A red and black logo

Description automatically generated

MSc in Data Science

Project Report

2024/25

Enhancing Image Captioning using ConvNeXt with LSTM and Transformer Decoders

Sajeel Nadeem Alam

Supervised by: Professor Kevin Ryan

Date of Submission: 30th September, 2025

By submitting this work, I declare that this work is entirely my own except those parts duly identified and referenced in my submission. It complies with any specified word limits and the requirements and regulations detailed in the assessment instructions and any other relevant programme and module documentation. In submitting this work I acknowledge that I have read and understood the regulations and code regarding academic misconduct, including that relating to plagiarism, as specified in the Programme Handbook. I also acknowledge that this work will be subject to a variety of checks for academic misconduct.

Signed: Sajeel Nadeem Alam

# Abstract

Due to the multi-domain nature of image captioning, researchers in computer vision and natural language processing have traditionally relied on encoder-decoder architectures with a CNN and LSTM. While effective, these models struggle with long-range dependencies and parallel processing. This study investigates the performance of a vision transformer inspired CNN called ConvNeXt, with a Transformer decoder as a replacement for the conventional LSTM with an attention module. The research’s primary objective was to quantitatively and qualitatively compare the performance of both decoders as well as investigate the impact of fine-tuning the ConvNeXt to identify the optimal fine-tuning depth. Secondary experiments include comparing training strategies, pre-trained word embeddings and decoding strategies. The models were trained on the MS COCO dataset. Key findings revealed that the Transformer decoder outperformed the LSTM in terms of BLEU scores. Furthermore, fine-tuning the ConvNeXt till layer 5 proved to be the most effective with a 34.96 BLEU-4 score. Training with teacher forcing as opposed to without it provided better inference time results contrary to previous studies. Experiments with pre-trained Word2Vec and GloVe embeddings displayed minimal quantitative improvements to a 35.06 BLEU-4 score and a qualitative analysis of generated captions showed similar results by greedy and beam search.

**Keywords:** Image Captioning, ConvNeXt, Transformer Decoder, Teacher Forcing, Pretrained Word Embeddings

Table Of Contents

[Abstract 2](#_Toc210053560)

[1. Introduction and Objectives 7](#_Toc210053561)

[1.1. Background 7](#_Toc210053562)

[1.2. Rationale and Beneficiaries 7](#_Toc210053563)

[1.3. Project Objectives 8](#_Toc210053564)

[1.4. Methodology and Work Plan 9](#_Toc210053565)

[1.5. Report Structure 10](#_Toc210053566)

[1.6. Use of GenAI 10](#_Toc210053567)

[2. Critical Context 11](#_Toc210053568)

[2.1. Early Image Captioning Techniques 11](#_Toc210053569)

[2.2. Encoder – Decoder Architecture 11](#_Toc210053570)

[2.3. Improvements in the Encoder – Decoder Architecture 12](#_Toc210053571)

[2.4. Self-Attention and Transformers 13](#_Toc210053572)

[2.5. ConvNeXt Encoder 13](#_Toc210053573)

[2.6. Training Strategies 15](#_Toc210053574)

[2.7. Pretrained Word Embeddings 16](#_Toc210053575)

[2.8. Decoding Strategies 16](#_Toc210053576)

[3. Methods 17](#_Toc210053577)

[3.1. Dataset and Preprocessing 17](#_Toc210053578)

[3.1.1. Dataset 17](#_Toc210053579)

[3.1.2. Preprocessing 17](#_Toc210053580)

[3.1.3. Dataloader 18](#_Toc210053581)

[3.2. Model Architecture 18](#_Toc210053582)

[3.2.1. ConvNeXt Encoder 19](#_Toc210053583)

[3.2.2. Baseline Decoder: LSTM with Attention 19](#_Toc210053584)

[3.2.3. Proposed Decoder: Transformer Decoder 20](#_Toc210053585)

[3.3. Experimental Design 21](#_Toc210053586)

[3.3.1. Training Strategies 21](#_Toc210053587)

[3.3.2. Fine-tuning ConvNeXt 22](#_Toc210053588)

[3.3.3. Supplementary Analysis of Attention Regularization 23](#_Toc210053589)

[3.3.4. Word Embeddings 23](#_Toc210053590)

[3.3.5. Decoding Strategy 24](#_Toc210053591)

[3.4. Training and Hyperparameters 24](#_Toc210053592)

[3.5. Evaluation Metrics 25](#_Toc210053593)

[4. Results 27](#_Toc210053594)

[4.1. Preliminary Results 27](#_Toc210053595)

[4.1.1. Results on Flickr8k 27](#_Toc210053596)

[4.1.2. Results with Original Study’s Codebase 27](#_Toc210053597)

[4.2. Baseline Performance on MS COCO 28](#_Toc210053598)

[4.2.1. The Impact of Training Strategies 28](#_Toc210053599)

[4.2.2. Selecting the Best Decoder 32](#_Toc210053600)

[4.3. The Impact of Fine-tuning ConvNeXt 34](#_Toc210053601)

[4.3.1. Quantitative Analysis 34](#_Toc210053602)

[4.3.2. Qualitative Analysis 35](#_Toc210053603)

[4.4. Analysis of Attention Regularization 37](#_Toc210053604)

[4.5. The Impact of Pretrained Word Embeddings 38](#_Toc210053605)

[4.5.1. Quantitative Analysis 38](#_Toc210053606)

[4.5.2. Qualitative Analysis 39](#_Toc210053607)

[4.6. Greedy Search vs Beam Search 40](#_Toc210053608)

[5. Discussion 42](#_Toc210053609)

[5.1. Addressing Project Objectives 42](#_Toc210053610)

[5.1.1. Implement the ConvNeXt encoder, LSTM and Transformer decoders 42](#_Toc210053611)

[5.1.2. Compare the model’s performance training with and without teacher forcing 42](#_Toc210053612)

[5.1.3. Train and select the best decoder on the basis of their baseline performance 43](#_Toc210053613)

[5.1.4. Analyze the performance of fine-tuning different layers of the ConvNeXt 44](#_Toc210053614)

[5.1.5. Train the architecture with pretrained Word2Vec and GloVe embeddings 44](#_Toc210053615)

[5.1.6. Compare caption quality with greedy search and beam search 45](#_Toc210053616)

[5.2. Answering the Research Question 45](#_Toc210053617)

[5.3. Broader Discussions and Implications 46](#_Toc210053618)

[5.3.1. Comparing with Existing Literature 46](#_Toc210053619)

[5.3.2. Architectural and Practical Implications 47](#_Toc210053620)

[5.3.3. Training and Methodological Implications 47](#_Toc210053621)

[6. Evaluation, Reflection and Conclusion 48](#_Toc210053622)

[6.1. Project Evaluation 48](#_Toc210053623)

[6.2. Reflection 49](#_Toc210053624)

[6.3. Conclusion 49](#_Toc210053625)

[References 51](#_Toc210053626)

[A. Appendix: Exploratory Data Analysis and Metric Formulae 56](#_Toc210053627)

[A.1. MS COCO Captions and Words 56](#_Toc210053628)

[A.2. Wordmap used to encode captions 57](#_Toc210053629)

[A.3. Caption encoding example using wordmap 57](#_Toc210053630)

[A.4. Formulae for BLEU score and attention regularization loss 58](#_Toc210053631)

[B. Appendix: Additional Experiments and Results 59](#_Toc210053632)

[B.1. Generated captions of unseen sample images (Teacher Forcing vs Non-Teacher Forcing) 59](#_Toc210053633)

[B.2. Generated captions of unseen sample images (Fine-tuning ConvNeXt) 61](#_Toc210053634)

[B.3. Generated captions of unseen sample images (Pretrained Word Embeddings) 63](#_Toc210053635)

[C. Appendix: Project Proposal Document 65](#_Toc210053636)

[D. Appendix: GenAI Prompts and Responses 77](#_Toc210053637)

[E. Appendix: Code Availability 82](#_Toc210053638)

[F. Appendix: Word Count Table 83](#_Toc210053639)

# 1. Introduction and Objectives

## 1.1. Background

Humans have the ability to describe their environment with immense precision. They are able to explain a visual scenario even after glancing at it for a moment. This is due to the sense of sight, which from the eyes and through optic nerves sends visual information to the brain that has the ability to convert it to natural language descriptions. Making computers replicate this task has been a challenge in recent years for researchers in artificial intelligence (Bai & An, 2018). This task is formally known as ‘Image Captioning’ and can be defined as generating a descriptive natural language caption for an image in the form of a detailed, comprehensive sentence describing the objects in the image and their interactions with each other (Chen et al., 2024). The system takes the image as an input, uses a visual understanding model and a language model to generate meaningful captions which are outputted (Stefanini et al., 2021). Hence this task employs both computer vision and natural language processing, and is usually treated as supervised learning. The key problem faced is linking relevant visual information from the image along with previously generated words at each step of the caption generation process. The field has evolved significantly from early template-based approaches to sophisticated encoder–decoder architectures in which an image model e.g., a CNN is used to extract features from an image and a language model e.g., an RNN or LSTM uses them to construct captions, and this study aims to improve both elements of this architecture.

## 1.2. Rationale and Beneficiaries

While Long Short-Term Memory (LSTM) networks have performed well as the standard decoder for encoder-decoder systems due to their ability to process sequential data, the recent dominance of the Transformer architecture in language modeling presents an opportunity for a comparative study laying the grounds for this project. The key motivation of this study is to investigate if the architectural advantages of the Transformer such as multi-headed cross and self-attention allowing improved handling of long-range dependencies can result in higher quality captions as compared to an LSTM decoder within the same encoder-decoder based image captioning framework. The encoder used by this study is a ConvNeXt which is a CNN built on ResNet and inspired by vision transformers. The secondary area of investigation further motivating this study is to find the optimal depth of layers to fine-tune the ConvNeXt for this downstream task. The rationale for improving computers’ ability in image captioning is generated by the potential of using captioning systems in real-world applications (Dognin et al., 2022). The beneficiaries of this work are visually impaired individuals who will benefit from an improved image captioning system allowing them to have a better quality of life (Makav & Kilic, 2019). Moreover, fine-tuning this model on a medical image dataset can support individuals in the healthcare industry to diagnose medical scans (Ayesha et al., 2021). Farmers can also benefit from image captioning systems that are fine-tuned for monitoring plant conditions from close-up images (Putra et al., 2020). E-commerce businesses will benefit by leveraging it for product retrieval through image-generated text descriptions allowing for better search functionalities (Tang et al., 2024) whereas supply chain managers will benefit from image captioning assisting industrial robots to make informed decisions by allowing them to understand visual data (Luo et al., 2019). Lastly, researchers in the field of computer vision and natural language processing will benefit by the findings of this study to inform their architectural choices for future generative models.

## 1.3. Project Objectives

The question this study aims to answer is “How can fine-tuning a ConvNeXt encoder, combined with an LSTM or Transformer decoder enhance image captioning performance, and what is the impact of teacher forcing, pre-trained word embeddings and decoding strategies on the quality of generated captions?” In order to confidently answer the question, this study aims to achieve the primary objectives stated below.

1. **Objective**: Implement the ConvNeXt encoder, LSTM and Transformer decoders and connect them together so that data can pass through them and loss can be backpropagated to train the architecture.

**Test:** The successful training of the architecture with validation checkpoints calculating BLEU scores.

1. **Objective:** Train the architecture with and without teacher forcing to investigate which training strategy performs better for both decoders on the MS COCO dataset.

**Test:** Compare the training and validation losses, BLEU scores and quality of generated captions of the models trained with and without teacher forcing to select the better performing strategy.

1. **Objective:** Train the architecture with the best training strategy using both the LSTM and Transformer decoders separately on the MS COCO dataset and select the best performing one.

**Test:** A quantitative analysis of the models’ BLEU scores on the test set along with a qualitative analysis comparing the quality of generated captions for unseen images, with a clear conclusion of one architecture performing better than the other.

1. **Objective:** Fine-tune different depths of layers of the pre-trained ConvNeXt encoder and identify the optimal depth in terms of architecture performance.

**Test:** A series of experiments will be conducted by fine-tuning different layers of the ConvNeXt. The best performance on the validation set during training along with the quality of captions generated for unseen images will be evaluated to determine the optimal fine-tuning strategy for the ConvNeXt.

1. **Objective:** Train the architecture with pretrained word embeddings and compare the performance with random embeddings to investigate whether prior linguistic knowledge improves performance.

**Test:** A comparison of BLEU scores and captions generated by the architectures trained with random embeddings, Word2Vec and GloVe embeddings to determine which one performs the best.

1. **Objective:** Test the best performing model on unseen images using greedy and beam search, and compare the quality of generated captions.

**Test:** The comparison of the quality of generated captions on sample unseen images using both decoding strategies.

The key deliverables of this study include a working architecture of the image captioning system, an open-source implementation of the architecture and knowledge about the project objectives. This study will only focus on generating captions for static images and will be limited to the MS COCO dataset that contains well-annotated images. Hence the models will be trained on general images and will not be fine-tuned for a specific domain. The scope of this work does not include the development of real-time image captioning systems for generating captions for videos as this would require more temporally complex models.

## 1.4. Methodology and Work Plan

The methodology of this project follows a sequential process illustrated in Figure 5 of appendix C. The initial stages focus on the implementation of a robust data pipeline and the ConvNeXt encoder along with the LSTM and Transformer decoders to test the functionality of the architecture on the smaller Flickr8k dataset. This was followed by training both decoders on the larger MS COCO dataset with and without teacher forcing to select the better performing training strategy and decoder. The project then included a series of experiments to determine the optimal fine-tuning depth for the ConvNeXt encoder and investigate whether pretrained word embeddings and beam search decoding strategy result in an improvement in performance.

## 1.5. Report Structure

Chapter 2 presents a comprehensive review on the existing works and literature in this field, providing valuable critical context. Chapter 3 explains this study’s methodology and experimental setup. Chapter 4 presents the results of the experiments whereas Chapter 5 discusses the findings, their implications and the extent to which the research question has been answered. Lastly, Chapter 6 provides a project evaluation, personal reflections and final conclusions.

## 1.6. Use of GenAI

Gemini which is a generative artificial intelligence chatbot developed by Google AI was used in two instances in this project.

Firstly, it was used to consolidate the logic of the helper functions in PyTorch’s source code for the forward method of TransformerDecoderLayer into a single, unified forward method for this study’s CustomTransformerDecoderLayer. The source code is publicly available at PyTorch’s Transformer’s official GitHub repository linked to its TransformerDecoderLayer’s documentation section. The prompt used and response received is mentioned in Prompt 1 in appendix D.

Secondly, Gemini was used to generate 3 lines of code to compute the average cross-attention weights for the current word in the transformer decoder and update alphas accordingly. This was done to ensure that despite complex, high-dimensional tensors, the average cross-attention weights are computed correctly. These lines are used in the transformerDecoderAttVis.py and caption.py files in the code as mentioned in the comments in the respective files. The prompt used and response received is mentioned in Prompt 2 in appendix D.

# 2. Critical Context

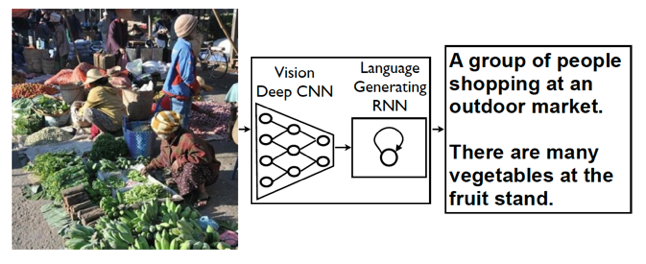
This chapter provides a critical review of the existing works in the field of image captioning. It shows how techniques have built on top of one another and improved over time, and also acts a source of motivation for the methods used in this study.

## 2.1. Early Image Captioning Techniques

The first image captioning systems employed template-based methods, where fixed sentence templates with blank slots were filled with identified objects and their relationships to generate captions (Hossain et al., 2019). These systems aimed to produce grammatically correct captions for specific domains but were rigid and produced weak, contextually limited captions that lacked generalization across diverse images (Hossain et al., 2019). Another approach, retrieval-based methods, selected the caption for a query image from a pool of captions of similar images. While this allowed the captioning of large amounts of image data, it was limited to reusing existing descriptions which did not cater to specific details or context of individual images (Hossain et al., 2019).

## 2.2. Encoder – Decoder Architecture

With advancements in image and language models, deep-learning based image captioning approaches became popular. Models that have the ability to learn relevant features from images and use them to generate specific, context-aware captions were used which allowed the research to improve from fixed captions to novel caption generation (Hossain et al., 2019). The most prominent architecture using these models is the encoder-decoder architecture inspired by sequence-to-sequence learning for language translation (Sutskever et al., 2014). In encoder-decoder architectures, an image model (encoder) extracts relevant features from the image which are passed onto a language model (decoder) that generates a natural language description for the image using provided visual features (Stefanini et al., 2023) as illustrated in Figure 1. One prominent work which used this architecture was a team from Google and their implementation consisted of fine-tuning GoogleLeNet as an encoder on the MS COCO dataset while training an LSTM as a decoder using stochastic gradient descent (Vinyals et al., 2016). They surpassed the existing state-of-art results by achieving a 59 BLEU score on the Pascal dataset and won the 2015 MS COCO Image Captioning Challenge. Since then, in most existing literature, the encoder-decoder architecture has been implemented with a CNN encoder and an RNN/LSTM decoder.



**Figure 1.** Basic encoder – decoder architecture (Vinyals et al., 2016)

Although it shows promising results, this approach has its limitations since the image features are provided to the RNN only at the beginning leading to vanishing gradient issues. This is mitigated by the LSTM however, it faces the issue of weakened influence of the image’s semantic as the caption progresses resulting in less context aware captions (Singh et al., 2024). Moreover, the CNN focuses on the image as a whole instead of the individual objects and their relationships.

## 2.3. Improvements in the Encoder – Decoder Architecture

In order to overcome the issues mentioned earlier, researchers have tried to modify individual components of the encoder-decoder architecture. In one paper, researchers used Faster R-CNN with ResNet-101 to identify objects in the image, encode their geometric information and spatial relationships, and generate feature vectors for each region in the image which helped overcome the issue of only focusing on the image as a whole (Herdade et al., 2019). In another study, authors used a deep bidirectional LSTM as a decoder which had the ability to process the sentence and capture context in both directions resulting in more contextually aware captions (Wang et al., 2016). Although there was an improvement in performance, contextual information from the image was not provided to the decoder during the caption generation process which was a limitation. Addressing this issue, a study used a 16-layer-Oxford-Net to extract image features and cross-modal retrieval to find relevant texts in the image. The image features were fed to a guided LSTM and relevant texts were incorporated in each LSTM gate to provide semantic information (Jia et al., 2015). This gave contextual information about the image to the decoder during caption generation allowing for more contextually aware captions however, this information was static and did not adapt during the caption generation process. To address this issue, another study used GoogleNet to extract image features that were fed into an LSTM, which had an integrated semantic attention module. This module, using attribute detectors, identified semantic concepts in the image and dynamically updated attention based on the previously generated word and its semantic context, guiding the LSTM at every step to relevant image regions during caption generation resulting in contextually rich captions (You et al., 2016). However, this approach provided semantic information about local features only, focusing on specific regions of the image at each step. Combined with sequential processing in the LSTM, where the model cannot consider the entire image and generated caption so far (i.e., all previously generated words) simultaneously, this resulted in a limitation, as it was unable to capture the global context and relationships across the entire image which are equally important as local, fine-grained features.

## 2.4. Self-Attention and Transformers

In recent years, with the introduction of self-attention and transformers, researchers have explored their potential use in overcoming the limitation of global context faced in CNN-LSTM architectures. In one study, researchers incorporated a self-attention mechanism in both the CNN encoder and LSTM decoder to dynamically assign weights to each part of the image and generated caption, providing a global and local context while generating the next word. They further enhanced this with an attention-on-attention module that filtered out irrelevant image regions, allowing the model to focus on more relevant parts and capture global dependencies during caption generation which allowed them to achieve better results (Huang et al., 2019). In another study, authors replaced the CNN-LSTM architecture with a full transformer architecture for encoding and decoding. By processing the image as a sequence of patches and leveraging the transformer’s multi-headed self-attention mechanism to focus on previously generated words and its multi-headed cross-attention mechanism to focus on relevant regions of the image, the model was able to capture both global and local relationships between the image and the generated caption so far. The parallel processing ability of the transformer due to its multiple heads considered all regions of the image and the generated caption simultaneously, providing global context during the caption generation process (Liu et al., 2021). Using a full transformer network instead of a CNN-LSTM architecture with an attention module overcomes the limitation of relying on a CNN for feature extraction which processes image information sequentially thus limiting the global context across the image and caption. On the other hand, transformers are able to capture both global and local contexts but might not always handle fine-grained local features as well as CNNs. The key takeaway is to find an optimal balance between both architectures.

## 2.5. ConvNeXt Encoder

In order to achieve this balance, in 2022 the research team at Meta developed an improved CNN called ConvNeXt which builds on the foundational model of ResNet and incorporates upgrades from vision transformers. Upgrades such as using larger convolution kernels enabled the model to gather broader context across the image while a hierarchical feature learning approach inspired from Swin Transformers, processed the image at different resolutions. This allowed the model to capture fine-grained local features at lower resolutions combined with broader context at higher resolutions and integrate them in the feature vectors (Liu et al., 2022). This balance between local and global context serves as motivation to use ConvNeXt for image captioning as was done by a team of researchers in 2024. In their paper titled ‘A Study of ConvNeXt Architectures for Enhanced Image Captioning’ a ConvNeXt encoder was used with an LSTM decoder integrated with an attention module to provide more context during the caption generation process.

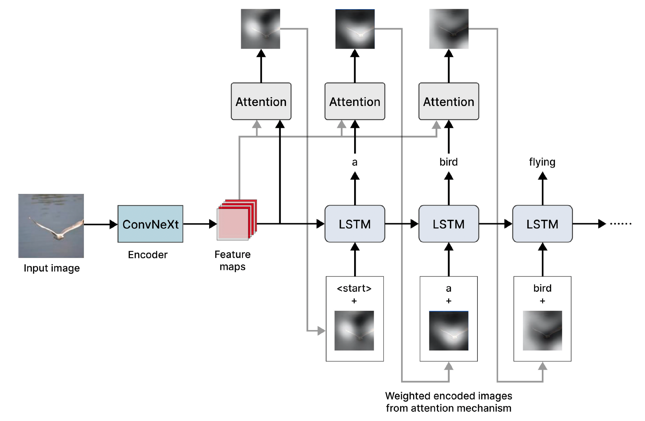
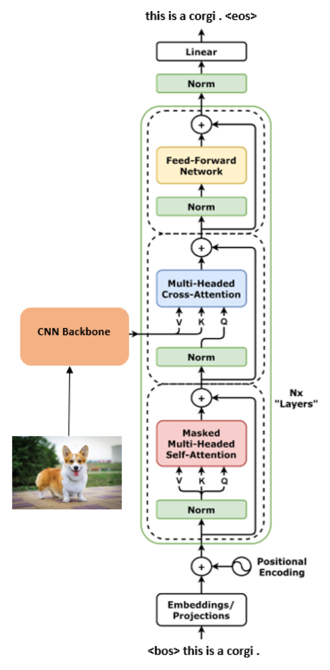




Figure 2 shows that at each stage in the LSTM, feature vectors extracted from the image were multiplied with attention weights to create a context vector which was concatenated with the input word embedding. The attention weights were calculated using the previous hidden state and determined which part of the image to focus on. The combination of context aware image features from the ConvNeXt and an attention module integrated in the LSTM provided the model with a balance of local and global contexts during the caption generation process thus outperforming vision transformers (Ramos et al., 2024).

The ability of ConvNeXt to extract locally and globally context aware image features and its use case in the paper serve as a source of motivation to use it as an encoder in this study. Exploring further improvements by fine-tuning various layers of the ConvNeXt is a key objective of this study since the paper did not fine-tune any layers. Considering its strengths mentioned earlier, this study will explore using an LSTM decoder with an integrated attention module similar to the paper (Ramos et al., 2024).





However, in a recent study (Cahyono & Jusuf, 2024) authors used a hybrid architecture that incorporates a CNN encoder with a transformer decoder as shown in Figure 3. Considering the transformer’s ability for parallel processing, which allows it to simultaneously attend to the entire image and the part of the caption generated so far, the model can effectively utilize local and global contexts at each step to generate more context-aware captions. Based on this, this study will also explore using a transformer decoder and select the better performing decoder for further analysis.

## 2.6. Training Strategies

The study (Bengio et al., 2015) highlights that training an encoder-decoder architecture using teacher forcing in which words from the ground truth caption are provided at each time point to generate the next word achieves a 28.8 BLEU-4 on the MS COCO dataset whereas training without teacher forcing in which the model’s own predictions are used to generate the next word achieves a 11.2 BLEU-4 score with the same architecture. Although teacher forcing exposes the model to different conditions in training and inference causing it to perform poorly at the time of inference with its own predictions known as the exposure bias problem (Rennie et al., 2017), it provides the model with stable training. Non-teacher forcing results in error compounding and unstable gradient updates if the model makes a mistake early on resulting in poor performance but it avoids the exposure bias problem potentially resulting in better inference time performance. This study will train the architecture using both strategies to find the balance between generalization and how fast the model converges.

## 2.7. Pretrained Word Embeddings

Individual words in the true and generated captions are represented in a continuous vector space known as word embeddings which capture the semantic relationship between words. These embeddings can either be randomly initialized and learnt during training or initialized with weights from pre-trained models such as Word2Vec which learns embeddings by analyzing the local context of words within a sentence (Mikolov et al., 2013) or GloVe which uses the global word co-occurrence statistics from an entire corpus (Pennington et al., 2014). A study (Atliha et al, 2021) investigated the effect of using pre-trained word embeddings against random word embeddings to encode the caption before passing it to the decoder and fine-tuning them during training to analyze if prior semantic knowledge from a pre-trained corpora improves model performance. They observed BLEU-4 scores of 26.4, 26.7 and 26.9 on the MS COCO dataset using random embeddings, pre-trained Word2Vec and GloVe embeddings respectively. Although the improvement is not significant, this study aims to explore whether this trend stays consistent for the proposed architecture.

## 2.8. Decoding Strategies

While selecting the next word in training without teacher forcing or at inference, traditionally greedy search and beam search have been used. In greedy search, the model selects the word with the highest logit score at every word position in the generated caption which becomes the input for the next time step. Although computationally efficient, it is known to get stuck in a local minimum (Shi et al., 2024). Whereas in beam search, the top k sequences at each time point are carried on to the next step and when all top k sequences have finished generating, the sequence with the highest overall score is selected. This makes beam search computationally expensive however, it allows the model to explore more possible sequences to produce a potentially better caption (Shi et al., 2024). In a study (Vinyals et al., 2016), researchers from Google discovered that for image captioning, beam search with a small beam size of 3 was optimal and improved their greedy search results by 2 BLEU scores on average. This study aims to compare the quality of generated captions using both decoding strategies.

# 3. Methods

This chapter goes through various stages of the project such as the dataset used, pipeline built to load it into the model, baseline and proposed model architectures, experiments conducted and the evaluation metrics used at each stage.

## 3.1. Dataset and Preprocessing

### 3.1.1. Dataset

This study uses the MS COCO dataset (Microsoft, 2014) which was made by Microsoft for the COCO 2015 Image Captioning challenge and has since been used widely by researchers for the image captioning task, making it a standard benchmark and allowing for meaningful comparisons of this study’s results with existing works. In the study’s initial stages, the smaller Flickr8k dataset was used to test the robustness of the architecture with fewer images. For both datasets, the popular Karpathy split (Karpathy & Fei-Fei, 2017) was used according to which the MS COCO dataset has 123,287 images out of which 5000 are for validation, 5000 are for testing and the rest are for training whereas the Flickr8k dataset has 8000 images out of which 1000 are for validation, 1000 are for testing and the rest are for training. Each image is well-annotated with five captions providing variety for evaluation. Figure 12 in appendix A1 shows the distribution of caption lengths with some outliers and the average length being 12.46 tokens. Figure 13 in appendix A1 shows the word count of the top 20 most occurring words which gives an overview of the type of descriptions the models will be trained on.

### 3.1.2. Preprocessing

The image and caption data are preprocessed for efficient data loading and model training. Firstly, the Karpathy split file is loaded which contains image metadata and a list of captions for each image organized into training, validation and testing splits. Using this file, all the captions and their respective images are iterated over and captions with lengths greater than 50 are discarded, followed by splitting the images and the captions into training, validation and testing. A word map is made in which each word is mapped to a unique ID and words appearing less than 5 times across all the captions are excluded to mitigate the effect of rare words. Four special tokens <unk>, <start>, <end> and <pad> are added to the vocabulary with their own unique IDs. The word map is serialized as a JSON file to be used during training and inference, a section of which is displayed in Figure 14 in appendix A2.

The core of preprocessing involves saving raw .jpg images and captions into file formats suitable for efficient data loading. For each split (train, validation and test) three files are generated. The first file consists of image data for which all the images in that split are loaded, converted to RBG, resized to 256x256 pixels and stored in a single .HDF5 file which allows efficient loading of image data. The second file stores all the captions in JSON format. For each image, its five captions are encoded into numerical sequences using the generated word map. Each sequence is prepended with the <start> token, appended with the <end> token and padded with the <pad> token until it reaches a maximum caption length of 50 as displayed in Figure 15 in appendix A3. The third file which is also a JSON stores the true length of each caption for every image. This preprocessing ensures that the image and caption data are aligned, standardized and optimized to be used efficiently for training and inference.

### 3.1.3. Dataloader

To ensure efficient data loading from the .HDF5 and JSON files, a custom PyTorch dataset class is set up which implements lazy loading by ensuring that the large .HDF5 file containing all the image data is opened only once at the time of first data retrieval and subsequent images are retrieved only when required. This prevents out of memory issues. The smaller captions and caption lengths JSON files are loaded fully at the time of initialization for faster access. The dataset is configured to handle the structure of MS COCO where each image has five captions and ensures that every caption is returned with its corresponding image. This is essential since at inference every generated caption requires all five of its corresponding true captions to calculate the evaluation metrics.

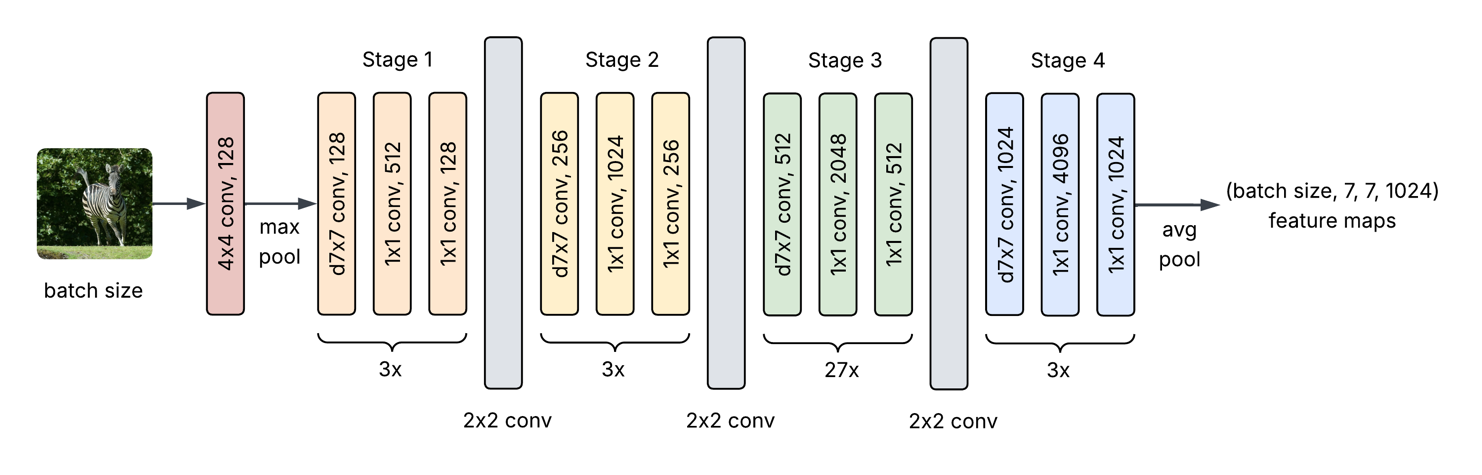
This dataset class is passed into PyTorch’s dataloader which loads data in batches to feed the model. The dataloader loads 32 images, captions and their lengths in one batch. The training data is shuffled at the start of each epoch to prevent the model from learning the order of the data. The dataloader performs multi-process data loading with 6 worker processes allowing the CPU to pre-fetch data while the GPU is computing resulting in faster training. Worker processes are kept alive throughout the training process to reduce the overhead of re-initializing them and opening the .HDF5 file again at the start of each epoch.

## 3.2. Model Architecture

For a fair comparison with the transformer decoder of this study, the encoder and LSTM decoder are adapted from the codebase (Ramos et al., 2024) which seems to have adapted it from (Vinodababu, 2019).

### 3.2.1. ConvNeXt Encoder

This study uses an encoder-decoder architecture in which the encoder is the base version of ConvNeXt from PyTorch which is pretrained on the ImageNet-1k dataset. It has a feature extractor layer followed by a pooling layer and a classification head. For image captioning, the pooling layer and classification head are removed since ConvNeXt is used only to extract image features. Figure 4 shows that the feature extractor of ConvNeXt has seven sequential layers, out of which four are convolutional blocks containing several convolutional layers and three are 2x2 convolutional down sampling layers.





Images are passed as batches to the ConvNeXt. The output of the feature extractor is a high-dimensional feature map where the channel dimension is 1024. This is passed through an adaptive average pooling layer which resizes the feature map to a fixed size of 7x7 which is permuted to the shape (batch size, 7, 7, 1024). This ensures that the ConvNeXt encoder can accept images of varying size but produces consistent sized output tensors that are compatible with the decoder. A key aspect of the encoder is the ability to freeze certain layers while fine-tuning preventing any updates to the pretrained weights.

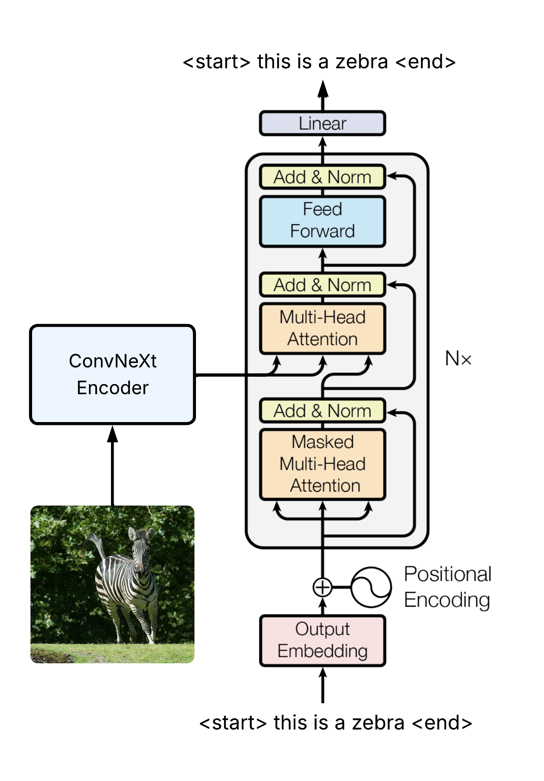
### 3.2.2. Baseline Decoder: LSTM with Attention

In the baseline model, the decoder is an LSTM with an integrated attention module inspired from (Ramos et al., 2024). The decoding process begins by averaging the ConvNeXt’s image features to get the initial hidden and cell states for the LSTM and their dimension is set to 512. At each time step, the LSTM’s current hidden state is passed to an attention module along with the encoder’s image features. The attention module learns to compute a relevance score for each image feature based on the hidden state. The scores are normalized using a SoftMax function to get attention weights which are multiplied element-wise to the encoder’s image features and then summed to get an attention-weighted context vector which highlights the relevant regions of the image to predict the next word. The weighted context vector is passed through a gating mechanism based on the hidden state which allows the model to reduce its focus on features that it has already described.

As shown in Figure 2, the gated attention-weighted context vector is concatenated with the embeddings of the previous word which is embedded using a standard embedding layer from PyTorch and has a dimension of 512. This concatenated vector is fed into a LSTM cell along with the hidden and cell states to return updated hidden and cell states. The updated hidden state is passed through a fully connected layer to get the logits across the entire vocabulary. These logits are used to calculate the loss during training or generate the next word for inference. This iterative process continues to calculate the prediction scores across the vocabulary for each word in the caption until the <end> token is generated or the maximum caption length is reached.

### 3.2.3. Proposed Decoder: Transformer Decoder

This study proposes a transformer decoder displayed in Figure 5 to replace the LSTM. The true encoded captions are embedded using a standard embedding layer and have a dimension of 512. Since the decoder processes all the words in parallel, it is not aware of the position of each word in the sequence hence the caption embeddings are passed through a positional encoding module which incorporates sine and cosine signals in the embeddings to provide the model with positional information for each word.





The core of the decoder is a stack of six transformer decoder layers and each layer has eight heads for multi-headed attention. Positional encoded embeddings of the caption along with image features from the encoder are passed to the transformer decoder where each head in each layer contains two attention mechanisms. In self-attention, the head does a dot product of the query vector of the current word with key vectors of the previous words to get attention weights that are multiplied with the corresponding value vectors to get a weighted sum of value vectors which tells the head where to focus on in the previous words of the caption while generating the current word. A mask is applied to prevent the head from focusing on future words. The weighted sum of value vectors becomes the enriched query vector in cross-attention, where the process is repeated using key, value vectors derived from image features from the ConvNeXt. The weighted sum of value vectors in cross-attention tells the head where to focus on in the image while generating the current word. This process is done in every head while generating each word, allowing the transformer to get simultaneous, diverse views of the previous words and input image while generating the current word. In teacher forcing, the true caption is embedded whereas in non-teacher forcing all the previously generated words are embedded, followed by positional encoding before being fed into the transformer decoder along with the image features to generate the caption. The word with the highest score is appended to the generated caption which acts as the updated input for the next step. This iterative process continues until the <end> token is generated or the maximum caption length is reached.

The output from the transformer decoder is a sequence of feature vectors each with a dimension of 512. This output is passed through a fully connected layer which maps these features to logits across the vocabulary for each word position in the sequence. These logits are used to calculate the loss during training or generate words at the time of inference.

## 3.3. Experimental Design

This section outlines the experiments that were conducted at each stage corresponding to the project’s objectives.

### 3.3.1. Training Strategies

The models were trained using teacher forcing and non-teacher forcing (autoregressive) training strategies in the decoder. While replicating the original codebase (Ramos et al., 2024) it was discovered that although their study stated that the model was trained without teacher-forcing and in fact performed better, their codebase only trained the model with teacher forcing. Moreover, at the time of validation and testing, inference was done using teacher forcing in the decoder which is incorrect since at inference time it is assumed that the model does not have access to true captions. In order to address these issues and provide a robust comparison, this study first implements teacher forcing for both decoders during training and inference to replicate what was done in the original study. Then non-teacher forcing is implemented and in separate experiments both the LSTM and transformer decoders are trained with and without teacher forcing. Inference at the time of validation and testing is done the correct way without teacher forcing. The results are compared and the decoder + training strategy with the highest score on the test set is selected for further experiments.

### 3.3.2. Fine-tuning ConvNeXt

Initially the ConvNeXt is frozen to get a baseline performance. This is followed by experiments in which layers of varying depths are fine-tuned. Earlier layers are responsible for general features such as edges and lines whereas later layers recognize more complex patterns that are task specific. In the experiments where the encoder was fine-tuned, it was frozen for the first 20 epochs to allow the decoder’s gradients to reach some stability and avoid corrupting the ConvNeXt’s pre-trained weights with large initial updates. There were four main scenarios while fine-tuning the ConvNeXt.

1. **Frozen encoder – no fine-tuning**: All layers of the ConvNeXt were frozen to replicate the study (Ramos et al., 2024). Both LSTM and transformer decoders were trained with teacher forcing and non-teacher forcing with the frozen ConvNeXt. The decoder + training strategy with the best performance was used for further experiments.
2. **Fine-tuning layers 5-7:** The last three layers of the ConvNeXt were fine-tuned. A learning rate of 1×10-4 was used with a patience of 20 epochs for early stopping. To investigate the effects of gradual convergence, two additional experiments were conducted using lower learning rates of 1×10-5 and 1×10-6, both with an increased patience of 40 epochs.
3. **Fine-tuning layers 3-7:** To explore the impact of fine-tuning deeper into the network to determine if adapting intermediate layers that combine simple features into more complex patterns will result in an improvement. A learning rate of 1×10-4 with a patience of 20 epochs was used.
4. **Fine-tuning layers 1-7:** The entire ConvNeXt encoder was fine-tuned. To ensure stable convergence and prevent significant alteration of the pre-trained weights, a low learning rate of 1×10-6 was used with a patience of 40 epochs.

For each experiment, the best performing checkpoint on the validation set during training was saved and its results on the validation set were used for comparison.

### 3.3.3. Supplementary Analysis of Attention Regularization

In the study (Ramos et al., 2024), a doubly stochastic attention regularization loss was added to the total loss of the LSTM decoder. Figure 16 in appendix A4 shows that regularization loss was calculated by finding the difference between the perfect sum of attention weights and the sum of the model’s attention weights, squaring it and then averaging it across all the images. This encourages the model to focus on all parts of the image rather than just one area. Initially this study implements regularization only for the LSTM decoder. However, a qualitative analysis of the attention maps revealed that the transformer decoder has a tendency to focus only on a few specific regions throughout the caption generation process. To encourage it to shift its attention across the image, supplementary experiments were conducted in which attention regularization identical to the one used with the LSTM, was added to the training loss of the transformer.

### 3.3.4. Word Embeddings

After selecting the best performing decoder, training strategy and ConvNeXt fine-tuned layers, various word embeddings were evaluated. For a baseline performance the decoder’s embedding layer which mapped the encoded captions to word embeddings was initialized with random weights. A dimensionality value of 512 was chosen for these embeddings and their values were learnt from scratch during the training.

To explore the effect of external knowledge on the model’s performance, two more experiments were conducted in which the decoder’s embedding layer was initialized with vectors from Word2Vec and GloVe models. The Word2Vec embeddings have a dimension of 300 and were pretrained on part of the Google News dataset containing 100 billion words (FSE - HuggingFace, 2021) whereas the GloVe embeddings have a dimension of 200 and were pretrained on 2B tweets, 27B tokens, 1.2M vocab (FSE - HuggingFace, 2021) This provided the model with a rich, pre-existing understanding of word meanings and relationships while mapping the encoded captions to word embeddings, which were then fine-tuned during training to adapt to the image captioning task. The models’ performances were compared to the one with random word embeddings.

### 3.3.5. Decoding Strategy

During training without teacher forcing and at the time of inference, the decoder does not have access to the true caption and relies on its own outputs to generate the next word. In order to select the next word, this study implements greedy search and beam search. In all the experiments greedy search is used at the time of inference on the validation and test sets for quantitative analysis since it provides a computationally efficient baseline for calculating evaluation metrics (Shi et al., 2024). For qualitative analysis and generating captions for a sample of unseen images, beam search with a beam size of k=5 was used to allow the model to showcase its full generative potential. Lastly, the best performing model from this study is used to generate captions using both greedy and beam search to investigate which decoding strategy produces higher quality captions.

## 3.4. Training and Hyperparameters

The models are trained for 120 epochs with an early stopping set if there is no improvement in the validation score for 20 epochs. Cross-entropy is used as the loss function. The optimizers for both the encoder and decoder are Adam. The learning rate for the decoder optimizer is 1×10-4 whereas for the encoder optimizer the values 1×10-4, 1×10-5 and 1×10-6 are tested. Learning rates are scaled down to 80% if there is no improvement for 8 epochs. After every epoch a checkpoint containing all the information to resume training from that point is saved. The best checkpoint based on the validation score is saved and updated throughout the training process. Once training is complete, the best checkpoint is used for testing on the test set.

To train the architecture, images are loaded in batches of size 32 using the dataloader. For each batch, images are passed to the ConvNeXt which extracts features for every image. These image features along with their respective captions and caption lengths are passed to the decoder. For each image, at every word position the decoder calculates logits across the entire vocabulary to generate a caption. The resulting logits which are the model’s prediction are compared against the ground truth word labels in the cross-entropy loss function which calculates the average loss across all the predicted tokens in the batch. The loss is backpropagated throughout the architecture, and optimizers for the encoder and decoder update their respective model’s parameters. To avoid exploding gradients and ensure stable model convergence, gradient clipping is applied to clamp the parameters of both the encoder and decoder optimizers to a threshold of 5. This iterative process is repeated to update the parameters of the entire architecture after every batch in each epoch.

In the case of teacher forcing, the length of the generated caption and the true caption are always same. To ensure that loss calculation is performed only on relevant tokens, padding tokens following the <end> token are excluded from the predicted logits and ground truth labels. In non-teacher forcing, there is a possibility that the generated caption may be shorter or longer than the true caption. To align the predicted logits and ground truth word labels for loss calculation on only the relevant tokens, the length of the generated caption till the <end> token is calculated and used to slice the predicted logits and true caption followed by a non-padding mask to ensure that padding tokens are filtered out.

The models are trained using two NVIDIA A-100 40GB GPUs. A multi-GPU training system is set up using PyTorch’s Distributed Data Parallel (DDP) package in which the number of batches are divided amongst the GPUs. Each GPU calculates its local loss and during backpropagation the gradients of this local loss with respect to every parameter on that specific GPU are calculated. PyTorch’s DDP collects the gradients for all model parameters from both the GPUs and averages them before broadcasting them. This ensures that on each GPU, the model’s weights are updated with a globally consistent gradient.

## 3.5. Evaluation Metrics

The architecture’s performance was measured using a set of evaluation metrics that provide strong insights about the training process and quality of generated captions.

Cross-entropy loss was calculated to understand how well the model’s predicted probability distribution aligns with the ground truth words. A lower loss means a higher probability assigned to the correct word indicating a more confident model. Moreover, top-5 accuracy was evaluated which represents the percentage of correctly predicted words for which the true word existed in the decoder’s top five most probable predictions. A high top-5 accuracy score indicates that although the decoder might not always select the correct word, it consistently ranks the correct word very highly. Both the loss and top-5 accuracy are calculated for each batch and then averaged over an entire epoch. They are calculated at training, validation and test times.

At inference time, the quality of the generated captions is measured quantitatively by BLEU-scores which measure how many n-grams i.e., contiguous sequence of words in the generated caption also appear in the ground truth captions. As shown in Figure 17 in appendix A4, it calculates a precision score based on the overlapping n-grams from unigram to 4-gram resulting in BLEU-1, BLUE-2, BLUE-3 and BLUE-4 scores respectively. A brevity penalty is applied to captions that are too short. Higher BLEU scores indicate a greater overlap between the generated and ground truth captions which means that the generated caption is grammatically correct and has captured most of the semantic. A single BLEU score is calculated by aggregating the n-gram overlap and brevity penalty over the all the tokens in the validation/test set, comparing each of the five generated captions against the set of five true captions for each image. This provides a robust singular value, which represents the overall quality of the model’s output. In this study, BLEU-4 score is used during validation to assess the performance of the architecture when checking for early stopping and saving the best checkpoint. It is also used as the primary evaluation metric when comparing models and experiments.

To qualitatively assess the model’s performance, the best performing checkpoint during training is used to generate captions for unseen images from the test set. The generated captions are assessed against the images and their actual captions to determine their quality. The codebase (Ramos et al., 2024) has a function which generates the caption using beam search with the LSTM decoder and stores the attention weights while predicting each word to display the model’s focus at each time point using attention maps. However, attention maps were not included in their paper. Upon further investigation, it was noted that the function had some faults hence this study fixes those faults and also adapts it to generate captions and store the average cross-attention weights across all the heads and layers from the transformer decoder. The weights are used to generate attention maps to assess the model’s focus across the image while generating each word.

# 4. Results

This chapter presents the findings of the experiments conducted corresponding to each project objective.

## 4.1. Preliminary Results

### 4.1.1. Results on Flickr8k

The system is initially run on the Flickr8k dataset to test the pipeline’s robustness and models’ convergence on a smaller dataset. Using the default training setup with teacher forcing in training, frozen ConvNeXt and decoder learning rate as 1×10-4, initial experiments are run with the LSTM and transformer decoder.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Test loss** | **Test top5 acc** | **Bleu-1** | **Bleu-2** | **Bleu-3** | **Bleu-4** |
| **LSTM + Att** | 3.25 | 71.11 | 67.03 | 45.18 | 28.60 | 17.46 |
| **Transformer** | 2.76 | 71.81 | 66.45 | 44.95 | 28.98 | 18.09 |

**Table 1.** Test metrics on Flickr8k

Table 1 shows the metrics of the best performing checkpoint tested on the test set. The transformer decoder is a more confident model since it has a lower loss value. It also has a slightly higher BLEU-4 score suggesting that it produces captions with a greater 4-gram overlap with the ground truth captions. However, it is important to note that these results are with teacher forcing at inference which is a flaw in the original study’s codebase (Ramos et al., 2024) and was fixed for later experiments once the pipeline’s robustness was confirmed.

### 4.1.2. Results with Original Study’s Codebase

Both decoders are trained with teacher forcing on the MS COCO dataset with inference done using teacher forcing as well to replicate the codebase of the original study. The results obtained along with the results of (Ramos et al., 2024) are displayed in Table 2. The original study only states the validation bleu scores of the best performing checkpoint during training.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Training loss** | **Validation loss** | **Bleu-1** | **Bleu-2** | **Bleu-3** | **Bleu-4** |
| **LSTM + Att (Ramos et al., 2024)** |  |  | 73.17 | 53.16 | 36.17 | 24.63 |
| **LSTM + Att** | 2.57 | 2.80 | 73.22 | 53.24 | 36.63 | 24.71 |
| **Transformer** | 1.94 | 2.14 | 73.58 | 53.96 | 37.73 | 25.86 |

**Table 2.** Training and validation metrics on MS COCO with teacher forcing at training and inference

It can be observed that the metrics are comparable to the ones in the paper (Ramos et al., 2024) showing that the models are learning and this study was able to replicate their work for comparison. The transformer decoder built in this study achieved lower loss values and slightly higher bleu scores suggesting that it is a more confident model producing accurate captions as compared to the LSTM decoder.

## 4.2. Baseline Performance on MS COCO

After validating the system’s robustness, this study fixes the flaws of the original study’s codebase by implementing training and inference without teacher forcing. In all future experiments, inference is done using the model’s own predictions. In these initial experiments, the ConvNeXt is frozen to get a baseline performance and to select the best performing training strategy and decoder combination. The decoder learning rate is 1×10-4.

### 4.2.1. The Impact of Training Strategies

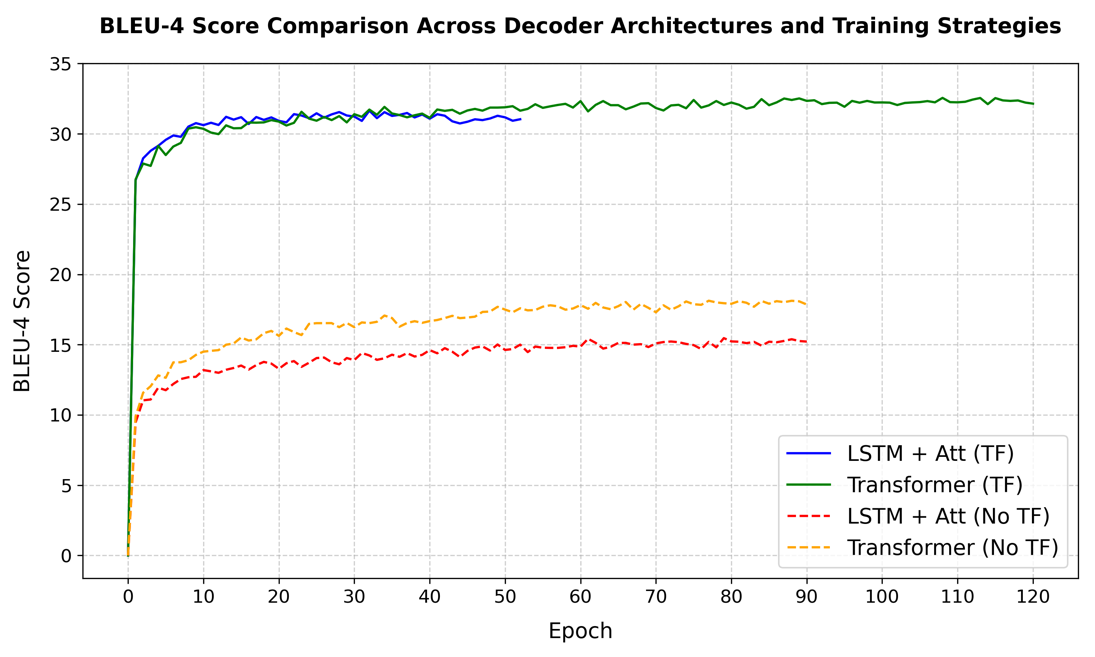
To investigate the impact of training strategies on the model’s performance, both the LSTM and transformer decoders are trained with teacher forcing and without teacher forcing.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **Train loss** | **Train**  **top5 acc** | **Val loss** | **Val top5 acc** | **Bleu-1** | **Bleu-2** | **Bleu-3** | **Bleu-4** |
| **No TF** | **LSTM + Att** | 4.20 | 52.81 | 4.53 | 48.15 | 66.60 | 42.63 | 26.05 | 15.46 |
| **Transformer** | 3.48 | 51.59 | 3.85 | 48.63 | 67.63 | 45.16 | 29.03 | 18.37 |
| **LSTM + Att (Ramos et al., 2024)** | - | - | 2.78 | 77.80 | 74.79 | 58.07 | 44.61 | 34.76 |
| **TF** | **LSTM + Att** | 2.64 | 79.42 | 7.38 | 31.96 | **76.78** | **58.10** | 43.39 | 31.65 |
| **Transformer** | 1.91 | 81.03 | 8.10 | 31.61 | 76.16 | 58.01 | **43.86** | **32.56** |
| **LSTM + Att (Ramos et al., 2024)** | - | - | - | - | 73.17 | 53.16 | 36.17 | 24.63 |

**Table 3.** Training and validation metrics training with and without teacher forcing (TF)

Table 3 shows the training and validation metrics of the best performing checkpoint during training along with the best performing checkpoint from (Ramos et al., 2024). In the case of teacher forcing for both decoders, the validation loss is much higher than the train loss and the validation top 5 accuracy is much lower than the train top 5 accuracy. This is due to the mismatch in data distribution that the model is exposed to during training and inference resulting in the exposure bias problem. Previous studies (Ranzato et al., 2016) and (Rennie et al., 2017) describe this as while training with teacher forcing, the model is exposed to the ground truth captions at every point and never learns to recover from its mistakes. However, during inference the model uses its own outputs to generate the next word and if it makes a mistake then a context is created to which the model was never exposed to during training and the error gets compounded resulting in poor performance at inference. On the other hand, when training without teacher forcing, the distribution that the model is exposed to during training is similar to that at inference and the model learns to recover from its own mistakes during training. Hence at inference time it is more confident in its predictions.

Moreover, in Table 3 it can be observed that in the original study (Ramos et al., 2024), the LSTM decoder performs much better in terms of BLEU scores when trained without teacher forcing as compared to when trained with teacher forcing. However, it is important to note that in their codebase, there no implementation of training without teacher forcing and inference was done with teacher forcing which is incorrect. This study fixes those issues and Table 3 shows that training without teacher forcing results in lower BLEU scores as compared to training with teacher forcing. This can be explained due to slow convergence caused by gradient instability since the model relies on its own output to predict the next word. An error in prediction can be compounded for each subsequent word resulting in gradients being updated in the wrong direction and the model taking time to converge resulting in less accurate captions and lower BLEU scores. Whereas in the case of teacher forcing, the decoder is given the correct word at each time point resulting in stable, faster convergence and higher BLEU scores.



**Figure 6.** BLEU-4 curves for LSTM and transformer decoders training with and without teacher forcing (TF)

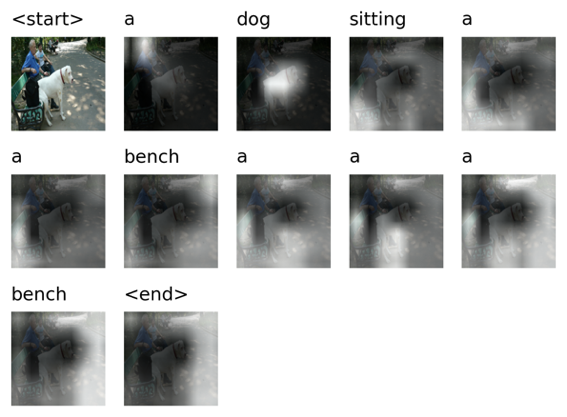
Figure 6 backs this argument since decoders trained without teacher forcing display slow convergence and lower BLEU-4 scores as compared to training with teacher forcing. The experiment ran till 90 epochs for both decoders after which it was timed-out by the system since training without teacher forcing requires waiting for the model’s output at each step which is time-consuming. However, the models were yet to converge as they did not stop due to early stopping and showed small improvements after every other epoch. On the other hand, in the case of training with teacher forcing for both decoders majority of the improvement in BLEU-4 scores took place till epochs 20-30 after which they reached convergence. The LSTM decoder stopped at epoch 53 due to early stopping whereas the transformer decoder showed small improvements till epoch 120.

To qualitatively analyze which training strategy gives more accurate captions, the best checkpoints of each model and strategy are used to generate captions of unseen images and are compared.

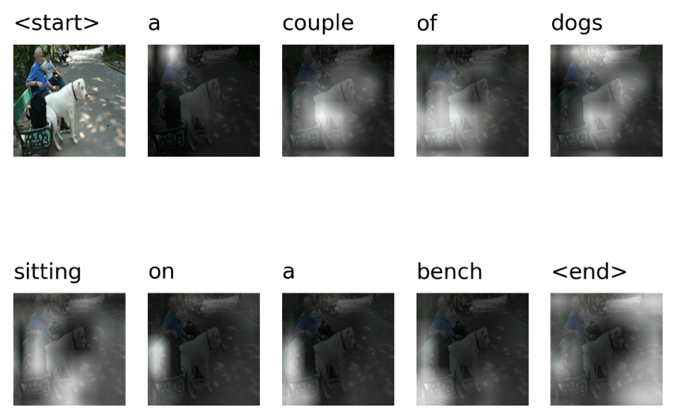
|  |  |  |
| --- | --- | --- |
|  | **Decoder + Strategy** | **Generated Caption** |
| (LSTM + Att) + No TF | A dog sitting a a bench a a a bench |
| Transformer + No TF | A white dog a a a a a a |
| (LSTM + Att) + TF | A couple of dogs sitting on a bench |
| Transformer + TF | A white dog is sitting on a bench |
| True Captions | 1. A large white dog is sitting on a bench beside an elderly man  2. A large white dog sits on a bench with people next to a path  3. A large dog sits just his bottom on a park bench  4. A dog sitting on a bench next to an old man  5. A couple of people sitting on a bench next to a dog |

**Table 4.** Captions generated by LSTM and transformer decoder with and without TF

Table 4 and Table 13 in appendix B1 show that the captions generated by both decoders trained without teacher forcing do not make sense grammatically and are incomplete sentences. This relates to the fact that without teacher forcing, the models have unstable training and low BLEU scores resulting in poor captions. However, with teacher forcing both decoders generate grammatically correct and complete captions since they are able to achieve convergence and display higher BLEU scores. The LSTM decoder generates a caption that is partially correct whereas the transformer decoder generates a correct caption which can be further detailed.



**Figure 7a.** Attention map of LSTM decoder without teacher forcing



**Figure 7b.** Attention map of LSTM decoder with teacher forcing

Figures 7a and 7b show that for the initial stop word ‘a’, both training strategies focus on the same spot in the image. In Figure 7a, for the next word, training without teacher forcing focuses on the dog in the image while generating the word ‘dog’ which shows that the model is learning initially however, for all the remaining words the model’s focus gets stuck in the same regions and it is unable to generate a correct caption. The reinstates the idea that training without teacher forcing results in unstable training and poor final weights which causes the model to make mistakes while generating captions. In Figure 7b, when trained with teacher forcing, the model makes a mistake while predicting multiple dogs as it focuses on another white region in the background and might have misunderstood it as another dog. However, during training since the model was able to reach convergence and obtain optimal weights, while predicting the words ‘sitting’ and ‘bench’ it is able to overcome its prior mistake and focuses on relevant regions in the image allowing it to generate a grammatically correct caption.

For both decoders, although training with teacher forcing results in exposure bias, it gives higher BLEU scores and accurate captions which is the more appropriate metric for NLP tasks hence it is selected as the training strategy for further experiments.

### 4.2.2. Selecting the Best Decoder

The LSTM + attention and transformer decoders are integrated separately in the image captioning architecture with ConvNeXt and trained with teacher forcing. Their best checkpoints during training based on the validation BLEU-4 score were saved and tested on the test set.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Decoder** | **Test Loss** | **Test top5 Acc** | **Bleu-1** | **Bleu-2** | **Bleu-3** | **Bleu-4** |
| **LSTM** | 6.79 | 31.42 | 75.85 | 56.65 | 41.98 | 30.47 |
| **LSTM + Att** | 7.39 | 31.94 | 76.48 | 57.83 | 43.21 | 31.66 |
| **Transformer** | 8.10 | 31.85 | 75.96 | 57.61 | 43.51 | **32.34** |

**Table 5.** Test metrics of LSTM and Transformer decoders

According to Table 5, the LSTM + Att decoder performs slightly better in terms of test loss and top 5 accuracy making it a more confident model. The BLEU scores are quite similar for both decoders however, the transformer decoder has the higher BLEU-4 score of 32.34. Table 4 and Table 13 in appendix B1 also show that for some images the transformer decoder generates slightly more detailed captions. This supports the original hypothesis of the study that since transformers process the image features and previously generated words in parallel using multiple heads to generate the next word, they are able to capture both local and global contexts allowing them to generate more accurate captions. However, the LSTM decoder has similar performance due to its integrated attention module which helps it to focus on relevant parts of the image features while generating each word providing it with the appropriate context. An additional experiment with the LSTM decoder without the attention module was conducted to test this theory and it can be observed that it has lower BLEU scores.

|  |  |
| --- | --- |
|  |  |
|  |  |

**Figure 8.** Attention maps of LSTM + Att decoder (left) and Transformer decoder (right)

Figure 8 shows that multiplying the attention weights calculated by the attention module in the LSTM + Att model with the image features from the ConvNeXt, allows the model to focus on relevant regions in the image at each time point which results in a grammatically correct and somewhat accurate caption. On the other hand, for the first image although the transformer decoder focuses on the dog while generating the word ‘white’, it mainly focuses on a general region in both images throughout the caption generation process unlike the LSTM which shifts its focus. Despite this the transformer is able to generate a relatively accurate caption and this can be explained by the fact that it relies on two attention mechanisms. The first is cross-attention which similar to the attention module in the LSTM tells the model where to focus on the image at each time point and its weights are displayed in the attention maps. The second is multi-headed self-attention which guides the transformer on where to focus on in the previous words. Figure 8 shows that the cross-attention mechanism finds a good general region of the image to focus on to give the transformer relevant visual information however, the transformer may mainly rely on its multi-headed self-attention mechanism to generate an accurate caption. The attention maps provide a valuable insight that both decoders are learning in their own ways.

BLEU scores are more relevant metrics for NLP tasks and since BLEU-4 captures the highest n-gram overlap, it is the deciding metric for generating captions similar to the true captions. Since the transformer decoder has a higher BLEU-4 score of 32.34, it is selected as the decoder for further experiments. The theme of the transformer decoder outperforming the LSTM in terms of BLEU-4 scores during training and validation can also be observed in Table 3 and Figure 6.

## 4.3. The Impact of Fine-tuning ConvNeXt

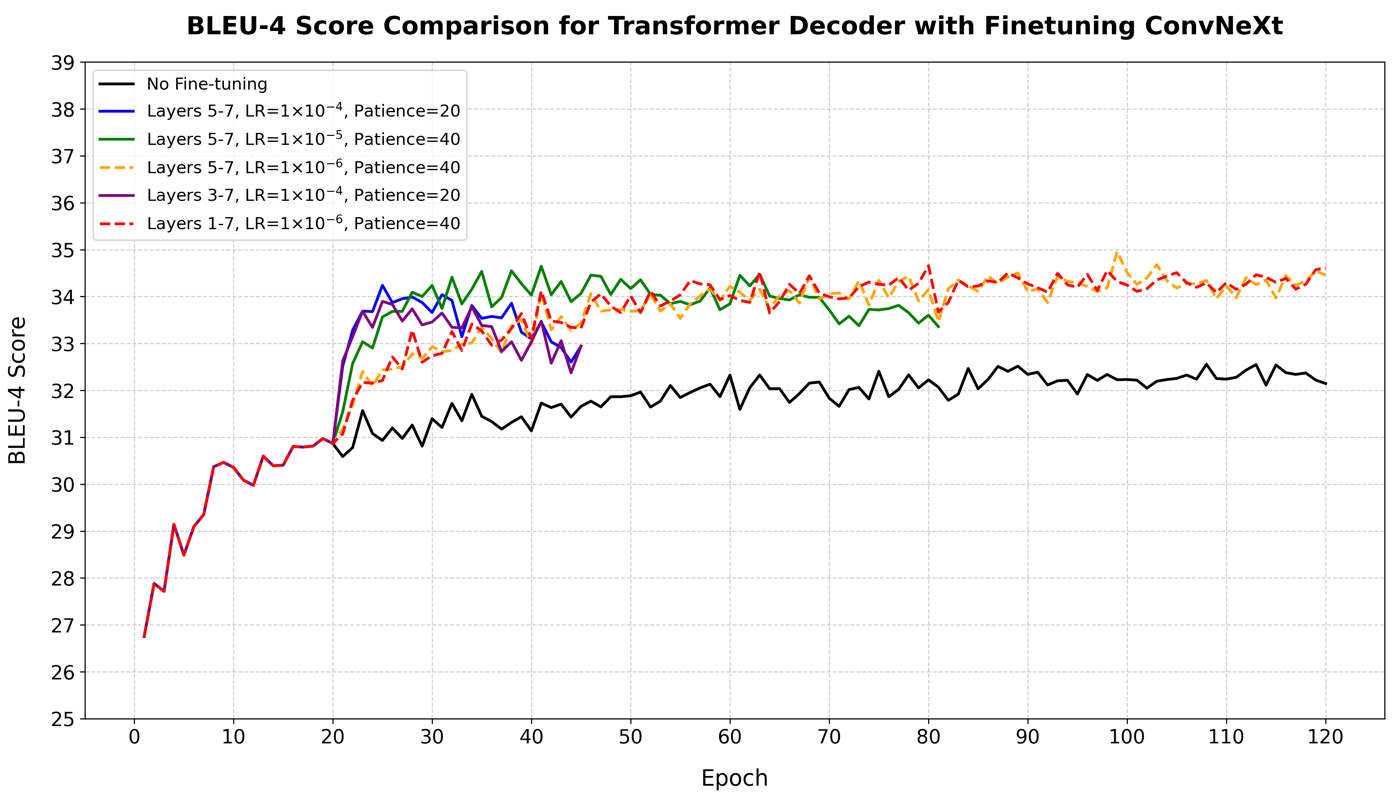
### 4.3.1. Quantitative Analysis

Different layers of the ConvNeXt are fine-tuned to investigate whether updating the weights for the image captioning task improves the quality of generated captions and if so then what are the optimal layers to fine-tune. These experiments are done using the transformer decoder trained with teacher forcing. Apart from the layers being fine-tuned, the learning rate along with the patience for early stopping are adjusted for smaller, more gradual updates. The ConvNeXt is frozen for the first 20 epochs to avoid early gradients from the transformer decoder corrupting the pretrained weights.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Layers Fine-tuned** | **Learning Rate** | **Patience (epochs)** | **Val loss** | **Val top5 acc** | **Bleu-1** | **Bleu-2** | **Bleu-3** | **Bleu-4** |
| None | - | 20 | 8.10 | 31.61 | 76.16 | 58.01 | 43.86 | 32.56 |
| 5 – 7 | 1×10-4 | 20 | 7.76 | 32.63 | 77.74 | 60.04 | 45.77 | 34.24 |
| 5 – 7 | 1×10-5 | 40 | 7.96 | 32.72 | 78.02 | 60.53 | 46.28 | 34.65 |
| 5 – 7 | 1×10-6 | 40 | 8.11 | 32.39 | 77.90 | 60.38 | 46.32 | **34.96** |
| 3 – 7 | 1×10-4 | 20 | 7.77 | 32.52 | 77.52 | 59.84 | 45.50 | 33.90 |
| 1 – 7 | 1×10-6 | 40 | 7.96 | 32.37 | 77.93 | 60.37 | 46.18 | 34.66 |

**Table 6.** Validation metrics of fine-tuning different layers of ConvNeXt

Table 6 shows that fine-tuning the ConvNeXt does not improve validation loss and top 5 accuracy by much since they are limited by the exposure bias problem however it does improve the model’s performance in terms of BLEU scores. It is interesting to see that fine-tuning deeper layers does not necessarily improve BLEU scores since fine-tuning layers 5-7 has a slightly higher score as compared to fine-tuning layers 3-7 and layers 1-7 while keeping the learning rate and patience constant. This shows that for the task of image captioning, it can be hypothesized that fine-tuning shallower layers that are responsible for more complex features is enough since deeper layers are responsible for simple features like edges and lines which are consistent for any visual task hence fine-tuning them has no added benefit. Moreover, decreasing the learning rate and increasing the patience for early stopping allows the model to make smaller updates to the pretrained weights preventing from corrupting them and making their values move in the right direction. Due to these reasons, fine-tuning layers 5-7 with a learning rate of 1×10-6 and patience of 40 epochs allows the architecture to achieve the highest BLEU-4 score of 34.96.



**Figure 9.** BLEU-4 curves with fine-tuning different layers of the ConvNeXt

Figure 9 shows that all the line graphs are identical for the first 20 epochs since the ConvNeXt is frozen. After epoch 20, the line graph of the frozen ConvNeXt has the least improvement across all epochs indicating that fine-tuning the ConvNeXt does result in an improvement in BLEU scores. Fine-tuning layers 5-7 and 3-7 with learning rate 1×10-4 and patience 20 shows a steep improvement initially till epoch 24 followed by a sharp decline and then stopping early at epoch 44 suggesting that the ConvNeXt’s pretrained weights might have been updated too aggressively resulting in them entering a non-optimal space and not being able to recover. Fine-tuning layers 5-7 with a slightly lower learning rate of 1×10-5 and patience of 40 epochs shows early signs of improvement as well which flattens out and then gradually declines coming to an early stop around epoch 80. However, fine-tuning layers 5-7 and 1-7 with a very low learning rate of 1×10-6 and patience of 40 epochs shows steady improvements after 20 epochs which continues till epoch 120 and manages to achieve the highest BLEU-4 scores. This shows that small and gradual updates to the pretrained weights results allows them to move in the right direction towards an optimal space without getting corrupted.

### 4.3.2. Qualitative Analysis

For each configuration of fine-tuning, the checkpoint that has the highest validation BLEU-4 score during training is used to generate captions for unseen images which are compared to assess which configuration generates the most accurate captions and whether BLEU-4 scores translate to the actual caption quality.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Layers Fine-tuned** | **Learning Rate** | **Patience** | **Generated Caption** |
| None | - | 20 | A white dog is sitting on a bench |
| 5 – 7 | 1×10-4 | 20 | A man and woman sitting on a bench with a dog |
| 5 – 7 | 1×10-5 | 40 | A man sitting on a bench with a dog |
| 5 – 7 | 1×10-6 | 40 | A white dog sitting next to a park bench |
| 3 – 7 | 1×10-4 | 20 | A couple of people that are sitting on a bench |
| 1 – 7 | 1×10-6 | 40 | A white dog sitting next to a park bench |

**Table 7.** Captions generated by fine-tuning different layers of the ConvNeXt

Observing the generated captions in Table 7 and Table 14 in appendix B2, it can be seen that when the ConvNeXt is not fine-tuned, relatively accurate captions are generated however they lack detail. After fine-tuning, the generated captions contain more detail as they mention the man, woman or people sitting on the bench as well. From Table 4, it can be seen that these details are mentioned in the true captions which explains the higher BLEU scores after fine-tuning. Fine-tuning layers 5-7 or layers 1-7 with the lowest learning rate of 1×10-6 and patience 40 epochs, generate the same caption which does not talk about the people on the bench but captures the detail that the dog is white and the bench is in a park. The qualitative analysis shows that higher BLEU scores do not necessarily mean more accurate captions however, it is important to note that this is the case for a single image and cannot be generalized to all images.

|  |  |
| --- | --- |
|  |  |

**Figure 10.** Attention maps of captions generated by Transformer decoder with fine-tuned ConvNeXt. Layers 5-7 fine-tuned with learning rates 1×10-4 (left) and 1×10-6 (right)

Figure 10 shows that in the attention map on the left, the transformer focuses slightly on the man and woman while generating those words and for the words ‘sitting on a’ some of its attention stays on the bench. However, in both attention maps the transformer decoder focuses on the same general bottom-right region throughout the caption generation process. Comparing this to Figure 8, a general trend can be seen in which unlike the LSTM which shifts its focus on relevant regions, the transformer decoder avoids shifting its cross-attention across the image and focuses on a general region through the caption generation.

Since fine-tuning layers 5-7 with a learning rate of 1×10-6 and patience of 40 epochs gives the highest BLEU-4 score, it is selected as the optimal fine-tuning strategy for further experiments.

## 4.4. Analysis of Attention Regularization

To encourage the transformer decoder to shift its attention across the image, an attention regularization term explained in Section 3.3.3 was added to the loss value while training. To quantitatively analyze any difference in performance, the validation metrics of the best performing checkpoint during training were saved.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Decoder** | **ConvNeXt Layers Fine-tuned** | **Learning Rate** | **Val Loss** | **Val top5 Acc** | **Bleu-1** | **Bleu-2** | **Bleu-3** | **Bleu-4** |
| **Transformer + Regularization** | None | - | 8.63 | 31.59 | 76.26 | 57.99 | 43.71 | 32.39 |
| **Transformer + Regularization** | 5 - 7 | 1×10-4 | 8.45 | 32.46 | 78.06 | 60.49 | 46.06 | 34.39 |
| **Transformer + Regularization** | 3 - 7 | 1×10-4 | 8.58 | 32.43 | 77.73 | 59.94 | 45.76 | 34.30 |
| **Transformer + Regularization** | 5 - 7 | 1×10-6 | 8.65 | 32.40 | 77.65 | 60.16 | 45.99 | 34.47 |

**Table 8.**  Validation metrics of transformer decoder trained with attention regularization

Comparing Table 8 to Table 6 shows that training the transformer decoder with attention regularization across all instances of ConvNeXt fine-tuning only adds to the loss value. There is no major change in the top-5 accuracies or BLEU scores which shows that the transformer decoder’s native attention mechanisms are already effective enough.

|  |  |
| --- | --- |
|  |  |
|  |  |

**Figure 11.** Attention maps of captions generated by Transformer decoder (with regularization) without fine-tuning ConvNeXt (top left), fine-tuning layers 3-7 with learning rate 1×10-4 (bottom left), fine-tuning layers 5-7 with learning rates 1×10-4 (top right) and 1×10-6 (bottom right)

Comparing Figure 8 with Figure 11, it can be seen that with and without regularization in the transformer decoder when the ConvNeXt is not fine-tuned, the attention maps are similar in which the model focuses on the dog while generating the word ‘white’ but for the rest of caption its focus remains on a general region in the image. Comparing Figure 10 with Figure 11, it is observed that the attention maps of the transformer decoder with the fine-tuned ConvNeXt with and without regularization are also quite similar. The model does slightly focus on the man, woman and people while generating those words and shifts its attention to the bench while generating ‘sitting on a'. However, for the entire caption generation process most of its attention is fixated on the bottom-right region of the image. This is an interesting insight because it shows that even with regularization, the transformer model learns differently from the LSTM by relying both on its cross-attention and self-attention mechanisms. Since regularization does not result in an improvement in performance, in further experiments the transformer decoder is trained without attention regularization.

## 4.5. The Impact of Pretrained Word Embeddings

The next set of experiments explore the impact of using pre-trained word embeddings to investigate whether representing the captions with prior linguistic knowledge results in contextually rich generated captions.

### 4.5.1. Quantitative Analysis

The embeddings layer in the transformer decoder was initialized with random word embeddings for a baseline performance. The experiment was repeated with pretrained Word2Vec and GloVe embeddings. In all three experiments, the embeddings were fine-tuned during the training process. The best checkpoint on the validation set was saved during the training process and tested on the test set.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Embeddings** | **Test Loss** | **Test top5 Acc** | **Bleu-1** | **Bleu-2** | **Bleu-3** | **Bleu-4** |
| **Random** | 8.09 | 32.41 | 77.63 | 59.90 | 45.85 | 34.42 |
| **Word2Vec** | 7.67 | 32.92 | 78.43 | 60.65 | 46.36 | 34.74 |
| **GloVe** | 7.39 | 32.96 | 78.56 | 60.69 | 46.43 | **34.77** |

**Table 9.** Test metrics with random and pretrained word embeddings

Table 9 shows that although the models with pretrained word embeddings have better metrics, the improvement in performance is minimal. The model with GloVe embeddings has the best performance with a BLEU-4 score of 34.77 on the test set and 35.06 on the validation set further building upon the results of the optimal ConvNeXt fine-tuning strategy in Section 4.3.1. However, it is important to note that BLEU scores only capture the n-gram overlap of the generated caption with the actual captions and do not take into account the semantic richness of the captions. Hence a caption that is contextually aware may have a low BLEU score if it has a low n-gram overlap with the true captions which makes it important to qualitatively assess the generated captions.

### 4.5.2. Qualitative Analysis

The checkpoints that were tested on the test set were used to generate captions for unseen images.

|  |  |  |
| --- | --- | --- |
|  | **Embeddings** | **Generated Caption** |
| Random | A white dog sitting next to a park bench |
| Word2Vec | A man sitting on a bench next to a white dog |
| GloVe | A man sitting on a bench next to a white dog |

**Table 10.** Captions generated by pretrained word embeddings

Table 10 shows that the captions generated by Word2Vec and GloVe embeddings incorporates the context of a man sitting on the bench which was missed by the caption generated using random embeddings however, only the model using random embeddings was able to capture that the bench is in a park. Table 15 in appendix B3 shows captions of more sample images and in most cases, the captions generated by all three embeddings are the same. Hence it cannot be said definitively that incorporating prior linguistic knowledge while representing captions generates higher quality captions.

## 4.6. Greedy Search vs Beam Search

Both greedy and beam search are used with the best performing model to generate captions for unseen images to determine which strategy generates higher quality captions. Beam size k=5 is used for beam search.

|  |  |  |  |
| --- | --- | --- | --- |
| 1 |  | **Strategy** | **Generated Caption** |
| Greedy | A small blue and white airplane is parked in the grass |
| Beam | A small blue and white airplane on a grassy field |
| 2 |  | Greedy | A motorcycle parked in a dirt field with a fence in the background |
| Beam | A motorcycle that is parked in the dirt |
| 3 |  | Greedy | A man sitting on a bench with a dog |
| Beam | A man sitting on a bench next to a white dog |
| 4 |  | Greedy | A traffic light with a sky background and clouds in the background |
| Beam | A traffic light with a sky background |
| 5 |  | Greedy | A table with a vase of flowers on it |
| Beam | A table with a vase of flowers on it |
| 6 |  | Greedy | A group of people standing around a street |
| Beam | A group of people walking down a sidewalk |
| 7 |  | Greedy | A herd of sheep standing in a field next to a barn |
| Beam | A herd of sheep standing on top of a lush green field |
| 8 |  | Greedy | A boat with flags hanging from the side of it |
| Beam | A boat filled with flags floating on top of a river |
| 9 |  | Greedy | A group of giraffes grazing in a field |
| Beam | A group of giraffes grazing in a field |
| 10 |  | Greedy | A fire hydrant spraying water onto a street |
| Beam | A fire hydrant spraying water onto a street |

**Table 11.** Captions generated by Greedy Search vs Beam search

Table 11 shows that for image numbers 2 and 7 greedy search generates more accurate captions that have a higher level of detail. Whereas for image number 3, beam search generates a more detailed caption. For image number 5, 9 and 10 the captions generated by both decoding strategies are the exact same and for the remaining images they are very similar. Hence both decoding strategies perform equally well in terms of caption quality. It would have been a more informative comparison if both strategies were tested on the test set and their evaluation metrics were compared however, the computational cost of beam search along with the code complexity of handling multiple beams for multiple images in a batch was a limitation.

# 5. Discussion

This chapter provides a comprehensive analysis of the experimental results presented in Chapter 4. It examines the findings in the context of each project objective, critiques the methods and discusses their generalizability and implications in the field of image captioning.

## 5.1. Addressing Project Objectives

This section directly assesses how the results address each project objective.

### 5.1.1. Implement the ConvNeXt encoder, LSTM and Transformer decoders

This objective was achieved in the early stages of the project. As mentioned in section 3.3, the ConvNeXt encoder and LSTM decoder were adapted from the original study’s codebase (Ramos et al., 2024) whereas the transformer decoder was developed as the proposed decoder in this study. All models were built using PyTorch. The encoder was connected to each decoder separately which made the entire architecture of the system. The proof of this objective being achieved is shown in section 4.1.1 and 4.1.2 which present the results of the initial runs conducted using the Flickr8k and MS COCO datasets respectively to ensure that data was passing through the implemented architecture accurately and the models were learning effectively.

### 5.1.2. Compare the model’s performance training with and without teacher forcing

This objective was achieved as presented in section 4.2.1. Both the LSTM and transformer decoders were trained with and without teacher forcing and the architecture’s performance was compared. It is important to note that the original codebase did not include training without teacher forcing hence this study implemented it from scratch. The decoders were trained using both training strategies with the experimental setup defined in section 3.4. Table 3 shows that training with teacher forcing results in low training loss and high validation loss as compared to training without teacher forcing which displays similar values for training and validation loss. This can be explained due to exposure bias faced by the model when trained using teacher forcing as it is exposed only to the true caption at the time of training and finds it challenging to recover from its own incorrect predictions during inference. However, Table 3 also shows that training with teacher forcing displays higher BLEU scores. Figure 6 helps to explain this observation since BLEU 4 scores of models trained with teacher forcing display fast convergence whereas models trained without teacher forcing display very small improvements and slow convergence. This may be due to unstable gradient updates when the model makes a mistake. Due to slow convergence and longer training times in the case of training without teacher forcing, training was timed out at epoch 90 unable to reach full convergence for both decoders. A qualitative analysis was also performed as displayed in Table 4 in which models trained with teacher forcing generated higher quality captions and also focused on relevant regions of the image shown in Figure 7b.

While the results provide valuable insights about the performance of both training strategies and teacher forcing was selected for further experiments, there were two limitations. Firstly, both decoders should have been trained till convergence in the case of non-teacher forcing as well. However, due to a 72-hour job time limit on the HPC node, training was timed out. Secondly, the transformer decoder had longer training times since it was implemented without key-value caching which meant that at each decoding step, the model had to recompute self-attention over the entire generated sequence from scratch to input the model’s previously generated outputs for the next step. This resulted in O(L3) time complexity with respect to the sequence length which was a substantial performance bottleneck in the case of training without teacher forcing. Implementing key-value caching requires an intricate understanding of PyTorch’s internal workings which was considered to be outside of the scope of this project. Future work should address these limitations for a more robust comparison.

### 5.1.3. Train and select the best decoder on the basis of their baseline performance

This objective was achieved in section 4.2.2 in which both decoders were trained with teacher forcing while the encoder was frozen, and the best checkpoint during training was saved and tested on the test set. Table 5 shows that the LSTM decoder integrated with an attention module had a slightly better performance than the transformer decoder in terms of test loss and top 5 accuracy and both decoders displayed similar BLEU scores. However, the transformer decoder had a slightly higher BLEU-4 score which is why it was selected as the decoder for further experiments. The results represent that the attention module in the LSTM allowed it to focus on relevant parts of the image at every step during decoding and give similar results to the transformer decoder since the LSTM without the attention module had lower BLEU scores. Although the transformer decoder did not shift its attention across the image during the caption generation process, it was able to perform slightly better because in addition to the multi-headed cross-attention mechanism, it also has masked multi-headed self-attention which allows it to attend to all the previously generated words simultaneously while getting diverse views from multiple heads helping it build a richer textual context.

### 5.1.4. Analyze the performance of fine-tuning different layers of the ConvNeXt

The objective of fine-tuning various depths of the ConvNeXt and comparing its performance was achieved in section 4.3. Experiments described in section 3.3.2 were conducted in which initially the ConvNeXt was frozen to get a baseline performance followed by fine-tuning layers 5-7, 3-7 and 1-7 with learning rates 1×10-4, 1×10-5 and 1×10-6. The quantitative results of these experiments are presented in Table 6 and it can be seen that fine-tuning the ConvNeXt regardless of the depth of layers, does not have a substantial improvement in terms validation loss and accuracy however, the BLEU scores specifically the BLEU-4 scores do improve. An important insight noted was that fine-tuning deeper layers results in similar performance as fine-tuning just the shallower layers since deeper layers tend to learn basic features which are common in most visual tasks whereas shallower layers focus more on complex patterns which are task specific making them more relevant to fine-tune. Hence fine-tuning layers 1-7 has no additional benefit. Moreover, Figure 9 shows that fine-tuning with a low learning rate and higher patience allowed the pretrained weights to update gradually and prevented them from getting corrupted resulting in stable improvement in BLEU-4 scores. As a result, fine-tuning layers 5-7 with a low learning rate of 1×10-6 and patience of 40 epochs achieved the highest BLEU-4 score of 34.96 which is a 7.4% improvement from the BLEU-4 score of 32.56 achieved without fine-tuning.

Table 7 shows that fine-tuning the ConvNeXt also improves the quality of generated captions as they include more detail however, the level of detail does not improve by fine-tuning deeper layers. Fine-tuning every sequence of layers from just layer 7, layers 6-7, layers 5-7 and so on till layers 1-7 would have provided a more enriched analysis however, fine-tuning both the ConvNeXt encoder and transformer decoder is very computationally expensive and time-consuming. For example, with the resources available fine-tuning the entire architecture for 120 epochs took 5 days. Hence for practical reasons, reasonable sequences which gave a fair idea of fine-tuning different depths of layers were selected and fine-tuning all sequences can be explored in future works.

### 5.1.5. Train the architecture with pretrained Word2Vec and GloVe embeddings

This objective was achieved in section 4.5 in which the model was trained using pretrained Word2Vec and GloVe embeddings. The quantitative results along with the actual captions generated for sample images were compared. It was discovered that although GloVe embeddings had the highest BLEU-4 score, using pretrained word embeddings does not offer a major improvement in performance. In terms of quality of generated captions, for some images using pretrained word embeddings incorporated details that were missed by random embeddings but they also missed minor details that were included by models using random embeddings. Moreover, in other images the level of detail is identical hence it cannot be said definitively that incorporating prior linguistic knowledge by using pretrained embeddings drastically improves the quality of generated captions.

### 5.1.6. Compare caption quality with greedy search and beam search

This objective was partially achieved in section 4.6 where both decoding strategies were used to generate caption for sample images. For some images, greedy search generated more accurate captions with higher detail whereas for other images beam search generated better quality captions. The remaining images had the same captions from greedy and beam search thus it cannot be said conclusively that one decoding strategy outperforms the other in terms of caption quality. Since beam search considers the top-k sequences at every time point and selects the best one, it was expected that it will outperform greedy search. However, since the model in this study is trained using teacher forcing, it is used to making confident, single-step predictions at each time point and encounters a similar situation with greedy search. Beam search makes the model consider multiple possibilities which it is not used seeing hence it may struggle. This links back to the exposure bias issue faced by the model when it is trained under ideal conditions but is forced to explore sequences it has not observed before at inference. Hence both decoding strategies display equal performance in terms of caption quality. The lack of quantitative comparison on the test set due to computational cost and code complexity of beam search with multiple images is a limitation which did not allow the objective to be fully achieved and can be explored in future works.

## 5.2. Answering the Research Question

Referring to section 5.1, it can be said that by achieving all the objectives the research question “How can fine-tuning a ConvNeXt encoder, combined with an LSTM or Transformer decoder enhance image captioning performance, and what is the impact of teacher forcing, pre-trained word embeddings and decoding strategies on the quality of generated captions?” has been answered successfully. The results show that for the task of image captioning, training with teacher forcing results in faster convergence and captions of higher quality. Moreover, a transformer decoder performs slightly better than an LSTM with an attention module, and fine-tuning only the shallow layers 5-7 of the ConvNeXt is enough to improve the performance of image captioning. Using pretrained word embeddings improves the quality of generated captions in some cases whereas in other, the quality of generated caption is the same as with models using random embeddings. Both greedy search and beam search generate captions of similar quality for the model in this study.

## 5.3. Broader Discussions and Implications

The findings of this study are not restricted to a simple comparison of architectures and training strategies. They extend beyond, offering broader insights into the design and training of modern image captioning systems.

### 5.3.1. Comparing with Existing Literature

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Method** | **BLEU-1** | **BLEU-2** | **BLEU-3** | **BLEU-4** |
| VGGNet + Bi-LSTM (Wang et al., 2016) | 67.2 | 49.2 | 35.2 | 24.4 |
| GoogleLeNet + LSTM (Vinyals et al., 2016) | - | - | - | 32.1 |
| GoogleNet + LSTM + Att (You et al., 2016) | 70.9 | 53.7 | 40.2 | 30.4 |
| Distilled Vision Transformer (Castro et al., 2022) | - | - | - | 34.4 |
| ViTCNN + Att (Cahyono & Jusuf, 2024) | - | - | - | 31.2 |
| ConvNeXt + LSTM + Att (Ramos et al., 2024) | 74.79 | 58.07 | 44.61 | 34.76 |
| **ConvNeXt (Fine-tuned) + Transformer** | 77.90 | 60.38 | 46.32 | **34.96** |
| **ConvNeXt (Fine-tuned) + Transformer + GloVe** | 78.52 | 60.99 | 46.68 | **35.06** |
| CaPtion TransformeR (CPTR) (Liu et al., 2021) | 81.7 | 66.6 | 52.2 | **40.0** |

**Table 12.** Comparison of BLEU scores of previous studies with the results of this study on MS COCO

Table 12 shows that in the earlier stages researchers tried to improve performance by trying different image models. With the introduction of vision transformers, there was some increase in performance by using them as image models instead of traditional CNNs. However, by using the ConvNeXt on the same architecture there was a significant boost in performance. This study was able to improve results by fine-tuning the ConvNeXt and replacing the attention integrated LSTM module with a transformer decoder, which was further improved by incorporating prior linguistic knowledge using pretrained GloVe embeddings to encode captions. However, there are some studies such as CaPtion TransformeR that have achieved state-of-the-art results using a vision transformer as the image model and a transformer decoder. The key in their performance seems to stem from increasing the image resolution to 384x384, dividing the image into patches and reshaping them into a 1-dimensional sequence before passing it into the architecture which are preprocessing steps that should be explored in future works.

### 5.3.2. Architectural and Practical Implications

The primary finding of this study in which the transformer decoder outperforms the LSTM decoder reinforces the trend in deep learning that attention-based (self + cross-attention) models are replacing sequential RNN models for most sequence-to-sequence tasks. The transformer’s ability to capture both local and global contexts along with modelling long range dependencies using its multi-headed attention mechanisms makes it well-suited for the sequential task of using image features to generate structured captions. Hence in future works for general image captioning, transformers should be considered as decoders instead of LSTMs. Moreover, the results obtained by fine-tuning the ConvNeXt show that by fine-tuning a pre-trained vision model, superior performance can be achieved for vision-language tasks without having to train a vision model from scratch. A novel finding of this study is that while fine-tuning, it is sufficient to fine-tune only the middle to later layers of the encoder instead of the entire model since fine-tuning earlier layers do not offer any additional improvement in performance for the task of image captioning. Both of these insights save significant computational resources when developing modern image captioning systems.

### 5.3.3. Training and Methodological Implications

The training strategies compared in this study offer strong methodological implications on the stability of training. The finding that training without teacher forcing results in slow convergence, error compounding and poor results goes against the reported findings of the study (Ramos et al., 2024) and highlights the importance of training with teacher forcing for stable training and faster convergence in modern image captioning systems. However, the exposure bias problem linked to teacher forcing presents a need for combining both training strategies in scheduled sampling in which the model non-deterministically replaces sampling from the ground truth caption with its own predictions during training (Bengio et al., 2015). Moreover, the finding that using pretrained word embeddings and beam search do not result in major improvements in performance can saves computational costs. Using a standard dataset (MS COCO Karpathy split) and directly comparing results to a codebase missing certain reported methods not only contributes a new set of validated, comparable results but also reinforces the importance of code transparency and reproducibility in research. The systematic nature of experiments conducted along with appropriate evaluation metrics and qualitative analysis make this study’s findings highly valid. The results of this study are generalizable to a degree hence similar results can be expected by applying the same architecture to other general purpose image captioning tasks.

# 6. Evaluation, Reflection and Conclusion

## 6.1. Project Evaluation

The overall design of the project was effective in answering the research question. Although the objectives were not too detailed and specific, they were well-defined and achievable within the timeframe. Their direct correlation with the research question ensured that the methods and results were targeted towards gathering enough relevant insights to answer it confidently. A primary challenge which was not specified in the objectives and was faced during the experiments was accurately assessing the quality of generated captions. The methods mention using BLEU scores however, they do not fully capture the contextual accuracy of the generated text. This study aimed to mitigate this issue by qualitatively assessing the actual captions for sample images but it is not feasible to do so for all the images and highlights the need for a better evaluation metric.

The literature review was helpful in understanding the evolution of the encoder-decoder architecture and improvements made to it over the years. This laid a robust foundation for the architectural changes in this study which were highly relevant for the first three objectives. The critical context was however, lacking an in-depth exploration of literature on training strategies, pre-trained word embeddings and decoding strategies which would have provided more insights about the later objectives of this study.

The methods were systematic and findings from each set of experiments were carried forward to the next set allowing previous objectives to contribute to the next one. This approach allowed the study to achieve each objective and obtain a model which outperformed the original study (Ramos et al., 2024). The best training and fine-tuning strategy were selected on the basis of validation metrics to align with the methodology of (Ramos et al., 2024) whereas the better performing decoder and word embeddings were selected on the basis of test metrics. This could have been standardized to avoid confusion. While fine-tuning the ConvNeXt more layers could have been explored however, the computation time was a limitation. Attention maps were not a part of the original project plan but their addition gave valuable insights about how the models learn differently. The objective regarding decoding strategies was limited by the computational cost of beam search and the complexity of its implementation for multiple images in a batch which prevented a quantitative analysis of both strategies to confidently determine which strategy performs better.

## 6.2. Reflection

This project was a significant learning experience especially in terms of project planning and being prepared for potential risks in research. Getting a codebase setup and validating the pipeline through initial experiments using the Flickr8k dataset prior to the official start date set a solid foundation for the rest of the project. A major lesson learnt was validating existing work as a flaw was discovered in the training and inference of the codebase of the original study (Ramos et al., 2024). However, by being ahead of the project timeline, I was able to implement the non-teacher forcing from scratch without adjusting the project plan. The project also provided an opportunity to enhance my theoretical understanding of complex deep learning architectures particularly the Transformer decoder while also developing my proficiency in PyTorch to build these models. I gained an understanding of the complementary challenges of training large models such as building computationally efficient data pipelines, efficient image formats and learning Linux commands to use the HPC node. The biggest achievement apart from the results is learning how to set up multi-GPU training using PyTorch’s Data Distributed Parallel package which significantly sped up training and allowed the study to carry out more experiments.

With the benefit of hindsight, I would have explored the literature to come up with a more relevant evaluation metric that accurately captures the quality of generated captions beyond n-gram overlap with true captions. I would have also dedicated more time to overcoming the implementation challenges of beam search to perform a quantitative analysis against greedy search. Implementing a transformer only architecture to investigate whether the bottleneck in performance lies within the architecture itself would also be an interesting objective. These changes would have provided a richer set of results and strengthened the overall conclusions of the study.

## 6.3. Conclusion

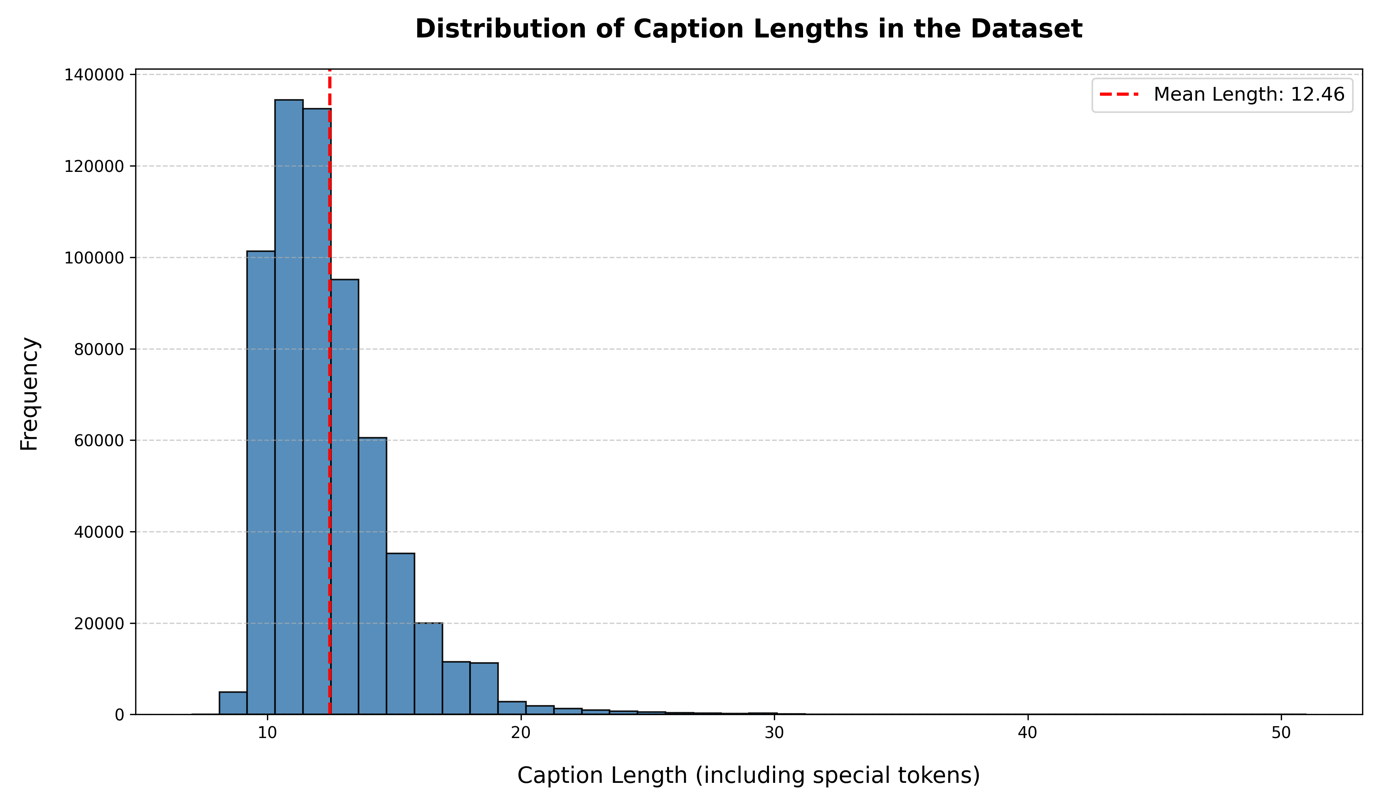
The project successfully demonstrated that in an encoder-decoder architecture with a ConvNeXt as an encoder, the transformer decoder slightly outperforms the LSTM decoder with an attention module in terms of BLEU-4 scores. The key takeaway is that attending to relevant regions in the image and previously generated words at each step in the caption generation process plays a significant role in generating accurate captions. The empirical and qualitative evidence from this study suggests that training with teacher forcing is superior in terms of training stability, convergence and BLEU scores as opposed to training without teacher forcing. Fine-tuning the shallow layers of ConvNeXt is good enough to see an improvement in performance whereas pre-trained word embeddings and beam search decoding do not offer significant improvement in performance. Based on these conclusions, future work should focus on exploring scheduled sampling training strategies and advanced decoding strategies beyond greedy and beam search to see if they can improve the diversity of generated captions. Additionally, the architecture can be further modified by replacing the ConvNeXt with other pre-trained vision backbones or comparing this architecture with a full Transformer network to evaluate how they perform in similar conditions.

# References

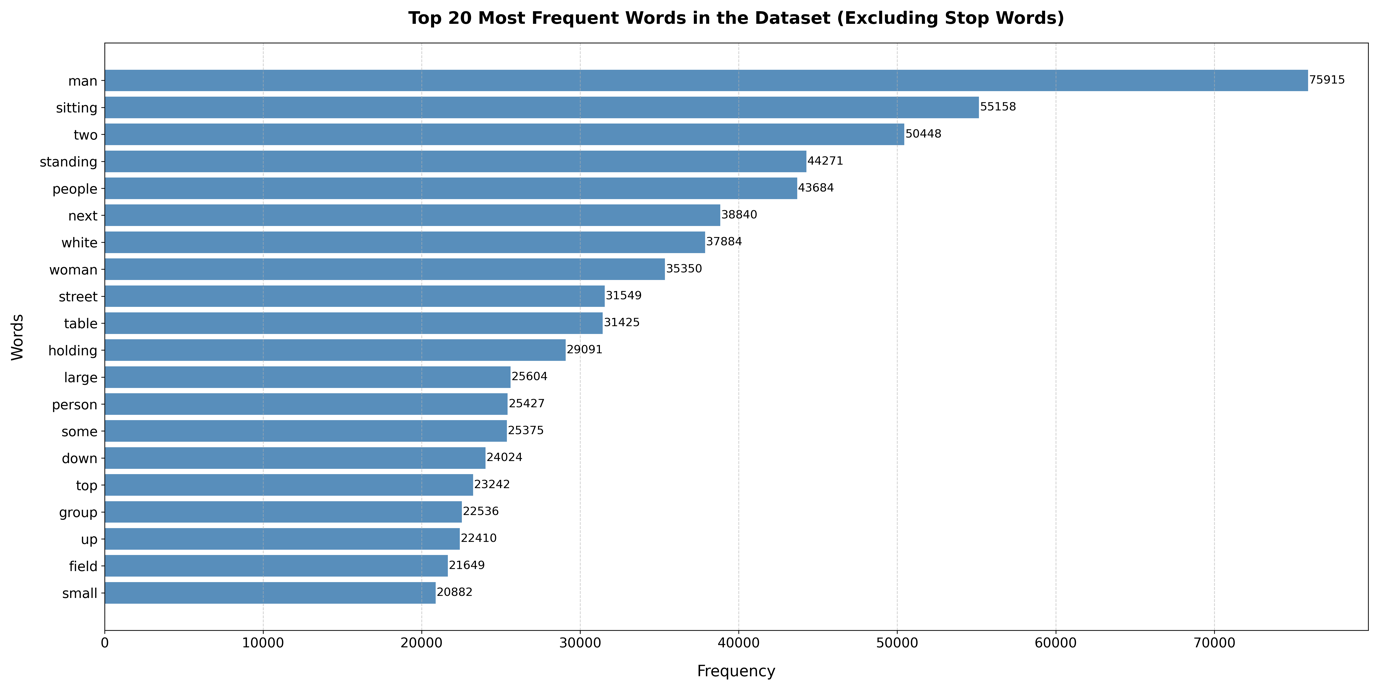
1. Bai, S., & An, S. (2018). ‘A survey on automatic image caption generation’. *Neurocomputing*, *311*, 291–304.
2. Chen, D., Cahyawijaya, S., Ishii, E., Chan, H. S., Bang, Y. and Fung, P. (2024) ‘What makes for good image captions?’, *Computer Vision and Pattern Recognition.* arXiv.
3. Stefanini, M., Cornia, M., Baraldi, L., Cascianelli, S., Fiameni, G. and Cucchiara, R. (2021) ‘From show to tell: A survey on deep learning-based image captioning’, *Computer Vision and Pattern Recognition*, 45(1), pp. 539–559
4. Dognin, P., Melnyk, I., Mroueh, Y., Padhi, I., Rigotti, M., Ross, J., Schiff, Y., Young, R. A. and Belgodere, B. (2022) ‘Image captioning as an assistive technology: Lessons learned from vizwiz 2020 challenge’, *Journal of Artificial Intelligence Research*, 73, pp. 437–459.
5. Makav, B. and Kilic, V. (2019) ‘A new image captioning approach for visually impaired people’, *2019 11th International Conference on Electrical and Electronics Engineering (ELECO)*
6. Ayesha, H., Iqbal, S., Tariq, M., Abrar, M., Sanaullah, M., Abbas, I., Rehman, A., Niazi, M. F. and Hussain, S. (2021) ‘Automatic Medical Image Interpretation: State of the art and Future Directions’, *Pattern Recognition*, 114, p. 107856.
7. Putra, B. T., Soni, P., Marhaenanto, B., Pujiyanto, Sisbudi Harsono, S. and Fountas, S. (2020) ‘Using information from images for plantation monitoring: A review of solutions for Smallholders’, *Information Processing in Agriculture*, 7(1), pp. 109–119.
8. Tang, J., McGoldrick, G., Al-Ghossein, M. and Chen, C.-W. (2024) ‘Captions are worth a thousand words: Enhancing product retrieval with pretrained image-to-text models’, *Computer Vision and Pattern Recognition*. arXiv.
9. Luo, R. C., Hsu, Y.-T., Wen, Y.-C. and Ye, H.-J. (2019) ‘Visual image caption generation for service robotics and industrial applications’, *2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS)*, pp. 827–832.
10. Hossain, MD. Z., Sohel, F., Shiratuddin, M. F. and Laga, H. (2019) ‘A comprehensive survey of Deep Learning for Image captioning’, *ACM Computing Surveys*, 51(6), pp. 1–36.
11. Sutskever, I., Vinyals, O. and V. Le, Q. (2014) ‘Sequence to sequence learning with neural networks’, *NIPS’14: Proceedings of the 28th International Conference on Neural Information Processing Systems*, 2, pp. 3104–3112.
12. Stefanini, M., Cornia, M., Baraldi, L., Cascianelli, S., Fiameni, G. and Cucchiara, R. (2023) ‘From Show to Tell: A survey on deep learning-based image captioning’, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1), pp. 539–559.
13. Vinyals, O., Toshev, A., Bengio, S. and Erhan, D. (2016) ‘Show and tell: Lessons learned from the 2015 MSCOCO Image captioning challenge’, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4), pp. 652–663.
14. Singh, H., Sharma, A. and Pant, M. (2024) ‘Pixels to Prose: Understanding the art of Image Captioning’, *Computer Vision and Pattern Recognition*. arXiv
15. Herdade, S., Kappeler, A., Boakye, K. and Soares, J. (2019) ‘Image Captioning: Transforming Objects into Words’, *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pp. 11137–11147.
16. Wang, C., Yang, H., Bartz, C. and Meinel, C. (2016) ‘Image captioning with deep bidirectional lstms’, *Proceedings of the 24th ACM international conference on Multimedia*, pp. 988–997.
17. Jia, X., Gavves, E., Fernando, B. and Tuytelaars, T. (2015) ‘Guiding the long-short term memory model for image caption generation’, *2015 IEEE International Conference on Computer Vision*
18. You, Q., Jin, H., Wang, Z., Fang, C. and Luo, J. (2016) ‘Image captioning with semantic attention’, *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
19. Huang, L., Wang, W., Chen, J. and Wei, X.-Y. (2019) ‘Attention on attention for image captioning’, *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*.
20. Castro, R. *et al.* (2022) ‘Deep Learning Approaches Based on Transformer Architectures for Image Captioning Tasks’, *IEEE Access*, 10, pp. 33679–33694. doi:10.1109/access.2022.3161428.
21. Liu, W., Chen, S., Guo, L., Zhu, X. and Liu, J. (2021) ‘CPTR: Full Transformer Network for Image Captioning’, *Computer Vision and Pattern Recognition*. arXiv
22. Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T. and Xie, S. (2022) ‘A ConvNet for the 2020s’, *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
23. Ramos, L., Casas, E., Romero, C., Rivas-Echeverría, F. and Morocho-Cayamcela, M. E. (2024) ‘A Study of ConvNeXt Architectures for Enhanced Image captioning’, *IEEE Access*, 12
24. Cahyono, J. A. and Jusuf, J. N. (2024) ‘Automated Image Captioning with CNNs and Transformers’, *Computer Vision and Pattern Recognition*. arXiv
25. Bengio, S. *et al.* (2015) ‘Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks’, *NIPS’15: Proceedings of the 29th International Conference on Neural Information Processing Systems*, 1, pp. 117–1179.
26. Rennie, S.J. *et al.* (2017) ‘Self-Critical Sequence Training for Image Captioning’, *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7008–7024. doi:10.1109/cvpr.2017.131.
27. Mikolov, T. *et al.* (2013) ‘Efficient Estimation of Word Representations in Vector Space’, *Conference: Proceedings of the International Conference on Learning Representations (ICLR 2013)* [Preprint].
28. Pennington, J., Socher, R. and Manning, C. (2014) ‘Glove: Global vectors for word representation’, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543. doi:10.3115/v1/d14-1162.
29. Atliha, V. and Sesok, D. (2021) ‘Pretrained word embeddings for image captioning’, *2021 IEEE Open Conference of Electrical, Electronic and Information Sciences (eStream)*
30. Shi, C. *et al.* (2024) ‘A Thorough Examination of Decoding Methods in the Era of LLMS’, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 8601–8629. doi:10.18653/v1/2024.emnlp-main.489.
31. *Common objects in context* *COCO*. Microsoft. Available at: https://cocodataset.org/#home
32. Karpathy, A. and Fei-Fei, L. (2017) ‘Deep Visual-Semantic Alignments for Generating Image Descriptions’, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4)
33. FSE (2021) *FSE/word2vec-google-news-300*, *Fast Sentence Embedding - word2vec-google-news-300 - Hugging Face*. Available at: https://huggingface.co/fse/word2vec-google-news-300 (Accessed: 18 September 2025)
34. FSE (2021) *FSE/glove-wiki-gigaword-200*, *Fast Sentence Embedding - glove-wiki-gigaword-200 - Hugging Face*. Available at: https://huggingface.co/fse/glove-wiki-gigaword-200 (Accessed: 18 September 2025)
35. Ranzato, M. *et al.* (2016) ‘Sequence Level Training with Recurrent Neural Networks’, *ICLR*.
36. Vinodababu, S. (2019). *A PyTorch Tutorial to Image Captioning*. [Online]. GitHub. Available at: <https://github.com/sgrvinod/a-PyTorch-Tutorial-to-Image-Captioning>
37. Xu, K. *et al.* (2015) ‘Show, Attend and Tell: Neural Image Caption Generation with Visual Attention’, *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37}*, pp. 2048–2057. doi:10.5555/3045118.3045336
38. Sarkar, A. (2025) *Transformer Model Tutorial in PyTorch: From Theory to Code*, *Datacamp*. Available at: https://www.datacamp.com/tutorial/building-a-transformer-with-py-torch (Accessed: 17 September 2025)
39. PyTorch. *TransformerDecoderLayer - PyTorch 2.8 documentation*. Available at: https://docs.pytorch.org/docs/stable/generated/torch.nn.TransformerDecoderLayer.html
40. PyTorch. *TransformerDecoder - PyTorch 2.8 documentation*. Available at: <https://docs.pytorch.org/docs/stable/generated/torch.nn.TransformerDecoder.html>
41. PyTorch. *Pytorch - transformer.py*, *GitHub*. Available at: https://github.com/pytorch/pytorch/blob/v2.8.0/torch/nn/modules/transformer.py#L966
42. GeeksforGeeks (2025) *Pre-trained word embedding in NLP*, *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/nlp/pre-trained-word-embedding-in-nlp/ (Accessed: 17 September 2025)
43. Manna, S. (2025) *The Practical Guide to distributed training using PYTORCH - part 4: On multiple nodes using Slurm*, *Medium*. Available at: https://medium.com/the-owl/the-practical-guide-to-distributed-training-using-pytorch-part-4-on-multiple-nodes-using-slurm-83cf306a3373
44. PyTorch. *Multi-node training using Slurm*, *Multi-Node Training using SLURM*. Available at: https://pytorch-geometric.readthedocs.io/en/2.6.0/tutorial/multi\_node\_multi\_gpu\_vanilla.html
45. Diakogiannis, F. (2024) *Distributed training on Slurm Cluster*, *PyTorch Forums*. Available at: https://discuss.pytorch.org/t/distributed-training-on-slurm-cluster/150417/13
46. PyTorch. *DistributedDataParallel - PyTorch 2.8 documentation*. Available at: https://docs.pytorch.org/docs/stable/generated/torch.nn.parallel.DistributedDataParallel.html
47. PyTorch (2017) *Distributed Communication Package*, *Distributed communication package - torch.distributed - PyTorch 2.8 documentation*. Available at: https://docs.pytorch.org/docs/2.8/distributed.html
48. namespace-Pt (2021) *A Comprehensive Tutorial to Pytorch DistributedDataParallel*, *Medium*. Available at: https://medium.com/codex/a-comprehensive-tutorial-to-pytorch-distributeddataparallel-1f4b42bb1b51

# A. Appendix: Exploratory Data Analysis and Metric Formulae

## A.1. MS COCO Captions and Words



**Figure 12.** Distribution of caption lengths in the MS COCO dataset



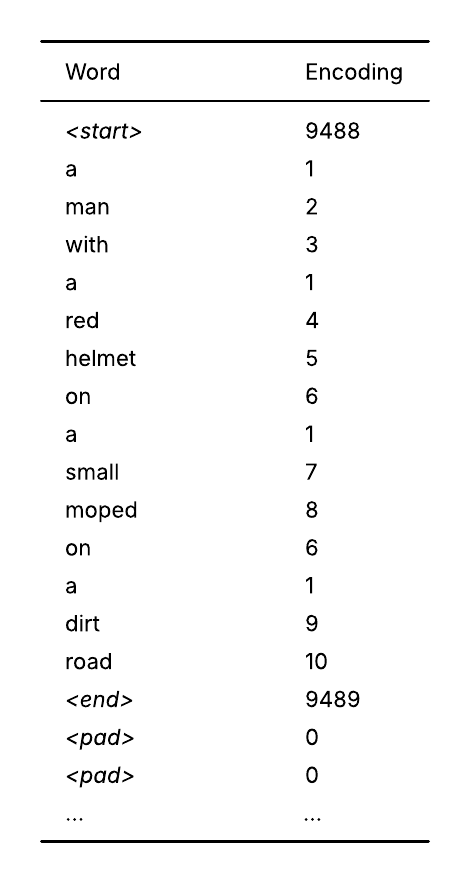
**Figure 13.** Frequencies of the top 20 most occurring words in the MS COCO dataset

## A.2. Wordmap used to encode captions



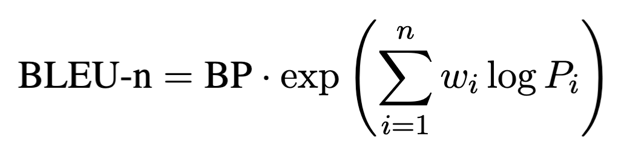
**Figure 14.** Snippet of wordmap.json file used to encode the captions

## A.3. Caption encoding example using wordmap



**Figure 15.** Example of caption encoded using wordmap

## A.4. Formulae for BLEU score and attention regularization loss

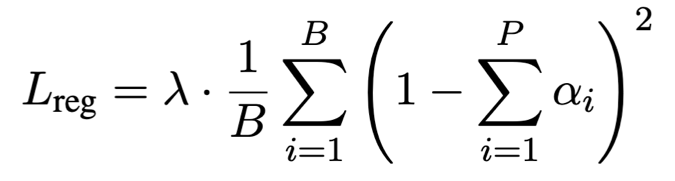


**Figure 16.** Formula for evaluating BLEU score (Ramos et al., 2024)

BP = brevity penalty

*wi* = weights for each n-gram level

*Pi* = precision at each n-gram level



**Figure 17.** Formula for evaluating attention regularization term

l = alphaC which is the hyperparameter controlling the strength of the penalty

*B* = batch size

*P*= number of image regions

*ai* = attention weights

# B. Appendix: Additional Experiments and Results

## B.1. Generated captions of unseen sample images (Teacher Forcing vs Non-Teacher Forcing)

|  |  |  |
| --- | --- | --- |
| **Image** | **Decoder + Strategy** | **Generated Caption** |
|  | (LSTM + Att) + No TF | A small and and blue plane in in the |
| Transformer + No TF | A small airplane plane plane in a a |
| (LSTM + Att) + TF | A small airplane is flying through the air |
| Transformer + TF | A small blue and white airplane flying in the sky |
|  | (LSTM + Att) + No TF | A motorcycle parked on a a a a |
| Transformer + No TF | A motorcycle bike parked a a a a |
| (LSTM + Att) + TF | A motorcycle is parked on a dirt road |
| Transformer + TF | A motorcycle parked on a dirt road next to a fence |
|  | (LSTM + Att) + No TF | A dog sitting a a bench a a a bench |
| Transformer + No TF | A white dog a a a a a a |
| (LSTM + Att) + TF | A couple of dogs sitting on a bench |
| Transformer + TF | A white dog is sitting on a bench |
|  | (LSTM + Att) + No TF | A traffic light with a a a a |
| Transformer + No TF | A traffic light with a lights a a |
| (LSTM + Att) + TF | A traffic light that has a red light on it |
| Transformer + TF | A traffic light with a sky background |
|  | (LSTM + Att) + No TF | A table table with a table and a a |
| Transformer + No TF | A dining table with a a a a |
| (LSTM + Att) + TF | A dining table with a vase of flowers on it |
| Transformer + TF | A dining room table with a vase of flowers on it |
|  | (LSTM + Att) + No TF | A group of people playing a a a |
| Transformer + No TF | A group of people playing a a a |
| (LSTM + Att) + TF | A group of people walking down a street |
| Transformer + TF | A group of people walking down a sidewalk next to a building |
|  | (LSTM + Att) + No TF | A group of sheep standing in a a |
| Transformer + No TF | A herd of sheep standing in a a |
| (LSTM + Att) + TF | A herd of sheep standing on top of a lush green field |
| Transformer + TF | A herd of sheep standing on top of a lush green field |
|  | (LSTM + Att) + No TF | A boat of a a on a a |
| Transformer + No TF | A group of a boat boat a boa |
| (LSTM + Att) + TF | A boat that is sitting in the water |
| Transformer + TF | A boat with a flag hanging from it |
|  | (LSTM + Att) + No TF | Two giraffes standing a a in in a |
| Transformer + No TF | Two giraffes of giraffes standing in a trees |
| (LSTM + Att) + TF | A couple of giraffe standing next to each other |
| Transformer + TF | Two giraffes standing next to each other in a field |
|  | (LSTM + Att) + No TF | A fire hydrant hydrant a a a a |
| Transformer + No TF | A fire hydrant hydrant water hydrant a a |
| (LSTM + Att) + TF | A fire hydrant in the middle of a street |
| Transformer + TF | A fire hydrant is spraying water onto a sidewalk |

**Table 13.** Captions generated by LSTM and Transformer decoders with and without teacher forcing

## B.2. Generated captions of unseen sample images (Fine-tuning ConvNeXt)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Image** | **Layers Fine-tuned** | **Learning Rate** | **Patience** | **Generated Caption** |
|  | None | - | 20 | A small blue and white airplane flying in the sky |
| 5 – 7 | 1×10-4 | 20 | A small blue and white airplane is on the grass |
| 5 – 7 | 1×10-5 | 40 | A small blue and white airplane flying in a field |
| 5 – 7 | 1×10-6 | 40 | A blue and white airplane is flying in the air |
| 3 – 7 | 1×10-4 | 20 | A small blue and white airplane on a grassy field |
| 1 – 7 | 1×10-6 | 40 | A small blue and white plane flying in a field |
|  | None | - | 20 | A motorcycle parked on a dirt road next to a fence |
| 5 – 7 | 1×10-4 | 20 | A red and black motorcycle parked in a dirt field |
| 5 – 7 | 1×10-5 | 40 | A motorcycle is parked in a fenced in area |
| 5 – 7 | 1×10-6 | 40 | A motorcycle parked on a dirt road near a fence |
| 3 – 7 | 1×10-4 | 20 | A red and blue motorcycle parked in a dirt field |
| 1 – 7 | 1×10-6 | 40 | A motorcycle parked in a fenced in area |
|  | None | - | 20 | A white dog is sitting on a bench |
| 5 – 7 | 1×10-4 | 20 | A man and woman sitting on a bench with a dog |
| 5 – 7 | 1×10-5 | 40 | A man sitting on a bench with a dog |
| 5 – 7 | 1×10-6 | 40 | A white dog standing next to a park bench |
| 3 – 7 | 1×10-4 | 20 | A couple of people that are sitting on a bench |
| 1 – 7 | 1×10-6 | 40 | A white dog sitting next to a park bench |
|  | None | - | 20 | A traffic light with a sky background |
| 5 – 7 | 1×10-4 | 20 | A traffic light with a sky background |
| 5 – 7 | 1×10-5 | 40 | A couple of traffic lights on a pole |
| 5 – 7 | 1×10-6 | 40 | A traffic light with a sky background |
| 3 – 7 | 1×10-4 | 20 | A traffic light with a sky background |
| 1 – 7 | 1×10-6 | 40 | A couple of traffic lights hanging from a pole |
|  | None | - | 20 | A dining room table with a vase of flowers on it |
| 5 – 7 | 1×10-4 | 20 | A table with a vase of flowers on it |
| 5 – 7 | 1×10-5 | 40 | A table with a vase of flowers on it |
| 5 – 7 | 1×10-6 | 40 | A table with a vase of flowers on it |
| 3 – 7 | 1×10-4 | 20 | A table with a vase of yellow flowers on it |
| 1 – 7 | 1×10-6 | 40 | A wooden table topped with a vase of flowers |
|  | None | - | 20 | A group of people walking down a sidewalk next to a building |
| 5 – 7 | 1×10-4 | 20 | A group of people standing around a cage |
| 5 – 7 | 1×10-5 | 40 | A group of people walking down a street |
| 5 – 7 | 1×10-6 | 40 | A group of people walking down a sidewalk |
| 3 – 7 | 1×10-4 | 20 | A group of people standing around a fence |
| 1 – 7 | 1×10-6 | 40 | A group of people walking down a sidewalk |
|  | None | - | 20 | A herd of sheep standing on top of a lush green field |
| 5 – 7 | 1×10-4 | 20 | A herd of sheep standing on top of a lush green field |
| 5 – 7 | 1×10-5 | 40 | A herd of sheep standing on top of a lush green field |
| 5 – 7 | 1×10-6 | 40 | A herd of sheep standing on top of a lush green field |
| 3 – 7 | 1×10-4 | 20 | A herd of sheep standing on top of a lush green field |
| 1 – 7 | 1×10-6 | 40 | A herd of sheep standing on top of a lush green field |
|  | None | - | 20 | A boat with a flag hanging from it |
| 5 – 7 | 1×10-4 | 20 | A boat filled with lots of food next to a river |
| 5 – 7 | 1×10-5 | 40 | A boat with a flag hanging on the side of it |
| 5 – 7 | 1×10-6 | 40 | A boat filled with lots of items floating on top of a river |
| 3 – 7 | 1×10-4 | 20 | A boat with a flag and a flag on it |
| 1 – 7 | 1×10-6 | 40 | A boat with a flag hanging from the side of it |
|  | None | - | 20 | Two giraffes standing next to each other in a field |
| 5 – 7 | 1×10-4 | 20 | A giraffe standing next to another giraffe in a field |
| 5 – 7 | 1×10-5 | 40 | A giraffe standing next to another giraffe in a field |
| 5 – 7 | 1×10-6 | 40 | A group of giraffes grazing in a field |
| 3 – 7 | 1×10-4 | 20 | A group of giraffes grazing in a field |
| 1 – 7 | 1×10-6 | 40 | A group of giraffes grazing in a field |
|  | None | - | 20 | A fire hydrant is spraying water onto a sidewalk |
| 5 – 7 | 1×10-4 | 20 | A fire hydrant with water spraying out of it |
| 5 – 7 | 1×10-5 | 40 | A fire hydrant with water flowing out of it |
| 5 – 7 | 1×10-6 | 40 | A fire hydrant with water spraying out of it |
| 3 – 7 | 1×10-4 | 20 | A yellow fire hydrant sitting next to a building |
| 1 – 7 | 1×10-6 | 40 | A fire hydrant spraying water onto a sidewalk |

**Table 14.** Captions generated by fine-tuning different layers of the ConvNeXt

## B.3. Generated captions of unseen sample images (Pretrained Word Embeddings)

|  |  |  |
| --- | --- | --- |
| **Image** | **Word Embeddings** | **Generated Caption** |
|  | Random | A blue and white airplane is flying in the air |
| Word2Vec | A small blue and white airplane on a grassy field |
| GloVe | A small blue and white airplane on a grassy field |
|  | Random | A motorcycle parked on a dirt road near a fence |
| Word2Vec | A motorcycle that is parked in the dirt |
| GloVe | A motorcycle that is parked in the dirt |
|  | Random | A white dog sitting next to a park bench |
| Word2Vec | A man sitting on a bench next to a white dog |
| GloVe | A man sitting on a bench next to a white dog |
|  | Random | A traffic light with a sky background |
| Word2Vec | A traffic light with a sky background |
| GloVe | A traffic light with a sky background |
|  | Random | A table with a vase of flowers on it |
| Word2Vec | A table with a vase of flowers on it |
| GloVe | A table with a vase of flowers on it |
|  | Random | A group of people walking down a sidewalk |
| Word2Vec | A group of people walking down a sidewalk |
| GloVe | A group of people walking down a sidewalk |
|  | Random | A herd of sheep standing on top of a lush green field |
| Word2Vec | A herd of sheep standing on top of a lush green field |
| GloVe | A herd of sheep standing on top of a lush green field |
|  | Random | A boat filled with lots of items floating on top of a river |
| Word2Vec | A boat filled with supplies floating on top of a river |
| GloVe | A boat filled with flags floating on top of a river |
|  | Random | A group of giraffes grazing in a field |
| Word2Vec | A herd of giraffe standing next to each other |
| GloVe | A group of giraffes grazing in a field |
|  | Random | A fire hydrant with water spraying out of it |
| Word2Vec | A fire hydrant with water spraying from it |
| GloVe | A fire hydrant spraying water onto a street |

**Table 15.** Captions generated by pretrained word embeddings

# C. Appendix: Project Proposal Document

Enhancing Image Captioning using ConvNeXt with LSTM and Transformer Decoders

Sajeel Nadeem Alam - 240045404 - MSc Data Science

**Supervisor:** Professor Kevin Ryan

Introduction

Humans have the ability to describe their environment with immense precision. They are able to explain a visual scenario even after glancing at it for a moment. This is due to the sense of sight, which from the eyes and through optic nerves sends visual information to the brain that has the ability to convert it to natural language descriptions. Making computers replicate this task has been a challenge in recent years for researchers in artificial intelligence (Bai & An, 2018). This task is formally known as ‘Image Captioning’ and can be described as converting visual content into descriptive natural language texts (Chen et al., 2024). The motivation for improving computers’ ability in image captioning is generated by the potential of using captioning systems in real-world applications (Dognin et al., 2022). The diverse use-cases of image captioning include assisting visually impaired people by providing descriptions of their physical and digital environments (Gurari et al., 2020), supporting the diagnosis of medical scans by generating their descriptions (Ayesha et al., 2021), improving product retrieval in e-commerce through image-generated text descriptions (Tang et al., 2024) and assisting industrial robots to make informed decisions by allowing them to understand visual data (Luo et al., 2019).

The problem statement of image captioning can be defined as the task of generating a descriptive natural language caption for an image in the form of a detailed, comprehensive sentence describing the objects in the image and their interactions with each other. The system takes the image as an input, uses a visual understanding model and a language model to generate meaningful captions which are outputted (Stefanini et al., 2021). Hence this task employs both computer vision and natural language processing (NLP), each presenting its own set of challenges. In terms of computer vision, image captioning is a complex task because it consists of not only identifying and describing multiple objects within an image but also accurately conveying the relationship between these objects in the context of the image. It also has to take into account both local features containing information about prominent objects in the image as well as global features representing the background context (Yang & Hu, 2019). In terms of NLP, the system has to use these extracted features to construct a description that is not only grammatically correct but also accurate in terms of describing the relationship between objects and the overall scenario in the image. Thus, the model has to be contextually aware while generating the next word which is a challenge in itself (Yang & Hu, 2019). Image captioning is usually treated as supervised learning and the subjective nature of natural language presents another challenge since an image can have multiple accurate captions. Hence in datasets designed for image captioning it is common to have multiple captions (labels) for each image which is why the evaluation metrics differ from the ones used in general tasks. A prominent implementation that has achieved promising results in existing literature despite these challenges is the encoder-decoder architecture in which an image model e.g., a CNN is used with a language model e.g., an RNN or LSTM to extract features from an image and use them to construct captions, and this study aims to improve both elements in this architecture.

The question this study aims to answer is “*How can fine-tuning a ConvNeXt encoder, combined with an LSTM or Transformer decoder enhance image captioning performance, and what is the impact of teacher forcing, decoding strategies and word embeddings on the quality of generated captions?”* In order to effectively answer this question, the objectives of this study involve implementing the ConvNeXt model which is a specialized CNN pre-trained on ImageNet, and the decoder whether it is an LSTM or transformer. This will be followed by the objective of connecting both models in a unified manner such that data can pass through them and the error can be back-propagated throughout the architecture. The next objective would be to train and select the best decoder on the basis of their baseline performance for the task of image captioning using the MS COCO dataset (Microsoft, 2014). Fine-tuning the ConvNeXt and analyzing its performance on unfreezing different depths of layers is another key objective. In order to answer the second part of the question, other objectives involve training the model with and without teacher forcing, implementing decoding strategies such as greedy search and beam search in the decoder while selecting the next word during inference, and applying different word embeddings models such as Word2Vec and GloVe to encode the ground truth caption before passing it to the decoder. The model’s performance in terms of BLEU and CIDEr scores will be measured in each configuration and compared.

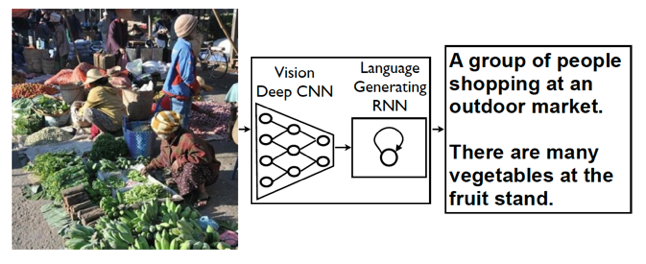
The key deliverables of this study include a working architecture of the image captioning system, a detailed project report with a section on the model’s performance and an open-source implementation of the architecture. The aim is that this work will serve as a benchmark for future research. The beneficiaries of this work will be visually impaired individuals who will benefit from an improved image captioning system allowing them to have a better quality of life (Makav & Kilic, 2019). Moreover, fine-tuning this model on a medical image dataset can support individuals in the healthcare industry to diagnose medical scans (Ayesha et al., 2021). Farmers can also benefit from image captioning systems that are fine-tuned for monitoring plant conditions from close-up images (Putra et al., 2020). E-commerce businesses will benefit by leveraging image captioning for product retrieval through image-generated text descriptions allowing for better search functionalities (Tang et al., 2024). This study will only focus on generating captions for static images and will be limited to the MS COCO dataset that contains well-annotated images. Hence the model will be trained on general images and will not be fine-tuned for a specific domain. The scope of this work does not include the development of real-time image captioning systems for generating captions for videos as this would require more temporally complex models.

Critical Context

**Early Image Captioning Techniques**

The first image captioning systems employed template-based methods, where fixed sentence templates with blank slots were filled with identified objects and their relationships to generate captions (Hossain et al., 2019). These systems aimed to produce grammatically correct captions for specific domains but were rigid and produced weak, contextually limited captions that lacked generalization across diverse images (Hossain et al., 2019). Another approach, retrieval-based methods, selected the caption for a query image from a pool of captions of similar images. While this allowed the captioning of large amounts of image data, it was limited to reusing existing descriptions which did not cater to specific details or the context of each individual image (Hossain et al., 2019).

**Encoder-Decoder Architecture**

With advancements in image and language models over the years, deep-learning based image captioning approaches became popular. In these approaches, models that have the ability to learn relevant features from images and use them to generate specific, context-aware captions were used which allowed the research to improve from fixed captions to novel caption generation (Hossain et al., 2019). The most prominent architecture using these models is the encoder-decoder architecture inspired by sequence-to-sequence learning for language translation (Sutskever et al., 2014). In encoder-decoder architectures, an image model (encoder) extracts relevant features from the image which are passed onto a language model (decoder) that generates a natural language description for the image using provided visual features (Stefanini et al., 2023). One prominent work which used this architecture was a team from Google and their implementation consisted of fine-tuning GoogleLeNet as an encoder on the MS COCO dataset while training an LSTM as a decoder using stochastic gradient descent (Vinyals et al., 2017). This approach allowed them to surpass the existing state-of-art results by achieving a 59 BLEU score on the Pascal dataset and win the 2015 MS COCO Image Captioning Challenge. Since then, in most existing literature, the encoder-decoder architecture has been implemented with a CNN encoder and an RNN/LSTM decoder.

Although it shows promising results, this approach has its limitations since the image features are provided to the RNN only at the beginning leading to issues like vanishing gradients which are mitigated by the LSTM however, it faces the issue of weakened influence of the image’s semantic as the caption progresses resulting in less context aware captions (Singh et al., 2024). Moreover, the CNN focuses on the image as a whole instead of the individual objects and their relationships.

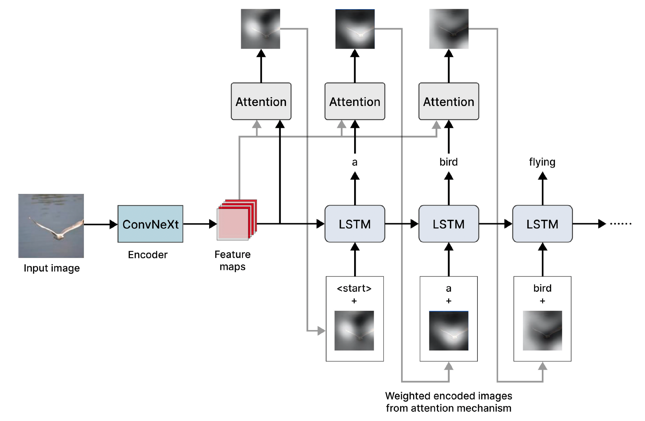
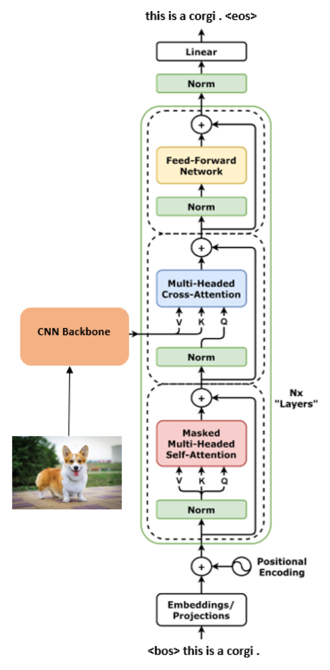
**Improvements in the Encoder-Decoder Architecture**

In order to overcome the issues mentioned earlier, researchers have tried to modify individual components of the encoder-decoder architecture. In one paper, researchers used Faster R-CNN with ResNet-101 to identify objects in the image, encode their geometric information and spatial relationships, and generate feature vectors for each region in the image which helped overcome the issue of only focusing on the image as a whole (Herdade et al., 2019). In another study, authors used a deep bidirectional LSTM as a decoder which had the ability to process the sentence and capture context in both directions resulting in more contextually aware captions (Wang et al., 2016). Although there was an improvement in performance, contextual information from the image was not provided to the decoder during caption generation which was a limitation. Addressing this issue, a study used a 16-layer-Oxford-Net to extract image features and cross-modal retrieval to find relevant texts in the image. The image features were fed to a guided LSTM and relevant texts were incorporated in each LSTM gate to provide semantic information (Jia et al., 2015). This gave contextual information about the image to the decoder during caption generation allowing for more contextually aware captions however, this information was static and did not adapt during the caption generation process. To address this issue, another study used GoogleNet to extract image features that were fed into an LSTM, which had an integrated semantic attention module. This module, using attribute detectors, identified semantic concepts in the image and dynamically updated attention based on the previously generated word and its semantic context, guiding the LSTM at every step to relevant image regions during caption generation resulting in contextually rich captions (You et al., 2016). However, this approach provided semantic information about local features only, focusing on specific regions of the image at each step. Combined with sequential processing in the LSTM, where the model cannot consider the entire image and generated caption so far (i.e., all previously generated words) simultaneously, this resulted in a limitation, as it was unable to capture the global context and relationships across the entire image which are equally important as local, fine-grained features.

**Self-Attention and Transformers**

In recent years, with the introduction of self-attention and transformers, researchers have explored their potential use in overcoming the limitation of global context faced in CNN-LSTM architectures as mentioned earlier. In one study, researchers incorporated a self-attention mechanism in both the CNN encoder and LSTM decoder to dynamically assign weights to each part of the image and generated caption, providing a global and local context while generating the next word. They further enhanced this with an attention-on-attention module that filtered out irrelevant image regions, allowing the model to focus on more relevant parts and capture global dependencies during caption generation (Huang et al., 2019). In another study, authors replaced the CNN-LSTM architecture with a full transformer architecture for encoding and decoding. By processing the image as a sequence of patches and using self-attention throughout the architecture, the model was able to capture both global and local relationships between different regions of the image and the generated caption so far. The inherent parallel processing ability of the transformer considered all regions of the image and the generated caption simultaneously, providing global context during the caption generation process (Liu et al., 2021). Using a full transformer network instead of a CNN-LSTM architecture with an attention module overcomes the limitation of relying on a CNN for feature extraction which processes image information sequentially thus limiting the global context across the image and caption. On the other hand, transformers are able to capture both global and local contexts but might not always handle fine-grained local features as well as CNNs. The key takeaway is to find an optimal balance between both architectures.

**ConvNeXt Encoder**

In order to achieve this balance, in 2022 the research team at Meta developed an improved CNN called ConvNeXt which builds on the foundational model of ResNet and incorporates upgrades from vision transformers. Upgrades such as using larger convolution kernels enabled the model to gather broader context across the image while a hierarchical feature learning approach inspired from Swin Transformers, processed the image at different resolutions. This allowed the model to capture fine-grained local features at lower resolutions combined with broader context at higher resolutions and integrate them in the feature vectors (Liu et al., 2022). This balance between local and global context serves as motivation to fine-tune ConvNeXt for image captioning as was done by a team of researchers in 2024. In their paper titled ‘A Study of ConvNeXt Architectures for Enhanced Image Captioning’ a ConvNeXt encoder was used with an LSTM decoder integrated with an attention module to provide more context in the caption generation process. At each stage in the LSTM, feature vectors extracted from the image were multiplied with attention weights to create a context vector which was concatenated with the input word embedding. The attention weights were calculated using the previous hidden state and determined which part of the image to focus on as shown in Figure 2 (Ramos et al., 2024). The combination of context aware image features from the ConvNeXt and the attention modules integrated in the LSTM provided the model with a balance of local and global contexts during the caption generation process thus outperforming vision transformers (Ramos et al., 2024).

The ability of ConvNeXt to extract locally and globally context aware image features and its use case in the paper serve as a source of motivation to use it as an encoder in this study. Exploring further improvements by fine-tuning various layers of the ConvNeXt is a key objective of this study since the paper only fine-tuned the final layer. Considering its strengths mentioned earlier, for the decoder this study will explore using an LSTM with an integrated attention module similar to the paper (Ramos et al., 2024). However, in a recent study (Cahyono & Jusuf, 2024) authors used a hybrid architecture that incorporates a CNN encoder with a transformer decoder as shown in Figure 3. Considering the transformer’s ability for parallel processing, which allows it to simultaneously attend to the entire image and the part of the caption generated so far, the model can effectively utilize local and global contexts at each step to generate more context-aware captions. Based on this, this study will also explore using a transformer decoder and select the better performing decoder for further analysis.

**Training, Decoding Strategies and Word Embeddings**

Inspired by the ConvNeXt+LSTM paper (Ramos et al., 2024) where the authors train the model with and without teacher forcing, this study will do the same to find the balance between generalization and how fast the model converges. In another paper (Vinyals et al., 2017), researchers from Google compare greedy search and beam search in the decoder to select the next word during inference, thus analyzing their effects on diversity in generated captions and also checking the model for overfitting (Vinyals et al., 2017). This study aims to do the same. Moreover, researchers have investigated the effect of Word2Vec, GloVe and non-pretrained word embedding models to encode the caption before passing it to the decoder during training to analyze if prior semantic knowledge from a pre-trained corpora improves model performance [28]. This study has an objective to do the same with the LSTM decoder since transformer decoders learn their own embeddings during training.

Approaches

**Data**

This study will use the 2014 MS COCO dataset which was originally curated by Microsoft for the COCO 2015 Image Captioning challenge. Since then, it has been widely used by researchers for the image captioning task, making it a standard benchmark and allowing for meaningful comparisons of this study’s results with existing works. In order to keep things consistent with the paper (Ramos et al., 2024), this study will use the Karpathy split (Karpathy & Fei-Fei, 2017) according to which the dataset contains 123,287 images from which 5000 images are used for validation, 5000 images are used for testing and the rest for training, providing a suitable number of diverse and general domain images for training robust models. Each image is well-annotated with at least five captions, stored alongside other metadata in JSON format. Multiple captions provide good variety for evaluation. The dataset will be loaded using the codebase provided by Karpathy (Karpathy & Fei-Fei, 2017) in their public GitHub repository, supported by the COCO API and its official documentation on the MS COCO website. Since the study will use a pre-trained ConvNeXt encoder, images will be resized to 224x224 pixels and normalized using the mean and standard deviations of ImageNet (Ramos et al., 2024). Captions will be tokenized, starting and ending tags will be concatenated to them, and they will be padded to ensure uniform lengths for the LSTM decoder. These padded captions will be encoded using word embeddings before being passed into the LSTM decoder. For the transformer decoder, raw sentences will be passed directly, allowing the model to handle the tokenization and embedding internally.

**Model Architecture**

As mentioned earlier, the architecture in this study will use a ConvNeXt encoder with an LSTM decoder integrated with an attention module, or a transformer decoder. In both cases, pre-processed images will be fed to the ConvNeXt which will create image features before passing them to the decoder. In order to connect the LSTM to the ConvNeXt, similar to the implementation in (Ramos et al., 2024), at each stage in the LSTM, feature vectors extracted from the image will be multiplied with attention weights to create a context vector which will be concatenated with the input word embedding. The attention weights will be calculated using the previous hidden state. The LSTM will generate one word at a time and will output the entire caption at the end. In the case of the transformer decoder, following the approach in (Cahyono & Jusuf, 2024), image features from the ConvNeXt will be the key and value inputs to the transformer’s attention mechanism, whereas causal masking will be applied so that the decoder attends only to previously generated tokens ensuring global context till the point the caption has been generated without information leakage from future words. The codebases for both implementations have been identified from their respective research papers. The baseline architectures will be setup referring to them and later will be fine-tuned for this study’s objectives. The architectures will be implemented using PyTorch.

**Training**

The models will be trained using backpropagation, with cross entropy loss calculating the loss between the predicted and true caption. This loss will be backpropagated through the architecture and stochastic gradient descent or Adam will be used with an appropriate learning rate to minimize it. Since ConvNeXt is pre-trained on ImageNet, its final classification layer will be removed to obtain the image features directly. Both baseline LSTM and baseline transformer decoders will be trained and used separately for image captioning with the ConvNeXt, and the decoder which gives better results will be selected for further analysis. While fine-tuning the ConvNeXt, according to a paper (Vinyals et al., 2017), it is recommended to first let the parameters of the decoder stabilize and then unfreeze the ConvNeXt layers to prevent the initial gradients from the decoder corrupt the pre-trained weights of the ConvNeXt. This study will aim to replicate this approach and fine-tune different depths of layers to identify the best configuration. Moreover, as a key objective of this study, using the best configuration of the ConvNeXt, training will be done both with and without teacher forcing. Without teacher forcing, the decoder will rely on its own predicted tokens during training which may improve generalizability whereas with teacher forcing, ground truth tokens from the caption will be provided to the decoder at each time step which may speed up convergence. In order to train the models using effective computational resources, City’s High Performance Computing facilities will be explored.

**Greedy Search vs Beam Search**

At the time of inference, the decoder generates captions by selecting the next word based on their predicted probabilities at each step. After selecting the best training strategy, this study will explore two decoding strategies. The first one is greedy search in which the decoder selects the next word with the highest probability from the vocabulary given the generated sequence, which produces captions quickly but may not always generate the best overall caption since it does not explore multiple paths. The second one is beam search in which the decoder explores multiple candidate sequences simultaneously. At each step, it expands all current sequences by considering all possible next words, calculates the combined probability for each expanded sequence, and keeps track of the top k (beam size) sequences to continue to the next step. At the end, the sequence with the highest overall probability is selected as the caption. The log softmax of predicted probabilities will be used to evaluate candidate sequences. Beam search explores multiple paths and may identify better suited sequences that might be overlooked by greedy search. Previous studies (Vinyals et al., 2017) (Cahyono & Jusuf, 2024) show that a beam size of 3 performs well which will be used as one of the values explored in this study. Overfitted models may perform poorly with large beam sizes since they over-rely on learned trained captions (Vinyals et al., 2017). Hence higher beam sizes will also be explored to check for model overfitting.

**Word Embeddings**

Before captions are passed into the decoder, they have to be encoded for the model to process them. This study will use word embeddings to encode the captions and will explore three types of word embeddings. The first one will be randomly initialized word embeddings which will be learned during training and will serve as a baseline. The other two will be pre-trained Word2Vec and GloVe embeddings which provide semantic information from large text corpora. The model’s performance will be analyzed and compared while using each embedding to assess if semantic knowledge improves the quality of generated captions.

**Evaluation Metrics**

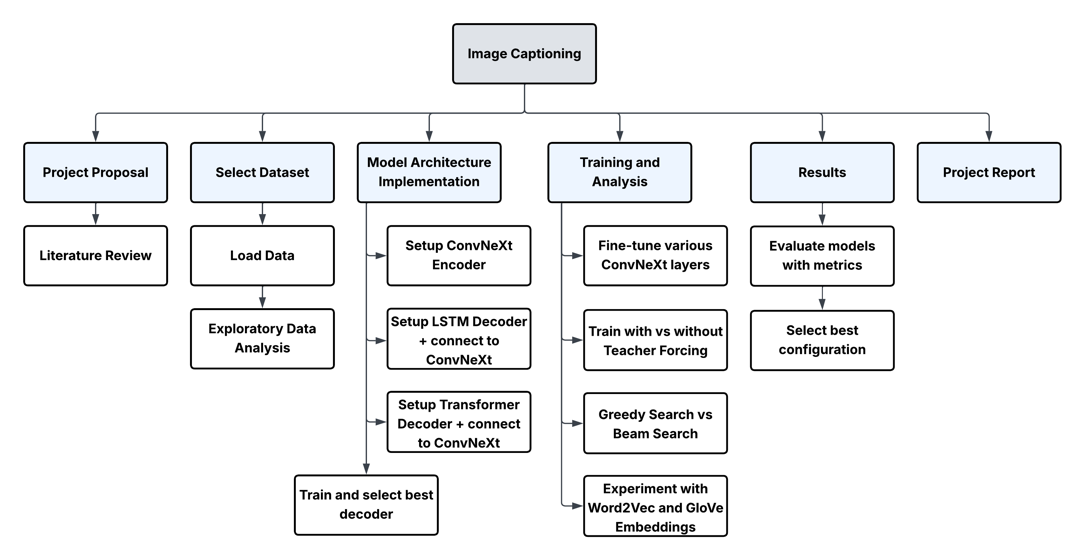
This study will use BLEU and CIDEr scores to evaluate the quality of generated captions by comparing them to the ground truth captions for each image. BLEU score measures how many n-grams i.e., contiguous sequence of words in the generated caption also appear in the ground truth captions. It calculates a precision score based on the overlapping n-grams from unigram to 4-gram resulting in BLEU-1, BLUE-2, BLUE-3 and BLUE-4 scores respectively. A brevity penalty is applied to captions that are too short. Higher BLEU scores indicate a greater overlap between the generated and ground truth captions which means that the generated caption is grammatically correct and has captured most of the semantic correctly. Since the MS COCO dataset has 5 true captions per image, this study will calculate the n-gram overlaps between the generated caption and each of the true captions, and then aggregate them using the average to get a single BLEU score for the generated caption. The BLEU scores of all the generated captions will be averaged to get a single score for the model, and the score ranges from 0 to 1 (0 to 100 when represented as a percentage). On the other hand, CIDEr has been specifically designed to evaluate image captioning models. It compares TF-IDF weighted n-grams between the generated caption and the ground truth captions to measure their similarity. The weighting emphasizes rare but important words over common ones making it more sensitive to semantic context. A higher CIDEr score indicates that the generated caption is a better match to the ground truth captions. In this study, each image has multiple ground truth captions so all the true captions will be converted to TF-IDF weighted n-grams to form a collective representation. This will be compared against the n-grams of the generated caption to calculate the CIDEr score which represents how well the generated caption matches the common semantic shared across multiple ground truth captions. The CIDEr scores of all the generated captions will be averaged to get a single score for the model.

**Ethical Considerations**

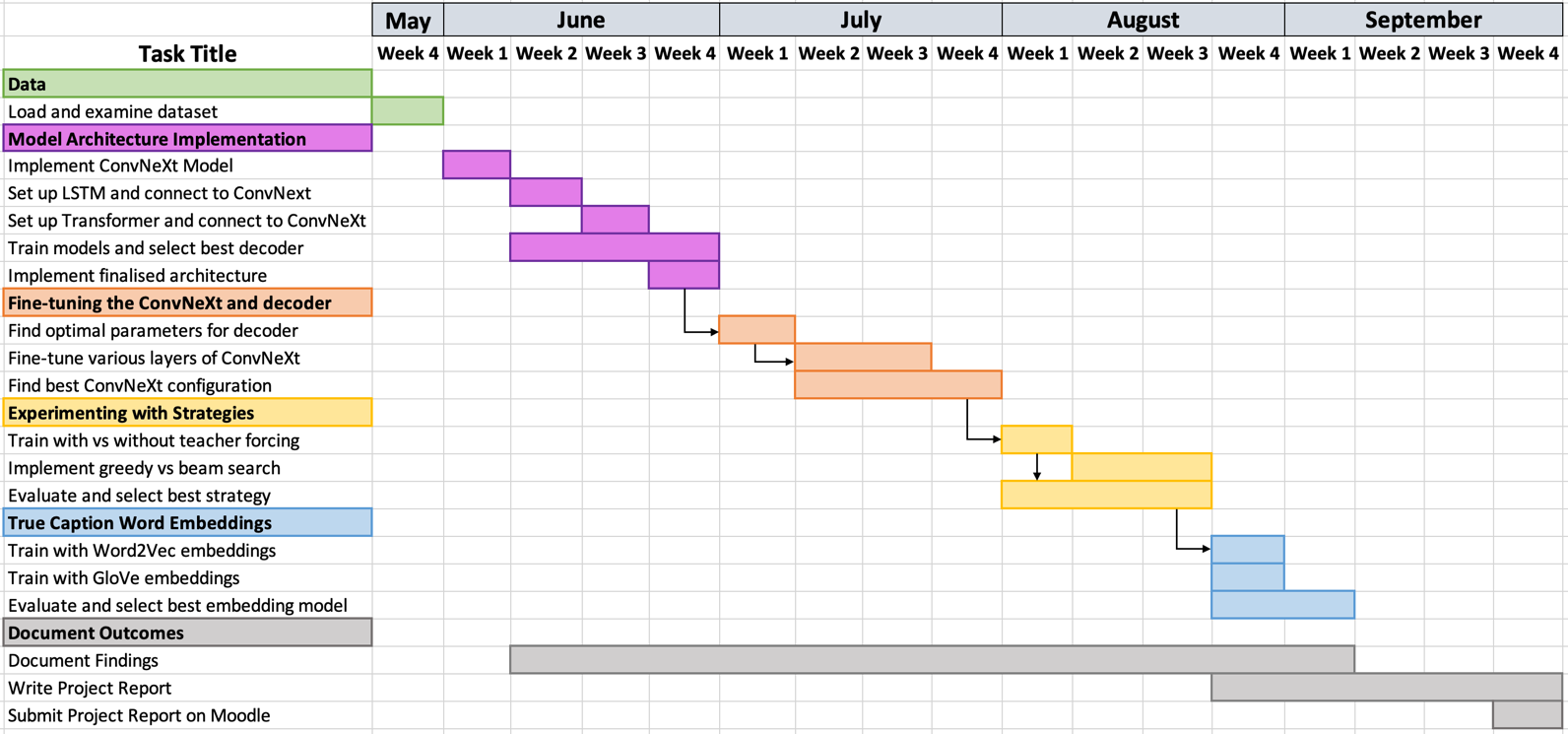
This study will use the MS COCO dataset, which is a publicly accessible and published dataset available for research purposes. No participants or personal data will be used in this research. As the research does not involve participants or sensitive/inappropriate data, after filling Part A: Ethics Checklist, it is not required to fill Part B: Ethics Proportionate Review Form for this study. The system being developed in this study will be used solely for research purposes and will not be deployed for real-world applications, hence any incorrect output generated will not have real-world consequences. Moreover, the models which will be developed will not interact with users or impact their well-being, as they will only be used for the purpose of research in image captioning.

Work Plan

The project’s work plan is illustrated below. In Figure 4, the project is broken down into key modules and each module is displayed with its important sub-tasks. This is followed by Figure 5, which shows the project’s timeline in the form of a Gantt chart. The timeline for each key module and its individual sub-tasks is presented in detail along with any dependencies.



**Figure 4.** Project Work Plan



**Figure 5.** Project Timeline

Risks

|  |  |  |  |
| --- | --- | --- | --- |
| **Risk Description** | **Likelihood** | **Impact** | **Mitigation and Recovery Plan** |
| Unable to connect transformer decoder to ConvNeXt encoder to pass data and backpropagate loss | Medium | High | Start early in June and adapt codebase from identified research paper (Cahyono & Jusuf, 2024). If issue persists, use LSTM decoder instead |
| Insufficient computational resources resulting in model taking too long to train | Medium | High | Utilize City’s HPC facilities; if resources are insufficient consider using GPU from Google Collab Pro or use a smaller dataset e.g., Flickr 8k |
| Programming difficulties in fine-tuning various ConvNeXt layers | Low | Medium | Keep 1 week buffer time to study how to unfreeze different layers for fine-tuning |
| Model fails to converge and has training issues | Medium | High | Use pre-trained ConvNeXt weights and freeze them while training decoder. If issue persists, adjust parameters to the ones used in identified papers or use a simpler CNN+LSTM architecture |
| Programming difficulties in implementing beam search | Low | Medium | Keep 1 week buffer time to resolve bugs. If issue persists, fall back on greedy search for further analysis |
| Ethical risks from incorrect generated captions causing misunderstandings for the users | Low | Medium | Clearly specify that the system is for research only and not for real-world applications. Refine the model before any deployment in the future |
| Loss of data and codebase corruption | Low | High | Save a copy of the data on Google Drive. Use version control for the code by making a repository on GitHub and maintaining it using GitHub Desktop |

References

1. Bai, S., & An, S. (2018). ‘A survey on automatic image caption generation’. *Neurocomputing*, *311*, 291–304.
2. Chen, D., Cahyawijaya, S., Ishii, E., Chan, H. S., Bang, Y. and Fung, P. (2024) ‘What makes for good image captions?’, *Computer Vision and Pattern Recognition.* arXiv.
3. Dognin, P., Melnyk, I., Mroueh, Y., Padhi, I., Rigotti, M., Ross, J., Schiff, Y., Young, R. A. and Belgodere, B. (2022) ‘Image captioning as an assistive technology: Lessons learned from vizwiz 2020 challenge’, *Journal of Artificial Intelligence Research*, 73, pp. 437–459.
4. Gurari, D., Zhao, Y., Zhang, M. and Bhattacharya, N. (2020) ‘Captioning images taken by people who are blind’, *Computer Vision – ECCV 2020*
5. Ayesha, H., Iqbal, S., Tariq, M., Abrar, M., Sanaullah, M., Abbas, I., Rehman, A., Niazi, M. F. and Hussain, S. (2021) ‘Automatic Medical Image Interpretation: State of the art and Future Directions’, *Pattern Recognition*, 114, p. 107856.
6. Tang, J., McGoldrick, G., Al-Ghossein, M. and Chen, C.-W. (2024) ‘Captions are worth a thousand words: Enhancing product retrieval with pretrained image-to-text models’, *Computer Vision and Pattern Recognition*. arXiv.
7. Luo, R. C., Hsu, Y.-T., Wen, Y.-C. and Ye, H.-J. (2019) ‘Visual image caption generation for service robotics and industrial applications’, *2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS)*, pp. 827–832.
8. Stefanini, M., Cornia, M., Baraldi, L., Cascianelli, S., Fiameni, G. and Cucchiara, R. (2021) ‘From show to tell: A survey on deep learning-based image captioning’, *Computer Vision and Pattern Recognition*, 45(1), pp. 539–559
9. Yang, L. and Hu, H. (2019) ‘Adaptive syncretic attention for constrained image captioning’, *Neural Processing Letters*, 50(1), pp. 549–564.
10. *Common objects in context* *COCO*. Microsoft. Available at: https://cocodataset.org/#home
11. Makav, B. and Kilic, V. (2019) ‘A new image captioning approach for visually impaired people’, *2019 11th International Conference on Electrical and Electronics Engineering (ELECO)*
12. Putra, B. T., Soni, P., Marhaenanto, B., Pujiyanto, Sisbudi Harsono, S. and Fountas, S. (2020) ‘Using information from images for plantation monitoring: A review of solutions for Smallholders’, *Information Processing in Agriculture*, 7(1), pp. 109–119.
13. Hossain, MD. Z., Sohel, F., Shiratuddin, M. F. and Laga, H. (2019) ‘A comprehensive survey of Deep Learning for Image captioning’, *ACM Computing Surveys*, 51(6), pp. 1–36.
14. Sutskever, I., Vinyals, O. and V. Le, Q. (2014) ‘Sequence to sequence learning with neural networks’, *NIPS’14: Proceedings of the 28th International Conference on Neural Information Processing Systems*, 2, pp. 3104–3112.
15. Stefanini, M., Cornia, M., Baraldi, L., Cascianelli, S., Fiameni, G. and Cucchiara, R. (2023b) ‘From Show to Tell: A survey on deep learning-based image captioning’, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1), pp. 539–559.
16. Vinyals, O., Toshev, A., Bengio, S. and Erhan, D. (2017) ‘Show and tell: Lessons learned from the 2015 MSCOCO Image captioning challenge’, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4), pp. 652–663.
17. Singh, H., Sharma, A. and Pant, M. (2024) ‘Pixels to Prose: Understanding the art of Image Captioning’, *Computer Vision and Pattern Recognition*. arXiv
18. Herdade, S., Kappeler, A., Boakye, K. and Soares, J. (2019) ‘Image Captioning: Transforming Objects into Words’, *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pp. 11137–11147.
19. Wang, C., Yang, H., Bartz, C. and Meinel, C. (2016) ‘Image captioning with deep bidirectional lstms’, *Proceedings of the 24th ACM international conference on Multimedia*, pp. 988–997.
20. Jia, X., Gavves, E., Fernando, B. and Tuytelaars, T. (2015) ‘Guiding the long-short term memory model for image caption generation’, *2015 IEEE International Conference on Computer Vision*
21. You, Q., Jin, H., Wang, Z., Fang, C. and Luo, J. (2016) ‘Image captioning with semantic attention’, *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
22. Huang, L., Wang, W., Chen, J. and Wei, X.-Y. (2019) ‘Attention on attention for image captioning’, *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*.
23. Liu, W., Chen, S., Guo, L., Zhu, X. and Liu, J. (2021) ‘CPTR: Full Transformer Network for Image Captioning’, *Computer Vision and Pattern Recognition*. arXiv
24. Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T. and Xie, S. (2022) ‘A ConvNet for the 2020s’, *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
25. Ramos, L., Casas, E., Romero, C., Rivas-Echeverría, F. and Morocho-Cayamcela, M. E. (2024) ‘A Study of ConvNeXt Architectures for Enhanced Image captioning’, *IEEE Access*, 12
26. Cahyono, J. A. and Jusuf, J. N. (2024) ‘Automated Image Captioning with CNNs and Transformers’, *Computer Vision and Pattern Recognition*. arXiv
27. Atliha, V. and Sesok, D. (2021) ‘Pretrained word embeddings for image captioning’, *2021 IEEE Open Conference of Electrical, Electronic and Information Sciences (eStream)*
28. Karpathy, A. and Fei-Fei, L. (2017) ‘Deep Visual-Semantic Alignments for Generating Image Descriptions’, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4)

|  |
| --- |
| **Research Ethics Review Form for MSc Projects**  **Computer Science Research Ethics Committee (CSREC)**  <https://www.city.ac.uk/about/governance/committees/cs-research-ethics> |

Postgraduate students undertaking their final project in the Department of Computer Science must consider the ethics of their project work and ensure that it complies with research ethics guidelines and the law for data protection. In some cases, a project will need approval from an ethics committee before it can proceed. Usually, but not always, this will be because the student is involving other people (“participants”) in the project.

To ensure that they give appropriate consideration to ethical issues, all students must complete this form and attach it to their project proposal document. There are two parts:

***PART A: Ethics Checklist***. All students must complete this part.

The checklist identifies whether the project requires ethical approval and, if so, where to apply for approval.

***PART B: Ethics Proportionate Review Form****.* Students who have answered “no” to all questions in A1, A2 and A3 and “yes” to question 4 in A4 in the ethics checklist must complete part B as well. The project supervisor has delegated authority to provide approval in such cases that are considered to involve MINIMAL risk. The approval may be ***provisional*** *– identifying the planned work with human end user participants as**likely* to involve MINIMAL RISK. In such cases you must additionally seek ***full approval*** from the supervisor as the project progresses and details are established. ***Full approval*** must be acquired in writing, before recruiting and engaging with human end users participants for your project.

**Part A: Ethics Checklist**

|  |  |  |
| --- | --- | --- |
| **A.1 If you answer YES to any of the questions in this block, you must apply to an appropriate external ethics committee for approval and log this approval as an External Application through Research Ethics Online -** [**https://researchmanager.city.ac.uk/**](https://researchmanager.city.ac.uk/do/city-login/login)**. This type of research is not covered by City’s process, and external approval from an appropriate institution is required.** | | *Delete as appropriate* |
| 1.1 | Does your research require approval from the National Research Ethics Service (NRES)?  e.g. because you are recruiting current NHS patients or staff?  If you are unsure try - <https://www.hra.nhs.uk/approvals-amendments/what-approvals-do-i-need/> | **NO** |
| 1.2 | Will you recruit participants who are covered by the Mental Capacity Act 2005?  Such research needs to be approved by an external ethics committee such as NRES or the Social Care Research Ethics Committee - <http://www.scie.org.uk/research/ethics-committee/> | **NO** |
| 1.3 | Will you recruit any participants who are covered by the Criminal Justice System, for example, people on remand, prisoners and those on probation?  Such research needs to be authorised by the ethics approval system of the National Offender Management Service. | **NO** |
| **A.2 If you answer YES to any of the questions in this block, then unless you are applying to an external ethics committee, you must apply for approval from the Senate Research Ethics Committee (SREC) through Research Ethics Online -**  [**https://researchmanager.city.ac.uk/**](https://researchmanager.city.ac.uk/do/city-login/login) | | *Delete as appropriate* |
| 2.1 | Does your research involve participants who are unable to give informed consent?  For example, people who may have a degree of learning disability or mental health problem, that means they are unable to make an informed decision on their own behalf. | **NO** |
| 2.2 | Is there a risk that your research might lead to disclosures from participants concerning their involvement in illegal activities? | **NO** |
| 2.3 | Is there a risk that obscene and or illegal material may need to be accessed for your research study (including online content and other material)? | **NO** |
| 2.4 | Does your project involve participants disclosing information about protected characteristics (as identified by the Equality Act 2010)?  *For example, to: racial or ethnic origin; political opinions; religious beliefs; trade union membership; physical or mental health; sexual life; criminal offences and proceedings* | **NO** |
| 2.5 | Does your research involve you travelling to another country outside of the UK, where the Foreign & Commonwealth Office has issued a travel warning that affects the area in which you will study?  *Please check the latest guidance from the FCO -* [*http://www.fco.gov.uk/en/*](http://www.fco.gov.uk/en/) | **NO** |
| 2.6 | Does your research involve invasive or intrusive procedures?  These may include, but are not limited to, electrical stimulation, heat, cold or bruising. | **NO** |
| 2.7 | Does your research involve animals? | **NO** |
| 2.8 | Does your research involve the administration of drugs, placebos or other substances to study participants? | **NO** |
| **A.3 If you answer YES to any of the questions in this block, then unless you are applying to an external ethics committee or the Senate Research Ethics Committee (SREC), you must apply for approval from the Computer Science Research Ethics Committee (CSREC) through Research Ethics Online -** [**https://researchmanager.city.ac.uk/**](https://researchmanager.city.ac.uk/do/city-login/login)**. Depending on the level of risk associated with your application, it may be referred to the Senate Research Ethics Committee.** | | *Delete as appropriate* |
| 3.1 | Does your research involve participants who are under the age of 18? | **NO** |
| 3.2 | Does your research involve adults who are vulnerable because of their social, psychological or medical circumstances (vulnerable adults)?  This includes adults with cognitive and / or learning disabilities, adults with physical disabilities and older people. | **NO** |
| 3.3 | Are participants recruited because they are staff or students of City, University of London?  For example, students studying on a particular course or module.  If yes, then approval is also required from the Head of Department or Programme Director. | **NO** |
| 3.4 | Does your research involve intentional deception of participants? | **NO** |
| 3.5 | Does your research involve participants taking part without their informed consent? | **NO** |
| 3.5 | Is the risk posed to participants greater than that in normal working life? | **NO** |
| 3.7 | Is the risk posed to you, the researcher(s), greater than that in normal working life? | **NO** |

|  |  |  |
| --- | --- | --- |
| **A.4 If you answer YES to the following question and your answers to all other questions in sections A1, A2 and A3 are NO, then your project is deemed to be of MINIMAL RISK.**  **If this is the case, then you can apply for approval through your supervisor under PROPORTIONATE REVIEW. You do so by completing PART B of this form.**  **If you have answered NO to all questions on this form, then your project does not require ethical approval. You should submit and retain this form as evidence of this.** | | *Delete as appropriate* |
| 4 | Does your project involve human participants or their identifiable personal data?  *For example, as interviewees, respondents to a survey or participants in testing.* | **NO** |

# D. Appendix: GenAI Prompts and Responses

**Prompt 1**: “Take the torch.nn.TransformerDecoderLayer source code attached below. Consolidate the logic of \_sa\_block, \_mha\_block, and \_ff\_block into a single forward method. The new forward must retain the self.norm\_first conditional structure and return the self-attention and cross-attention weight tensors.

def forward(self, tgt, memory, tgt\_mask, memory\_mask, tgt\_key\_padding\_mask, memory\_key\_padding\_mask, tgt\_is\_causal, memory\_is\_causal):

x = tgt

if self.norm\_first:

x = x + self.\_sa\_block(

self.norm1(x), tgt\_mask, tgt\_key\_padding\_mask, tgt\_is\_causal

)

x = x + self.\_mha\_block(

self.norm2(x),

memory,

memory\_mask,

memory\_key\_padding\_mask,

memory\_is\_causal,

)

x = x + self.\_ff\_block(self.norm3(x))

else:

x = self.norm1(

x + self.\_sa\_block(x, tgt\_mask, tgt\_key\_padding\_mask, tgt\_is\_causal)

)

x = self.norm2(

x

+ self.\_mha\_block(

x, memory, memory\_mask, memory\_key\_padding\_mask, memory\_is\_causal

)

)

x = self.norm3(x + self.\_ff\_block(x))

return x

# self-attention block

def \_sa\_block(

self,

x: Tensor,

attn\_mask: Optional[Tensor],

key\_padding\_mask: Optional[Tensor],

is\_causal: bool = False,

) -> Tensor:

x = self.self\_attn(

x,

x,

x,

attn\_mask=attn\_mask,

key\_padding\_mask=key\_padding\_mask,

is\_causal=is\_causal,

need\_weights=False,

)[0]

return self.dropout1(x)

# multihead attention block

def \_mha\_block(

self,

x: Tensor,

mem: Tensor,

attn\_mask: Optional[Tensor],

key\_padding\_mask: Optional[Tensor],

is\_causal: bool = False,

) -> Tensor:

x = self.multihead\_attn(

x,

mem,

mem,

attn\_mask=attn\_mask,

key\_padding\_mask=key\_padding\_mask,

is\_causal=is\_causal,

need\_weights=False,

)[0]

return self.dropout2(x)

# feed forward block

def \_ff\_block(self, x: Tensor) -> Tensor:

x = self.linear2(self.dropout(self.activation(self.linear1(x))))

return self.dropout3(x)”

The consolidated forward method received as a response is referenced in transformerDecoderAttVis.py file and stated below:

def forward(self, tgt, memory= None, tgt\_mask= None, memory\_mask = None, tgt\_key\_padding\_mask= None, memory\_key\_padding\_mask= None, is\_causal= False, output\_attentions = False):

x = tgt

attn\_weights\_sa = None

if self.norm\_first:

\_self\_attn\_input = self.norm1(x)

else:

\_self\_attn\_input = x

\_self\_attn\_output, attn\_weights\_sa = self.self\_attn(\_self\_attn\_input, \_self\_attn\_input, \_self\_attn\_input, attn\_mask=tgt\_mask, key\_padding\_mask=tgt\_key\_padding\_mask, is\_causal=is\_causal, need\_weights=output\_attentions, average\_attn\_weights=False)

x = x + self.dropout1(\_self\_attn\_output)

if not self.norm\_first:

x = self.norm1(x)

attn\_weights\_ca = None

if memory is not None:

if self.norm\_first:

\_cross\_attn\_input = self.norm2(x)

else:

\_cross\_attn\_input = x

\_cross\_attn\_output, attn\_weights\_ca = self.multihead\_attn(\_cross\_attn\_input, memory, memory, attn\_mask=memory\_mask, key\_padding\_mask=memory\_key\_padding\_mask, need\_weights=output\_attentions, average\_attn\_weights=False)

x = x + self.dropout2(\_cross\_attn\_output)

if not self.norm\_first:

x = self.norm2(x)

if self.norm\_first:

\_ffn\_input = self.norm3(x)

else:

\_ffn\_input = x

\_ffn\_output = self.linear2(self.dropout\_ffn(self.activation(self.linear1(\_ffn\_input))))

x = x + self.dropout3(\_ffn\_output)

if not self.norm\_first:

x = self.norm3(x)

return x, attn\_weights\_sa, attn\_weights\_ca

**Prompt 2:** “Given the following code which collects the cross-attention weights for all layers into the list all\_layer\_cross\_attentions\_for\_step:

all\_layer\_cross\_attentions\_for\_step = []

for layer\_idx, layer in enumerate(self.decoder\_layers):

layer\_output, self\_attn\_weights, cross\_attn\_weights = layer(

current\_layer\_output,

encoder\_out[:, active\_indices, :],

tgt\_mask=tgt\_mask,

output\_attentions=True)

current\_layer\_output = layer\_output

all\_layer\_cross\_attentions\_for\_step.append(cross\_attn\_weights)

how would I get the average cross attention weights per token for the current word and update the alphas accordingly?”

The output code received was:

stacked\_cross\_attentions = torch.stack(all\_layer\_cross\_attentions\_for\_step, dim=0)

cross\_attn\_for\_current\_token = stacked\_cross\_attentions[:, :, :, -1, :]

avg\_cross\_attention\_per\_token = cross\_attn\_for\_current\_token.mean(dim=(0, 2))

alphas[active\_indices, t, :] = avg\_cross\_attention\_per\_token

# E. Appendix: Code Availability

The code is open source at the project’s GitHub repository linked at: <https://github.com/sa06840/ImageCaptioningConvNeXt.git>

Apart from this, the entire project code has been submitted separately as a projectCode.pdf file on Moodle.

All the trained models and results files are uploaded on one drive linked at: [ImageCaptioning](https://cityuni-my.sharepoint.com/:f:/g/personal/sajeel_alam_city_ac_uk/Eq0vkdjw41FAsU620KXPQ9oB_wCTyFsYinSAuxY6sP3Vcw?e=9QUeDF)

# F. Appendix: Word Count Table

|  |  |
| --- | --- |
| **Chapter** | **Word Count** |
| Chapter 1: Introduction and Objectives | 1258 |
| Chapter 2: Critical Context | 2095 |
| Chapter 3: Methods | 3787 |
| Chapter 4: Results | 4534 |
| Chapter 5: Discussion | 2489 |
| Chapter 6: Evaluation, Reflections and Conclusions | 891 |

Total words: 15054