

Yavar Internship Selection Assessment – May 2025

Assessment Problem Statement-2

Image Captioning from Contextual Metadata Using Vision-Language Models (VLMs)

Approach Overview

Developed an end-to-end image captioning system using a **fine-tuned Vision-Language Model (VLM)** — specifically, the **BLIP model (Salesforce/blip-image-captioning-base)**. The goal was to generate both a **concise** and a **detailed** caption for each image, grounded in both its **visual content** and **surrounding textual metadata**.

Each image is paired with a structured metadata file that contains fields such as `section_header`, `above_text`, `caption`, `below_text`, and `footnote`. Our system combines these textual fields into a meaningful context, which is used alongside the image during both training and inference.

The pipeline includes preprocessing, model fine-tuning, caption generation, confidence scoring, inconsistency detection, and annotated output generation.

Preprocessing

Image Preprocessing:

- All images are resized to **384×384 pixels** to match the BLIP model's expected input size.
- Images are normalized using ImageNet mean and standard deviation.
- Images are converted to RGB and passed as tensors to the model.

Metadata Preprocessing:

- Each image has a corresponding `.txt` file with fields:
 - `section_header`

- `above_text`
 - `caption`
 - `below_text`
 - `footnote`
 - `concise_caption` (ground truth)
 - `detailed_caption` (ground truth)
- Missing values (marked as `null`) are replaced with empty strings.
 - These fields are combined into a single context string used during training and inference.
-

◆ Model Architecture

- Used the **BLIP (Bootstrapped Language-Image Pretraining)** model from Hugging Face.
 - The model accepts both an image and text input to generate captions.
 - During training, the model learns to map image + metadata context → target caption.
-

◆ Training Details

- **Model:** `Salesforce/blip-image-captioning-base`
- **Training Data:** Images wnot given for training dataith paired metadata [ground-truth `concise_caption` and `detailed_caption` not given for training data]
- **Loss Function:** Cross-entropy loss on generated token sequence
- **Batch Size:** 4
- **Epochs:** 5 (can be increased for better results)

- **Learning Rate:** 5e-5
 - **Fine-Tuning Modes:**
 - Trained separately on concise and detailed captions
 - Controlled by a config flag (**TRAINING_CAPTION_TYPE**)
 - **Optimiser:**
AdamW
-

◆ Inference & Caption Generation

- For each image:
 - Metadata fields are combined into a rich context.
 - The model generates:
 - A **concise caption** (short summary, 30 tokens max)
 - A **detailed caption** (longer explanation, up to 100 tokens)
 - Captions are saved in **captions.json** and overlaid on the image using OpenCV.
-

◆ Evaluation & Verifiability

- Each caption is scored using :
 - **ROUGE-L**
- A **confidence score (0–1)** is assigned to each caption.

Since the model not generated a meaningful caption. Metadata is reflected in the output.
- Captions are checked for consistency against metadata:
 - If the section header has no overlap with the caption → marked as inconsistent

- **Low-confidence or inconsistent captions** are:
 - Logged in a timestamped log file
 - Underlined and highlighted in yellow in the annotated output

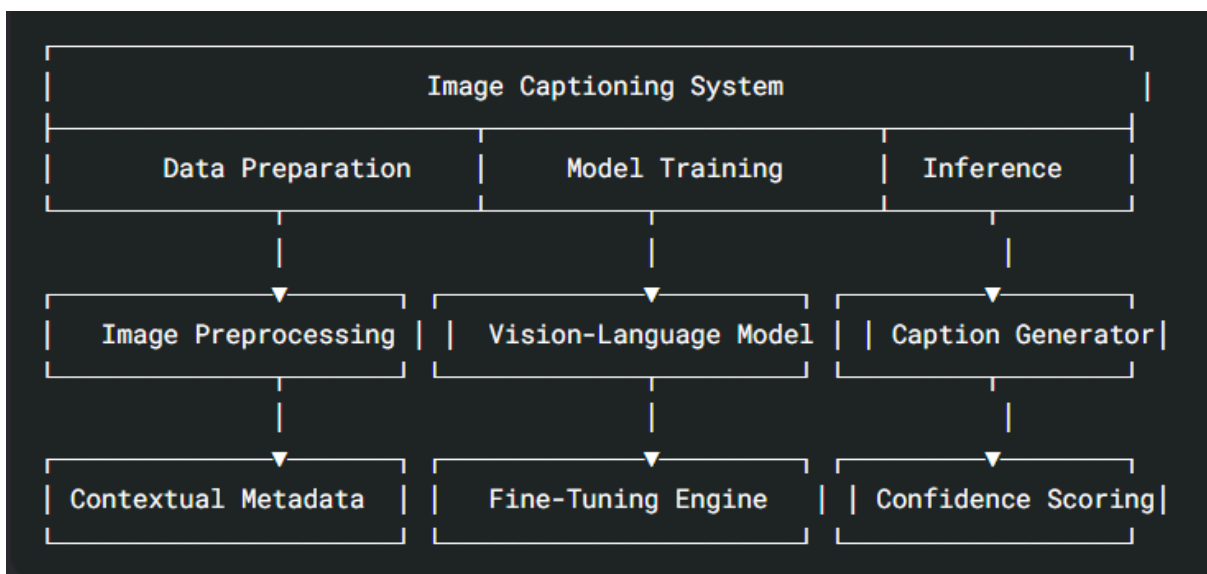
♦ Output Structure

Outputs are saved in `output_folder/`:

- 📷 Annotated images with:
 - Blue text = Concise caption
 - underline = Low confidence or inconsistency
- 📄 `captions.json` file with:
 - Concise & detailed captions
 - Confidence scores
 - Context used
 - Inconsistency reports

Related Images

1.Architecture overview



2. Component Breakdown

A. Data Preparation Layer

| Data Preparation | | |
|---|--|--|
| Image Preprocessor | Metadata Extractor | Dataset Validator |
| <div><div>- Resize images</div><div>- Normalize</div><div>- Augment</div></div> | <div><div>- Parse text files</div><div>- Extract fields</div><div>- Clean text</div></div> | <div><div>- Verify image-metadata correspondence</div><div>- Check caption quality</div></div> |

B. Model Training Layer

| Model Training Stack | | |
|--|---|---|
| BLIP Model | Training Controller | Evaluation Module |
| <div><div>- Base VLM</div><div>- Custom head</div><div>- Multi-task loss</div></div> | <div><div>- LR scheduling</div><div>- Batch management</div><div>- Early stopping</div></div> | <div><div>- ROUGE/BLEU metrics</div><div>- Visual grounding check</div><div>- Overfitting detection</div></div> |

C. Inference Layer

| Inference Pipeline | | |
|--|--|---|
| Caption Generator | Context Integrator | Output Formatter |
| <div><div>- Beam search</div><div>- Length control</div><div>- Repetition pen.</div></div> | <div><div>- Metadata fusion</div><div>- Attention masking</div><div>- Confidence calc.</div></div> | <div><div>- JSON output</div><div>- Image annotation</div><div>- Highlight low confidence</div></div> |

3. Data Flow

```

1. Input Phase:
  [Image] → Preprocessing → [Normalized Image Tensor]
  [Metadata] → Parsing → [Structured Context Embedding]

2. Training Phase:
  [Image Tensor] ———→
                        |→ [BLIP Encoder] → [Multimodal Fusion] → [Decoder] → [Loss Calc]
  [Context Embedding]↑

3. Inference Phase:
  [Processed Image] → [Caption Generation] → [Context Refinement] →
  [Confidence Scoring] → [Formatted Output]

```

Features

BLIP-based image captioning (Salesforce/blip-image-captioning-base)

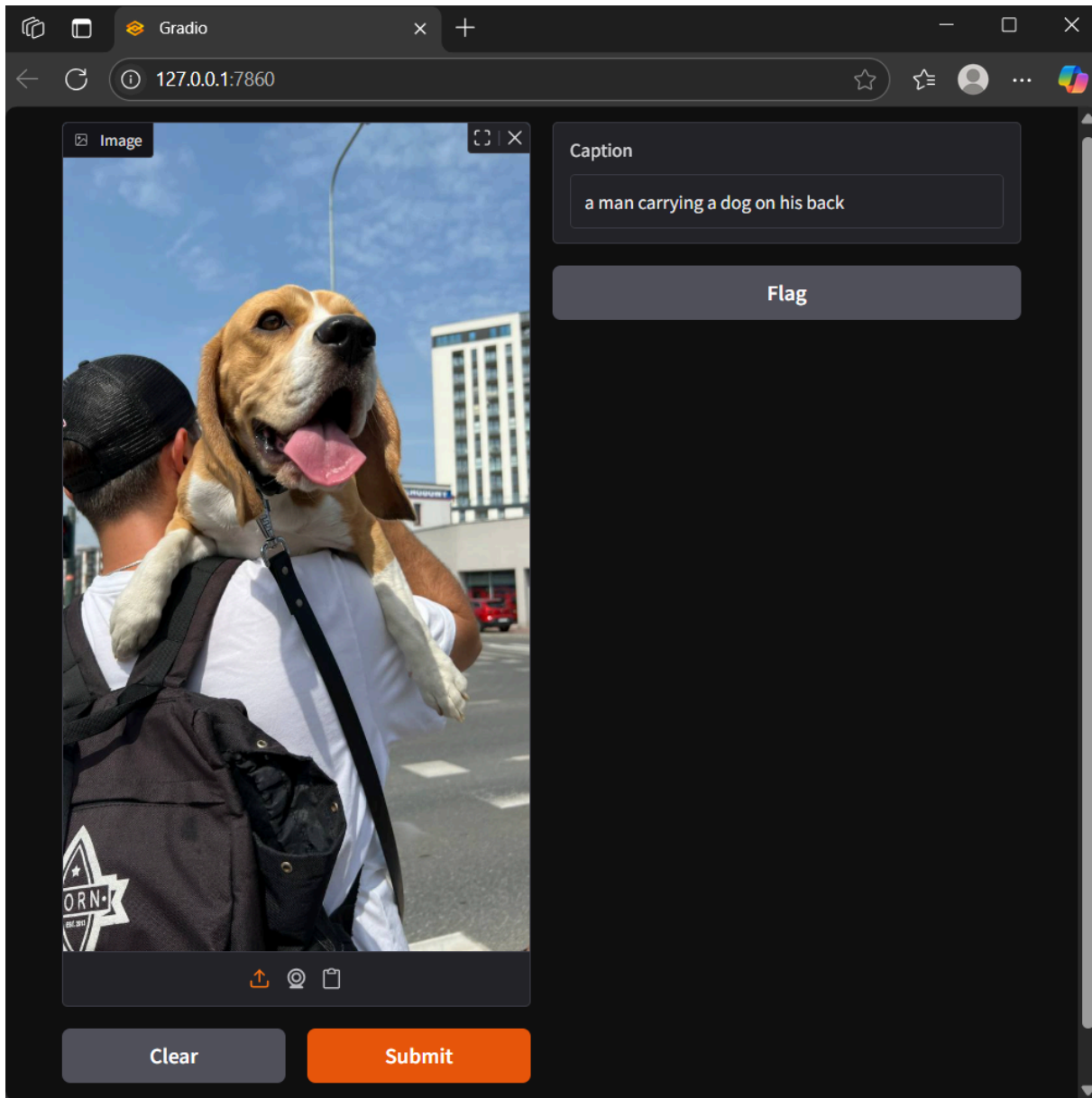
Custom dataset support with metadata (section headers, captions, etc.)

Training and inference scripts

Caption verifiability checks (semantic similarity, consistency)

Annotated output images with generated captions and confidence scores

Salesforce/blip-image-captioning-base model without fine-tuning



Folder Structure

```
.
├── img_folder/           # Input images
├── metadata_folder/      # Metadata files (one per image, .txt format)
├── output_folder/        # Output images and results
├── config.py             # Configuration file
├── data_loader.py        # Custom dataset loader
├── train.py              # Training script
├── inference.py          # Inference and evaluation script
├── utils.py              # Utility functions
└── requirements.txt      # Python dependencies
```

Images (circuits, graphs, logos, charts)



Metadata File Format

```

section_header: "Section Title"
above_text: "Text above the image"
caption: "Ground-truth caption"
picture_id: "image_file_name"
footnote: "Footnote text"
below_text: "Text below the image"
concise: "Short summary"
detailed: "Detailed description"

```

Training

```

File Edit Selection View Go Run Terminal Help train.py - ImageCaption - Cursor
Problems Output Debug Console Terminal Ports Postman Console
> _pycache_
> blip_finetuning
> img_folder
> metadata_folder
> output_folder
> annotated_images
1) captions.json
> venv
> Include
> Lib
> Scripts
> share
> pyvenv.cfg
config.py
data_loader.py
inference.py
main.py
temp_input.jpg
train.py
utils.py

Epoch 1/5: 100% | 5/5 [01:11<00:00, 14.28s/it]
Evaluating: 0%
decoder-only architecture is being used, but right-padding was detected! For correct generation results, please set `padding_side='left'` when initializing the tokenizer.
Evaluating: 100% | 2/2 [03:20<00:00, 100.15s/it]

Epoch 1 Results:
- Avg Loss: 10.3834
- Eval ROUGE-L: 0.5306
New best model saved with ROUGE-L: 0.5306

Epoch 2/5: 100% | 5/5 [01:06<00:00, 13.33s/it]
Evaluating: 0%
decoder-only architecture is being used, but right-padding was detected! For correct generation results, please set `padding_side='left'` when initializing the tokenizer.
Evaluating: 100% | 2/2 [03:17<00:00, 98.66s/it]

Epoch 2 Results:
- Avg Loss: 8.3758
- Eval ROUGE-L: 0.6528
New best model saved with ROUGE-L: 0.6528

Epoch 3/5: 100% | 5/5 [01:06<00:00, 13.32s/it]
Evaluating: 0%
decoder-only architecture is being used, but right-padding was detected! For correct generation results, please set `padding_side='left'` when initializing the tokenizer.
Evaluating: 100% | 2/2 [00:17<00:00, 8.57s/it]

Epoch 3 Results:
- Avg Loss: 7.5338
- Eval ROUGE-L: 0.9949
New best model saved with ROUGE-L: 0.9949

Epoch 4/5: 100% | 5/5 [01:06<00:00, 13.30s/it]
Evaluating: 0%
decoder-only architecture is being used, but right-padding was detected! For correct generation results, please set `padding_side='left'` when initializing the tokenizer.
Evaluating: 100% | 2/2 [00:17<00:00, 8.54s/it]

Epoch 4 Results:
- Avg Loss: 6.8695
- Eval ROUGE-L: 0.9949

Epoch 5/5: 100% | 5/5 [01:06<00:00, 13.33s/it]
Evaluating: 0%
decoder-only architecture is being used, but right-padding was detected! For correct generation results, please set `padding_side='left'` when initializing the tokenizer.
Evaluating: 100% | 2/2 [00:17<00:00, 8.53s/it]

Epoch 5 Results:

```


output



output_folder > annotated_images > table1_annotated.jpg

Concise: section _ header : "student records" above _ text : "stu

| ID | NAME | CLASS | MARK | GENDER |
|----|-------------|-------|------|--------|
| 1 | John Deo | Four | 75 | female |
| 2 | Max Ruin | Three | 85 | male |
| 3 | Arnold | Three | 55 | male |
| 4 | Krish Star | Four | 60 | female |
| 5 | John Mike | Four | 60 | female |
| 6 | Alex John | Four | 55 | male |
| 7 | My John Bob | Five | 78 | male |
| 8 | Arnold | Five | 85 | male |
| 9 | Tes Qry | Six | 78 | male |
| 10 | Big John | Four | 55 | female |

Detailed: section _ header : "student records" above _ text : "student performance data including id, name, class, mark, and gender." caption : "table 1 : student academic records" footnote : "gender data includes some inconsistencies (e. g., ' john deo ' marked as female)." below _ text : " marks range from 55 to 85 with male students predominating in higher scores. " (Confidence: 0.00)

Setup

1.Clone the repository

git clone https://github.com/DharaniSowmiyan/YaavarHackathon-22PD11

cd your-repo

2.Install dependencies:

pip install -r requirements.txt

3.Prepare your data:

Place images in img_folder/

Place corresponding metadata .txt files in metadata_folder/

Resources:

Creating an image dataset:

https://huggingface.co/docs/datasets/image_dataset

Fine-tune BLIP on an image captioning dataset:

<https://colab.research.google.com/drive/1lbqiSiA0sDF7JDWPeS0tccrM85LloVha?usp=sharing#scrollTo=AFGnjCgDoLIJ>

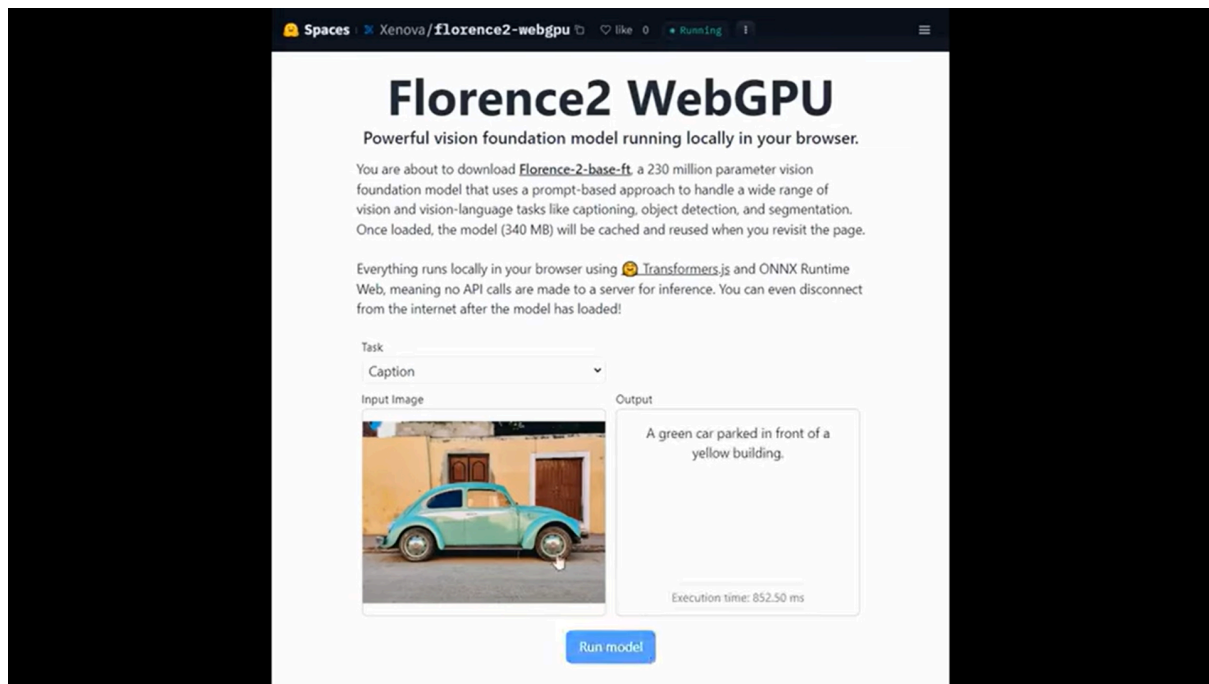
Various other models explored:

1. Florence-2

<https://blog.roboflow.com/fine-tune-florence-2-object-detection/>

| hot computer vision models 🔥 | | | | | |
|------------------------------|---|-------------|-------------|---------------|------------------------------|
| model | task | license | self-deploy | speed (fps)** | parameters |
| Florence-2 | captioning, detection, segmentation, grounding, OCR | MIT | \$ | 4 | 230M, 770M |
| PaliGemma | captioning, VQA detection, segmentation | custom | \$\$ | 1 | 3B |
| LLaVA | captioning, VQA, OCR | Apache-2.0* | \$\$\$ | 0.25 | 13B |
| GPT-4V/4o | captioning, VQA, OCR | ✗ | ✗ | ✗ | ✗ |
| YOLO-World | zero-shot detection | GPL-3.0 | \$ | 52 - 74 | 13M - 48M (77M - 110M)*** |
| GroundingDINO | zero-shot detection | Apache-2.0 | \$\$ | 1.5 | 172M |
| YOLOv8 | detection | AGPL-3.0 | \$ | 78 - 565 | 3.2M - 68.2M |

* - code is under Apache-2.0, but model was trained on ChatGPT results and can't be used in commercial setting
** - benchmarked on NVIDIA V100
*** - parameters count before re-parametrization



Why it did not work:

Florence-2 requires **flash-attn**, which needs:

- GPUs like **A100**, **L4**, or **H100**.
- **CUDA 11.8+** and **PyTorch 2.1+**.

2.Salesforce/blip2-opt-2.7b

<https://huggingface.co/Salesforce/blip2-opt-2.7b>

3.Salesforce/blip2-flan-t5-xl

<https://huggingface.co/Salesforce/blip2-flan-t5-x>

(2,3)Why it did not work: while running the model-out of memory since my gpu VRAM is 4GB

4.huggingface.co/yifeihu/TF-ID-base

<https://huggingface.co/yifeihu/TF-ID-base>

Why it did not work: it detects tables/figures image captioning not supported.

Future work:

Making them(image+context and caption) better Relevance: How well the caption aligns with both the image and the provided context.