IMAGE CAPTION GENERATOR A Deep Learning approach using LSTM and CNN Network

Rama chandra Rao Gunnam, Rowan Ronald Undekoti

gg6501@srmist.edu.in, uu0712@srmist.edu.in

CINTEL Department, SRM University

Mahatma Gandhi Rd, Potheri, SRM Nagar Kattankulathur, Tamil Nadu 603203

Abstract—This paper proposes a new image caption generator model that achieves state-of-the-art results. It can be used for various applications, including generating captions for social media images, helping visually impaired people, and generating captions for product images and search engines.

Keywords— image captioning, deep learning, convolutional neural networks (CNNs), long short-term memory (LSTM) networks, natural language processing (NLP), computer vision, visual attention, transformed, reinforcement learning, contrastive learning.

I. Problem Statement

The problem statement for image caption generation using CNNs and LSTMs is to develop a model that can generate natural language descriptions of images accurately, informatively, grammatically correctly, and fluently. The model should handle a wide variety of images and audiences. Evaluation is based on accuracy, informativeness, fluency, diversity, and context.

II. Introduction

Image captioning is difficult in computer vision and NLP. A model must interpret an image's visual information and create a natural language description. Combining CNNs with LSTM networks for picture caption creation is effective. CNNs extract high-level picture characteristics, whereas LSTMs represent natural language sequential relationships.

A CNN-LSTM image caption generator model has two key parts:

Encoder: Our CNN encoder extracts features from the input picture. ImageNet is used to pre-train the encoder. Decoder: An LSTM network creates a caption.

A collection of image-caption pairings trains the decoder. To maximize picture caption accuracy, the encoder and decoder are trained together. The model may create captions for fresh photos by feeding them to the encoder and decoding them after training.

Applications for image caption generators include:

- Captioning social media photographs
- Helping visually impaired individuals perceive images
- Captioning e-commerce product photographs
- Search engine picture captioning

Benefits of CNNs and LSTMs for picture captioning

- CNNs and LSTMs have many benefits for picture caption generation:
- CNNs extract high-level picture information for accurate and informative captioning.
- For grammatically accurate and fluent captions, LSTMs model natural language sequential dependencies.
- CNNs and LSTMs can be trained together to maximize picture caption accuracy.

Image captioning issues

- Image caption generation remains difficult despite recent advances. Challenges include:
- Create accurate, informative captions.
- Captioning with grammar and fluency.
- Create different, non-repetitive subtitles.
- Create captions that represent the image's context, including object connections.

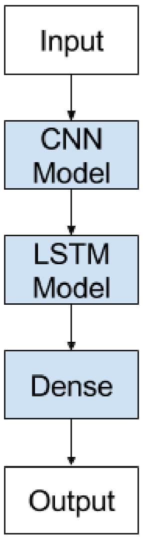
Image caption generation models are being improved by researchers. Research topics of interest include deep learning models for complicated image and language representations.

Train picture caption generator models with reinforcement learning to produce more human-like captions.

Train image caption generator models to learn from images and text using multimodal learning.

III. Model Architechture

CNN and LSTM are two distinct neural network architectures that are combined to create the CNN-LSTM architecture. When it comes to completing tasks like image component recognition, CNNs are thought to be incredibly effective. While LSTMs produce predictions for data sequences, CNN layers extract properties from the incoming data. Language generation and computer vision are related to image captioning. Because they can manage long-term dependencies, recurrent neural networks with LSTM components work incredibly well in various applications. For tasks requiring the prediction of sequences with spatial inputs, such as pictures or videos, this approach is perfect. It can be used for many different things, like action recognition, image description, and video description.

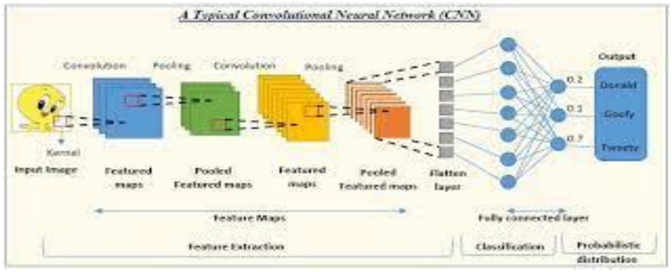


CNN-LSTM model structure

The CNN-LSTM architecture is particularly useful when the input data has both spatial and temporal structure. Spatial structure refers to the location of components in a specific space, such as the arrangement of pixels in an image or words in a sentence, paragraph, or document. This technology is also useful in satellite imaging for deforestation analysis [13]. This architecture is also used when the output is expected to have a temporal structure, such as a sequence of words in a textual description. In summary, CNN-LSTMs are used when the input data has both spatial and temporal structure and the output is expected to have temporal structure as well.

A. Convolutional Neural Network

Convolutional neural networks analyze visual data. It processes images by dividing them and finding repeating patterns. The model can identify objects and features in images and make predictions. Applying filters at different resolutions to the input image works. These filters extract edges, shapes, and colors from images. The CNN uses these features to create an image representation for object and pattern recognition. Convolutional neural networks have layers for processing information. Hidden layers analyze and identify important features while the input layer receives the image. After processing the hidden layers' data, the output layer predicts. Image captions are closely related to image component semantics. CNN intermediate layers generate these features. These differ in abstraction. Edges and shapes are combined to form image components and high-level shapes. CNN extracts objects from images using its training data. CNNs extract meaningful information from images, starting the caption-generation process. CNNs are good image describers.



CNN layers ABLE I

B. Long-Short-Term Memory (LSTM)

LSTM networks can handle sequential data like time series, text, and speech because they can remember previous inputs for a set time. LSTM networks regulate information flow with the help of memory cells and gates. The gates control information flow into and out of the memory cell, storing it for a long time. The gates determine what data is sent to other network areas, removed, and added to the memory cell. Thus, the LSTM network efficiently processes sequential data and makes long-term dependency predictions. These elements allow the network to selectively remember or forget information based on input to improve predictions and decisions. LSTM models track their past data and use it to guide their current processing. Memory cells in the LSTM model will track this information. Input/forget gates control system data flow. Thus, the LSTM can decide which information is relevant to the task and which to keep or discard. LSTM gates are unique structures that decide what data to keep or delete. If gates are closed, memory cell data remains unchanged. Open or close the gates. Thus, the model can learn and remember important information for long periods of time, which it can use to caption photos or make predictions. The LSTM model addresses the vanishing gradient issue in many recurrent neural network models.

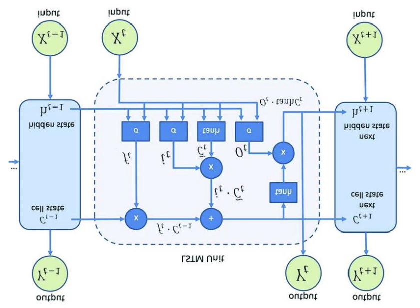
Deep network training is difficult due to the "vanishing gradient problem," which occurs when the gradient rapidly decreases as network depth increases. LSTM models learn language and vision interactions by monitoring and predicting future context. This LSTM behaviour aids visual data captioning. Captioning images requires word sequences. LSTM predicts caption words from input images.A Long Short-Term Memory (LSTM) Network is composed of four distinct gates that serve specific functions. These gates work together to regulate the flow of information into and out of the memory cell and to control the stability of the gradient during training. They are:

Forget Gate(f): By computing a value between 0 and 1, the forget gate in an LSTM regulates how much data should be kept from the previous state and uses that value to update the current state. The amount of information that should be retained and the amount that should be thrown away are determined using the prior output and input. This mechanism aids in retaining only pertinent data and eliminating unnecessary data, improving the model's performance.

Input Gate(i): Based on the current input and the previous output, the input gate in an LSTM calculates how much new data needs to be stored in the cell state. It creates a scalar between 0 and 1, which serves as a weighting factor, by combining these two signals. The new data to be stored in the cell state is then determined by applying this weighting factor to the output of the tanh activation function. The LSTM is able to retain its context over time by appending this new data to the prior state.

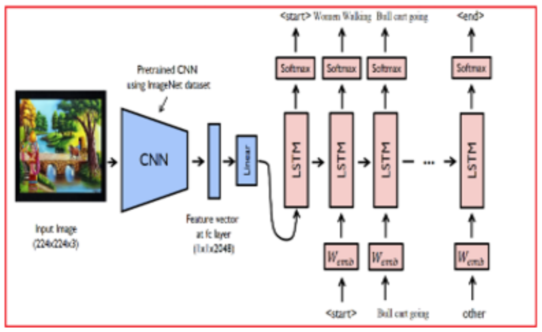
Input Modulation Gate(g): The Input Modulation Gate in LSTM is a part of the input gate that adjusts the incoming information before it is stored in the internal state cell. It modulates the input data by using a non-linear transformation, which helps to standardize the data and ensure that it has zero mean. The Input gate will respond to the internal state cell by making the input zero-mean through non-linearity. This helps to speed up the learning process by reducing the time it takes for the network to converge. While this gate's actions are not as crucial as the others, It is considered optimal to incorporate it within the design of the LSTM cell.

Output Gate(o): The output gate in LSTM uses the current input and the state before the current to determine how much of the current state should be outputted. This output is then passed through a tanh function to add non-linearity and make it zero-mean before being scaled by the output gate's fraction. This final output is then used as the input for the next LSTM block and also given back into the current LSTM block as part of the state. The LSTM also uses an additional internal state, called the cell state, to pass information from one time step to the next. This allows the LSTM to retain relevant information for longer periods of time, making it well-suited for tasks such as sequence prediction.



.

In our image caption generator model, we will combine these two network architectures, which is commonly referred to as a CNN-RNN hybrid model shown in Fig



<start> Women Walking Bull cart going
<end>
Softmax
Pretrained CNN using ImageNet dataset
CNN
Linear
LSTM
Input Image (224x224x3)
Feature vector at fc layer (1x1x2048)
W_{emb}
<start>
Bull cart going
other

CNN – LSTM Model

A pre-trained CNN is required to extract important features from an input image, which are then used as input for an LSTM network that has been trained to model language. The feature representation obtained from the CNN is altered to fit the input specifications of the LSTM network, enabling it to produce output based on the characteristics of the image.

The label and target text for training an LSTM model would be the text description needed to be generated. For example, if there is a picture containing an old man who is wearing some hat, the label and target text would be:

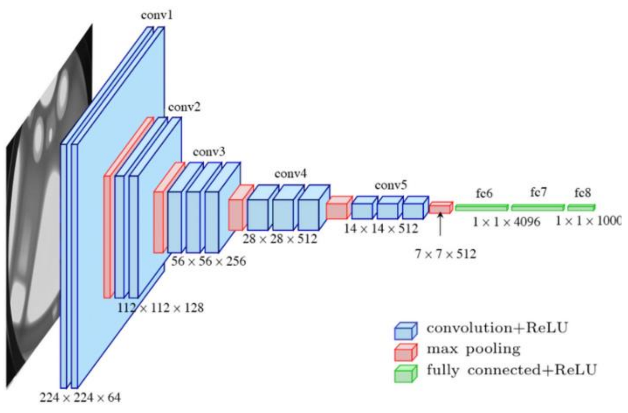
Label — [<start> ,An, old, man, is, wearing, a , hat]

Target — [An old man is wearing a hat .,<End>]
This is done so that the model can identify the beginning and end of the caption and understand how the words in the captions relate to each other.

C. VGG16

VGG16 is a popular variant of CNN and is widely considered to be one of the most advanced models for computer vision tasks. The VGG16 model is a highly advanced convolutional neural network that was created with the goal of improving upon previous computer vision models. The creators of this model experimented with different architectural designs and ultimately decided to use a deep network with small convolution filters. This resulted in a model with a large number of layers and trainable parameters, and ultimately led to VGG16 becoming one of the best-performing models in the field of computer vision. It is used for the classification and identification of images belonging to diverse categories.

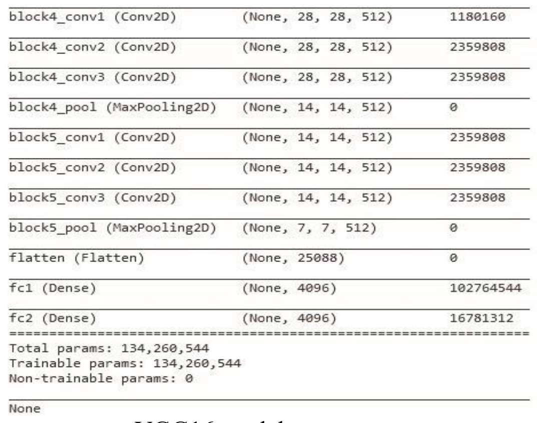
VGG16 Architecture



VGG16 Model

Loading VGG16 Model

After loading the VGG16 model, the output produced by the final dense layer, which is usually 4096 in size and seen in Fig. 6, is passed to the encoder and decoder modules. The encoder module takes the image embeddings and encodes them into a tensor whereas the decoder generates output (captions) at each step. The encoder decoder model which combines the CNN and LSTM is very efficient in generating rich text captions of input images.



```
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160

block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808

block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808

block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0

block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808

block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808

block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808

block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0

flatten (Flatten)            (None, 25088)             0

fc1 (Dense)                  (None, 4096)              102764544

fc2 (Dense)                  (None, 4096)              16781312
=================================================================
Total params: 134,260,544
Trainable params: 134,260,544
Non-trainable params: 0

None
```

VGG16 model parameters

IV. Data Set

The Flickr8k dataset is a collection of 8092 photographs in JPEG format, along with accompanying text descriptions of the images. The dataset is organized into two main directories, one containing the images themselves, and the other containing text files with different sources of descriptions for the photographs. The main file of the dataset is called Flickr8k.token and it is located in the Flickr8k_text folder. This file contains the names of the images and their corresponding captions, with each image and its captions separated by a newline character ("\n"). Overall, the dataset includes 8091 images, each with 5 English captions. The Flickr8k is a 1GB dataset comprised of images and accompanying text descriptions, organized into three subsets. The training set contains 6000 images, the testing set has 1000, and the validation set also has 1000 images. In order to prepare the dataset for use, the text descriptions have undergone a process of cleaning and formatting. This includes removing punctuation, converting all words to lowercase, and removing any numerical values. Additionally, the dataset has been organized in a sequential pattern, using a tool called tqdm to track progress.

V. Methodology

A. Pre-requisites

To work on this project, it is important to have a strong understanding of various technologies and tools, including deep learning, Python programming, working with Kaggle notebooks, and using the Keras library. Additionally, experience with NumPy and natural language processing is necessary. It is also important to ensure that all of the necessary libraries are installed, in order to effectively run the project

Tensorflow: TensorFlow [11] is an open-source platform that provides extensive support for machine learning and artificial intelligence. With TensorFlow, developers can take advantage of its compatibility with multiple hardware platforms, user-friendly APIs, and the support of an active community.

Keras: Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano.One of the key strengths of Keras is its simplicity and ease of use. Keras also provides a number of pre-trained models, making it easy to get started with transfer learning and fine-tuning existing models.

Relevant Python libraries like NumPy, Pickle- for serializing and deserializing Python objects, tqdm-to get progress feedback through the progress bar widget for long-running tasks or iterations over large datasets.

Tokenizer: Tokenization is the process of breaking down a piece of text into smaller units, known as tokens. Tokens can be words, phrases, symbols, or other elements of the text, depending on the specific use case. Tokenization is a fundamental step in many natural language processing (NLP) tasks, including text classification, sentiment analysis, and information retrieval. Tokenization is important because it allows us to represent text in a structured format that can be easily processed by computers. Tokens can be used as features in machine learning models and can be further processed

to extract more meaningful information, such as word embeddings or n-grams.

B. Pre-processing the Image

In our project, we are using a pre-trained model named VGG16 from the Visual Geometry Group for image recognition. This model is a available within the Keras library, and thus it will not require additional installation or setup. For this project, the image features are extracted by resizing the images to a size of 224*224 pixels. The extraction process is performed on the image just before the last layer of the classification model. This location is chosen because it is used for predicting the classification of a photo, but since it is not required to classify the images, we exclude the last layer during the feature extraction process.

C. Creating vocabulary for the image:

Before using text data in a machine learning or deep learning model, it is necessary to clean and prepare it for the model. This process includes splitting the text into individual words, and handling issues with punctuation and case sensitivity. Additionally, since computers do not understand English words, we need to represent them with numbers. This is done by creating a vocabulary and mapping each word to a unique index value, and then encoding each word into a fixed-sized vector. Only after this process, the text become readable by the machine and can be used to generate captions for images. We plan to reduce the size of our vocabulary by processing the text in the following sequence through cleaning.

D. Training the model:

The dataset includes a file called "Flickr_8k.trainImages.txt" Compiling a list of 8000 image names that will be utilized during the training phase of the project.

The first step is to load the features extracted from the pre-trained CNN model. To train the model, we will be using the 8000 training images by breaking them into smaller chunks called batches and using them to generate input and output sequences. These sequences are then used to fit the model. The training process is set to run for 20 cycles known as epochs.

E. Model evaluation – Bilingual evaluation understudy score(BLUE'S):

BLEU is a method to measure the similarity between a generated sentence and a reference sentence. It is mostly used to evaluate the performance of machine translation systems. The score ranges from 0.0 to 1.0, where 1.0 represents a perfect match and 0.0 represents a complete mismatch.

To evaluate the performance of the model, the generated captions are compared with the actual captions. The similarity between these two sets is measured using the BLEU score, which is a method to evaluate text similarity. This score is calculated for the entire set of captions, and it provides a summary of how well the generated captions match the expected captions.

VI. Results

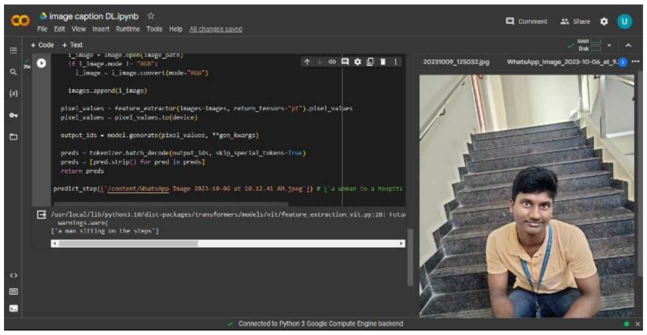
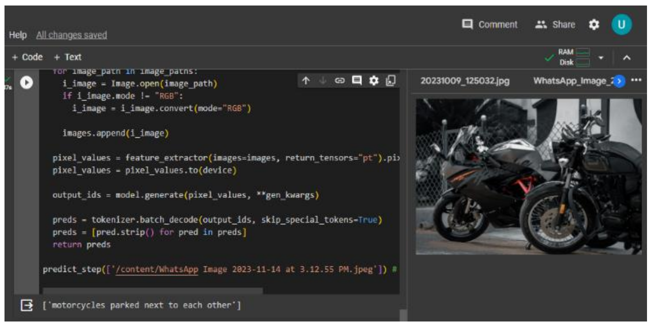
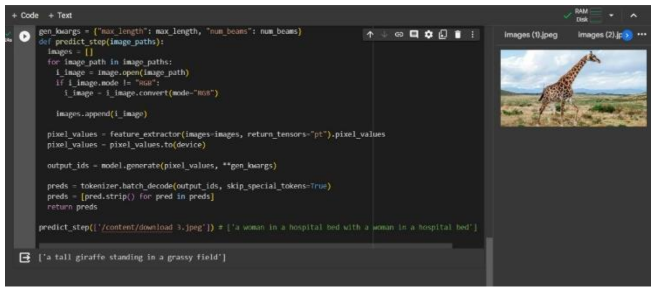


image caption DL.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
20231009_125032.jpg
WhatsApp_Image_2023-10-06_at_9.
Connected to Python 3 Google Compute Engine backend

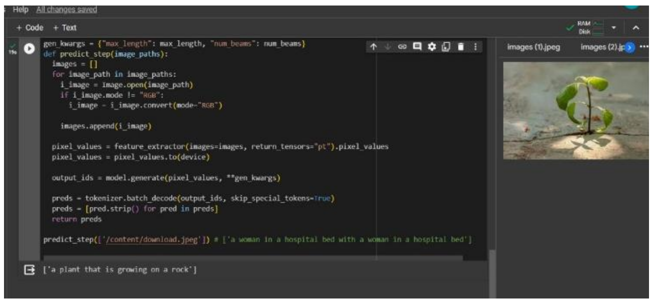
```
    i_image = image.open(image_path)
    if i_image.mode != "RGB":
      i_image = i_image.convert(mode="RGB")

    images.append(i_image)

  pixel_values = feature_extractor(images=images, return_tensors="pt").pixel_values
  pixel_values = pixel_values.to(device)

  output_ids = model.generate(pixel_values, **gen_kwargs)

  preds = tokenizer.batch_decode(output_ids, skip_special_tokens=True)
  preds = [pred.strip() for pred in preds]
  return preds

predict_step(['/content/WhatsApp Image 2023-10-06 at 10.12.41 AM.jpeg']) # ['a woman in a hospit

/usr/local/lib/python3.10/dist-packages/transformers/models/vit/feature_extraction_vit.py:28: Futu
  warnings.warn(
['a man sitting on the steps']
```



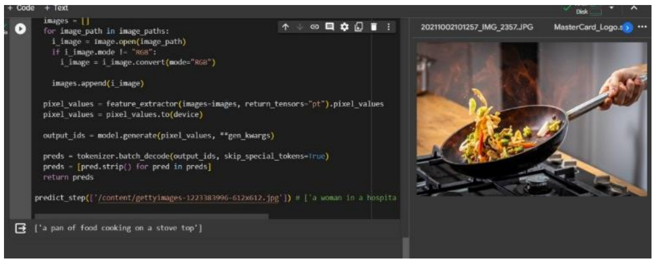
```
  for image_path in image_paths:
    i_image = Image.open(image_path)
    if i_image.mode != "RGB":
      i_image = i_image.convert(mode="RGB")

    images.append(i_image)

  pixel_values = feature_extractor(images=images, return_tensors="pt").pi
  pixel_values = pixel_values.to(device)

  output_ids = model.generate(pixel_values, **gen_kwargs)

  preds = tokenizer.batch_decode(output_ids, skip_special_tokens=True)
  preds = [pred.strip() for pred in preds]
  return preds

predict_step(['/content/WhatsApp Image 2023-11-14 at 3.12.55 PM.jpeg']) #

['motorcycles parked next to each other']
```



```
gen_kwargs = {"max_length": max_length, "num_beams": num_beams}
def predict_step(image_paths):
  images = []
  for image_path in image_paths:
    i_image = Image.open(image_path)
    if i_image.mode != "RGB":
      i_image = i_image.convert(mode="RGB")

    images.append(i_image)

  pixel_values = feature_extractor(images=images, return_tensors="pt").pixel_values
  pixel_values = pixel_values.to(device)

  output_ids = model.generate(pixel_values, **gen_kwargs)

  preds = tokenizer.batch_decode(output_ids, skip_special_tokens=True)
  preds = [pred.strip() for pred in preds]
  return preds

predict_step(['/content/download 3.jpeg']) # ['a woman in a hospital bed with a woman in a hospital bed']

['a tall giraffe standing in a grassy field']
```



```
gen_kwargs = {"max_length": max_length, "num_beams": num_beams}
def predict_step(image_paths):
  images = []
  for image_path in image_paths:
    i_image = Image.open(image_path)
    if i_image.mode != "RGB":
      i_image = i_image.convert(mode="RGB")

    images.append(i_image)

  pixel_values = feature_extractor(images=images, return_tensors="pt").pixel_values
  pixel_values = pixel_values.to(device)

  output_ids = model.generate(pixel_values, **gen_kwargs)

  preds = tokenizer.batch_decode(output_ids, skip_special_tokens=True)
  preds = [pred.strip() for pred in preds]
  return preds

predict_step(['/content/download.jpeg']) # ['a woman in a hospital bed with a woman in a hospital bed']

['a plant that is growing on a rock']
```



```
  images = []
  for image_path in image_paths:
    i_image = Image.open(image_path)
    if i_image.mode != "RGB":
      i_image = i_image.convert(mode="RGB")

    images.append(i_image)

  pixel_values = feature_extractor(images=images, return_tensors="pt").pixel_values
  pixel_values = pixel_values.to(device)

  output_ids = model.generate(pixel_values, **gen_kwargs)

  preds = tokenizer.batch_decode(output_ids, skip_special_tokens=True)
  preds = [pred.strip() for pred in preds]
  return preds

predict_step(['/content/gettyimages-1223383996-612x612.jpg']) # ['a woman in a hospita

['a pan of food cooking on a stove top']
```

VII. Conclusion

We have built an Image Caption Generator using a combination of a CNN and an LSTM model. This architecture, known as CNN-LSTM, can be used in a variety of areas such as computer vision and natural language processing. Our specific implementation uses an encoder-decoder approach to generate grammatically correct captions for images. The model we are proposing uses a CNN as an encoder and an LSTM as a decoder. We have evaluated the model using the standard metric, called BLEU, on the Flickr8k Captions dataset. Our findings indicate that the model performs comparably with other leading techniques, as evaluated by the BLEU metric. Our proposed model has displayed positive results in terms of BLEU scores, though there is still scope for enhancement. In future we plan to improve the semantic relevance of the generated captions by implementing the attention mechanism where attention scores are used to change the attention strength on various image attributes.

References

[1] Md. Z. Hossain, F. Sohel, Md. F. Shiratuddin, and H. Laga, “A Comprehensive Survey of Deep Learning for Image Captioning,” arXiv:1810.04020v2 [cs.CV] 14 Oct 2018.

[2] Q. You, H. Jin, Z. Wang, C. Fang, and J. Luo, “Image Captioning with Semantic Attention”, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 46514659.

[3] C. Park, B. Kim and G. Kim, “Attend to You: Personalized Image Captioning With Context Sequence Memory Networks”, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 895-903

[4] J. Johnson, K. Karpathy and L. Fei-Fei, “DenseCap: Fully Convolutional Localization Networks for Dense Captioning”, Proceedings of the IEEE Conference on Computer Vision and

[5] Pattern Recognition (CVPR), 2016, pp. 4565-4574

[6] J. Aneja, A. Deshpande and A.G. Schwing, “Convolutional Image Captioning”, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 5561-5570

[7] G. Srivastava and R. Srivastava, “A Survey on Automatic Image Captioning”, Mathematics and Computing. ICMC 2018. Communications in Computer and Information Science, vol 834. Springer.

[8] H. Wang, Y. Zhang and X. Yu, "An Overview of Image Caption Generation Methods", Computational Intelligence and Neuroscience, vol. 2020, Article ID 3062706, 13 pages, 2020.

[9] P. Sharma, N. Ding, S. Goodman, and R. Soricut,” Conceptual Captions: A Cleaned, Hypernymed, Image Alt-text Dataset For Automatic Image Captioning”, In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 2556–2565, Melbourne, Australia 2018.

[10] G. Sairam, M. Mandha, P. Prashanth and P. Swetha, "Image Captioning using CNN and LSTM," 4th Smart Cities Symposium (SCS 2021), Online Conference, Bahrain, 2021, pp. 274-277.

[11] V. Agrawal, S. Dhekane, N. Tuniya and V. Vyas, "Image Caption Generator Using Attention Mechanism," 12th International Conference on Computing Communication and Networking Technologies (ICCCNT), Kharagpur, India, 2021, pp. 1-6.

[12] Online: https://www.tensorflow.org

[13] Z. Shi, X. Zhou, X. Qiu and X. Zhu, “Improving Image Captioning with Better Use of Captions”, Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 7454–7464, July 5 - 10, 2020. c©2020 Association for Computational Linguistics.

[14] G. Geetha, T. Kirthigadevi, G.Godwin Ponsam, T. Karthik and M. Safa, “Image Captioning Using Deep Convolutional Neural Networks (CNNs)”, Journal of Physics.: Conf. Ser. 1712 012015, 2020

[15] C. Wang, H. Yang, C. Bartz and C. Meinel, “Image Captioning with Deep Bidirectional LSTMs”, MM '16: Proceedings of the 24th ACM international conference on Multimedia. 2016.

[16] A. K. Poddar and Dr. R. Rani, “Hybrid Architecture using CNN and LSTM for Image Captioning in Hindi Language”, International Conference on Machine Learning and Data Engineering, Procedia Computer Science 218 (2023) 686–696.