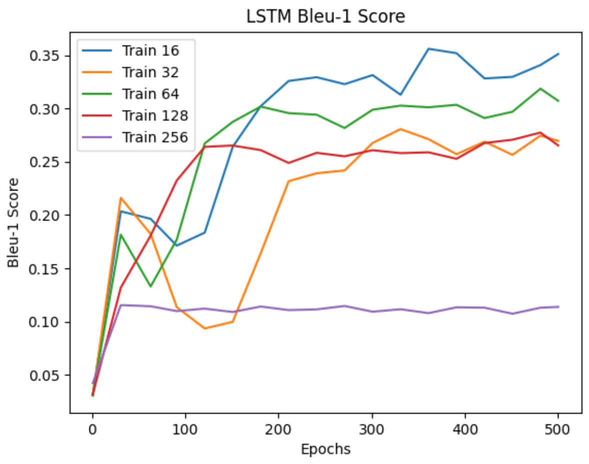
While attempting to train our LSTM model on 75% of the Flickr8k dataset, we encountered a challenge where the loss failed to converge to 0; instead, it remained stagnant around 5, as illustrated in Figure 7 for datasets exceeding 128 images. To explore the model's data-handling capacity during training, we initiated experiments with smaller image counts, ranging from 16 to 128. This approach proved effective as it leads to proper loss convergence and successful overfitting on datasets of this size. However, when attempting to train the model with datasets containing more than 128 images, we observed incorrect loss convergence, preventing successful overfitting.

To assess the accuracy of loss convergence in different datasets (16 images, 32 images, 64 images, 128 images and more than 128 images) and determine if a decrease corresponds to improved model performance on familiar images, we calculated the Bleu-1 score in each epoch. Figure 8a depicts a consistent increase in the Bleu-1 score throughout the training process, except for the training dataset with more than 128 images. This observation aligns with the loss results presented in Figure 7, reinforcing the indication that the model struggles to converge correctly on datasets exceeding 128 images.

One example of the overfitting model's qualitative results on 64 images is depicted in Figure 9a, showing the caption predicted by the model. Notably, the model showed overfitting behavior, as evidenced by the repetition of one of the actual captions. Also in Figure 9b, a result of a prediction made by our LSTM model trained on 256 images, the predicted caption has nothing to do with the picture given.

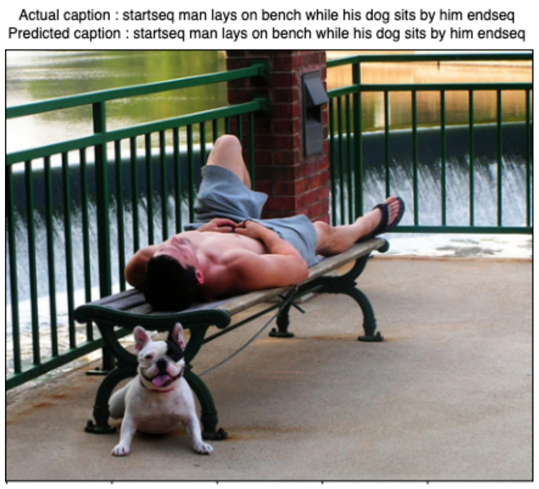


(a) Bleu-1 Score on the different training datasets calculated on each epoch

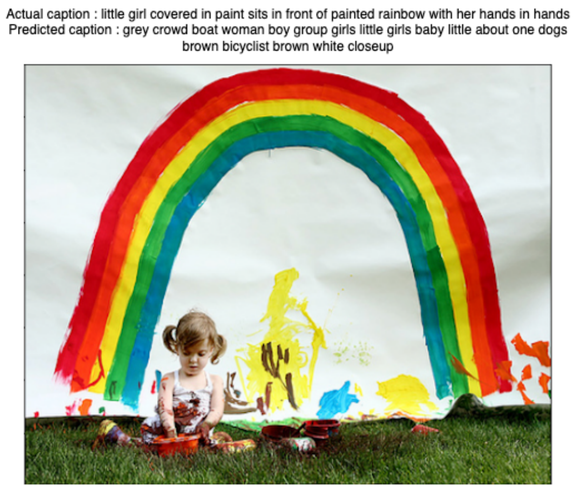


(b) Bleu-1 Score on the test dataset, calculated when the model is trained on different training datasets

Figure 8. Bleu-1 Score in the training and testing phase


Actual caption : startseq man lays on bench while his dog sits by him endseq
Predicted caption : startseq man lays on bench while his dog sits by him endseq

(a) Caption prediction on an image from the training dataset when our model is overfitted on a 64 image dataset


Actual caption : little girl covered in paint sits in front of painted rainbow with her hands in hands
Predicted caption : grey crowd boat woman boy group girls little girls baby little about one dogs brown bicyclist brown white closeup

(b) Caption prediction on an image from the training dataset when our model is trained on a 256 image dataset

Figure 9. Two caption prediction examples from the training phase when the model is trained on 64 or 256 images

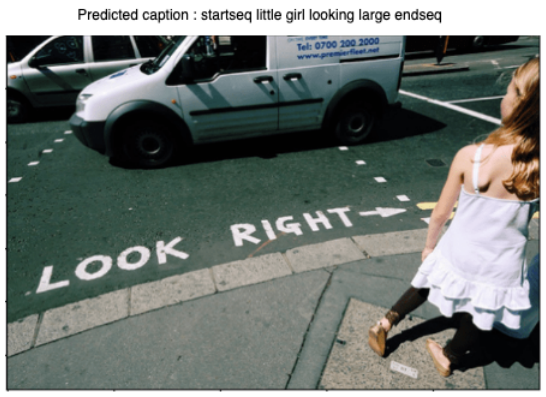

Predicted caption : startseq little girl looking large endseq

Figure 10. Caption prediction on a test image when the model is trained on a 64 image dataset

When training our model on these different datasets, we also evaluated it on 200 images from the test dataset. The bleu-1 score was calculated after each training epoch, as illustrated in Figure 8b. This figure reveals that our model exhibits limitations in handling a substantial amount of training data, with the highest bleu-1 score consistently achieved when training on a dataset of 16 images. Interestingly, the bleu score demonstrated a slight decrease as more images were added to the training dataset.

We also examined the predicted captions for select images using the model trained on 64 images. Figure 11 presents an example of a predicted caption generated by this model.

4.2.2 Transformer Results

In the transformer, we got fairly better results compared to the LSTM part; For starters we manage to let our model overfit on the whole training dataset as shown in Figure 11. This means that the transformer model is more complex than LSTM and can learn from larger datasets. Regarding the test loss, it remained consistent across epochs, which is expected given that cross-entropy loss considers the similarity of words in the same position within the sequence. It is worth noting that a descriptive caption can be considered valid even if it lacks words from the target sequence.

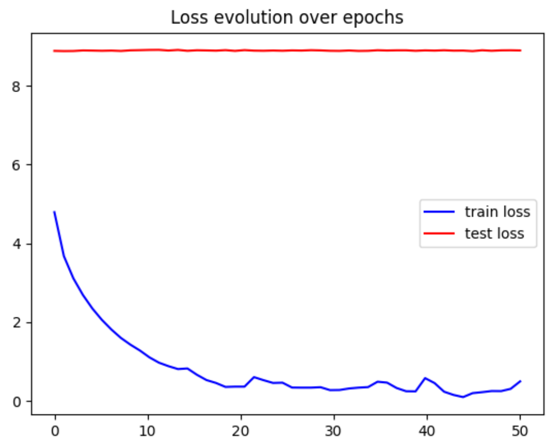


Figure 11. Loss

As for the BLEU-1 score, we notice an overall improvement on train data over epochs as shown in Figure 12. As for test data, we notice a slight improvement on test data in the first epochs. However, the results exhibited inconsistency over subsequent epochs (Figure 13), making it challenging to draw conclusive insights about the representativeness of this metric in evaluating our model's capabilities. Turning to our final metric `human evaluation` we conducted a supervised analysis of generated captions for specific images. This evaluation spanned images from the training dataset, test dataset, and even images sourced from the internet lacking descriptive captions. In Figure 14, it is evident that the model successfully overfits on training images, producing identical captions. In Figures 15, 17, and 16, while the generated captions are not flawless, the model effectively captures essential aspects of the images, such as a woman smiling, surfing, and skiing.

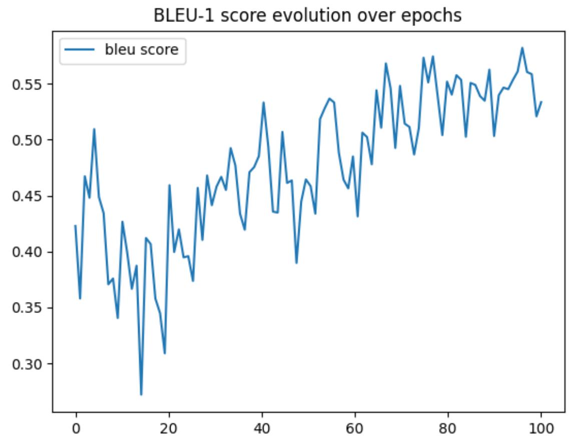


Figure 12. BLEU score on train images over epochs

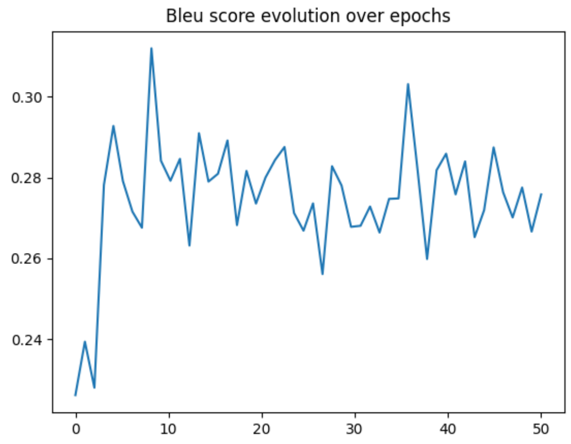
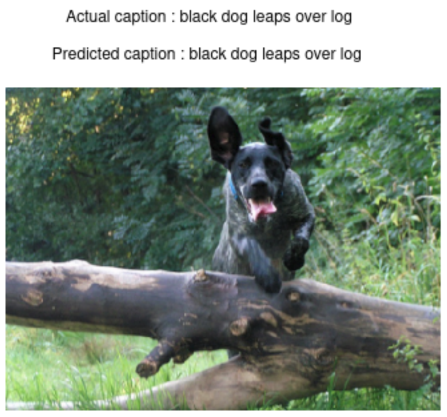


Figure 13. BLEU score on test images over epochs



Actual caption : black dog leaps over log

Predicted caption : black dog leaps over log

Figure 14. Generation example on a train sample

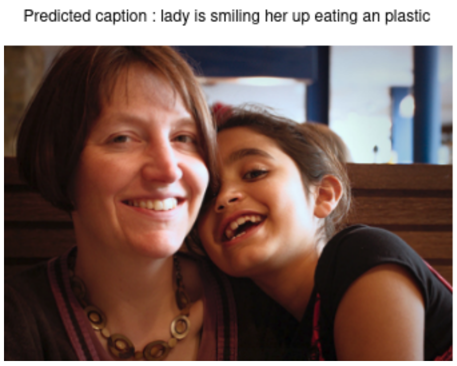

Predicted caption : lady is smiling her up eating an plastic

Figure 15. Generation example on a test sample

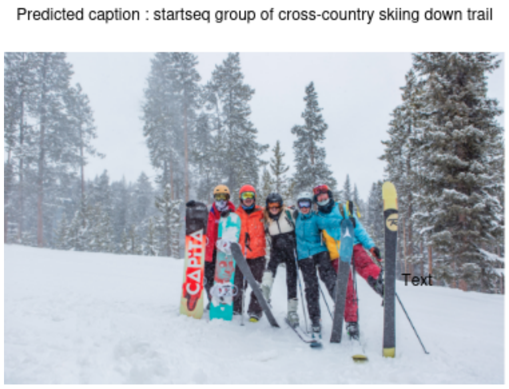

Predicted caption : startseq group of cross-country skiing down trail
Text

Figure 16. Generation example on a new image

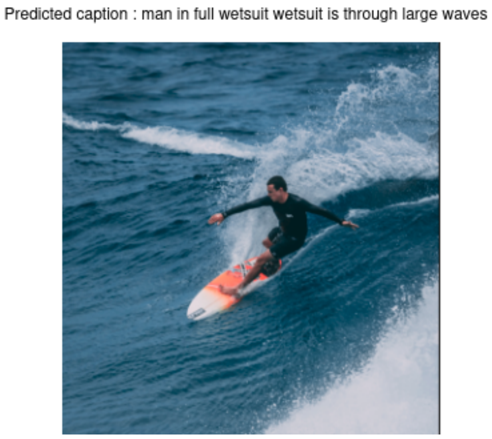

Predicted caption : man in full wetsuit wetsuit is through large waves

Figure 17. Generation example on a new image

5. Conclusion

In conclusion, this report presents two methods that were implemented for image captioning. The first method involved the CNN (Yolov8) coupled with the LSTM model, which yielded to somewhat satisfactory results for datasets with fewer than 128 images but poor results for datasets with more than 128 images. The second method employed the Transformers model, demonstrating promising results. This model exhibited significant improvements, handling the entire dataset with a notably higher BLEU score. Additionally, it exhibited enhanced accuracy in predicting captions and demonstrated the ability to predict captions for out-of-distribution images.

This project has helped to understand why Transformers have become state-of-the-art models for various computer vision problems, particularly in the context of image captioning. A comparison between the LSTM-CNN and Transformer models highlighted their strengths in handling complex relationships within visual data.

References

[1] adityajn105. Flickr 8k. https://www.kaggle.com/datasets/adityajn105/flickr8k. 1

[2] Alexander Kolesnikov Dirk Weissenborn Xiaohua Zhai Thomas Unterthiner Mostafa Dehghani Matthias Minderer Georg Heigold Sylvain Gelly Jakob Uszkoreit Neil Houlsby Alexey Dosovitskiy, Lucas Beyer. An image is worth 16x16 words: Transformers for image recognition at scale, 2020. 3

[3] Niki Parmar Jakob Uszkoreit Llion Jones Aidan N. Gomez Lukasz Kaiser Illia Polosukhin Ashish Vaswani, Noam Shazeer. Attention is all you need, 2017. 1, 3, 4

[4] Samy Bengio Dumitru Erhan Oriol Vinyals, Alexander Toshev. Show and tell: A neural image caption generator, 2015. 1, 2