San Jose State University
SJSU ScholarWorks

Master's Projects | Master's Theses and Graduate Research

Spring 2023

Image Captioning using Reinforcement Learning

Venkat Teja Golamaru
San Jose State University

Follow this and additional works at: https://scholarworks.sjsu.edu/etd_projects

 Part of the Artificial Intelligence and Robotics Commons

Recommended Citation

Golamaru, Venkat Teja, "Image Captioning using Reinforcement Learning" (2023). *Master's Projects*. 1225.
DOI: https://doi.org/10.31979/etd.6yj7-7qpf
https://scholarworks.sjsu.edu/etd_projects/1225

This Master's Project is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Projects by an authorized administrator of SJSU ScholarWorks. For more information, please contact scholarworks@sjsu.edu.

Image Captioning using Reinforcement Learning

Project Report

Presented to

Department of Computer Science

San Jose State University

In Partial Fulfillment

Of the Requirements for the Class

Spring-2023: CS 298

By

Venkat Teja Golamaru

© 2023

Venkat Teja Golamaru

ALL RIGHTS RESERVED

Image Captioning using Reinforcement Learning

By Venkat Teja Golamaru

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSÉ STATE UNIVERSITY

2023

Prof. William Andreopoulos	Department of Computer Science
Prof. Fabio Di Troia	Department of Computer Science
Prof. Nada Attar	Department of Computer Science

ACKNOWLEDGEMENT

I wish to extend my deepest appreciation to my project supervisor, Prof. William Andreopoulos, for his unwavering support and guidance throughout the development and completion of this master's project. His expertise, patience, and dedication to helping me tackle the challenges and intricacies of the project have been invaluable in its successful realization. Prof. Andreopoulos' timely feedback and insights have greatly influenced the trajectory of this project, enhancing its overall quality and impact. In addition, I would like to acknowledge the faculty members of the Department of Computer Science at San José State University for imparting their knowledge and expertise in the advanced computer science courses that have contributed to my growth in this field. Their commitment to teaching and nurturing the next generation of computer scientists is truly commendable. Finally, I wish to express my gratitude to my family and friends, whose unwavering support and encouragement have been the foundation upon which I have built my success in completing the Master of Science in Computer Science program at San Jose State University.

ABSTRACT

Image captioning is a crucial technology with numerous applications, including enhancing accessibility for the visually impaired, developing automated image indexing and retrieval systems, and enriching social media experiences. However, accurately describing the content of an image in natural language remains a challenge, particularly in low-resource settings where data and computational power are limited. The most advanced image captioning architectures currently use encoder-decoder structures that incorporate a sequential recurrent prediction model. This study adopts a typical Convolutional Neural Network (CNN) encoder Recurrent Neural Network (RNN) decoder structure for image captioning, but it has framed the problem as a sequential decision-making task. The image captioning models in this research used reinforcement learning (RL) as a means of training to improve performance. The study uses a policy network to anticipate the following word in a caption based on earlier predicted words and a value network to assess the entire caption and its possible variations. Both these networks have been trained using a reinforcement learning model that relies on visual-semantic embeddings. This method outperforms the standard encoder-decoder framework even with minimal training on a smaller subset of the Microsoft COCO dataset.

Keywords - Image Captioning, RL (Reinforcement Learning), CNN (Convolutional Neural Network), RNN (Recurrent Neural Network)

TABLE OF CONTENTS

I. INTRODUCTION

Image captioning is the process of producing textual descriptions of a given image. In computer vision, the field of image captioning has garnered substantial interest due to its goal of endowing robots with human-like intelligence, enabling them to understand visual information and convey it in natural language [1, 2, 3, 4, 5, 6, 7]. Many cutting-edge methods encode visual data with CNNs and use RNNs for decoding the data into coherent sentences. The recurrent hidden state is leveraged to estimate the likelihood of the succeeding word during both the training and inference stages.

This work combines policy and value networks to select the most appropriate words at each time step, with the help of a new framework. The policy network functions as a conventional maximum likelihood estimation (MLE) estimator for estimating the most likely outcome and produces predictions about the following word, utilizing the present state as a reference point. On the other hand, the value network evaluates all possible reward values of all possible situations that can come from the current situation, to give better global and lookahead guidance. This allows the policy network to generate more accurate predictions for the next word. By incorporating both local and global cues, the proposed approach enhances the accuracy of the predicted words, including those with low likelihood scores from the policy network alone.

This system employs an actor-critic reinforcement learning approach [8] that is based on visual-semantic embedding reward [9, 10, 11, 12]. Simultaneously, the training is improved by this method for both the value and policy networks. Initially, the value network undergoes pre-training through mean squared loss, while the policy network goes through pre-training using cross-entropy loss and supervised learning.

The tests were conducted on a downsampled training dataset of MS COCO [13] dataset against various evaluation metrics, including BLEU [14], Meteor [15], Rouge [16], and CIDEr [17]. The results consistently demonstrated that this system outperforms the standard encoder-decoder technique that lacks reinforcement learning optimization.

II. Related Works

A. Image Captioning

Image captioning generates a description of a given image, which involves an ML algorithm analyzing the visual attributes of the input image and describing it in natural language. This task is challenging because it requires processing natural language, which is inherently complex, as well as visual perception, which is also complex. This study offers a complete overview of image captioning, including the most important image captioning approaches and techniques, as well as the problems and limits of the approach.

Previous techniques analyzed the primary content items of an image using image recognition models and then utilized language models to integrate the important inferred characteristics into a caption [18,19,20,21,22]. Yet, this disjointed training approach led to poor generalization and overall performance. Yet, since the effort of converting or "translating" image contents into captions is conceptually similar to the task of converting one language to another in machine translation, researchers adapted machine translation techniques related to captioning of images. With the introduction of a good architecture of encoder-decoder [1, 2, 6, 7], it has become the most used approach for image captioning. Many vision architectures and modeling algorithms have been developed for the encoder component [23, 24, 25, 26], ranging from simple image recognition to contemporary attention-based methods emulating human visual systems [3, 4].

Graph-based approaches [27], such as scene graphs, have been used by the encoder to encode the training images, followed using graph-oriented DL models like Graph Convolutional Networks (GCN) for inference. The dearth of pre-trained models that are readily applicable for constructing scene graphs necessitates that researchers invest considerable time and computational resources into training such models from scratch [42].

The embeddings produced by the encoders are supplied into the decoder, which is some type of language model, such as RNN or LSTM [28, 29], or the latest state-of-the-art transformer-based decoders. Most inference techniques employ decoder mechanisms that employ beam or greedy search. Many inference techniques in image captioning use decoder mechanisms such as beam or greedy search. These mechanisms choose words based on local confidence and move to the next state, anticipating the words with the highest confidence. Top local confidence score refers to the probability score assigned by the model to the most probable word at each step of generating a caption. It is used to select the next word in the caption generation process based on the word with the highest score at that moment. However, this approach may ignore potentially better words in the early stages, leading to suboptimal captions. To address this issue, training image captioning models with reinforcement learning can overcome the limitations of next-word prediction based on local confidence scores.

B. Sequential Decision-Making

Several real-world problems can be formulated as sequential decision-making challenges, such as teaching agents to play computer games, algorithmic trading, and controlling robots [30, 31, 32]. Reinforcement Learning (RL) involves training an agent to make decisions based on its

interactions with the environment. RL algorithms such as Actor-Critic, Q-learning, and Reinforce are commonly used to solve decision-making problems [33, 34, 35].

The environment for image captioning using RL embeddings comprises an image and a caption that has been partially constructed. The agent is compensated based on the degree to which the created caption matches the ground truth caption. The agent learns to generate captions that garner higher rewards by modifying its rules depending on user feedback.

The RL models are also extremely dependent on the formulation of rewards. Earlier efforts of RL in text creation optimized only certain measures, necessitating re-training if the metrics were altered and making them non-generalizable.

A hybrid visual-semantic embedding has been employed, which is metric-independent and works well across a variety of assessment measures without retraining. Furthermore, a decision-making framework has been employed to enhance the produced captions, as opposed to typical approaches that adhere to the existing encoder-decoder structure [36].

III. TECHNICAL BACKGROUND

A. Convolutional Neural Networks

Convolutional Neural Networks (CNNs) [37] have proven to be remarkably effective across various computer vision applications. They work by applying convolutional operations to input images to extract meaningful features. A typical CNN consists of fully connected layers, pooling layers, and multiple convolutional layers. A group of filters or kernels, which can be adjusted through training, is embedded in every convolutional layer and applied to the input image to generate a set of feature maps.

The process of convolution entails shifting a kernel across the input image and computing the dot product between the kernel and the input at each location, resulting in a value that corresponds to a location on the feature map. For every convolutional layer, several kernels are utilized to capture distinct features of the input image and a non-linear activation function is employed to introduce non-linearity. Here is how the convolution operation is defined:

$$y_{i,j} = \sum_{m=1}^{M} \sum_{n=1}^{N} x_{i+m-1,j+n-1} w_{m,n}$$

Pooling is a crucial CNN technique used to lower the input's dimensionality and downsample the feature maps. Max pooling is a popular pooling procedure that chooses the maximum value inside a fixed-size window and discards the remainder. The formula for maximum pooling is:

$$y_{i,j} = \max_{m,n} x_{i+m-1,j+n-1}$$

In CNNs, the size of the output feature maps can be regulated by using padding. It includes adding additional pixels or values around the borders of the input image to maintain spatial information and prevent information loss during convolution. There are typically two forms of padding: valid padding, which includes no padding, and same padding, which adds and adjusts the input image padding to ensure that the output feature map has specific spatial dimensions.

B. Recurrent Neural Networks

The use of Recurrent Neural Networks (RNNs) [28] to generate image descriptions has grown widespread. In this strategy, the RNN model gradually analyzes the input image to produce a collection of words as output. This model is responsible for digesting each input element and remembering previous inputs to generate the output caption.

Initially, an input image is processed through a pre-trained CNN to obtain a fixed-length vector representation. This vector is then passed through an embedding layer that transforms the continuous image characteristics into discrete embeddings. Subsequently, the resulting embeddings are used as input into the RNN model.

The RNN model consists of many recurrent layers, each of which is responsible for processing a separate input element (an embedding corresponding to a word in this case). The output of each recurrent layer serves as input for the succeeding layer, enabling the model to recall previous inputs. The last recurrent layer's output is routed via a fully connected layer to generate the output caption. During training, the RNN model is optimized by decreasing the cross-entropy loss between the predicted and real caption.

Inference is achieved by repeatedly feeding the model's output into the input until the end-of-sentence token is created. The RNN model may be quantitatively expressed as follows:

Forward Pass:

$$h_t = f(W_{xh}x_t + W_{hh}h_{t-1} + b_h)$$

$$y_t = g(W_{hy}h_t + b_y)$$

where x_t is the input, h_t is the hidden state, y_t is the output at the time step t. The output and recurrent layers use the activation functions g and f, respectively. The matrices W_{hy}, W_{hh}, and W_{xh} are used to represent the weights, while the vectors used to represent the biases are b_y and b_h.

The backward pass of an RNN computes the gradients of the loss function with respect to the model parameters using the chain rule. Gradients can be determined using the following formulas:

Backward Pass:

$$\frac{\partial L}{\partial h_t} = \frac{\partial L}{\partial y_t} \cdot \frac{\partial y_t}{\partial h_t} + \frac{\partial L}{\partial h_{t+1}} \cdot \frac{\partial h_{t+1}}{\partial h_t}$$

$$\frac{\partial L}{\partial W_{xh}} = \sum_{t=1}^{T} \frac{\partial L}{\partial h_t} \cdot \frac{\partial h_t}{\partial W_{xh}}$$

$$\frac{\partial L}{\partial W_{hh}} = \sum_{t=1}^{T} \frac{\partial L}{\partial h_t} \cdot \frac{\partial h_t}{\partial W_{hh}}$$

$$\frac{\partial L}{\partial W_{hy}} = \sum_{t=1}^{T} \frac{\partial L}{\partial y_t} \cdot \frac{\partial y_t}{\partial W_{hy}}$$

where the derivatives are obtained using the chain rule, where the loss function is denoted as L and the length of the series is represented by T. Hence, RNN-based image captioning has been shown to be an effective way for generating image captions.

Gradient clipping is frequently employed to prevent gradients from getting too big during training, which can result in gradients bursting. This includes rescaling gradients that surpass a certain threshold. The gradient clipping formula is:

$$\text{if } ||\nabla|| > \text{threshold} : \nabla \leftarrow \frac{\text{threshold}}{||\nabla||} \cdot \nabla$$

where ∇ is the gradient vector and threshold is the maximum allowed norm.∇ is the gradient vector and threshold is the maximum allowed norm [43].

C. Greedy Search

Greedy search is a common decoding technique used in image captioning that constructs a sentence by picking the most probable word at each stage.

Formally, let's denote an input image by I, and a sequence of T words representing the caption by $w_{1:T}$ denote. The goal is to maximize the likelihood of the caption of the input image, i.e., $P(w_{1:T}|I)$.

At each time step t, the greedy search algorithm picks the most probable word $\hat{w}_t$ given the previous words and the image and feeds it as input to the next step. In mathematical terms, it may be expressed as:

$$\hat{w}t = \arg\max w \in VP(w|I, \hat{w}_{1:t-1})$$

where V is the vocabulary of words.

Typically a transformer or an RNN is used in computing the probability $P(w|I, \hat{w}_{1:t-1})$. The visual characteristics and the embedding of the preceding word are given as input to the model, and it generates a probability distribution across the vocabulary.

The greedy search method has several benefits, including simplicity and speed. Yet, it is susceptible to the issue of suboptimality, in which locally optimum judgments at each phase do not result in the globally optimal solution. This might lead to repeated or insufficient captions. Beam search and other complex decoding methods can be employed to overcome this issue.

D. Beam Search

Beam search [38] is an extension of the greedy search technique utilized in image captioning. This algorithm creates a sequence of words by keeping track of the top k most probable sequences at each step, where k is the beam width.

Formally, let's denote an input image by I, and a sequence of T words representing the caption by $w_{1:T}$ denote. Similar to the greedy search method, the goal is to maximize the likelihood of the caption of the input image, i.e., $P(w_{1:T}|I)$. The beam search algorithm maintains a collection of k partial sequences $s_1, s_2, ..., s_k$, where each sequence s_i is assigned a probability score p_i. At each t-th time step, the algorithm creates the next k potential extensions of each sequence in the set and retains the k sequences with the greatest probability scores. In mathematical terms, it may be expressed as:

$$S_t = s' \mid s \in S_{t-1},\ w \in V,\ s' = s + [w],\ p_{s'} = p_s \cdot P(w|I, s')$$

where V is the lexicon of words, p_s is the sequence probability score, and $[w]$ is the word embedding of w.

The beam search process finishes when the produced sequences reach a certain length or when the top k sequences cannot be improved further. The output sequence is the one with the highest probability score.

The beam search method explores a bigger search space than the greedy search strategy, which can result in higher-quality captions. Due to the limited beam width, it may also experience the problem of becoming trapped in suboptimal solutions. Diverse beam search is one strategy that may be utilized to alleviate this issue.

E. Principal Component Analysis

A well-known technique for reducing dimensionality in data analysis is Principal Component Analysis (PCA) [39]. In the realm of image captioning, one can utilize PCA to decrease the image feature vector size generated by a CNN such as VGG-16. The resulting smaller feature vectors can then be inputted into a decoder powered by an RNN for generating captions for the images.

PCA is founded on the principle of projecting high-dimensional data onto a lower-dimensional space while preserving as much of the data's original variance as feasible. The fundamental concept is to select a collection of new orthogonal axes, called principle components, that encapsulates the greatest amount of data variation. The first principal component is the axis that captures the most variance orthogonal to the first, and so on. The number of maintained primary components is dependent on the desired degree of dimensionality reduction.

Using the original data's covariance matrix's eigenvectors and eigenvalues, the major components are calculated. Let X be an $n \times m$ matrix consisting n samples of m-dimensional feature vectors. To generate a centered matrix Y, the mean of the feature vectors is first removed from each sample:

$$Y = X - \bar{X}$$

where $\bar{X}$ is the mean of the columns of X. The covariance matrix C of Y is then computed as:

$$C = \frac{1}{n-1} Y Y^T$$

Next, the eigenvectors and eigenvalues of C are computed, and the eigenvectors are ranked based on their associated eigenvalues. The initial k eigenvectors are then chosen to create the new $mtimesk$ matrix V_k. The $ntimesm$ original feature matrix X is projected onto the k-dimensional subspace covered by the columns of $V - k$:

$$Z = XV_k$$

The resultant matrix Z comprises the lower-dimensional feature vectors that may be fed into the RNN decoder.

PCA can reduce the computational cost of training an image captioning model and improve its generalization performance by lowering the danger of overfitting. Nonetheless, it is crucial to select the number of major components to keep with care, since preserving too few might result in the loss of valuable information, whilst retaining too many can result in overfitting [42].

F. VGG16

Andrew Zisserman and Karen Simonyan of the University of Oxford created VGG16 in 2014, which is a deep CNN architecture. It has become one of the most popular image classification models and has been modified for additional computer vision applications, including image captioning.

The VGG16 architecture [40] consists of sixteen layers. To obtain the information from the input image, the thirteen convolutional layers are utilized, whilst the classification or regression tasks are handled by the three fully connected layers. Each of the first thirteen convolutional layers employs 3x3 pixel filters, while the last three fully linked layers include 4096 units each. With around 138 million parameters, the VGG16 architecture is one of the biggest deep CNN architectures.

The VGG16 architecture is frequently deployed as a pre-trained model for feature extraction in image captioning. The visual characteristics extracted by the convolutional layers are then transferred to an RNN for caption synthesis. The RNN creates the caption word by word depending on the system's current state, which includes image attributes and previously created caption words.

Several approaches, such as maximum likelihood estimation (MLE) or reinforcement learning, may be employed to fine-tune the VGG16 architecture for image captioning (RL). MLE entails optimizing the likelihood of producing the ground-truth caption given the image input. RL entails learning to develop captions that receive high rewards based on their resemblance to the ground truth caption. Typically, the fully connected layers of the VGG16 model are deleted or frozen

during fine-tuning, and the remaining layers are trained using the image-caption pairings in the dataset.

In summary, VGG16 architecture is a proven architecture for image captioning that has been extensively adopted in the industry. It can be fine-tuned for image captioning using different techniques such as MLE or RL. After generating the features, an RNN is employed to create captions. Some advanced image captioning methods currently use VGG16.

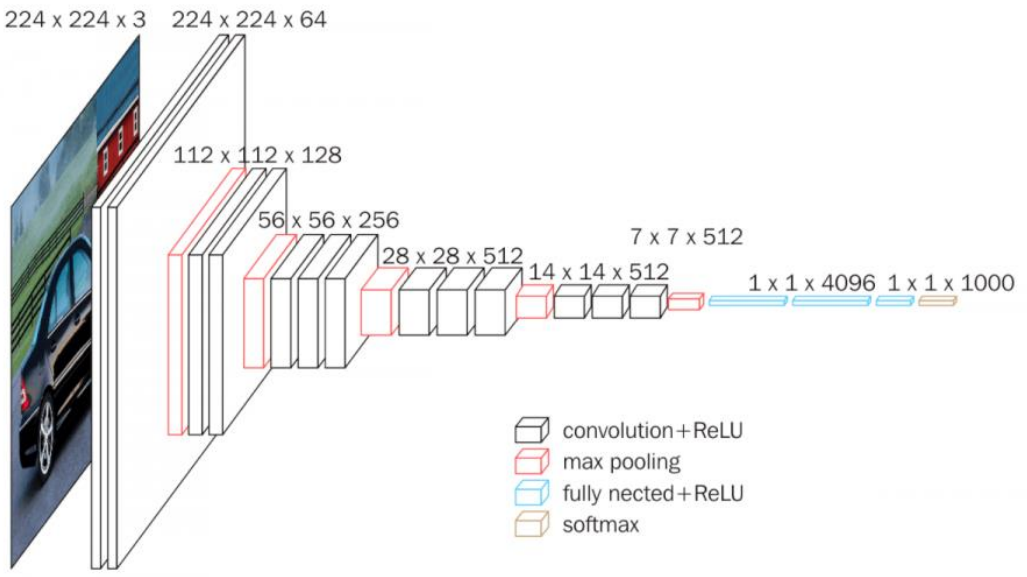


Figure 1. VGG 16 architecture

IV. EXPERIMENTAL DETAILS

A. Dataset

The dataset “Microsoft Common Objects in Context”, or “MS COCO” [13] is a frequently utilized benchmark for image captioning tasks. It consists of 330K photos with 2.5 million item

instances and 5 descriptions per image that have been carefully annotated by humans. The dataset is divided into training, validation, and testing subsets, each containing distinct sets of images. These images belong to a diverse range of categories such as people, animals, and cars, as well as indoor and outdoor scenes.

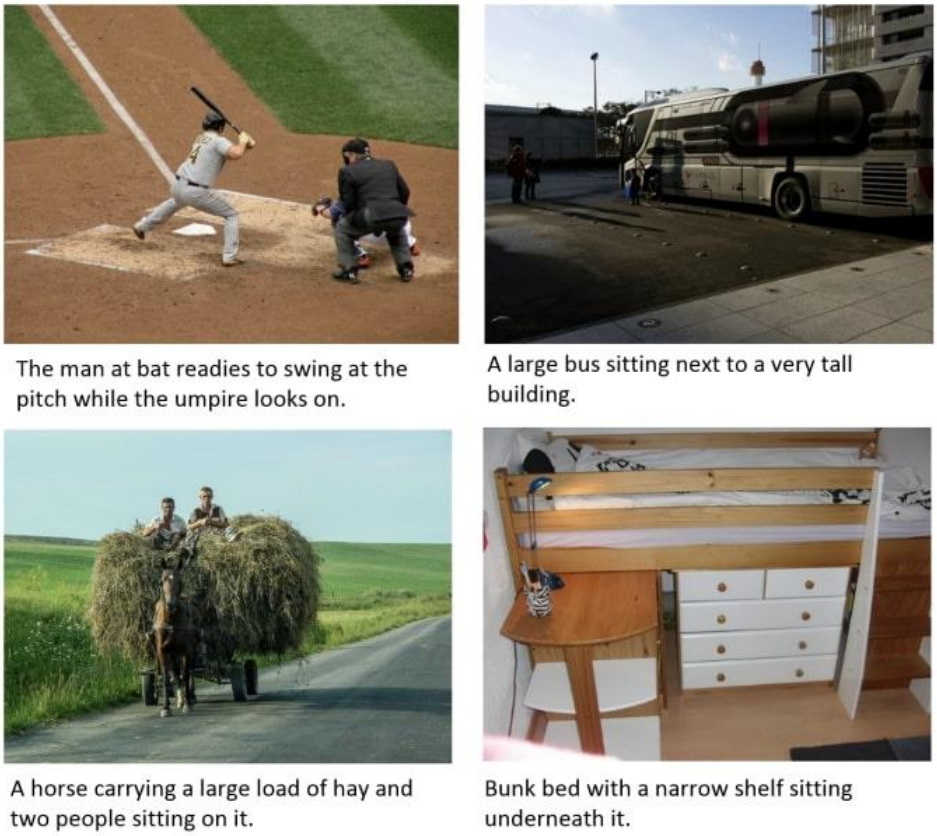


Figure 2. Examples of MS COCO Dataset contents

Because of its numerous annotations and high-quality images, the MS COCO dataset is frequently utilized for image captioning applications. Generating written explanations for images is known as image captioning, and it is a challenging undertaking that necessitates the use of both computer vision and natural language processing (NLP) techniques. The MS COCO dataset is ideally suited for this purpose since it contains a huge quantity of labeled images that can be utilized to train image captioning models. This dataset has been used in multiple works related to image captioning. Typically, a CNN is trained to extract visual features from image data, which

are then processed by an RNN to generate captions. RNN is educated to anticipate the following term in the caption by utilizing picture features and past words.

Nevertheless, an alternative approach is to employ pre-trained transformer-based vision model such as ViT, BEiT, DeiT, Swin as the encoder and any pre-trained language model such as RoBERTa, GPT2, BERT, DistilBERT as the decoder, which can create captions directly from image information. These models are trained on substantial volumes of textual data and can produce captions that are both grammatically accurate and semantically relevant. The MS COCO dataset is a great resource for image captioning tasks and to train and evaluate machine learning models as they provide a vast and diverse collection of labeled images that can be used.

B. Preprocessing

A subset of the MS COCO dataset, with the 2014 splits was used for the training and inference of the model. Instead of using raw images, 512-dimensional feature vectors were used, which were extracted from the fc7 layer of VGG-16. The original feature vectors were 4096-dimensional, but PCA was applied to reduce them to 512 dimensions. The word vectors used in the model were also 512-dimensional.

C. Problem Definition

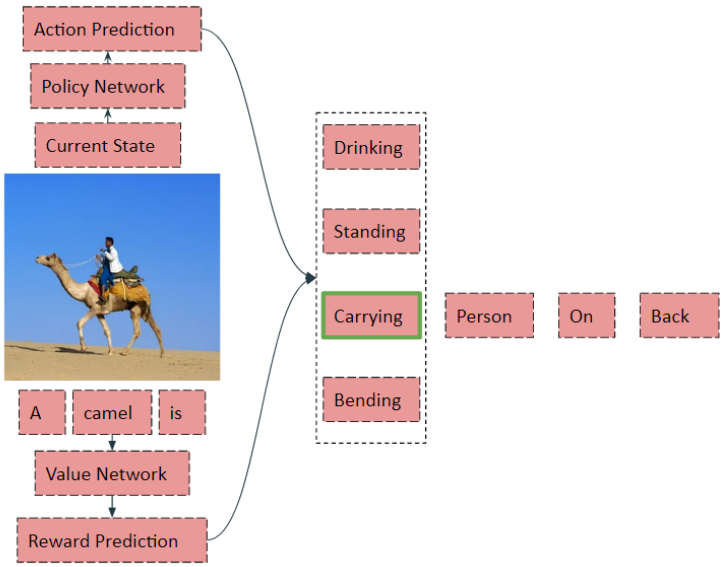


Figure 3. Proposed Framework where Value and Policy Network collaboratively produce better captions

As a decision-making process, RL-based algorithms have been utilized to optimize the training of the encoder-decoder. $S = w_1, w_2, ..., w_T$ aims to generate an accurate and comprehensive caption that describes the content of an image I. The agent is represented by a policy and value network p_π, v_θ respectively, whereas the environment includes image I and the generated word ($w1, ..., wt$). At each time increment, the agent takes action by predicting the next word (w_{t+1}).

This formulation can be represented mathematically as:

Agent: policy and value network p_π, v_θ respectively.

Action: predicting the following word w_{t+1}

Environment: predicted words $\{w_1, ..., w_t\}$, and input image I

Goal: Generate a sentence, to accurately portray the image I, denoted as $S = \{w_1, w_2, ..., w_T\}$

D. State and action space

In reinforcement learning (RL), states and actions are auto-regressive components. A new state is observed by the agent after each action. When generating captions for an image, the current state at time step (t) consists of both the words produced up to that point ($w_1, ..., w_t$) and the image (I).

At each time step, the agent takes action by selecting the most probable next word based on the policy network. However, unlike conventional RL problems with a limited action space, the action space in image captioning is the entire dictionary (Y) from which the words are drawn.

$$s_t = I, w_1, ..., w_t$$

$$a_t \subseteq Y$$

E. Policy Network and Value Network

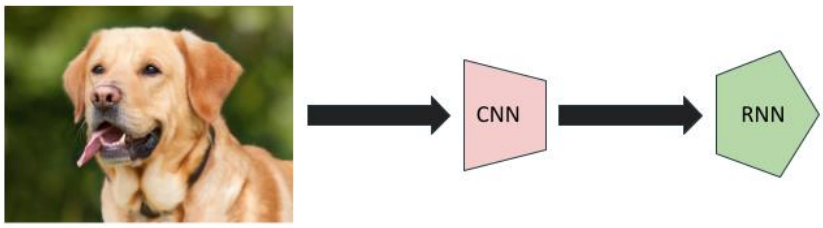


Figure 4. Policy Network that predicts next word (action) depending on the current state

The proposal presented in this paper introduces CNN_p and RNN_p, the two policy networks that are used for the purpose of determining the likelihood of an agent taking a certain action, $a_t = w_{t+1}$, at a specific state $s_t = I, w_1, ..., w_t$. The network employs a combination of a CNN and an RNN, similar to the encoder-decoder model utilized for generating image captions.

The image I uses the CNN_p network to encode its visual information and passes it onto the starting input node $x_0 \in R^n$ of the RNN_p network. The policy for taking actions in the RNN_p network at each time step is determined as the hidden state $h_t \in R^m$ evolves over time. During each iteration, RNN_p receives the output word w_t as input x_{t+1}, causing the state of RNN_p to change to h_{t+1} from h_t. The following set of equations specifies the policy network p_π:

$$x_0 = W_{x,v} CNNp(I)$$
$$h_t = RNNp(h_{t-1}, x_t)$$
$$x_t = \phi(w_{t-1}), t > 0$$
$$p_\pi(a_t|s_t) = \phi(h_t)$$

The linear embedding model's weights for visual data are referred to as $W_{x,v}$, and the RNN_p input and outputs are represented by ϕ and ψ, respectively. The policy network aims to boost an agent's decision-making capabilities in a particular state by considering both visual information and past actions.

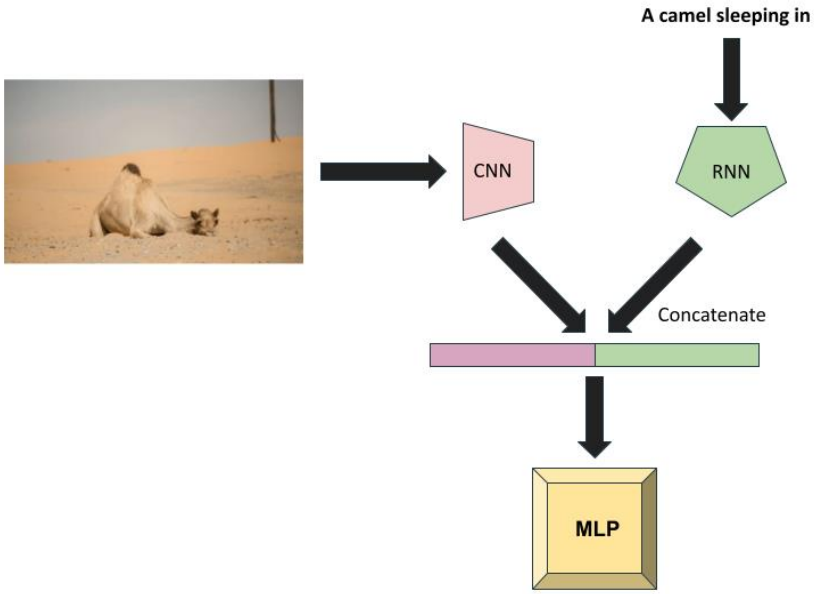


Figure 5. Value Network calculates value function of

the current state (image+generated caption upto time step t)

The value network is utilized to estimate the value function with reasonable accuracy, given the current state in the problem formulation. The value function, in turn, represents the cumulative net reward that can be achieved when starting from a particular state and following a prescribed policy by taking subsequent actions.

In mathematical terms, it is expressed as:

$$v_p(s) = \mathbb{E}a_t, ..., a_T \sim p[r|s_t = s, a_t, ..., a_T]$$

Where, v_θ is used to approximately estimate the value function. This network evaluates the state $s_t = I, w_1, ..., w_t$, comprising the partially generated sentences $(w_1, ..., w_t)$ and the raw image I. The value network includes three types of neural networks: CNN_v, RNN_v, and the multilayer perceptron (MLP_v). The RNN_v encodes the semantic information of the sentence that has been generated partially, while the CNN_v is responsible for encoding the visual information of the image I. From the state s_t, to estimate its scalar reward, all the components of the value network are trained at the same time.

F. Visual-Semantic Embedding Reward

Having a well-defined and reasonable optimization objective is crucial in reinforcement learning, as it serves as the reward for the learning agent. In terms of the reward signal, the use of visual-semantic embedding similarity is proposed. Visual-semantic embedding has proven to be a successful method in computer vision applications such as image captioning, classification, and retrieval. The proposed embedding model comprises a linear mapping layer, an RNN, and a CNN. Each of these components is denoted by f_e, RNN_e and CNN_e. The model creates

mappings among sentences and images in a common semantic embedded space, which enables it to gauge their likeness.

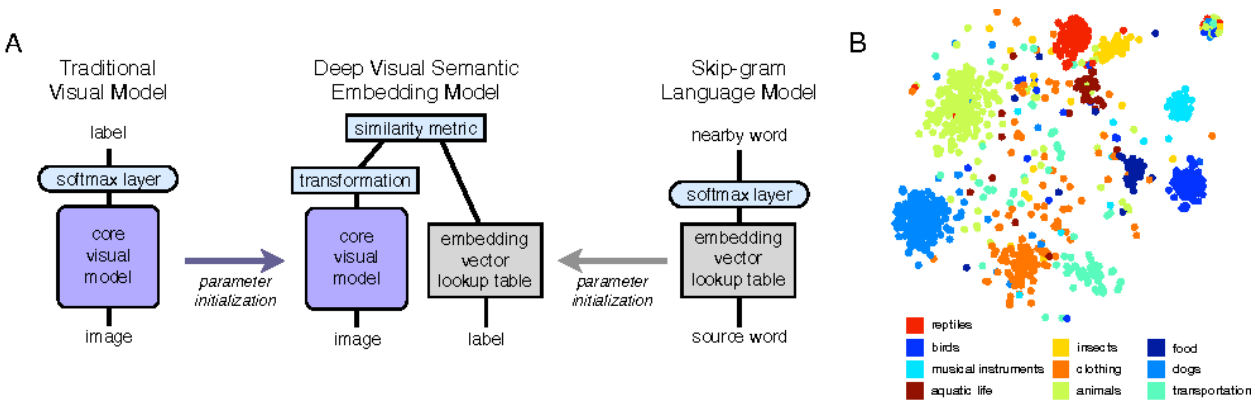


Figure 6. Joint Visual Semantic Embedding generation architecture [9]

The final hidden state of an RNN_e, denoted by $h_0(S)^T$, is used to represent the embedding feature of a given sentence S. The image I feature vector is represented with v, which is extracted using the CNN_e method. A mapping function, f_e, is used to map the image features to the embedding space. The image-caption pairs utilized for image captioning are utilized in the training of the embedding model. The weights of CNN_e are fixed, while the weights of RNN_e and f_e are learned using a bidirectional ranking loss, which is defined in the following manner:

$$L_e = \sum_{v} \sum_{S^-} \max(0, \beta - fe(v) \cdot h_0(S) + fe(v) \cdot h_0(S^-)) +$$

$$\sum_{S} \sum_{v^-} \max(0, \beta - h_0(S) \cdot fe(v) + h_0(S) \cdot fe(v^-))$$

In this context, the margin hyperparameter β is subject to cross-validation. The pairs of image and sentence denoted as (v, S) represent the true matches, while S^- refers to a negative representation of the image associated with v, and vice versa with v^-.

$$r = \frac{fe(v) \cdot h'_t(S_b)}{|fe(v)||h'_t(S_b)|}$$

In the above equation, at the time step t, the hidden state of the RNN is demonstrated by $h'_t(S_b)$ for S_b, the generated sentence. The reward r is normalized by the product of the norms of the image and sentence embedding vectors.

G. Training using deep reinforcement learning

A two-step learning approach has been adopted [30] to train both the value network v_θ and policy network p_π. Initially, using the cross-entropy loss with supervised learning to train the policy network. The loss function, L_{p_0}, is the negative logarithm of the likelihood of a sentence generated by the policy π given input image I.

$$L_{p_0} = -\log p(w_1, ..., w_T|I; \pi) = -\sum_{t=1}^{T} \log p_\pi(a_t|s_t)$$

Keeping mean square loss as low as possible, the value network is trained:

$$||v_\theta(s_i) - r||_2^2$$

The final reward associated with a produced sentence is represented by $$r$$, while $$s_i$$ indicates a state that is chosen at random during the sentence generation. Because each subsequent state in sentence generation is highly correlated to the previous one and differs by only a single word, the regression objective is applied to and shared over the entire process of image captioning. To avoid the problem of overfitting, a single state is randomly selected from each unique sentence.

Algorithm 1 A2C algorithm

Require: Initialize Actor-Critic network G with parameter θ.

for each episode **do**
  Get initial state s
  Initialize a storage buffer S, A, R, V, S'
  for $i = 1, 2, 3.., N$ **do**
    Sample an action $a \sim G(s)_\pi$ and get the associeted value $v \leftarrow G(s)_v$
    Run the action a through the environment, obtain the reward and next state $r, s' \leftarrow ENV(s, a)$
    Collect and store $S, A, R, V, S' \leftarrow s, a, r, v, s'$
    $s \leftarrow s'$
  end for
  Compute the discounted returns $\hat{V} = \sum_{l=0}^{N-1} \gamma^l r_{t+l}$
  Compute an advantage function $\psi(V, R, S')$
  Optimize θ to minimize $-\log(G(A \mid S)_\pi)\psi(V, R, S') + \lambda\|V - \hat{V}\|$
  Empty S, A, R, V, S'

Figure 7. Pseudo Code of A2C (Advantage Actor-Critic) Algorithm [8]

The next step involves training the value and policy networks jointly. Deep RL is used in the training process. In this step, the agent's parameters are denoted by $\Theta = \pi, \theta$, aiming to maximize the total reward it can acquire by engaging with its environment. To achieve this objective, Θ is learned through training.

$$J(\Theta) = \mathbb{E}s_1 ... s_T \sim p\pi \left[\sum_{t=1}^{T} r_t \right]$$

where $r_t = 0$ for all $0 < t < T$ and $r_T = r$. It is challenging to achieve precise optimization of the function J due to the need to estimate an expectation over complex interaction sequences in high dimensions. Moreover, these sequences may involve unknown environmental dynamics, further adding to the complexity. To overcome this challenge, the study adopts reinforcement learning techniques and approaches this issue like a partially observable Markov decision process

(POMDP). Previous studies have demonstrated that utilizing a policy gradient based on samples can be effective[32,33,34,35].

$$\nabla_\pi J \approx \sum_{t=1}^{T} \nabla_\pi \log p_\pi(a_t|s_t)(r_t - v_\theta(s_t))$$

$$\nabla_\theta J = \nabla_\theta v_\theta(s_t)(r - v_\theta(s_t))$$

In this technique, a value network v_θ is utilized as a baseline to enhance the stability of the policy gradient. To decrease the policy gradient's variance, it is necessary to subtract the value network evaluation from the gradient estimate. The term $r - v_\theta(s_t)$ is the approximation of the gain of the action taken in state s_t. Here in actor-critic architecture, p_π performs the role of the actor while the role of the critic is taken by v_θ, .

Due to the significant number of possible actions, which is roughly equivalent in size to the vocabulary (approximately 10^3), training the reinforcement algorithm becomes quite difficult. To overcome this challenge, the approach utilized curriculum learning [41] and followed the method introduced in [36] so that the actor-critic model will be trained.

To gradually increase the complexity of the training examples, the rest of the $i\Delta$ words were used to train the actor-critic model while keeping the first $(T - i\Delta)$ word constant. The value of i was increased from 1 to the maximum sentence length until the whole sentence was used for reinforcement learning. During this process, to train the model on fixed parts of sentences I've employed cross-entropy loss.

H. Hyperparameters

Maximum caption length = 17

Beam Length = 5

Policy Network: Epochs = 100000; Optimizer = Adam; Learning Rate = 0.0001

Reward Network: Epochs = 50000; Optimizer = Adam; Learning Rate = 0.001

Value Network: Epochs = 50000; Optimizer = Adam; Learning Rate = 0.0001

Advantage Actor-Critic Network: Number of Curriculum Levels= 8; Epochs = 1000; Optimizer = Adam; Learning Rate = 0.0001

I. Using policy network and value network for lookahead inference

The policy and value networks are utilized in this architecture to improve the quality of the generated captions during inference. This involves utilizing the policy network's local guidance and the value network's global guidance in the lookahead inference process.

Beam Search (BS) is a widely used decoding technique in which the top-B (B refers to the beam length) highest-scoring candidates are stored during each time step, and this process is repeated at each subsequent time step. Although a scoring function may be used to evaluate the quality of captions, it is not always accurate as it may not recognize some good captions where not all words have high probabilities. Sometimes, it is advantageous to choose actions with low probabilities to optimize the final reward.

To address this problem, a lookahead inference is implemented, which merges the policy network, and value network to examine each and every possible choice in the set of options $W_t + 1$. Every action is performed by taking into account both the present policy and the reward assessment acquired from the lookahead.

$$S(w_{b,\lceil t \rceil}) = S(\{w_{b,\lceil t \rceil}, w_{b,t+1}\})$$
$$= S(w_{b,\lceil t \rceil}) + \lambda \log p_\pi(a_t|s_t) + (1-\lambda) v_\theta(\{s_t, w_{b,t+1}\})$$

V. METRICS AND RESULTS

A. BLEU Score

The BLEU score is frequently utilized in the evaluation of the quality of machine-generated translations, as it measures the similarity of n-gram words between a machine translation and reference translations created by humans. A higher score indicates better translation quality. The formula for computing the n-gram BLEU score involves the n-gram precision p_n, the assigned weight w_n, a brevity penalty (BP), and a maximum n-gram length N.

BLEU formula:

$$BLEU = BP \times \exp\left(\sum_{n=1}^{N} w_n \log p_n\right)$$

i. BLEU-1 (Unigram BLEU)

BLEU-1, commonly referred to as Unigram BLEU is a version of BLEU that examines just the overlap of unigrams in the reference captions and the machine-generated captions. This is used to evaluate the effectiveness of translation algorithms that tend to create extremely brief or fragmented phrases.

BLEU-1 formula:

$$BLEU - 1 = BP \times \exp\left(w_1 \log p_1\right)$$

where BP and p_1 are the same as in the BLEU formula, and w_1 is the weight assigned to the unigram precision.

ii. BLEU-2 (Bigram BLEU)

BLEU-2, commonly referred to as Bigram BLEU, is an additional form of BLEU that considers the overlap of bigrams in the reference captions and the machine-generated captions. This score is important for evaluating the performance of translation algorithms that create longer, more complicated phrases.

BLEU-2 formula:

$$BLEU - 2 = BP \times \exp\left(w_1 \log p_1 + w_2 \log p_2\right)$$

where BP and p_1 are the same as in the BLEU formula, the weight given to unigram precision is represented by w_1, while the bigram precision is represented by p_2, and w_2 is the weight assigned to the bigram precision.

B. METEOR

"Metric for Evaluation of Translation with Explicit Ordering", or "METEOR", calculates a score using precision, recall, and alignment measures, along with a semantic similarity component. The formula for calculating the METEOR score is:

$$\text{METEOR} = (1-\alpha)\frac{\text{precision} \times \text{recall}}{(1-\alpha) \times \text{precision} + \alpha \times \text{recall}}^{\beta} \times \text{similarity}$$

where α, β are tunable parameters, and similarity is the harmonic mean of unigram, bigram, and concept/entity matches.

The calculation of precision and recall depends on the count of correctly matched unigrams and bigrams between the output produced by the machine and the reference translations. To align these, a modified version of the Hungarian method is used, which aims to find the best match between the words in the machine-generated output and the reference translations.

Using WordNet and the WordNet-based overlap coefficient, the semantic similarity component is calculated. It assesses the degree to which ideas and entities in the machine-generated output and reference translations coincide.

Overall, METEOR provides a thorough evaluation of machine translation quality, taking into account variables such as fluency and adequacy as well as alignment and semantic similarity of the reference captions with the machine-generated output.

C. ROUGE_L

“Recall-Oriented Understudy for Gisting Evaluation”, or “ROUGE_L”, is frequently used as an evaluation metric in NLP for assessing the performance of text summarization. Its purpose is to calculate the longest common subsequence (LCS) of words between one or more reference summaries and a summary generated by a model.

This calculation helps to determine how much overlap there is between the two summaries. ROUGE_L is designed to prioritize recall, which means that it places more value on ensuring that

the machine-generated summary covers the information present in the reference summary, rather than focusing solely on the precision or exactness of the generated summary.

The ROUGE_L score is determined below:

$$ROUGE_L = \frac{\sum_{r \in Reference} \sum_{g \in Generated} LCS(r, g)}{\sum_{r \in Reference} \sum_{g \in Generated} |r|}$$

Here, "Reference" denotes the set of reference summaries, "Generated" refers to the summary generated by the machine, "LCS" represents the longest common subsequence of words shared amongst the reference summary "r" and generated summary "g", and "|r|" is the word count of the reference summary "r".

The numerator in this formula indicates the overall length of the LCS across all reference summaries and the machine-generated summary, whereas the denominator is the total length of all reference summaries in words. A higher ROUGE L score indicates a greater degree of overlap between the machine-generated and reference summaries.

D. CIDEr

CIDEr is commonly used to assess the effectiveness of image captioning algorithms. It evaluates machine-generated captions by comparing them to a collection of reference captions annotated by humans. CIDEr employs a consensus-based technique that considers the diversity and complexity of the reference captions, giving it a more robust and discriminatory statistic than other image captioning metrics.

The CIDEr score is calculated as follows:

$$CIDEr = \frac{1}{n}\sum_{i=1}^{n}\frac{\sum_{s\in S_i}\sum_{w\in s}\min(c(w), r(w))}{\sum_{s\in S_i}|s|}$$

The set of reference captions for the image I is denoted by S_i , and the dataset contains a certain amount of images, and this number is denoted by the symbol n. In addition, $c(w)$ is the frequency of word w in the machine-generated caption, while $r(w)$ is the frequency of word w in the reference captions.

The equation's numerator calculates the sum of the minimum occurrences of a word in both machine-generated and reference captions, while the denominator represents the mean length of the reference captions for image I.

In this formula, the CIDEr goes from 0 to 1, with a larger score showing a more favorable agreement between the machine-generated and reference captions. It has been demonstrated that the CIDEr metric exhibits a strong correlation with human evaluations of the quality of image captions and is widely used in research and industry for evaluating image captioning models.

VI. RESULTS

In conclusion, CNN-RNN based encoder-decoder architectures have exhibited significant success in image captioning tasks, although modeling long-term connections and generating diverse and coherent captions remain difficult. New developments such as reinforcement learning, and the utilization of attention mechanisms have tackled some of the difficulties and enhanced the effectiveness of basic image captioning models.

Three methods were used to perform inference on the validation dataset. The first method used the standard greedy approach, the second used the commonly used beam search approach,

and the third used the RL agent-based lookahead inference approach that was explored in this work.

It was observed that the RL-based approach outperformed the beam search method, which in turn outperformed the greedy inference method.

	“Greedy”	“Beam”	“RL Agent”
“BLEU – 1”	0.270771012	0.2727798112116863	0.278154106689217
“BLEU – 2”	0.12869147484950863	0.130068312	0.13207907633440683
“BLEU - 3“	0.06291573921435081	0.067936661	0.0704492797131235
“BLEU - 4”	0.03165603792113884	0.037411395	0.03961546368562404

Table 1. BLEU Metrics Tabular Comparison

	“Greedy”	“Beam”	“RL Agent”
“METEOR”	0.10094801455050988	0.10741747931952489	0.1086774165301927
“ROUGE_L”	0.25198083543261013	0.25966471919799977	0.2611785262449472
“CIDEr”	0.3844851625484268	0.41232866942825297	0.4247729924235357

Table 2. Metrics Evaluated for Greedy, Beam and RL agent

Greedy

{“Bleu_1”: 0.27077101225511235, “Bleu_2”: 0.12869147484950863, “Bleu_3”: 0.06291573921435081, “Bleu_4”: 0.03165603792113884, “METEOR”: 0.10094801455050988, “ROUGE_L”: 0.25198083543261013, “CIDEr”: 0.3844851625484268}

Beam

{“Bleu_1”: 0.2727798112116863, “Bleu_2”: 0.13006831245570807, “Bleu_3”: 0.067936661964644, “Bleu_4”: 0.0374113957268933, “METEOR”: 0.10741747931952489, “ROUGE_L”: 0.25966471919799977, “CIDEr”: 0.41232866942825297}

Agent

{“Bleu_1”: 0.278154106689217, “Bleu_2”: 0.13207907633440683, “Bleu_3”: 0.0704492797131235, “Bleu_4”: 0.03961546368562404, “METEOR”: 0.1086774165301927, “ROUGE_L”: 0.2611785262449472, “CIDEr”: 0.4247729924235357}

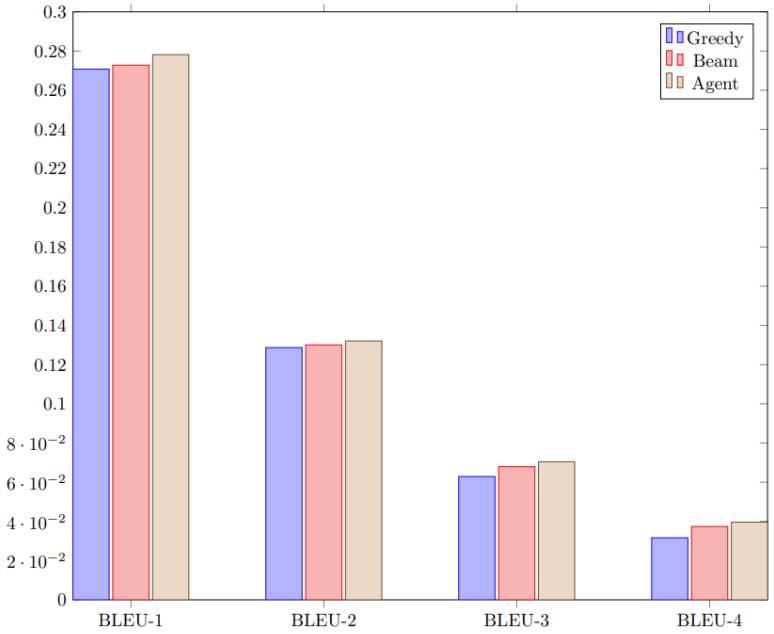


Figure 8. BLUE Metrics Comparison

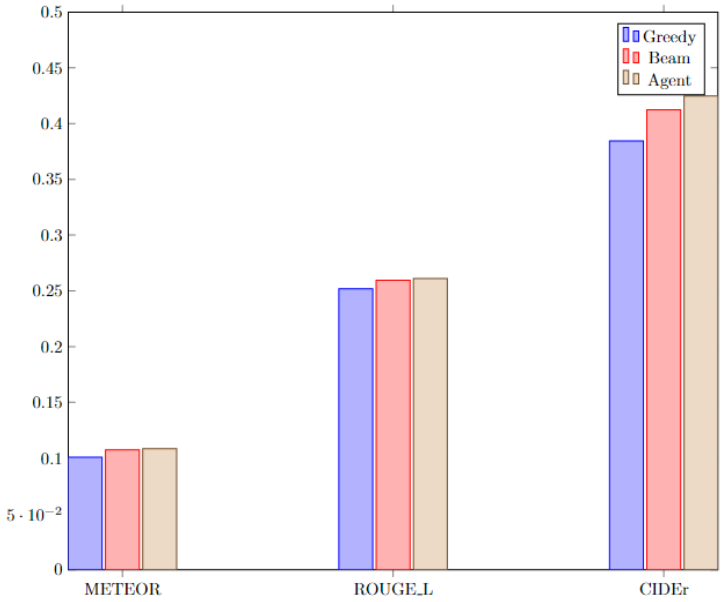


Figure 9. Other Metrics Comparison

Comparative Analysis of Results through Examples

GT : Ground Truth SLG : Supervised Learning Greedy Search (Baseline)

SLBS : Supervised Learning Beam Search RLL : Reinforcement Learning Lookahead inference (Improved Method)

GT: <START> a slice of a dish pizza sitting on a white plate <END>

SLG: <START> there is a plate with some pizza and <UNK> in it <END>

SLBS: <START> two slices of pizza sitting on top on a plate <END

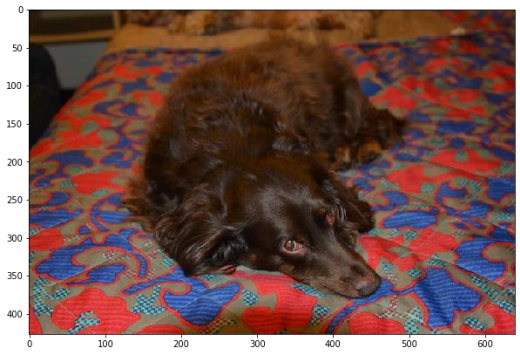
RLL: <START> a slices of pizza sitting on top on a white plate <END>

GT: <START> a person on skis riding through the snow <END>

SLG: <START> a person in the snow riding a red snowboard <UNK> <END>

SLBS: <START> a person skiing in the snow <END>

RLL: <START> a man that is skiing in the snow <END>

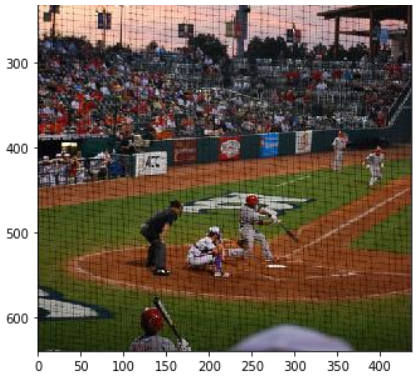


GT: <START> a close up of a dog laying on a bed <END>

SLG: <START> a cat that is laying down on a computer <END>

SLBS: <START> a grey and black dog laying on a bed <END>

RLL: <START> a dark brown dog laying on a bed <END>



GT: <START> baseball players playing a game of baseball during a baseball game <END>

SLG: <START> a baseball game showing a batter up to the ground <END>

SLBS: <START> a baseball player is trying to swing at a baseball game <END>

RLL: <START> a young baseball player is taking swing at a pitch <END>

GT: <START> a person stands in the snow with a snowboard <END>

SLG: <START> a person stands in the <UNK> on a <UNK> day <END>

SLBS: <START> there is a man standing in the <UNK> alone <END>

RLL: <START> a couple of people standing in the snow <END>

GT: <START> view of overhead jet plane on landing approach <END>

SLG: <START> a plane is flying through the sky <END>

SLBS: <START> a passenger airplane is flying high in the <END>

RLL: <START> a passenger airplane is flying high in the blue sky <END>

VII. Conclusion And Future Work

The given bar graph [Figure 8] shows the comparison of BLEU scores of three different methods: Standard Supervised Learning based Greedy Search which is the baseline, Supervised Learning based Beam Search, and the improved Reinforcement Learning based Lookahead Inference Agent. The x-axis represents the different n-gram values (BLEU-1, BLEU-2, BLEU-3, and BLEU-4), and the y-axis represents the BLEU scores.

The results clearly illustrate that the RL Agent method outperforms the Greedy and Beam methods in all the n-gram values. Among the n-gram values, the highest score is obtained for BLEU-1 for all three methods, and the lowest score is obtained for BLEU-4. This is expected because as the n-gram value increases the matching becomes more stringent.

The bar graph [Figure 9] is a performance comparison of three distinct techniques used for generating captions in an image captioning task. The three evaluation metrics used in this graph are METEOR, ROUGE_L, and CIDEr. It is evident that the RL Agent method outperforms the Greedy and Beam methods across all three evaluation metrics. The Agent method consistently scores higher for all metrics, with the largest difference between the methods seen in the CIDEr metric. The Beam method performs better than the Greedy method for all metrics, with the largest difference seen in the METEOR metric.

It should also be noted that we've obtained good results, despite the fact we've used a downsampled training dataset and have used dimensionality reduced embedding vectors of VGG-16 embeddings so as to improve training efficiency due to limited computing power. The reduction

in METEOR and CIDEr metrics are much less in proportion to the reduction in training dataset size and epochs the model was trained for, when compared to a vanilla CNN - RNN based image caption generator [44], thus proving the effectiveness of the RL agent-based lookahead inference method over the vanilla encoder-decoder frameworks. This is of significant importance, particularly in low-resource situations where data and compute power are both constrained, using the RL-based approach will lead to way better performance than vanilla approaches that require more data to give similar results.

There are still many ways to improve the proposed image captioning model. First and foremost scaling the model training to more training examples and epochs itself will lead to improved performance. Besides using better RL algorithms like A3C to improve the policy and value network training can also be tried. Moreover using better encoder-decoder architectures like full dimensional embeddings (instead of dimensionally reduced embeddings by PCA), and attention and transformer-based decoder architectures too can boost performance a lot. These avenues will be explored in future works.

REFERENCES

[1] Chen, X., & Zitnick, C.L. (2015). Mind's eye: A recurrent visual representation for image caption generation. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2422-2431.

[2] Vinyals, O., Toshev, A., Bengio, S., & Erhan, D. (2014). Show and tell: A neural image caption generator. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 3156-3164.

[3] Karpathy, A., & Fei-Fei, L. (2014). Deep Visual-Semantic Alignments for Generating Image Descriptions. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39, 664-676.

[4] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A.C., Salakhutdinov, R., Zemel, R.S., & Bengio, Y. (2015). Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. International Conference on Machine Learning.

[5] You, Q., Jin, H., Wang, Z., Fang, C., & Luo, J. (2016). Image Captioning with Semantic Attention. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 4651-4659.

[6] Vijayakumar, A.K., Cogswell, M., Selvaraju, R.R., Sun, Q., Lee, S., Crandall, D.J., & Batra, D. (2016). Diverse Beam Search: Decoding Diverse Solutions from Neural Sequence Models. ArXiv, abs/1610.02424.

[7] Jia, X., Gavves, E., Fernando, B., & Tuytelaars, T. (2015). Guiding the Long-Short Term Memory Model for Image Caption Generation. 2015 IEEE International Conference on Computer Vision (ICCV), 2407-2415.

[8] Konda, V.R., & Tsitsiklis, J.N. (1999). Actor-Critic Algorithms. NIPS.

[9] Frome, A., Corrado, G.S., Shlens, J., Bengio, S., Dean, J., Ranzato, M., & Mikolov, T. (2013). DeViSE: A Deep Visual-Semantic Embedding Model. NIPS.

[10] Kiros, R., Salakhutdinov, R., & Zemel, R.S. (2014). Unifying Visual-Semantic Embeddings with Multimodal Neural Language Models. ArXiv, abs/1411.2539.

[11] Ren, Z., Jin, H., Lin, Z.L., Fang, C., & Yuille, A.L. (2015). Multi-Instance Visual-Semantic Embedding. ArXiv, abs/1512.06963.

[12] Ren, Z., Jin, H., Lin, Z.L., Fang, C., & Yuille, A.L. (2016). Joint Image-Text Representation by Gaussian Visual-Semantic Embedding. Proceedings of the 24th ACM international conference on Multimedia.

[13] Lin, T., Maire, M., Belongie, S.J., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C.L. (2014). Microsoft COCO: Common Objects in Context. European Conference on Computer Vision.

[14] Papineni, K., Roukos, S., Ward, T., & Zhu, W. (2002). Bleu: a Method for Automatic Evaluation of Machine Translation. Annual Meeting of the Association for Computational Linguistics.

[15] Lavie, A., & Denkowski, M.J. (2009). The Meteor metric for automatic evaluation of machine translation. Machine Translation, 23, 105-115.

[16] Lin, C. (2004). ROUGE: A Package for Automatic Evaluation of Summaries. Annual Meeting of the Association for Computational Linguistics.

[17] Vedantam, R., Zitnick, C.L., & Parikh, D. (2014). CIDEr: Consensus-based image description evaluation. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 4566-4575.

[18] Fang, H., Gupta, S., Iandola, F.N., Srivastava, R.K., Deng, L., Dollár, P., Gao, J., He, X., Mitchell, M., Platt, J.C., Zitnick, C.L., & Zweig, G. (2014). From captions to visual concepts and back. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 1473-1482.

[19] Farhadi, A., Hejrati, M., Sadeghi, M.A., Young, P., Rashtchian, C., Hockenmaier, J., & Forsyth, D.A. (2010). Every Picture Tells a Story: Generating Sentences from Images. European Conference on Computer Vision.

[20] Kulkarni, G., Premraj, V., Dhar, S., Li, S., Choi, Y., Berg, A.C., & Berg, T.L. (2011). Baby Talk : Understanding and Generating Image Descriptions.

[21] Kuznetsova, P., Ordonez, V., Berg, A.C., Berg, T.L., & Choi, Y. (2012). Collective Generation of Natural Image Descriptions. Annual Meeting of the Association for Computational Linguistics.

[22] Lebret, R., Pinheiro, P.H., & Collobert, R. (2014). Simple Image Description Generator via a Linear Phrase-Based Approach. CoRR, abs/1412.8419.

[23] Krizhevsky, A., Sutskever, I., & Hinton, G.E. (2012). ImageNet classification with deep convolutional neural networks. Communications of the ACM, 60, 84 - 90.

[24] Deng, J., Dong, W., Socher, R., Li, L., Li, K., & Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. 2009 IEEE Conference on Computer Vision and Pattern Recognition, 248-255.

[25] Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. CoRR, abs/1409.1556.

[26] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S.E., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2014). Going deeper with convolutions. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 1-9.

[27] Nguyen, K., Tripathi, S., Du, B., Guha, T., & Nguyen, T.Q. (2021). In Defense of Scene Graphs for Image Captioning. 2021 IEEE/CVF International Conference on Computer Vision (ICCV), 1387-1396.

[28] Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9, 1735-1780.

[29] Chung, J., Gülçehre, Ç., Cho, K., & Bengio, Y. (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. ArXiv, abs/1412.3555.

[30] Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Driessche, G.V., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T.P., Leach, M., Kavukcuoglu, K., Graepel, T., &

Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. Nature, 529, 484-489.

[31] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M.A., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. Nature, 518, 529-533.

[32] Zhu, Y., Mottaghi, R., Kolve, E., Lim, J.J., Gupta, A.K., Fei-Fei, L., & Farhadi, A. (2016). Target-driven visual navigation in indoor scenes using deep reinforcement learning. 2017 IEEE International Conference on Robotics and Automation (ICRA), 3357-3364.

[33] Williams, R.J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learning, 8, 229-256.

[34] Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T.P., Harley, T., Silver, D., & Kavukcuoglu, K. (2016). Asynchronous Methods for Deep Reinforcement Learning. ArXiv, abs/1602.01783.

[35] Sutton, R.S., McAllester, D.A., Singh, S., & Mansour, Y. (1999). Policy Gradient Methods for Reinforcement Learning with Function Approximation. NIPS.

[36] Ranzato, M., Chopra, S., Auli, M., & Zaremba, W. (2015). Sequence Level Training with Recurrent Neural Networks. CoRR, abs/1511.06732.

[37] Valueva, M.V., Nagornov, N.N., Lyakhov, P.A., Valuev, G.V., & Chervyakov, N.I. (2020). Application of the residue number system to reduce hardware costs of the convolutional neural network implementation. Math. Comput. Simul., 177, 232-243.

[38] Freitag, M., & Al-Onaizan, Y. (2017). Beam Search Strategies for Neural Machine Translation. NMT@ACL.

[39] Maćkiewicz, A., & Ratajczak, W. (1993). Principal Components Analysis (PCA). Computers & Geosciences, 19, 303-342.

[40] Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. CoRR, abs/1409.1556.

[41] Bengio, Y., Louradour, J., Collobert, R., & Weston, J. (2009). Curriculum learning. International Conference on Machine Learning.

[42] Jolliffe, Ian T., and Jorge Cadima. "Principal component analysis: a review and recent developments." Philosophical transactions of the royal society A: Mathematical, Physical and Engineering Sciences 374.2065 (2016): 20150202.

[43] Pascanu, Razvan, Tomas Mikolov, and Yoshua Bengio. "On the difficulty of training recurrent neural networks." International conference on machine learning. Pmlr, 2013.

[44] Mathur, Pratistha. "A Survey on Various Deep Learning Models for Automatic Image Captioning." Journal of Physics: Conference Series. Vol. 1950. No. 1. IOP Publishing, 2021.