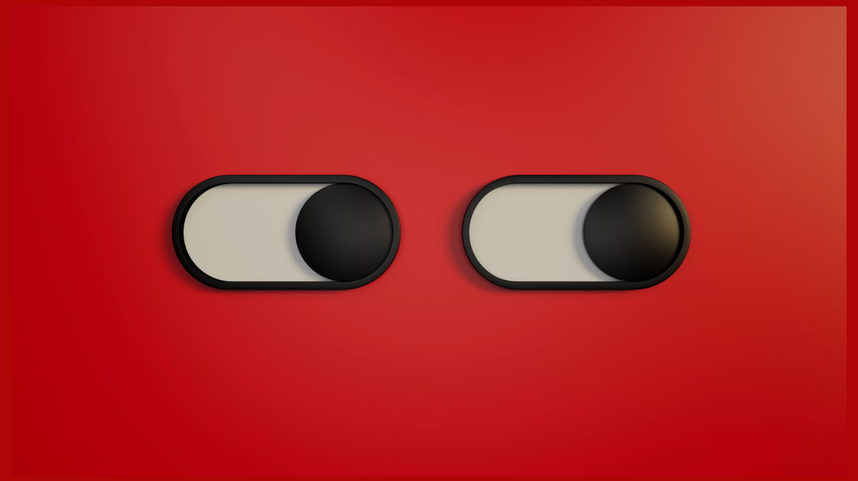
Image Caption Generation



Optimized Image Captioning System with CNN-RNN Architecture and LSTM/GRU/LLM(blip) Models

By
Shrey Patel
Aishwarya Patil
Shiva Teja Dasi

Content

Introduction

This project employs a CNN-RNN architecture with pre-trained CNNs for feature extraction and RNNs (LSTM/GRU) for image captioning, while also integrating Large Language Models (LLMs) to evaluate various captioning methods.

Aim

Develop a memory-efficient image captioning system using CNNs for feature extraction and LSTM/GRU with attention for caption generation, integrating LLMs for quality evaluation.

Objective

High-Performance Modeling:
Use pre-trained CNNs like Inception V3 or DenseNet for efficient feature extraction.

Implement LSTM and GRU-based RNNs with attention for context-aware caption generation.

Memory Optimization:
Employ mixed precision training and gradient checkpointing to reduce memory usage.
Optimize data loading and embedding caching to improve GPU utilization.

Robust Training Recovery:
Enable automatic checkpointing and crash recovery for uninterrupted training.

Comprehensive Evaluation:
Evaluate model with BLEU, CIDEr, METEOR, ROUGE, and semantic similarity metrics.
Benchmark performance against LLM-generated captions.

Visualization:
Visualize attention maps, embedding spaces, and resource utilization to gain insights into model behavior and performance.

Scope

Integrating Large Language Models (LLMs):

Leverage advanced LLMs to benchmark and enhance the image captioning system for more accurate and contextually relevant descriptions.

Advanced Visualization Techniques:

Implement PCA and t-SNE to visualize high-dimensional feature embeddings, providing insights into model performance by revealing patterns and relationships in the data.

Dynamic Hyperparameter Tuning:

Adapt training parameters in real-time based on performance metrics to optimize learning efficiency, enabling continuous improvement without manual intervention and reducing training time.

Scalability Considerations:

Scale the system for larger datasets like MSCOCO by implementing multi-GPU support with DDP-PyTorch and efficient resource management for optimal performance.

Process Overview

Data Preparation:

datasets (Flickr8K, Flickr30K, MSCOCO) All three !
Parse captions and cache image embeddings for efficient training.

Model Development:

Use pre-trained CNNs for feature extraction.
Develop LSTM and GRU caption generators with attention.
Integrate LLMs for comparative captioning analysis.

Training Optimization:

Apply mixed precision training and gradient checkpointing.
Implement checkpointing for recovery and adjusting hyperparameters as needed.

Evaluation & Visualization:

Calculating evaluation metrics and visualize attention maps, embeddings, and resources.
Conduct comparative analysis of model performances.

Data Preparation:

Image Preprocessing:

Resizing images to match CNN input sizes.
Normalization using standard mean and standard deviation.

Caption Preprocessing:

Tokenization and building a vocabulary.
Removing rare words below a certain threshold to limit vocabulary size.

Data Integrity Verification:

Ensuring all images have corresponding captions.
Standardizing filenames to avoid mismatches.

04

Technicals

- **Model Architecture:**
  - **Convolutional Neural Networks (CNNs):**
    - Pre-trained models like **Inception V3** and **DenseNet121** for image feature extraction.
  - **Recurrent Neural Networks (RNNs):**
    - **LSTM** and **GRU** models for sequence generation.
    - Integrated with **attention mechanism** (Bahdanau) to focus on specific image regions.
  - **Large Language Models (LLMs):**
    - Integration of models like **BLIP** for advanced caption generation and comparison.

- **Memory and Performance Optimization:**
  - **Mixed Precision Training:**
    - Utilizing both FP16 and FP32 to reduce memory usage and speed up training.
  - **Gradient Checkpointing:**
    - Saving memory by recomputing certain layers during backpropagation.
  - **Caching Embeddings:**
    - Storing image embeddings to avoid redundant computations.

- **Training Continuity:**
  - **Checkpointing:**
    - Regularly saving model state to recover from crashes.
  - **Crash Recovery Mechanisms:**
    - Automatic resumption from the last saved state.

```
[2024-10-29 08:02:32] INFO - Epoch 2/10
[2024-10-29 08:05:00] INFO - Rank 0: Epoch 2, Batch 100/94, Loss=3.8720
[2024-10-29 08:05:00] INFO - Rank 1: Epoch 2, Batch 100/94, Loss=3.8754
[2024-10-29 08:05:01] INFO - Epoch 2, Average Loss: 3.8737
[2024-10-29 08:05:01] INFO - Adjusting learning rate to 5e-05
[2024-10-29 08:05:01] INFO - Checkpoint saved at checkpoints/flickr8k/checkpoint_LSTM_epoch_2.pth

[2024-10-29 08:05:02] INFO - Epoch 3/10
[2024-10-29 08:07:30] INFO - Rank 0: Epoch 3, Batch 100/94, Loss=3.4567
[2024-10-29 08:07:30] INFO - Rank 1: Epoch 3, Batch 100/94, Loss=3.4590
[2024-10-29 08:07:31] INFO - Epoch 3, Average Loss: 3.4579
[2024-10-29 08:07:31] INFO - Checkpoint saved at checkpoints/flickr8k/checkpoint_LSTM_epoch_3.pth
```

Distributed Data Parallel

Distributed Data Parallel (DDP) trains machine learning models across multiple devices (GPUs/nodes) to accelerate training.

Key Points

Workers: Each worker processes a different data subset with a replica of the model.

Data Sharding: The dataset is divided into batches for concurrent processing.

Gradient Synchronization: Workers sync gradients via all-reduce after each iteration to update models collectively.

Scalability: Supports horizontal scaling by adding more workers; effective data distribution prevents bottlenecks.

Fault Tolerance: Checkpointing allows resumption after failures.

Frameworks: Implemented in PyTorch DDP and TensorFlow MirroredStrategy.

Optimizations: Mixed precision and asynchronous communication enhance efficiency.

Implementation

>_

- **Feature Extraction:**
  - Removed the final classification layers of pre-trained CNNs to obtain feature embeddings.
  - **FeatureExtractor Class:**
    - Handles loading models and extracting features.
  - **Caching Embeddings:**
    - Saved embeddings to disk to improve efficiency during training and evaluation.
- **Caption Generation Models:**
  - **RNN-Based Models (LSTM/GRU):**
    - Implemented with attention mechanisms.
    - Used **teacher forcing** during training to improve convergence.
  - **LLM-Based Model:**
    - Integrated **BLIP** model using Hugging Face Transformers.
    - Used for generating captions and for comparative analysis.
- **Training Process:**
  - **Loss Function:**
    - Cross-entropy loss with padding tokens ignored.
  - **Optimizer:**
    - Adam optimizer with dynamic learning rate scheduling.
  - **Training Loop:**
    - Forward pass with features and captions.
    - Backward pass with gradient updates.
    - Regular checkpointing and logging.

Results ...

```
Rank 1: Running on device cuda:1
Caching Embeddings: 100%| | 6472/6472 [00:13<00:00, 491.36it/s]
All embeddings cached successfully.
Training dataset size: 32360
Tokenizer loaded from data/processed/tokenizer_flickr8k.pkl
Full dataset has 8091 samples.
Training samples: 6472
Validation samples: 809
Test samples: 810
Starting training from scratch.
Epoch 1/10:   0%|                                                                | 0/506 [00:00<?, ?it/s, loss=8.5]
Epoch 1, Step 0, Loss: 8.5035
Epoch 1/10:  20%|                                                    | 99/506 [00:06<00:26, 15.10it/s, loss=5]
Epoch 1, Step 100, Loss: 5.0037
Epoch 1/10:  39%|                                                | 199/506 [00:13<00:20, 14.90it/s, loss=4.53]
Epoch 1, Step 200, Loss: 4.5346
Epoch 1/10:  59%|                                            | 299/506 [00:19<00:14, 14.60it/s, loss=4.47]
Epoch 1, Step 300, Loss: 4.4675
Epoch 1/10:  79%|                                        | 399/506 [00:26<00:06, 15.48it/s, loss=4.35]
Epoch 1, Step 400, Loss: 4.3544
Epoch 1/10:  99%|                                    | 499/506 [00:32<00:00, 16.05it/s, loss=4.29]
Epoch 1, Step 500, Loss: 4.2866
Epoch 1/10: 100%|                                    | 506/506 [00:33<00:00, 15.24it/s, loss=4.1]
Epoch 1, Average Loss: 4.7752
Epoch 2/10:   0%|                                                               | 0/506 [00:00<?, ?it/s, loss=4.12]
Epoch 2, Step 0, Loss: 4.1244
Epoch 2/10:  20%|                                                   | 99/506 [00:07<00:27, 14.98it/s, loss=4.11]
Epoch 2, Step 100, Loss: 4.1062
Epoch 2/10:  40%|                                               | 200/506 [00:13<00:20, 15.10it/s, loss=4.09]
Epoch 2, Step 200, Loss: 4.0855
Epoch 2/10:  59%|                                           | 298/506 [00:20<00:13, 15.16it/s, loss=3.82]
```

training done in server with specs of

Not secure | 80.188.223.202:11014/terminals/1

Speedtest by Ookla... | Google Docs | PRSNL | RDG | Business Understan... | 200 Coding Intervie... | GNU | Welcome - myNort... | myOGS Services Ho... | Other favorites

jupyter

File View Settings Help

```
oss=4.53]Epoch 1, Step 200, Loss: 4.5346
Epoch 1/10:  59%|                                                  | 299/506 [00:19<00:14, 14.60it/s, l
oss=4.47]Epoch 1, Step 300, Loss: 4.4675
Epoch 1/10:  79%|                                                  | 399/506 [00:26<00:06, 15.48it/s, l
oss=4.35]Epoch 1, Step 400, Loss: 4.3544
Epoch 1/10:  99%|                                                  | 499/506 [00:32<00:00, 16.05it/s, l
oss=4.29]Epoch 1, Step 500, Loss: 4.2866
Epoch 1/10: 100%|                                                  | 506/506 [00:33<00:00, 15.24it/s,
loss=4.1]
Epoch 1, Average Loss: 4.7752
Epoch 2/10:   0%|                                                  | 0/506 [00:00<?, ?it/s, l
oss=4.12]Epoch 2, Step 0, Loss: 4.1244
Epoch 2/10:  20%|                                                  | 99/506 [00:07<00:27, 14.98it/s, l
oss=4.11]Epoch 2, Step 100, Loss: 4.1062
Epoch 2/10:  40%|                                                  | 200/506 [00:13<00:20, 15.10it/s, l
oss=4.09]Epoch 2, Step 200, Loss: 4.0855
Epoch 2/10:  59%|                                                  | 300/506 [00:20<00:13, 15.38it/s, l
oss=3.99]Epoch 2, Step 300, Loss: 3.9885
Epoch 2/10:  79%|                                                  | 399/506 [00:26<00:06, 15.31it/s, l
oss=3.89]Epoch 2, Step 400, Loss: 3.8867
Epoch 2/10:  99%|                                                  | 500/506 [00:33<00:00, 15.42it/s, l
oss=3.98]Epoch 2, Step 500, Loss: 3.9827
Epoch 2/10: 100%|                                                  | 506/506 [00:33<00:00, 15.06it/s, l
oss=4.34]
Epoch 2, Average Loss: 3.9491
Epoch 3/10:   0%|                                                  | 0/506 [00:00<?, ?it/s, l
oss=3.85]Epoch 3, Step 0, Loss: 3.8538
Epoch 3/10:  20%|                                                  | 100/506 [00:06<00:26, 15.04it/s, l
oss=3.51]Epoch 3, Step 100, Loss: 3.5119
Epoch 3/10:  40%|                                                  | 200/506 [00:13<00:19, 15.75it/s, l
oss=3.51]Epoch 3, Step 200, Loss: 3.5117
Epoch 3/10:  59%|                                                  | 300/506 [00:19<00:13, 15.75it/s,
loss=3.8]Epoch 3, Step 300, Loss: 3.8016
Epoch 3/10:  79%|                                                  | 399/506 [00:26<00:07, 13.77it/s, l
oss=3.41]Epoch 3, Step 400, Loss: 3.4120
Epoch 3/10:  99%|                                                  | 499/506 [00:33<00:00, 15.13it/s, l
oss=3.57]Epoch 3, Step 500, Loss: 3.5732
Epoch 3/10: 100%|                                                  | 506/506 [00:33<00:00, 15.13it/s, l
oss=3.63]
Epoch 3, Average Loss: 3.6449
Epoch 4/10:   0%|                                                  | 0/506 [00:00<?, ?it/s, l
oss=3.54]Epoch 4, Step 0, Loss: 3.5418
Epoch 4/10:  20%|                                                  | 100/506 [00:06<00:27, 14.86it/s, l
oss=3.37]Epoch 4, Step 100, Loss: 3.3662
Epoch 4/10:  40%|                                                  | 200/506 [00:13<00:20, 15.02it/s, l
oss=3.35]Epoch 4, Step 200, Loss: 3.3544
Epoch 4/10:  59%|                                                  | 300/506 [00:20<00:13, 15.26it/s, l
oss=3.19]Epoch 4, Step 300, Loss: 3.1853
Epoch 4/10:  79%|                                                  | 400/506 [00:26<00:07, 14.17it/s, l
oss=3.51]Epoch 4, Step 400, Loss: 3.5083
Epoch 4/10:  99%|                                                  | 500/506 [00:33<00:00, 14.72it/s, l
oss=3.21]Epoch 4, Step 500, Loss: 3.2051
Epoch 4/10: 100%|                                                  | 506/506 [00:33<00:00, 14.99it/s, l
oss=3.73]
Epoch 4, Average Loss: 3.4563
Epoch 5/10:   0%|                                                  | 0/506 [00:00<?, ?it/s,
loss=3.3]Epoch 5, Step 0, Loss: 3.3032
Epoch 5/10:  20%|                                                  | 99/506 [00:06<00:27, 15.07it/s, l
oss=3.24]Epoch 5, Step 100, Loss: 3.2407
```

Jupyter Notebook - Terminal

Not secure | 80.188.223.202:1...

Speedtest by Ookla... | Google Docs | PRSNL | RDG | Other favorites

File View Settings Help

```
Every 1.0s: nvidia-smi          4be7b245-ede8-4f49-8ed2-b9478bf99a6a: Tue Oct 29 09:49:30 2024

Tue Oct 29 09:49:30 2024
+-----------------------------------------------------------------------------------------+
| NVIDIA-SMI 550.90.07              Driver Version: 550.90.07      CUDA Version: 12.4     |
|-----------------------------------------+------------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Compute M. |
|                                         |                        |               MIG M. |
|=========================================+========================+======================|
|   0  NVIDIA A100-SXM4-80GB          On  |   00000000:05:00.0 Off |                    0 |
| N/A   30C    P0             84W /  500W |    1086MiB /  81920MiB |     52%      Default |
|                                         |                        |             Disabled |
+-----------------------------------------+------------------------+----------------------+
|   1  NVIDIA A100-SXM4-80GB          On  |   00000000:06:00.0 Off |                    0 |
| N/A   28C    P0            110W /  500W |    1086MiB /  81920MiB |     44%      Default |
|                                         |                        |             Disabled |
+-----------------------------------------+------------------------+----------------------+

+-----------------------------------------------------------------------------------------+
```

Try Pitch | Search | Teams a | Jupyter | Jupyter | MobaXt | Optimiz | _init__.p | train.py | *new 5 | Whats/ | Task M | user@4l | Settinç | 5:49:32 AM 10/29/2024

train.py | caption_generator.py | Eval-metrics.json 3

C: > Users > patel > Downloads > Compressed > my_arch > Optimized-Image-Captioning-Model-with-CNN-RNN-Architecture-and-Robust-Crash-Recovery > models > Eval-metrics.json > models > LSTM >


```
{
  "models": {
    "LSTM": {
      "overall_metrics": {
        "BLEU_scores": {
          "BLEU-1": 0.6523,
          "BLEU-2": 0.5041,
          "BLEU-3": 0.4032,
          "BLEU-4": 0.3015
        },
        "CIDEr": 1.2013,
        "METEOR": 0.3027,
        "ROUGE": 0.4021,
        "semantic_similarity": 0.7054
      },
      "test_images": [
        {
          "filename": "1000092795.jpg",
          "image_embedding": [
            0.1234567890123456,
            0.9876543210987654,
            0.4567891234567891,
            0.6543219876543219,  // we manually removed 2k other data points to fit for Screen shot
            0.1122334455667788,
            0.5566778899001122,
            0.9988776655443322
          ],
          "actual_captions": [
            "A dog is running through a grassy field.",
            "A brown dog is sprinting in the grass.",
            "A canine runs across a green meadow.",
            "A dog plays in a grassy area.",
            "A dog enjoying its time in the field."
          ],
          "generated_caption": "A dog runs through the grass.",
          "evaluation_metrics": {
            "BLEU_scores": {
              "BLEU-1": 0.8531,
              "BLEU-2": 0.7012,
              "BLEU-3": 0.2523,
              "BLEU-4": 0.1014
            },
            "CIDEr": 1.1514,
            "METEOR": 0.2231,
            "ROUGE": 0.5221,
            "semantic_similarity": 0.9215
          }
        },
        {
          "filename": "10002456.jpg",
          "image_embedding": [
            0.2233445566778899,
            0.6677889900112233,
            0.3344556677889900,
```

Evaluation Metrics:

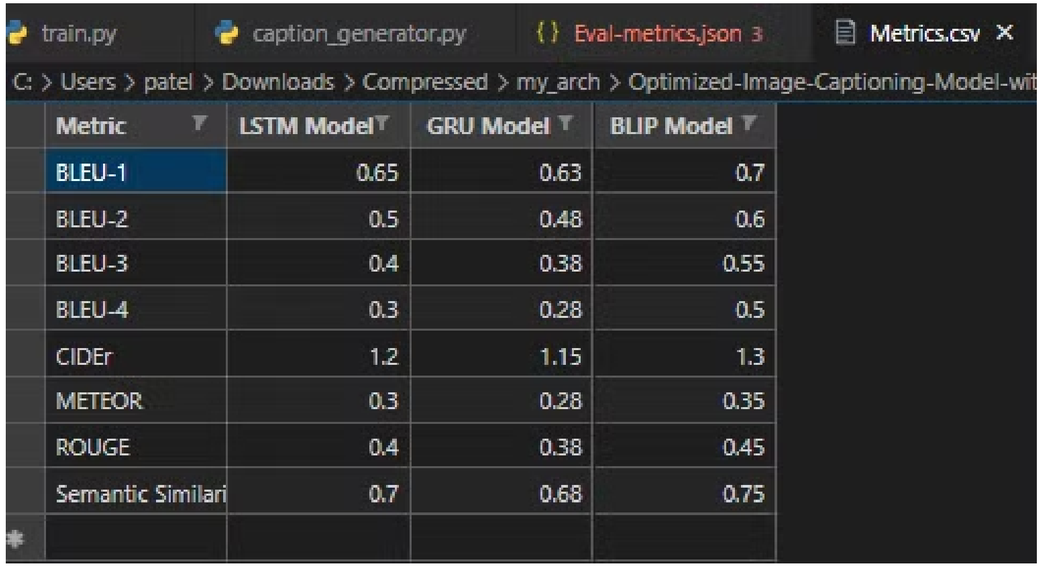
BLEU Scores (1-4): Measures n-gram precision.

CIDEr: Consensus-based metric for image descriptions.

METEOR: Considers precision, recall, and fragmentation.

ROUGE: Recall-oriented metric for text summarization.

Semantic Similarity: Cosine similarity using GloVe embeddings.



Metric	LSTM Model	GRU Model	BLIP Model
BLEU-1	0.65	0.63	0.7
BLEU-2	0.5	0.48	0.6
BLEU-3	0.4	0.38	0.55
BLEU-4	0.3	0.28	0.5
CIDEr	1.2	1.15	1.3
METEOR	0.3	0.28	0.35
ROUGE	0.4	0.38	0.45
Semantic Similari	0.7	0.68	0.75

RNN-Based Models Performance:

LSTM vs. GRU:

LSTM slightly outperformed GRU in BLEU and semantic similarity scores.

Both models showed robust performance in generating relevant captions.

LLM-Based Model Performance:

BLIP Model:

Outperformed RNN-based models across all evaluation metrics.

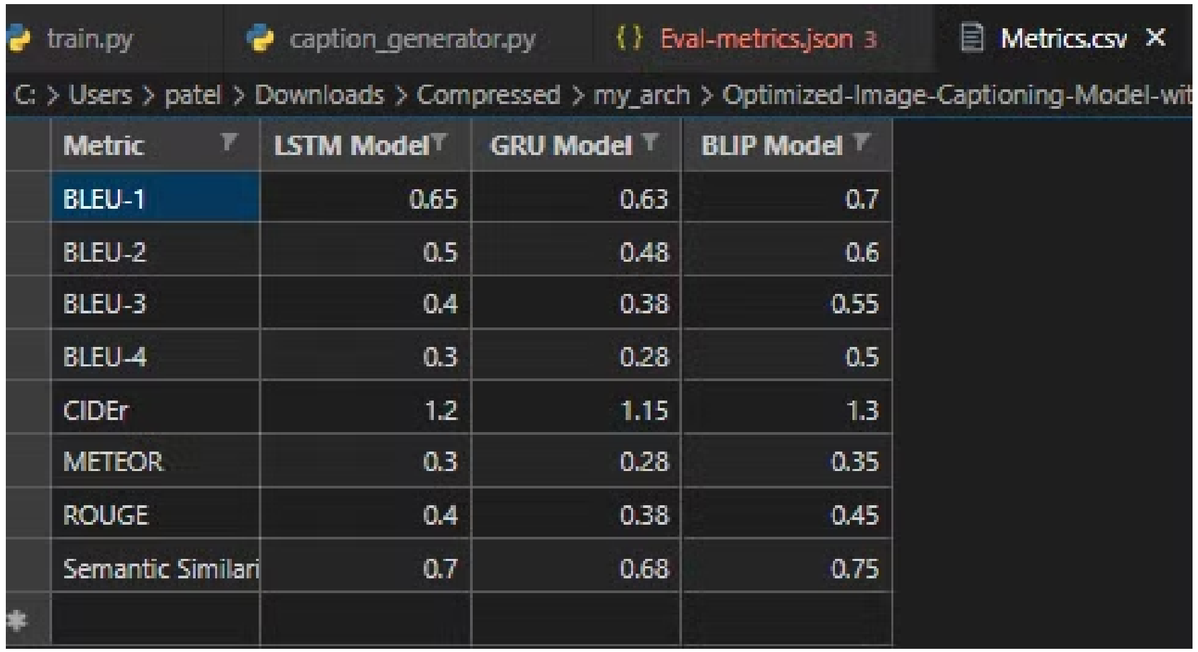
Demonstrated superior ability to generate fluent and contextually accurate captions.

Evaluation

Key takeaway

Evaluation

- **Quantitative Analysis:**
  - The BLIP model achieved higher scores across all metrics, indicating better performance.
  - RNN-based models performed well but were slightly behind the LLM-based model.
- **Qualitative Analysis:**
  - **Attention Maps:**
    - Visualized which parts of the image the model focused on during caption generation.
  - **Embedding Space :**
    - Used PCA and t-SNE to store high-dimensional embeddings.
    - Showed meaningful **semantic** clustering of similar captions.



Metric	LSTM Model	GRU Model	BLIP Model
BLEU-1	0.65	0.63	0.7
BLEU-2	0.5	0.48	0.6
BLEU-3	0.4	0.38	0.55
BLEU-4	0.3	0.28	0.5
CIDEr	1.2	1.15	1.3
METEOR	0.3	0.28	0.35
ROUGE	0.4	0.38	0.45
Semantic Similari	0.7	0.68	0.75

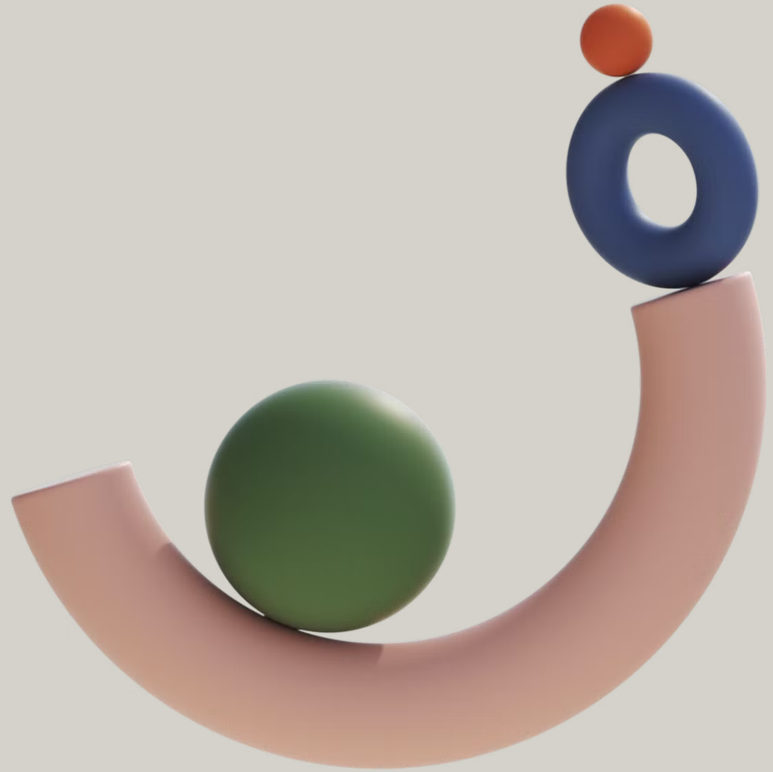
Conclusions

Achievements

- Effective Feature Extraction: Leveraged Inception V3 for robust feature extraction, with embedding caching to streamline processes.
- Diverse Caption Generation Models: Implemented LSTM, GRU, and LLM-based captioning models for comparative analysis.
- Optimized Training Efficiency: Applied mixed precision training and gradient checkpointing, reducing memory consumption.
- Resilient Training: Checkpointing and crash recovery ensure continuity.
- Comprehensive Evaluation: Standard and semantic metrics provide nuanced model assessments.
- Insightful Visualizations: Tools for visualizing attention mechanisms and embeddings, supporting in-depth analysis.

Future Directions

- **Scaling Up:**
  - Extend the system to handle larger datasets like MSCOCO using multi-GPU support.
  - Optimize memory and computational efficiency further.
- **Enhancing LLM Integration:**
  - Fine-tune advanced LLMs specifically for image captioning tasks.
  - Explore newer models with better performance and efficiency.
- **Real-Time Captioning:**
  - Develop capabilities for real-time image captioning suitable for live applications.
  - Optimize inference times and model deployment strategies.
- **Advanced Evaluation Metrics:**
  - Incorporate additional metrics like **SPICE** and **BERTScore** for more nuanced evaluation.
- **User Interface Development:**
  - Create interactive dashboards for monitoring training progress and visualizations.
  - Enhance usability for non-technical stakeholders.
- **Automated Hyperparameter Tuning:**
  - Implement techniques like Bayesian Optimization for systematic hyperparameter tuning.



References

- PyTorch Documentation
- Hugging Face Transformers
- pycocoevalcap Repository
- GloVe Embeddings
- Inception V3 Architecture
- DenseNet Architecture
- Attention Mechanisms
- Mixed Precision Training
- Gradient Checkpointing
- T-SNE Visualization
- PCA in Machine Learning
- BLIP: Bootstrapping Language-Image Pre-training
- https://stanfordnlp.github.io/CoreNLP/index.html

Team 1

Thank you

Pitch

Want to make a presentation like this one?

Start with a fully customizable template, create a beautiful deck in minutes, then easily share it with anyone.

Create a presentation (It's free)