

# Image Captioning: A comparative study of deep learning techniques

Rahul Chinthala  
New York University  
rc4080@nyu.edu

Hemanth Teja Y  
New York University  
hy1713@nyu.edu

## 1. Problem Statement

Image captioning, defined as the automatic description of the content of an image, is a fundamental problem in artificial intelligence that lies at the intersection of computer vision and natural language processing. It requires recognizing the important objects, their attributes, and their relationships in an image, and also generating syntactically and semantically correct sentences that describe the image. Deep learning-based techniques, which are known to work well for a wide array of complex tasks, have been used for Image Captioning as well. Here, we aim to present a comparative study of the performance of two deep learning-based image captioning techniques on the benchmark MS-COCO dataset. We, thus, aim to compare these models using Natural Language Processing performance metrics.

## 2. Literature Survey

In this section we provide relevant background on previous work on image caption generation and attention. Much research effort has been devoted to automatic image captioning, and it can be categorized into template-based image captioning, retrieval-based image captioning, and novel image caption generation [5].

Template-based image captioning first detects the objects/attributes/actions and then fills the blanks slots in a fixed template [1]. Retrieval-based approaches first find the visually similar images with their captions from the training dataset, and then the image caption is selected from similar images with captions [6]. These methods are able to generate syntactically correct captions but are unable to generate image-specific and semantically correct captions. Both these approaches are typically involved an intermediate “generalization” step to remove the specifics of a caption that are only relevant to the retrieved image, such as the name of a city. Both of these approaches have since fallen out of favour to the now dominant neural network methods.

Differently, the novel image caption generation approaches first analyze the visual content of the image and then generate image captions from the visual content using a language model [7]. The visual representation comes from a convolutional neural network which is often pretrained for image classification on large-scale datasets [18]. Translation is accomplished through recurrent neural networks based language models. The main advantage of this approach is that the entire system can be trained from end to end, i.e., all the parameters can be learned from data.

The first approach to using neural networks for caption generation was proposed by Kiros et al. (2014a) who used a multimodal log-bilinear model that was biased by features from the image. This work was later followed by Kiros et al. (2014b). Mao et al. (2014) who used a similar approach to generation but replaced a feedforward neural language model with a recurrent one. Both Vinyals et al. (2014) and Donahue et al. (2014) used recurrent neural networks (RNN) based on long short-term memory (LSTM) units (Hochreiter Schmidhuber, 1997) for their models.

The problem with these methods is that, when the model is trying to generate the next word of the caption, this word is usually describing only a part of the image. It is unable to capture the essence of the entire input image. Using the whole representation of the image to condition the generation of each word cannot efficiently produce different words for different parts of the image. This is exactly where an Attention mechanism is helpful.

### **3. Description of Dataset**

MS-COCO (Common objects in context) dataset is used in this project. The dataset is commonly used to train and benchmark object detection, segmentation, and captioning algorithms. Some of the features of the dataset are: - 80 object categories, 91 stuff categories, 330K images and 5 captions per image. Since the dataset has fewer categories but more instances per category, we have opted to use it for training the model.

### **4. Data Preprocessing**

We tokenized the captions and performed some basic cleaning like lower-casing all the words, removing special tokens etc. Furthermore, we also created a vocabulary of all the unique words present across all the image captions ( $8000 \times 5 = 40,000$  data points) in the data set, leaving behind 8267 unique words. We then selected the top 5000/7000 words from the vocabulary as this helps the model become more robust to outliers and make less mistakes. According to PyTorch and Keras documentation, pre-trained Inception V3 and ResNet50 models expect that input is as 3-channel RGB images of shape  $(3 \times H \times W)$ , where H and W are expected to be at least 224. Therefore, data preprocessing involves loading and resizing images. The tokens and images are converted to embeddings of size 256 that can be fed to the decoder.

### **5. Description of Models Implemented**

#### **5.1. Vanilla RNN Image Captioner**

Here, we use CNN and RNN to generate image captions with CNN acting as the encoder and RNN as the decoder.

##### **5.1.1. Encoder-CNN**

In order to generate a description, we feed a particular image into a pre-trained CNN architecture often called the encoder because it encodes the content of the image into a smaller feature vector. CNN, as we know, scans images from left to right and top to bottom to pull out important features from the image and combines the feature to classify images. At the end of this CNN network is

a softmax classifier that outputs a vector of class scores, but we don't want to classify an image, instead we want a set of features that represents the spatial content in the image. To get that kind of spatial content, we're going to remove the final fully connected layer that classifies the image and look at it's earlier layer that distills the spatial information in the image. This way, we're using the CNN as a feature extractor that compresses the huge amount of extraction contained in the original image into a smaller representation.

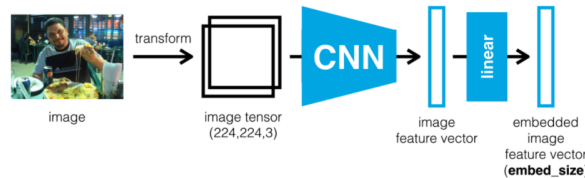


Figure 1. Architecture of the CNN Encoder

### 5.1.2. Decoder-CNN

The job of the RNN is to decode the process vector output by the CNN and turn it into a sequence of words. Thus, this portion of the network is often called a decoder.

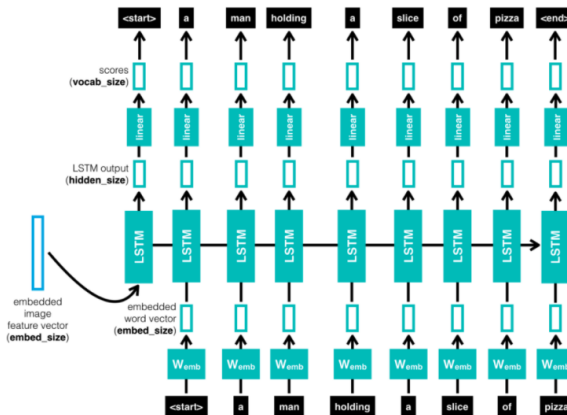


Figure 2. Architecture of the CNN Decoder

**Our model consists of 3 main phases:**

**A. Image Feature Extraction:** The features of the images from the MS Coco dataset are extracted using the ResNet50 model. We obtain a 2048 vector element representation of the photo and pass it on to the LSTM layer.

**B. Sequence processor:** The function of a sequence processor is for handling the text input by acting as a word embedding layer. The embedded layer consists of rules to extract the required features of the text and consists of a mask to ignore padded values. The network is then connected to a LSTM for the final phase of the image captioning.

**C. Decoder:** The final phase of the model combines the input from the Image extractor phase and the sequence processor phase using an additional operation then fed to a 256 neuron layer and then to a final output Dense layer that produces a softmax prediction of the next word in the caption over the entire vocabulary which was formed from the text data that was processed in the sequence processor phase.

### 5.1.3. Loss Function

The nature of our RNN output is a series likelihood of words' occurrences, and in order to quantify the quality of the RNN output, we propose to use Cross Entropy Loss. This is the most popular and effective measurement for the performance of a classification model whose output is a probability value between 0 and 1.

$$Loss = - \sum_n^N t_k^n \ln p_k^n$$

Here, N is the number of classes , the vocab size in this context, t is either 0 or 1, and  $p_k^n$  is the predicted possibility that observation k is of class n

## 5.2. Attention-Based Image Captioning

In this technique, the image is first divided into n parts, and we compute Convolutional Neural Network (CNN) representations of each part  $h_1, \dots, h_n$ . When the RNN is generating a new word, the attention allows it to focus on the part of the image most relevant to the word it is going to utter next. The model learns where to look , and As we generate a caption, word by word, we can see the model's gaze shifting across the image.

The Encoder encodes the input image with 3 color channels into a smaller image with "learned" channels. This smaller encoded image is a summary representation of all that's useful in the original image. We don't train an encoder from scratch , but instead use pre-trained models ,trained on millions of images ,which capture the essence of an image very well. The Decoder looks at the encoded image and generates a caption word by word. Instead of the simple average, we use the weighted average across all pixels, with the weights of the important pixels being greater. This weighted representation of the image is concatenated with the previously generated word at each step to generate the next word. It's the Attention network that computes these weights.

We will use soft Attention, where the weights of the pixels add up to 1. If there are P pixels in our encoded image, then at each timestep t –

$$\sum_t^T \alpha_{p,t} \approx 1$$

### 5.2.1. Bahdanau Attention

In this project , We have used Bahdanau Attention, which is also known as Additive attention as it performs a linear combination of encoder states and the decoder states. In the Bahdanau Attention mechanism all the encoder hidden states, along with the decoder hidden state are used to generate the Context vector.

We have to calculate the Alignment scores between the previous decoder hidden state and each of the encoder's hidden states. The alignment scores for each encoder hidden state are combined

and represented in a single vector and then softmax-ed. The alignment vector is a vector that has the same length as the source sequence. Each of its values is the score (or the probability) of the corresponding word within the source sequence. Alignment vectors put weights on the encoder's output. With those weights, the Decoder decides what to focus on at each time step. The encoder hidden states and their respective alignment scores (attention weights in the above equation) are multiplied to form the context vector. The context vector is used to compute the final output of the decoder.

**Bahdanau's Additive Style:**  $score(h_t, h'_s) = v_a^T \tanh(W_1 h_t + W_2 h'_s)$

**Attention Weights:**  $\alpha_{ts} = \frac{\exp(score(h_t, h'_s))}{\sum_{s'=1}^S \exp(score(h_t, h'_{s'}))}$

**Context Vector:**  $c_t = \sum_s \alpha_{ts} h'_s$

**Attention Vector:**  $a_t = f(c_t, h_t) = \tanh(W_c[c_t; h_t])$

### 5.2.2. Loss Function

Since we're generating a sequence of words, we use CrossEntropyLoss. We only need to submit the raw scores from the final layer in the Decoder, and the loss function will perform the softmax and log operations. The model is trained end-to-end by minimizing the following penalized negative log-likelihood:

$$Loss = - \sum_n^N t_k^n \ln p_k^n$$

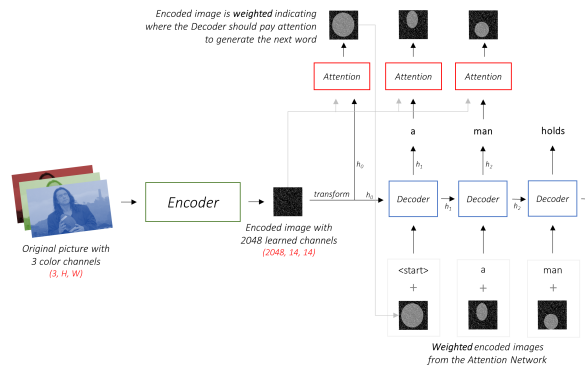


Figure 3. Architecture of the attention based model

### 5.2.3. Training Details

**ResNet-50:** It is a convolutional neural network that is 50 layers deep, 48 Convolution layers along with 1 MaxPool and 1 Average Pool layer. It has  $3.8 \times 10^9$  Floating points operations. You can load a pretrained version of the network trained on more than a million images from the ImageNet database [1]. The pretrained network can classify images into 1000 object categories, such

as keyboard, mouse, pencil, and many animals.

**Inception V3:** Inception v3 mainly is a convolutional neural network that is 48 layers deep, which focuses on burning less computational power by modifying the previous Inception architectures.

**Teacher Forcing:** We have used Teacher forcing instead of feeding the output of a time step at the next time-step as it helps quickly and efficiently train recurrent neural network models that uses the ground truth from a prior time step as input. It is method critical to the development of deep learning language models used in machine translation, text summarization, and image captioning, among many other applications. It helps handle Slow convergence and improves Model instability.

**Vocab Size:** We have varied the vocab size( Top 5000, Top 7000 as , Ideally, we want a vocabulary that is both expressive and as small as possible. A smaller vocabulary will result in a smaller model that will train faster. W

**Learning Rate Size:** There are several hyperparamters we adjusted in our experiments. Initially, in the training section, we used learning rate of  $4 \times 10^4$ . However, under this condition, the oscillation of training loss is very significant.

**Batch Size:** We also adjusted the batch size in our experiments. Initially, we used a batch size of 16. However, the loss didn't converge so we changed the batch size to 64, as we were constrained by the the 12gb maximum GPU limit offering by Colab.

**Weight Initialization:** We have used Glorot/Xavier initialization for the weights a it keeps gradients, Z-values and Activations similar along all the layers and also, the Sigmoid activation function poses a problem for this, as the activation values will have a mean of 0.5, not zero!

## 6. Evaluation Metrics

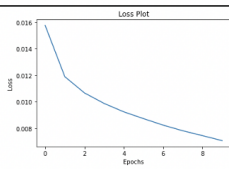
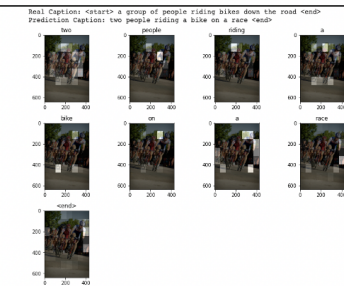
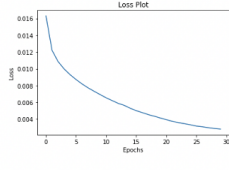
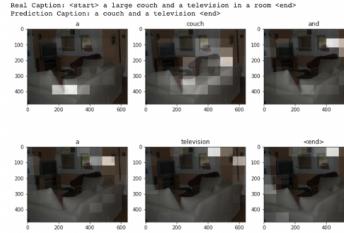
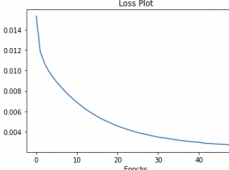

BLEU-1, BLEU-2, BLEU-3, and BLEU-4 (Bilingual Evaluation Understudy) is the most commonly reported metrics in evaluating the quality of text generation in natural language processing tasks. Here we chose 4-gram BLEU score(BLEU-4) as our primary evaluation metric. Furthermore, we also use METEOR score as a metric. The metric is based on the harmonic mean of unigram precision and recall, with recall weighted higher than precision.

Additionally, We also use GLEU (for sentence level fluency) and WER(Word Error Rate) - which are common metrics of the performance of a speech recognition or machine translation system.

## 7. Results

### 7.1. Model: Bahdanau Attention + Inception V3

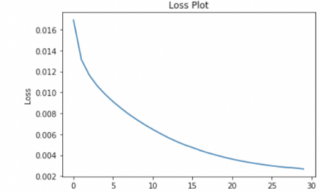
Vocab	Total Images	Batch Size	EPOCHS	BLEU 1	BLEU 2	BLEU 3	BLEU 4	GLEU (1 - 4 grams)	GLEU (1-2 grams)	METEOR
Vocab = 5000 words Captions/image = 5	6000	64	10	11.1	33.3	48.4	57.7	13.3	23.5	35.4
Vocab = 7000 words Captions/image = 5	6000	64	30	16.6	40.8	55.3	63.8	36.6	47.0	43.2
Vocab = 7000 words Captions/image = 5	10000	64	50	6.2	25.0	40.0	50.0	15.5	29.0	54.5

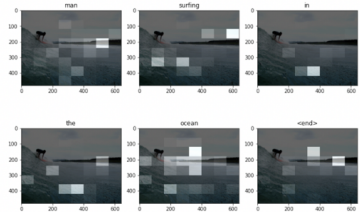
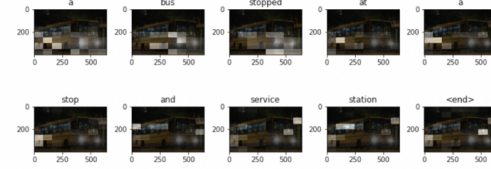
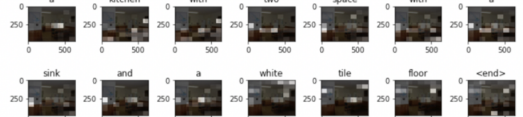
Configuration	Loss Curve	WER Matrix	Predictions
Vocab = 5000 words Total Images = 6000 Captions/image = 5 Epochs = 10 Batch Size = 64		WER matrix (9x10): [[0. 1. 2. 3. 4. 5. 6. 7. 8. 9.] [1. 1. 2. 3. 4. 5. 6. 7. 8. 9.] [2. 2. 2. 3. 4. 5. 6. 7. 8. 9.] [3. 3. 3. 4. 5. 6. 7. 8. 9.] [4. 4. 4. 4. 5. 6. 7. 8. 9.] [5. 5. 5. 5. 5. 6. 7. 8. 9.] [6. 6. 6. 6. 6. 6. 7. 8. 9.] [7. 7. 7. 7. 7. 7. 8. 9.] [8. 8. 8. 8. 8. 8. 8. 9. 9.]	Real Caption: <start> a group of people riding bikes down the road <end> Prediction Caption: two people riding a bike on a race <end> 
Vocab = 7000 words Total Images = 6000 Captions/image = 5 Epochs = 30 Batch Size = 64		WER matrix (6x9): [[0. 1. 2. 3. 4. 5. 6. 7. 8.] [1. 1. 1. 2. 3. 4. 5. 6. 7.] [2. 2. 2. 1. 2. 3. 4. 5. 6.] [3. 3. 3. 2. 1. 2. 3. 4. 5.] [4. 4. 4. 3. 2. 1. 2. 3. 4.] [5. 5. 5. 4. 3. 2. 2. 3. 4.]	Real Caption: <start> a large couch and a television in a room <end> Prediction Caption: a couch and a television <end> 
Vocab = 7000 words Total Images = 10,000 Captions/image = 5 Epochs = 30 Batch Size = 64		WER matrix (16x10): [[0. 1. 2. 3. 4. 5. 6. 7. 8. 9.] [1. 1. 2. 3. 4. 5. 6. 7. 8. 9.] [2. 2. 2. 3. 4. 5. 6. 7. 8. 9.] [3. 3. 3. 4. 5. 6. 7. 8. 9.] [4. 4. 4. 4. 4. 5. 6. 7. 8. 9.] [5. 5. 5. 5. 5. 6. 7. 8. 9.] [6. 6. 6. 6. 6. 6. 7. 8. 9.] [7. 7. 7. 7. 7. 7. 8. 9.] [8. 8. 7. 7. 8. 7. 8. 8. 8. 9.] [9. 9. 8. 8. 8. 8. 7. 8. 9. 9.] [10. 10. 9. 9. 9. 9. 8. 8. 9. 10.] [11. 11. 10. 10. 10. 10. 9. 9. 10.] [12. 12. 11. 11. 11. 11. 10. 9. 9.] [13. 13. 12. 12. 12. 11. 10. 9. 9.] [14. 14. 13. 13. 13. 13. 12. 11. 10. 9.] [15. 15. 14. 14. 14. 14. 13. 12. 11. 10.]	Real Caption: <start> a greasy looking piece of pizza is on a plate <end> Prediction Caption: a couch bun topped with a tasty looking pizza is sitting on a white plate <end> 

## INFERENCE

It can be seen that the METEOR score increases as we increase the vocabulary size and epochs. That is when the model is trained with higher configuration parameters and data. Moreover , the configuration with vocabulary size 7000 , epochs 30 and 6000 images has the best BLEU - 1,2,3,4 and GLEU scores. The scores decreased for the configuration vocabulary size 7000 epochs 50 and data 10000. This could possibly be due to over fitting of the model.

## 7.2. Model: Bahdanau Attention + ResNet50

Vocab	Total Images	Batch Size	EPOCHS	Captions/image	Training: Loss Curve
7000 words	6000	64	30	5	

Prediction Quality	METRICS		WER Matrix	Predictions
GOOD	BLEU 1	11.1	WER matrix (9x10): [[0. 1. 2. 3. 4. 5. 6. 7. 8. 9.] [1. 1. 2. 3. 4. 5. 6. 7. 8. 9.] [2. 2. 2. 3. 4. 5. 6. 7. 8.] [3. 3. 3. 3. 3. 4. 5. 6. 7. 8.] [4. 4. 4. 4. 3. 4. 5. 6. 7. 8.] [5. 5. 5. 5. 4. 3. 4. 5. 6. 7.] [6. 6. 6. 6. 5. 4. 4. 5. 6. 7.] [7. 7. 7. 7. 6. 5. 5. 5. 6. 7.] [8. 8. 8. 8. 7. 6. 6. 6. 6. 7.]] 7	Real Caption: <start> the surfer is riding on a wave in the ocean <end> Prediction Caption: man surfing in the ocean <end> 
	BLEU 2	33.3		
	BLEU 3	48.4		
	BLEU 4	57.7		
	GLEU (1 to 4 grams)	11.6		
	GLEU (1 to 2 grams)	21.0		
	METEOR	30.9		
FAIR	BLEU 1	10	WER matrix (10x10): [[0. 1. 2. 3. 4. 5. 6. 7. 8. 9.] [1. 1. 1. 2. 3. 4. 5. 6. 7. 8.] [2. 2. 2. 2. 3. 4. 5. 6. 7. 8.] [3. 3. 3. 3. 2. 3. 4. 5. 6. 7.] [4. 4. 4. 4. 3. 2. 3. 4. 5. 6.] [5. 5. 5. 5. 4. 3. 2. 3. 4. 5.] [6. 6. 6. 6. 5. 4. 3. 3. 4. 5.] [7. 7. 7. 7. 6. 5. 4. 4. 4. 5.] [8. 8. 8. 8. 7. 6. 5. 5. 5. 5.] [9. 9. 9. 9. 8. 7. 6. 6. 6. 6.]] 6	Real Caption: <start> a commuter bus parked at a stop at night time <end> Prediction Caption: a bus stopped at a stop and service station <end> 
	BLEU 2	31.6		
	BLEU 3	46.7		
	BLEU 4	56.2		
	GLEU (1 to 4 grams)	23.5		
	GLEU (1 to 2 grams)	36.8		
	METEOR	37.5		
BAD	BLEU 1	7.1	WER matrix (14x8): [[0. 1. 2. 3. 4. 5. 6. 7.] [1. 0. 1. 2. 3. 4. 5. 6.] [2. 1. 0. 1. 2. 3. 4. 5.] [3. 2. 1. 1. 2. 3. 4. 5.] [4. 3. 2. 2. 2. 3. 4. 5.] [5. 4. 3. 3. 3. 3. 4. 5.] [6. 5. 4. 3. 4. 4. 4. 5.] [7. 6. 5. 4. 4. 5. 5. 5.] [8. 7. 6. 5. 5. 5. 6. 6.] [9. 8. 7. 6. 6. 6. 6. 7.] [10. 9. 8. 7. 7. 7. 7. 7.] [11. 10. 9. 8. 8. 8. 8. 8.] [12. 11. 10. 9. 9. 9. 9. 9.] [13. 12. 11. 10. 10. 10. 10.]] 10	Real Caption: <start> a kitchen with a map on the wall <end> Prediction Caption: a kitchen with two space with a sink and a white tile floor <end> 
	BLEU 2	26.7		
	BLEU 3	41.8		
	BLEU 4	51.6		
	GLEU (1 to 4 grams)	16		
	GLEU (1 to 2 grams)	25.9		
	METEOR	23.5		

View: [Detailed Results](#)

## INFERENCE

We have observed that in some cases the METEOR Scores haven't corroborated well with the human judgements. For example , in the table above while the human judgement rated a picture as being good , the METEOR score for the same ( 30.9 ) was lower than that of the human rated Fair prediction (37.5). A possible explanation for the same is that the model picked up on the many nuances in the image, and the corresponding words while neglecting the semantic order. As METEOR score primarily emphasises on word similarity, this resulted in a good METEOR score, but poor human judgement score.

Typically, It is believed that a lower word error rate shows a superior accuracy in recognition of speech, compared to a higher word error rate. From the table above, we can observe that the Fair prediction has a lower WER than the Good prediction. There by showing that a true understanding of language relies on more than just high word error rate/accuracy.



On the contrary, the BLEU-4 scores are in line with the human judgement because it takes into consideration both adequacy and fluency. Here, Adequacy measures if all the meaning of the source was expressed in the prediction, Fidelity measures translation accuracy, and Fluency measures how grammatically well formed the prediction is. Essentially, These three factors are what make a good sentence according to humans. Hence, the alignment between BLEU-4 scores and human judgement.

### 7.3. Model: Vanilla RNN vs Attention

MODEL	METRICS		WER	Predictions
ATTENTION + ResNet	BLEU 1	16.6	WER matrix (6x9): [[0. 1. 2. 3. 4. 5. 6. 7. 8.] [1. 0. 1. 2. 3. 4. 5. 6. 7.] [2. 1. 1. 2. 3. 4. 5. 6. 7.] [3. 2. 1. 2. 3. 4. 5. 6. 7.] [4. 3. 2. 2. 3. 4. 5. 6. 7.] [5. 4. 3. 3. 3. 4. 5. 6. 7.]] 7	
	BLEU 2	40.8		
	BLEU 3	55.3		
	BLEU 4	63.8		
	GLEU (1 to 4 grams)	13.3		
	GLEU (1 to 2 grams)	23.5		
	METEOR	17.44		
Vanilla + ResNet	BLEU 1	7.6	WER matrix (13x9): [[1. 0. 1. 2. 3. 4. 5. 6. 7. 8.] [1. 1. 1. 2. 3. 4. 5. 6. 7. 8.] [2. 2. 2. 3. 4. 5. 6. 7. 8.] [3. 3. 3. 2. 3. 4. 5. 6. 7.] [4. 4. 4. 3. 3. 4. 5. 6. 7.] [5. 5. 5. 4. 4. 4. 5. 6. 7.] [6. 6. 6. 5. 5. 5. 5. 5. 6.] [7. 6. 7. 6. 6. 6. 6. 6. 6.] [8. 7. 7. 7. 7. 7. 7. 7. 7.] [9. 8. 8. 8. 8. 8. 8. 8. 8.] [10. 9. 9. 9. 9. 9. 9. 9. 9.] [11. 10. 10. 10. 10. 10. 10. 10.] [12. 11. 11. 11. 11. 11. 11. 11.]] 11	Real Caption: <start> a man surfing a long board down a wave <end> Prediction Caption: the picture of a picture of a man is flying low waves <end> 
	BLEU 2	27.7		
	BLEU 3	42.8		
	BLEU 4	52.6		
	GLEU (1 to 4 grams)	8.6		
	GLEU (1 to 2 grams)	16		
	METEOR	32.2		

## INFERENCE

It can be observed that for the given two images, the attention model performs better than the vanilla model on all the automatic NLP metrics except METEOR, and also on human judgement. Furthermore, the attention model owing to its nature is more descriptive than the vanilla model. Moreover, we can also observe the various regions of the image along with the contribution to the specific words of the caption, thus demonstrating the interpretability of the attention model. The attention model has shown better performance on other images as well.

**View:** [Detailed Results](#)

## 8. Repository

**GITHUB REPOSITORY:** [Open Repository](#)

<https://github.com/HemanthTejaY/Deep-Learning-Image-Captioning—A-comparitive-study>

## 9. Conclusions

Having implemented and studied the performance of both Vanilla RNN and Attention based Image Captioning techniques on MS COCO Dataset, Natural Images and Abstract Images, we can conclude that (in terms of performance on both human judgement and automatic metrics) Attention Mechanism is superior to the Vanilla RNN Image captioning on COCO dataset and also Natural Images with fewer subjects. However, both the models perform about the same on abstract paintings and on Natural images

with a large number of subjects. Design considerations such as the choice of CNN-Encoder too affect the performance of the model. Moreover, we have observed that the currently used automatic metrics for image captioning, judge the caption quality by determining similarity between candidate and references captions. As a result, these metrics fail to achieve adequate levels of correlation with human judgements at the sentence level, which reflects the fact that they do not fully capture the set of criteria that humans use in evaluating caption quality.

We have accomplished our goal of comparing and understanding the architecture and performance aspects of deep learning based image captioning models. In the process, we have learnt about interplay between Computer-Vision and NLP, encoder-decoder architectures, loss functions in complex architectures, and the crucial ability to comprehend academic research papers in deep learning.

## References

- [1] Shetty, Rakshith, et al. “Speaking the Same Language: Matching Machine to Human Captions by Adversarial Training.” ArXiv:1703.10476 [Cs], Nov. 2017. arXiv.org, <http://arxiv.org/abs/1703.10476>.
- [2] Hossain, Md Zakir, et al. “A Comprehensive Survey of Deep Learning for Image Captioning.” ArXiv:1810.04020 [Cs, Stat], Oct. 2018. arXiv.org, <http://arxiv.org/abs/1810.04020>.
- [3] Tavakoli, Hamed R., et al. “Paying Attention to Descriptions Generated by Image Captioning Models.” ArXiv:1704.07434 [Cs], Aug. 2017. arXiv.org, <http://arxiv.org/abs/1704.07434>.
- [4] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. CoRR, abs/1411.4555, 2014.
- [5] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. CoRR, abs/1502.03044, 2015.
- [6] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollar, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, Computer Vision – ECCV 2014, pages 740–755, Cham, 2014. Springer International Publishing.
- [7] Polina Kuznetsova, Vicente Ordonez, Alexander C. Berg, Tamara L. Berg, and Yejin Choi. Collective generation of natural image descriptions. pages 359–368, 2012.
- [8] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. IEEE Trans. Pattern Anal. Mach. Intell., 39(4):664–676, Apr. 2017.
- [9] PyTorch. [https://ml-cheatsheet.readthedocs.io/en/latest/loss\\_functions.html](https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html).