Study Oriented Project (CS F266)

Akshay Shukla (2022A7PS0087P)

Project title: Multilingual Text Visual Question Answering System
Professor-In-Charge: Prof. Yashvardhan Sharma
PhD Mentor: Ms Vijay Kumari

Introduction

Text Visual Question Answering (Text - VQA) models aim to answer questions based on an image given as input by extracting information from the visual elements as well as the embedded text in the images. They try to effectively use the individual and multimodal attention mechanism for joining all the modalities in a space to provide the exact answer. There are several models available which deal with VQA. These include PaLI, TAP, M4C and SSBaseline to name a few. Direct applications of such models include helping the visually impaired by answering questions on images in daily life as well as helping them in navigation from one location to another.

Making neural networks larger has proven beneficial for tasks involving language and vision modeling. Well-known models like mT5, GPT-3, and ViT have demonstrated clear advantages from scaling, resulting in better performance in tasks such as image captioning and visual question-answering. One of the most recently proposed models is the PaLI model introduced by Google Research in 2022. PaLI was introduced as a model that handles image-only, language-only, and combined tasks across multiple languages through a unified interface. It achieves a balanced parameter distribution between language and vision components, leveraging existing large models like mT5-XXL and ViT-e for efficient training and transfer learning. PaLI-17B, trained on the WebLI dataset with billions of image-text pairs in over 100 languages, achieves state-of-the-art results on benchmarks like COCO Captioning and VQAv2. PaLI has a test time accuracy of around 73% on most popular datasets.

The objective of this project is to try to improve on the capabilities of the PaLI model while specifically focussing on Indian languages. This implies that we either diversify the WebLI dataset (10 billion images from 109 languages) to include more Indian languages or we fine-tune the existing model to be compatible with Indian languages. The multilingual functionalities should be included in a way that it supplements the OCR, NLP and decoder functions. Further we also attempt to improve the accuracy of the PaLI model while working on this expanded dataset. This can be done by implementation of the 2D-Attention and variational encoder for the already proposed Text-VQA model.

Related Work

The existing baseline models for Text Visual Question Answering are LoRRA, M4C and SSBaseline.

- LoRRA (27.63% accuracy) - A basic model using self and contextual attention over visual and textual features along with a module to copy OCR tokens directly to the answers.
- M4C (40.46% accuracy) - Different modalities taken through transformer embeddings and then a joint embedding is created and sent to a single-multimodal transformer.
- SSBaseline (45.66% accuracy) - OCR and Visual features are handled separately by vanilla attention blocks and then served to a transformer.

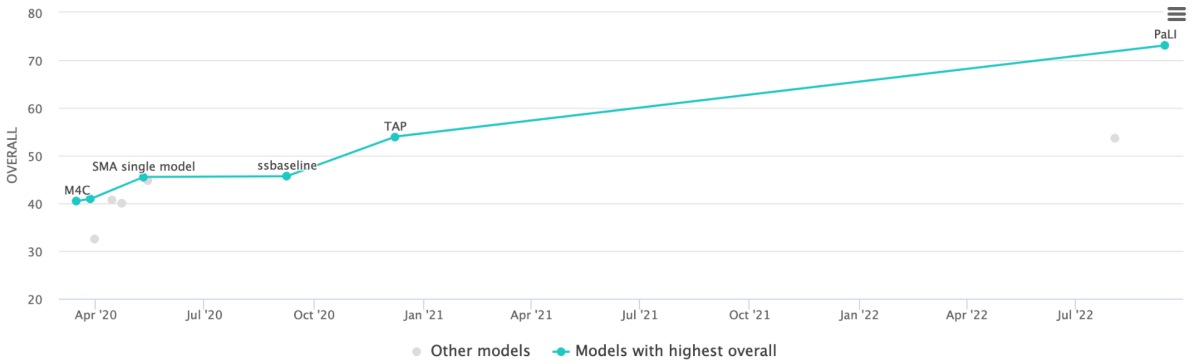
Of these, the model which was heavily focused on in the past is SSBaseline. This model has several features included within it. It uses the BERT model (designed by Google Research) for the Natural Language Processing (NLP) task. This provides us with an entry point to pose a question to the model. The ResNet-152 model is used for reading the image provided to the model. All questions asked to the model will be answered with the help of this image. Faster R-CNN is used for Optical Character Recognition (OCR) from the visual images given to the model. This model is trained over the VQA 2.0 (https://visualqa.org/download.html) and TextVQA (https://textvqa.org/dataset/) datasets. The TextVQA dataset contains 45336 questions posed over 28408 images. A Transformer based neural network with self attention mechanism is used for encoding the output vectors obtained by a combination of the above mentioned models. Additionally, Long Short-Term Memory (LSTM) is used for decoding the vector and generating the output sequence of characters. This model possessed the desired Multilingual Text-VQA functionality but did not have a sufficient number of Indian languages included in its dataset nor did it have a decently satisfying test-time accuracy. The SSBaseline dataset is tricky to work with and the OCR tokens had a defective reading ability. All of this added up forced us to look for an alternative model.

Proposed Techniques and Algorithms

We looked at various multimodal models and their overall success rates and found that the PaLI model is the most successful model.

Leaderboard Dataset

View overall by Date for All models All competition entries



Rank	Model	overall	Extra Training Data	Paper	Code	Result	Year	Tags
1	**PaLI**	73.1	×	PaLI: A Jointly-Scaled Multilingual Language-Image Model			2022	
2	**TAP**	53.97	×				2020	
3	**TAG**	53.69	×	TAG: Boosting Text-VQA via Text-aware Visual Question-answer Generation			2022	
4	**ssbaseline**	45.66	×				2020	
5	**SMA single model**	45.51	×				2020	
6	**SAM** (Single Model)	44.8	×				2020	
7	**colab_buaa**	44.73	×				2020	
8	**CRN** (Single Model)	40.96	×				2020	
9	**CIG**	40.77	×				2020	
10	**M4C**	40.46	✓				2020	
11	**Shuai**	39.95	×				2020	
12	**mmgnn**	32.46	×				2020	

With the goal of answering questions based on an image, PaLI aims to perform unimodal (language, vision) and multimodal (language and vision) tasks with the help of a simple and scalable main architecture. It uses an encoder-decoder transformer model, with a large-capacity ViT component for image processing. For the text encoder-decoder, it uses pre-trained mT5 models, while for the image encoder, it reuses large vanilla ViT models.

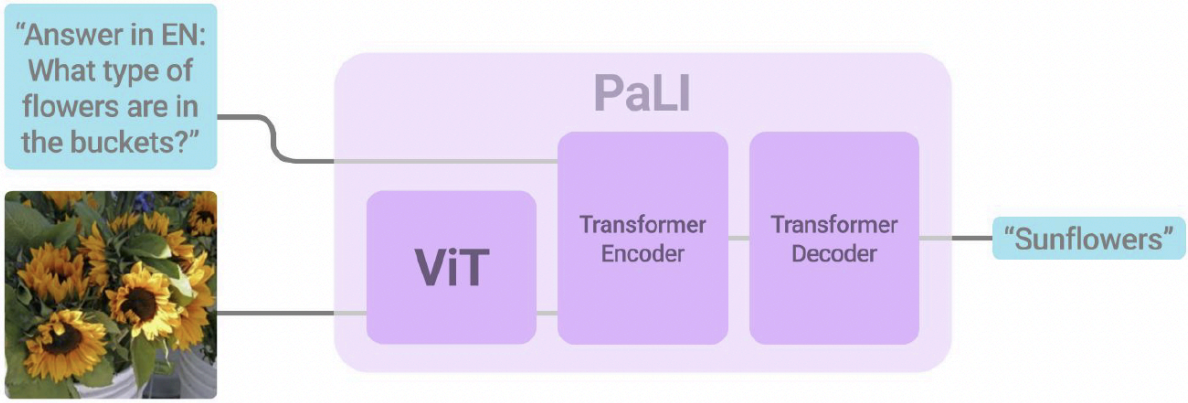


Figure above shows a high-level schematic of the model architecture. At its core, PaLI has a text encoder decoder Transformer. To include vision as input, the text encoder is fed with a sequence of visual 'tokens': output patch features of a Vision Transformer which takes as input an image. It uses previously trained unimodal checkpoints.

The visual component uses the largest vanilla ViT architecture to date, named ViT-e. ViT-e has the same architecture and uses the same training recipe as the 1.8B parameter ViT-G model, while scaling to 4B parameters. The only other difference is that it applies learning rate cool-down twice, once with and once without inception crop augmentation, and averages the weights of the two models.

The language component uses the mT5 backbone as the language component. Pre-trained mT5-Large (1B parameters) and mT5-XXL (13B parameters) were experimented and the language encoder-decoder of PaLI was initialized.

Overall, the model includes three model sizes:

1. PaLI-3B, where the language component is initialized from mT5-Large (1B parameters), and the vision component is ViT-G (1.8B parameters).
2. PaLI-15B, where the language component is initialized from mT5-XXL (13B parameters), and the vision component is ViT-G (1.8B parameters).
3. PaLI-17B, where the language model is initialized from mT5-XXL, and the vision component is the newly-trained ViT-e model (4B parameters).

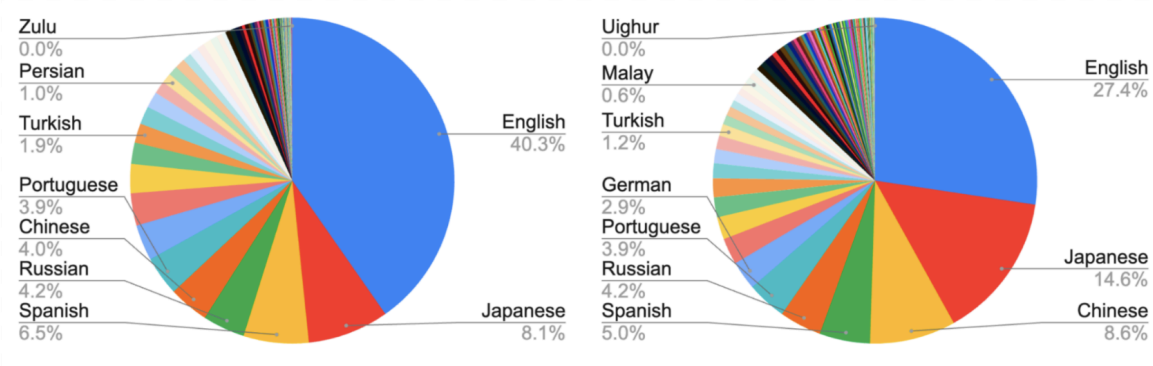
Data Set Used

Scaling studies for deep learning show that larger models require larger datasets to train effectively. In order to effectively utilize a model like PaLI-17B, we require a large dataset. To

unlock the potential of multilingual image-language pre-training, PaLI uses WebLI, a multilingual imagelanguage dataset built from images and texts available on the public web.

Sampled images from WebLI associated with multilingual alt-text and OCR. The second image is by jopradier (original), used under the CC BY-NC-SA 2.0 license. Remaining images are also used with permission.

WebLI scales up the image language data collection from English-only datasets to 109 languages, which enables us to pre-train PaLI multilingually, and perform downstream tasks across many languages.



Statistics of recognized languages from alt-text and OCR in WebLI.

Due to the abundance of multilingual content on the internet, the collection process for the WebLI dataset can be scaled to cover 10 billion images and 12 billion alt-texts. In addition to annotation with web text, publicly available automatic service is used to extract OCR annotations on all images, resulting in 29 billion image-OCR pairs. To balance quality and retain scale, the dataset was filtered to the highest quality subset retaining only the top 10% scoring of the original WebLI image-text pairs (about 1B examples), which were finally used to train PaLI.

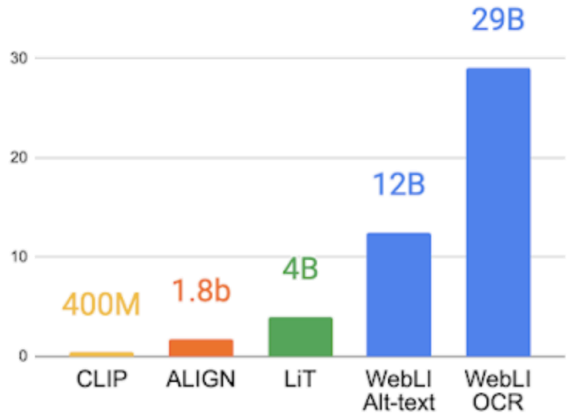


*Image-text pair counts of WebLI and other large-scale vision-language datasets, **CLIP**, **ALIGN** and **LiT**.*

Experiments and Results

Using the code available on the big_vision github page (https://github.com/google-research/big_vision/tree/main), we were able to enter the training phase of the PaLI model while using the ImageNet 2012 dataset. This is primarily a unimodal image to text training. The goal is to perform this training on image to text, text to text and then multimodal image+text to text.

The hardware available with us is a single Nvidia A100-PCIE-40GB GPU machine.

```
p20190065@a100:~/big_vision$ nvidia-smi
Tue Mar 26 22:09:53 2024
+-----------------------------------------------------------------------------------------+
| NVIDIA-SMI 550.54.15              Driver Version: 550.54.15      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-PCIE-40GB          Off |   00000000:2B:00.0 Off |                    0 |
| N/A   35C    P0             33W /  250W |      13MiB /  40960MiB |      0%      Default |
|                                         |                        |             Disabled |
+-----------------------------------------+------------------------+----------------------+

+-----------------------------------------------------------------------------------------+
| Processes:                                                                              |
|  GPU   GI   CI        PID   Type   Process name                              GPU Memory |
|        ID   ID                                                               Usage      |
|=========================================================================================|
|    0   N/A  N/A      2792      G   /usr/lib/xorg/Xorg                              4MiB |
+-----------------------------------------------------------------------------------------+
p20190065@a100:~/big_vision$
```

It took some time and availability of SUDO access for us to get the Nvidia CUDA drivers, CUDA Toolkit, CuDNN and TensorRT working. These were essential for the training of the model on the GPU.

We had significant challenges with respect to memory limitations. The original batch size was 1024 which was too large for the machine to handle due to RAM limitations and hence the training process killed before a single step of training took place. We changed the batch size in order to understand the effect it would have on the training process.

We were eventually able to run with a batch size of 4. This is obviously very small as compared to the original batch size of 1024. With a batch size of 4, the model would take 546 days to train which is an unreasonably long time to wait for. These are the results obtained after 950 steps out of a total of 28,537,988 steps.

```
I0402 23:38:49.033500 140387774026368 utils.py:1230] [950] global_schedule = 0.09489999711513519
I0402 23:38:49.034461 140387774026368 utils.py:1230] [950] l2_grads = 8.111105918884277
I0402 23:38:49.034769 140387774026368 utils.py:1230] [950] l2_params = 208.18948364257812
I0402 23:38:49.034918 140387774026368 utils.py:1230] [950] l2_updates = 0.1585773527622223
I0402 23:38:49.035069 140387774026368 utils.py:1230] [950] training_loss = 7.120850563049316
I0402 23:38:49.035208 140387774026368 utils.py:1230] [950] uptime = 8137.712212622981
I0402 23:38:49.035350 140387774026368 utils.py:1230] [950] examples_seen = 3800.0
I0402 23:38:49.035499 140387774026368 utils.py:1230] [950] progress = 3.328896206698244e-05
I0402 23:38:49.035681 140387774026368 utils.py:1230] [950] epoch = 0.002996006638519973
I0402 23:38:49.035818 140387774026368 utils.py:1230] [950] img/sec/core = 2.4312112450317405
I0402 23:38:49.035964 140387774026368 utils.py:1230] [950] core_hours_cpu = 0.4351216241886141
I0402 23:38:49.036088 140387774026368 utils.py:1230] [950] core_hours = 0.4351216241886141
I0402 23:38:49.036260 140387774026368 train.py:112] NOTE: Steps:950/28537988 [0.0%]
Walltime:2h15m (0s eval)
ETA:545d18h11m
Total train time:545d18h37m
```

Given the size of the dataset and the number of parameters, we believe we will need a larger GPU cluster.

Conclusion and Future Work

We are still working on the model. Having gotten the Nvidia GPUs working after a lot of hit & trial and a special SUDO access for the purpose, we are now exploring optimizing the training on the available hardware. A more comprehensive description of the same is below in the ‘Work Update’ section.

Work Update

Work Completed

We initially tried to set up the server by using the local installations of the various CUDA drivers and other dependencies required for running the model but were unsuccessful because of several clashes with the system-wide dependency versions. Got the Nvidia CUDA drivers, CUDA Toolkit, CuDNN and TensorRT working on server p20190065@172.24.16.136 after SUDO access was given to us for the server.

Had significant hardware challenges. We have an Nvidia A100 GPU with 40GB RAM but training this dataset requires a significantly higher amount of memory and processing power. The original model is configured with a batch size of 1024 but the best we could get running on the given hardware was a batch size of 4.

Work Remaining

Given that we have got the training initiated with a batch size of 4, we will explore changing configurations to support a higher batch size. One possibility for this is by running the model on distributed GPUs so that we can tap into the processing power of each one to speed up the training process.

Once we succeed in training the model in a reasonable amount of time we can work towards the next goal of expanding the model to be compatible with Indian languages which can be done by either diversifying the WebLI dataset or by appropriate fine-tuning of the model.

References

- https://github.com/google-research/big_vision/tree/main - Github repository with the code for PaLI model
- https://paperswithcode.com/paper/pali-a-jointly-scaled-multilingual-language - Paper by Google Research describing the PaLI model
- https://docs.nvidia.com/cuda/cuda-installation-guide-microsoft-windows/ - Cuda Installation guide
- https://developer.nvidia.com/cuda-12-0-0-download-archive?target_os=Linux&target_arch=x86_64&Distribution=Ubuntu&target_version=22.04&target_type=deb_network - used for downloading CUDA Toolkit 12.0 required for the server
- https://docs.nvidia.com/deeplearning/cudnn/installation/linux.html Guide for installing CUDA Deep Neural Network (CUDNN)
- https://docs.nvidia.com/deeplearning/tensorrt/install-guide/ NVIDIA Deep Learning TensorRT installation guide

- https://paperswithcode.com/sota/visual-question-answering-on-textvqa-test-1 Comparision of various models used for Text VQA
- https://github.com/ZephyrZhuQi/ssbaseline Github repository with the code for the SSBaseline model
- 2012.05153.pdf (arxiv.org) Research paper by Meta on the SSBaseline model