Nanyang Technological University

College of Computing and Data Science

SC4001 NEURAL NETWORK & DEEP LEARNING

SC4001 Project Report

Automated Image Captioning with CNNs and Transformers

Name	Matric No.
Muhammad Sufyan Bin Mohd Jais	U2122751J
Lian Hong Shen Jordan	U2122011E
Irfaan Bin Ahmad	U2121388D

Table of Contents

1 Problem Description

The objective of this project is to develop a system that automatically generates meaningful and descriptive captioning for input images using deep learning approaches. Convolutional Neural Networks (CNNs) will be leveraged to extract the features of an image while Recurrent Neural Networks (RNNs), specifically Long Short-term Memory (LSTM) networks will be implemented for sequence modelling for text generation. Additionally, Transformer-based architectures will also be used as an alternative to LSTM to potentially improve the performance.

This project focuses on comparing on three architectures for image captioning:

1. CNN encoder with LSTM decoder
2. CNN encoder with standard Transformer decoder
3. CNN encoder with customised Transformer decoder

By comparing the three architectures, it aims to highlight the advantages as well as the challenges of each approach. It also addresses factors such as dataset selection, data preprocessing, model architecture design, hyperparameter optimisation and evaluation metric selection.

Pytorch was utilised for its extensive provision of tools and libraries for deep learning tasks pertaining to image captioning. Furthermore, by examining multiple approaches, we seek to provide insights to the performance of different architectures and identify the most effective strategies for automated image captioning.

2 Background and Literature Review

Image captioning represents a complex intersection of computer vision and natural language processing, requiring systems to not only recognise visual elements but also describe them in coherent natural language. This field has evolved tremendously over the years, advancing from simple rule-based systems to sophisticated deep learning models.

Traditional image captioning approaches were largely template-based or retrieval-based. Template-based methods worked by detecting objects within an image and generating captions by fitting the objects into a sentence template [1]. Meanwhile, retrieval-based methods compared an image with a large dataset of images that had existing captions, subsequently retrieving the caption of the most similar image from the dataset for the new image [2]. Although straightforward, these methods were not flexible and lacked the ability to generate new captions for unseen images. This limitation called for the need to develop more complex models that are capable of understanding visual content and generating new captions in a more coherent natural language.

The integration of deep learning approaches for image captioning offered more flexibility in terms of generating new captions on unseen images. The 'Show and Tell' model combined CNNs for feature extraction with LSTM networks for generating sequences, setting a new standard for accuracy and flexibility in image captioning [3].

Later on, attention mechanisms were incorporated into models such as the 'Show, Attend, and Tell' framework, allowing the system to focus on relevant parts of an image while generating each word. This led to more descriptive and contextually accurate captions, mimicking human-like patterns of visual attention [4].

Following these developments, transformer-based architectures emerged, using self-attention layers to capture dependencies across a sequence more effectively than RNNs or LSTM networks. These architectures are able to process information in parallel, resulting in faster training times and enhanced accuracy [5].

To assess the quality of generated captions, various evaluation metrics are commonly used, including Bilingual Evaluation Understudy (BLEU), Metric for Evaluation of Translation with Explicit Ordering (METEOR), and Consensus-based Image Description Evaluation (CIDEr). BLEU emphasises n-gram precision, METEOR focuses on recall and synonym matching, while CIDEr captures consensus across multiple references. These metrics complement each other, balancing considerations of linguistic accuracy, semantic similarity, and human judgement.

Overall, the introduction of deep learning models like CNN-RNN and CNN-Transformer has significantly advanced image captioning, enabling the generation of more descriptive and contextually rich captions for images, offering greater flexibility and accuracy.

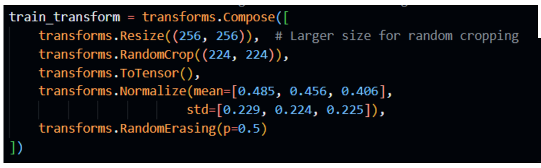
3 Methodology

3.1 Dataset Selection and Preprocessing

Microsoft Common Objects in Context (MS COCO) Dataset
The MS COCO dataset serves as the primary dataset for this project. We utilised the MS COCO 2017 with images in JPG format, which includes training images from the train2017 split with 118,000 images, validation images from the val2017 split with 5,000 images. Each image is paired with five different human-annotated captions, providing diverse linguistic descriptions, providing a rich variety of complex scenes with multiple objects and interactions [6].

Data Augmentation and Normalisation
We have implemented a comprehensive data preprocessing pipeline to enhance model robustness and training effectiveness. In general, we resize the image from MSCOCO to be larger for random cropping. We apply the following transforms according to the pytorch page "All pre-trained models expect input images normalised in the same way The images have to be loaded in to a range of [0, 1] and then normalised using mean = [0.485, 0.456, 0.406] and std = [0.229, 0.224, 0.225]" [7]. We then apply random erasing to improve model robustness and generalisation.

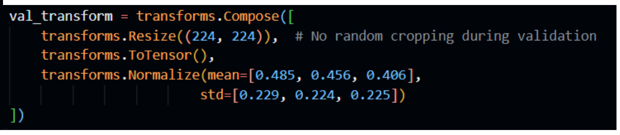
For training:



```
train_transform = transforms.Compose([
    transforms.Resize((256, 256)),  # Larger size for random cropping
    transforms.RandomCrop((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],
                         std=[0.229, 0.224, 0.225]),
    transforms.RandomErasing(p=0.5)
])
```

Fig. 1. Transforms applied on training images

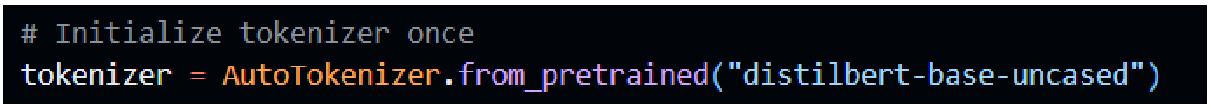
For validation and testing:



```
val_transform = transforms.Compose([
    transforms.Resize((224, 224)),  # No random cropping during validation
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],
                         std=[0.229, 0.224, 0.225])
])
```

Fig. 2. Transforms applied on validation/testing images

Tokenization
Text preprocessing and tokenization were handled using the HuggingFace's DistilBERT tokenizer where it is a distilled version of BERT which runs 60% faster while preserving over 95% of BERT's performances as measured on the GLUE language understanding benchmark [8]. It has special functions, having special tokens [CLS] for sequence start, [SEP] for sequence end, padding and truncation for fixed-length sequences and automatic handling of unknown tokens [8][9].



```python
# Initialize tokenizer once
tokenizer = AutoTokenizer.from_pretrained("distilbert-base-uncased")
```

Fig. 3. DistilBERT Tokenizer implementation

3.2 Evaluation Metrics

The evaluation of image captioning systems requires sophisticated metrics that can assess both the semantic correctness and linguistic quality of generated captions. We will be using Bilingual Evaluation Understudy (BLEU) and Consensus-based Image Description Evaluation (CIDEr) from pycocoevalcap [10].

BLEU is a statistical measure to assess the quality of the generated captions through two components, which are n-gram precision and brevity penalty [11]. N-gram precision evaluates the consecutive word sequences (n-grams) between generated and reference captions and we used four levels of n-grams (1 to 4). This prevents inflated score from repeated phrases by capping counts at a maximum reference occurrence (such as ‘a boy a boy a boy’ matching ‘a boy is flying a kite’ and combining score using geometric averaging. Brevity penalty penalises overly short captions such as “the” and is applied when the generated caption is shorter than reference caption and ensures the system can’t get high score by being overly conservative. The final BLEU score is calculated by multiplying the n-gram score with the n-gram score with our brevity penalty.

CIDEr is specifically designed for evaluating image descriptions and offers several advantages such as, using Term Frequency-Inverse Document Frequency (TF-IDF) weighting to emphasise important words, capturing consensus among multiple reference captions, penalising rarely used words that appear in generated captions and considering the entire caption dataset when computing similarity [12].

Our evaluation pipeline processes the generated captions through multiple steps. Firstly, for each image in the validation set, we collect all available reference captions from the COCO dataset. The model then generates the predicted caption for the image in the validation set. Both metrics are then computed against all reference captions and the scores are then aggregated across the validation set to produce the final metrics.

3.3 Model Architecture

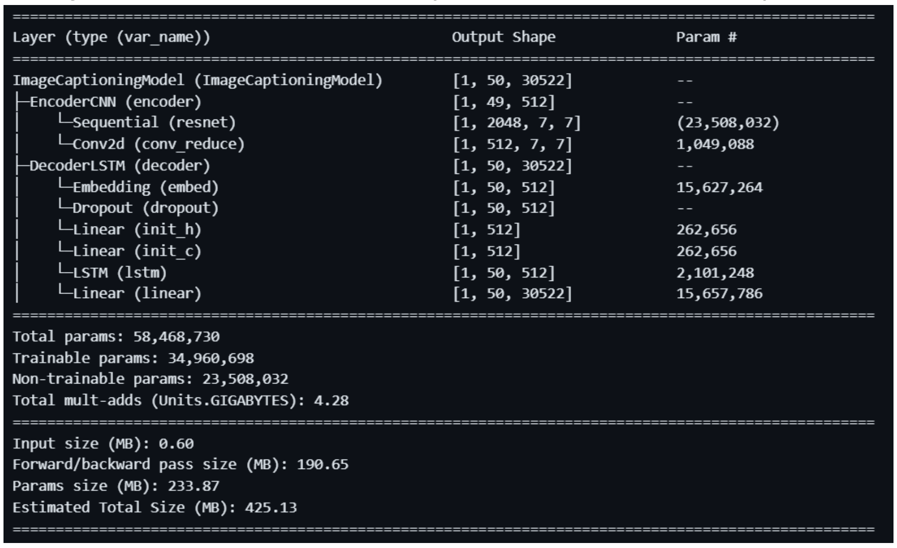
3.3.1 CNN Feature Extractor (ResNet50)

The image feature extraction is implemented using a CNN architecture, specifically ResNet50, which comprises 49 convolutional layers and 1 fully connected layer [13]. ResNet50 was selected for its optimal balance between feature extraction accuracy and computational efficiency. The implementation utilises the first 48 convolutional layers with pre-trained weights, which are frozen to optimise computational cost while maintaining performance. A point-wise convolution layer reduces dimensionality, further enhancing computational efficiency while preserving spatial information. This CNN encoder serves as the foundation for both LSTM and Transformer Decoder implementations.

3.3.2 Image Captioning with LSTM

The decoder component translates image features into natural language captions using Recurrent Neural Network (RNN) architecture. The Long Short-Term Memory (LSTM) variant was chosen for its ability to capture both long-term and short-term dependencies while mitigating gradient explosion issues [14]. The architecture processes both image features and captions through the LSTM Decoder to generate caption outputs.

The LSTM implementation incorporates word embeddings derived from the token vocabulary. This embedding process creates a vector space representation that enables the model to learn meaningful relationships between words in the captions. The training process consisted of 30 epochs, with model checkpoints saved every 5 epochs and at the lowest validation loss point. Caption generation employs a greedy search algorithm, chosen for its efficiency and effectiveness in word prediction.



```
==========================================================================================
Layer (type (var_name))                           Output Shape              Param #
==========================================================================================
ImageCaptioningModel (ImageCaptioningModel)       [1, 50, 30522]            --
├─EncoderCNN (encoder)                            [1, 49, 512]              --
│    └─Sequential (resnet)                        [1, 2048, 7, 7]           (23,508,032)
│    └─Conv2d (conv_reduce)                       [1, 512, 7, 7]            1,049,088
├─DecoderLSTM (decoder)                           [1, 50, 30522]            --
│    └─Embedding (embed)                          [1, 50, 512]              15,627,264
│    └─Dropout (dropout)                          [1, 50, 512]              --
│    └─Linear (init_h)                            [1, 512]                  262,656
│    └─Linear (init_c)                            [1, 512]                  262,656
│    └─LSTM (lstm)                                [1, 50, 512]              2,101,248
│    └─Linear (linear)                            [1, 50, 30522]            15,657,786
==========================================================================================
Total params: 58,468,730
Trainable params: 34,960,698
Non-trainable params: 23,508,032
Total mult-adds (Units.GIGABYTES): 4.28
==========================================================================================
Input size (MB): 0.60
Forward/backward pass size (MB): 190.65
Params size (MB): 233.87
Estimated Total Size (MB): 425.13
==========================================================================================
```

Fig. 4. Summary of LSTM Model

The model's architecture parameters include:

- Embed size: 512 dimensions
- Hidden size: 512 dimensions
- Dropout for improved generalisation

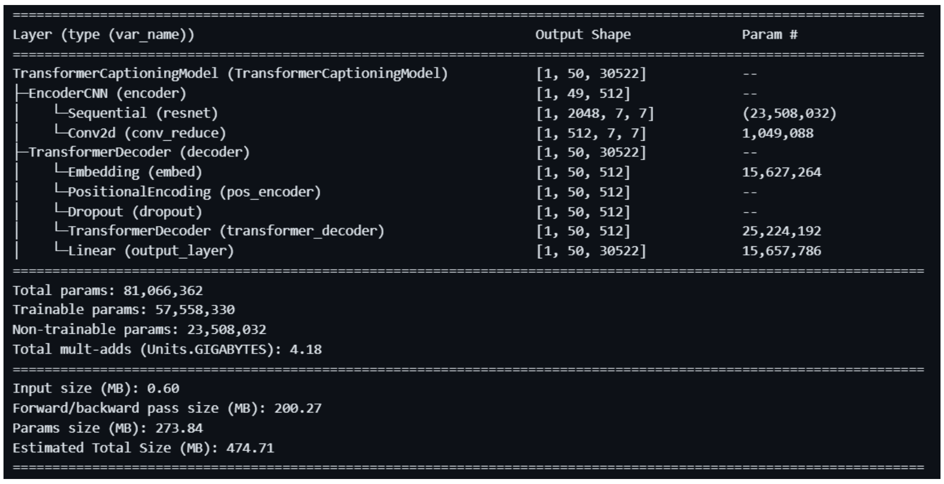
3.3.3 Image Captioning with Default Transformer

Transformer can also be used as the decoder component. The transformer-based decoder architecture utilises PyTorch's transformer modules. The transformer architecture employs positional embeddings and self-attention mechanisms to capture global context without relying on recurrent connections or memory structures.

The implementation is based on the architecture proposed in "Attention is All You Need", utilising PyTorch's nn.TransformerDecoderLayer and nn.TransformerDecoder classes [5]. The decoder consists of a stack of six standard decoder layers, each containing two types of attention mechanisms followed by a feed-forward neural network.

The input processing pipeline includes:

1. Embedding and Positional Encoding -> Traditional positional encoding using sinusoidal functions added to word embeddings to maintain sequence order.
2. Decoder Layer Architecture -> Each decoder layer contains dual attention mechanisms (self-attention for processing target sequence and cross-attention for encoder-decoder attention), followed by a feed-forward network for further feature transformation.
3. Skip Connections -> Residual connections implemented around each sub-layer (attention and feed-forward blocks) with layer normalisation to maintain gradient flow through the network.



```
==========================================================================================================
Layer (type (var_name))                                            Output Shape              Param #
==========================================================================================================
TransformerCaptioningModel (TransformerCaptioningModel)            [1, 50, 30522]            --
├─EncoderCNN (encoder)                                             [1, 49, 512]              --
│    └─Sequential (resnet)                                         [1, 2048, 7, 7]           (23,508,032)
│    └─Conv2d (conv_reduce)                                        [1, 512, 7, 7]            1,049,088
├─TransformerDecoder (decoder)                                     [1, 50, 30522]            --
│    └─Embedding (embed)                                           [1, 50, 512]              15,627,264
│    └─PositionalEncoding (pos_encoder)                            [1, 50, 512]              --
│    └─Dropout (dropout)                                           [1, 50, 512]              --
│    └─TransformerDecoder (transformer_decoder)                    [1, 50, 512]              25,224,192
│    └─Linear (output_layer)                                       [1, 50, 30522]            15,657,786
==========================================================================================================
Total params: 81,066,362
Trainable params: 57,558,330
Non-trainable params: 23,508,032
Total mult-adds (Units.GIGABYTES): 4.18
==========================================================================================================
Input size (MB): 0.60
Forward/backward pass size (MB): 200.27
Params size (MB): 273.84
Estimated Total Size (MB): 474.71
==========================================================================================================
```

Fig. 5. Summary of Default Transformer

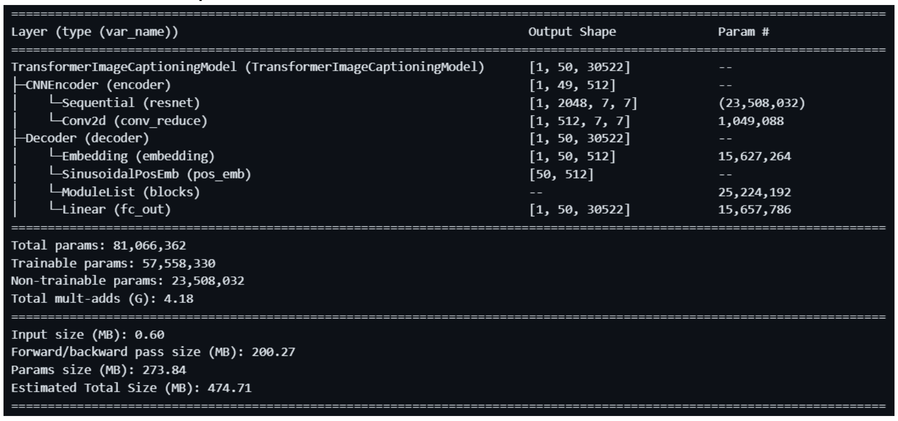
The model's architecture parameters include:

- Hidden size: 512 dimensions
- Feed-forward dimension: 2048 (4x hidden size)
- Number of decoder layers: 6
- Number of attention heads: 8
- Dropout regularisation (0.1) for improved generalisation

3.3.4 Image Captioning with Custom Transformer

To improve upon the default transformer implementation, we developed a custom transformer architecture that offers greater architectural flexibility and explicit control over each component. While maintaining the same basic principles of the transformer architecture, our custom implementation introduces several key modifications and modularity improvements:

1. Positional Encoding -> The custom transformer uses a more flexible way of handling variable sequence lengths using a dynamic sinusoidal positional embedding that computes on the fly as compared to the default implementation which precomputes a fixed sinusoidal positional encoding at initialisation
2. Attention Mechanism -> The custom transformer implements a modular attention block that explicitly separates self-attention and cross-attention components which provides direct control over masking operations.
3. Transformer Block Structure -> The custom implementation allows modification of individual components and insertion of additional layers such as feed-forward network with ELU activation, and no dropout within the blocks.



```
==========================================================================================================
Layer (type (var_name))                                              Output Shape            Param #
==========================================================================================================
TransformerImageCaptioningModel (TransformerImageCaptioningModel)    [1, 50, 30522]          --
├─CNNEncoder (encoder)                                               [1, 49, 512]            --
│    └─Sequential (resnet)                                           [1, 2048, 7, 7]         (23,508,032)
│    └─Conv2d (conv_reduce)                                          [1, 512, 7, 7]          1,049,088
├─Decoder (decoder)                                                  [1, 50, 30522]          --
│    └─Embedding (embedding)                                         [1, 50, 512]            15,627,264
│    └─SinusoidalPosEmb (pos_emb)                                    [50, 512]               --
│    └─ModuleList (blocks)                                           --                      25,224,192
│    └─Linear (fc_out)                                               [1, 50, 30522]          15,657,786
==========================================================================================================
Total params: 81,066,362
Trainable params: 57,558,330
Non-trainable params: 23,508,032
Total mult-adds (G): 4.18
==========================================================================================================
Input size (MB): 0.60
Forward/backward pass size (MB): 200.27
Params size (MB): 273.84
Estimated Total Size (MB): 474.71
==========================================================================================================
```

Fig. 6. Summary of Custom Transformer

The custom transformer architecture parameters remain consistent with the default implementation:

- Hidden size: 512 dimensions
- Feed-forward dimension: 2048 (4x hidden size)
- Number of decoder layers: 6
- Number of attention heads: 8

4 Experiments and Results

4.1 Image Captioning with LSTM

The model with LSTM was trained for 30 epochs and the loss and accuracy of the train and validation sets were plotted against the 30 epochs as shown in Fig. 7. We observed that while the model trained smoothly up until the 21st epoch, a dramatic spike in training loss occurred in the following epochs. This huge increase in spike is a phenomenon known as “exploding gradients”, a common issue in neural networks, especially when involving deep or long sequences like RNNs. Essentially, as the loss decreases and the gradients increase, the weights will update to a point where it becomes too large, leading to numerical instability. As a result, the training process would fail as the model would not converge to a good solution.

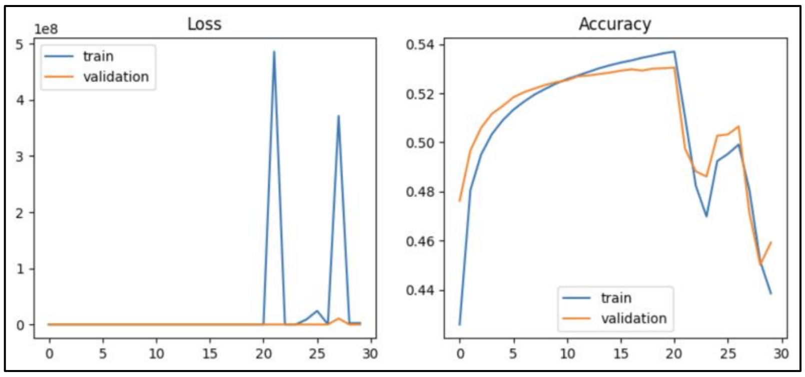


Fig. 7. Loss and accuracy for 30 epochs

Although not implemented in our model training, one of the ways to mitigate this issue is using a technique called ‘gradient clipping’. Gradient clipping involves setting a threshold value, whereby if the gradient exceeds the threshold value, the updating of weights will be scaled down, preventing them from growing too large. As a result, they will update at a more manageable rate, reducing the risk of numerical instability and improving convergence.

We then plotted the loss and accuracy of the train and validation sets for 21 epochs, before the model experienced the gradient explosion. As shown in Fig. 8, the training loss decreased steadily from approximately 3.1 to around 2.02, while the training accuracy increased from around 42% to about 53%. This implies that, up until the 21st epoch, the model was able to converge effectively. Hence, the 21st epoch was chosen as the best model.

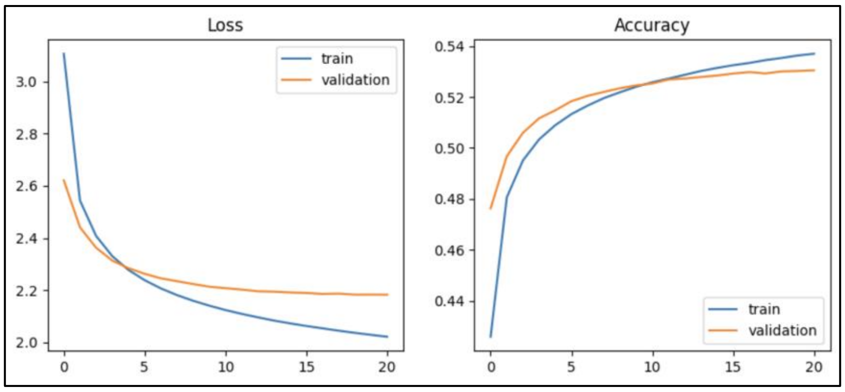
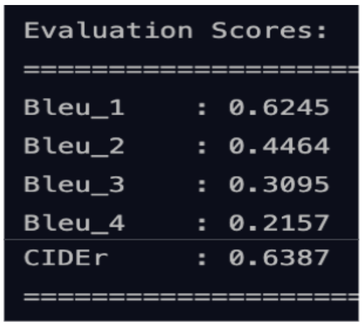


Fig. 8. Loss and accuracy for 21 epochs

The performance of the model was evaluated using metrics BLEU (BLEU-1 to BLEU-4) and CIDEr as demonstrated in Fig. 9, and it represents the average score across all the images in the validation dataset. A BLEU-1 score of 0.6245 suggests that the model does well in capturing single-word matches, identifying individual words accurately. However, the declining scores across longer n-grams (BLEU-2 to BLEU-4) means that it struggles with generating coherent short phrases. The moderate CIDEr score of 0.6387 indicates that while the captions are somewhat relevant, they lack depth in descriptive quality.



```
Evaluation Scores:
====================
Bleu_1     : 0.6245
Bleu_2     : 0.4464
Bleu_3     : 0.3095
Bleu_4     : 0.2157
CIDEr      : 0.6387
====================
```

Fig. 9. Average evaluation score of CNN-LSTM validation set

A few sample images were also evaluated as seen in Fig. 10. The left image achieved higher BLEU scores than the right image. The variability in BLEU scores may suggest that simpler images with distinct, recognisable objects may yield higher scores while more complex ones may result in lower scores. Another possible reason is that the image with the higher score is more frequently represented in the training set compared to the one with the lower score.

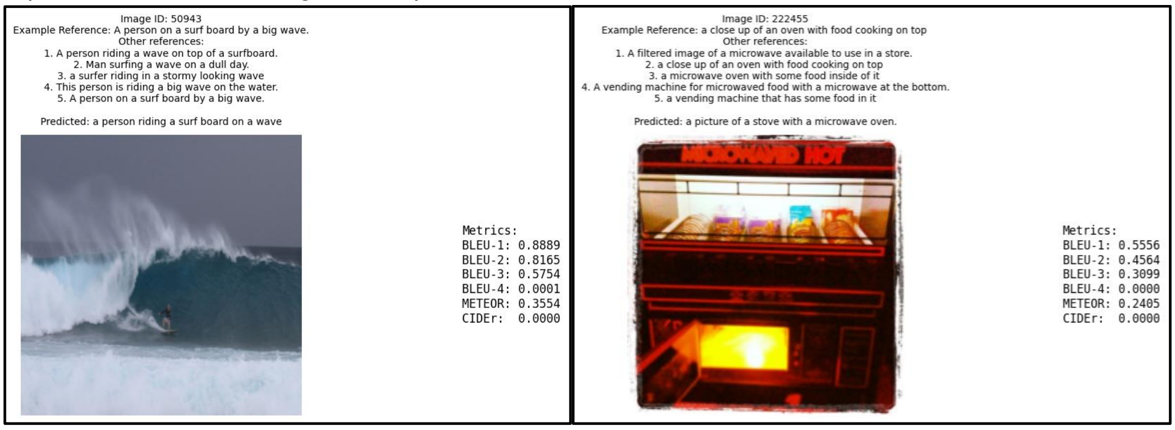


Image ID: 50943
Example Reference: A person on a surf board by a big wave.
Other references:
1. A person riding a wave on top of a surfboard.
2. Man surfing a wave on a dull day.
3. a surfer riding in a stormy looking wave
4. This person is riding a big wave on the water.
5. A person on a surf board by a big wave.

Predicted: a person riding a surf board on a wave

```
Metrics:
BLEU-1: 0.8889
BLEU-2: 0.8165
BLEU-3: 0.5754
BLEU-4: 0.0001
METEOR: 0.3554
CIDEr:  0.0000
```

Image ID: 222455
Example Reference: a close up of an oven with food cooking on top
Other references:
1. A filtered image of a microwave available to use in a store.
2. a close up of an oven with food cooking on top
3. a microwave oven with some food inside of it
4. A vending machine for microwaved food with a microwave at the bottom.
5. a vending machine that has some food in it

Predicted: a picture of a stove with a microwave oven.

```
Metrics:
BLEU-1: 0.5556
BLEU-2: 0.4564
BLEU-3: 0.3099
BLEU-4: 0.0000
METEOR: 0.2405
CIDEr:  0.0000
```

Fig. 10. Evaluation scores of sample images

4.2 Image Captioning with Default Transformer

We ran the model for 30 epochs and plot the loss and accuracy on a graph as seen in Fig. 11. The best validation loss was at the 14th epoch. We can see that the model starts to overfit from the 14th epoch where the training loss decreased while the validation loss increased from thereon.

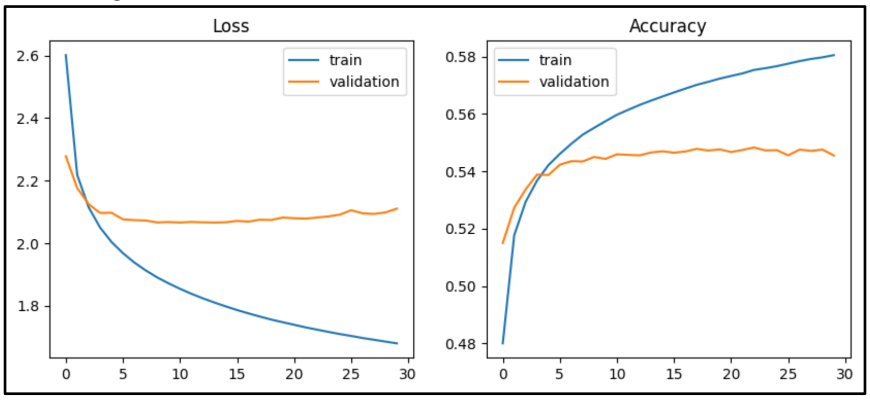
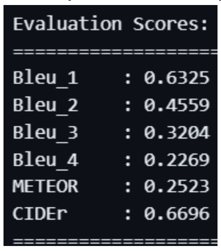


Fig.11. Loss and accuracy for 30 epoch of CNN+Default Transformer

Loading the model from the 14th epoch with the lowest validation loss as our best model, it was benchmarked against the same evaluation metrics used in the previous model. We note that it shows slightly improved performance compared to the CNN + LSTM model across all metrics. This suggests a more consistent performance across different types of images as higher CIDEr scores suggests a better capture of consensus among human reference caption and improvement in BLEU score shows better handling of longer phrase structures compared to the previous model.



```
Evaluation Scores:
==================
Bleu_1    : 0.6325
Bleu_2    : 0.4559
Bleu_3    : 0.3204
Bleu_4    : 0.2269
METEOR    : 0.2523
CIDEr     : 0.6696
==================
```

Fig.12. Average Evaluation score of CNN+Default Transformer

Below are some captions that we generated from passing a random image from the validation dataset into the model along with the reference captions for that image. We note that generated captions maintained semantic relevance however with an exception where the evaluation score for the image on the right seems to be lower. This might be due to the dynamic nature of the scene and multiple possible valid descriptions.

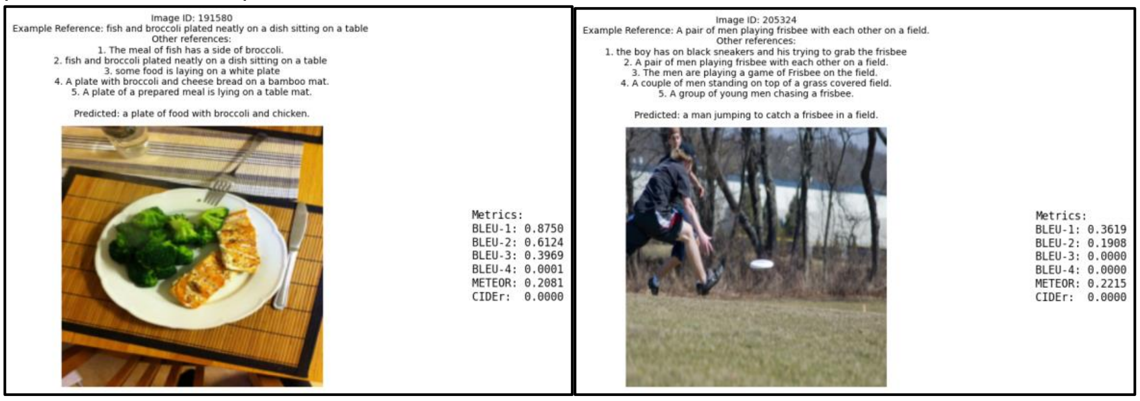


Image ID: 191580
Example Reference: fish and broccoli plated neatly on a dish sitting on a table
Other references:
1. The meal of fish has a side of broccoli.
2. fish and broccoli plated neatly on a dish sitting on a table
3. some food is laying on a white plate
4. A plate with broccoli and cheese bread on a bamboo mat.
5. A plate of a prepared meal is lying on a table mat.

Predicted: a plate of food with broccoli and chicken.

```
Metrics:
BLEU-1: 0.8750
BLEU-2: 0.6124
BLEU-3: 0.3969
BLEU-4: 0.0001
METEOR: 0.2081
CIDEr:  0.0000
```

Image ID: 205324
Example Reference: A pair of men playing frisbee with each other on a field.
Other references:
1. the boy has on black sneakers and his trying to grab the frisbee
2. A pair of men playing frisbee with each other on a field.
3. The men are playing a game of Frisbee on the field.
4. A couple of men standing on top of a grass covered field.
5. A group of young men chasing a frisbee.

Predicted: a man jumping to catch a frisbee in a field.

```
Metrics:
BLEU-1: 0.3619
BLEU-2: 0.1908
BLEU-3: 0.0000
BLEU-4: 0.0000
METEOR: 0.2215
CIDEr:  0.0000
```

Fig.13. Evaluation scores of sample images

4.3 Image Captioning with Custom Transformer

We conducted an additional 30 epochs of training using the custom transformer model and plotted the loss and accuracy as seen in Fig. 14.

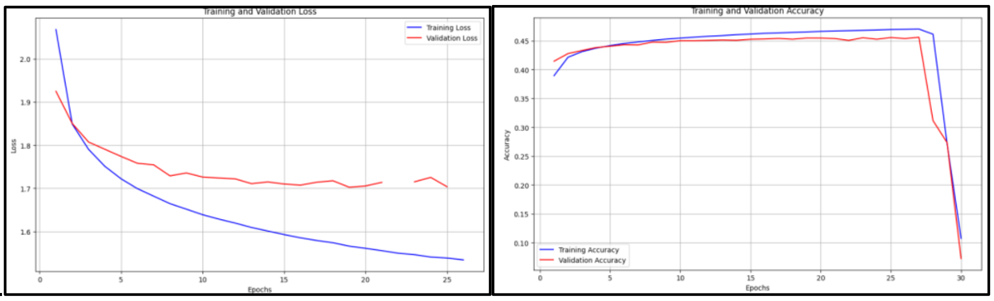
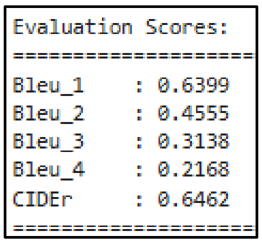


Fig.14. Loss and Accuracy for 30 epochs of CNN+Custom Transformer

The lowest validation loss occurred during the 19th epoch. After this point, there were indications of overfitting, as the training loss continued to decrease while the validation loss showed a slight increase. Additionally, some epochs resulted in NaN (Not a Number) values, which could be indicative of exploding gradients. Furthermore, there is a sudden drop in accuracy due to kernel timeout.

We loaded the model from the 19th epoch and benchmarked it against the same evaluation metrics used for the previous models. The results demonstrated a slight performance improvement over the CNN + LSTM model across all metrics, with minimal differences compared to the default transformer model. This suggests that the custom transformer and the default transformer have similar performance characteristics. However, the custom transformer offers greater flexibility and allows for easier parameter adjustments.



```
Evaluation Scores:
====================
Bleu_1     : 0.6399
Bleu_2     : 0.4555
Bleu_3     : 0.3138
Bleu_4     : 0.2168
CIDEr      : 0.6462
====================
```

Fig.15. Average evaluation score of CNN+Custom Transformer

A few samples were visualised at random, as shown in Figure 16 below. Both examples achieved relatively high metric scores, indicating that the model maintained semantic relevance in its outputs.

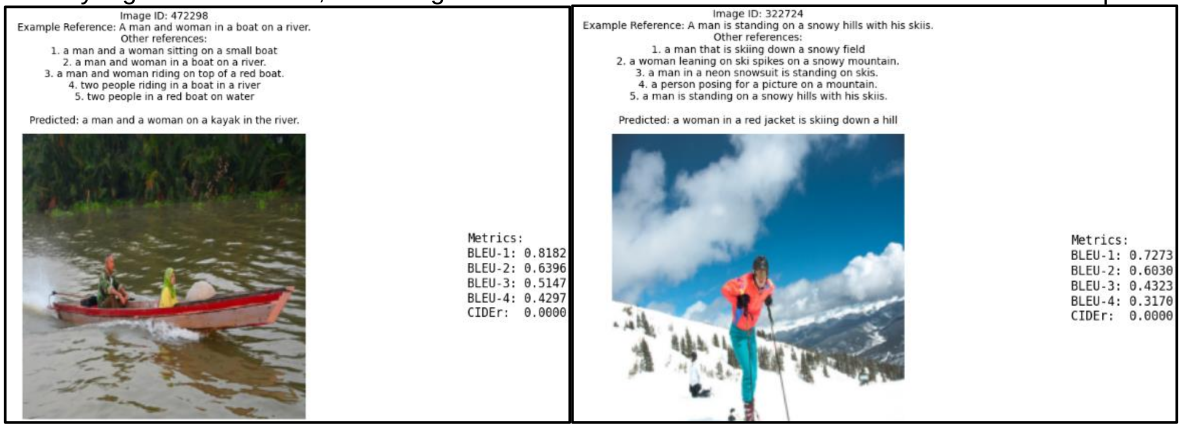


Image ID: 472298
Example Reference: A man and woman in a boat on a river.
Other references:
1. a man and a woman sitting on a small boat
2. a man and woman in a boat on a river.
3. a man and woman riding on top of a red boat.
4. two people riding in a boat in a river
5. two people in a red boat on water

Predicted: a man and a woman on a kayak in the river.

```
Metrics:
BLEU-1: 0.8182
BLEU-2: 0.6396
BLEU-3: 0.5147
BLEU-4: 0.4297
CIDEr:  0.0000
```

Image ID: 322724
Example Reference: A man is standing on a snowy hills with his skiis.
Other references:
1. a man that is skiing down a snowy field
2. a woman leaning on ski spikes on a snowy mountain.
3. a man in a neon snowsuit is standing on skis.
4. a person posing for a picture on a mountain.
5. a man is standing on a snowy hills with his skiis.

Predicted: a woman in a red jacket is skiing down a hill

```
Metrics:
BLEU-1: 0.7273
BLEU-2: 0.6030
BLEU-3: 0.4323
BLEU-4: 0.3170
CIDEr:  0.0000
```

Fig.16. Evaluation scores of sample images for Custom Transformer

5 Conclusion

This project explored and compared the performance of three models for image captioning, each with different architecture: LSTM, Default Transformer and Custom Transformer.

The LSTM model performed quite well in capturing single-word matches based on the good BLEU-1 scores. However, its performance in generating longer and more contextually rich captions tends to be limited, as seen in the longer n-gram BLEU and CIDEr scores. Furthermore, its performance may be affected by the exploding gradient problem, especially when involving longer sequences.

The Default Transformer performed similarly to the LSTM, with slight improvements to the loss and accuracy, as well as the BLEU and CIDEr scores. The increase in accuracy is likely due to its self-attention mechanism, which allows it to better capture relationships within the input sequence. Additionally, the Default Transformer also converged faster at 14 epochs as compared to 21 epochs for the LSTM. This faster convergence is likely attributed to its parallel processing capabilities, allowing it to process tokens simultaneously instead of sequentially.

The Custom Transformer exhibited comparable performance to the default transformer across all metrics. This outcome suggests that the current approach to model the Custom Transformer might have mirrored the Default Transformer too closely, leading to similar results. While the custom architecture provides greater flexibility and more granular parameter tuning, leveraging this advantage effectively may require further experimentation. Adjustments such as deeper hyperparameter optimization or integration of advanced regularisation techniques could differentiate its performance more significantly from the default transformer.

Future works could focus on several key areas for improvement where gradient clipping could address the gradient explosion issues observed, custom transformer architecture presents opportunities for further optimisation through advanced regularisation techniques and refined parameter tuning and visual attention mechanisms could enhance the feature extraction capabilities across all models.

Overall, this demonstrates that while transformer-based architectures offer certain advantages over LSTM in terms of training efficiency and stability, the choice of architecture should boil down to specific use case requirements, computational resources and the need for architectural flexibility.

References

[1] A. Farhadi *et al.*, ‘Every Picture Tells a Story: Generating Sentences from Images’, in *Computer Vision – ECCV 2010*, K. Daniilidis, P. Maragos, and N. Paragios, Eds., Berlin, Heidelberg: Springer, 2010, pp. 15–29. doi: 10.1007/978-3-642-15561-1_2.

[2] V. Ordonez, G. Kulkarni, and T. Berg, ‘Im2Text: Describing Images Using 1 Million Captioned Photographs’, in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2011. Accessed: Nov. 13, 2024. [Online]. Available: https://papers.nips.cc/paper_files/paper/2011/hash/5dd9db5e033da9c6fb5ba83c7a7ebea9-Abstract.html

[3] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, ‘Show and tell: A neural image caption generator’, in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015, pp. 3156–3164. doi: 10.1109/CVPR.2015.7298935.

[4] K. Xu *et al.*, ‘Show, Attend and Tell: Neural Image Caption Generation with Visual Attention’, Apr. 19, 2016, *arXiv*: arXiv:1502.03044. doi: 10.48550/arXiv.1502.03044.

[5] A. Vaswani *et al.*, ‘Attention Is All You Need’, Aug. 02, 2023, *arXiv*: arXiv:1706.03762. doi: 10.48550/arXiv.1706.03762.

[6] ‘COCO - Common Objects in Context’. Accessed: Nov. 13, 2024. [Online]. Available: https://cocodataset.org/#download

[7] ‘torchvision.models — Torchvision 0.8.1 documentation’. Accessed: Nov. 13, 2024. [Online]. Available: https://pytorch.org/vision/0.8/models.html

[8] ‘DistilBERT’. Accessed: Nov. 13, 2024. [Online]. Available: https://huggingface.co/docs/transformers/en/model_doc/distilbert

[9] ‘Padding and truncation’. Accessed: Nov. 13, 2024. [Online]. Available: https://huggingface.co/docs/transformers/en/pad_truncation

[10] salaniz, *salaniz/pycocoevalcap*. (Nov. 06, 2024). Python. Accessed: Nov. 15, 2024. [Online]. Available: https://github.com/salaniz/pycocoevalcap

[11] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, ‘Bleu: a Method for Automatic Evaluation of Machine Translation’, in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, P. Isabelle, E. Charniak, and D. Lin, Eds., Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, Jul. 2002, pp. 311–318. doi: 10.3115/1073083.1073135.

[12] R. Vedantam, C. L. Zitnick, and D. Parikh, ‘CIDEr: Consensus-based Image Description Evaluation’, Jun. 03, 2015, *arXiv*: arXiv:1411.5726. doi: 10.48550/arXiv.1411.5726.

[13] A. Oğuz and Ö. F. Ertuğrul, “Introduction to deep learning and diagnosis in medicine,” Diagnostic Biomedical Signal and Image Processing Applications with Deep Learning Methods. Elsevier, pp. 1–40, 2023. doi: 10.1016/b978-0-323-96129-5.00003-2. https://doi.org/10.1016/B978-0-323-96129-5.00003-2

[14] K. Liu and J. Zhang, “A Dual-Layer Attention-Based LSTM Network for Fed-batch Fermentation Process Modelling,” Computer Aided Chemical Engineering. Elsevier, pp. 541–547, 2021. doi: 10.1016/b978-0-323-88506-5.50086-3. https://doi.org/10.1016/B978-0-323-88506-5.50086-3