Image Captioning Using CNN and LSTM Models

Github: https://github.com/Abkhan45/image_captioning_using_CNN_and_LSTM.git

1. Introduction

The goal of this project is to build a system that can look at an image and generate a sentence that describes it.
To do this, we use two types of models:

1. A CNN (Convolutional Neural Network) to understand the image.
2. An LSTM (Long Short-Term Memory) network to generate the text caption.

The report explains how the data was processed, how the models were built, how they were trained, and how well they performed.

2. Data Processing

2.1 Image Processing

We created a fast and clean way to load and prepare the images using tf.data.
The steps were:

- Read the image file
- Resize it to a fixed size (for example 224×224)
- Convert pixel values to the range [0,1]
- Batch the images
- Prefetch to make training faster

This makes sure all images are the same shape and easy for the CNN to learn from.

2.2 Caption Processing

For captions (text), we used the TextVectorization layer.

Steps:

- Clean and standardise the text
- Build a vocabulary using encoder.adapt()
- Turn each word into a number
- Pad all sequences so they have the same length
- Create batches of (image, caption) pairs

This made the text ready for the LSTM model.

3. Model Definition

3.1 CNN for Images: We built the CNN using the model summary that was given in the assignment.
The CNN included:

- Convolution layers
- ReLU activation

- MaxPooling
- Flatten
- Dense layer for final image features

The CNN learns what objects look like by finding shapes, edges, and textures in the images.

3.2 LSTM Model for Captions

The caption model had:

1. Embedding layer – turns numbers into word vectors
2. LSTM layer – learns the order of words in sentences
3. Dense layer – predicts the next word

This model tries to create a full sentence by predicting one word at a time.

4. Training the Models

We used callbacks such as:

- EarlyStopping
- ModelCheckpoint
- TensorBoard

Both models were trained for a set number of epochs.
During training:

- The loss went down, meaning the model was learning.
- After some point, the improvement slowed down.
- The caption model started to overfit a little (training loss much lower than validation loss).

This is common when the dataset is small.

5. Evaluation of the Models

The model worked well when:

- the image had one clear object
- the background was simple

Example:
Dog image → "A dog is sitting on the grass."
This shows the CNN understood the main object.

The model struggled when:

- many objects were in the image
- the image was crowded or complex

Example:
Busy street → caption said something generic like

"A group of people are walking."
This means the model created a safe guess instead of a specific caption.

5.2 General Performance

Training and validation curves showed that:

- The model learned basic patterns
- The captions were simple
- Some predictions were wrong but still meaningful

This is normal for a small CNN + LSTM model without attention.

6. Discussion

6.1 Why the Model Works Like This

- CNN is small
- Vocabulary is limited
- LSTM is limited
- No attention mechanism

6.2 Strengths

- Works well for simple images
- Sentences are mostly correct in grammar
- Data pipeline is efficient
- Training is smooth and fast

6.3 Weaknesses

- Captions are often too general
- Missing details
- Hard scenes confuse the model
- No attention → cannot focus on important areas

7. Conclusion

In this project, we successfully built:

- an image processing pipeline
- a caption processing pipeline
- a CNN for image features
- an LSTM model for generating captions

The model produced reasonable captions for simple images but struggled with complex ones.
The results show that the model learned basic relationships between images and sentences, but it needs more advanced methods (like attention or Transformers) to produce high-quality captions.