**Image Caption Generator**

**Project Report**

**Anoushka Saha**

**Introduction**

An image caption generator is a technology or system that automatically generates descriptive captions or textual explanations for images. It involves the convergence of two critical fields: computer vision and natural language processing. Typically, image captioning systems adopt an encoder-decoder framework where an initial image is encoded to create an intermediary representation that captures the image's content. Subsequently, this representation is decoded to produce a coherent and descriptive sequence of text that encapsulates the image's content and context.

To perform Image Captioning we will require two deep learning models combined into one for the training purpose:

* CNNs extract the features from the image of some vector size aka the vector embeddings. The size of these embeddings depends on the type of pretrained network being used for the feature extraction
* LSTMs are used for the text generation process. The image embeddings are concatenated with the word embeddings and passed to the LSTM to generate the next word.

In this project we use the flickr8k dataset which consists of one Image folder and a text file containing the captions for the images.

**Data Pre-Processing**

The sentences in the Caption file are converted into lowercase, special characters and numbers present in the text are removed, extra spaces and single characters are removed. A starting and an ending tag to the sentences are added to indicate the beginning and the ending of a sentence.

**Tokenization and encoded representation**

Machines are not familiar with complex English words so, to process model’s data they need a simple numerical representation. That’s why every word of the vocabulary is mapped to a separate unique index value. They are then encoded in a one hot representation. These encodings are then passed to the embeddings layer to generate word embeddings.

**DenseNet-201 Architecture**

DenseNet-201 is a deep convolutional neural network architecture that belongs to the family of DenseNet models. DenseNet, short for "Densely Connected Convolutional Networks," is designed to address some of the challenges associated with training deep networks and promote better feature reuse and information flow throughout the network. A brief description of DenseNet Architecture is as follows:

1. **Dense Connections:** DenseNet introduces the concept of "dense connections" between layers. In DenseNet, each layer receives input from all preceding layers.
2. **Dense Blocks:** The key building block of DenseNet is the "dense block." A dense block consists of multiple layers grouped together.
3. **Bottleneck Layers:** Within dense blocks, bottleneck layers are often used to reduce the number of input channels before the densely connected layers.
4. **Transition Layers:** DenseNet also incorporates "transition layers" between dense blocks. These transition layers perform two functions: they down sample the spatial dimensions of the feature maps and reduce the number of feature channels.
5. **Global Average Pooling and Classification:** Towards the end of the network, global average pooling is commonly used to convert the spatial information into a single vector. This vector is then connected to a fully connected layer for final classification.

DenseNet-201 specifically refers to a variant of DenseNet where the number "201" indicates the total number of layers in the network, including convolutional layers, batch normalization layers, activation functions, pooling layers, and fully connected layers.

DenseNet architectures have gained popularity due to their efficiency in training, reduced risk of vanishing gradients, and strong performance on various computer vision tasks, including image classification, object detection, and segmentation. However, it's important to note that as the architecture becomes deeper, it also becomes more computationally intensive, which can impact training time and resource requirements.

**Image Feature Extraction**

DenseNet 201 Architecture is used to extract the features from the images. Since the Global Average Pooling layer is selected as the final layer of the DenseNet201 model for our feature extraction, our image embeddings will be a vector of size 1920.

**Data Generation**

Since Image Caption model training like any other neural network training is a highly resource utilizing process we cannot load the data into the main memory all at once, and hence we need to generate the data in the required format batch wise. The inputs will be the image embeddings and their corresponding caption text embeddings for the training process. The text embeddings are passed word by word for the caption generation during inference time.

**LSTM Architecture**

Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN) architecture designed to capture sequential patterns and dependencies in data. LSTMs are particularly well-suited for tasks involving sequences, such as natural language processing, speech recognition, time series analysis, and more. A brief description of LSTM Architecture is as follows:

1. **Sequential Data Handling:** LSTMs are designed to work with sequences of data, where the order of elements matters.
2. **Memory Cell:** The core component of an LSTM is the memory cell, which allows the network to maintain and update information over long sequences.
3. **Gates:** LSTMs use three types of gates to regulate the flow of information: the input gate, forget gate, and output gate. These gates are neural network layers that control how much information to read from the input, how much to forget from the previous state, and how much to reveal to the output.
4. **Input Gate:** The input gate controls which values from the input sequence should be added to the cell state. It uses a sigmoid activation function to decide which values to update.
5. **Forget Gate:** The forget gate determines which information from the previous cell state should be discarded. It uses a sigmoid activation function to produce values between 0 and 1, indicating how much to retain or forget.
6. **Output Gate:** The output gate regulates how much of the cell state should contribute to the output. It also uses a sigmoid activation function to control this process.
7. **Cell State Update:** The cell state is updated based on the input gate and the forget gate. New information is combined with existing information, allowing the LSTM to remember relevant information over long sequences.
8. **Hidden State:** The hidden state is the output of the LSTM cell for a given time step. It's influenced by the cell state and is computed based on the output gate.

LSTMs have the ability to learn and capture long-range dependencies in sequential data, making them effective for tasks like language modeling, machine translation, sentiment analysis, and more. However, as sequences become very long, LSTMs can still face challenges with learning and memory retention, leading to the development of more advanced architectures like Gated Recurrent Units (GRUs) and attention mechanisms, which aim to improve on certain aspects of the LSTM design.

**Modelling**

The image embedding representations are concatenated with the first word of sentence, that is, ‘starseq’ and passed to the LSTM network. The LSTM network starts generating words after each input thus forming a sentence at the end.

**Model Modification**

A slight change has been made in the original model architecture to push the performance. The image feature embeddings are added to the output of the LSTMs and then passed on to the fully connected layers. This slightly improves the performance of the model.

The dataset is divided into training set containing 85% of the data and a Validation Set containing 15%. The model is then trained. On plotting the loss curve, it is clear that the model is overfitted, possibly due to less amount of data.

**Conclusion**

We have taken 15 random samples for caption prediction. We see that some of the images are wrongly captioned and there is some redundant caption generation, for example, ‘Dog running through the water, over-usage of blue shirt for any other colored cloth.

The model performance can be further improved by training on more data and using Attention mechanism so that the model can focus on relevant areas during the text generation. Also one can leverage the interpretability of the attention mechanism to understand which areas of the image leads to the generation of which word.