1 Image Captioning

Image captioning is a field of study that focuses on automatically generating descriptive textual captions for images. It plays a crucial role in bridging the gap between visual understanding and language comprehension. By enabling computers to interpret and describe the contents of images in human-like language, image captioning has significant implications across various domains.

The importance of image captioning lies in its ability to enhance accessibility for individuals with visual impairments. By providing textual descriptions of images, it allows visually impaired individuals to perceive and comprehend visual content that would otherwise be inaccessible to them.

2 Dataset

The Flickr8K dataset is a widely used collection of 8,000 diverse images with five captions per image, making a total of 40,000 captioned images. It serves as a benchmark dataset for image captioning tasks, providing manually annotated captions for training and evaluating models. With its diverse content, human-generated captions, and evaluation metrics like BLEU and METEOR, the Flickr8K dataset has played a crucial role in advancing research and development in the field of image captioning, enabling the exploration of various approaches and techniques to generate accurate and meaningful descriptions for images.

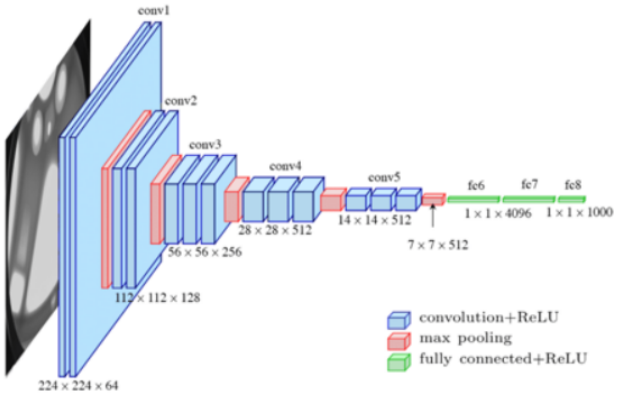
3 CNN and pre-trained models

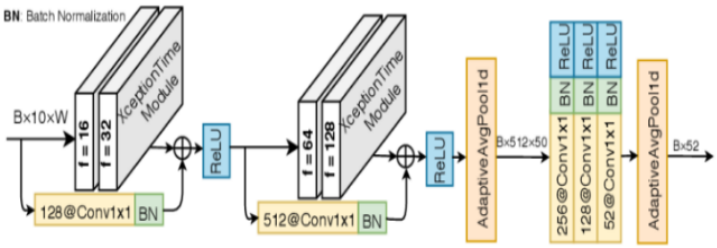
Convolutional Neural Networks (CNNs) are a class of deep learning models widely used in computer vision tasks, including image classification, object detection, and feature extraction. CNNs are specifically designed to process grid-like data, such as images, by using convolutional layers to extract hierarchical features. These layers consist of multiple filters that slide over the input image, performing element-wise multiplications and summations to generate feature maps. The filters capture local patterns and progressively learn more complex representations as the network goes deeper through additional convolutional and pooling layers. This hierarchical feature extraction enables CNNs to effectively learn and discriminate between different visual patterns in images, making them highly effective for various computer vision tasks.

Pretrained models, such as VGG16 and Xception, have revolutionized the field of computer vision by providing pre-trained weights learned on massive datasets like ImageNet. VGG16, proposed by the Visual Geometry Group, consists of 16 convolutional and fully connected layers, with 13 convolutional layers stacked on top of each other. VGG16 is known for its simplicity and uniformity in architecture, making it easy to understand and implement. Xception, on the other hand, is an extension of the Inception architecture that emphasizes the separation of cross-channel correlations and spatial correlations, achieving state-of-the-art performance on various visual recognition tasks. It uses depthwise separable convolutions, which split the standard convolution into separate depthwise and pointwise convolutions, reducing computational complexity while maintaining representation power. By leveraging these pre-trained models, researchers and practitioners can benefit from the learned feature representations and utilize them as a starting point for their own computer vision tasks, saving significant training time and improving performance.

The final layer in a neural network is typically responsible for classification or generating the

desired output. When we remove this final layer, we are left with the feature vector obtained from the penultimate layer. This feature vector represents the high-level abstract features extracted by the neural network during the training process.



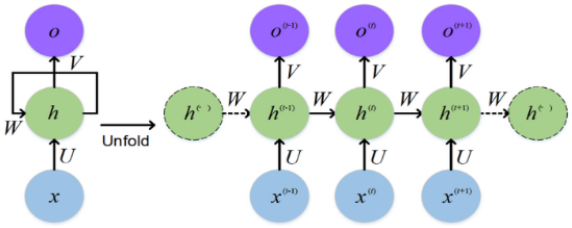


4 RNN

Recurrent Neural Networks (RNNs) have proven to be effective in caption detection tasks, where the goal is to generate descriptive captions for images or videos. RNNs are well-suited for this task due to their ability to model sequential dependencies and maintain context throughout the generation process.

In caption detection, an RNN-based model typically employs an encoder-decoder architecture. The encoder, often a Convolutional Neural Network (CNN), processes the input image or video frames and extracts meaningful visual features. These features are then passed to the decoder, which is an RNN-based language model responsible for generating the caption word by word.

However, traditional RNNs like vanilla RNNs suffer from the vanishing gradient problem. When the gradient signal diminishes as it propagates backward through time, long-term dependencies become difficult to capture. This limitation can affect the quality and coherence of the generated captions. This is where Long Short-Term Memory (LSTM) networks come into play.

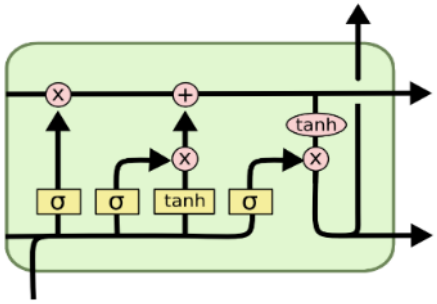


5 LSTM

LSTMs are a variant of RNNs that address the vanishing gradient problem. With their gated memory cell and different gates (input, forget, and output), LSTMs can selectively retain or forget information over multiple time steps, making them more effective in modeling long-term dependencies. LSTMs have been successfully applied in caption detection tasks to generate more accurate and contextually rich captions.

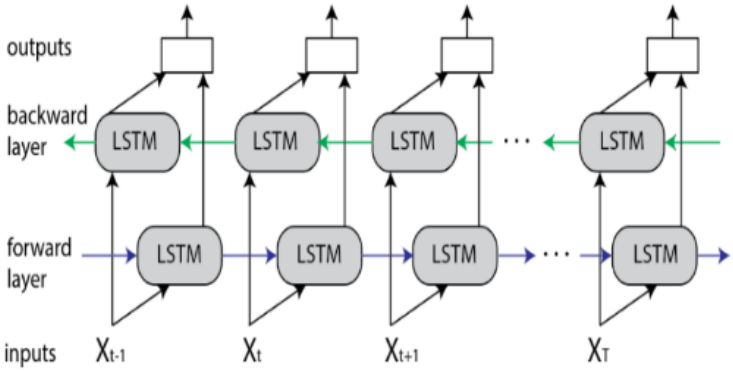
By using LSTMs, the model can capture relevant information over longer sequences and maintain better memory of past inputs, resulting in improved caption quality. LSTMs mitigate the vanishing gradient problem, allowing for better modeling of complex relationships and dependencies in the captions.

In summary, RNNs, particularly with LSTM units, have proven to be powerful tools for caption detection. LSTMs overcome the limitations of traditional RNNs in capturing long-term dependencies, leading to more coherent and contextually accurate generated captions.



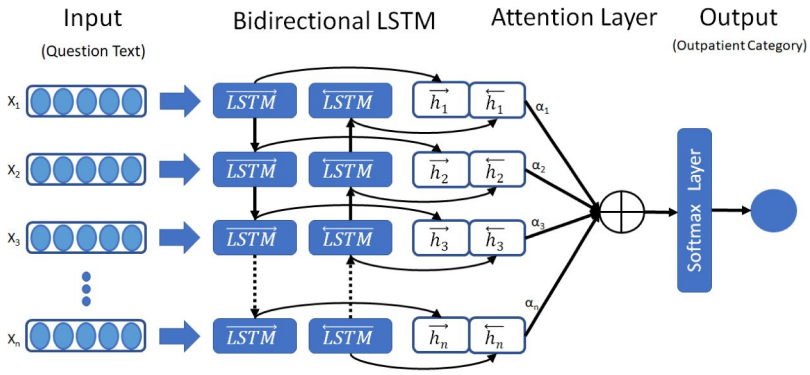
6 Bidirectional LSTMs

Bidirectional LSTMs are an extension of LSTMs that process the input sequence in both forward and backward directions. This bidirectional processing enables the model to capture information from both past and future time steps simultaneously. By considering context from both directions, bidirectional LSTMs have a better understanding of the input sequence, leading to improved representation learning and capturing dependencies that may not be apparent in a unidirectional LSTM. This makes bidirectional LSTMs particularly effective in tasks where context from both past and future contexts is crucial, such as sentiment analysis or speech recognition. The combination of bidirectional processing and the memory capabilities of LSTMs enhances the model's ability to capture long-term dependencies and context, resulting in more accurate and comprehensive representations of sequential data.



7 Attention

LSTMs with attention mechanisms are a powerful combination in sequence-to-sequence tasks such as caption generation or machine translation. LSTMs, or Long Short-Term Memory networks, are a type of recurrent neural network (RNN) architecture designed to capture long-term dependencies in sequential data. They address the vanishing gradient problem and allow for effective modeling of context over extended sequences. By incorporating attention mechanisms, LSTMs can dynamically focus on different parts of the input sequence during the decoding process. Attention enables the model to assign varying levels of importance to different input elements, aligning them with relevant output elements. This improves the model's ability to generate accurate and contextually rich sequences, resulting in better performance in tasks requiring precise alignment and context understanding.



8 Word2Vec and GLoVe

Word2Vec is a widely used word embedding technique that maps words from a vocabulary to dense vector representations in a continuous vector space. It leverages the distributional hypothesis, which suggests that words appearing in similar contexts are likely to have similar meanings. Word2Vec models, such as Skip-gram and Continuous Bag-of-Words (CBOW), are trained on large text corpora to learn word embeddings that encode semantic relationships and capture word similarities. These embeddings allow for efficient and meaningful representation of words in natural language processing tasks, enabling algorithms to operate on words as continuous vectors and capture semantic information.

GloVe (Global Vectors for Word Representation) is another popular word embedding technique that combines global word co-occurrence statistics with a factorization method to generate word representations. GloVe represents words in a vector space where the dot product between word vectors approximates the logarithm of the word co-occurrence probability. By incorporating both local and global information, GloVe embeddings capture semantic relationships between words and offer rich contextual representations. GloVe is trained on large-scale text corpora and has been widely adopted in various natural language processing tasks, such as sentiment analysis, machine translation, and question-answering. Its ability to capture nuanced word relationships makes it a valuable tool for tasks that require a deep understanding of word semantics.

9 Image Feature Extraction

In our project, we utilized the VGG model to extract image features. By loading the VGG model using the VGG class, we were able to access the internal representations or "features" learned by the model. However, since our task was not image classification, we removed the last layer of the loaded model. This last layer is responsible for predicting the class of an image, which was not our primary focus. Instead, we were interested in the intermediate representation of the image right before the classification step, which corresponds to the extracted features. These features capture essential visual information and serve as meaningful representations of the images.

By removing the last layer, we were able to access the image features that the model had learned. These features provided us with valuable insights into the internal representation of the photos. We leveraged these extracted features to enhance our image captioning system, combining them with word embeddings so as to generate accurate and contextually relevant captions for the given images. This approach allowed us to make the most of the pre-trained VGG model's knowledge and adapt it to our specific task of image captioning.

In our image captioning project, we performed preprocessing on the input images using a sequence of transformations. The 'transforms.Compose' function from the 'torchvision.transforms' module allowed us to combine these transformations into a pipeline. The first transformation, 'transforms.Resize', resized the images to a fixed size of (224, 224) pixels. This step ensured that all images were of the same dimensions, as required by the CNN model we utilized. The subsequent transformation, 'transforms.ToTensor', converted the images into tensors, which are the fundamental data structures used in PyTorch. This conversion facilitated efficient computation and integration with the deep learning framework.

To further normalize the images, we applied the 'transforms.Normalize' transformation. By specifying the mean and standard deviation values of [0.485, 0.456, 0.406] and [0.229, 0.224, 0.225], respectively, we normalized the pixel values of the images. This normalization step adjusted the pixel intensities to have a standardized distribution based on the statistics obtained from the ImageNet dataset. Normalizing the images enhanced the stability and convergence of the CNN model during training, promoting better performance in subsequent tasks such as feature extraction and caption generation.By employing this preprocessing pipeline, we ensured that the input images were consistently resized, converted to tensors, and normalized, making them suitable for further processing by the CNN model. This approach allowed us to prepare the images in a standardized manner, enabling effective feature extraction and enhancing the overall performance of our image captioning system.

The final features extracted from an image are represented as a 1-dimensional vector of 4,096 elements. These features capture the essential visual information and characteristics of the image in a compact representation.

10 Merge Architecture

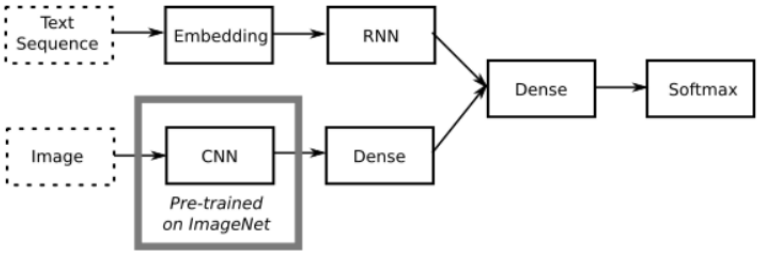
The process of fusing textual information from a language model with visual information from an image is referred to as merged architecture. The objective is to provide a combined representation that conveys the connection between the image and the narrative that goes with it.

The merge architecture typically involves the following steps:

- **Image Encoding:** The input image is first passed through a convolutional neural network (CNN), such as VGG or Xception. The CNN extracts visual features from the image, resulting in a fixed-length vector representation called the image embedding or image feature vector.
- **Text Encoding:** Using a language model, the input caption or textual description of the image is encoded. Text encoding is frequently carried out by Recurrent Neural Networks (RNNs), such as LSTM or GRU, in image captioning models. Each time a time step is reached, the RNN updates its hidden state as it reads the input caption word by word.
- **Merge Operation:** The merge operation combines the visual and textual information to provide a joint representation after the image and text have been encoded separately. Concatenation, element-wise addition or multiplication, or more complex attention mechanisms are some techniques for combining the two modalities.
- **Decoding and Caption Generation:** The combined representation is then fed through a decoder, which is commonly an RNN. Based on the combined representation and the previously created words, the decoder creates the caption word by word. This procedure keeps on until a predicted end-of-sentence token or the predetermined maximum caption length is achieved.

The merge architecture allows the model to learn the correspondence between visual and textual information, enabling it to generate meaningful and coherent captions for images. By combining both modalities, the model can leverage the visual cues from the image and the semantic knowledge from the language model to produce accurate and descriptive captions.

It's important to note that there have been various advancements in image captioning models, including the incorporation of attention mechanisms, transformer-based architectures, and reinforcement learning techniques. These advancements further improve the merged architecture and enhance the model's ability to attend to relevant image regions and capture complex relationships between the visual and textual information.



11 Code Implementation of Merge Architecture

1. **Extracting image features:** We are extracting image features from Flicker8k image Dataset using the VGG16 model and storing the extracted features in a pickle file. The VGG16 model, a pre-trained deep learning model for image classification, is loaded and restructured to remove the last classification layer. Each image in the specified directory is loaded, preprocessed, and passed through the VGG16 model to extract its high-level features.

The features are then stored in a dictionary with the image ID as the key. Finally, the extracted features are saved to a pickle file for future use or analysis. This approach allows for the efficient extraction of meaningful image representations that can be utilized in image captioning.

```
def extract_features(directory):
    # load the model
    model = VGG16()
    # re-structure the model
    model = Model(inputs=model.inputs, outputs=model.layers[-2].output)
    print(model.summary())
    features = dict()
    for name in listdir(directory):
        filename = directory + '/' + name
        image = Image.open(filename)
        image = image.resize((224, 224))
        image = np.array(image)
        image = image.reshape((1, image.shape[0], image.shape[1], image.shape[2]))
        image = preprocess_input(image)
        feature = model.predict(image, verbose=0)
        image_id = name.split('.')[0]
        features[image_id] = feature
        print('>%s' % name)
    return features
```

2. **Preparing Text Data:** In the task of image captioning, the dataset contains multiple descriptions for each photograph, and it is necessary to perform some minimal cleaning on the text to ensure its suitability for analysis and modeling.

   - **Cleaning the Text Data:** Cleaning the description text is a crucial step in reducing the complexity and size of the vocabulary to be used in subsequent processes.

     The following steps were applied to clean the text data:

     (a) Lowercasing
     (b) Punctuation Removal
     (c) One-Character Word Removal
     (d) Number-Containing Word removal

   - **Tokenization:** The descriptions in the dataset are already tokenized, meaning they have been split into individual words or tokens. Tokenization enables further analysis and modeling by allowing the text to be processed and understood on a per-word basis. This step facilitates subsequent operations such as building vocabularies and training language models.
   - **Vocabulary Reduction:** A essential step to control computational complexity and enhance model performance is to reduce the vocabulary size. We may drastically lower the number of unique words in the dataset by cleaning the text data and deleting unnecessary or noise-producing terms. In turn, this makes later tasks easier, such as language modeling and helps to concentrate on the words that are the most instructive and relevant.

3. **Defining and Fitting The Model:** We will define and train a deep learning model using the provided training dataset. The process involves loading the data, defining the model architecture, and fitting the model to the data.

- **Defining the Model:** We will define our model based on Merge Architecture discussed above. The model consists of three components: a photo feature extractor, a sequence processor, and a decoder. The input to the photo feature extractor is a 4,096-dimensional vector representing the photo features. The output is processed by a Dense layer, resulting in a 256-dimensional photo representation.

  The sequence processor expects input sequences of a fixed length (34 words). It uses an Embedding layer with masking capabilities to handle padded values. The embedded sequences are then passed through an LSTM layer with 256 memory units.

  Both the photo feature extractor and sequence processor output 256-dimensional vectors. Regularization is applied in the form of 50% dropout.

  The decoder merges the vectors from both input models using addition. The merged vector is further processed by a Dense layer with 256 neurons, followed by a final output Dense layer that predicts the next word using softmax activation.
- **Fitting the Model:** The problem of overfitting, in which the model gets overly specialized in learning the training data and performs badly on fresh, unknown data, must be addressed during the training phase. The performance of the model is regularly tracked on a separate development dataset in order to mitigate this. The model's proficiency on the development dataset is assessed at the conclusion of each epoch (iteration over the whole training dataset). The entire model, including its architecture and learned weights, is stored in a file if performance is improved over the previous epoch.

  We can determine the epoch in which the model performs best by tracking its proficiency on the development dataset. The model saved during the epoch is chosen as the final model at the conclusion of the whole training procedure. The balance between learning from the training dataset and successfully generalizing to new data is best represented by this model.

  Using the saved model with the best skill on the training dataset ensures that the final model possesses the highest level of performance achieved during training. It provides us with a reliable and effective model that can be used for generating accurate captions for new images.

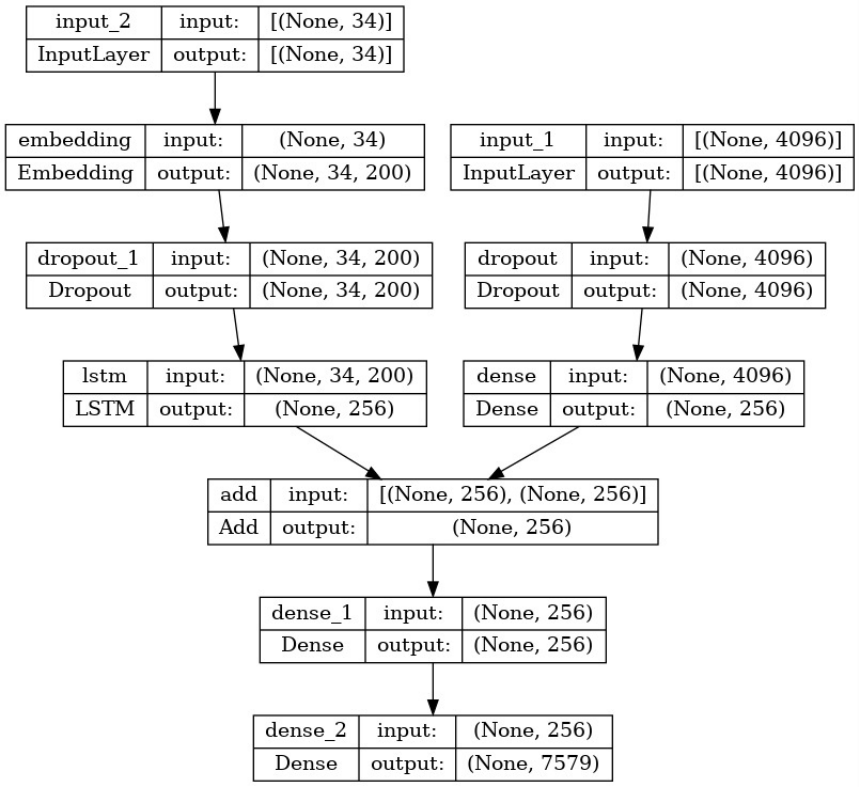
```
model = captioning_model(vocab_size, embedding_dim, max_length, embedding_matrix)

num_epochs = 15
steps_per_epoch = len(train_descriptions)
loss_history = []
epoch_history = []

for i in range(num_epochs):
    generator = generate_data(train_descriptions, train_ImageFeatures, caption_tokenizer, max_length, vocab_size)
    history = model.fit_generator(generator, epochs=1, steps_per_epoch=steps_per_epoch, verbose=1)
    model.save('VGG16_Word2Vec_model' + str(i) + '.h5')
    loss_history.append(history.history['loss'][0])
    epoch_history.append(i+1)
```

12 Different Architectures

For the caption detection system, we initially employed a simple baseline model. However, to improve performance, a modified architecture was introduced. The modified model incorporates

several key enhancements, including an attention mechanism that dynamically focuses on relevant image/video regions during caption generation. The architectural modifications result in improved caption quality by effectively aligning information, incorporating contextual understanding, and enhancing training efficiency through normalization techniques. **Simple Baseline:** The simple baseline architecture for image captioning consists of several components:

- **CNN Architecture:** VGG16 - The Convolutional Neural Network (CNN) component, specifically VGG16, is used as the photo feature extractor. VGG16 is a popular pre-trained CNN model that has been trained on a large dataset called ImageNet. It is capable of extracting high-level visual features from input images.
- **LSTM:** Bidirectional LSTM - The caption generation component utilizes a Long Short-Term Memory (LSTM) recurrent neural network. In this case, a bidirectional LSTM is used, which processes the input sequence in both forward and backward directions. LSTMs are effective in capturing sequential information and are commonly used in natural language processing tasks.
- **Word Embedding:** Word2Vec - Word embedding is used to represent words as dense vectors in a continuous space. In this architecture, the Word2Vec algorithm is employed to learn word embeddings from a large corpus of text data. Word embeddings capture semantic relationships between words and provide a more meaningful representation for the LSTM to generate captions.

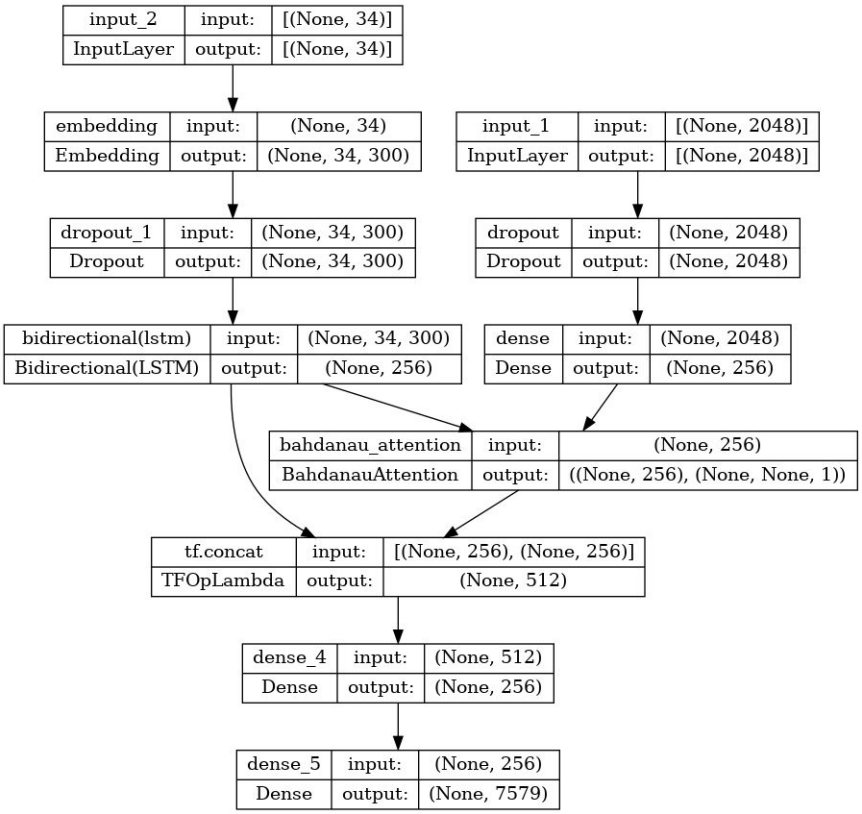


By combining the VGG16 CNN for image feature extraction, a bidirectional LSTM for sequential processing, and Word2Vec word embeddings, the simple baseline architecture aims to generate captions for images that capture both the visual content and the context of the image.

Modified Baseline: The modified baseline architecture for image captioning includes the following components:

- **CNN Architecture:** Xception - The Convolutional Neural Network (CNN) component uses the Xception model as the photo feature extractor. Xception is a powerful pre-trained CNN architecture that is known for its exceptional performance in image classification tasks. It has been trained on a large-scale dataset and can extract high-level visual features from input images effectively.
- **LSTM:** Bidirectional LSTM + Attention Mechanisms - The caption generation component utilizes a Bidirectional LSTM (Long Short-Term Memory) recurrent neural network. The bidirectional LSTM processes the input sequence in both forward and backward directions, allowing it to capture contextual information from both past and future tokens. Additionally, attention mechanisms are incorporated into the LSTM, enabling the model to focus on relevant parts of the input sequence when generating captions.
- **Word Embedding:** GloVe - Word embedding is employed to represent words as dense vectors in a continuous space. In this architecture, the Global Vectors for Word Representation (GloVe) algorithm is used to learn word embeddings. GloVe embeddings are based on the co-occurrence statistics of words in a large corpus of text and capture semantic relationships between words.

By utilizing the Xception CNN for image feature extraction, a bidirectional LSTM with attention mechanisms for contextual caption generation, and GloVe word embeddings for meaningful word representations, the modified baseline architecture aims to enhance the image captioning performance by leveraging advanced neural network components and pre-trained models.



The Modified Baseline model offers several advantages over the simple baseline model:

- **Enhanced Vision Model:** The modified baseline replaces the VGG 16 vision model with Xception. Xception is a more advanced convolutional neural network architecture known for its improved performance in image recognition tasks. By utilizing Xception, the modified baseline can extract more detailed and accurate visual features, leading to better caption generation.
- **Bidirectional LSTM with Attention Mechanisms:** Unlike the simple baseline's LSTM, the modified baseline incorporates bidirectional LSTM layers along with attention mechanisms in the decoder. The attention mechanisms help the model focus on relevant parts of the image or video frames, aligning the visual features with the corresponding words during caption generation. This attention mechanism enhances the model's ability to generate more informative and contextually meaningful captions.
- **Improved Word Representation:** While the simple baseline uses Word2Vec for word representation, the modified baseline employs GloVE (Global Vectors for Word Representation). The GloVE is a more advanced and comprehensive word embedding technique that captures semantic relationships and word similarities more effectively. By utilizing GloVE, the modified baseline benefits from richer word representations, allowing for more accurate and nuanced caption generation.

Overall, the modified baseline model leverages an advanced vision model, bidirectional LSTM with attention mechanisms, and improved word representation to enhance caption generation performance. These enhancements result in more accurate and contextually relevant captions compared to the simple baseline model.

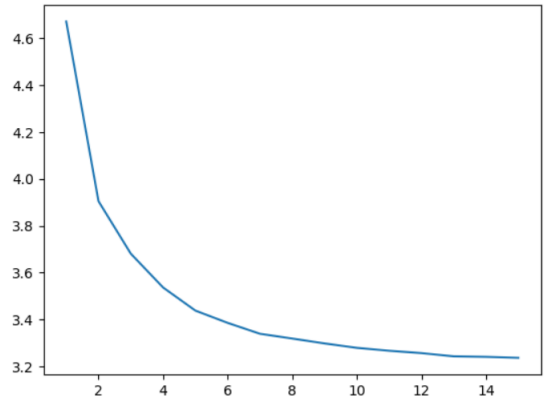
13 Results

- **Simple Baseline**

  For each epoch, the code utilizes a data generator to create training data by combining image features and caption sequences. The model is then trained using the generated data for one epoch. This process is repeated for the specified number of epochs. The code tracks the loss values for each epoch, allowing for evaluation of the model's performance over time.

```
6000/6000 [==============================] - 66s 9ms/step - loss: 4.6701
6000/6000 [==============================] - 54s 9ms/step - loss: 3.9045
6000/6000 [==============================] - 54s 9ms/step - loss: 3.6811
6000/6000 [==============================] - 55s 9ms/step - loss: 3.5370
6000/6000 [==============================] - 56s 9ms/step - loss: 3.4385
6000/6000 [==============================] - 55s 9ms/step - loss: 3.3863
6000/6000 [==============================] - 54s 9ms/step - loss: 3.3399
6000/6000 [==============================] - 54s 9ms/step - loss: 3.3195
6000/6000 [==============================] - 55s 9ms/step - loss: 3.2987
6000/6000 [==============================] - 54s 9ms/step - loss: 3.2799
6000/6000 [==============================] - 55s 9ms/step - loss: 3.2675
6000/6000 [==============================] - 55s 9ms/step - loss: 3.2575
6000/6000 [==============================] - 54s 9ms/step - loss: 3.2437
6000/6000 [==============================] - 55s 9ms/step - loss: 3.2415
6000/6000 [==============================] - 54s 9ms/step - loss: 3.2374
```

  The plot of Loss Vs Epoch Number(Loss value on the y-axis)

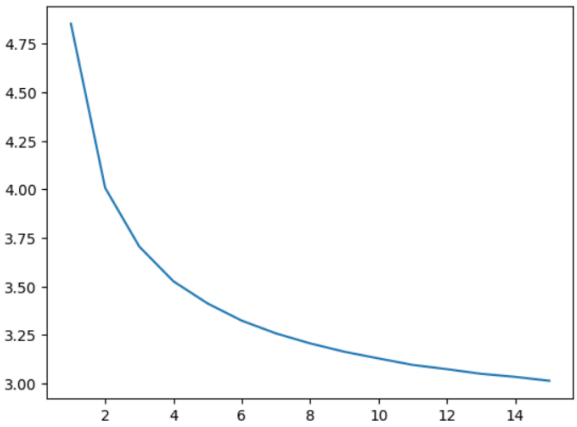


- **Modified Baseline**

  The Loss value for every each on the Modified Baseline:

```
6000/6000 [==============================] - 87s 13ms/step - loss: 4.8539
6000/6000 [==============================] - 77s 13ms/step - loss: 4.0088
6000/6000 [==============================] - 79s 13ms/step - loss: 3.7059
6000/6000 [==============================] - 76s 13ms/step - loss: 3.5267
6000/6000 [==============================] - 78s 13ms/step - loss: 3.4126
6000/6000 [==============================] - 76s 13ms/step - loss: 3.3240
6000/6000 [==============================] - 79s 13ms/step - loss: 3.2585
6000/6000 [==============================] - 77s 13ms/step - loss: 3.2069
6000/6000 [==============================] - 79s 13ms/step - loss: 3.1640
6000/6000 [==============================] - 78s 13ms/step - loss: 3.1295
6000/6000 [==============================] - 77s 13ms/step - loss: 3.0961
6000/6000 [==============================] - 77s 13ms/step - loss: 3.0742
6000/6000 [==============================] - 78s 13ms/step - loss: 3.0503
6000/6000 [==============================] - 77s 13ms/step - loss: 3.0345
6000/6000 [==============================] - 79s 13ms/step - loss: 3.0142
```

  The plot of Loss Vs Epoch Number(Loss value on the y-axis)



14 Evaluation

BLEU Score: The BLEU (Bilingual Evaluation Understudy) score is a metric commonly used to evaluate the quality of the machine-generated text, such as machine translation or image captioning. It measures the similarity between a generated text and one or more reference texts (typically human-generated) by computing the overlap of n-grams (contiguous sequences of n words) between the generated and reference texts.

The BLEU score ranges from 0 to 1, where a higher score indicates better similarity to the reference texts. It is a widely adopted metric in natural language processing tasks and provides a quantitative measure of how well a generated text aligns with human-generated references.

The last epoch represents the point at which the model has undergone training for the maximum number of epochs specified. By the last epoch, the model has potentially converged to its optimal state, meaning it has learned the patterns and features in the data necessary for generating accurate captions. So, To assess the performance of the image captioning model, we will calculate the BLEU (Bilingual Evaluation Understudy) scores for the last epoch.

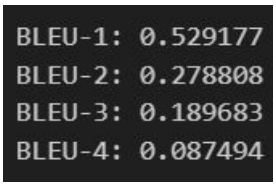


Figure 1: BLEU scores for Simple Baseline Model

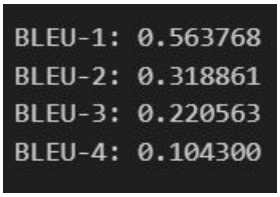


Figure 2: BLEU scores for Modfied Baseline Model

Several significant improvements have been incorporated into the updated baseline model, which results in higher BLEU values than the simple baseline. First off, by using the Xception CNN architecture, the model is able to extract more complex and in-depth image features, resulting in a richer representation of the visual material. As a result, the photos are better understood as a whole, leading to captions that are more accurate and pertinent given the context.

Second, adding bidirectional LSTM and attention processes enhances the model's capacity to generate captions. The bidirectional LSTM improves the model's comprehension of the sequential character of language by enabling it to record relationships in both forward and backward directions. When creating captions, the model may focus on key areas of the image thanks to the attention processes, aligning

Additionally, the semantic representation of words in the captions is improved by the use of GloVe word embeddings. The model better comprehends the semantic relationships between words by utilizing pre-trained word embeddings that encapsulate these associations. As a result, captions

show a better level of semantic coherence and significance in addition to being syntactically correct.

These improvements to the updated baseline model work together to improve its BLEU values. The generation of captions that are more closely aligned with the reference captions is made possible by the model's capacity to extract more precise image features, capture contextual dependencies through bidirectional LSTM, incorporate attention mechanisms for focused caption generation, and utilize semantic word embeddings.

15 Image Captioning Predictions of both Baseline models

When comparing the image captions generated by the simple baseline and modified baseline architectures, it becomes evident that the modified baseline outperforms the simple baseline in terms of caption quality. The modified baseline incorporates Architecture improvements that contribute to its superior performance.

Caption 1

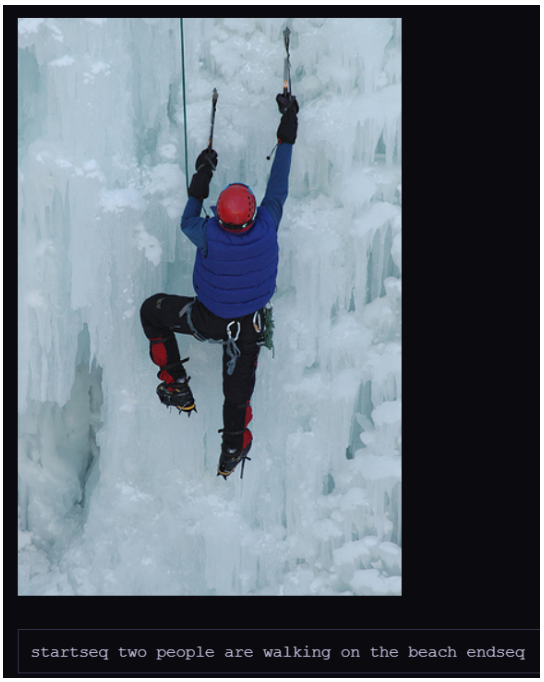


Figure 3: Image caption generated by Simple Baseline Model

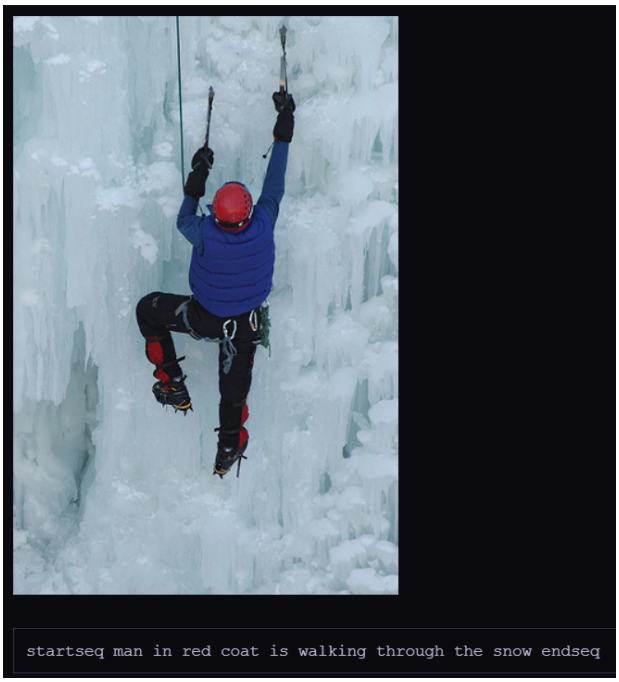


Figure 4: Image caption generated by Modified Baseline Model

Based on the provided image captions generated by two models, it can be observed that the second model, which produced the caption "Man in red coat is walking through the snow," yielded better results compared to the first model's caption of "Two people are walking on the beach." Here are some observations:

1. **Contextual relevance:** The second model's caption appears to be more contextually relevant to the image. It accurately describes the visual elements captured in the image, mentioning the presence of a man wearing a red coat and the snowy environment. This indicates

that the model has successfully captured important visual cues and translated them into a meaningful caption.

2. **Specificity and detail:** The second model's caption provides more specific details about the scene, mentioning the man and his red coat, as well as the snowy setting. This level of detail adds richness to the generated caption, making it more informative and descriptive.

3. **Language fluency:** The second model's caption is grammatically correct and fluent, indicating that the model has a good understanding of sentence structure and language usage. This contributes to the overall quality and readability of the caption.

4. **Visual understanding:** The second model demonstrates a better understanding of the visual content in the image. It accurately recognizes the key elements, such as the person, their attire, and the surrounding environment. This suggests that the model has learned to extract meaningful features from images and generate captions that align with the visual context.

Caption 2

Figure 5: Simple Baseline Model

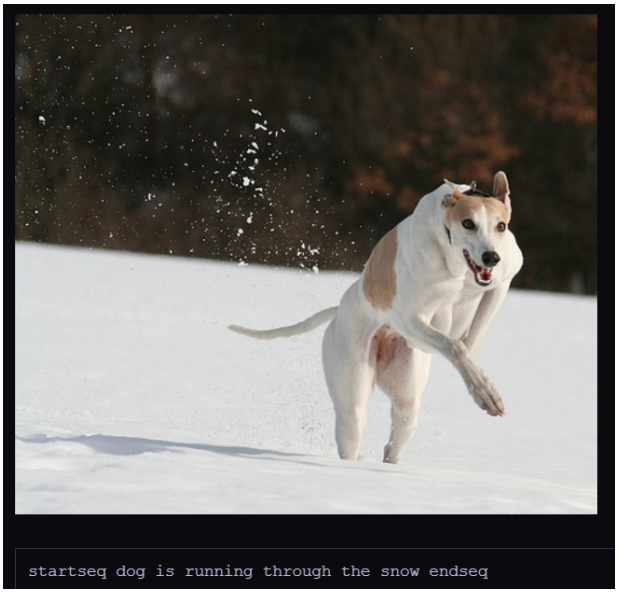


Figure 6: Modified Baseline Model

Observations for the second image, where Model 2 generated the caption "Dog is running through the snow" compared to Model 1's caption "Two dogs are playing in the water," indicate that Model 2 performed better. Model 2's caption accurately describes the visual scene, mentioning a dog running in the snow, while Model 1's caption is less accurate, mentioning dogs playing in the water. This suggests that Model 2 captures the key elements of the image more effectively, including the specific action and environment, resulting in a more precise and contextually relevant caption.

Caption 3

Observations for the second image, where Model 2 generated the caption "Black dog is running through the water" compared to Model 1's caption of "Two dogs are playing in the water," indicate

that Model 2 performed better. Model 2's caption is more specific, mentioning a black dog running through the water, which aligns closely with the visual scene. In contrast, Model 1's caption is more general, mentioning two dogs playing without specifying any particular characteristics or actions. Model 2's caption demonstrates a higher level of detail and accuracy, capturing the color of the dog and its specific activity, resulting in a more refined and contextually relevant description.

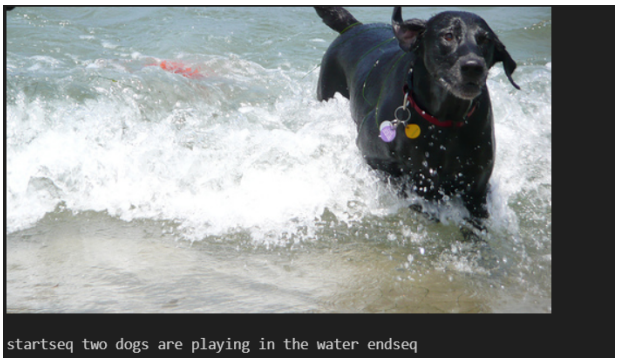


Figure 7: Simple Baseline Model

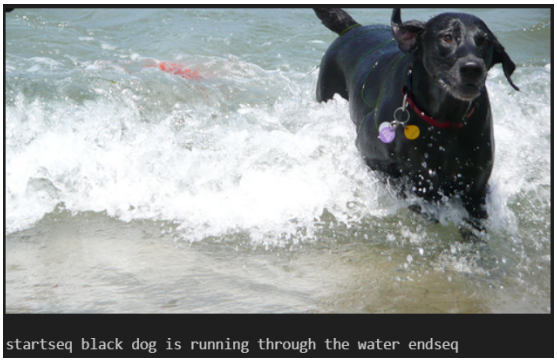


Figure 8: Modified Baseline Model

Conclusions: Upon analyzing the remaining captions, we can observe that both models struggled to generate accurate and contextually relevant descriptions for the images. Model 1 and Model 2 exhibited varying levels of inaccuracies in their generated captions. While Model 2 generally produced captions that were closer to the actual image content compared to Model 1, there were still instances where both models failed to capture key details or misinterpreted the scene. These observations highlight the challenges involved in image captioning tasks and the need for further advancements in the model's ability to generate precise and meaningful descriptions that align closely with the visual content.

Caption 4

Figure 9: Simple Baseline Model

Figure 10: Modified Baseline Model

Caption 5

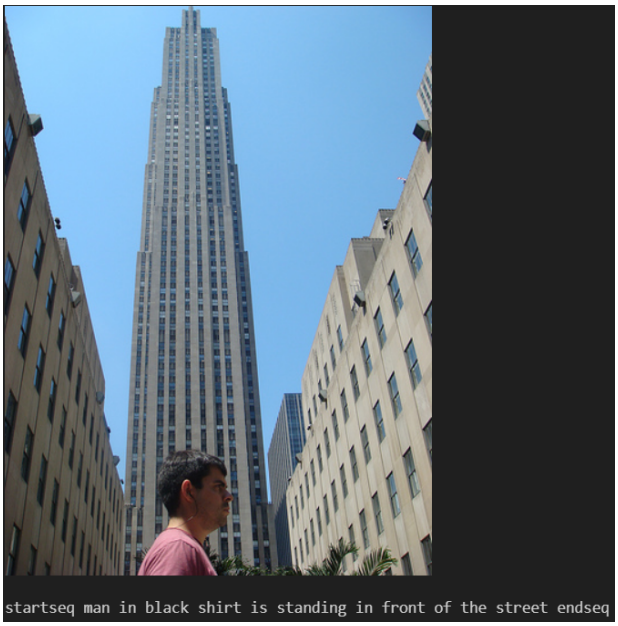


Figure 11: Simple Baseline Model

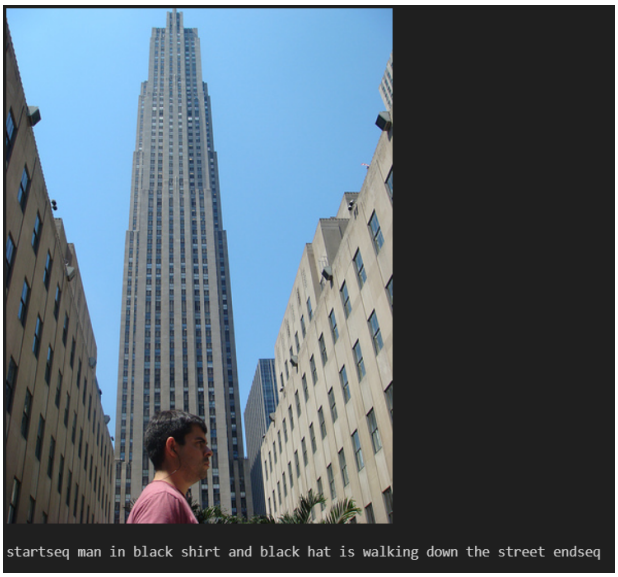


Figure 12: Modified Baseline Model