A red and black logo

Description automatically generated

MSc in Data Science

Project Report

2025/26

Enhancing Image Captioning using ConvNeXt with LSTM and Transformer Decoders

Sajeel Nadeem Alam

Supervised by: Dr. Kevin Ryan

27th 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

Keywords:

# Table Of Contents

[Abstract 2](#_Toc208431831)

[Table Of Contents 3](#_Toc208431832)

[1. Introduction and Objectives 5](#_Toc208431833)

[1.1. Background 5](#_Toc208431834)

[1.2. Rationale and Beneficiaries 5](#_Toc208431835)

[1.3. Project Objectives 6](#_Toc208431836)

[1.4. Methodology and Work Plan 7](#_Toc208431837)

[1.5. Report Structure 7](#_Toc208431838)

[2. Critical Context 7](#_Toc208431839)

[2.1. Early Image Captioning Techniques 7](#_Toc208431840)

[2.2. Encoder – Decoder Architecture 7](#_Toc208431841)

[2.3. Improvements in the Encoder – Decoder Architecture 8](#_Toc208431842)

[2.4. Self-Attention and Transformers 9](#_Toc208431843)

[2.5. ConvNeXt Encoder 9](#_Toc208431844)

[2.6. Training, Pretrained Word Embeddings and Decoding Strategies 10](#_Toc208431845)

[3. Methods 11](#_Toc208431846)

[3.1. Dataset and Preprocessing 11](#_Toc208431847)

[3.1.1. Dataset 11](#_Toc208431848)

[3.1.2. Preprocessing 11](#_Toc208431849)

[3.1.3. Dataloader 12](#_Toc208431850)

[3.2. Model Architecture 12](#_Toc208431851)

[3.2.1. ConvNeXt Encoder 12](#_Toc208431852)

[3.2.2. Baseline Decoder: LSTM with Attention 13](#_Toc208431853)

[3.2.3. Proposed Decoder: Transformer Decoder 14](#_Toc208431854)

[3.3. Experimental Design 14](#_Toc208431855)

[3.3.1. Training Strategies 14](#_Toc208431856)

[3.3.2. Finetuning ConvNeXt 15](#_Toc208431857)

[3.3.3. Supplementary Analysis of Attention Regularization 16](#_Toc208431858)

[3.3.4. Word Embeddings 16](#_Toc208431859)

[3.3.5. Decoding Strategy 17](#_Toc208431860)

[3.4. Training and Hyperparameters 17](#_Toc208431861)

[3.5. Evaluation Metrics 18](#_Toc208431862)

[4. Results 19](#_Toc208431863)

[4.1. Preliminary Results 19](#_Toc208431864)

[4.1.1. Results on Flickr8k 19](#_Toc208431865)

[4.1.2. Results with Original Study’s Codebase 20](#_Toc208431866)

[4.2. Baseline Performance on MS COCO 20](#_Toc208431867)

[4.2.1. The Impact of Training Strategies 20](#_Toc208431868)

[4.2.2. Selecting the Best Decoder 23](#_Toc208431869)

[4.3. The Impact of Finetuning ConvNeXt 25](#_Toc208431870)

[4.3.1. Quantitative Analysis 25](#_Toc208431871)

[4.3.2. Qualitative Analysis 27](#_Toc208431872)

[4.4. Analysis of Attention Regularization 28](#_Toc208431873)

[4.5. The Impact of Pretrained Word Embeddings 29](#_Toc208431874)

[4.5.1. Quantitative Analysis 29](#_Toc208431875)

[4.5.2. Qualitative Analysis 30](#_Toc208431876)

[4.6. Greedy Search vs Beam Search 31](#_Toc208431877)

[5. Discussion 32](#_Toc208431878)

[5.1. Addressing Project Objectives 33](#_Toc208431879)

[5.1.1. Implement the ConvNeXt encoder, LSTM and Transformer decoders 33](#_Toc208431880)

[5.1.2. Compare the model’s performance training with and without teacher forcing 33](#_Toc208431881)

[5.1.3. Train and select the best decoder on the basis of their baseline performance 34](#_Toc208431882)

[5.1.4. Analyze the performance of finetuning different layers of the ConvNeXt 34](#_Toc208431883)

[5.1.5. Train the architecture with pretrained Word2Vec and GloVe embeddings 35](#_Toc208431884)

[5.1.6. Compare caption quality with greedy search and beam search 35](#_Toc208431885)

[5.2. Answering the Research Question 35](#_Toc208431886)

[5.3. Broader Discussions and Implications 36](#_Toc208431887)

[5.3.1. Architectural and Practical Implications 36](#_Toc208431888)

[5.3.2. Training and Methodological Implications 36](#_Toc208431889)

[6. Evaluation, Reflection and Conclusion 37](#_Toc208431890)

[6.1. Project Evaluation 37](#_Toc208431891)

[6.2. Reflection 37](#_Toc208431892)

[6.3. Conclusion 37](#_Toc208431893)

[Glossary 37](#_Toc208431894)

[References 37](#_Toc208431895)

[Appendices and additional files 39](#_Toc208431896)

# 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 (NLP), each presenting its own set of challenges. Image captioning is usually treated as supervised learning. 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 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.

## 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 cross and self-attention along with 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 secondary area of investigation further motivating this study is to select a well-performing, pre-trained image model and find the optimal depth of layers to finetune it for this downstream task. The rationale for improving computers’ ability in image captioning is also 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 image captioning 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 some 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:** Finetune 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 finetuning 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 finetuning strategy for the ConvNeXt.

1. **Objective:** Train the architecture with pre-trained 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.

## 1.4. Methodology and Work Plan

The methodology of this project follows a sequential process beginning with 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 finetuning 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

This structure of the report is as follows. Chapter 2 provides 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.

# 2. Critical Context

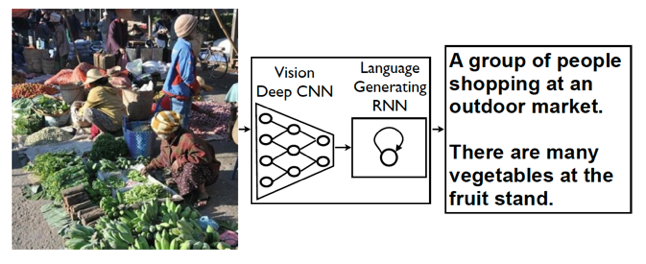
This chapter provides a critical review of the existing works in the field of image captioning. It shows how techniques have built on top of one another and improved over time, and also acts a source of motivation for the methods used in this study.

## 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 the context of each individual image (Hossain et al., 2019).

## 2.2. 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) 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., 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.



**Figure 1.** Basic encoder – decoder architecture (Vinyals et al., 2017)

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. 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

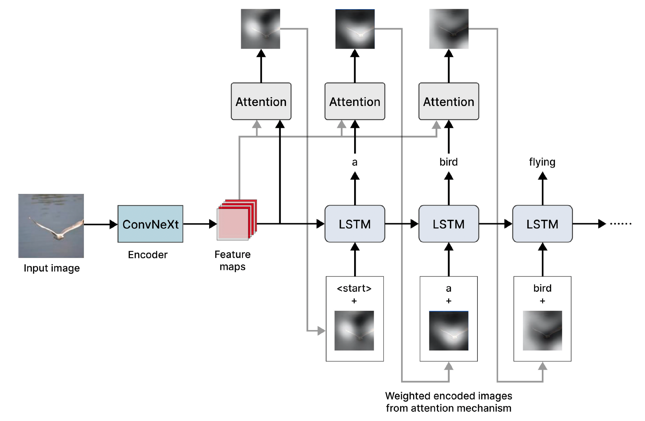
In order to overcome the issues mentioned earlier, researchers have tried to modify individual components of the encoder-decoder architecture. In one paper, researchers used Faster R-CNN with ResNet-101 to identify objects in the image, encode their geometric information and spatial relationships, and 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 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.

## 2.4. Self-Attention and Transformers

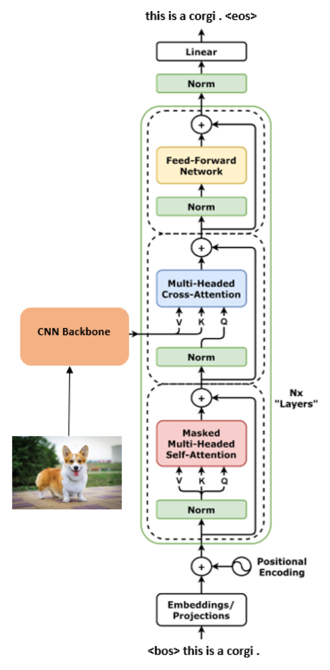
In recent years, with the introduction of self-attention and transformers, researchers have explored their potential use in overcoming the limitation of global context faced in CNN-LSTM architectures as mentioned earlier. In one study, researchers incorporated a self-attention mechanism in both the CNN 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.

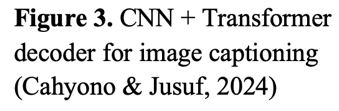
## 2.5. ConvNeXt Encoder

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 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).



**Figure 2.** ConvNeXt + LSTM (with attention module) for image captioning (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, 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.

## 2.6. Training Strategies

The study (Bengio, 2015) highlights that training an encoder-decoder architecture using teacher forcing in which the word from the actual caption is 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, 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 observe the difference in loss and find the balance between generalization and how fast the model converges.

## 2.7. Pretrained Word Embeddings and Decoding Strategies

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 (Vinyals et al., 2017). This study aims to do the same. Moreover, researchers have investigated the effect of pre-trained Word2Vec, GloVe and random word embeddings 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 (Atliha et al, 2021). This study has an objective to do the same while fine-tuning the embeddings during training.

# 3. Methods

This chapter goes through the experimental and technical setup of the image captioning system developed in this study. The methodology explains the dataset used, the pipeline built to load it into the model, the baseline and proposed model architectures, the experiments conducted and the evaluation metrics used.

## 3.1. Dataset and Preprocessing

### 3.1.1. Dataset

The dataset used in this study is the MS COCO dataset (Microsoft, 2014) which was originally 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 in the community and allowing for meaningful comparisons of this study’s results with existing works. In the initial stages of the study, the smaller Flickr8k dataset was used to test the robustness of the architecture with fewer number of 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 at least five captions providing variety for evaluation.

### 3.1.2. Preprocessing

The image and caption data were preprocessed using multiple steps for efficient data loading and model training. Firstly, the Karpathy split file is loaded as a JSON which contains image metadata and a list of captions for each image, already 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 vocabulary, or a word map is made in which each word is mapped to a unique index and words appearing less than 5 times across all the captions are not included to mitigate the effect of rare words. Four special tokens such as <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 consistently across model training and inference.

(can insert json file image)

The core of the preprocessing involves converting raw .jpg images and captions into a format suitable for the model. 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. This file format allows for efficient loading of image data which is essential for training using large datasets. The second file consist of all the captions stored in JSON format. For each image, its five captions are sampled and then encoded into numerical sequences using the generated word map. Each sequence is prepended with the <start> token, appended with the <end> token and then padded with the <pad> token until it reaches a maximum caption of length of 50. 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 loading of data from the .HDF5 and JSON files, a custom PyTorch dataset class is set up. This dataset class ensures that the large .HDF5 file containing all the image data is not loaded altogether at the time of initialization. Instead, lazy loading is implemented which opens the .HDF5 file only once at the time of first data retrieval and the subsequent images are retrieved only when required. This approach prevents out of memory (OOM) issues from loading the entire image data at once. 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 important for validation and testing since every image requires all five of its captions to calculate the evaluation metrics.

This dataset is passed into PyTorch’s dataloader which manages the loading of data in batches to feed the model. In this study, the batch size is 32 so the dataloader loads 32 images, captions and their lengths together 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. These 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. These careful data loading and batching strategies are crucial for loading data efficiently thus speeding up the training and inference process.

## 3.2. Model Architecture

### 3.2.1. ConvNeXt Encoder

As mentioned earlier, this study uses an encoder-decoder architecture in which the encoder is a ConvNeXt. The ConvNeXt is an improved CNN built by the research team at Meta which builds on the foundational ResNet model and incorporates features from vision transformers. Upgrades such as using larger convolution kernels allow the model to gather broader context across the image while a hierarchical feature learning approach inspired from Swin Transformers processes the image at different resolutions. This enables 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 study uses 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 the use case of image captioning, the pooling layer and classification head are removed since ConvNeXt is used only to extract image features. The feature extractor of ConvNeXt has seven sequential layers, each containing several convolutional 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. Initially, all the layers in the feature extractor are frozen to get a baseline performance and in later experiments, certain layers are unfrozen to check the effect on the performance of the architecture. This flexibility allows the study to empirically investigate how fine-tuning till certain depths of the encoder affects the model’s overall performance.

### 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 or alpha scores which are multiplied element-wise to the encoder’s image features and then summed to get an attention-weighted context vector which highlights the most 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.

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 also has a dimension of 512. In the case of teacher forcing, this word is the ground truth and in the case of no teacher forcing, this word is the model’s own prediction from the previous step. 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 next 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 to replace the LSTM. The inherent parallel processing ability of the transformer considers all regions of the image and the generated caption simultaneously, providing global context during the caption generation process (Liu et al., 2021). 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. The positional encoded embeddings of the caption along with the image features from the encoder are passed into the transformer decoder which contains two attention mechanisms. The masked multi-headed self-attention allows the decoder to attend to all the previous words in the caption sequence. A mask is applied to prevent the decoder from looking at the future words while generating the next word. The multi-headed cross-attention enables the decoder to create a visual context vector using the image features which makes it focus on the relevant parts of the image while generating the next word. In the case of teacher forcing, the true caption is embedded followed by positional encoding before being fed into the transformer decoder along with the image features to generate the caption. In the case of no teacher forcing, the caption is generated word by word instead of in parallel and in order to generate the next word, all the previously generated words are embedded, positionally encoded and then fed into the decoder along with the image features. In the case of greedy search, 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.

In both cases, the output from the transformer decoder is a sequence of feature vectors each with a dimension of 512, with the sequence being the length of the maximum caption length. 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 different experiments and strategies that were used to analyze the architecture’s performance. It is important to note that the encoder and LSTM decoder is inspired from the codebase (Ramos et al., 2024) so that it can be compared with the transformer decoder in this study.

### 3.3.1. Training Strategies

The models were trained using both teacher forcing and non-teacher forcing training strategies in the decoder to implement what was done in the study (Ramos et al., 2024). It is important to note that each image has five true captions and the model generates a caption for every true caption hence five captions are generated for each image. In the case of teacher forcing, previous words of the true caption are provided to the decoder at each step to generate the next word. In non-teacher forcing the model’s own previously generated words are used at each step to generate the next word making it an autoregressive approach.

However, while replicating the original codebase it was discovered that although their study stated that the model was trained without teacher-forcing as well 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 not correct 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 just to replicate what was done in the original study. Then non-teacher forcing is implemented from scratch 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 was selected for further experiments.

### 3.3.2. Finetuning ConvNeXt

In the original study (Ramos et al., 2024), the authors chose not to fine-tune the ConvNeXt and relied on the pre-trained weights. However, a key area of investigation in this study is to evaluate how the depth of fine-tuning the pre-trained ConvNeXt during training affects the overall performance. Earlier layers are responsible for more general features such as edges and lines whereas later layers recognize more complex patterns that are specific to the task making this an interesting investigation. The hypothesis is that fine-tuning deeper layers will yield better results. As mentioned earlier, the ConvNeXt has 7 sequential layers. In the experiments where the encoder was fine-tuned, it was frozen for the first 20 epochs to allow the gradients of the decoder to reach some stability and avoid corrupting the pre-trained weights of the ConvNeXt with large initial updates. There were four main scenarios when it comes to fine-tuning the ConvNeXt in this study as shown below:

1. **Frozen encoder – no fine-tuning**: All the layers of the ConvNeXt were frozen and no fine-tuning was done to replicate the original 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 which performed the best in this scenario was then used for further experiments.
2. **Fine-tuning layers 5-7:** The last three layers of the ConvNeXt were fine-tuned. The initial experiment in this scenario used a learning rate of 1×10-4 with a patience of 20 epochs for early stopping. To investigate the effects of a more 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:** This experiment was conducted 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:** In this experiment, the entire ConvNeXt encoder was fine-tuned to explore the effects of adapting the full network. To ensure stable convergence and prevent significant alteration of the pre-trained features, a low learning rate of 1×10-6 was used with a patience of 40 epochs.

For each experiment, the model which resulted in the best performance on the validation set during training was saved as a checkpoint. The results of these checkpoints on the validation set were then used for comparison.

### 3.3.3. Supplementary Analysis of Attention Regularization

In the original study (Ramos et al., 2024), in the case of the LSTM decoder, a doubly stochastic attention regularization loss was added to the total loss. The 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, after conducting the experiments of fine-tuning the ConvNeXt, a qualitative analysis of the attention maps revealed that although the transformer decoder performed well, it had a tendency to focus only on a few specific regions throughout the caption generation process. To mitigate this behavior and investigate its impact, supplementary experiments were conducted in which an attention regularization identical to the one used with the LSTM, was implemented and added to the training loss of the transformer. The purpose of this was not to achieve a primary objective but to investigate whether training the transformer with this regularization would encourage it to shift its attention across the image.

### 3.3.4. Word Embeddings

Another key objective of this study was to investigate the effect of prior linguistic knowledge on the quality of generated captions. For this purpose, after selecting the best performing decoder, training strategy and ConvNeXt fine-tuned layers, various word embeddings were evaluated. For the initial experiments, 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 process.

To explore the effect of external knowledge on the model’s performance, two more experiments were conducted with pre-trained word embeddings while keeping the rest of the configuration constant. The decoder’s embedding layer was initialized with vectors from both 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 [https://huggingface.co/fse/word2vec-google-news-300], whereas the GloVe embeddings have a dimension of 200 and were pretrained on 2B tweets, 27B tokens, 1.2M vocab [https://huggingface.co/fse/glove-wiki-gigaword-200]. 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 specifically to the image captioning task. The models’ performances were then 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. As mentioned earlier, at every word position the decoder calculates logit scores across all the words in the vocabulary. In order to select the next word, this study implements two decoding strategies which are greedy search and beam search. In greedy search, the model selects the word with the highest logit score at every word position in the generated caption. This becomes the input for the next time step. 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 one 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. [add reference]

In all the experiments greedy search is used at the time of inference on the validation and test sets for quantitative analysis and comparing models, since it provides a computationally efficient baseline for calculating evaluation metrics. However, for qualitative analysis and generating captions for a sample of unseen images, beam search with a beam size of k=5 was utilized 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 caption for this use case.

## 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. The 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 cycle. Once training is complete, the best checkpoint is used for testing on the test set.

In order to train the encoder-decoder architecture, images are loaded in batches of size 32 using the dataloader. For each batch, images are passed to the ConvNeXt which extracts image 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 a set of raw scores known as 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. As mentioned earlier, in the case of the LSTM decoder, a doubly stochastic attention regularization loss is added to this total loss to encourage the model to focus on all parts of the image rather than just one area. The loss is backpropagated throughout the architecture and the 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 the same thus the predicted logits are always aligned with the ground truth word labels to calculate the loss. To ensure that the loss calculation is performed only on relevant tokens, the padding tokens following the <end> token are excluded from both the predicted logits and the ground truth labels. In non-teacher forcing, there is a possibility that the generated caption may be shorter or longer than the true caption with a maximum caption length set at 50. To overcome this challenge, 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 any padding tokens at the end of the predicted logits or true captions are filtered out. This aligns the predicted logits and ground truth word labels for loss calculation on only the relevant tokens.

The models were trained using two NVIDIA A-100 40GB GPUs. A multi-GPU training system was set up in which the number of batches are divided amongst the two GPUs thus speeding up the training process which was essential when dealing with large image data. Each GPU calculated its own local loss and during backpropagation the gradients of this local loss with respect to every parameter on that specific GPU were calculated. However, PyTorch’s Distributed Data Parallel package collects the gradients for all model parameters from both the GPUs and averages them before broadcasting them back to both the GPUs. 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 gave a strong insight into both the training process and the quality of the generated captions. Firstly, the efficiency of the training process was calculated using the data time which measured the time taken to load a single batch of data, and the batch time which measured the duration of a single training iteration. Both these metrics were measured for each batch and then averaged over an entire epoch.

The cross-entropy loss was calculated to give an insight into how well the model’s predicted probability distribution aligns with the actual ground truth words. A lower loss indicates a higher probability assigned to the correct word indicating a more confident model. Only the actual words in the captions were included while the padding tokens were removed. In addition to the loss, 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 also both calculated at training, validation and test times.

At the time of inference, the quality of the generated captions is measured by the BLEU-scores. 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. In order to calculate it, the reference (true captions) and hypothesis (generated captions) corpora are stored. The references are a nested list in which each inner list contains the five ground-truth captions for a single image, whereas the hypotheses are a single list containing five generated captions for each image. 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 approach provides a robust and singular value, ranging from 0 to 1, that represents the overall quality of the model’s output. In this study, the BLEU-4 score is used 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.

In order to qualitatively assess the model’s performance, the best performing checkpoint during training is used to generate captions for sample unseen images from the test set. As mentioned earlier, this is done using beam search with a beam size of 5. The generated caption is assessed against the image and its actual captions to determine its quality. The original codebase (Ramos et al., 2024) has a function which generates the caption using beam search with the LSTM decoder and also stores the attention weights while predicting each word to later display the model’s corresponding 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 proposed 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

## 4.1. Preliminary Results

### 4.1.1. Results on Flickr8k

As mentioned earlier, in this study the system is initially run on the Flickr8k dataset to test the pipeline’s robustness and models’ convergence on a smaller dataset. This helped to configure the number of workers needed in the dataloader to optimize GPU usage during training. Using the default training setup with teacher forcing in training, frozen ConvNeXt, greedy decoding at inference and decoder learning rate as 1×10-4, initial experiments are run with the LSTM and transformer decoder and the results obtained are displayed in Table 1. It is important to note that these metrics correspond to the best performing epoch according to the validation BLEU-4 score during training which is saved and tested on the test set.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **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

It can be seen that 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

As mentioned earlier, both decoders are trained with teacher forcing (TF) 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 TF at training and inference

It can be observed that the results obtained in this study are comparable to the ones in the paper (Ramos et al., 2024) showcasing 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

Once the system’s robustness is validated and the original codebase’s LSTM is replicated, this study fixes the flaws by implementing training and inference without teacher forcing. In all future experiments including these ones, inference is done without teacher forcing. 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.

### The Impact of Training Strategies

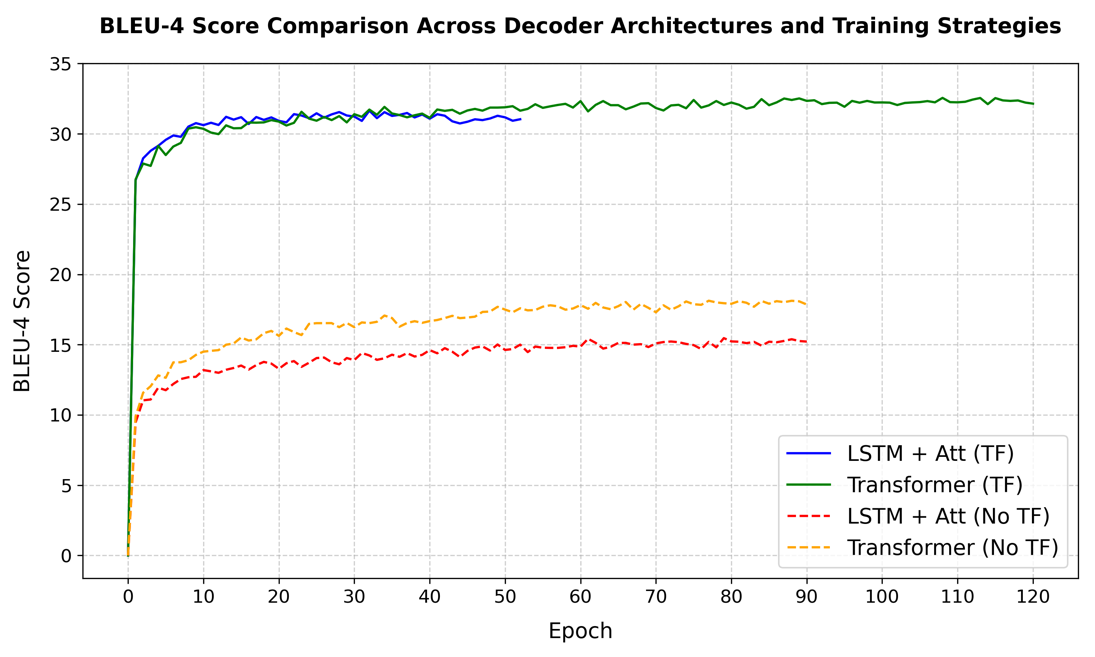
In order to investigate the impact of training strategies on the model’s performance, both the LSTM and transformer decoders are trained with teacher forcing (TF) and without teacher forcing. The training and validation evaluation metrics of the best performing checkpoint during training are displayed in Table 3 along with the best performing checkpoint from (Ramos et al., 2024).

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **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)

In Table 3 it can be seen that 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 exposure bias problem which is caused due to the mismatch in data distribution that the model is exposed to during training and inference. 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 data 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, comparing Tables 2 and 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 4.** BLEU-4 curves for LSTM and transformer decoders training with and without TF

Figure 4 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 an unseen image 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 shows 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 5.** Attention maps of LSTM decoder without teacher forcing (left) and with teacher forcing (right)

Figure 5 shows the corresponding attention maps of the LSTM decoder trained with and without teacher forcing. It can be seen that for the initial stop word ‘a’, both training strategies focus on the same spot in the image. 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 the case of training 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 the model to generate a grammatically correct caption.

Thus, 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.

### Selecting the Best Decoder

The core objective of this study was to implement a transformer decoder that is able to capture both local and global context from the image features and incorporate them accurately in the generated captions which would be compared to the LSTM + attention decoder in the original study (Ramos et al., 2024). For this purpose, both decoders were built and integrated separately in the image captioning architecture with ConvNeXt and trained using the same conditions. In section 4.2.1, it was observed that training with teacher forcing generates better results hence both decoders were trained with teacher forcing and their best checkpoints based on the validation BLEU-4 score was saved and tested on an unseen test set. The results are presented in Table 5.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **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 the results, it can be seen that 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. This supports the original hypothesis of the study which stated that since transformers process the image features and previously generated words in parallel 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 6.** Attention maps of LSTM + Att decoder (left) and Transformer decoder (right)

Figure 6 shows that multiplying the attention weights calculated by the attention module in the LSTM + Att model with the image features from the encoder, 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 the transformer relies on two attention mechanisms. The first is the cross-attention mechanism 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 the self-attention mechanism which guides the transformer on where to focus on its own previously generated words. Figure 6 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 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 decoder in terms of BLEU-4 scores during training and validation can be observed in Table 3 and Figure 4.

## 4.3. The Impact of Finetuning ConvNeXt

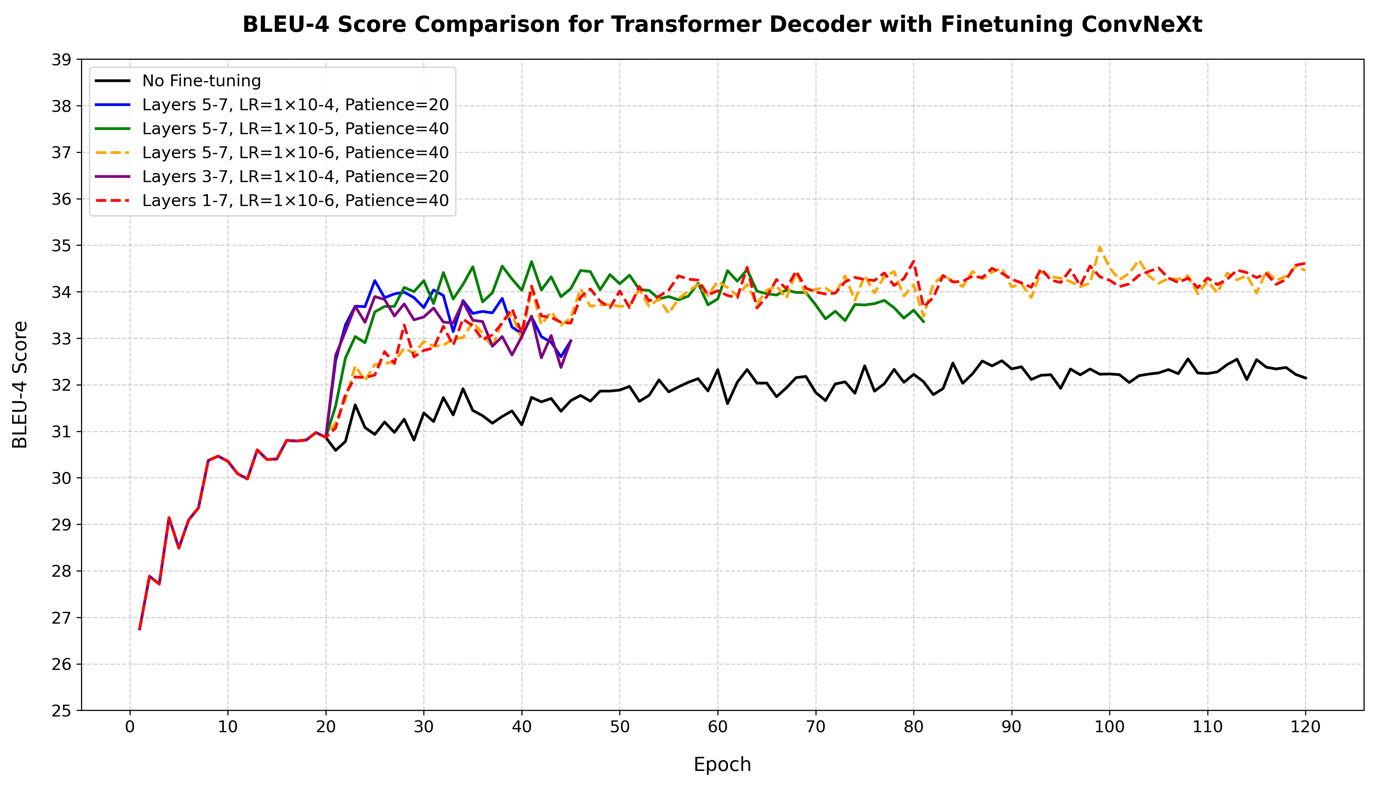
### 4.3.1. Quantitative Analysis

In the next set of experiments, different layers of the ConvNeXt are also finetuned 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 finetune. As mentioned earlier, these experiments are done using the transformer decoder trained with teacher forcing. Apart from the layers being finetuned, 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 Finetuned** | **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 finetuning different layers of ConvNeXt

According to Table 6, it can be seen that finetuning 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 finetuning deeper layers does not necessarily improve performance since finetuning layers 5-7 has a slightly better performance in terms of BLEU scores as compared to finetuning layers 3-7 and layers 1-7 while keeping the learning rate and patience constant. This proves that for the task of image captioning, finetuning later layers that are responsible for more complex features is enough since earlier layers are responsible for simple features like edges and lines which are consistent for any visual task hence finetuning 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, finetuning 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 which outperforms the original study (Ramos et al., 2024) that achieved the highest score of 34.76.



**Figure 7.** BLEU-4 curves with finetuning different layers of the ConvNeXt

Figure 7 shows that all the line graphs are identical for the first 20 epochs since the ConvNeXt is frozen. After epoch 20, the line graph when there is no finetuning has the least improvement across all epochs which supports the argument that finetuning the weights for the image captioning task does result in an improvement in BLEU scores. Finetuning 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. Finetuning 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 then flattens out and then gradually declines coming to an early stop around epoch 80. However, finetuning 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 in them moving in the right direction towards an optimal space without getting corrupted.

### Qualitative Analysis

For each configuration of finetuning, the checkpoint that has the highest validation BLEU-4 score during training is used to generate a caption for an unseen image. These captions are then compared to carry out a qualitative analysis to assess which configuration generates the most accurate captions and whether the BLEU-4 scores translate to the actual caption quality.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Layers Finetuned** | **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 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 standing next to a park bench |

**Table 7.** Captions generated by finetuning different layers of the ConvNeXt

Observing the generated captions in Table 7, it can be seen that when the ConvNeXt is not finetuned, a relatively accurate caption is generated however it lacks detail. After finetuning the ConvNeXt, the generated captions contain some 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 finetuning. Finetuning 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 bench is in a park. This is also mentioned in one of the true captions in Table 4. 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 8.** Attention maps of captions generated by Transformer decoder with fine-tuned ConvNeXt. Layers 5-7 fine-tuned with 1×10-4 learning rate (left) and 1×10-6 learning rate (right)

Figure 8 shows the attention maps of the captions generated by the transformer decoder with the ConvNeXt fine-tuned from layers 5-7 with learning rates 1×10-4 and 1×10-6. For the attention map on the left, it can be seen that 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, for the entire caption generation process most of its its focus stays constant on a general bottom-right region of the image. Whereas for the attention map on the right, the transformer decoder focuses on the same general bottom-right region throughout the caption generation process. Comparing this to Figure 6, 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 process. Despite this it is able generate good quality captions which as explained earlier in Section 4.2.2. could be because it relies more on its self-attention mechanism while generating captions.

## Analysis of Attention Regularization

A supplementary set of experiments were conducted to investigate the effect of attention regularization on the transformer decoder’s performance. As shown by the attention maps in Figures 6 and 8, the transformer decoder tends to only focus on a general region of the image while generating the caption. To encourage it to shifts 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 are saved.

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

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

**Figure 9.** 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 6 with Figure 9, it can be seen that with and without regularization in the transformer decoder when the ConvNeXt that 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 9 with Figure 8, it is observed that the attention maps of the transformer decoder with the fine-tuned ConvNeXt with and without regularization are 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 bottom-right region of the image. This is an interesting insight because it shows how 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, further experiments are conducted with training the transformer decoder without attention regularization.

## 4.5. The Impact of Pretrained Word Embeddings

After identifying the optimal finetuning strategy, 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 that had a dimension of 512 which were finetuned during the training process to act as a baseline performance. The experiment was repeated using 300-dimensional Word2Vec embeddings that were pretrained on part of the Google News dataset containing 100 billion words (FSE - Hugging Face, 2021) as well as using 200-dimensional GloVe embeddings that were pretrained on 2B tweets, 27B tokens, 1.2M vocab (FSE - Hugging Face, 2021). Both embeddings were also finetuned 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** | 7.62 | 32.69 | 78.39 | 60.42 | 46.15 | 34.50 |
| **GloVe** | 7.39 | 32.90 | 78.61 | 60.89 | 46.50 | **34.67** |

**Table 9.** Test metrics with different word embeddings

Table 9 shows that although the model performs slightly better with pretrained word embeddings, the improvement in performance is very minimal. The model with GloVe embeddings has the best performance with a BLEU-4 score of 34.67. 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 also qualitatively assess the quality of generated captions.

### 4.5.2. Qualitative Analysis

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

|  |  |  |
| --- | --- | --- |
|  | **Embeddings** | **Generated Caption** |
| Random | A white dog standing next to a park bench |
| Word2Vec | A man sitting on a bench next to a 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 caption generated with Word2Vec embeddings misses out on the colour of the dog however it incorporates the context of a man sitting on the bench which was missed by the caption generated using random embeddings. Moreover, the model using GloVe embeddings mentions that the dog is white along with the man sitting on the bench. Although the captions generated by the pretrained embeddings have slightly more detail, only the model using random embeddings was able to capture that the bench is in a park. 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

As mentioned earlier, for all the previous experiments greedy search is implemented at the time of inference with the validation and test sets to get the evaluation metrics for comparing models quantitatively whereas beam search is used to explore more possible sequences while generating the actual caption of sample images for comparing qualitatively. However, in this next set of experiments, the best performing model is used with both decoding strategies to generate captions for unseen images to determine which strategy performs better in this case.

|  |  |  |  |
| --- | --- | --- | --- |
| 1 |  | **Strategy** | **Generated Caption** |
| Greedy | A small airplane is parked on the grass |
| Beam (k=5) | 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 (k=5) | A motorcycle parked in a fenced in area |
| 3 |  | Greedy | A white dog sitting on a park bench next to a man |
| Beam (k=5) | A white dog standing next to a park bench |
| 4 |  | Greedy | A traffic light with a sky background |
| Beam (k=5) | A traffic light with a sky background |
| 5 |  | Greedy | A table with a vase of flowers on it |
| Beam (k=5) | A table with a vase of flowers on it |
| 6 |  | Greedy | A group of people standing around a park |
| Beam (k=5) | A group of people that are standing in the street |
| 7 |  | Greedy | A herd of sheep standing on top of a lush green field |
| Beam (k=5) | A herd of sheep standing on top of a lush green field |
| 8 |  | Greedy | A boat with a flag on it and a flag on the back |
| Beam (k=5) | A boat filled with lots of items floating on top of a river |
| 9 |  | Greedy | A giraffe is standing in the grass next to a tree |
| Beam (k=5) | A herd of giraffe standing on top of a lush green field |
| 10 |  | Greedy | A fire hydrant is spraying water onto a sidewalk |
| Beam (k=5) | A fire hydrant with water spraying out of it |

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

Table 11 shows that for image numbers 2, 3 and 9 greedy search generates more accurate captions that also have a higher level of detail. Whereas for image number 1 and 8, beam search generates higher quality captions. For image number 4, 5 and 6 the captions generated by both decoding strategies are the exact same and for the remaining images they are very similar. Hence in this case both decoding strategies perform equally well in terms of caption quality. It would have been a more informative comparison if both decoding 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 acted as 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 inspired 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 which presents the results of the initial runs conducted using the Flickr8k dataset to ensure that data was passing through the implemented architecture accurately and the models were being trained 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 only exposed to the true caption at the time of training and finds it challenging to recover from its own mistakes during inference. However, Table 3 also shows that training with teacher forcing displays higher BLEU scores. Figure 4 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.

While the results provided valuable insights about the performance with the two training strategies and training with teacher forcing was selected for further experiments, there were two limitations in the methods. Firstly, for a more robust comparison, both decoders have been trained till convergence in the case of non-teacher forcing as well. However, due to limited computational resources on the HPC node, training was stopped since jobs time out after 72 hours. 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 output 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 could address these limitations by leveraging more powerful computational resources to train for a longer period and by implementing key-value caching for the transformer.

### 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. The transformer decoder was able to perform slightly better because in addition to the cross-attention mechanism similar to the attention module in the LSTM, it also has masked self-attention which allows it to attend to all the previously generated words as well in parallel helping it build a richer textual context.

### 5.1.4. Analyze the performance of finetuning different layers of the ConvNeXt

The main objective of this study which was to finetune different layers of the ConvNeXt encoder and compare their performances. This objective was achieved as presented in section 4.3. Various experiments as shown in section 3.3.2 were conducted in which initially the ConvNeXt was frozen to get a baseline performance followed by finetuning 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 5 and it can be seen that finetuning 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 finetuning deeper layers results in similar performance as finetuning 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 finetune. Hence finetuning layers 1-7 has no additional benefit. Moreover, Figure 7 shows that finetuning 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, finetuning 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 finetuning.

Table 7 shows that finetuning the ConvNeXt also improves the quality of generated captions as they include more detail however, the level of detail does not improve by finetuning deeper layers. These insights are valuable and finetuning 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 in this study finetuning the entire architecture takes around an hour per epoch on average which amounts to 5 days for every 120 epochs. Hence for practical reasons, reasonable sequences which gave a fair idea of finetuning different depths of layers were selected and finetuning 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 a sample image 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 10 sample images. It was observed that for some images greedy search generated more accurate captions with higher details 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. On the other hand, 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 object to be fully achieved and can be explored in future works.

## 5.2. Answering the Research Question

As mentioned earlier, this study aimed to answer the 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?” Referring to section 5.1, it can be said that by achieving all the objectives, the research question 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 finetuning only the shallow layers 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.

Consider adding a table which compares my best model to other studies from the original paper

## 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. 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 parallel attention mechanisms makes it well-suited for the non-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 finetuning the ConvNeXt have substantial practical implications. They show that by finetuning a pre-trained vision model, superior performance can be achieved for vision-language tasks without having to train a vision model from scratch. While finetuning, it is sufficient to finetune only the middle to later layers instead of the entire model. Both of these insights save significant computational resources.

### 5.3.2. 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. Moreover, the findings that using pretrained word embeddings and beam search does not result in major improvements in performance can save 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 initial 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 computational time was a limitation. Attention maps were a part of the original project plan but gave valuable insights about how the models learn differently. The final 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 my 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 missing implementation 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 develop these models. I also gained an understanding of the complementary challenges of training large models such as developing 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 play 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, this study offers several proposals for future work such as further exploring 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 Vision Transformers to evaluate how they perform in similar conditions.

# Glossary

# 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. (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.
13. 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.
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. 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
21. 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)*.
22. 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
23. Cahyono, J. A. and Jusuf, J. N. (2024) ‘Automated Image Captioning with CNNs and Transformers’, *Computer Vision and Pattern Recognition*. arXiv
24. Atliha, V. and Sesok, D. (2021) ‘Pretrained word embeddings for image captioning’, *2021 IEEE Open Conference of Electrical, Electronic and Information Sciences (eStream)*
25. *Common objects in context* *COCO*. Microsoft. Available at: https://cocodataset.org/#home
26. 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)
27. 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)
28. 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)
29. Vinodababu, S. (2019). *A PyTorch Tutorial to Image Captioning*. [Online]. GitHub. Available at: <https://github.com/sgrvinod/a-PyTorch-Tutorial-to-Image-Captioning>
30. 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
31. 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)
32. PyTorch. *TransformerDecoderLayer - PyTorch 2.8 documentation*. Available at: https://docs.pytorch.org/docs/stable/generated/torch.nn.TransformerDecoderLayer.html
33. PyTorch. *TransformerDecoder - PyTorch 2.8 documentation*. Available at: <https://docs.pytorch.org/docs/stable/generated/torch.nn.TransformerDecoder.html>
34. PyTorch. *Pytorch - transformer.py*, *GitHub*. Available at: https://github.com/pytorch/pytorch/blob/v2.8.0/torch/nn/modules/transformer.py#L966
35. 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)
36. 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
37. 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
38. Diakogiannis, F. (2024) *Distributed training on Slurm Cluster*, *PyTorch Forums*. Available at: https://discuss.pytorch.org/t/distributed-training-on-slurm-cluster/150417/13
39. PyTorch. *DistributedDataParallel - PyTorch 2.8 documentation*. Available at: https://docs.pytorch.org/docs/stable/generated/torch.nn.parallel.DistributedDataParallel.html
40. 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
41. namespace-Pt (2021) *A Comprehensive Tutorial to Pytorch DistributedDataParallel*, *Medium*. Available at: https://medium.com/codex/a-comprehensive-tutorial-to-pytorch-distributeddataparallel-1f4b42bb1b51

# Appendices and additional files