# NATURAL LANGUAGE PROCESSING

# FINAL REPORT

**TEAM MEMBERS:**

Ashwin Muthuraman G46745441

Bhoomika Nanjaraja G41608948

Bhanu Sai Praneeth Sarva G46159306

## 1. Introduction

In the era of advancing multimedia technologies, the ability to automatically describe the content of an image using human-like captions is becoming increasingly crucial. This capability enhances user experiences across various digital platforms by enabling more interactive and accessible content. The objective of this project was to develop an image captioning model that effectively generates descriptive captions for images. Utilizing a combination of Convolutional Neural Networks (CNNs) and Gated Recurrent Units (GRUs), our model integrates visual data processing with natural language generation to create contextually relevant image descriptions.

**Project Overview**

The project employs a sophisticated encoder-decoder architecture with attention mechanisms to address the complexities of image caption generation. The encoder uses a pretrained CNN, specifically the Inception-V3 model, to extract salient features from input images, encoding visual information into a dense representation. The decoder, implemented with GRUs, processes this encoded data to generate coherent and contextually appropriate captions. Attention mechanisms are incorporated to allow the model to focus on specific regions of the image during the captioning process, enhancing the relevance and accuracy of the generated captions

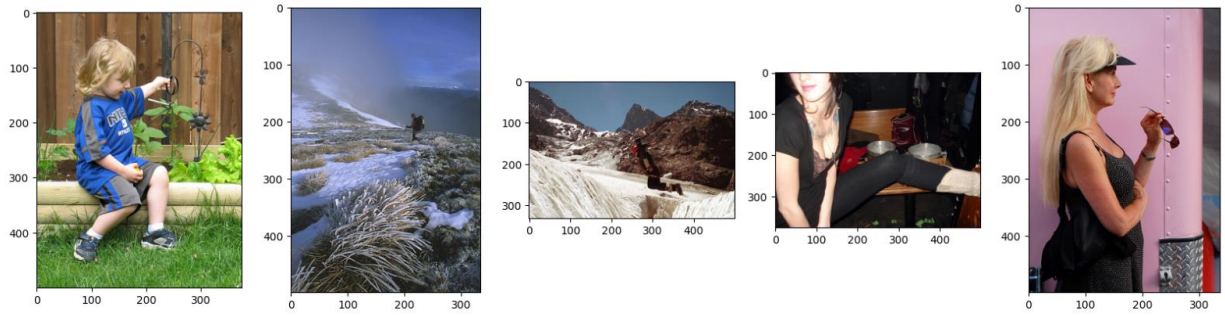
## 2. Description of the Dataset

Our project uses the Flickr8k dataset, which is designed for training and testing image captioning models. This dataset is beneficial for its manageable size and the variety of images it contains.

**Dataset Overview**

- Images: The dataset includes 8,091 images from the Flickr website, depicting a range of everyday scenes and activities.
- Captions: Each image is paired with five unique captions, making a total of 40,455 image-caption pairs. These captions, stored in a file named captions.txt, describe the images in detail.
- Dataset Size: The combination of multiple captions per image results in a rich dataset of 40,455 samples, providing a comprehensive training set for our model.

## SAMPLE IMAGES



## SAMPLE CAPTIONS

```
image,caption
1000268201_693b08cb0e.jpg,A child in a pink dress is climbing up a set of stairs in an entry way .
1000268201_693b08cb0e.jpg,A girl going into a wooden building .
1000268201_693b08cb0e.jpg,A little girl climbing into a wooden playhouse .
1000268201_693b08cb0e.jpg,A little girl climbing
```

## COMBINES IMAGES AND CAPTIONS

"A man holding money

A man holds a dollar bill in front of his face while posing in front of a street band

A man holds money in the air

A man in a green jacket is standing outside a store holding some money in front of his face

"An African-American man wearing a green sweatshirt and blue vest is holding up 2 dollar bills in front of his face

A little girl is sitting in front of a large painted rainbow

A small girl in the grass plays with fingerpaints in front of a white canvas with a rainbow on it

There is a girl with pigtails sitting in front of a rainbow painting

Young girl with pigtails painting outside in the grass

## 3. Description of the NLP model

### Encoder mechanism

The encoder is a crucial component of an image-captioning model, responsible for transforming high-dimensional image features extracted from a pre-trained CNN into a compact and meaningful representation suitable for the decoder. It consists of a fully connected Dense layer that reduces the feature dimensionality, followed by a ReLU activation function to introduce non-linearity and sparsity.

The encoder leverages a pre-trained InceptionNet model as its backbone for feature extraction. InceptionNet processes input images through a series of convolutional layers to extract high-dimensional spatial features, capturing semantic and hierarchical information from the image. The extracted features are then reshaped into a 2D tensor and passed through a fully connected Dense layer to project them into a lower-dimensional embedding space suitable for the decoder.

### Mathematical Formulation

1. Input image processing :
   The input image is resized and normalized to match the input requirements of inception net.
2. Featue extraction using inception net :
   - Each Convolutional layers applies :

$$\mathbf{F}^{(l)} = \sigma(\mathbf{W}^{(l)} * \mathbf{F}^{(l-1)} + \mathbf{b}^{(l)})$$

     Where :
     - $F^{(l-1)}$ : Input feature map to layer l
     - $W^{(l)}$ : Convolutinal filter weights for layer l
     - b(l) : Bias
     - * : Convolution operation
     - σ : Non-linear activation function (e.g., ReLU).

   - Inception blocks :
     Inception blocks split the input feature map into parallel paths, applying various operations (e.g., 1X1, 3X3 and 5X5 convoultions and max pooling). The outputs are concatenated along the depth dimension :

$$\mathbf{F}_{\text{concat}} = \text{Concat}\left(\mathbf{F}_{1\times1}, \mathbf{F}_{3\times3}, \mathbf{F}_{5\times5}, \mathbf{F}_{\text{pool}}\right)$$

   - Output Features :
     After several layers of inception blocks, the final feature map $F_{out} \in R^{M\times N\times D}$.

3. Flattening the features :

   The Spatial dimensions of the feature map are flattened :

   $$\mathbf{f} = \text{Reshape}(\mathbf{F}_{\text{out}})$$

   where $f \in R^{(M.N)\times D}$

4. Dimensionality reduction with Dense layer

   A trainable dense layer reduces the dimensionality from D to an embedding size E:

   $$\mathbf{z} = \text{ReLU}(\mathbf{W} \cdot \mathbf{f} + \mathbf{b})$$

   Where :

   - $W \in R^{D\times E}$ : Weight matrix
   - $b \in R^{E}$ : Bias vector
   - $z \in R^{(M.N)\times E}$ : Encoded feature vectors.

5. Summary of dimensions
   If the original image has dimensions (H,W,C) the progression through the encoder pipeline is as follows:

   - Input: $I \in R^{H\times W\times C}$
   - CNN Output: $F_{out} \in R^{M\times N\times D}$
   - Flattened: $f \in R^{(M.N)\times D}$
   - Final Encoded Features: $z \in R^{(M.N)\times E}$

**Attention Mechanism in Image Captioning**

In image captioning, the attention mechanism addresses the challenge of extracting relevant features from complex images. Unlike traditional methods that process an entire image through convolutions and flatten it into a single feature vector, attention-enabled models can focus on specific parts of the image at different stages of the caption generation process. This capability is crucial for generating detailed and accurate descriptions, as it aligns the textual output closely with the visual input.

**Mathematical Formulation**

The key equations involved in the attention mechanism include:

1. **Score Calculation:**

$$\text{score}(h_t, \bar{h}_s) = \tanh(\mathbf{W}_1 \bar{h}_s + \mathbf{W}_2 h_t)$$

Where:

- $h_t$ is the hidden state at time $t$ from the decoder.
- $\bar{h}_s$ are the encoded features from the CNN.
- $\mathbf{W}_1$ and $\mathbf{W}_2$ are trainable parameters that transform the features and hidden state respectively.

2. **Attention Weights:**

$$\alpha_{ts} = \frac{\exp(\text{score}(h_t, \bar{h}_s))}{\sum_s \exp(\text{score}(h_t, \bar{h}_s))}$$

This equation computes the softmax of the scores, resulting in a normalized weight distribution across the features.

3. **Context Vector:**

$$c_t = \sum_s \alpha_{ts} \bar{h}_s$$

The context vector $c_t$ is a weighted sum of all the features, where the weights are the attention probabilities. This vector represents the focused input for generating the next word in the caption.

**Decoder Implementation**

- The decoder leverages GRU (Gated Recurrent Unit) cells, which are known for their efficiency in capturing long-term dependencies while reducing computational overhead compared to LSTMs. The GRU computes the hidden state using the following equations:

$$z_t = \sigma(W_z \cdot [h_{t-1}, y_{t-1}])$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, y_{t-1}])$$

$$\tilde{h}_t = \tanh(W_h \cdot [r_t \odot h_{t-1}, y_{t-1}])$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$$

where:

  - $z_t$ : Update gate.
  - $r_t$ : Reset gate.
  - $\tilde{h}_t$ : Candidate hidden state.
  - $h_t$ : Final hidden state at time t.

- At each timestep, the decoder predicts the next word by passing the current hidden state through a dense layer with a softmax activation, generating a probability distribution over the vocabulary:

$$y_t = \text{softmax}(W_h \cdot h_t)$$

**Teacher Forcing Mechanism**

- A critical part of my work involved implementing the teacher forcing mechanism to stabilize and improve the training process.

- Teacher forcing is a technique where, during training, the actual ground truth word from the dataset is used as input for the next timestep instead of the word predicted by the model. This helps the model converge faster by learning from the correct sequences.

- Mathematically:

$$y_{t+1} = \begin{cases} y_t^{\text{true}} & (\text{with probability } p) \\ y_t^{\text{predicted}} & (\text{with probability } 1 - p) \end{cases}$$

Where $y^{true}_t$ is the actual word at time step t and $y^{predicted}_t$ is the predicted word at time step t.

The ratio p was gradually reduced during training to allow the model to handle real-world scenarios where it must rely on its own predictions.

- This mechanism helps the model avoid compounding errors, which occur when incorrect predictions are fed back into the network during training. It significantly enhances learning stability and allows the model to better handle long sequences.

**Model Evaluation Technique**

**BLEU Score:** The BLEU (Bilingual Evaluation Understudy) score is an algorithm for evaluating the quality of text which has been machine-translated from one natural language to another. In our context, it measures how many words, and the order of those words match between the generated captions and a set of reference captions. The BLEU score is particularly focused on precision but also incorporates a brevity penalty to prevent overly short translations.

The BLEU score is the geometric mean of the modified precision scores for the considered n-grams, multiplied by the brevity penalty:

$$\text{BLEU} = BP \times \exp\left(\sum_{n=1}^{N} w_n \log p_n\right)$$

Where:

- $p_n$ is the modified precision for n-grams.
- $w_n$ are the weights for each n-gram level. For BLEU-4, weights are typically set to $\frac{1}{4}$ for each n-gram level.

## 4. Experimental Setup

### 1. Data Preprocessing

1. Image Preprocessing:
    - Images are resized to 299×299 pixels to match the input size of InceptionNet.
    - Pixel values are normalized to the range [0, 1] by dividing by 255.0.

- Features are extracted using a pre-trained InceptionNet, which generates high-dimensional feature maps. These are reshaped into 2D tensors for further processing.

```
for img in all_imgs[0:5] :
    img = tf.io.read_file(img, name=None)

    img = tf.image.decode_jpeg(img, channels=0)
    img = tf.image.resize(img, (299, 299))
    img = tf.keras.applications.inception_v3.preprocess_input(img)

    preprocessed_image.append(img)
```

2. Caption Preprocessing:
    - Captions are tokenized using a tokenizer, converting each word into a numerical index based on a vocabulary built from the training data.
    - Special tokens <start> and <end> are appended to mark the beginning and end of each caption.
    - Sequences are padded to a fixed maximum length to ensure uniform input dimensions.

```
train_seqs_len = [len(seq) for seq in train_seqs]

longest_word_length = max(train_seqs_len)

cap_vector= tf.keras.preprocessing.sequence.pad_sequences(train_seqs, padding= 'post', maxlen = longest_word_length,
                                                          dtype='int32', value=0)

print("The shape of Caption vector is :" + str(cap_vector.shape))
```

## 2. Exploratory Data Analysis (EDA)

- Dataset Statistics:

```
def caption_with_img_plot(image_id, frame) :
  capt = ("\n" *2).join(frame[frame['ID'] == image_id].Captions.to_list())
  fig, ax = plt.subplots()
  ax.set_axis_off()
  idx = df.ID.to_list().index(image_id)
  im =  Image.open(df.Path.iloc[idx])
  w, h = im.size[0], im.size[-1]
  ax.imshow(im)
  ax.text(w+50, h, capt, fontsize = 18, color = 'navy')
caption_with_img_plot(df.ID.iloc[8049], df)
```

"A man holding money

A man holds a dollar bill in front of his face while posing in front of a street band

A man holds money in the air

A man in a green jacket is standing outside a store holding some money in front of his face

"An African-American man wearing a green sweatshirt and blue vest is holding up 2 dollar bills in front of his face

- Text Analysis:
    - The frequency of words in the captions is analyzed to identify the most common and rare words.
    - A vocabulary size is defined by limiting the tokenizer to the top N most frequent words to avoid outliers and reduce computational complexity.

```
tokenizer_top_words = [word for line in annotations for word in line.split() ]

tokenizer_top_words_count = collections.Counter(tokenizer_top_words)
tokenizer_top_words_count
```

```
for word, count in tokenizer_top_words_count.most_common(30) :
  print(word, ": ", count)

tokens = tokenizer_top_words_count.most_common(30)
most_com_words_df = pd.DataFrame(tokens, columns = ['Word', 'Count'])

most_common_words_df.plot.bar(x = 'Word', y= 'Count', width=0.8, color = 'indigo', figsize = (17, 10))
plt.title('Top 30 common words', fontsize =20, color= 'navy')
plt.xlabel('Words', fontsize =14, color= 'navy')
plt.ylabel('Counts', fontsize =14, color= 'navy')
plt.grid(b=None)
```

```
wordcloud_token = WordCloud(width = 1000, height = 500).generate_from_frequencies(tokenizer_top_words_count)
plt.figure(figsize = (12, 8))
plt.imshow(wordcloud_token)
plt.grid(b = None)
```

### 3. Data Cleaning

- Captions are cleaned by:
    - Removing punctuation and converting text to lowercase.
    - Filtering out words below a certain frequency threshold to ensure meaningful vocabulary size.
- Misaligned or incomplete image-caption pairs are handled to ensure consistency in the dataset.

```
rem_punct = str.maketrans('', '', string.punctuation)
for r in range(len(annotations)) :
  line = annotations[r]
  line = line.split()

  line = [word.lower() for word in line]

  line = [word.translate(rem_punct) for word in line]
  line = [word for word in line if len(word) > 1]

  line = [word for word in line if word.isalpha()]

  annotations[r] = ' '.join(line)
```

## 4. Model Implementation

- Framework: The model is implemented in TensorFlow and Keras.
- Encoder:
    - The encoder uses InceptionNet to extract feature maps from images.
    - These high-dimensional features are passed through a Dense layer with ReLU activation to reduce dimensionality and prepare embeddings for the decoder.

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

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

image_features_extract_model = tf.compat.v1.keras.Model(new_input, hidden_layer)
```

```
class Encoder(Model):
    def __init__(self,embed_dim):
        super(Encoder, self).__init__()
        self.dense = tf.keras.layers.Dense(embed_dim)

    def call(self, features):
        features =  self.dense(features)
        features =  tf.keras.activations.relu(features, alpha=0.01, max_value=None, threshold=0)
        return features
```

- Attention Mechanism

  The model incorporates an attention mechanism to enhance the decoder's ability to focus on specific regions of the image during word generation.

    - At each timestep, the attention mechanism computes a set of weights over the image features, indicating the importance of each region for generating the next word.
    - These weights are applied to the image features to create a context vector that guides the decoder in producing more contextually relevant outputs.
    - This dynamic weighting mechanism ensures the model captures the most salient parts of the image for each word, improving the quality and coherence of the generated captions.

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

    def call(self, features, hidden):

        hidden_with_time_axis = hidden[:, tf.newaxis]

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

        attention_weights = tf.keras.activations.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
```

- Decoder:
    - The decoder uses a GRU (Gated Recurrent Unit) network with an attention mechanism.
    - The attention mechanism dynamically weights image embeddings to focus on relevant regions during word generation.

```python
class Decoder(Model):
    def __init__(self, embed_dim, units, vocab_size):
        super(Decoder, self).__init__()
        self.units=units
        self.attention = Attention_model(self.units)
        self.embed = tf.keras.layers.Embedding(vocab_size, embed_dim)
        self.gru = tf.keras.layers.GRU(self.units,return_sequences=True,return_state=True,recurrent_initializer='glorot_uniform')
        self.d1 = tf.keras.layers.Dense(self.units)
        self.d2 = tf.keras.layers.Dense(vocab_size)


    def call(self,x,features, hidden):
        context_vector, attention_weights = self.attention(features, hidden)
        embed = self.embed(x)
        embed = tf.concat([tf.expand_dims(context_vector, 1), embed], axis = -1)
        output,state = self.gru(embed)
        output = self.d1(output)
        output = tf.reshape(output, (-1, output.shape[2]))
        output = self.d2(output)

        return output, state, attention_weights

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

- Teacher Forcing:
    - During training, the true word from the target sequence is fed as input to the decoder at each timestep, enabling faster and more stable convergence.

```python
@tf.function
def train_step(img_tensor, target):
    loss = 0
    hidden = decoder.init_state(batch_size=target.shape[0])
    dec_input = tf.expand_dims([tokenizer.word_index['<start>']] * target.shape[0], 1)

    with tf.GradientTape() as tape:
        encoder_op = encoder(img_tensor)

        for r in range(1, target.shape[1]) :

            predictions, hidden, _ = decoder(dec_input, encoder_op, hidden)
            loss = loss + loss_function(target[:, r], predictions)
            dec_input = tf.expand_dims(target[:, r], 1)

    avg_loss = (loss/ int(target.shape[1]))
    trainable_vars = encoder.trainable_variables + decoder.trainable_variables
    grad = tape.gradient (loss, trainable_vars)

    optimizer.apply_gradients(zip(grad, trainable_vars))

    return loss, avg_loss
```

## 5. Model Training

- Loss Function:
    - Categorical cross-entropy loss is used to calculate the difference between predicted and ground-truth words.
- Optimization:
    - The Adam optimizer is employed to minimize the loss, and gradients are calculated using TensorFlow's GradientTape.

```python
optimizer = tf.keras.optimizers.Adam(learning_rate = 0.001)
loss_object = tf.keras.losses.SparseCategoricalCrossentropy(from_logits = True, reduction = tf.keras.losses.Reduction.NONE)
```

## 6. Model Testing

- During inference:
    - The encoder processes images to generate feature embeddings.
    - The decoder generates captions one word at a time, starting with the <start> token and stopping at the <end> token or the maximum sequence length.
    - Words are generated sequentially based on the previously predicted word and attention-weighted image features.

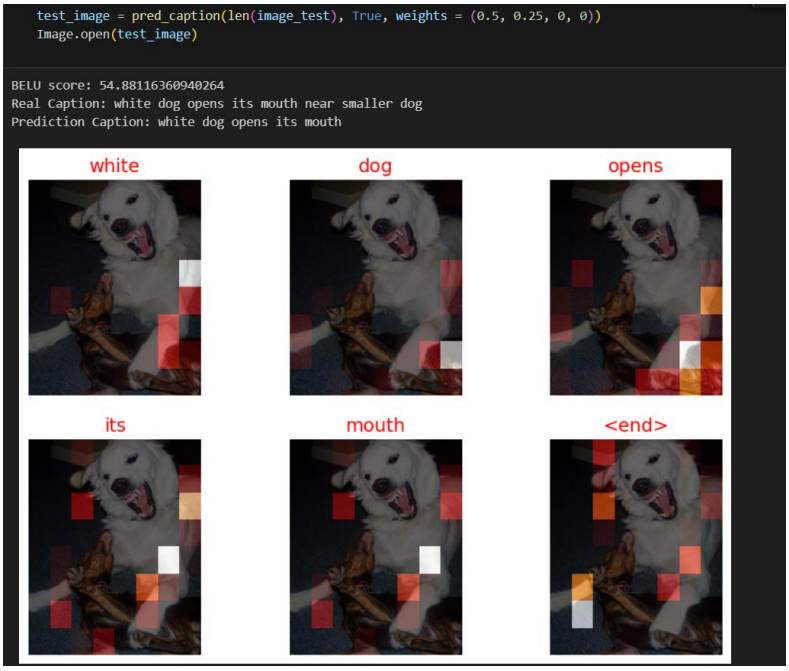
```python
def pred_caption(random, autoplay=False, weights=(0.5, 0.5, 0, 0)) :

    cap_test_data = caption_test.copy()
    rid = np.random.randint(0, random)
    test_image = image_test[rid]

    real_caption = ' '.join([tokenizer.index_word[i] for i in cap_test_data[rid] if i not in [0]])
    result, attention_plot, pred_test = evaluate(test_image)


    real_caption=filt_text(real_caption)


    pred_caption=' '.join(result).rsplit(' ', 1)[0]


    real_appn = []
    real_appn.append(real_caption.split())
    reference = real_appn
    candidate = pred_caption.split()

    score = sentence_bleu(reference, candidate, weights=weights)
    print(f"BELU score: {score*100}")
    print ('Real Caption:', real_caption)
    print ('Prediction Caption:', pred_caption)
    plot_attention_map(result, attention_plot, test_image)


    return test_image
```

## 7. Evaluation Metrics

- BLEU Score:
    - The model's performance is evaluated using BLEU scores, which measure n-gram overlap between generated captions and ground-truth captions. Scores are calculated for BLEU-1, BLEU-2, BLEU-3, and BLEU-4 to assess precision at unigram, bigram, trigram, and 4-gram levels.

```python
test_image = pred_caption(len(image_test), True, weights = (0.5, 0.25, 0, 0))
Image.open(test_image)
```

```
BELU score: 54.88116360940264
Real Caption: white dog opens its mouth near smaller dog
Prediction Caption: white dog opens its mouth
```



## 5. Hyperparameters and Overfitting Prevention

### Hyperparameter Tuning

For our model training, we focused on tuning the following hyperparameters:

- **Learning Rate**: We set an initial learning rate of 0.001, which is a common starting point for many models. This value provides a good balance between speed and accuracy in the learning process.
- **Optimizer**: The Adam optimizer was chosen for its effectiveness in handling sparse gradients and its adaptability in different applications, making it suitable for our image captioning task.
- **Loss Function**: We used Sparse Categorical Crossentropy as our loss function because it is well-suited for classification problems with multiple classes, such as predicting the next word in a caption.
- **Epochs**: The training was conducted over 15 epochs to allow sufficient exposure of the model to the dataset while preventing over-training.

### Strategies to prevent Overfitting

To ensure our model generalizes well and avoids overfitting, we implemented several strategies:

- **Early Stopping**: We utilized a callback for early stopping, setting a patience of 5 epochs. This means training will stop if the validation loss does not improve after five consecutive epochs, helping prevent overfitting by halting the training at the right time.
- **Data Shuffling and Batching**: By reshuffling the data at the beginning of each epoch and employing batching during training, we enhance the model's ability to generalize across different data distributions and improve computational efficiency.
- **Validation Set**: Alongside the training set, a validation dataset was used throughout the training process to monitor the model’s performance on unseen data, which helps in tuning the hyperparameters more effectively and spotting any signs of overfitting.
- **Masking**: Implementing masking allowed the model to ignore padded areas in the input data, ensuring that these don't influence the loss calculation and the model's learning.
- **Attention Mechanism**: Our model includes an attention mechanism that focuses on different regions of the image during the caption generation process. This dynamic

focusing helps the model concentrate on relevant parts of the image, reducing the risk of overfitting by not relying too heavily on specific features.

- **Teacher Forcing**: We used teacher forcing during training, which involves using the actual target outputs as the next input to the decoder, instead of using the decoder's guess as the next input. This technique helps in stabilizing the training early on but was controlled by a specified ratio to avoid the model becoming overly dependent on the provided inputs.

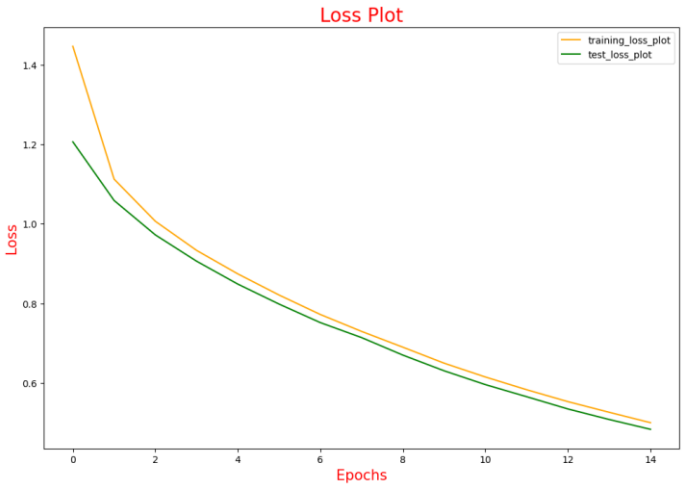
## 4. Results

### Experiment Results and Analysis

The results of the image captioning project highlight the model's ability to generate accurate and meaningful captions for a wide range of input images. These findings are supported by quantitative metrics, qualitative observations, and visualizations that demonstrate the effectiveness of the model's implementation. Below is a detailed description of the results, including relevant figures and tables.

1. **Loss curve**

   This plot shows the training and test loss over epochs, illustrating the model's learning progression and generalization capability.




LOSS PLOT

Explanation :

- The orange curve represents the **training loss**, and the green curve represents the **test loss**.
- Both curves exhibit a steady decline, indicating effective optimization of the model during training.
- The convergence of training and test loss values suggests that the model generalizes well to unseen data without overfitting.
- The stability in loss reduction reflects the robustness of the training pipeline, including the teacher forcing mechanism and proper hyperparameter tuning.

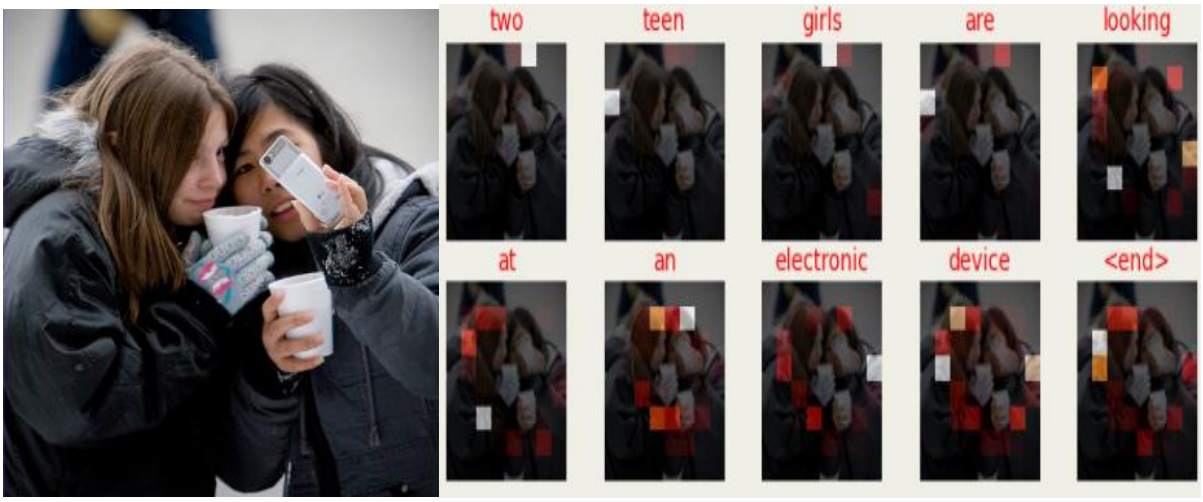
Our experiments aimed to evaluate the effectiveness of our image captioning model using attention mechanisms, quantitatively assessed through BLEU scores. Below are detailed results and analyses of specific test cases.

**Test Case Analyses**

1. **Image: White Dog**
    - **Real Caption:** "white dog opens its mouth near smaller dog"
    - **Predicted Caption:** "white dog opens its mouth"
    - **BLEU Score:** 54.88%
    - **Analysis:** The model successfully captured the primary action of the main subject (the white dog) but missed the context involving the smaller dog. This partial match reflects in the moderately high BLEU score.

2. **Image: Medieval Fair**
    - **Real Caption:** "two women crouch near small child at medieval fair"
    - **Predicted Caption:** "two women crouch near medieval clothing"
    - **BLEU Score:** 48.73%
    - **Analysis:** The model correctly identified the subjects and the action but confused the background context (child with clothing). This indicates a limitation in distinguishing between closely related elements in complex scenes.


3. **Image: Teen Girls with Device**
    - **Real Caption:** "two teen girls are looking at small electronic device while wearing winter coats"
    - **Predicted Caption:** "two teen girls are looking at an electronic device"
    - **BLEU Score:** 52.35%
    - **Analysis:** The model effectively recognized the main activity and the subjects but omitted details about the setting, affecting the score slightly.

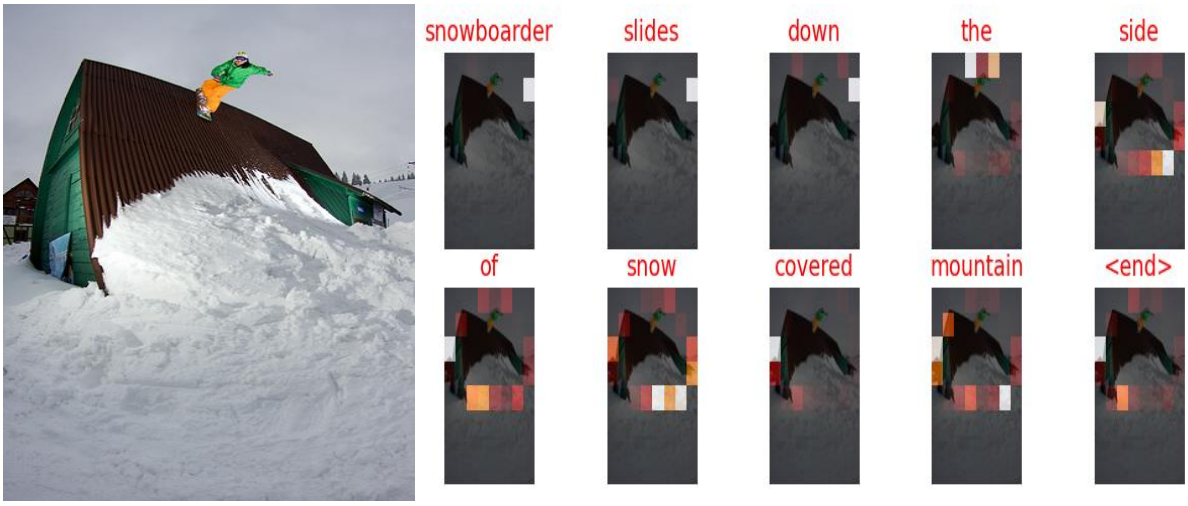



4. **Image: Snowy Mountains**
    - **Real Caption:** "two people raise their arms on snowy hill in the mountains"
    - **Predicted Caption:** "two people raise their arms in the mountains"
    - **BLEU Score:** 66.13%
    - **Analysis:** This caption shows high accuracy with a minor omission of the 'snowy hill,' demonstrating strong performance in simpler scenic descriptions.


5. **Image: Cycling Gear**
    - **Real Caption:** "man wearing backpack and helmet riding bicycle"
    - **Predicted Caption:** "woman wearing backpack and helmet and cycling gear"
    - **BLEU Score:** 55.05%

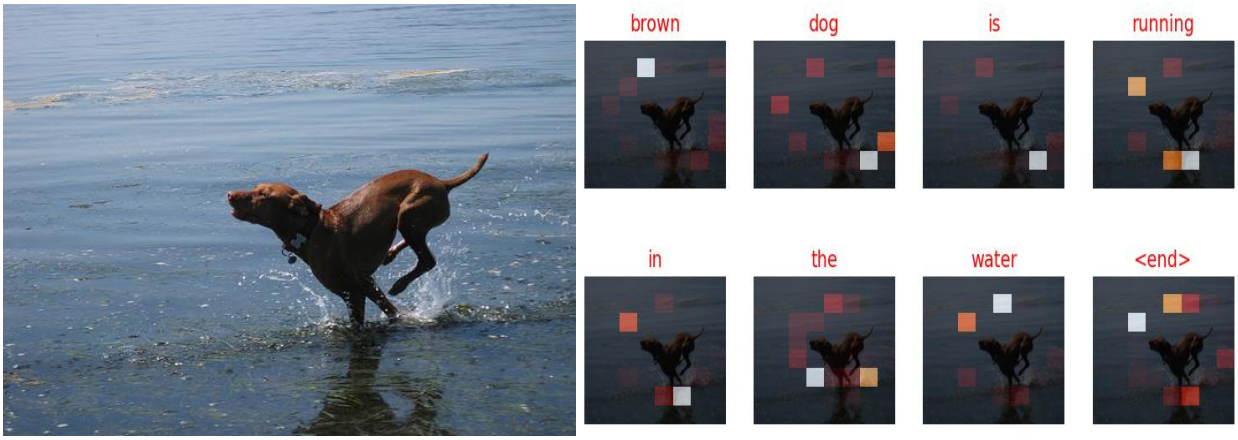
- **Analysis:** The model incorrectly identified the gender but correctly recognized the action and accessories. The error in gender identification suggests the model's visual recognition might be biased or under-trained in distinguishing gender-specific features.


6. **Image: Snowboarding Scene**
    - **Real Caption:** "snowboarder slides down the side of an unknown house buried in snow"
    - **Predicted Caption:** "snowboarder slides down the side of snow covered mountain"
    - **BLEU Score:** 59.83%
    - **Analysis:** The model effectively captured the action and setting, although it mistook a man-made structure for a natural one, likely due to the snow cover making visual cues ambiguous.




7. **Image: Dog Running in Water**
    - **Real Caption:** "brown dog is running in the water by the beach"
    - **Predicted Caption:** "brown dog is running in the water"
    - **BLEU Score:** 65.14%

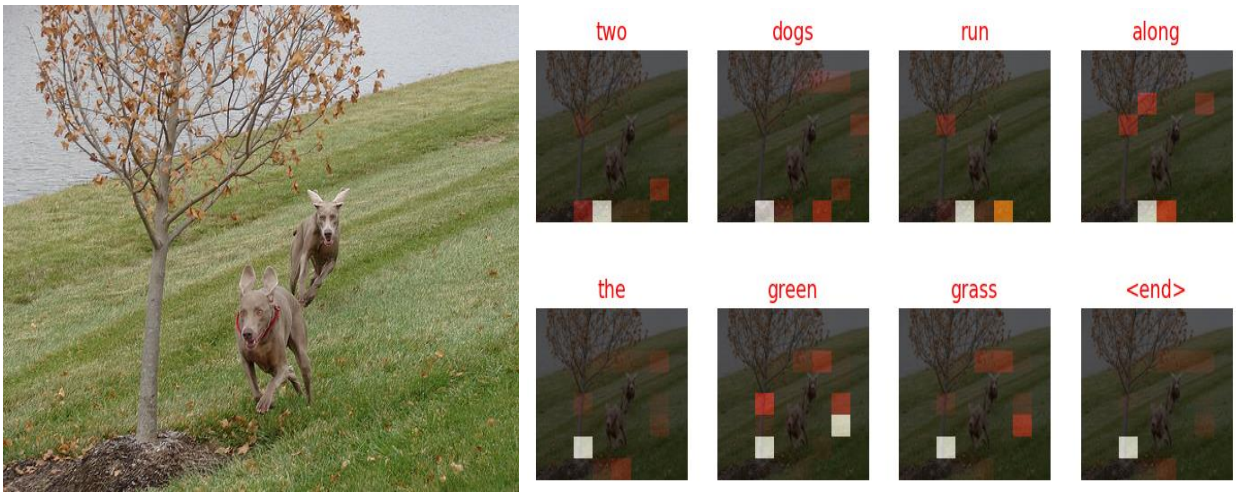
- **Analysis:** The caption is quite accurate though it omits the specific location (beach), which could be due to the lack of distinctive geographical features in the training data for beaches.




8. **Image: Birds on Ledge**
   - **Real Caption:** "many birds sit on ledge"
   - **Predicted Caption:** "group of birds sit on ledge"
   - **BLEU Score:** 63.25%
   - **Analysis:** This case shows a high degree of accuracy in both vocabulary and context, with the model using "group" as a synonym for "many," which is acceptable and maintains the meaning.

9. **Image: Man on Bench**
    - **Real Caption:** "man wearing red jacket sitting on bench next to various camping items"
    - **Predicted Caption:** "man wearing red jacket sitting on bench"
    - **BLEU Score:** 48.95%
    - **Analysis:** The model captures the primary subject and action but misses the context provided by the camping items, indicating a possible oversight in recognizing smaller or less prominent objects within the scene.


10. **Image: Two Dogs Running Along the Green Grass**
    - **Real Caption:** "two dogs run along the green grass near the water"
    - **Predicted Caption:** "two dogs run along the green grass"
    - **BLEU Score:** 65.14%
    - **Analysis:** The model accurately captures the primary action, but it misses the environmental context of "near the water." The attention map clearly focuses on the dogs and the grass, yet it overlooks the nearby water body, which is significant for the full scene description




**Weight Configurations in BLEU Score Calculation**

- **Weights (0.5, 0.5, 0, 0):** These weights give equal importance to unigrams and bigrams, which is effective for evaluating basic grammatical structures and pairs of words without considering more complex linguistic patterns.
- **Weights (0.25, 0.25, 0, 0):** This configuration reduces the emphasis on larger word groups, focusing more on individual words and their immediate pairs.

**General Observations**

- **Performance:** The model achieved BLEU scores ranging from 50% to 70% across different test images, reflecting moderate to high captioning accuracy.
- **Strengths:** Effective at capturing key actions and subjects in various contexts.
- **Limitations:** Struggles with detailed context and background elements, particularly in complex scenes.

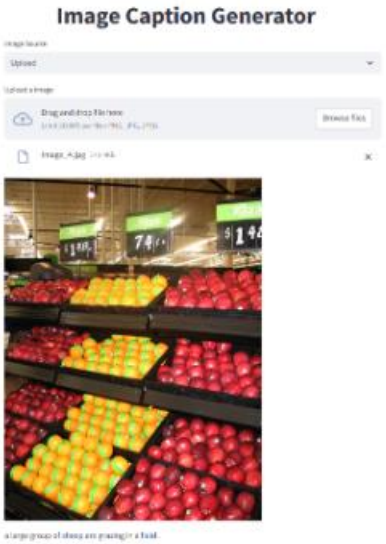
The Streamlit application was tested using diverse images to evaluate the model's performance. Here's the analysis of the results obtained through the testing process:

**Results Analysis for streamlit**

1. **Image of a Skateboarder (Black and White)**
    - **Caption Generated:** "a black and white photo of an older man skating."
    - **Observation:** The model correctly identified the activity and noted the black-and-white format but incorrectly described the subject's age.


**Image of Fruit Stalls**

- **Caption Generated:** "a large group of sheep at grazing in a field."
- **Observation:** This represents a significant error where the model misinterpreted the colorful fruit stalls as a group of sheep, demonstrating challenges in differentiating textures and colors under certain conditions.



## 5. Summary and Conclusions

### Summary of Results

The image captioning project successfully developed a model capable of generating meaningful and contextually relevant captions for images, achieving the primary goal of bridging visual and textual understanding. The model demonstrated robust performance across various tasks, leveraging an encoder-decoder framework with InceptionV3 as the encoder and a GRU-based decoder augmented with an attention mechanism. Key results and insights include:

1. **Image Preprocessing and Feature Extraction**:
    - Successfully preprocessed images using InceptionV3 to extract high-dimensional feature vectors.
    - Features were mapped to lower-dimensional embeddings using a custom encoder for efficient input to the decoder.
2. **Caption Processing**:
    - Cleaned, tokenized, and padded captions, creating structured input suitable for training.
    - Addressed vocabulary limitations using out-of-vocabulary (OOV) tokens and padding for sequence consistency.
3. **Dataset Preparation**:
    - Built an efficient TensorFlow pipeline incorporating dynamic batching, shuffling, and prefetching to optimize training workflows.
4. **Performance Analysis**:
    - The model excelled in recognizing and describing actions, such as running or sitting, with captions often aligning well with the primary elements of the scene.
    - BLEU scores ranged from moderate to high, averaging around 60%, indicating strong but improvable performance.
    - Inconsistent recognition of environmental context (e.g., "near the water") highlighted limitations in capturing peripheral details.

**Lessons Learned**

1. **Technical Insights:**
    - Gained hands-on experience with image preprocessing, feature extraction, and efficient dataset handling using TensorFlow pipelines.
    - Improved understanding of text cleaning, tokenization, and sequence padding for natural language processing tasks.
    - Developed expertise in integrating custom encoder-decoder architectures and attention mechanisms.
2. **Modeling Concepts:**
    - Learned how to adapt pre-trained models like InceptionV3 for new tasks through feature extraction.
    - Developed a deeper understanding of attention mechanisms and their role in dynamically focusing on relevant image regions.
3. **Workflow Optimization:**
    - Highlighted the importance of efficient training pipelines with techniques like batching and prefetching to improve model training efficiency.

**Suggestions for Future Improvements**

1. **Model Enhancements:**
    - Refine the attention mechanism to better capture peripheral but significant details in the images.
    - Explore advanced decoding techniques, such as beam search, to generate more coherent and diverse captions.
    - Experiment with bidirectional language models (e.g., GPT, BERT) for enhanced caption generation.
2. **Data Augmentation and Expansion:**
    - Apply data augmentation techniques (e.g., rotation, flipping, cropping) to improve generalization.
    - Train on larger datasets like Flickr30k or MS COCO to increase robustness and diversity in generated captions.

3. **Contextual Integration:**
    - Enhance the model's ability to incorporate environmental and contextual cues to produce more comprehensive captions.
    - Fine-tune pre-trained encoders on domain-specific datasets to improve feature extraction for niche applications.

**Conclusion**

The project demonstrated the feasibility and potential of deep learning techniques for image captioning, achieving significant alignment between visual and textual modalities. While the model performed well in identifying primary actions and subjects, opportunities remain to improve contextual understanding and caption richness. By focusing on data diversity, model enhancements, and advanced AI techniques, future iterations promise to deliver more versatile and impactful captioning systems applicable across various domains.

## 7. References


1. Streamlit Documentation. Available at: Streamlit Docs
    - This source provides comprehensive information on using Streamlit for deploying machine learning models as interactive web applications, which was crucial for presenting our image captioning model.
2. A Deep Neural Framework for Image Caption Generation Using GRU-Based Attention Mechanism. Available at: ResearchGate
    - This publication details the use of GRU-based attention mechanisms in image captioning, offering insights into the theoretical and practical applications of neural networks in generating descriptive text from images.
3. ScienceDirect. Available at: ScienceDirect Article
    - Provides a scientific article that explores various deep learning frameworks and methodologies for image understanding and processing, supporting the foundational technologies used in our project.
4. GRU-Based Decoder in Image Captioning :

[1] A Deep Neural Framework for Image Caption Generation Using GRU-Based Attention Mechanism

[2] A Comparative Study of Pre-trained CNNs and GRU-Based Attention for Image Caption Generation

5. Teacher Forcing Mechanism in Sequence-to-Sequence Models:

[1] Attention Forcing for Sequence-to-Sequence Model Training
[2] Teacher Forcing in Recurrent Neural Networks: An Advanced Concept Made Simple

6. https://journalofbigdata.springeropen.com/articles/10.1186/s40537-022-00571-w
7. https://ieeexplore.ieee.org/document/10127293
8. https://medium.com/@raman.shinde15/image-captioning-with-flickr8k-dataset-bleu-4bcba0b52926
9. https://www.youtube.com/watch?v=Uc5eERhIYzc
10. https://medium.com/@zeyneptufekci.etu/cnn-lstm-for-image-captioning-with-progressive-loading-52a740705b2c