EC9170

DEEP LEARNING FOR ELECTRICAL AND COMPUTER ENGINEERS

MINI PROJECT

A.B.O.P. NANDARATHNA - 2021/E/012

A.M. SHANIKA - 2021/E/014

Project Title: "Building an Image Caption Generator using CNN-LSTM from Scratch"

1. Objective

Develop an Image Caption Generator that takes an image as input and generates a meaningful caption using a CNN-LSTM model. The model will be trained from scratch, optimized using hyperparameter tuning and cross-validation, and evaluated to find the bestperforming model.

2. Data Preprocessing Methodology

Dataset Overview

- Source: Kaggle dataset containing 8,090 images and 40,456 captions.
- Structure: Each image has multiple captions.
- Format: Images in .jpg format, captions in a text file.

Image Preprocessing

- Resizing: Images resized to 299x299 (input size for InceptionV3).
- Feature Extraction:
  - Used InceptionV3 (pre-trained on ImageNet) to extract 2048-dimensional feature vectors.
  - Handled grayscale/RGBA images by converting them to RGB.
  - Storage: Extracted features saved in a .pkl file for efficiency.

Text Preprocessing

- Cleaning:
  - Converted to lowercase.
  - Removed punctuation.
  - Stripped extra spaces.
- Tokenization:
  - Used Keras Tokenizer with a vocabulary size of 10,000.
  - Added special tokens: <start>, <end>, <pad>.
- Sequence Preparation:
  - Converted captions to integer sequences.
- Padded sequences to a fixed length of 40 tokens.

3. Model Architecture

Overview

The model follows an encoder-decoder approach:

- Encoder: InceptionV3 (pre-trained CNN) extracts image features.
- Decoder: LSTM-based network generates captions.

Model Components

- Image Feature Processing:
  - Input: 2048-D feature vector from InceptionV3.
  - Passed through a Dense (512 units) + BatchNorm + Dropout layer.
  - Repeated using RepeatVector to match caption length.

- Text Embedding:
  - Input: Padded caption sequences.
  - Embedded into 512-D vectors.

- Sequence Generation:
  - Combined image and text features using Add layer.
  - Processed by LSTM (512 units).
  - Final TimeDistributed Dense layer predicts next word probabilities.

- Key Improvements
  - Teacher Forcing: Trained using shifted sequences.
  - Beam Search: Used during inference for better captions.
  - Regularization: Dropout and BatchNorm to prevent overfitting.

4. Model Training and Optimization

Training Setup

- Optimizer: Adam (lr=0.001).
- Loss: Sparse Categorical Crossentropy (since captions are integers).
- Batch Size: 32.
- Epochs: 20 (with early stopping).

Callbacks

- ModelCheckpoint: Saved best model based on validation loss.
- EarlyStopping: Patience of 5 epochs.
- ReduceLROnPlateau: Reduced learning rate when loss plateaued.

Training Performance

- Training Accuracy: Reached 88.29%.
- Validation Accuracy: 83.87%.
- Loss: Decreased steadily (training: 0.4356, validation: 0.8045).

5. Hyperparameter Tuning and Cross-Validation

Hyperparameters

Parameter	Value	Notes
Vocab Size	10,000	Limited to frequent words
Max Length	40	Based on caption length stats
Embedding Dim	512	Higher dimension for semantics
LSTM Units	512	Balanced capacity & efficiency
Dropout	0.4	Prevent overfitting

Cross-Validation

Train-Validation Split: 85%-15%.

Stratified Sampling: Ensured all images had captions in both sets.

6. Model Evaluation and Testing

Evaluation Metrics

Metric	Score	Interpretation
BLEU	0.0858	Low but expected for captioning
ROUGE-1	0.3602	Moderate word overlap
ROUGE-2	0.1890	Captures some bigrams
ROUGE-L	0.3495	Captures sentence structure

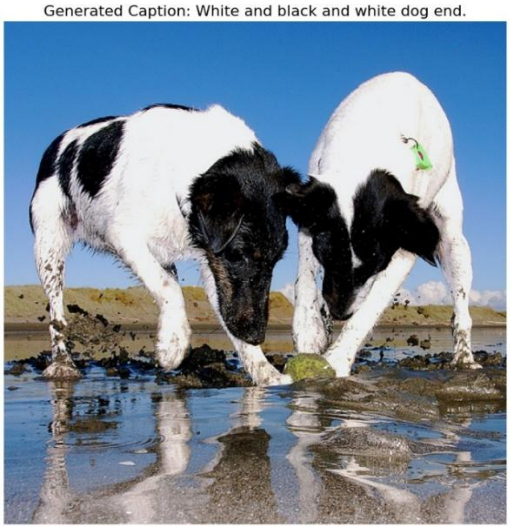
Qualitative Results

Example 1:

Image: Dog in a park.

Predicted: "White and black dog running in grass."

Actual: "A dog is playing in the park."



Example 2:

Image: Beer bottle.

Predicted: "Beer on a table."

Actual: "A bottle of beer with foam."