Report

Diagnosing Domain Robustness of VQA using LXMERT

CS490 Project Report

Shravya Suresh (210260046)

Under the Supervision of

Prof. Abir De [CSE]

Prof. Biplab Banerjee [CSRE]

November, 2023

Abstract

Machine Learning has significantly developed in recent times. With the advent of newer technologies such as Generative AI and Large Language Models, greater tasks can be completed due to enhanced computation power and intricate architectures, which paves the way for novel areas of research. One such emerging field of ongoing research is multi-modality and domain adaptation - how well a machine-learning model can adapt to unknown domains or generalise to diverse tasks it may have never encountered before. This essentially combined two originally independent machine learning tasks - image processing and natural language processing. Several models have been developed that dabble into this topic and combine the two powers of vision and language.

This RnD project aims to explore the same through diagnosing the domain robustness of the LXMERT model. The trajectory of the project begins with a thorough understanding of domain robustness and adaptability, visual question-answering, and the architecture of the LXMERT model, and culminates in fine-tuning LXMERT on three custom language datasets to determine its domain robustness through cross-domain adaptability.

Keywords: VQA, GQA, Visual Genome, Encoder, Question-Answering, Domain adaptability, Domain robustness, LXMERT, Fine tuning, Transformer

Acknowledgement

I extend my heartfelt gratitude towards my project guides, Prof. Abir De and Prof. Biplab Banerjee, and to Mr. Hassan for their continuous guidance throughout the course of this project. Their supreme and in-depth knowledge of the subject matter as well as their timely advice and direction helped me align my work in the right direction, and further my understanding of the topic of research.

Contents

1. Foundations - Model Robustness and Domain Adaptability

1.1 Robustness of a Model

In machine learning, robustness of a model refers to its ability to maintain its level of performance when faced with uncertainties or adversarial conditions. A robust model should be able to generalize well under any circumstances and provide reliable predictions even when dealing with novel inputs.

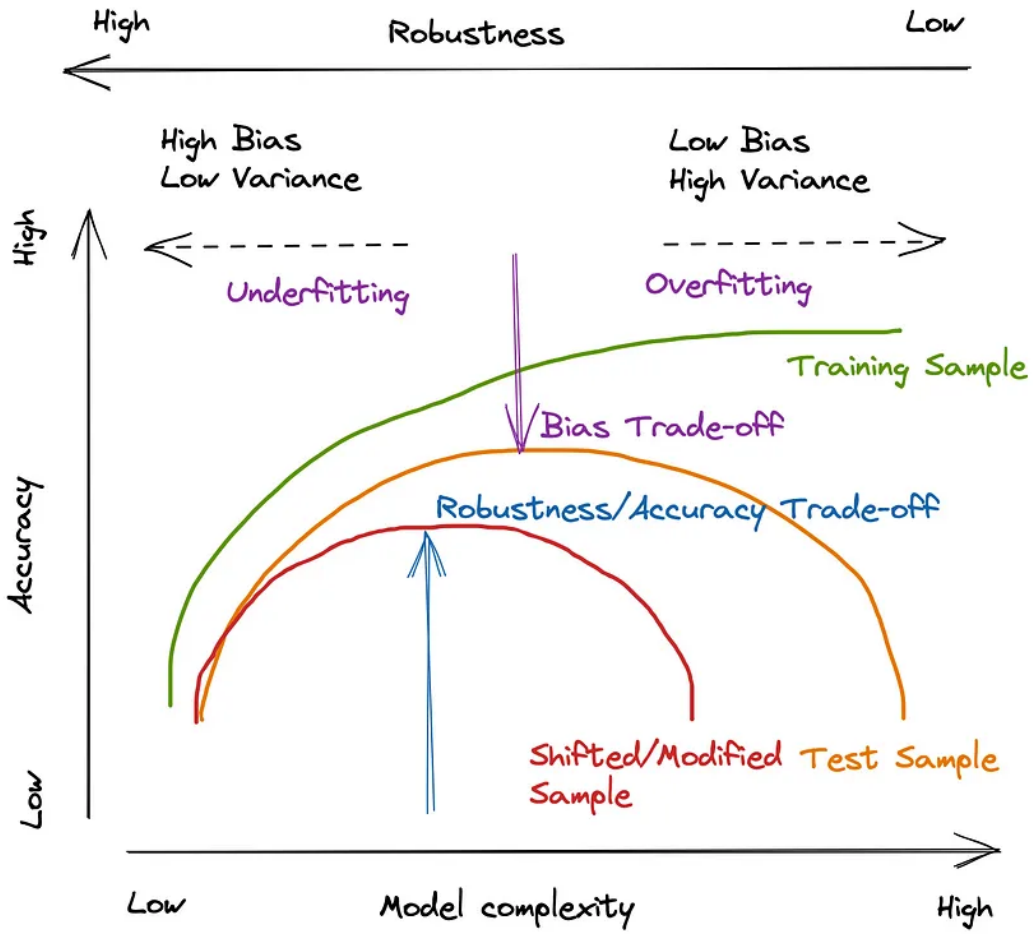
The real-world consequences of non-robust models can be dire, ranging from financial losses to compromised safety. For instance, an autonomous vehicle that relies on a non-robust object detection system could misinterpret road signs or fail to detect obstacles on the road, leading to serious accidents. Similarly, a non-robust fraud detection system might output false positives or negatives, which can further cause significant financial losses for businesses and consumers.

As machine learning becomes increasingly embedded in our daily lives, it is more and more prudent that our models are designed to be as robust as possible. In addition to ensuring accurate predictions, robust models also contribute to enhanced security, privacy, and user trust in AI systems.

1.2 Trade-off between Robustness and Accuracy

In the process of developing a machine learning model, there always comes a trade-off between robustness and accuracy of the model. Ideally, one would want to focus on engineering the model to achieve the highest possible accuracy on a given dataset, but in the process might overlook issues such as overfitting or a lack of generalization to new data. Overfitting occurs when a model is trained too well on the training data, hence capturing noise and random fluctuations instead of learning the underlying patterns. Consequently, the model performs poorly when exposed to new, unseen data - it does not generalise well across diverse tasks.

On the flip side, underfitting occurs when a model is too simple to capture the complexity of the data, resulting in suboptimal performance on both the training and test datasets. The model is not equipped enough to identify the foundational trend of data and adapt to generalised tasks. Hence, striking the right balance between robustness and accuracy is crucial for developing effective machine learning models.



The Trade-off between Robustness and Accuracy. Optimality is achieved when a fine balance is struck between robustness and accuracy (represented through the maxima peaks of the sample curves). Source: Medium

1.3 Challenges to achieve Robustness

Some challenges in achieving model robustness include:

1. Noisy data: Real-world data is often error-prone and noise, replete with inconsistencies and missing values (NaN, zero-values, etc). Achievening model robustness involves ensuring that the model can handle such data without compromising on its performance and accuracy.

2. Generalisability over other tasks: The data used to train a particular model may not always be representative of data it would encounter in real-world applications. Models need to be robust to such distribution shifts to maintain their performance over generalised tasks.

3. Adversarial attacks: Machine learning models are quite vulnerable to adversarial attacks - malicious attacks on data which are designed to passively manipulate results and hence cause misclassification in a machine learning pipeline. Robust models should be resilient to such attacks, and this can be achieved through various techniques such as imposing domain constraints on the data.

4. Model complexity: This is another trade-off to be dealt with. Overly complex models may be more prone to overfitting and less interpretable, while simpler models may struggle to capture the intricacies of the data. So while developing a model, one must keep in mind to strike a balance between complexity of the model and its robustness.

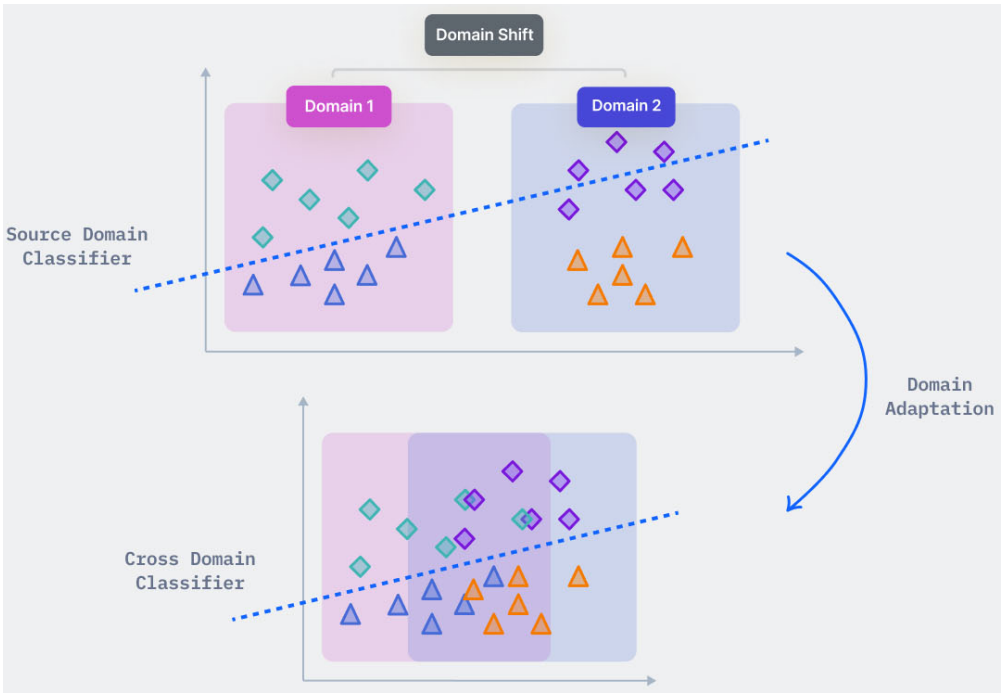
1.4 Suggested techniques to enhance Robustness

Some ways to enhance model robustness are:

1. Data augmentation and preprocessing: Data augmentation is a technique of synthetically increasing the size of the training data by creating modified copies of the dataset using the existing data within it. Preprocessing involves preliminary steps of refining the data to make it suitable to train the model. By augmenting and preprocessing data, we can improve the model's ability to handle noisy inputs as well as generalize to new data.

2. Data Regularization techniques: Data regularisation adds constraints to data, which further tackles issues like adversarial attacks and generalisation quality. Regularisation techniques such as L1 and L2 regularization, dropout, and early stopping, can lower the chances of overfitting and improve model robustness.

3. Ensemble learning and model diversity: Ensemble learning is a technique used to improve machine learning results by combining several models together. This approach allows us to generate better predictive performance as compared to a single model. Ensemble learning techniques such as bagging, boosting, and stacking, leverage this power to create a stronger, more robust predictor. Combining multiple models this way, with different strengths and weaknesses, can lead to a more robust overall system.



Domain Adaptation. Source: V7 Labs

4. Transfer learning and domain adaptation: Transfer learning is a technique that allows a model trained on one task to be fine-tuned for a related task, often with fewer and sometimes unseen training examples. Domain adaptation is a technique used to improve the performance of a model on a target domain by using the knowledge learned by the model from another related domain. It is essentially a form of unsupervised transfer learning where the model which has been trained on some given data is enhanced to perform adequately on unseen data or data of a different domain. Domain adaptation techniques enable models to adapt to distribution shifts and generalised tasks, hence enhancing their robustness to changes in the data.

2. Visual Question-Answering (VQA)

2.1 Definitions - Vision + Question-Answering

Visual Question Answering is the task of answering open-ended questions based on an image. It combines the specific dedicated tasks of both computer vision and natural language processing by allowing users to ask open-ended, multiple-choice, and common sense questions about the visual world. To answer these questions, the machine learning model requires an understanding of vision, language, and general contextual knowledge. VQA is hence quite a non-trivial concept of active research, because though artificial intelligence may be able to solve several specific tasks like such as image classification and sentiment analysis, it presents an interesting problem to combine two diverging concepts to solve a task which needs different types of data.

Examples of a VQA setting; each image has a dedicated question attached to it, whose answer is hidden within the image itself. Source: Domain-robust VQA with diverse datasets and methods but no target labels

2.2 Domain Adaptation for VQA

VQA is a step towards the completeness of AI: it combines perception (visual and linguistic) and cognition. The current focus of VQA research is highly directed at domain robustness and adaptability of VQA; how well can a VQA model generalise across diverse domains? Given its training dataset, can the trained model answer unseen questions about unseen visual data, based upon its learning over seen data? Several types of domain adaptability model exist, built upon diverse architectures -

1. Classical methods utilising CNN or LSTM embeddings (eg. Relational Networks)

2. Neuro-symbolic methods - learning by simply looking at images and reading its paired questions and answers (eg. Neuro-Symbolic Concept Learner (NS-CL))
3. Transformer methods (eg. LXMERT)

Despite the strong performance of recent VQA methods, they fall short of generalization over diverse tasks and true reasoning. Essentially, domain adaptation techniques cannot be directly applied effectively to a VQA model. The reason behind this is manifold. There is an inherent dataset bias; even though domain adaptation methods exist to tackle non-identical answer spaces, this setting is not easily obtainable since datasets tend to be highly specific or specialised according to the purpose that the data is fated to. This further leads to the use of domain-specific languages or domain-specific executable program annotations, or the requirement of VQA models to be trained separately for each new dataset. Another issue is that VQA models take inputs across multiple modalities (types of data - images, text) which each in turn bring a sort of domain specificity into the model. This makes domain shift a difficult problem to study. In addition, different VQA methods have multiple intermediate stages and individual processing steps over the diverse input types, while the visual and textual information also has to be processed side-by-side. This makes optimization challenging.

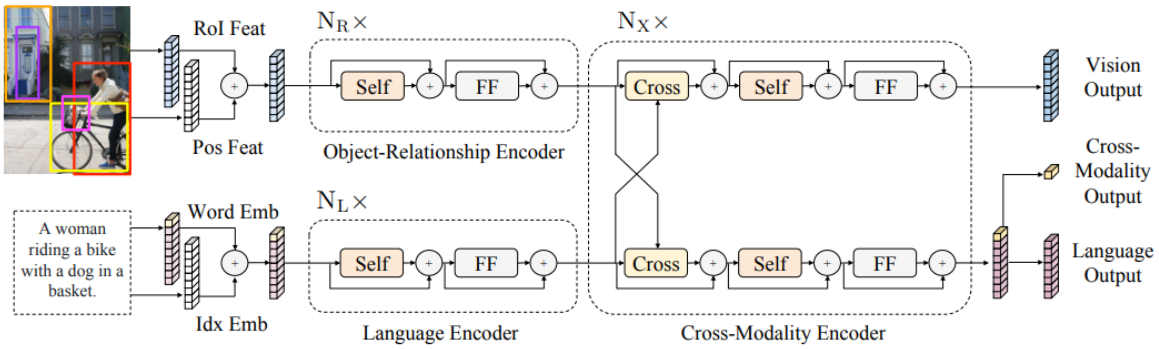
So, the question arises - just how robust are the existing models in generalisation across novel domains? For this RnD, I analysed the domain robustness of LXMERT - Learning Cross-Modality Encoder Representations from Transformers. I expect that LXMERT should perform more decently as compared to non-transformer models, given that it benefits from its massive pre-training.

3. LXMERT - Learning Cross-Modality Encoder Representations from Transformers

The LXMERT model is a transformer-based cross-modality model proposed by Hao Tan & Mohit Bansal. It focuses on learning vision-and-language interactions, especially for representations of a single image and its descriptive sentence. This framework is modeled after recent BERT-style innovations while further adapted to useful cross-modality scenarios. It is composed of a series of bidirectional transformer encoders - one for the vision modality, one for the language modality, and one to fuse both the modalities - which are pretrained using a combination of masked language modeling, visual-language text alignment, masked visual-attribute modeling, masked visual-object modeling, and visual-question answering objectives. The model has been pre-trained on a wide range of standard multi-modal datasets, including MSCOCO, Visual-Genome + Visual-Genome Question Answering, VQA 2.0, and GQA, to achieve a state-of-the-art status.

3.1 Architecture of the Model

The LXMERT model is built upon self-attention and cross-attention layers, keeping in line with the recent trend of designing natural language processing models using transformers. Each image is represented as a sequence of objects, and each sentence is represented as a sequence of words. The overall architecture of this model can be divided into the following components:



The architecture of the LXMERT model. 'Self' and 'Cross' are abbreviations for self-attention sub-layers and cross-attention sub-layers, respectively. 'FF' denotes a feed-forward sub-layer. Source: LXMERT: Learning Cross-Modality Encoder Representations from Transformers

3.1.1 Input Embeddings

The input embedding layers in LXMERT convert the image and the sentence inputs into two sequences of features: object-level image embeddings and word-level sentence embeddings respectively.

1. **Word-Level Sentence Embeddings**: A sentence is first split into words w_1, . . . , w_n through tokenisation. Next, each word w_i and its index i (w_i's absolute position in the sentence) are projected to vectors by embedding sub-layers, and then added to the index-aware word embeddings as follows:

   $\hat{w}_i = \text{WordEmbed } (w_i)$
   $\hat{u}_i = \text{IdxEmbed } (i)$
   $\hat{h}_i = \text{LayerNorm } (w_i + u_i)$

2. **Object-Level Image Embeddings**: Instead of using the feature map output by a convolutional neural network, the authors took the features of detected objects as the embeddings of images.

   Specifically, the object detector detects m objects o_1, . . . , o_m from the image (an example is denoted by bounding boxes on the image in the LXMERT model). Each object o_j is represented by its position feature (the coordinates of its bounding box) p_j and its 2048-dimensional region-of-interest (RoI) feature f_j (the portion of the image to be filtered or operated on). But instead of directly using the RoI feature f_j without considering its position p_j, the authors learnt a position-aware embedding v_j by adding outputs of 2 fully-connected layers:

   $\hat{f}_j = \text{LayerNorm } (W_F f_j + b_F)$
   $\hat{p}_j = \text{LayerNorm } (W_P p_j + b_P)$
   $\hat{v}_j = (f_j + p_j)/2$

   Since the image embedding layer and the following attention layers are agnostic (unbiased) to the absolute indices of their inputs, the order of the object does not need to be specified. The layer normalization ($\hat{f}_j$ and $\hat{p}_j$ is applied to the projected features before summation so as to balance the energy of the two different types of features.

3.1.2 Encoders

The encoders used in the model are the language encoder, the object-relationship encoder, and the cross-modality encoder. They are designed mostly on the basis of two kinds of attention layers: self-attention layers and cross-attention layers.

1. **Attention Layers**: Attention layers are designed to retrieve information from a set of context vectors y_j which are related to a query vector x. An attention layer first calculates the matching score a_j between the query vector x and each context vector y_j. These scores are then normalized using the softmax activation function as follows:

   $a_j = \text{score}(x, y_j)$
   $\alpha_j = \exp(a_j) / \sum_k \exp(a_k)$

   The output of an attention layer is the weighted sum of the context vectors with

respect to the softmax-normalized score:

$\text{Att}_{X \hookrightarrow Y}\ (x,\ y_j) = \sum_j \alpha_j y_j$

An attention layer is considered one of self-attention when the query vector x is present in the set of context vectors y_j.

2. **Single-Modality Encoders**: After the embedding layers, we first apply two transformer encoders, a *language encoder* ($\mathbf{N}_L$) and an *object-relationship* ($\mathbf{N}_R$) encoder. Each of them only focuses on a single modality (language or vision). Thus, the transformer encoder is applied to both vision inputs and language inputs. Each of these encoders contains a self-attention ('Self') sub-layer and a feed-forward ('FF') sub-layer, where the feed-forward sub-layer is further composed of two fully-connected sub-layers. The authors connected the sub-layers through a residual connection and layer normalization (annotated by the '+' sign in the model architecture).

3. **Cross-Modality Encoder**: The cross-modality encoder is composed of cross-modality layers, self-attention sub-layers and feed forward sub-layers. Each cross-modality layer consists of one bi-directional cross-attention sub-layer, two self-attention sub-layers and two feed-forward sub-layers. Using the output of the k^{th} layer as the input of $(k+1)^{th}$ layer, the authors stacked the cross-modality layers in their encoder implementation. Inside the k^{th} layer, the bi-directional cross-attention sub-layer ('Cross') is first applied, which contains two unidirectional cross-attention sub-layers: one from language to vision and one from vision to language. The query and context vectors are the outputs of the $(k-1)^{th}$ layer - the language features $\{h_i^{k-1}\}$ and vision features $\{v_i^{k-1}\}$):

$\hat{h}_i^k = \text{CrossAtt}_{L \hookrightarrow R}\ (h_i^{k-1},\ \{v_1^{k-1}, ..., v_m^{k-1}\})$
$\hat{v}_i^k = \text{CrossAtt}_{R \hookrightarrow L}\ (v_i^{k-1},\ \{h_1^{k-1}, ..., h_m^{k-1}\})$

The cross-attention sub-layer is used to exchange information and align entities between the two modalities so the model can learn joint cross-modality representations. Next, to build internal connections, the self-attention sub-layers ('Self') are applied to the output of the cross-attention sub-layer as follows:

$\tilde{h}_i^k = \text{SelfAtt}_{L \hookrightarrow L}\ (\hat{h}_i^k,\ \{\hat{h}_1^k, ..., \hat{h}_m^k\})$
$\tilde{v}_i^k = \text{SelfAtt}_{R \hookrightarrow R}\ (\hat{v}_i^k,\ \{\hat{v}_1^k, ..., \hat{h}_m^k\})$

Lastly, the k^{th} layer output $\{h_i^k\}$ and $\{v_i^k\}$ are produced by feed-forward sub-layers ('FF') on top of $\{\tilde{h}_i^k\}$ and $\{\tilde{v}_i^k\}$. The model is completed with a residual connection and layer normalization after each sub-layer, similar to how the authors designed the single-modality encoders.

3.2 Training and Testing

The LXMERT model has three outputs - one for language, one for vision, and one for cross-modality. The language and vision outputs are the respective feature sequences generated by the cross-modality encoder.

In order to better learn the cross-modal alignments between vision and language, the authors pre-trained the model with five diverse representative tasks:

1. Masked cross-modality language modeling
2. Masked object prediction via RoI-feature regression
3. Masked object prediction via detected-label classification
4. Cross-modality matching
5. Image question answering

Different from single-modality pre-training (e.g., Masked Language Modelling in BERT), this multi-modality pre-training allows the model to infer masked features either from the visible elements in the same modality, or from aligned components in the other modality.

The authors pre-trained the LXMERT model on a large aggregated dataset derived from five vision-and-language datasets whose images come from MS COCO or Visual Genome. Besides the two original captioning datasets, they also aggregated three large image question answering (image QA) datasets: VQA v2.0, GQA balanced version, and Visual-Genome QA (VG-QA). In terms of tokens, the pre-training data contain around 100M words and 6.5M image objects. The authors then used three datasets for evaluating the LXMERT framework: VQA v2.0 dataset, GQA, and NLVR2.

Method	VQA				GQA			$NLVR^2$	
	Binary	Number	Other	**Accu**	Binary	Open	**Accu**	Cons	**Accu**
Human	-	-	-	-	91.2	87.4	89.3	-	96.3
Image Only	-	-	-	-	36.1	1.74	17.8	7.40	51.9
Language Only	66.8	31.8	27.6	44.3	61.9	22.7	41.1	4.20	51.1
State-of-the-Art	85.8	53.7	60.7	70.4	76.0	40.4	57.1	12.0	53.5
LXMERT	**88.2**	**54.2**	**63.1**	**72.5**	**77.8**	**45.0**	**60.3**	**42.1**	**76.2**

Test-set results. VQA/GQA results are reported on the 'test-standard' splits and $NLVR^2$ results are reported on an unreleased test set. The highest method results are in bold. Source: LXMERT: Learning Cross-Modality Encoder Representations from Transformers

The test results effectively demonstrate the performance of LXMERT in a VQA problem statement. Now the question arises: how does it perform under domain adaptation?

4. The Task at hand

LXMERT is a state-of-the-art model for the task of Visual Question-Answering. And especially given the recent interest in domain robustness and adaptability of VQA across modalities and generalisation across diverse tasks, the aim of this RnD project is to determine how adaptable VQA is across other domains. For this I utilise the LXMERT model for a cross-domain adaptation. Essentially, I utilise the pre-trained weights of LXMERT and fine-tune it on three custom answer datasets in three different domains - GQA, Visual-Genome and VQA Abstract - and evaluate its performance on each of them.

The code for this project can be found at this Github repository.

4.1 Dataset

As described in the previous section, LXMERT has been pre-trained on a diverse variety of standardised VQA datasets. For the purpose of this RnD project, one common language dataset `vocab_common.txt` and three fine-tuning language datasets have been prepared. All three fine-tuning datasets are abstractions or concise versions of their parent language datasets; they are comprised of the top 1000 most frequently occurring answers only:

1. **VQA Abstract (`vocab_common_ab.txt`)**: This is an abstracted or concise version of the original VQA dataset used to pre-train LXMERT. The utility of this language dataset is to test how well the model can adapt to unseen information, given only a part of the whole dataset.
2. **GQA (`vocab_common_gqa.txt`)**: The GQA dataset is a vision and language dataset created by Stanford for the purpose of Question-Answering on image scene graphs. For this project, we created a concise version of the original VQA language dataset to test how LXMERT can adapt across domains.
3. **Visual Genome (VG) (`vocab_common_vg.txt`)**: This is another standardised dataset over which LXMERT has been pre-trained. Visual Genome contains Visual Question Answering data in a multi-choice setting. We again generated an abstracted version of the original language dataset and I fine-tuned LXMERT on it to test it's cross-domain adaptability.

   The dataset can be accessed here.

4.2 Methodology

First, I cloned the existing repository housing the pre-trained LXMERT model at this link. Then, I downloaded the pre-trained weights found at this link as directed by the authors. After that, I downloaded the required image files and completed the overall required set-up.

Next, I prepared the custom answer datasets by transforming them into a suitable format for fine-tuning. I generated 2 JSON files corresponding to each dataset - one

as answer-to-label form, and one as label-to-answer form - as utilised by LXMERT. So in total, 6 JSON files were generated to be utilised during fine-tuning.

Once the setup was completed, I first ran the trial fine-tuning of LXMERT as directed by the authors. Once this was complete, I sought out to fine-tune the model over the three custom answer datasets.

4.3 Modifications to suit Custom data

In order to fine-tune LXMERT on the custom datasets, several modifications had to be made to the previous edition of the code. In the codebase on the Github repository, I have renamed the pre-finetuning versions of the changed files as '<filename>_old.py', while the modified files used to fine-tune the custom data have assumes the original file names.

As mentioned previously, the custom answer datasets are abstractions of the original standardised datasets on which LXMERT was originally pre-trained. So, these datasets would certainly be missing a lot of answers on which LXMERT was trained, and such cases would arise during execution of the code. If not handled, this would continuously cause Key Errors due to missing values. Hence, I included several flagging conditions in various places to handle cases when answers are not a part of the custom fine-tuning dataset. One such example has been presented below:

Original

```
# Only kept the data with loaded image features
self.data = []
for datum in self.raw_dataset.data:
    if datum['img_id'] in self.imgid2img:
        self.data.append(datum)
print("Use %d data in torch dataset" % (len(self.data)))
```

Modified

```
# Only kept the data with loaded image features
self.ans2label = json.load(open("data/vqa/finetune_common_gqa_ans2label.json"))
self.label2ans = json.load(open("data/vqa/finetune_common_gqa_label2ans.json"))
self.data = []
for datum in self.raw_dataset.data:
    if datum['img_id'] in self.imgid2img:
        if datum['label'] and list(datum['label'].keys())[0]  in self.ans2label:
            self.data.append(datum)
```

Original and Modified Code to suit the fine-tuning of custom data.

I also had to modify and correct the datatypes of certain variables as derived from the custom data to suit the existing code. One such example is as follows:

```
for qid, l in zip(ques_id, label.cpu().numpy()):
    if l in list(dset.label2ans.keys()):
        ans = dset.label2ans[l]
        quesid2ans[qid.item()] = ans
```

Original

```
for qid, l in zip(ques_id, label.cpu().numpy()):
    if l in list(int(i) for i in dset.label2ans.keys()):
        ans = dset.label2ans[str(l)]
        quesid2ans[qid.item()] = ans
```

Modified

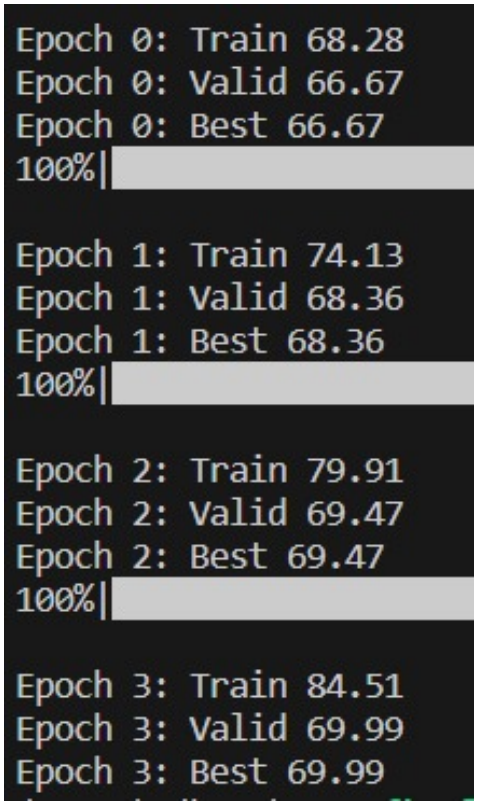
Original and Modified Code to correct the data type of custom data.

Having suitably modified the code, I then fine-tuned each of the three answer datasets - `vocab_common_ab.txt`, `vocab_common_gqa.txt` and `vocab_common_vg.txt` - and observed the results.

5. Performance and Analysis

5.1 Results of Original Fine-Tuning

I ran the original fine-tuning of LXMERT overnight. It took around 8 hours (2 hours per epoch * 4 epochs) to complete. It yielded a convergent validation accuracy of **69.99%**.

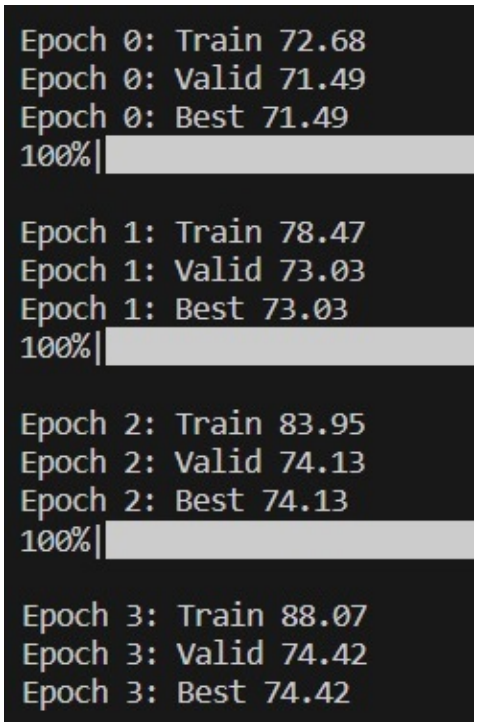


Validation accuracy of the original fine-tuning.

5.2 Results of Custom Fine-Tuning

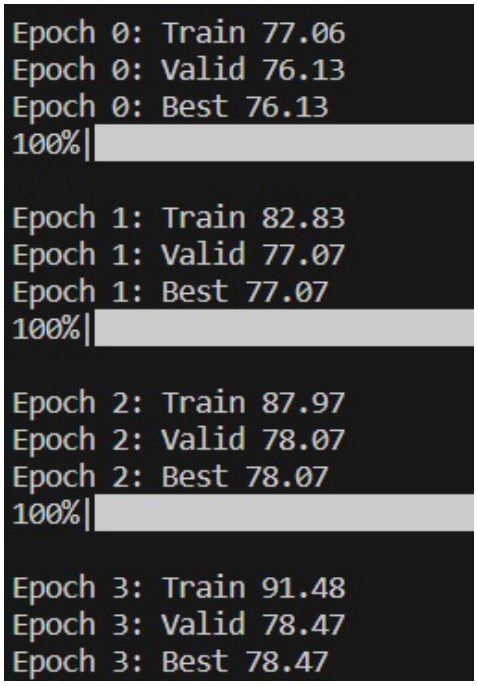
The fine-tuning of the custom answer datasets took around 2-3 hours each (30-40 minutes per epoch * 4 epochs).

The VQA Abstract custom dataset converged to a validation accuracy of **74.42%**, showing an improvement from the original fine-tuning.



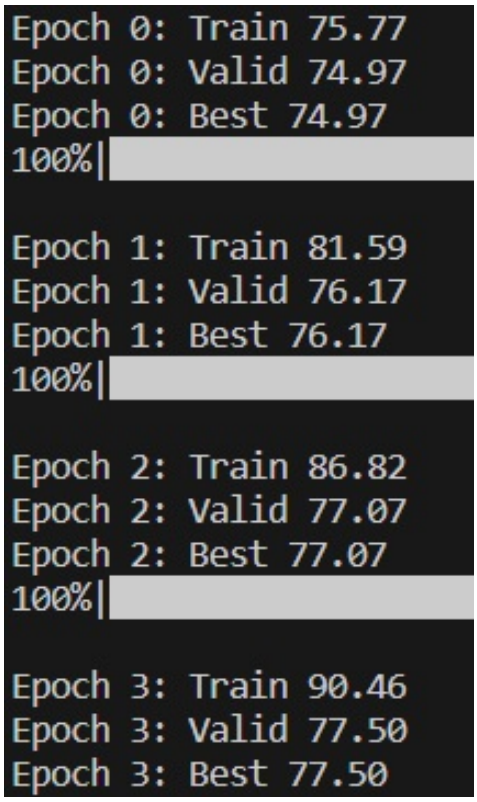
Validation accuracy of fine-tuning the VQA Abstract custom answer dataset.

The GQA custom dataset converged to a validation accuracy of **78.47%**, showing an improvement from the original fine-tuning as well as from the VQA Abstract dataset.



Validation accuracy of fine-tuning the GQA custom answer dataset.

The Visual Genome custom dataset converged to a validation accuracy of **77.50%**, showing an improvement from the original fine-tuning and VQA Abstract but not GQA.



Validation accuracy of fine-tuning the Visual Genome custom answer dataset.

6. Inference and Diagnosis

Thus, the LXMERT model does showcase the property of Domain Robustness. LXMERT performs decently well under cross-domain adaptation. This is evident from the results obtained through the cross-domain fine-tuning of the custom answer datasets, with GQA yielding the highest validation accuracy of 78.47%. This project thus suggests the domain robustness of state-of-the-art models like LXMERT, and opens the doors to further research and exploration on the solving and applications of multimodal tasks involving domain adaptation.

7. References

- Domain-robust VQA with diverse datasets and methods but no target labels
- Super-CLEVR: A Virtual Benchmark to Diagnose Domain Robustness in Visual Reasoning
- Understanding Machine Learning Robustness: Why It Matters and How It Affects Your Models
- Expect The Unexpected: The Importance of Model Robustness
- On the Robustness of Domain Constraints
- Domain Adaptation in Computer Vision: Everything You Need to Know
- Visual Question Answering
- Visual Question Answering — A Deep Learning Classification Case Study
- Visual Question Answering: A Survey on Techniques and Common Trends in Recent Literature
- A simple neural network module for relational reasoning
- The Neuro-Symbolic Concept Learner: Interpreting Scenes, Words, and Sentences From Natural Supervision
- LXMERT: Learning Cross-Modality Encoder Representations from Transformers
- The Attention Mechanism from Scratch
- The GQA Dataset
- The pre-trained LXMERT Model
- My code base