Image caption Generation using Deep Architecture

ABSTRACT

Accurately describing a complex scene requires a deeper representation of what's going on in the scene, capturing how the various objects relate to one another and translating it all into natural-sounding language. A lot of efforts to construct computer-generated natural descriptions of images propose combining current state-of-the-art techniques in both computer vision and natural language processing to form a complete image description approach. But what if we instead merged recent computer vision and language models into a single jointly trained system, taking an image and directly producing a human readable sequence of words to describe it? This idea comes from recent advances in machine translation between languages, where a Recurrent Neural Network (RNN) transforms, say, a French sentence into a vector representation, and a second RNN uses that vector representation to generate a target sentence in German. Now, what if we replaced that first RNN and its input words with a deep Convolutional Neural Network (CNN) trained to classify objects in images? Normally, the CNN's last layer is used in a final Softmax among known classes of objects, assigning a probability that each object might be in the image. But if we remove that final layer, we can instead feed the CNN's rich encoding of the image into a RNN designed to produce phrases. We can then train the whole system directly on images and their captions, so it maximizes the likelihood that descriptions it produces best match the training descriptions for each image.

The model combines a vision CNN with a language-generating RNN so it can take in an image and generate a fitting natural-language caption.

INTRODUCTION

The problem of image caption generation involves outputting a readable and concise description of the contents of a photograph. It is a challenging artificial intelligence problem as it requires both techniques from computer vision to interpret the contents of the photograph and techniques from natural language processing to generate the textual description.

Recently, deep learning methods have achieved state-of-the-art results on this challenging problem. The results are so impressive that this problem has become a standard demonstration problem for the capabilities of deep learning.

Human beings usually describe a scene using natural languages which are concise and compact. However, machine vision systems describe the scene by taking an image which is a two-dimension arrays. The idea is mapping the image and captions to the same space and learning a mapping from the image to the sentences.

PROBLEM STATEMENT

Automatically describing the content of an image is a fundamental problem in artificial intelligence that connects computer vision and natural language processing. This machine-learning system can automatically produce captions to accurately describe images the first time it sees them. This kind of system could eventually help visually impaired people understand pictures, provide alternate text for images in parts of the world where mobile connections are slow, and make it easier for everyone to search on Google for images.

OBJECTIVE

People can summarize a complex scene in a few words without thinking twice. It's much more difficult for computers. Automatically describing the content of an image is a fundamental problem in artificial intelligence that connects computer vision and natural language processing. This machine-learning system can automatically produce captions to accurately describe images the first time it sees them. This kind of system could eventually help visually impaired people understand pictures, provide alternate text for images in parts of the world where mobile connections are slow, and make it easier for everyone to search on Google for images.

Literature Survey

Auto image captioning is the process to automatically generate human like descriptions of the images. It is very dominant task with good practical and industrial significance. Auto Image captioning has a good practical use in industry, security, surveillance, medical, agriculture and many more prime domains. It is not just very crucial but also very challenging task in computer vision. Traditional object detection and image classification task just needed to identify objects within the image where the task of Auto image captioning is not just identifying the objects but also identifying the relationship between them and total scene understanding of the image. After understanding the scene, it is also required to generate a human like description of that image. Since the boost of automation and Artificial Intelligence lots of research is going on to give machine human like capabilities and reduce manual work. For machines acquiring results and accuracy as good as human in image captioning problem has always been a very challenging task.

Literature Survey

Auto image captioning is performed by following key tasks in order. At first features are extracted after proper extraction of features different objects from an image are detected, after that the relationship between objects are to be identified (i.e. if objects are cat and grass it is to be identified that if cat in on grass). Once objects are detected and relationships are identified now it is required to generate the text description, i.e., Sequence of words in orderly form that they make a good sentence according to the relationship between the image objects.

To perform above key tasks using deep learning different deep learning networks are used. we started by adopting an encoder-decoder architecture that incorporates visual attention mechanism to generate image captioning. The encoder part is CNN based and the decoder one uses the visual attention module.

Literature Survey

Under the encoder-decoder framework for image captioning, a CNN can produce a rich representation of the input image by embedding it to a fixed length vector representation. Many different CNN can be used, e.g., VGG, Inception V3, ResNet. In our project, we use Inception V3 model created by Google Research as encoder. We have pre-processed images with the Inception V3 model and have extracted features. The extractor produces L vectors, each of which is a D dimensional representation corresponding to a part of the image. Image Vector and Global Image Vector can be obtained by using a single layer perceptron with rectifier activation function.

For image captioning, attention tends to focus on specific regions in the image while generating descriptions. based on the hidden state, the decoder would attend to the specific regions of the image and compute context vector using the spatial image features from a convolution layer of a CNN. Using a single layer neural network followed by a softmax function to generate the attention distribution over some particular regions of the image. Based on the attention distribution, the context vector is be obtained. Given the image representations, a decoder is employed to translate the image into natural sentences. A decoder is a RNN which are typically implemented using either LSTM or GRU. Here we have used GRU as a decoder which has a simpler structure than LSTM. Also, unlike RNN, GRU does not suffer from the vanishing gradient problem.

Contributions

METHOD USED IN PROJECT –

The encoder-decoder image captioning system would encode the image, using a pre-trained Convolutional Neural Network that would produce a hidden state. Then, it would decode this hidden state by using a RNN and generate a caption.

For each sequence element, outputs from previous elements are used as inputs, in combination with new sequence data. This gives the RNN networks a sort of memory which might make captions more informative and context aware.

But RNNs tend to be computationally expensive to train and evaluate, so in practice, memory is limited to just a few elements. Attention models can help address this problem by selecting the most relevant elements from an input image.

Contributions

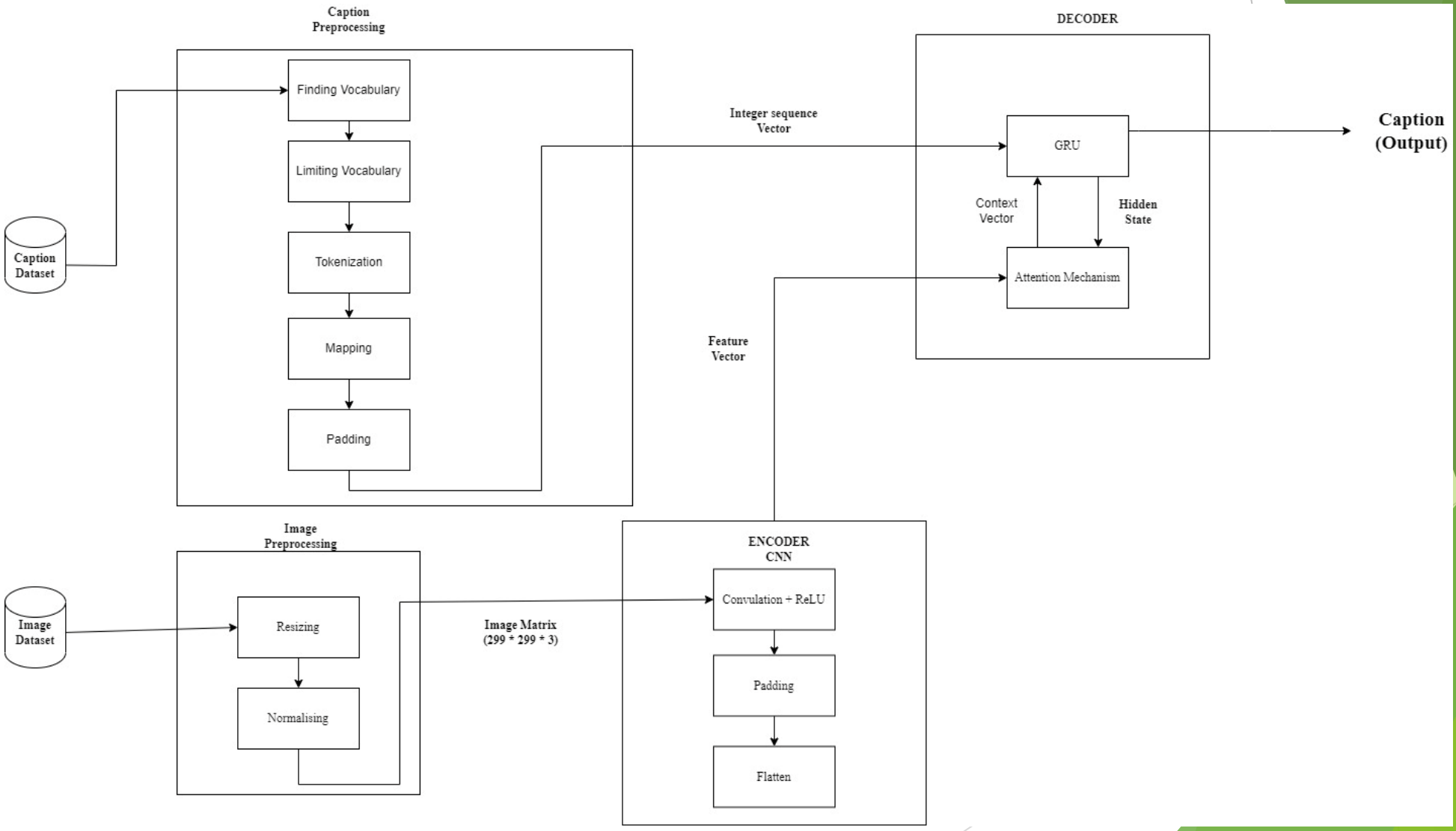
JUSTIFICATION –

Inception v3 is used a pre-trained CNN as it is a widely-used image recognition model that has been shown to attain greater than 78.1% accuracy on the ImageNet dataset. The Inception V3 model used several techniques for optimizing the network for better model adaptation. Images which contain more information can be treated well with InceptionV3. Also, inception v3 has less parameters than other models.

GRU use less training parameters and therefore use less memory, execute faster and train faster than LSTM's whereas LSTM is more accurate on dataset using longer sequence. GRU also does not suffer from vanishing gradient problem.

Attention mechanism helps to focus on important areas and improve the quality of generated captions.

BLOCK DIAGRAM



EXPLANATION OF OVERALL BLOCK DIAGRAM

Captions are pre-processed to clean the data, form the vocabulary and convert the sentences into a form understood by the machine. Images are pre-processed in order to meet the format required by the encoder.

Encoder is used to extract the features from the pre-processed image. Here we have used CNN (InceptionV3) as an encoder.

Then a decoder is employed to translate the image into natural sentences. A decoder is a RNN which are typically implemented using either LSTM or GRU. Here we have used GRU as a decoder along with an attention mechanism. Attention mechanism tends to focus on specific regions in the image while generating descriptions. The descriptions are more appropriate. This is an 'inject' architecture, where the context vector after going though attention mechanism is injected into the GRU. The GRU produces the appropriate words to form the caption.

Deliverables:

Input – Image

Output – Caption for the given image

MODULES

- Module 1 – IMAGE PREPROCESSING
- Module 2 – CAPTIONS PREPROCESSING
- Module 3 – ENCODER – CONVOLUTIONAL NEURAL NETWORK
- Module 4 – ATTENTION BASED MECHANISM
- Module 5 – DECODER – RECURRENT NEURAL NETWORK

MODULE1 – Caption Preprocessing

Transforming the text captions into integer sequences, with the following steps:

- Iterate over all captions, split the captions into words, and compute a vocabulary of the top 5,000 words to save memory.
- Tokenize all captions by mapping each word to its index in the vocabulary. This gives us a vocabulary of all of the unique words sorted by their frequency in the data.
- Finally, all sequences are padded to be the same length as the longest one.

INPUT – Captions

OUTPUT – Integer Sequence Vector

MODULE1 – Caption Preprocessing

CODE SNIPPETS:

Finding vocabulary:

```
vocabulary = []
for txt in data.caption.values:
    vocabulary.extend(txt.split())

print('Vocabulary Size: %d' % len(set(vocabulary)))

ct = Counter(vocabulary)
appen_1 = []
appen_2 = []

for i in ct.keys():
    appen_1.append(i)

for j in ct.values():
    appen_2.append(j)

data_word_count = {"word":appen_1 , "count":appen_2}

dfword = pd.DataFrame(data_word_count)
dfword = dfword.sort_values(by='count', ascending=False)
dfword = dfword.reset_index()[["word","count"]]
dfword.head(10)
```

MODULE1 – Caption Preprocessing

CODE SNIPPETS:

Tokenization:

```
top_k = 5000
tokenizer = tf.keras.preprocessing.text.Tokenizer(num_words=top_k,
                                                  oov_token="<unk>",
                                                  filters='!"#$%&()*+.,-/:;=?@[\]^_`
{|}~ ')
tokenizer.fit_on_texts(train_captions)
train_seqs = tokenizer.texts_to_sequences(train_captions)


tokenizer.word_index['<pad>'] = 0
tokenizer.index_word[0] = '<pad>'

train_seqs = tokenizer.texts_to_sequences(train_captions)

cap_vector = tf.keras.preprocessing.sequence.pad_sequences(train_seqs, padding='pos
t')

print(cap_vector.shape)
```

MODULE 2 – Image Preprocessing

We are using the Flickr8k dataset. It consists a total of 8000 images along with the captions. Each image contains 5 captions. The images are duplicated 5 times corresponding to each caption. Therefore, there are total of 40000 images with captions.

- The images are scaled to fit the model's input requirements prior to feature extraction. Inception v3 requires the input images to be in the shape of 299 x 299 x 3. Therefore, the images are resized.
- Pre-process the images using the preprocess_input method to normalize the image so that it contains pixels in the range of -1 to 1, which matches the format of the images used to train InceptionV3.

INPUT – Image

OUTPUT – Image matrix (299 x 299 x 3)

MODULE 2 – Image Preprocessing

CODE SNIPPETS:

```
def load_image(image_path):
    img = tf.io.read_file(image_path)
    img = tf.image.decode_jpeg(img, channels=3)
    img = tf.image.resize(img, (299, 299))
    img = tf.keras.applications.inception_v3.preprocess_input(img)
    return img, image_path
```

MODULE 3 – ENCODER

A very deep convolutional neural network model is proficient in extracting visual features from an image in a hierarchical manner, starting from very basic features like edge detectors, and then progressively building more complex features like shape detection.

We are using transfer learning method. Inception v3 (greater than 78.1% accuracy on the ImageNet dataset) is used to extract the features from the image. The last layer of InceptionV3 is the softmax classifier (layer with 1000 hidden neurons) which returns the probability of a class. This layer should be removed so as to get a feature representation of an image. We will use the last Dense layer (2048 hidden neurons) after popping the classifier layer.

INPUT – Image matrix (299 x 299 x 3)

OUTPUT – Feature vector (2048)

MODULE 3 – ENCODER

CODE SNIPPETS:

```
image_model = tf.keras.applications.InceptionV3(include_top = False, weights = 'image
net')

new_input = image_model.input
hidden_layer = image_model.layers[-1].output

image_features_extract_model = tf.keras.Model(new_input, hidden_layer)
image_features_extract_model.summary()
extracted_features = {}

for img, path in tqdm(image_dataset):
  batch_features = image_features_extract_model(img)
  batch_features = tf.reshape(batch_features,
                              (batch_features.shape[0], -1, batch_features.shape[3]))

  for bf, p in zip(batch_features, path):
    path_feature = p.numpy().decode("utf-8")
    extracted_features[path_feature] = bf.numpy()
class CNN_Encoder(tf.keras.Model):
    def __init__(self, embedding_dim):
        super(CNN_Encoder, self).__init__()
        self.fc = tf.keras.layers.Dense(embedding_dim)

    def call(self, x):
        x = self.fc(x)
        x = tf.nn.relu(x)
        return x
```

MODULE 4 – DECODER WITH ATTENTION MECHANISM

1. ATTENTION MECHANISM:

At each timestep, the Attention module takes the encoded image as input along with the GRU's hidden state for the previous timestep. It would attend to the specific regions of the image and compute context vector using the spatial image features from a convolution layer of a CNN. It produces an Attention Score that assigns a weight to each pixel of the encoded image. The higher the weight for a pixel, the more relevant it is for the word to be output at the next timestep. An attention vector related to time t is used to replace the fixed length vector obtained from the image encoder CNN.

INPUT – Feature vector from encoder and last hidden state

OUTPUT – Context vector

MODULE 4 – DECODER WITH ATTENTION MECHANISM

CODE SNIPPETS:

```
class BahdanauAttention(tf.keras.Model):
  def __init__(self, units):
    super(BahdanauAttention, self).__init__()
    self.W1 = tf.keras.layers.Dense(units)
    self.W2 = tf.keras.layers.Dense(units)
    self.V = tf.keras.layers.Dense(1)

  def call(self, features, hidden):
    hidden_with_time_axis = tf.expand_dims(hidden, 1)

    score = tf.nn.tanh(self.W1(features) + self.W2(hidden_with_time_axis))

    attention_weights = tf.nn.softmax(self.V(score), axis=1)

    context_vector = attention_weights * features
    context_vector = tf.reduce_sum(context_vector, axis=1)

    return context_vector, attention_weights
```

MODULE 4 – DECODER WITH ATTENTION MECHANISM

2. GRU:

The context vector after going though attention mechanism is injected into the GRU, for example by treating the image vector on a par with a 'word' and including it as part of the caption prefix. The context vector, hidden state (initialized to 0) and the decoder input (which is the start token) is passed to the decoder. It returns the predictions and the decoder hidden state. The hidden state is then passed back into the model and the predictions are used to calculate the loss. Teacher forcing to decide the next input to the decoder. It is the technique where the target word is passed as the next input to the decoder.

INPUT: Context vector, hidden state (initialized to 0) and the decoder input (which is the start token)

OUTPUT: Captions

MODULE 4 – DECODER WITH ATTENTION MECHANISM

CODE SNIPPETS:

```
class RNN_Decoder(tf.keras.Model):
  def __init__(self, embedding_dim, units, vocab_size):
    super(RNN_Decoder, self).__init__()
    self.units = units

    self.embedding = tf.keras.layers.Embedding(vocab_size, embedding_dim)
    self.gru = tf.keras.layers.GRU(self.units,
                                   return_sequences=True,
                                   return_state=True,
                                   recurrent_initializer='glorot_uniform')
    self.fc1 = tf.keras.layers.Dense(self.units)
    self.fc2 = tf.keras.layers.Dense(vocab_size)

    self.attention = BahdanauAttention(self.units)

  def call(self, x, features, hidden):
    context_vector, attention_weights = self.attention(features, hidden)

    x = self.embedding(x)
```

```
    x = tf.concat([tf.expand_dims(context_vector, 1), x], axis=-1)

    output, state = self.gru(x)

    x = self.fc1(output)

    x = tf.reshape(x, (-1, x.shape[2]))

    x = self.fc2(x)

    return x, state, attention_weights

  def reset_state(self, batch_size):
    return tf.zeros((batch_size, self.units))
```

RESULTS

DATASET –FLICKR 8k

Consists of 8092 images in jpg format. Each image contains 5 captions. Therefore, a total of 40460 images each with a caption.

60% of the total (i.e., 24,000) is used for training and 40% (i.e., 16,000) is used for testing.

The Flickr8k – Dataset contains all the images with a unique ID.

The Flickr8k text contains - Flickr8k.token.txt – all the image IDs and the corresponding 5 raw captions.

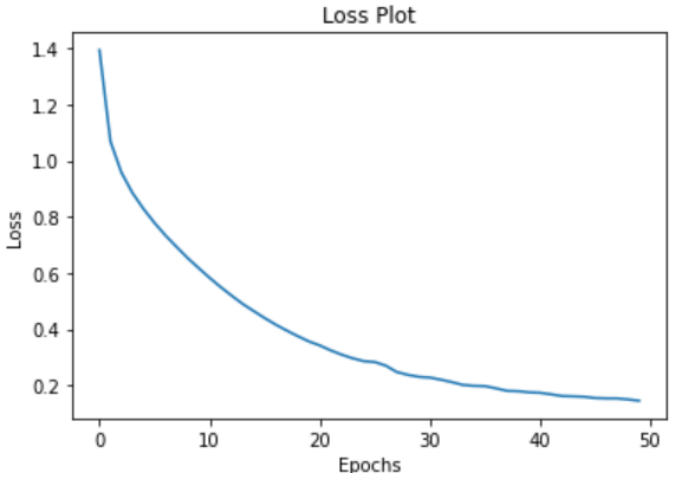
RESULTS – TRAINNG

S.No	Epoch	Loss	Accuracy
1.	1	139.4686	6.9699
2.	5	82.8560	12.1663
3.	10	61.7785	14.8151
4.	15	46.4258	17.4831
5.	20	35.6188	19.7863
6.	25	28.5543	21.4668
7.	30	22.9711	22.9470
8.	35	19.7586	23.9139
9.	40	17.4446	24.5974
10.	45	15.8576	25.1528
11.	50	14.4483	25.6063

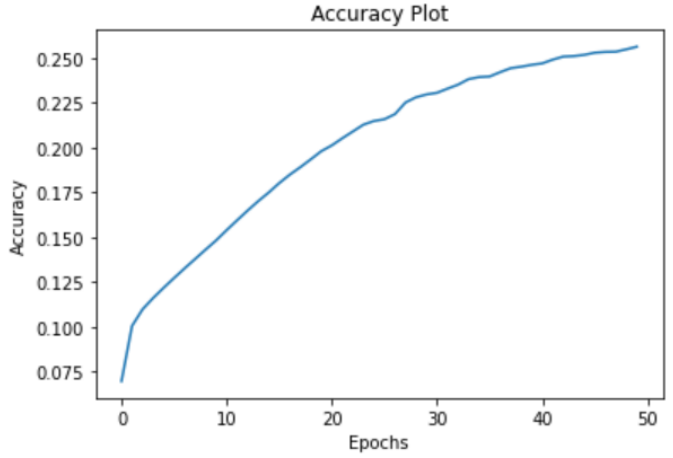
The loss is 14.5% at the end of 50 epochs.

RESULTS - PLOTS

LOSS PLOT



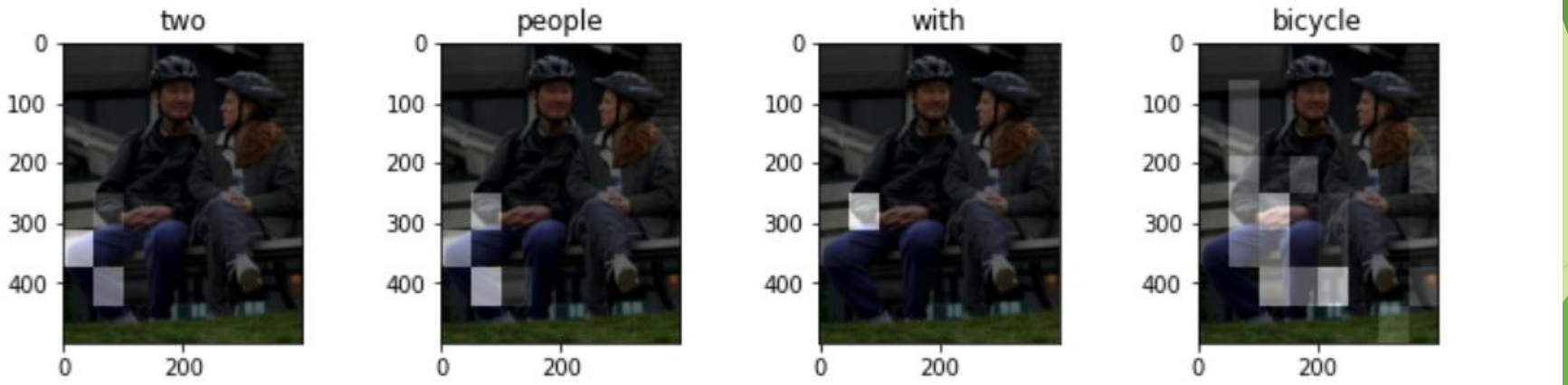
ACCURACY PLOT



RESULTS

INPUT

ATTENTION MECHANISM OUTPUT



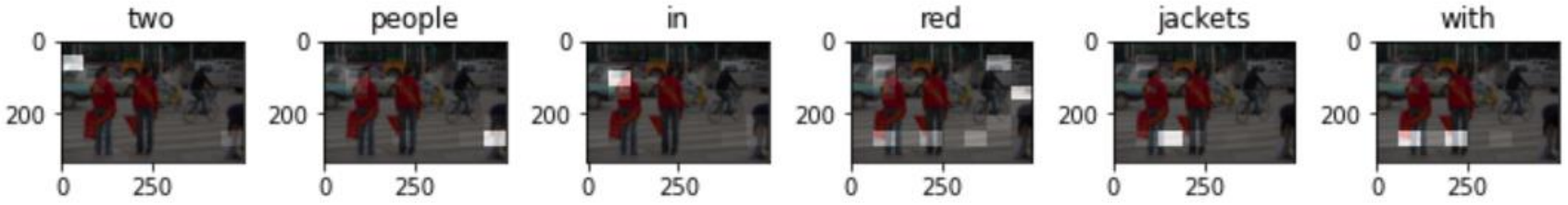
OUTPUT

```
Real Caption: <start> interracial couple looks at each other wearing biking helmets <end>
Prediction Caption: two people with bicycle helmets chatting on a bench
```

RESULTS

INPUT

ATTENTION MECHANISM OUTPUT



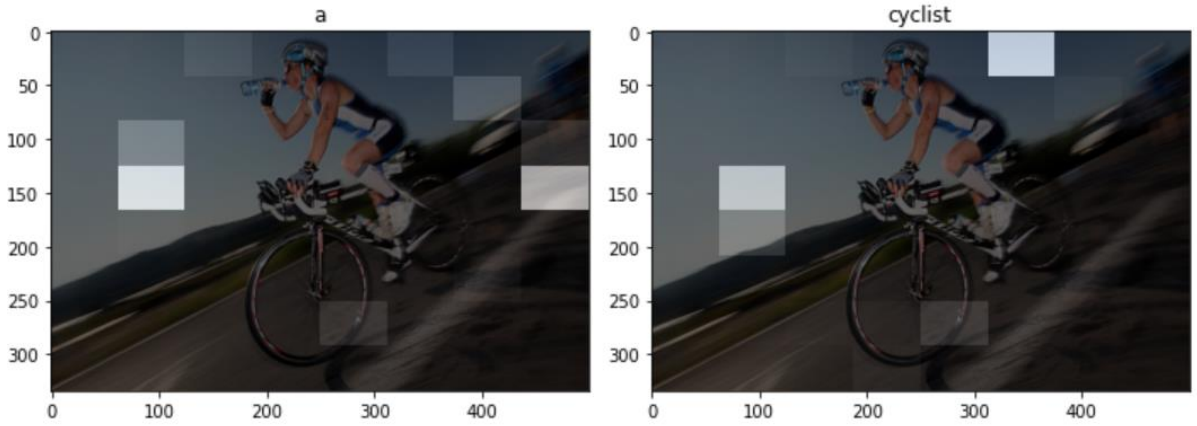
OUTPUT

```
Real Caption: <start> two women attempting to cross a busy street in china <end>
Prediction Caption: two people in red jackets with red sashes standing on the street
```

RESULTS

INPUT

ATTENTION MECHANISM OUTPUT

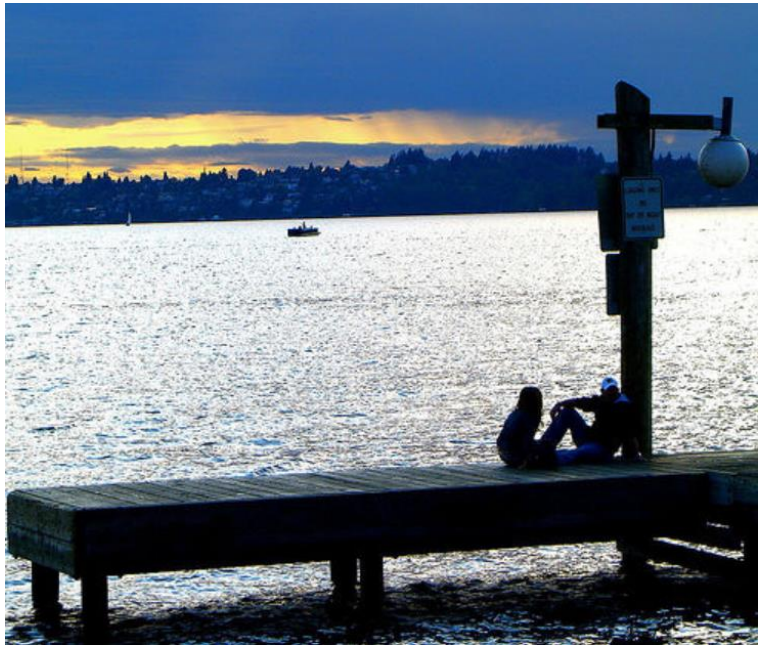


OUTPUT

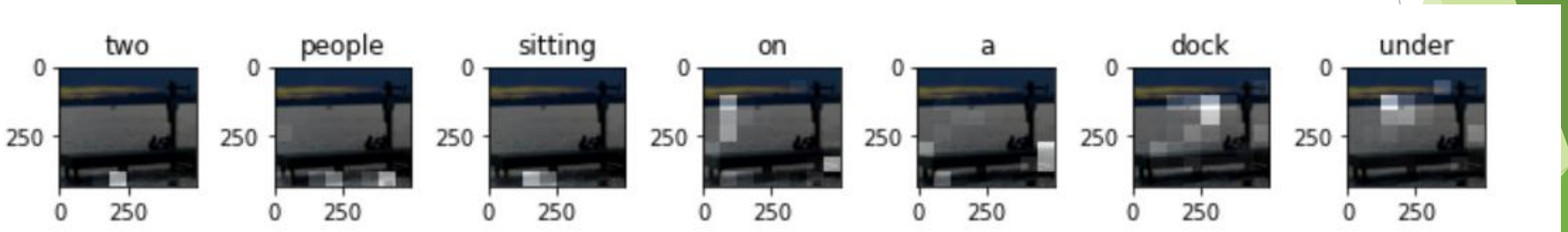
```
Real Caption: <start> a bicyclist is drinking out of a bottle of water while riding <end>
Prediction Caption: a cyclist drinks water
```

RESULTS

INPUT



ATTENTION MECHANISM OUTPUT

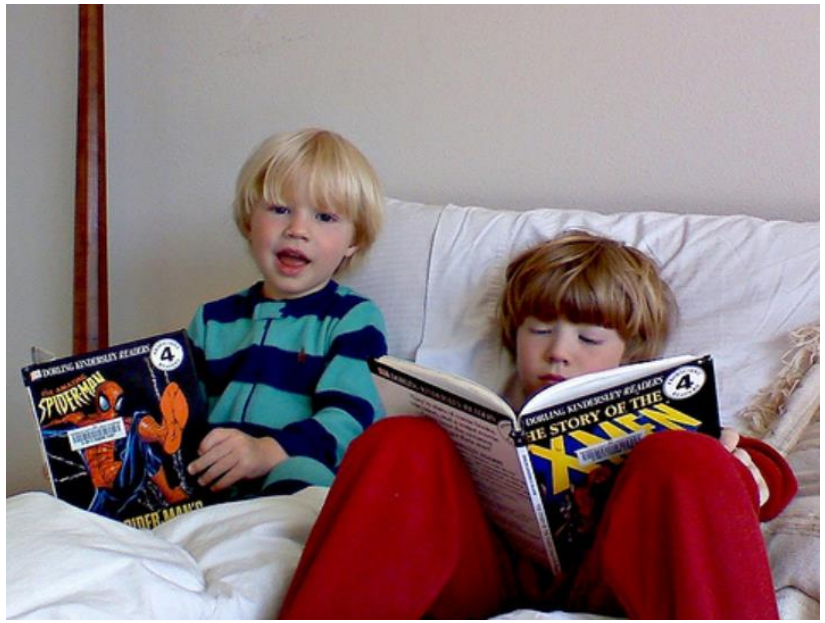


OUTPUT

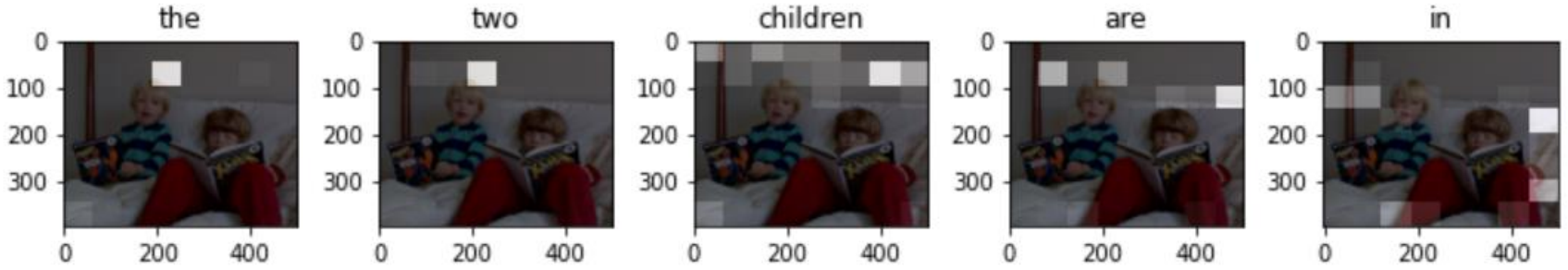
```
Real Caption: <start> silhouette of two people sitting on a dock at sunset <end>
Prediction Caption: two people sitting on a dock under a lamp post with clouds in the background
```

RESULTS

INPUT



ATTENTION MECHANISM OUTPUT



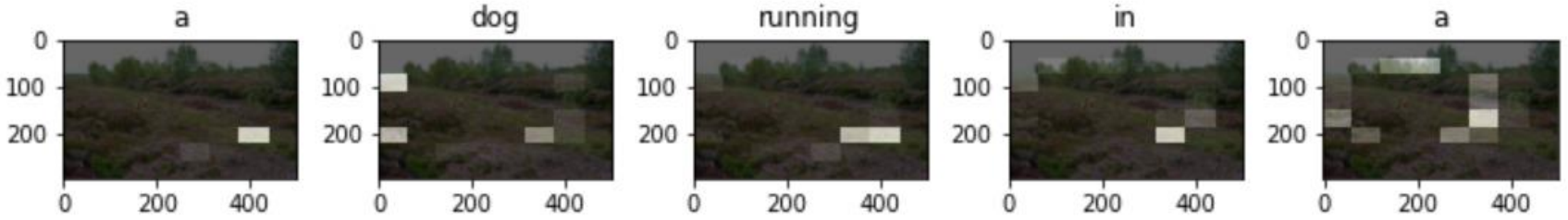
OUTPUT

```
Real Caption: <start> two young boys read comic books in bed <end>
Prediction Caption: the two children are in the bed with comic books
```

RESULTS

INPUT

ATTENTION MECHANISM OUTPUT



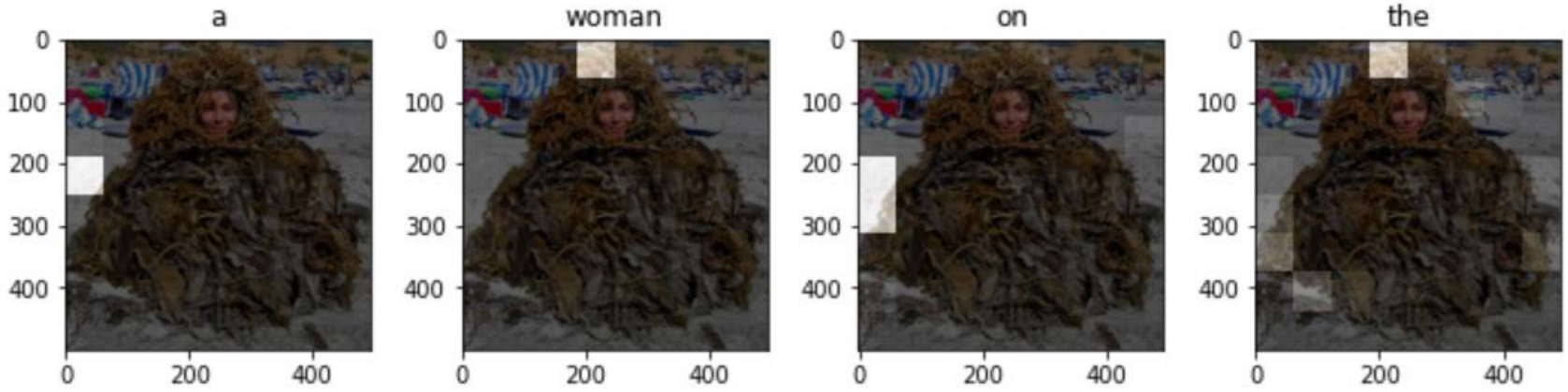
OUTPUT

```
Real Caption: <start> a brown dog running through the grass and flowers <end>
Prediction Caption: a dog running in a field of grass and flowers
```

RESULTS

INPUT

ATTENTION MECHANISM OUTPUT



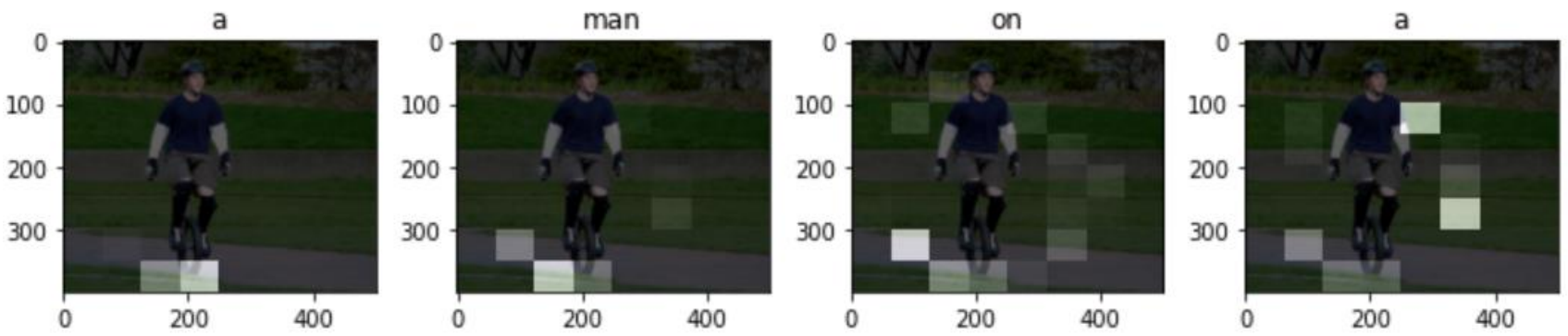
OUTPUT

```
Real Caption: <start> a young person is buried in a pile of seaweed on a beach <end>
Prediction Caption: a woman on the beach is buried in seaweed
```

RESULTS

INPUT

ATTENTION MECHANISM OUTPUT



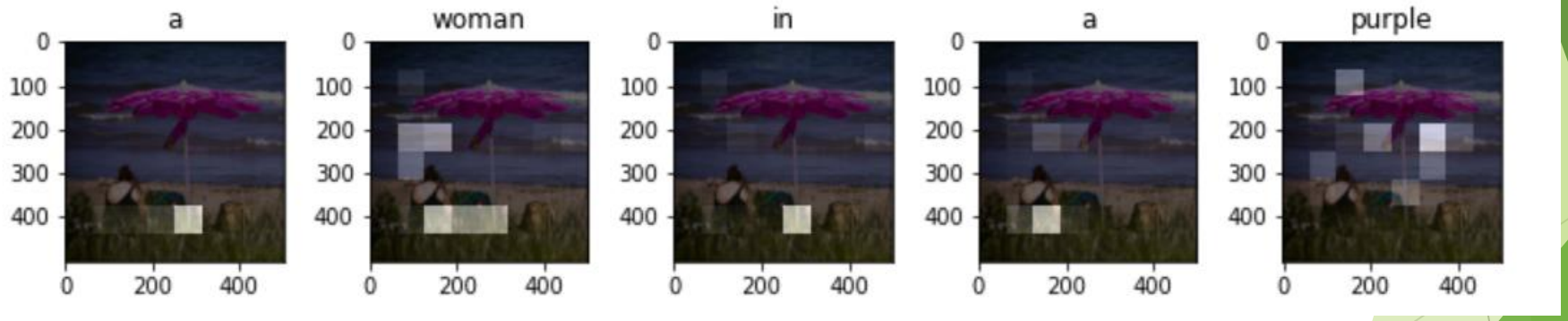
OUTPUT

```
Real Caption: <start> a young man rides a unicycle down a sidewalk <end>
Prediction Caption: a man on a unicycle down the background
```

RESULTS

INPUT

ATTENTION MECHANISM OUTPUT



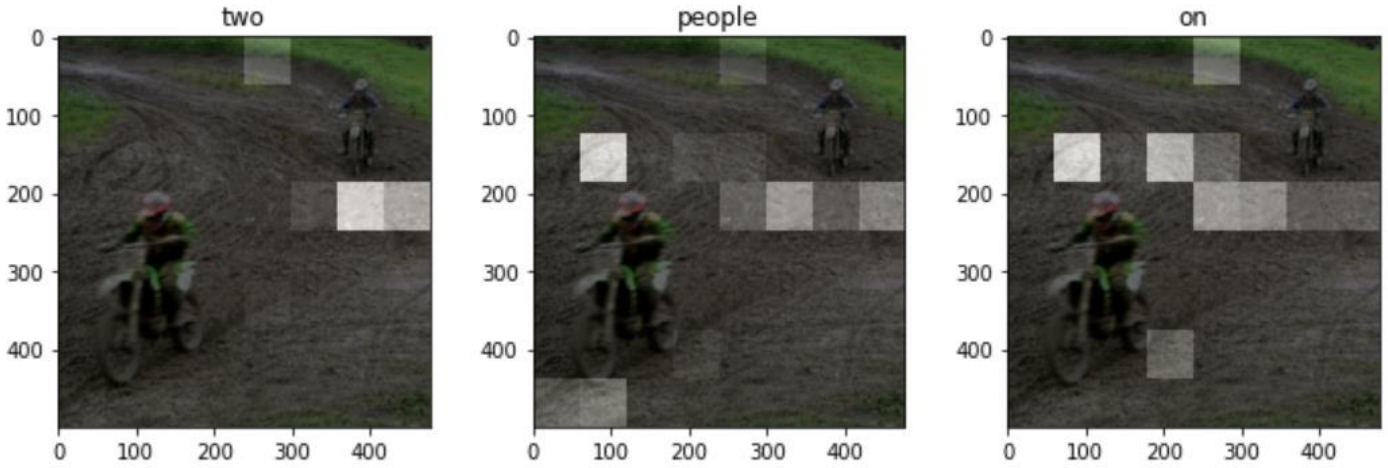
OUTPUT

```
Real Caption: <start> a girl sits on the beach under a bright pink sunshade <end>
Prediction Caption: a woman in a purple swimsuit looking at the beach
```

RESULTS

INPUT

ATTENTION MECHANISM OUTPUT



OUTPUT

```
Real Caption: <start> two dirt bike riders ride their dirt bikes <end>
Prediction Caption: two people on motorbikes through the mud
```

CONCLUSION

We have presented an attention-based image captioning model that can generate a caption for a given image. The model is based on a convolution neural network to encode an image into a fixed length vector representation. This is further used by attention layer to produce context vector which is then decoded by the use of the recurrent neural network. The obtained results are promising and competitive compared to results presented in the literature.

REFERENCE

- Towards Data Science - https://towardsdatascience.com/image-captions-with-attention-in-tensorflow-step-by-step-927dad3569fa
- Tensorflow Tutorial - https://www.tensorflow.org/tutorials/text/image_captioning