PROJECT REPORT

by - Shobhit Kumar

Under the guidance of Ramya Lakshmi, Research Associate
Supervised by Prof. Anurag S. Rathore
Department of Chemical Engineering, IITD

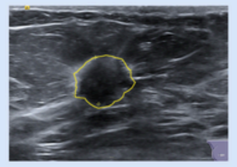
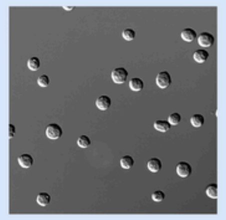
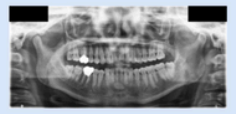
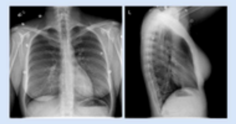
A FINE-TUNING STUDY OF PALI-GEMMA AND MEDGEMMA ON MULTIMODAL MEDICAL IMAGING TASKS

Outline

Project Overview

‘Unified Multimodal Medical Imaging Challenge’

This challenge aims to develop a single model capable of performing four key tasks—classification, detection, counting and regression—across nine medical imaging modalities

Tasks	Prompt		Answer	Metric
Disease Diagnosis (classification)		Question: What is the diagnostic interpretation of the contour-bordered area in the ultrasound image? Options: A) Benign lesion present B) Malignant lesion present C) No lesion, only normal tissue	B	Balanced Accuracy
Multi-label classification		As a dental imaging specialist, what abnormalities would you report on tooth 22? Select all that apply from the following, separate multiple selections with a semicolon: Dental Caries, Periradicular Lesions, Pulp Diseases, Chronic Injury of Tooth Hard Tissues, Disturbances of Eruption of Teeth, Periodontal Diseases, Endodontic Treatment, Restorative Treatment, and Complications	Dental Caries; Periradicular Lesions	F1 Score
Detection		Identify and locate all chromosomes in this image. Return a JSON mapping chromosome identifiers to their bounding box coordinates in [x, y, width, height] format	{"1": [[916, 361, 80, 35], [844, 640, 48, 74]], "2": [[966, 402, 52, 70], [663, 360, 73, 53]], ...}	F1 Score matching via IoU > 0.5
Cell Counting		What is the cell count for this particular microscope image?	24	Mean Absolute Error
Regression		Evaluate the ABR% from the bone levels shown in this orthopantomogram.	21.67%	Mean Absolute Error
Report Generation		Describe heart size, shape, and any cardiac abnormalities seen on the chest X-ray.	Cardiac contours are within normal limits.	GREEN Score

Icons by Icons8

Dataset Overview

Dataset Composition

- Total Datasets: 19
- Medical Modalities: 8
- Task Types: 7
- Images: 50,996
- QA Pairs: 58,112
- Sources: 9 (Public + Private)
- Storage: ~50 GB

Tasks

- Classification
- Multi-label Classification
- Detection
- Counting
- Regression
- Report Generation
- Instance Detection

Modalities

- Dermatology
- Clinical
- Microscopy
- Retinography
- X-ray
- Endoscopy
- Ultrasound
- Mammography

Sample Entry

```
{
 "TaskType": "Classification",
 "Modality": "X-ray",
 "ImageName":
"imagesTr/image001.png",
 "Question": "What abnormality is
visible?",
 "Answer": "Fracture",
 "Split": "train"
}
```

Dataset Structure

```
training/
 ├── modality/
 │   └── dataset_name/
 │   ├── imagesTr/
 │   └── dataset_questions_train.json
```

Paligemma Model

Paligemma

Overview

Model Type: Vision-Language Foundation Model Architecture: Pretrained multimodal transformer (image + text)

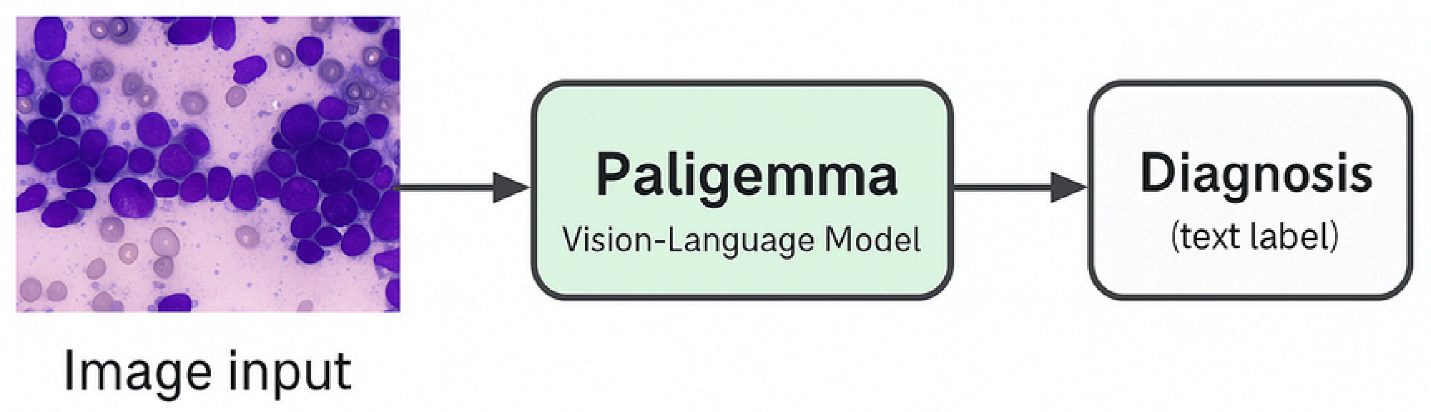
Why Paligemma?

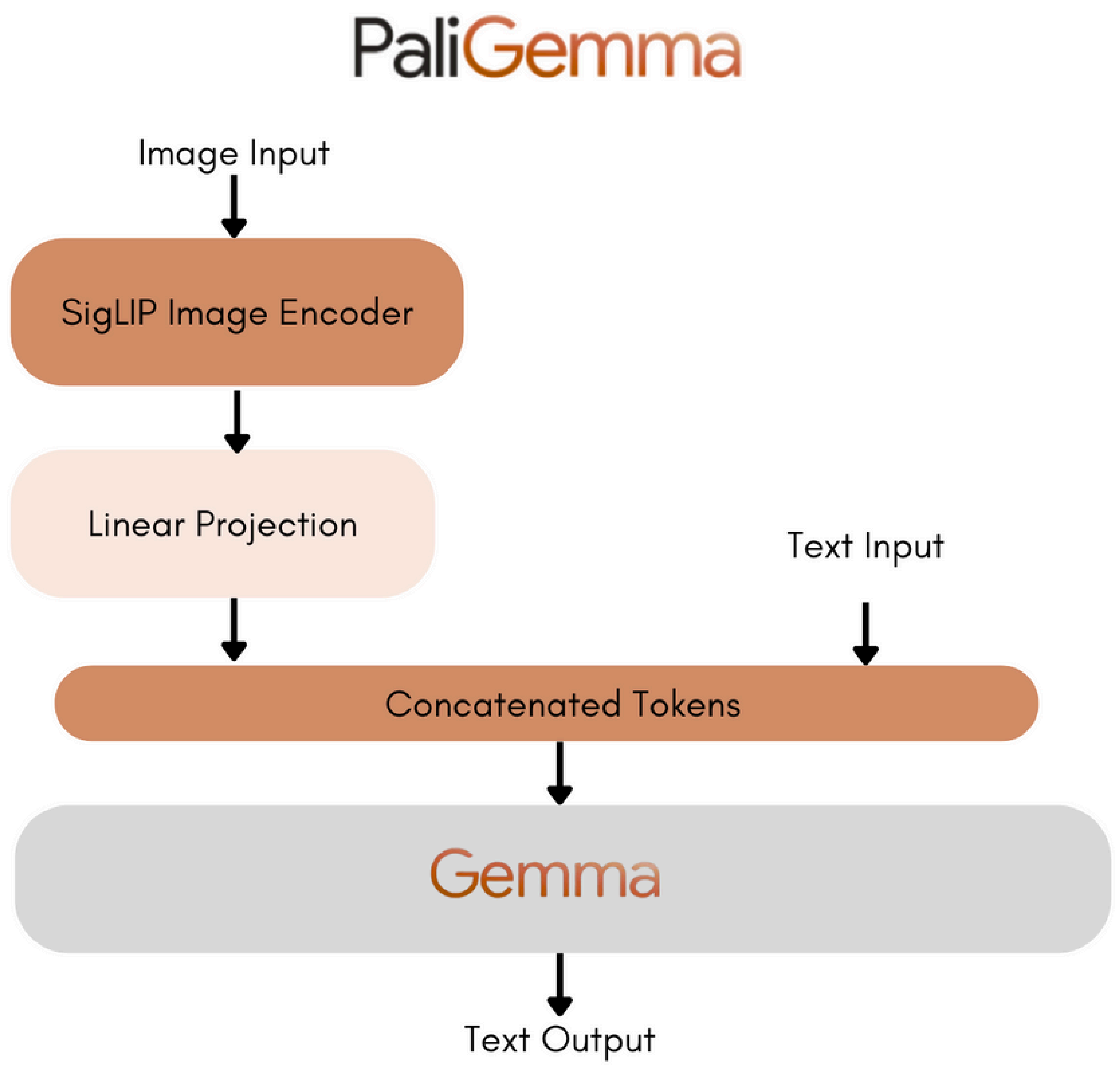
Multimodal Input : Processes both medical images and diagnostic text prompts
Multi-task Capability : Adaptable to classification, detection, regression, etc.
Scalable : Pretrained and fine-tunable for diverse medical imaging datasets
Efficient : Optimized inference for clinical deployment

Model Setup

Dataset: Full Dataset except X-ray
Task: Classification, Regression, Counting, Detection,
Fine-tuning:
- Loss Function: Custom masked token-level cross-entropy (JAX)
- Metric: Loss

Result: Achieved baseline classification performance on each modality





Methodology

- Created miniconda environment from .yml file
- Downloaded and imported pretrained PaLI-Gemma-2 checkpoint from Kaggle
- Shifted to HPC for large-scale data training
- Used full dataset of 30,000+ examples across 8 modalities
- Merged individual dataset JSON files into unified JSON
- Removed missing images examples, shuffled examples, split 80/10/10 (train/test/val)
- Images pre-processing
- Tokenized text with BOS/EOS, masks
- Fine-tuned attention sublayers using JAX & Flax with help of official Google's documentation.
- Used cosine LR scheduler with warmup; logged and validated periodically
- Trained the model on 4 different tasks using varying learning rates
- Saved checkpoints based on best validation loss and visualized predictions

Data Preprocessing

Image Preprocessing

All input images were standardized to meet model input requirements:

- Channel Adjustment:
  Grayscale images converted to RGB (3 identical channels)
  Images with alpha (transparency) channels had them removed
- Resizing:
  Scaled to 224×224 pixels using bilinear interpolation with anti-aliasing for high-quality resizing
- Normalization:
  Pixel values scaled from [0, 255] → [-1, 1] to match model expectations

Text Preprocessing

- Tokenizes image-associated prompts for training:
  Prefix: Always tokenized with a BOS (beginning-of-sequence) token
  Suffix: Optionally tokenized with EOS (end-of-sequence) token
- Attention Masks:
  mask_ar: 0 for prefix (full attention), 1 for suffix (causal attention)
  mask_loss: 0 for prefix (excluded from loss), 1 for suffix (included in loss)
- Sequence Handling:
  Pads all tokens/masks to a specified sequence length of 1400 characters using zeros

Training Strategy

Tools and Frameworks

- Used modules from Google's big_vision repository:
  big_vision.models.proj.paligemma for model architecture
  predict_fns for decoding
  big_vision.datasets.jsonl to load data
  big_vision.utils for LR scheduling, reshaping, parameter control
  big_vision.sharding for multi-device distribution
- Framework: JAX + Flax
  jax, jax.numpy: Tensor ops, gradients, JIT compilation
  jax.tree_util, jax.tree: PyTree parameter updates
  jax.sharding: Multi-GPU speedup via parallelism
- sentencepiece
  Tokenization and detokenization of string input.

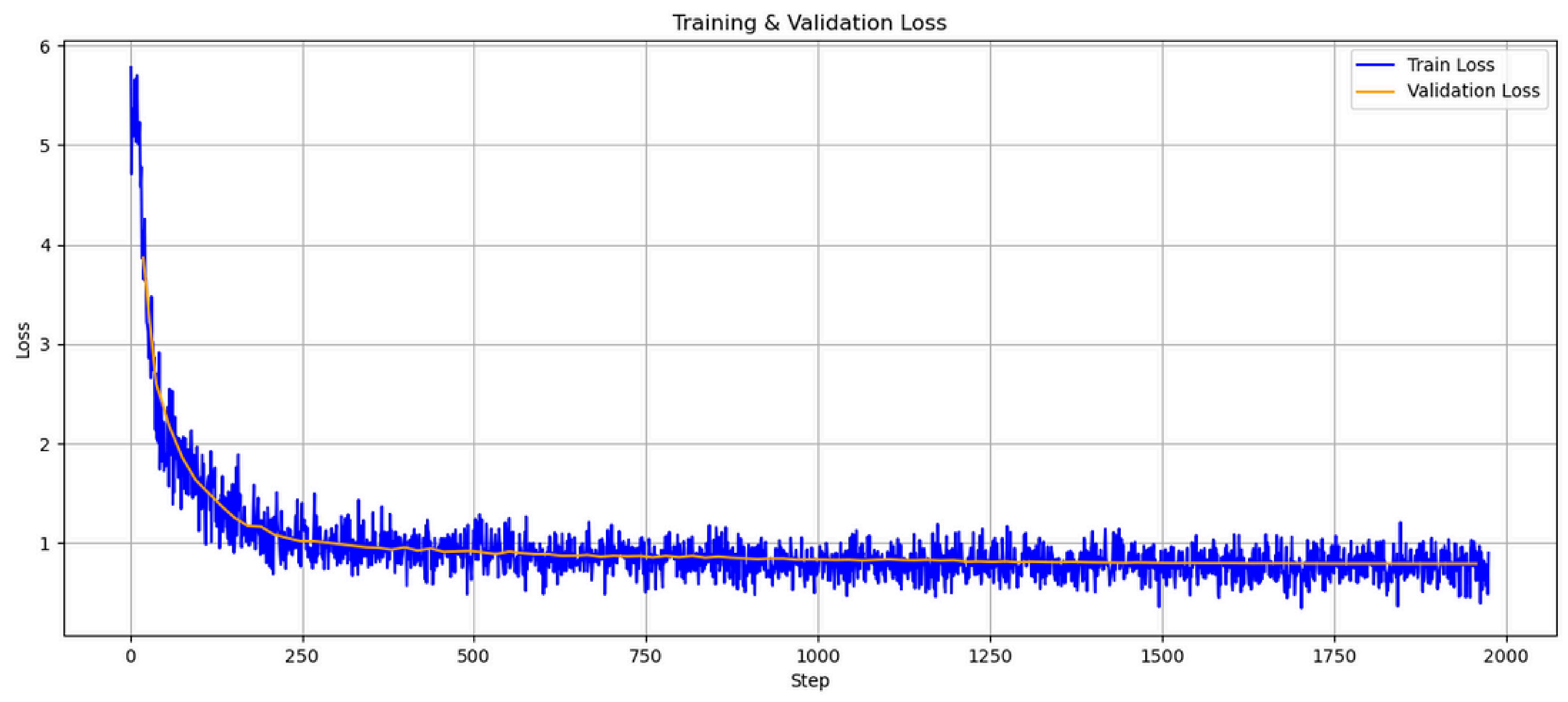
Fine Tuning

- Trainable Params: Only attention sublayers
  - llm/layers/attn/attn_vec_einsum/w
  - llm/layers/attn/kv_einsum/w
  - llm/layers/attn/q_einsum/w
- Loss: Masked cross-entropy
- Batch Size: 8 or 16
- Learning Rates: 0.00005 – 0.0005
- Used cosine learning rate schedule with 10% warmup and SGD optimizer
- Best checkpoint saved via validation loss
- Periodically evaluated validation loss on 1024 validation samples and visualized 4 predictions

Models	Batch Size	Learning Rate	Saved with minimum loss
Model 1	8	0.00005	0.1454
Model 2	8	0.00015	No
Model 3	8	0.0005	0.1373
Model 4	16	0.0001	0.2919
Model 5	16	0.0002	0.2642
Model 6	16	0.00005	0.7901
Model 7	16	0.00015	No

Model Training

- Models 1 to 3 were trained with a batch size of 8 using different learning rates: 0.00005, 0.00015, and 0.0005.
- Models 4 to 7 used a larger batch size of 16, with learning rates ranging from 0.00005 to 0.0002.
- The "Saved with minimum loss" column indicates the minimum validation loss at which the model checkpoint was saved.
- For Models 2 and 7, the model was not saved based on minimum loss — instead, only the final training checkpoint was saved.



Model - 6 (Batch size - 16, learning rate - 0.00005 losss - 0.7901)

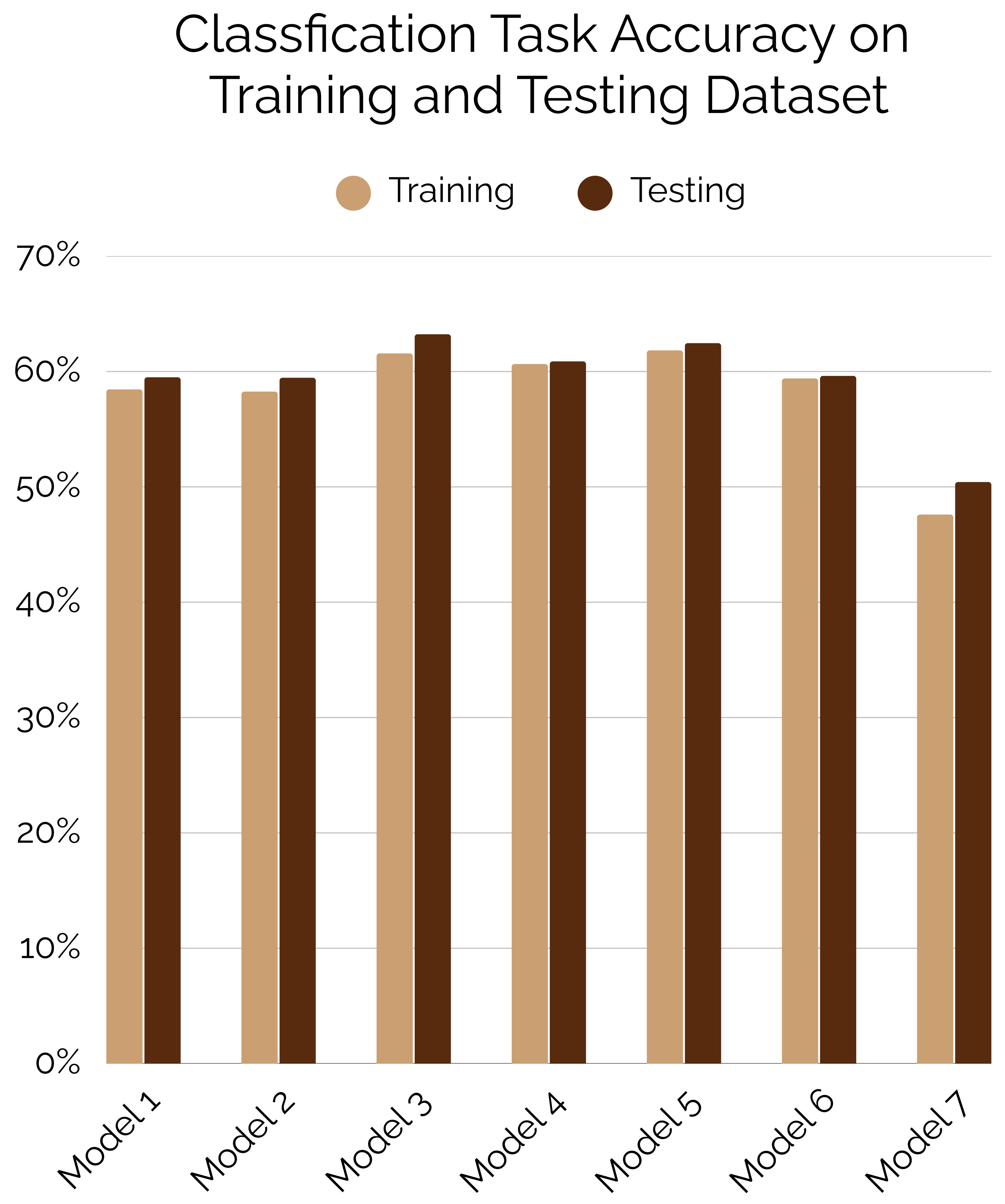
Loss Trends

The graph illustrates training and validation loss trends for the classification task using PaLI-Gemma-2. Training was done with a batch size of 16 and a learning rate of 0.00005, following a cosine learning rate schedule with 10% warmup. Loss dropped rapidly in the initial steps, indicating effective learning, and stabilized after step 500, suggesting convergence. Validation loss closely tracked the training loss, showing low overfitting. The final validation loss was around 0.7901, indicating good generalization. These results reflect a stable and effective training setup using the selected optimizer and fine-tuning configuration.

Paligemma Results

Classification

This chart shows the classification accuracy of seven models on training and testing data. Models 1–6 achieve consistent results (~60–62%), indicating good generalization. Model 7 underperforms (~48–50%), suggesting difficulty in learning or applying patterns. Accuracy was measured using exact string matching, a strict metric requiring perfect output matches.



Model	MAE (Train)	RMSE (Train)	R^2 (Train)	MAE (Test)	RMSE (Test)	R^2 (Test)
1	13.85	20.19	-0.26	14.12	19.52	-0.14
2	12.87	17.96	0.002	13.40	18.26	0.0004
3	13.45	19.70	-0.20	13.60	20.05	-0.20
4	12.66	17.97	0.001	13.44	19.08	-0.09
5	12.52	17.88	0.01	13.18	18.33	-0.01
6	14.19	20.18	-0.20	13.94	19.62	-0.15
7	13.08	18.61	-0.07	14.09	19.36	-0.12

Regression

- Overall performance is weak, as reflected by negative or near-zero R^2 values in most models, meaning the regression models are not fitting the data well.
- Models 2, 4, and 5 had slightly better generalization (R^2 close to 0 on test set), but still no strong predictive capability.
- MAE and RMSE values remain close between training and testing, suggesting no overfitting, but limited learning.
- The highest R^2 on test set is Model 2 with 0.0004, which is negligible—indicating regression formulation might need refinement or data might be imbalanced.

Cell Counting

Model	MAE (Train)	RMSE (Train)	R^2 (Train)	MAE (Test)	RMSE (Test)	R^2 (Test)
1	40.2	59.57	-0.5	63.9	97.76	-0.46
2	61	64.49	-0.081	74.4	86.32	-0.14
3	35.2	56.4	-0.35	53.9	93.22	-0.33
4	41	63.44	-0.7	59.7	93.18	-0.33
5	38	59.51	-0.49	58.5	94.26	0.17
6	42.1	62.56	-0.66	61.8	91.07	-0.47
7	47.9	67.56	-0.94	64.4	98.97	-0.5

- Model 5 performed best with a test R^2 of 0.17, indicating relatively better generalization.
- Most models had negative R^2 scores, meaning predictions were worse than using the mean.
- Model 2 and Model 7 showed signs of overfitting.
- Overall, results suggest the regression setup needs improvement for accurate cell count prediction.

Model	Xmin - MAE	Xmin - R2	Ymin - MAE	Ymin - R2	Xmax - MAE	Xmax - R2	Ymax - MAE	Ymax - R2	Wrong format outputs
1	72.07	-0.76	39.32	-0.24	73.81	-1.6	95.24	-0.03	1
2	45.46	-0.2	19.57	0.43	56.64	-1.02	35.91	0.75	10
3	40.4	0.16	19.87	0.43	44.46	0.07	31.46	0.83	0
4	43.57	0.04	18.95	0.52	70.96	-2.3	44.8	0.46	9
5	47.3	-1.21	21.44	0.35	56.26	-0.63	41.16	0.68	6
6	55.64	-0.52	38.51	-0.27	90.07	-2.88	119.51	-0.58	36
7	49.92	-0.01	31.3	-0.19	60.9	-1.15	61.49	0.42	15

Detection (Train)

The model's lesion detection performance was evaluated using the Mean Absolute Error (MAE) and R^2 (coefficient of determination) across bounding box coordinates (Xmin, Ymin, Xmax, Ymax) on both the training and testing datasets. Additionally, the number of outputs in an incorrect format (non-parsable or missing fields) was recorded.

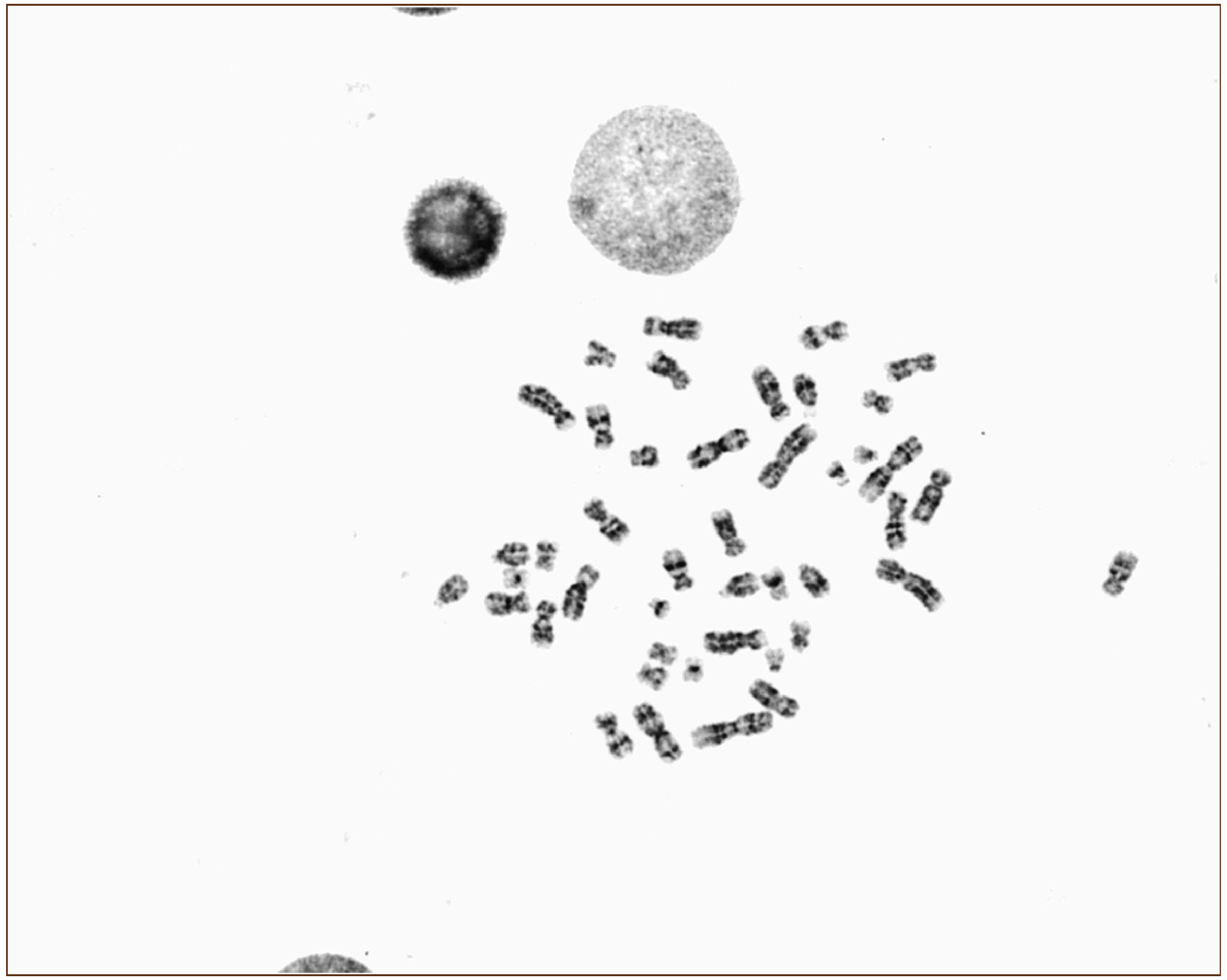
Detection (Test)

Model	Xmin - MAE	Xmin - R2	Ymin - MAE	Ymin - R2	Xmax - MAE	Xmax - R2	Ymax - MAE	Ymax - R2	Wrong format outputs
1	77.08	-0.83	41.15	-0.4	85.84	-1.7	98.98	-0.18	1
2	47.79	-0.07	18.82	0.46	58.72	-0.92	33.54	0.74	13
3	42.95	0.15	19.16	0.45	45.08	0.21	31.44	0.8	0
4	48.29	-0.92	18.23	0.55	74.41	-1.65	44.35	0.44	17
5	44.74	0.09	21.43	0.34	62.19	1.42	44.4	0.17	9
6	61.87	-0.54	41.95	-0.6	97.97	-2.44	124.27	-0.74	49
7	51.61	0.06	28.07	-0.11	65.47	-1	56.49	0.44	19

MAE:
- Xmin/Xmax MAE ranges from 40–98 → varied accuracy in horizontal localization.
- Ymin MAE is lower (18–41) → better vertical detection.
- Ymax MAE is highest (31–124) → likely due to large object height variation.

R^2 Score:
- Many negative R^2 values → poor prediction.
- Few positive R^2 (up to 0.83) → some moderate correlation.

Format Errors:
- Up to 49 wrong outputs in training, 36 in testing → output formatting needs fixing.



Instance Detection

Task - Predict bounding boxes for multiple object instances per class in an image.

Required Format

```
{
  "class_id": [[x, y, width, height], ...],
  "class_id": [[x, y, width, height], ...],
   .....
}
```

Although several models were trained, none consistently produced correctly formatted outputs. Common issues included missing keys, incorrect brackets, and invalid nesting, leading to high counts of "Wrong Format Outputs"

Medgemma Model

Model Medgemma

Overview

Model Type: Multimodal vision–language foundation model for medical image-to-text generation.
Model Architecture: Transformer-based architecture with a vision encoder and a causal language decoder integrated via cross-modal attention.

Why Medgemma?

- MedGemma is pretrained and instruction-tuned on medical images and clinical text.
- Optimized for structured medical report and findings generation.
- Captures fine-grained medical visual features more accurately.
- Lower hallucination risk in clinical descriptions than general VLMs.

Model Setup

Dataset: Full Dataset
Task: Classification, Regression, Counting, Detection,
Fine-tuning:
  Loss Function: standard language-model cross-entropy loss over tokens
  Metric: Loss
Result: Achieved baseline classification performance on each modality

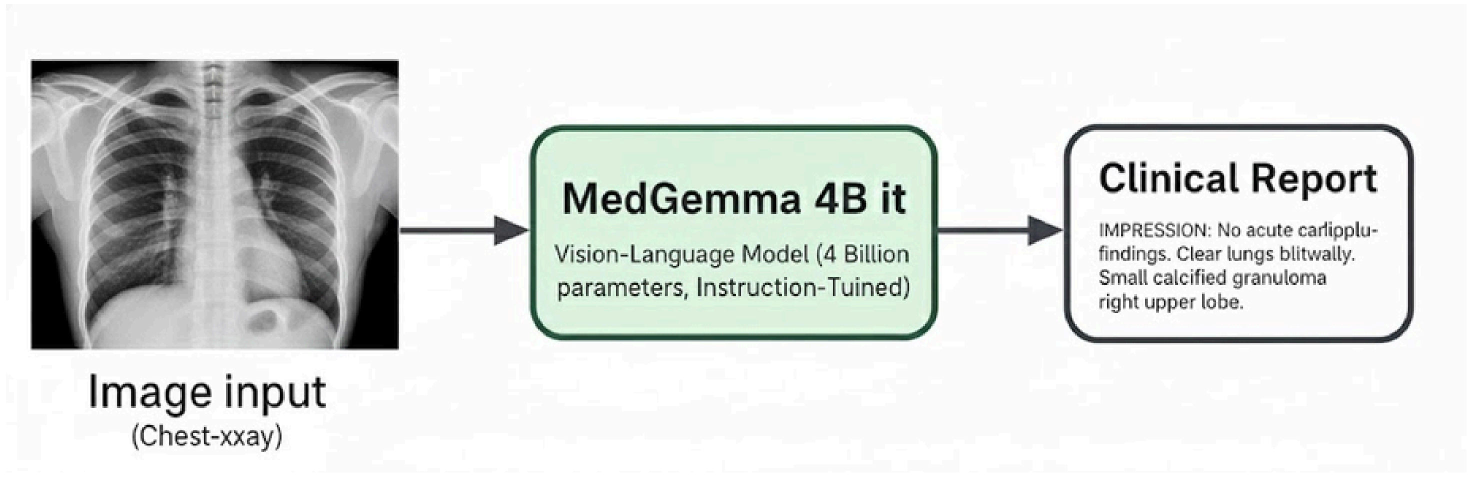
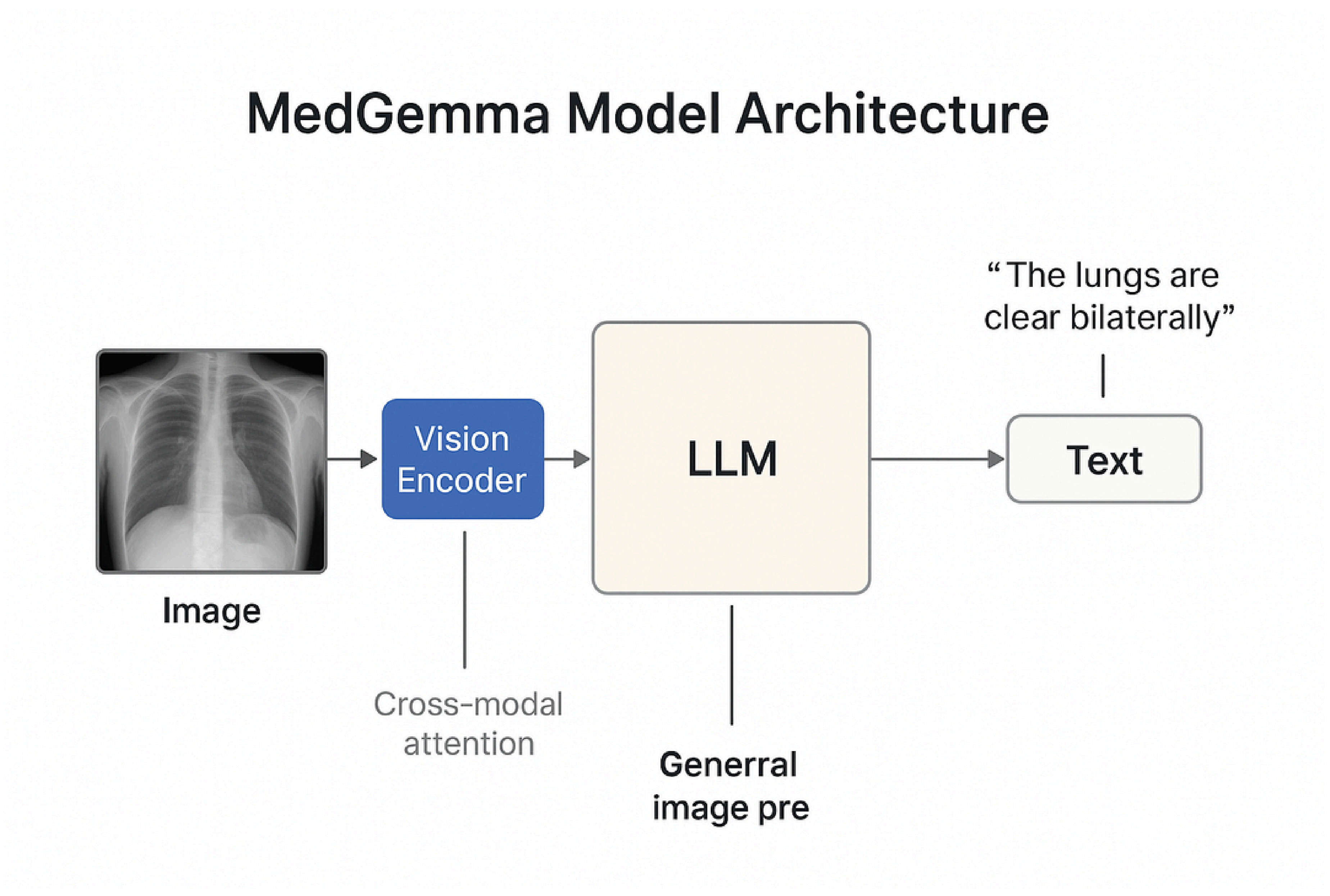


Image input
(Chest-xxay)
MedGemma 4B it
Vision-Language Model (4 Billion parameters, Instruction-Tuined)
Clinical Report
IMPRESSION: No acute carlipplu-findings. Clear lungs blitwally. Small calcified granuloma right upper lobe.



Methodology

- Dataset Prep: Loaded ~51k image–text pairs and split into train/validation with a fixed seed.
- Image Processing: Converted images to RGB, resized to 224×224, and decoded during training.
- Structured annotations into chat-style user–assistant prompts.
- Batched images/text, applied tokenizer templates, and masked padding/image tokens.
- Used Google MedGemma-4B-IT vision–language transformer.
- LoRA Fine-Tuning: Froze base weights and trained low-rank adapters (r=16, α=16).
- Applied SFTTrainer with AdamW, LR=2e-4, warmup, and gradient clipping.
- Used gradient checkpointing, small batches, and bfloat16 precision.
- Tracked loss, token accuracy, and entropy during training.
- Saved adapters and enabled prompt-based medical report generation.

Data Preprocessing

Image Preprocessing

- Each sample's image is converted to RGB mode and optionally resized to 224×224 earlier, then collected into a Python list of PIL images.
- The PIL image list is passed to processor(text=texts, images=images, return_tensors="pt", padding=True).
- Processor resizes images if needed, applies normalization to pixel values, and packs them into a batched tensor.
- Images are placed under the correct key expected by google/medgemma-4b-it model for multimodal processing.

Text Preprocessing

- For each sample, a messages list is built with a user turn (image placeholder + instruction text) and an assistant turn (target report text).
- processor.apply_chat_template(..., tokenize=False) converts each messages list into a single formatted chat-style text string.
- All formatted strings are collected into texts, and images into images.
- In collate_fn, processor(text=texts, images=images, return_tensors="pt", padding=True) tokenizes and pads text and preprocesses images into tensors, then labels are made by cloning input_ids and masking padding/image tokens with −100 so they are ignored in the loss.

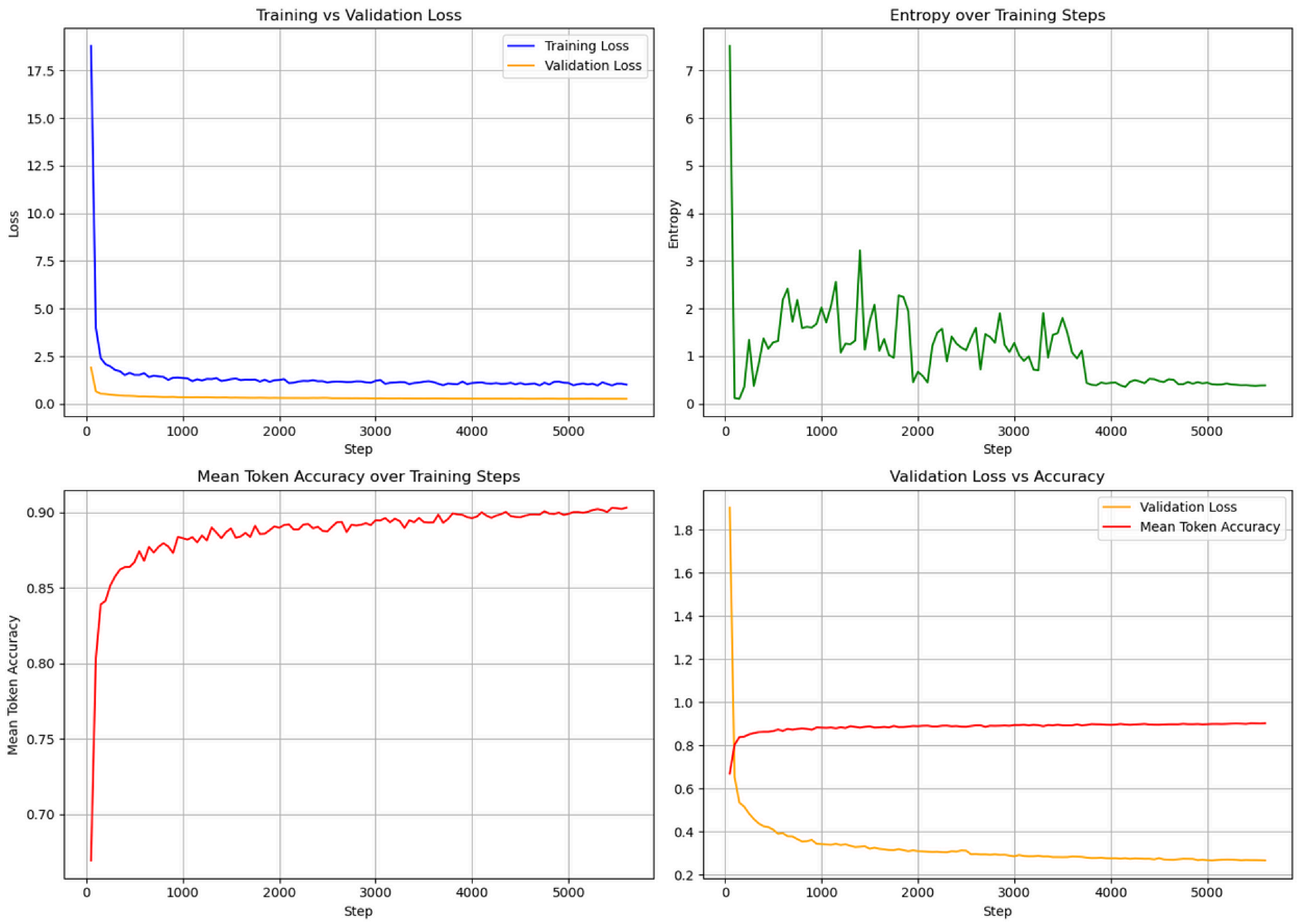
Training Strategy

Tools and Frameworks

- **Hugging Face Transformers:** Uses AutoModelForImageTextToText and AutoProcessor for google/medgemma-4b-it multimodal model loading and preprocessing.
- **TRL SFTTrainer**: Supervised Fine-Tuning trainer handles the full training loop with custom dataset and collator.
- **PEFT LoRAConfig:** Parameter-efficient fine-tuning with rank=16, alpha=16 targeting all linear layers and key modules.
- **Datasets library:** Loads JSONL files (44,966 train/6,425 validation) with custom Features for image+text processing.
- **PyTorch ecosystem:** bfloat16 precision, gradient checkpointing, fused AdamW optimizer, and automatic device mapping.

Fine Tuning

- **Custom collate_fn:** Converts raw examples to chat messages, tokenizes text, preprocesses images, masks padding/image tokens in labels.
- **LoRA adapters only:** Freezes base 4B model, trains low-rank matrices (r=16) on linear layers for memory efficiency.
- **Single epoch training:** 5,621 steps with batch size 2×4 accumulation, learning rate 2e-4, linear scheduler, 3% warmup.
- **Multimodal loss masking**: Cross-entropy ignores padding (-100) and image tokens, trains only on text generation positions.
- **Evaluation metrics:** Tracks validation loss, entropy, token count, and mean token accuracy (~90.3% final) every 50 steps.



Loss Trends

The plots illustrate a successful and stable machine learning training process over approximately 5,500 steps. Both the training and validation losses rapidly decreased and stabilized at low, closely aligned values, indicating effective learning without significant overfitting. Concurrently, the mean token accuracy quickly rose and plateaued at a high value, around 0.90. The entropy also decreased significantly, suggesting the model's predictions became highly confident. Overall, the model converged well, achieving high accuracy with a minimized loss.

Medgemma Results

Regression

Low R^2 indicates limited ability to model continuous numerical relationships despite moderate error values.

- MAE: 9.84
- RMSE: 15.78
- R^2: 0.035
- Accuracy (±5 threshold): 50.39%

Classification

- Accuracy: 65.73%
- Strong performance across multimodal classification tasks
- Benefits from medical-domain pretraining

Counting

Model struggles with precise count prediction; moderate R^2 suggests partial trend learning but poor exact accuracy.

- MAE: 47.6
- RMSE: 78.35
- R^2: 0.365
- Accuracy (±5 threshold): 15%

‘MedGemma performs best on classification, shows partial improvement in regression and counting, but insufficient for reliable use.‘

Detection

- To not only identify an object (e.g., a nodule, a mass) but also to determine its precise location within the medical image.
- The model must translate the visual information into a textual output, typically in a structured format like a JSON string containing the object's label and its bounding box coordinates (e.g., [x_min, y_min,x_max, y_max])
- Output metrics shown below

XMIN:
- MAE: 15.1958
- MSE: 468.1748
- RMSE: 21.6373
- R^2: 0.8615

YMIN:
- MAE: 11.4895
- MSE: 317.7972
- RMSE: 17.8269
- R^2: 0.8042

XMAX:
- MAE: 18.7692
- MSE: 742.6434
- RMSE: 27.2515
- R^2: 0.8858

YMAX:
- MAE: 21.4615
- MSE: 1044.6783
- RMSE: 32.3215
- R^2: 0.9320

Instance Detection

Task - Predict bounding boxes for multiple object instances per class in an image.

Required Format
```
{
  "class_id": [[x, y, width, height], ...],
  "class_id": [[x, y, width, height], ...],
   .....
}
```

R^2 Score: 0.7247 showing 72.47% of coordinate variability of x, y, w and h is explained

Challenges Faced

- The model frequently failed to produce correctly formatted JSON outputs for detection and instance detection tasks, with errors such as missing keys or invalid nesting. This led to 49 malformed outputs in training and 36 in testing in Paligemma model while it improved so much with good R-square in Medgemma.
- Counting and regression tasks showed weak performance, with MAE in an acceptable range but R^2 often negative, indicating poor model generalization in Paligemma, in Medgemma accuracy improved but still R^2 is not in acceptable range.
- Instance detection was unstable for PaLI-Gemma due to invalid multi-object bounding box outputs, making evaluation unreliable, while MedGemma produced more stable and better-formatted predictions.
- Managing large-scale multimodal data caused memory constraints and I/O bottlenecks, especially during image loading and batch preparation.
- Multi-task fine-tuning added complexity, and despite using a cosine learning rate schedule, several models showed unstable loss behavior or failed to converge in Paligemma.
- HPC environment was not compatible with bitsandbytes during Medgemma model training.

Conclusion

- This project evaluated PaLI-Gemma-2 and MedGemma-4B-IT on multimodal medical imaging tasks—classification, regression, counting, detection, and instance detection—using the Unified Multimodal Medical Imaging Challenge dataset.
- Classification was the strongest task for both models. PaLI-Gemma achieved 60–62% accuracy across modalities using exact string matching, showing stable generalization. MedGemma improved performance to 65.73%, benefiting from medical-domain pretraining and instruction tuning.
- Regression and counting remained weak for PaLI-Gemma, with mostly negative or near-zero R^2 despite stable MAE and RMSE. MedGemma reduced error metrics and improved threshold-based counting accuracy, but low R^2 values indicate limited numerical reasoning.
- Detection and instance detection highlighted the largest performance gap. PaLI-Gemma produced inconsistent localization and frequent malformed outputs, making evaluation unreliable. MedGemma showed clear improvement, achieving strong coordinate-level R^2 (up to 0.93) and an instance detection R^2 of 0.7247, reflecting better structured output generation.
- Overall, MedGemma consistently outperformed PaLI-Gemma, especially in detection and structured prediction tasks, while PaLI-Gemma performed best in classification. The results show that multimodal foundation models are effective for medical classification but require improved structured decoding, numerical grounding, and task-specific supervision for reliable detection and regression in clinical settings.

Acknowledgement

I would like to express my sincere gratitude to Prof. Anurag Singh Rathore, Department of Chemical Engineering, IIT Delhi, for providing me with the opportunity to work on this challenging and insightful internship project.
I am also deeply thankful to Ms. Ramya Lakshmi, Research Associate, for her continuous support, technical guidance, and mentorship throughout the internship. Her detailed feedback and encouragement played a key role in shaping the progress of this work.
I would also like to acknowledge the MICCAI FLARE 2025 Task 5 Organizers for curating a rich multimodal medical imaging dataset that enabled real-world experimentation with cutting-edge models.
Finally, I am grateful to the IIT Delhi HPC Facility for the computational resources that made large-scale model training possible.