AMAZON ML CHALLENGE 2024 | ATTACK ON PYTHON | IIT KHARAGPUR

Problem Statement

The goal is to create a machine learning model that extracts entity values from images. We have been given a training dataset consisting of 2,63,859 (image_url, group_id, entity_name, entity_value) triplet. The final model is evaluated on a test dataset consisting of 1,31,187 (image url, group_id, entity_name) pairs. The table below describes the attributes of the dataset with example:

Attribute	Description	Example
image_url	A link to the image from which entity value needs to be extracted	https://m.media-amazon.com/images/I/61wj40BBexL.jpg
group_id	Denotes a specific characteristic of the image	459516 has similar products
entity_name	The name of the entity which needs to be extracted	Item_weight, depth, width, height, voltage, wattage etc
entity_value	The value of the extracted entity (only present in the training dataset)	Item_weight = 7.0 gram for the figure above

Approach

Our main pipeline came from the observation that if a model is able to focus only on the text on the image which is related to the entity-name then it dramatically increases its chances of extracting the correct data from the image, which means we need Attention! We experimented with a range of light-weight multi-modality models which can fit on a **16GB T4-GPU**. We finally decided to **fine-tune** paligemma-3b-ft-docvqa-448 which is an open-source model fine-tuned on the DocVQA which is a **document visual question-answering benchmark**. We fine-tuned this model by creating a balanced sample of the training dataset. The **test-set was then split into 4-parts** and we performed **parallel inference** to create the predictions. These predictions were fed into the **cleaning-script** which made sure that the units are correct in terms of spellings and abbreviations. This output was fed into the **sanity-check script** and finally the predictions were uploaded to the platform.

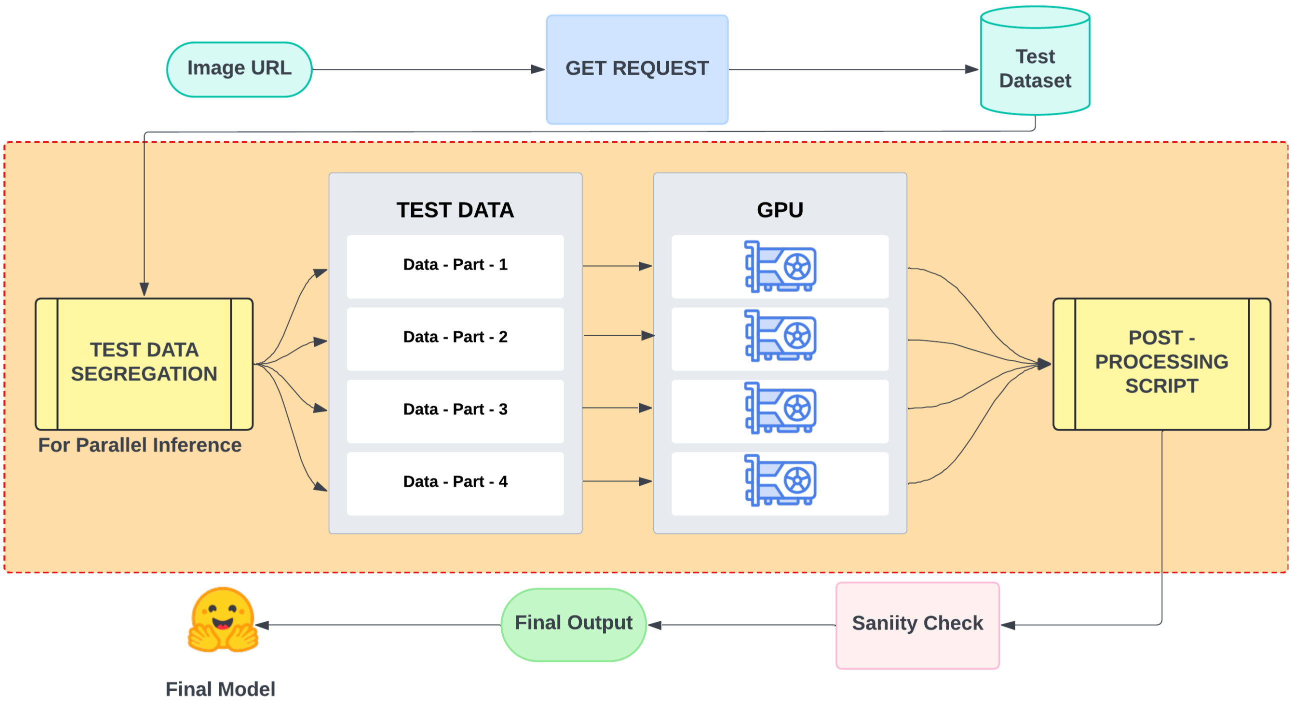

FIG 1: Inference pipeline used for generating the test predictions. The images are scrapped using the given URLs, the test dataset images are split into 4 parts and each of them are put for parallel processing

Model Architecture, Quantization, Training, and Inference Techniques

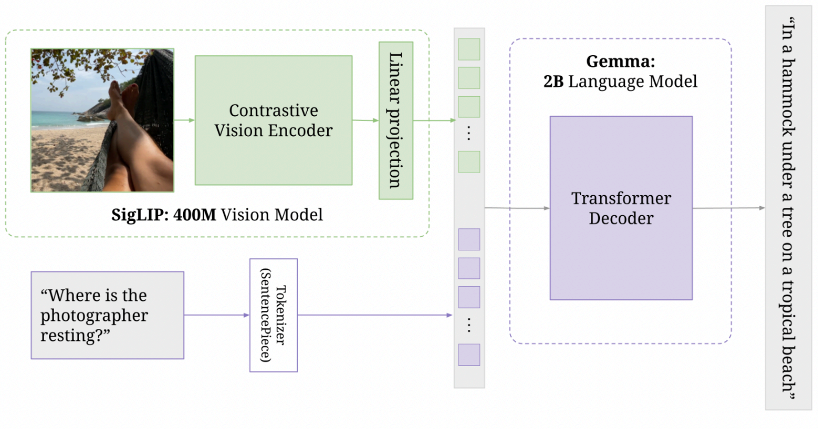
Contrastive Vision Encoder
Linear projection
SigLIP: 400M Vision Model
Gemma: 2B Language Model
Transformer Decoder
"In a hammock under a tree on a tropical beach"
"Where is the photographer resting?"
Tokenizer (SentencePiece)

FIG 2: The block diagram shows the architecture of the Pali-gemma model (ref).The model is made up of a vision component SigLip vision model and the Gemma language model. This model was fine-tuned on a sample of the training dataset.

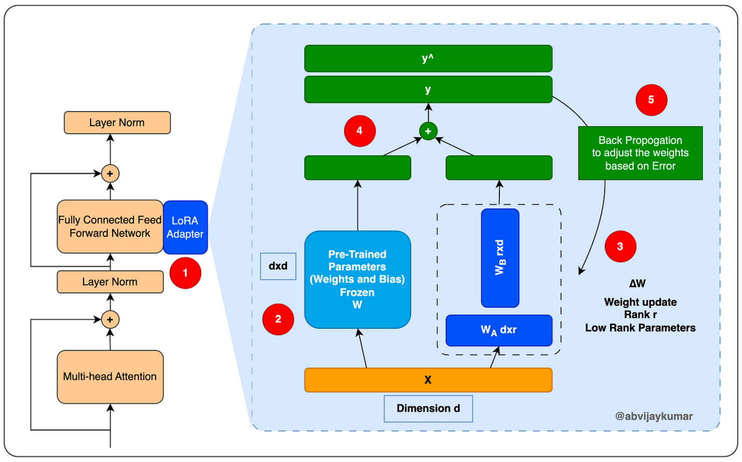
Layer Norm
Fully Connected Feed Forward Network
LoRA Adapter
Layer Norm
Multi-head Attention
y^
y
Back Propogation to adjust the weights based on Error
dxd
Pre-Trained Parameters (Weights and Bias) Frozen W
W_B rxd
W_A dxr
ΔW
Weight update Rank r Low Rank Parameters
x
Dimension d
@abvijaykumar

FIG 3: PEFT-LoRA fine-tuning for efficient training

Quantisation and Peft-LoRa Fine-tuning: Quantisation helps in reducing the size of models to host them in a resource constrained environment. Generally deep learning models use 32-bit floating-point numbers. Quantisation can reduce this to 16-bit, 8-bit, or even lower precision. PEFT stands for Parameter-Efficient Fine-Tuning, and LoRA stands for Low-Rank Adaptation. This is a technique used to fine-tune large language models more efficiently. Traditional fine-tuning methods update all non-frozen parameters of the model which significantly reduces the number of trainable parameters, lower memory requirements, computational costs and helps in faster training.

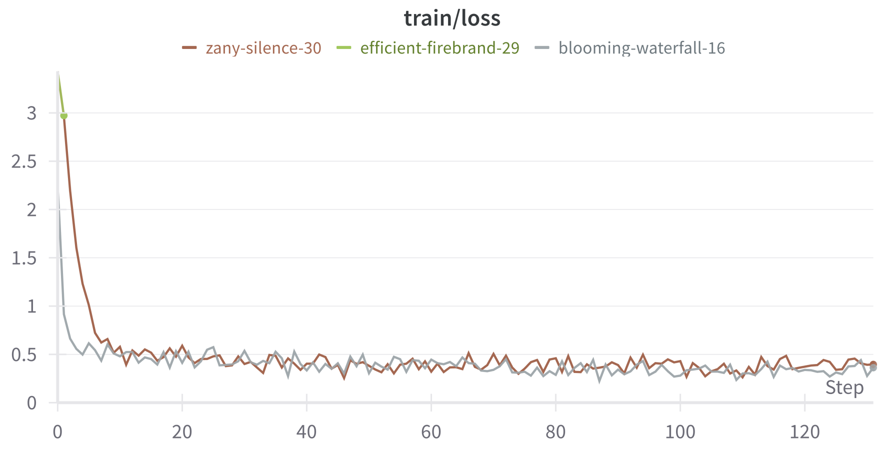

FIG 4: The loss function of our model when fine-tuned with rank-8 LoRa adapter on a balanced sample of the dataset blooming-waterfall-16 in on 6500 points, and zany-silence-30 is on 13,500 data points

RESULTS: blooming-waterfall-16 -> F1-Score: 0.621 | zany-silence-30 -> F1-Score: 0.666