Image Captioning using Visual Attention

PLS Lokesh Reddy 2100050118 P Hemanth Naidu 2100050117 G Lohith Reddy 2100050054 G Chanikya Prakash 2100050053

Abstract—Automatically predicting image captions is a challenging task. Using a convolutional neural network as an encoder and decoder may not capture semantic relationships effectively. To address this, we incorporated the Bahdanau attention mechanism with the Vector Geometry Group, a pre-trained CNN. The attention mechanism emphasizes semantic information in the caption alongside visual details, achieving a bilingual evaluation understudy score of 40. We validate the use of attention on Flickr8k using pre-trained CNNs

Index Terms—Image captioning; CNN; Bahdanau attention; Natural Language Processing, Transfer learning

1 Introduction

Describing images with coherent English sentences is a challenging task, especially compared to image classification or object recognition. This complexity arises from the need to capture not just objects but also their relationships, attributes, and activities in natural language. We use a unified model that takes an image as input and maximizes the likelihood of generating a descriptive word sequence. Moreover, the above semantic knowledge has to be expressed in a natural language like English, which means that a language model is needed in addition to visual understanding

The human visual system's attention mechanism, a remarkable aspect, allows salient features to dynamically take focus rather than compressing the entire image into a static representation. This is crucial in cluttered images. While using high-level representations is effective for distilling information, it may lead to a loss of useful details. Employing low-level representations can preserve this information, but it requires a robust mechanism to guide the model towards task-relevant information.

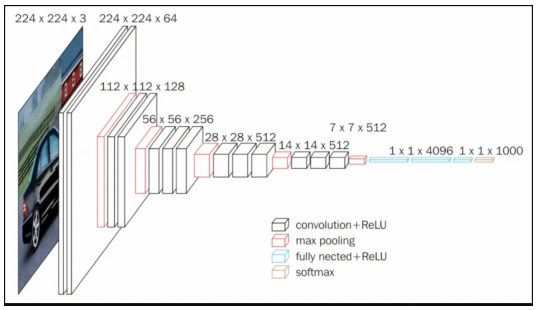
For our course project, we are designing an image captioning system using recurrent neural networks (RNNs) and attention models. RNNs tend to be computationally expensive to train and evaluate, so in practice memory is limited to just a few elements. Attention models help address this problem by selecting the most relevant elements from a larger bank of input data. Such schemes are called attention models by analogy to the biological phenomenon of focusing attention on a small fraction of the visual scene

In this work, we develop a system which extracts features from images using a convolutional neural network (CNN), combines the features with an attention model, and generates captions with an LSTM

2 Architecture

Our model has three main parts: a convolutional neural network (CNN) that extracts features from images, an attention mechanism that weights the image features, and an RNN that generates captions to describe the weighted image features. In the following sections, we will describe each in turn

2.1 CNN as an Encoder



Our model takes a single raw image and generates a caption y encoded as a sequence of 1-of-K encoded words

$$y = \{y_1, \ldots, y_C\}, \quad y_i \in \mathbb{R}^K$$

where K is the size of the vocabulary and C is the length of the caption. We use a VGG19 as our CNN in order to extract a set of feature vectors which we refer to as annotation vectors. The extractor produces L vectors, each of which is a D-dimensional representation corresponding to a part of the image

$$a = \{a_1, \ldots, a_L\}, \quad a_i \in \mathbb{R}^D$$

In order to obtain a correspondence between the feature vectors and portions of the 2-D image, we remove the last 2 layers of VGG19 to extract features from a lower convolutional layer. This allows the decoder to selectively focus on certain parts of an image by selecting a subset of all the feature vectors

2.2 Attention Mechanism

Traditional encoder-decoder systems, especially in image captioning, struggled with fixed-length vectors. The Bahdanau attention mechanism addresses this by allowing the model to focus on specific input vectors, significantly improving performance. This approach, guided by alignment scores, enhances the model's ability to predict accurate captions based on context vectors and relationships between source positions and previously generated target words.

The context vector $\hat{z}_t$ dynamically represents the relevant image part at time t. A mechanism ϕ computes $\hat{z}_t$ from annotation vectors a_i, associated with features from different image locations. The mechanism generates positive weights α_i for each location i, indicating either the probability of it being the focus for the next word or its importance in blending with others. These weights are computed by an attention model f_{att}, using a multilayer perceptron based on the prior hidden state h_{t-1}. This dynamic attention mechanism, introduced by Bahdanau et al. (2014), adjusts the hidden state as the output RNN progresses, determining the network's focus based on the generated word sequence.

$$e_{ti} = f_{\text{att}}(\mathbf{a}_i, \mathbf{h}_{t-1}) \tag{1}$$

$$\alpha_{ti} = \frac{\exp(e_{ti})}{\sum_{k=1}^{L} \exp(e_{tk})} \tag{2}$$

$$\hat{\mathbf{z}}_t = \sum_{i=1}^{L} \alpha_{t,i} \mathbf{a}_i \tag{3}$$

2.3 Decoder

This section explores two variations of the decoder framework, both incorporating the hidden state and attended features from the input image. The key difference lies in the architecture of the LSTM cell and the deep output layer. These components play a vital role in determining the probability of generating the output word. Understanding the intricacies of these variations is essential for a comprehensive grasp of the decoder's functionality.

2.3.1 *Variant1: Modified LSTM cell*

We use a modified LSTM network (Hochreiter & Schmidhuber, 1997) that produces a caption by generating one word at every time step conditioned on a context vector, the previous hidden state and the previously generated words.

We use $T_{s,t} : \mathbb{R}^s \rightarrow \mathbb{R}^t$ to denote a simple affine transformation with learnable parameters

$$\begin{pmatrix} \mathbf{i}_t \\ \mathbf{f}_t \\ \mathbf{o}_t \\ \mathbf{g}_t \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ tanh \end{pmatrix} T_{D+m+n,n} \begin{pmatrix} \mathbf{E}\mathbf{y}_{t-1} \\ \mathbf{h}_{t-1} \\ \hat{\mathbf{z}}_t \end{pmatrix} \tag{4}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t \tag{5}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot tanh(\mathbf{c}_t) \tag{6}$$

Here, $\mathbf{i}_t$, $\mathbf{f}_t$, $\mathbf{c}_t$, $\mathbf{o}_t$, $\mathbf{h}_t$ are the input, forget, memory, output and hidden state of the LSTM, respectively. The vector $\hat{\mathbf{z}}_t \in \mathbb{R}^D$ is the context vector, capturing the visual information associated with a particular input location as explained in above section, $\mathbf{E} \in \mathbb{R}_{m \times K}$ is an embedding matrix. Let m and n denote the embedding and LSTM dimensionality respectively.

The initial memory state and hidden state of the LSTM are predicted by an average of the annotation vectors fed through two separate MLPs (init,c and init,h)

$$\mathbf{c}_0 = f_{init,c}(\frac{1}{L}\sum_{i}^{L} \mathbf{a}_i) \tag{7}$$

$$\mathbf{h}_0 = f_{init,h}(\frac{1}{L}\sum_{i}^{L} \mathbf{a}_i) \tag{8}$$

The output word probability given the LSTM state at time step t, the context vector and the previous word is calculated through a deep output layer

$$p(\mathbf{y}_t|\mathbf{a}, \mathbf{y}_1^{t-1}) \propto exp(\mathbf{L}_o(\mathbf{E}\mathbf{y}_{t-1} + \mathbf{L}_h\mathbf{h}_t + \mathbf{L}_z\hat{\mathbf{z}}_t)) \tag{9}$$

Where $L_o \in \mathbb{R}^{K \times m}, L_h \in \mathbb{R}^{m \times n}, L_z \in \mathbb{R}^{m \times D}$ are learnable parameters intialized randomly

2.3.2 *Varient2: Modified input to LSTM cell*

We utilize the conventional LSTM cell, where the input to the LSTM cell is the $\mathbf{E}\mathbf{y}_{t-1} \bigoplus \mathbf{z}_t$, to get the $\mathbf{h}_t, \mathbf{c}_t$.

The output word probability given the LSTM state at time step t, is calculated through a deep output layer

$$p(\mathbf{y}_t|\mathbf{a}, \mathbf{y}_1^{t-1}) \propto exp(\mathbf{L}_h\mathbf{h}_t), \qquad L_h \in \mathbb{R}^{K \times n} \tag{10}$$

The remainder of the process aligns with variant1.

2.4 Training Procedure

Both model variations underwent training using stochastic gradient descent, with the Adam optimizer, with cross-entropy as our chosen loss metric. Among various convolutional neural networks (CNNs) like AlexNet, ResNet50, ResNet152, and VGG19, our experiments revealed that VGG19 produced the most favorable results.

To create the annotations a_i used by our decoder, we employed VGG19 pretrained on ImageNet without fine-tuning. Specifically, we utilized the 7×7×512 feature map from the fifth convolutional layer. Consequently, our decoder operates on the flattened 49 × 512 (L × D) encoding.

As our implementation requires time proportional to the length of the longest sentence per update, we pad all sentences with a ¡PAD¿ token to achieve uniform length (40). During training, we batch the training data and conduct training sessions on these batches. Our training regimen spans 10 epochs, taking approximately 90 minutes to complete. During training, we use ground truth as input, instead of model output from a prior time step as an input

3 Metrics

Given the huge volume of our data, we cannot possibly show the results for each image. Thus, we need a way to quantify the average accuracy of the system on the whole dataset. There are several metrics by which to judge the quality of machine-produced text, and none without criticism. The metrics we choose are BLEU and METEOR

3.1 BLEU

The BLEU (Bilingual Evaluation Understudy) score is a metric commonly used to evaluate the quality of machine-translated text. It measures the similarity between the machine-generated translation and one or more reference translations. The score ranges from 0 to 1, with 1 indicating a perfect match.

BLEU is calculated by comparing n-grams (contiguous sequences of n items, usually words) in the candidate translation to those in the reference translations. The precision of the candidate's n-grams is computed, and a brevity penalty is applied to account for overly concise translations. The final BLEU score is a geometric mean of the individual n-gram precisions, with the brevity penalty incorporated. Higher BLEU scores generally indicate better translation quality.

$$\text{Brevity Penalty} = \begin{cases} 1 & \text{if } c \geq r \\ e^{(1-r/c)} & \text{if } c < r \end{cases}$$

$$\text{BLEU} = \text{Brevity Penalty} \times \prod_{i=1}^{4} p_i^{w_i}$$

For BLEU-1, w1 = 1 and other weights are zero. For BLEU-2, w1 = w2 = 1/2 and rest are zero. For BLEU-3, w1 = w2 = w3 = 1/3 and w4 = 0. For BLEU-4, w1 = w2 = w3 = w4 = 1/4.

3.2 METEOR

METEOR calculates precision (P) as the ratio of mapped system unigrams to total system unigrams and recall (R) as the ratio of mapped system unigrams to total reference unigrams. Fmean combines these using a harmonic mean with an emphasis on recall, resulting in a concise evaluation formula:

$$F = \frac{(1+\beta^2) \cdot P \cdot R}{\beta^2 \cdot P + R}$$

METEOR considers longer matches by grouping system translation unigrams mapped to reference translation unigrams into chunks. These chunks minimize the number of unigrams and ensure adjacent positions in both translations. Longer n-grams result in fewer chunks; a perfect match yields a single chunk, while no bigram or longer matches lead to one chunk per unigram match. The penalty is calculated using a formula that captures the alignment characteristics, offering a comprehensive assessment beyond unigram matches.

$$\text{Penalty} = 0.5 \left(\frac{\text{\#chunks}}{\text{\#unigrams\_matched}} \right)^3$$

Finally, the METEOR Score for the given alignment is computed as follows:

$$\text{METEOR} = F * (1 - Penalty)$$

4 Results

In this section, we delve into a comprehensive comparison of the experimental outcomes derived from our model's training and evaluation phases. Our analysis comprises multiple facets: a detailed table presenting the BLEU and METEOR scores. Additionally, Figures 1 and 2 depict the performance variations among various pretrained CNNs. Furthermore, Figures 3 and 4 present a comparison between variant1 and variant2 of the decoder framework

Encoder CNN	BLEU-1	BLEU-2	BLEU-3	METEOR
AlexNet	0.34	0.18	0.10	0.26
VGG19(V2)	0.38	0.21	0.13	0.30
ResNet50	0.34	0.18	0.11	0.28
ResNet152	0.36	0.20	0.12	0.28
VGG19(V1)	0.39	0.23	0.15	0.32

TABLE 1: Comparing BLEU-n and METEOR Metrics

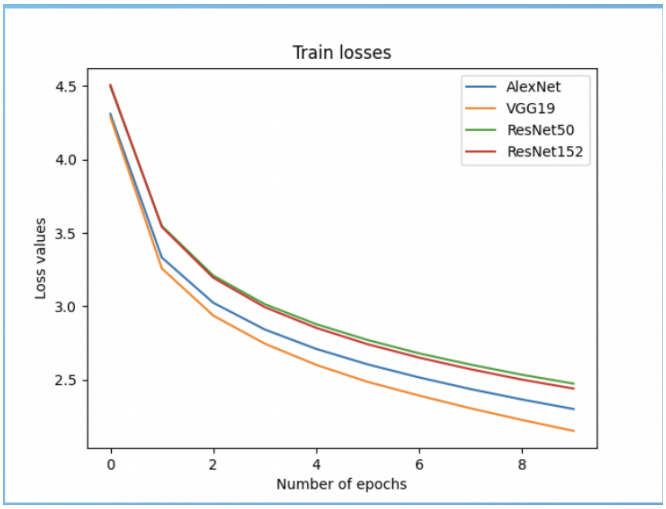


Fig. 1: Train losses vs epochs

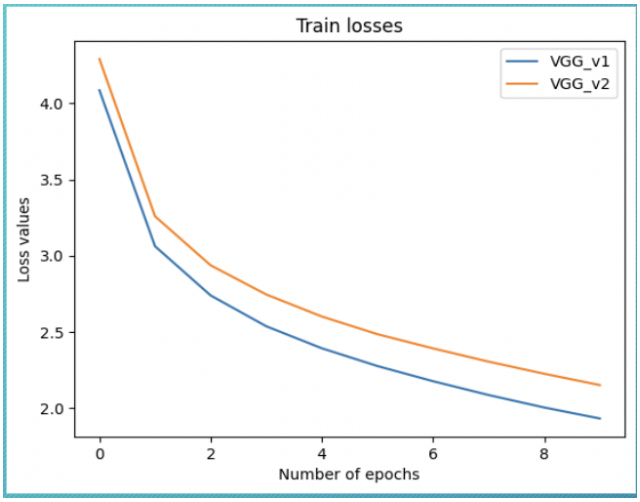


Fig. 3: Train losses vs epochs

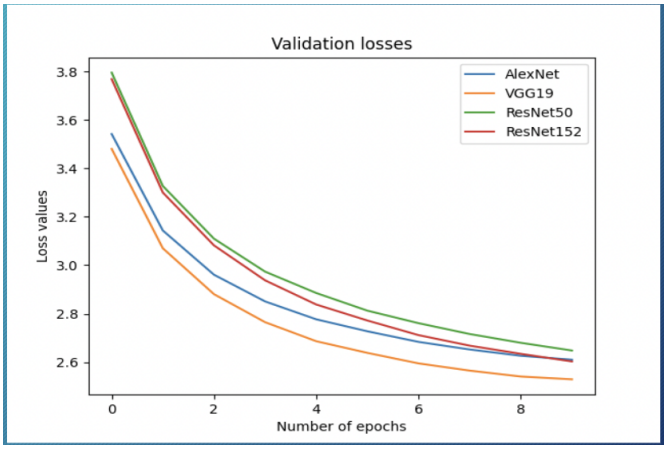


Fig. 2: Validation losses vs epochs

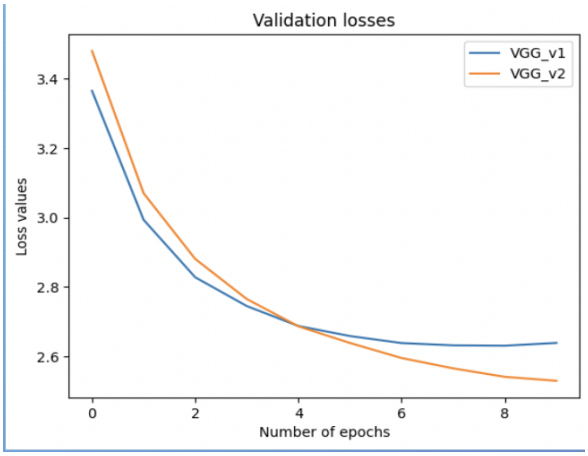


Fig. 4: Validation losses vs epochs

Based on the insights gleaned from Figures 1 and 2, a discernible pattern emerges: the utilization of VGG19 notably yields lower loss rates compared to other models. This observation aligns seamlessly with our predefined objectives and desired outcomes. The lower loss associated with VGG19 signifies its superior performance in minimizing errors during the training and validation phases. This advantage positions VGG19 as a particularly favorable choice within our model architecture, emphasizing its alignment with our overarching goals and objectives

The depicted figures (3 and 4) highlight variant1's gradual manifestation of overfitting towards the training data. This arises due to the iterative utilization of the context vector and prior embeddings for both predicting the subsequent word and computing the LSTM state. This repetition potentially results in an excessive alignment of the model with the nuances of the training data, potentially causing overfitting. Notably, although variant2 demonstrates reduced loss during validation, its BLEU score falls below that of variant1.

5 ILLUSTRATIVE EXAMPLES

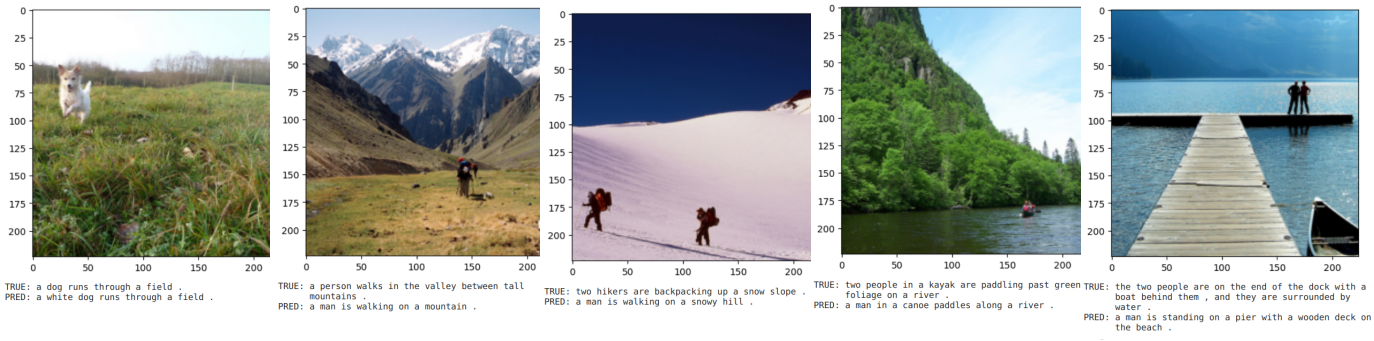


Fig. 5: Correctly classified examples

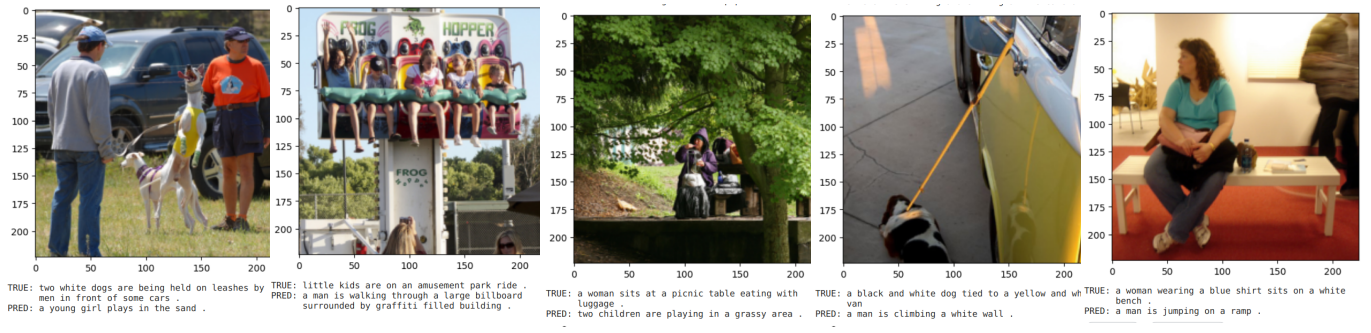


Fig. 6: Wrongly classified examples

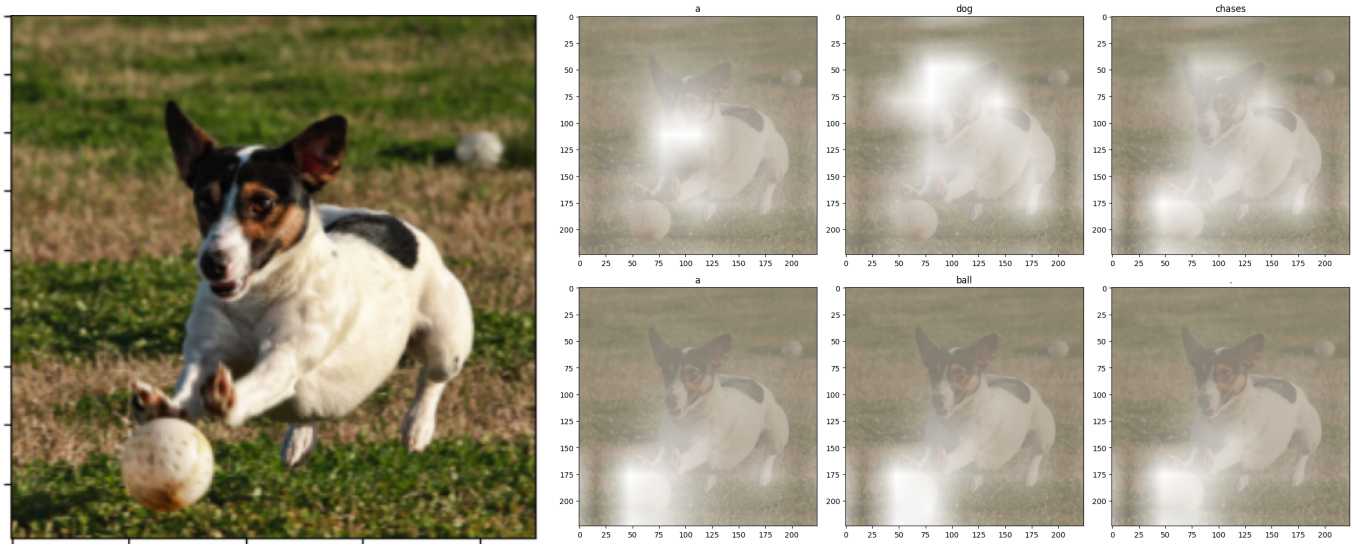


Fig. 7: Attention over time. As the model generates each word, its attention changes to reflect the relevant parts of the image

6 REFERENCES

[1] Show, Attend and Tell: Neural Image Caption Generation with Visual Attention
[2] Generating Image Captions Using Bahdanau Attention Mechanism and Transfer Learning