POLITECNICO DI MILANO

School of Industrial and Information Engineering

Master of Science in Computer Science and Engineering

Dipartimento di Elettronica, Informazione e Bioingegneria

Explainable Multimodal Fusion

Supervisor: Prof. Paolo Cremonesi

Co-Supervisor: Prof. Amir H. Payberah

Advisor: Karl Fredrik Erliksson

Thesis by:

Jaweriah Alvi, 938690

Academic Year 2020-2021

Abstract

Recently, there has been a lot of interest in explainable predictions, with new explainability approaches being created for specific data modalities like images and text. However, there is a dearth of understanding and minimal exploration in terms of explainability in the multimodal machine learning domain, where diverse data modalities are fused together in the model. In this thesis project, we look into two multimodal model architectures namely single-stream and dual-stream for the Visual Entailment (VE) task which compromises of image and text modalities. The models considered in this project are UNiversal Image-TExt Representation Learning (UNITER), Visual-Linguistic BERT (VL-BERT), Vision-and-Language BERT (ViLBERT) and Learning Cross-Modality Encoder Representations from Transformers (LXMERT). Furthermore, we conduct three different experiments for multimodal explainability by applying the Local Interpretable Model-agnostic Explanations (LIME) technique. Our results show that UNITER has the best accuracy among these models for the problem of VE. However, the explainability of all these models is similar.

Keywords

Multimodal, Explainability, Fusion, Interpretability

Acronyms

BERT Bidirectional Encoder Representation from Transformer

DeepLIFT Deep Learning Important Features

ITM Image Text Matching

IG Integrated Gradients

LIME Local Interpretable Model-agnostic Explanations

LXMERT Learning Cross-Modality Encoder Representations from Transformers

MLM Masked Language Modeling

MRM Masked Region Modeling

NLP Natural Language Processing

NLVR Natural Language for Visual Reasoning

NSP Next Sentence Prediction

R-CNN Region Based Convolutional Neural Networks

SHAP SHapley Additive exPlanations

SNLI Standford Natural Language Inference

UNITER UNiversal Image-TExt Representation Learning

VE Visual Entailment

ViLBERT Vision-and-Language BERT

VL-BERT Visual-Linguistic BERT

VOLTA VisiOLinguistic Transformer Architectures

VQA Visual Question Answering

WRA Word Region Alignment

Contents

Chapter 1

Introduction

Over the last decade, Deep Neural Networks have achieved new state-of-the-art performance across a wide range of machine learning problems. However, the size and complexity of these models make them opaque and difficult to understand, which limit the adoption for many real-world use cases. For instance in medical decision support systems, it is often not sufficient to solely provide a prediction, but a doctor would need some reasoning or explanation for the prediction to trust the model and use it in practice. Explaining a model's decision process can also be used as a tool for data scientists to debug and improve a model, and to detect potential data issues and underlying biases to help build fair AI systems. Explainable Predictions have received a lot of interest lately, with novel explainability methods being developed for specific data modalities like images and text. However, in the multimodal setting where different data modalities are fused together in the model, there is a lack of understanding and little explored in terms of explainability. Multimodal fusion in itself is a challenging task that has to deal with potential redundancies and varying quality between modalities, and sometimes the modality fusion can even hurt performance. This makes it even more important and interesting to try to understand and explain the fusion process.

1.1 Background

Multimodal machine learning is a thriving multi-disciplinary research area that integrates and models multiple communicative modalities, such as textual, auditory, and visual messages, to meet some of artificial intelligence's original goals. Given

the heterogeneity of the data and the contingency frequently encountered between modalities, this research area has presented some specific challenges for multimodal researchers, beginning with audio-visual speech recognition and more lately with language and vision projects such as image and video captioning.

1.2 Problem

There are many multimodal models available. The differences between them are the pre-training data, hyperparameters, initialization and the architecture. There are two main architecture of multimodal models namely single-stream and dual-stream encoders. We would like to find out which architecture is best in case of explainability?

Research Questions
Does single stream model perform better or dual stream model for visual entailment? How can explainability methods be used to understand what the multimodal models learn and are there any differences in learning between the two model types?

1.3 Purpose

Not much work has been done on explainability for multimodal models. If there is any work, that is with reference to a particular problem. Our thesis will help in making informed decisions on which model (single or dual-stream) to select.

1.4 Goal

The goal of this project is to compare and analyse different single-stream and dual-stream multimodal models. This has been divided into the following sub-goals:

1. Find which multimodal architecture is better for visual entailment problem.

2. Find which multimodal architecture has best explainability.

1.5 Benefits, Ethics and Sustainability

To be able to use any system, trust is highly important. This project will help in multimodality domain. One example in the area of medicine could be a system which takes the x-ray report (text) and image as input and then outputs the diagnosis. In this case explainability can help in highlighting the relevant text and image areas on which the diagnosis (prediction) has been made. Moreover, explainability can also help us in identifying biases in the system.

1.6 Outline

Chapter 2 describes the theoretical background of the thesis project. It explains the Transformers, multimodality, the single-stream and dual-stream models and explainability along with the relevant work in the domain. Chapter 3 describes the experiments performed and the technical limitations faced. Chapter 4 discusses the result of the experiments. Chapter 5 concludes our thesis project and discusses the future work that could be done for this project.

Chapter 2

Theoretical Background

This chapter provides basic background information about Transformers, different multimodal models and explainability. Additionally, this chapter also describes related work in multimodal models and explainability.

2.1 Transformers

Transformers were first introduced in the paper "Attention is all you Need" [23]. The Transformers is a type of neural network architecture and is most popular for Natural Language Processing (NLP). Transformers primarily use the attention mechanism and avoid recurrence. This is one of the reasons why the training can be heavily parallelized and thus train much faster. The Transformer model (Fig: 2.1.1) uses an encoder-decoder architecture. The input is feed into the encoder. The encoder learns the representation and then sends that representation to the decoder. The decoder receives the representation and then generates the output.

The Transformer components are explained below:

- **Encoders:** Transformer consists of N number of encoder layers. In the paper [23] they have used 6 encoders stacked together. The output of one encoder is sent to another encoder above it. Each encoder has two sub-layers namely, Multi-head attention and Feed-forward network.
- **Decoders:** Transformer also consists of N number of decoder layers. In the paper [23] they have used 6 decoders stacked together. The decoder receives two inputs, one from the encoder representation and one from the previous decoder.

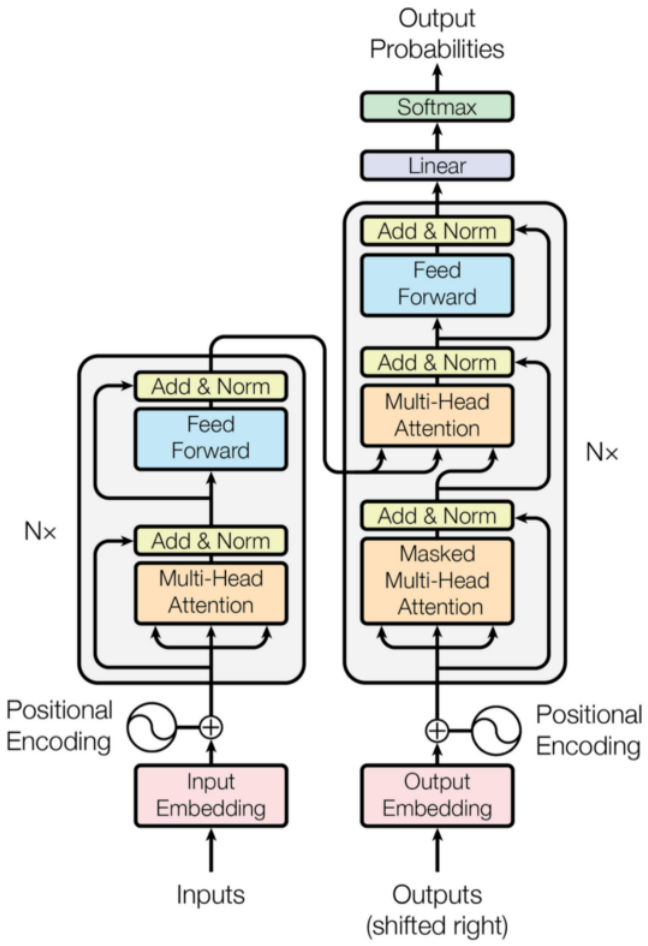


Figure 2.1.1: The Transformer - model architecture [23]

The decoder consists of one more additional sub-layer compared to the encoder which is the Masked multi-head attention layer.

- **Multi-head attention:** Multi-head attention uses multiple attention heads in order to provide an accurate attention matrix.
- **Feed-forward network:** The feed-forward network is made of two linear transformations with ReLU activation. While the linear transformations are the same across different positions, the parameters used in each layer change [23].
- **Masked multi-head attention:** It is same as multi-head attention except that during test the decoder can only see the words generated till the previous input step [15].

Bidirectional Encoder Representation from Transformer (BERT): BERT was introduced by Google in the paper "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding" [7]. BERT has been a state of the

art model and has provided excellent results for a variety of NLP tasks including sentence classification, question answering, text generation and many others. BERT's popularity is due to the fact that it is a context-based embedding model, as opposed to other common embedding models like word2vec, which is context-free [15].

BERT model is first trained on a large dataset for a specific task and the trained model is saved. For a new task we can use the saved weights of the previously trained model (pre-trained model) instead of initializing the new model with random weights. In this case instead of training a new model from scratch the pre-trained model is used and the weights are tweaked (fine-tuned) for the new down-stream task. The pre-training and fine-tuning of BERT is explained below:

- **Pre-training BERT:** Before feeding the input sequence to BERT, the input text is first embedded using three embedding layers namely Token embedding, Segment embedding and Position embedding. The token embedding layer first tokenizes the input sequence, then adds the classification token `[CLS]` at the beginning of the first sentence and then adds a separator token `[SEP]` at the end of every sentence. The segment embedding layer distinguishes between the two given sentences. It receives the input tokens and then returns either E_A if token belongs to sentence A or E_B if token belongs to sentence B as output. As the transformer can process all the words in parallel, the position embedding layer helps in marking the position of the word.

  BERT uses Masked Language Modeling (MLM) and Next Sentence Prediction (NSP) tasks for pre-training. The goal of the MLM is to predict the original vocabulary id of the masked word based only on its context, whereas NSP task jointly pre-trains text-pair representations [7].

  In MLM task, 15% of the words are randomly mask. To avoid any discrepancy between pre-training and fine-tuning the 80-10-10% rule is followed in masking which is 80% are replaced with `[MASK]`, 10% token with a random token and for the remaining 10% no changes are made.

  In NSP task two sentences are given as input and it finds the relationship between them. 50% of the time the sentence `B` is the actual next sentence of sentence `A` (labeled as `IsNext`) and remaining 50% of the time it is a random sentence (labeled as `NotNext`).

- **Fine-tuning BERT:** By swapping out the appropriate inputs and outputs BERT can be fine-tuned for different downstream tasks. Simply insert the task-specific inputs and outputs into BERT and fine-tune all of the parameters end-to-end for each task.

2.2 Multimodal models

Modality refers to the way something occurs or is experienced [4]. The aim of multimodal machine learning is to create models that can process and relate data from a variety of sources for instance images, text and video etc.

Multimodal machine learning has five main challenges [4]:

1. *Representation* - how to represent and summarize multimodal data in a way that exploits the complementarity and redundancy of multiple modalities.
2. *Translation* - how to translate (map) data from one modality to another. Not only is the data heterogeneous, but the relationship between modalities is often open-ended or subjective.
3. *Alignment* - to identify the direct relations between (sub)elements from two or more different modalities.
4. *Fusion* - to join information from two or more modalities to perform a prediction.
5. *Co-learning* - to transfer knowledge between modalities, their representation, and their predictive models.

In this thesis, we primarily consider data modality fusion and how it can be used for prediction tasks. There are two types of Fusion techniques:

1. *Early Fusion*: In this technique the features from each modality are combined at the start and then the full model architecture is applied on the combined features.
2. *Late Fusion*: In this technique the individual modalities are run through their own architecture and the outputs are combined at the end to make a prediction.

2.2.1 Single-stream models

In single-stream models, concatenation of image and text features are provided to a standard BERT architecture [5]. It comes under the category of early fusion.

UNITER:

UNITER [6] has a single-stream architecture model (Fig: 2.2.1). UNITER first uses an Image Embedder to encode image regions (visual features and bounding box features) and a Text Embedder to encode textual words (tokens and positions) into a single embedding space. Then it applies a Transformer module to learn the generalizable contextualized embeddings for each region and each word with the help of pre-training tasks [6].

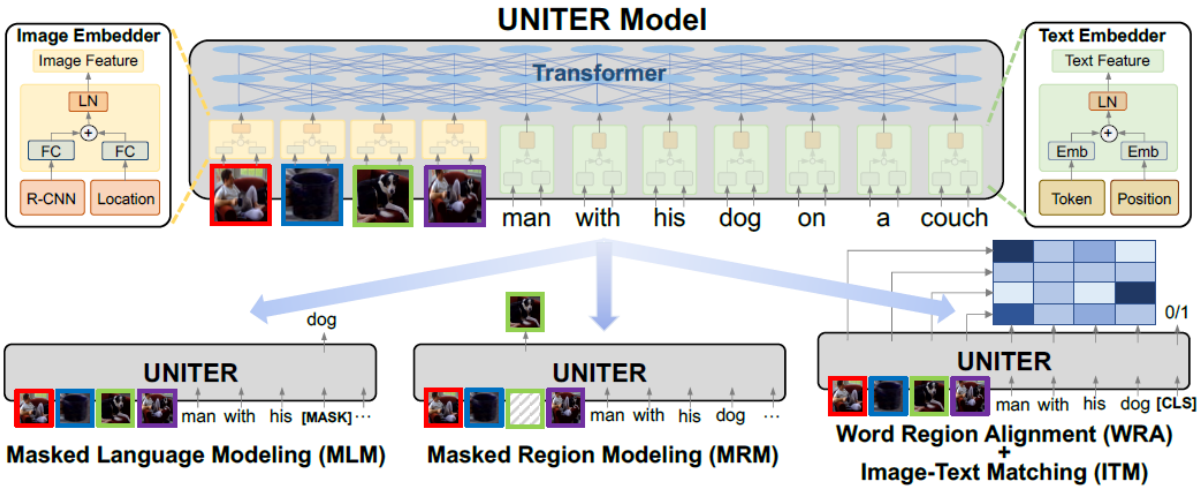


Figure 2.2.1: UNITER model [6]

UNITER is pre-trained on four image-text datasets namely COCO [10], Visual Genome [8], Conceptual Captions [18] and SBU Captions [14]. It uses four pre-training tasks: MLM, Masked Region Modeling (MRM) with three variants, Image Text Matching (ITM), and Word Region Alignment (WRA) [6].

UNITER achieved state of the art across six Vision + Language tasks (over nine datasets), including Visual Question Answering, Image-Text Retrieval, Referring Expression Comprehension, Visual Commonsense Reasoning, Visual Entailment, and Natural Language for Visual Reasoning (NLVR) [6].

VL-BERT:

The VL-BERT model is a single-stream architecture model (Fig: 2.2.2). It is based on a unified multi-modal Transformer architecture which takes both visual and linguistic embedded features as input. The main differences in the VL-BERT model are:

1. Sentence-Image Relationship prediction is not incorporated as it is of no help in pre-training visual-linguistic representations.
2. VL-BERT is pre-trained on both large visual-linguistic corpus (Conceptual Captions dataset) and a text-only (BooksCorpus) dataset.
3. In VL-BERT, the parameters of Fast Region Based Convolutional Neural Networks (R-CNN), deriving the visual features, are also updated. There is improved tuning of the visual representation.

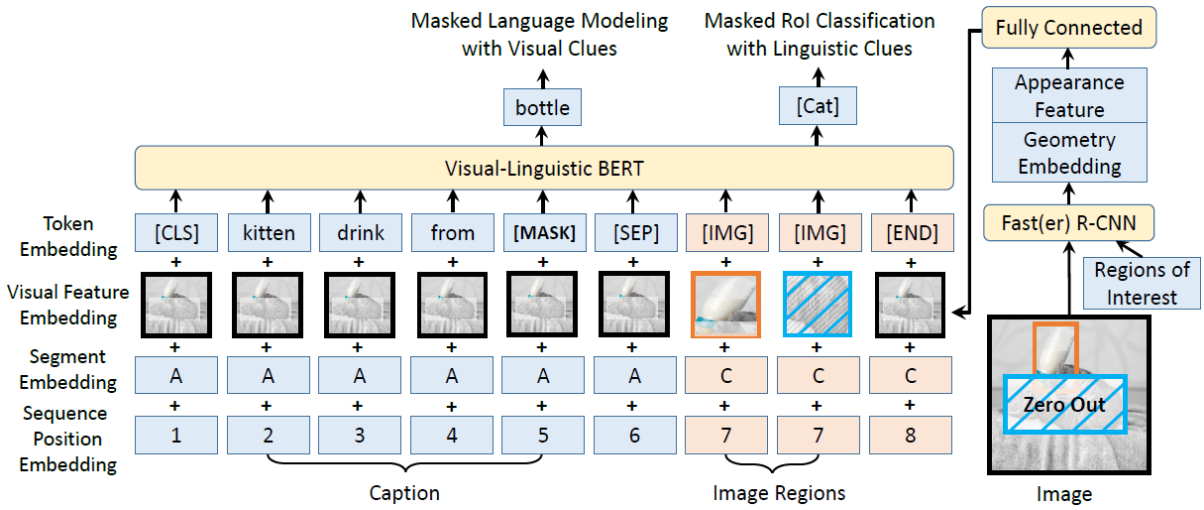


Figure 2.2.2: VL-BERT model [20]

With the exception of LXMERT, VL-BERT outperformed the other concurrent works. This is due to the fact that LXMERT has been pre-trained on a large amount of visual question answering data. While VL-BERT is solely pre-trained on captioning and text-only datasets, where there is still gap with the Visual Question Answering (VQA) task [20].

2.2.2 Dual-stream models

In dual stream models, the image and text features are first provided to two independent Transformer layers and then later into cross-modal Transformer layers

[5].

ViLBERT:

ViLBERT is a dual-stream model. It processes both visual and textuals inputs in separate streams that interacts through co-attentional transformer layers (Fig: 2.2.3). The ViLBERT model is pre-trained on 3.1 million image-caption pairs from the Conceptual Captions dataset. It accommodates each modality's unique processing requirements while also allowing for interaction across modalities at various representation depths. As per the paper [11], ViLBERT model outperformed the single-stream architecture models.

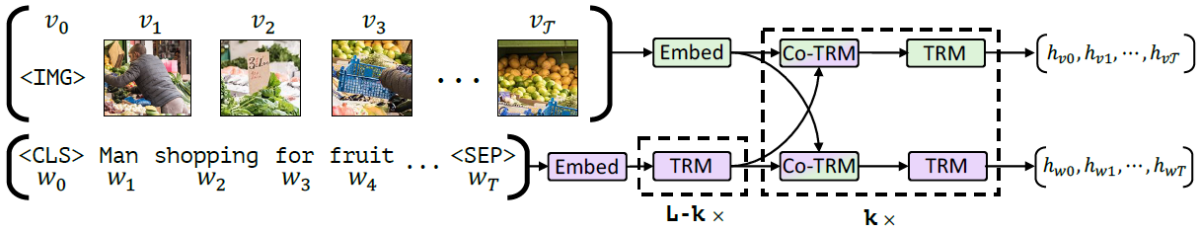


Figure 2.2.3: ViLBERT model [11]

LXMERT:

LXMERT is also a dual-stream model (Fig: 2.2.4). It has three Transformer encoders: an object relationship encoder, a language encoder, and a cross-modality encoder [22].

The main difference between LXMERT and ViLBERT is that the image representations obtained from the object detector are directly sent in ViLBERT whereas LXMERT further processes them through normalization layers.

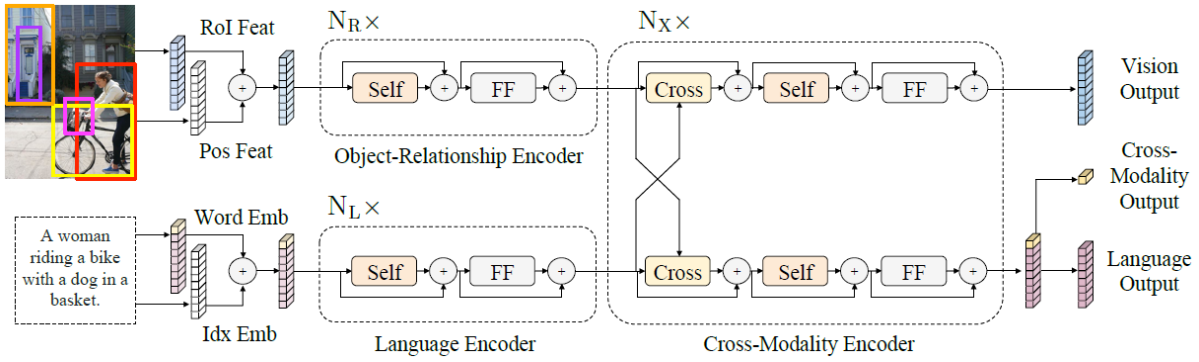


Figure 2.2.4: LXMERT model [22]

2.3 Explainability

Most of the deep learning models are black boxes and the user does not know on what basis the prediction has been made. In mission critical systems and in the field of medicine, it is really important to know what the model has learned. Explainability helps in increasing the trust of the users for the model [17].

There are different explainability techniques and the one that we have used in this thesis project is LIME.

2.3.1 LIME

Individual predictions of black box machine learning models can be explained using local surrogate models, which are interpretable models. Surrogate models are trained to approximate the underlying black box model's predictions. LIME [17] focuses on training local surrogate models to explain individual predictions rather than creating a global surrogate model [13].

The explanation produced by LIME is obtained by the following optimization problem:

$$\xi(x) = \underset{g \in G}{\operatorname{argmin}} L(f, g, \pi_x) + \Omega(g) \tag{2.1}$$

The explanation model for x is the model g (e.g. linear regression model) that minimizes loss L (e.g. mean squared error), which evaluates how close the explanation is to the prediction of the original black-box model f, while keeping the model complexity $\Omega(g)$ low (e.g. prefer fewer features). G stands for the group of interpretable models, such as all linear regression models. The proximity measure π_x specifies the size of the neighborhood that we evaluate for the explanation around instance x.

2.3.2 Other Explainability Techniques

SHapley Additive exPlanations (SHAP): In [12], they have provided strong theoretical proofs of the method. It calculates the feature contribution and uses the concept of baselines. For each prediction, SHAP provides an importance value to each feature. It has two novel components: (1) the identification of a new class of additive feature importance measures, and (2) theoretical results demonstrating that this class has a single solution with a set of desirable properties. It is a black box

explanation method as it does not need access to the underlying model. However, it is computationally complex and requires a lot of resources.

Integrated Gradients (IG): IG [21] takes less time to compute compared to SHAP. This technique works by summing up gradients along a path from a baseline to a specific input. The baseline for text models could be the zero embedding whereas for image networks it could be a black image. IG can be applied to any differential model.

Deep Learning Important Features (DeepLIFT): DeepLIFT [19] recognizes that what we care about is the slope, which describes how y changes as x differs from the baseline and not the gradient. It is fast as it requires one backward pass of the model to calculate feature importance values. Moreover, it is exact unlike integrated gradients, there is no approximation that takes place. However, it redefines how gradients are calculated so we need to dig deep into most deep learning frameworks to implement it. Furthermore, it has no theoretical guarantees compared to SHAP and IG.

2.4 Related Work

The paper "Multimodal Pretraining Unmasked: Unifying the Vision and Language BERTs" [5] has compared five different multimodal models namely ViLBERT [11], LXMERT [22], UNITER [6], VisualBERT [9] and VL-BERT [20]. They have created a controlled environment based on unified mathematical framework called VisiOLinguistic Transformer Architectures (VOLTA) to conduct their experiments and observe the differences. As per their experiments, there is not much difference in single-stream and dual-stream models. The main difference occurs in the result due to the training data and hyper parameters. Moreover, they also observed that the embedding layer plays a crucial role in these massive models [5]. In this thesis project we have used their controlled environment VOLTA to run our experiments in order to have a common ground to observe the differences in the explainability of single-stream and dual-stream models.

Not much work has been done on multimodal explainability. In the paper "Faithful Multimodal Explanation for Visual Question Answering" [24] they provide a novel way to construct a high-performing VQA system that can explain its answers with integrated textual and visual explanations that properly reflect key features of the

underlying reasoning process while capturing the style of understandable human explanations. This solution is specific to the VQA problem.

Chapter 3

Methodology

This chapter describes the dataset used and the experiments that we have performed in order to find out whether single-stream architecture is better or dual-stream architecture for the problem of visual entailment as well as in terms of explainability.

3.1 Dataset

We used the VE problem. In the VE task, a real world image premise P_{image} and a natural language hypothesis H_{text} are given, and the goal is to determine if H_{text} can be concluded given the information provided by P_{image} [25]. Based on the relationship indicated by the (P_{image}, H_{text}), three labels are assigned: .

- *Entailment* holds if there is enough evidence in P_{image} to conclude that H_{text} is true.

- *Contradiction* holds if there is enough evidence in P_{image} to conclude that H_{text} is false.

- Otherwise, the relationship is *neutral*, implying the evidence in P_{image} is insufficient to draw a conclusion about H_{text} [25].

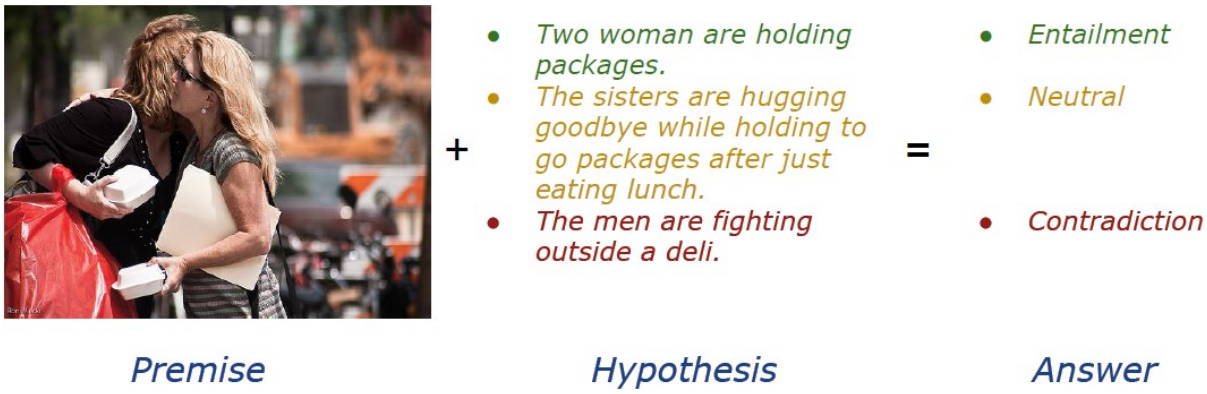


Figure 3.1.1: An example from SNLI-VE dataset [25]

The data that we have used for this thesis project is the SNLI-VE dataset. This dataset is based on Standford Natural Language Inference (SNLI) corpus and Flickr30k dataset. The figure 3.1.1 shows an example of the SNLI-VE dataset.

3.2 Feature Extractor

As mentioned in chapter 2, for the image the features are extracted and then sent as input to the models. To extract the features we use the Bottom-Up and Top-Down attention mechanism. In this approach the bottom-up mechanism (based on Faster R-CNN) proposes image regions, each with an associated feature vector, while the top-down mechanism determines feature weightings [3]. The attention mechanisms driven by non-visual or task-specific context are referred as 'top-down' whereas purely visual feed-forward attention mechanisms are referred as 'bottom-up'. The bottom-up attention is implemented by using Faster R-CNN [16], which represents a natural expression of a bottom-up attention mechanism. The top-down mechanism uses task-specific context to predict an attention distribution over the image regions. The attended feature vector is then computed as a weighted average of image features over all regions [3].

In VOLTA framework, they used the docker image of bottom-up attention [1]. The code base of bottom-up attention is written in python 2 whereas VOLTA framework uses python 3. In VOLTA they had already extracted the features of the images and then used those extracted features for their experiments. They were not extracting the features during run time. To run our explainability task, we had to extract features of the given image during run time and we could not store the features separately.

This was a very difficult task to achieve. We earlier thought of integrating the feature extraction code within our code base but due to python version differences this was not possible. We then created virtual python environment for VOLTA and from that virtual environment we called the bottom-up attention docker in order to extract the features.

3.3 Comparison

In order to find which multimodal architecture is better for visual entailment problem we used the VOLTA framework. The VOLTA github repository [5] provides with the pre-trained models of UNITER, VL-BERT, ViLBERT and LXMERT in the VOLTA controlled environment. In order to train the models for the VE down-stream task we ran the training script for **5** epochs. The hyperparameters were kept the same for all the multimodal models.

Once the training was completed we ran the test script. The performance evaluation metric used was accuracy. Accuracy is the number of correctly predicted data points out of all the data points.

3.4 Multimodal Explainability

In order to run our experiments, we firstly fine-tuned all our models for the SNLI-VE data set and once that was done we used the best trained model of each multimodal model for the experiments.

For explainability we have used the technique of LIME along with using the available LIME python library [2]. LIME follows the following steps for explainability:

1. Perturb the data
2. Calculate distance between perturbations and original instance
3. Make predictions using the model on the perturbed data
4. Fit a simple model to the perturbed data and use the distance as weights
5. Feature weights of this simple model predict the feature weights of the original model

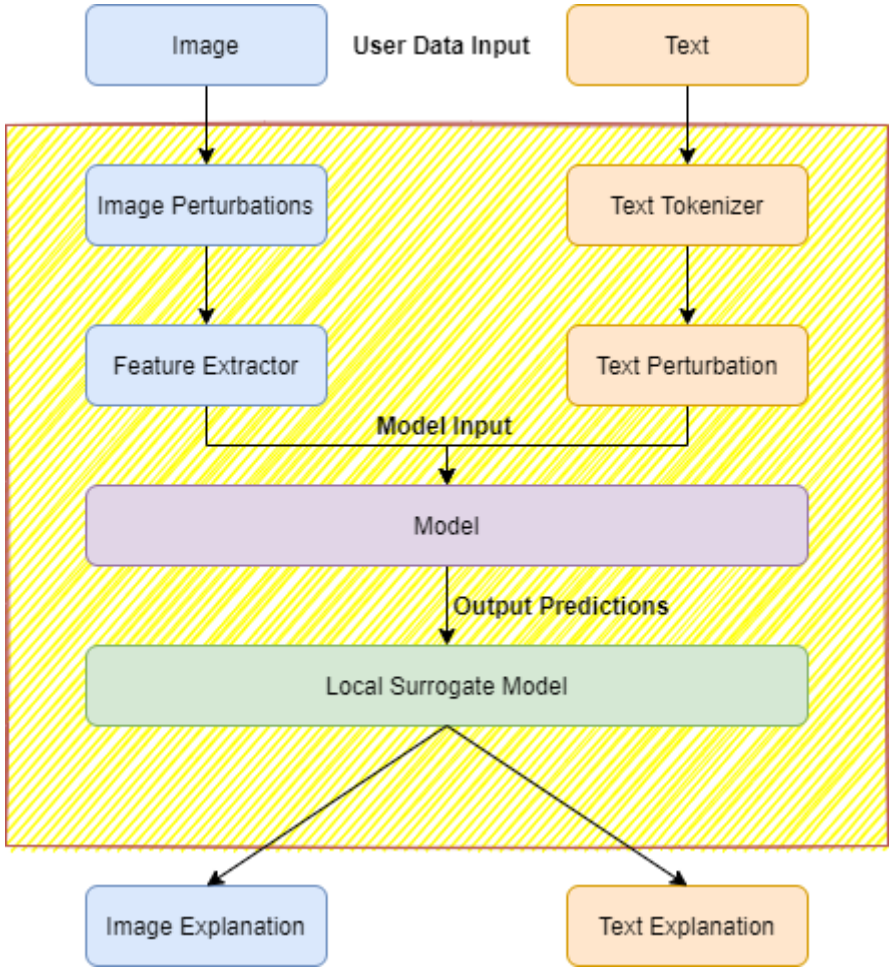


Figure 3.4.1: **Illustration of Multimodal Explainability.** The highlighted yellow box is the custom made multimodal explainer.

There are three ways in which we could perform the explainability tasks in image-text multimodality (Fig: 3.4.1):

1. Perturbing the image and keeping the text fixed
2. Perturbing the text and keeping the image fixed
3. Perturbing both image and text

3.4.1 Perturbing the image and keeping the text fixed

In this case we used the `LimeImageExplainer()` function. We gave the same text as input for all the image perturbations. The `LimeImageExplainer` explains predictions on the image data.

```
classlime.lime_image.LimeImageExplainer(kernel_width=0.25,
```

```
kernel=None, verbose=False, feature_selection='auto',
random_state=None)
```

Below is the description of `LimeImageExplainer` parameters [2]:

- **kernel_width** – kernel width for the exponential kernel.
- **kernel** – similarity kernel that takes euclidean distances and kernel width as input and outputs weights in $(0, 1)$. If None, defaults to an exponential kernel.
- **verbose** – if true, print local prediction values from linear model
- **feature_selection** – feature selection method. Can be 'forward_selection', 'lasso_path', 'none' or 'auto'.
- **random_state** – an integer or numpy. RandomState that will be used to generate random numbers. If None, the random state will be initialized using the internal numpy seed.

To generate explanation for the prediction we used the `explain_instance()` function. This function generates the neighborhood data by randomly perturbing features from the instance. It then learns locally the weighted linear model on the neighborhood data to explain each of the classes in an interpretable way.

```
explain_instance(image, classifier_fn, labels=(1, ),
    hide_color=None, top_labels=5, num_features=100000,
    num_samples=1000, batch_size=10, segmentation_fn=None,
    distance_metric='cosine', model_regressor=None,
    random_seed=None, progress_bar=True)
```

Below is the description of `explain_instance` parameters [2]:

- **image** – 3 dimension RGB image.
- **classifier_fn** – classifier prediction probability function, which takes a numpy array and outputs prediction probabilities.
- **labels** – iterable with labels to be explained.
- **top_labels** – if not None, ignore labels and produce explanations for the K labels with highest prediction probabilities, where K is this parameter.
- **num_features** – maximum number of features present in explanation

- **num_samples** – size of the neighborhood to learn the linear model
- **batch_size** – The number of images that are sent together in one batch.
- **distance_metric** – the distance metric to use for weights.
- **model_regressor** – sklearn regressor to use in explanation. Defaults Ridge regression in LimeBase.
- **segmentation_fn** – Segmentation Algorithm, wrapped skimage function (segmentation)
- **random_seed** – integer used as random seed for the segmentation algorithm. If None, a random integer, between 0 and 1000, will be generated using the internal random number generator.
- **progress_bar** – if True, show tqdm progress bar.

In the experiment we first initialized the `LimeImageExplainer` function and then called the `explain_instance` function by sending the image as an array input. The other parameters that were sent were the classification function, number of samples, batch size and the top label.

3.4.2 Perturbing the text and keeping the image fixed

In this case we used the `LimeTextExplainer()` function. We give the same image features as input for all the text perturbations. The `LimeTextExplainer` explains the text classifiers.

```
classlime.lime_text.LimeTextExplainer(kernel_width=25,
    kernel=None, verbose=False, class_names=None,
    feature_selection='auto', split_expression='\W+',
    bow=True, mask_string=None, random_state=None,
    char_level=False)
```

Below is the description of `LimeTextExplainer` parameters [2]:

- **kernel_width** – kernel width for the exponential kernel.
- **kernel** – similarity kernel that takes euclidean distances and kernel width as input and outputs weights in $(0, 1)$. If None, defaults to an exponential kernel.

- **verbose** – if true, print local prediction values from linear model
- **class_names** – list of class names, ordered according to whatever the classifier is using.
- **feature_selection** – feature selection method. Can be 'forward_selection', 'lasso_path', 'none' or 'auto'.
- **split_expression** – Regex string or callable.
- **bow** – if True (bag of words), will perturb input data by removing all occurrences of individual words or characters. Explanations will be in terms of these words. Otherwise, will explain in terms of word-positions, so that a word may be important the first time it appears and unimportant the second. Only set to false if the classifier uses word order in some way (bigrams, etc), or if you set char_level=True.
- **mask_string** – String used to mask tokens or characters if bow=False if None, will be 'UNKWORDZ' if char_level=False, chr(0) otherwise.
- **random_state** – an integer or numpy. Random State that will be used to generate random numbers. If None, the random state will be initialized using the internal numpy seed.
- **char_level** – an boolean identifying that we treat each character as an independent occurrence in the string.

To generate explanation for the prediction we again used the `explain_instance()` function. This function works similar to the previous one. The only difference is that the previous one was for image explanation whereas this function is for the text explanations.

Below is the description of `explain_instance` parameters [2]:

- **text_instance** – raw text string to be explained.
- **classifier_fn** – classifier prediction probability function, which takes a list of d strings and outputs a (d, k) numpy array with prediction probabilities, where k is the number of classes.
- **labels** – iterable with labels to be explained.

- **top_labels** – if not None, ignore labels and produce explanations for the K labels with highest prediction probabilities, where K is this parameter.
- **num_features** – maximum number of features present in explanation
- **num_samples** – size of the neighborhood to learn the linear model
- **distance_metric** – the distance metric to use for sample weighting, defaults to cosine similarity
- **model_regressor** – sklearn regressor to use in explanation. Defaults Ridge regression in LimeBase.

In this experiment we first initialized the `LimeTextExplainer` with the class names namely 'contradiction', 'neutral', 'entailment' and the bag of words set as False. We then called the `explain_instance` function by sending the text, the classification function, number of features and number of samples.

3.4.3 Perturbing both image and text

This was one of the most trickiest experiment. We had to create our own explainer function in order to perturb both image and text.

For this experiment we had computational limitations as feature extractor needed more GPU (at least 12GB) where as we only had 8GB GPU. Due to this limitation we could only run the experiment for 10 number of samples.

We implemented two separate functions for image perturbations and text perturbations. The image perturbation function creates multiple images by segmenting the image into 'super pixels' and then turning those super pixels on or off. Similarly, the text perturbation function works by randomly removing words from the original sentence. While perturbing both the functions saves the distances (or weights) from the original image and text respectively. These functions then return the weights and the perturbations.

We then implemented our main multimodal explainer function which calls the image perturbation and text perturbation function. The multimodal explainer follows the LIME technique. It sums the weights of image and text and also concatenates the perturbations of both image and text.

After this step, we need to get the predictions from the model and for that we

need to send the image features. For this purpose we call the feature extraction function. Once all this is complete we call the model prediction function and save those predictions.

We then create our local surrogate model by using Linear Regression. We give the concatenated perturbations as X, the model predictions as y and the combined weights as $sample\_weight$. The coefficient of this simple model is the feature importance. The starting coefficients are for the image and the later ones are for the text as while concatenating the perturbations we gave the image perturbations before the text.

Below we have provided a simplified code of our model explainer function:

```
def multimodal_explainer(image, sentence, answer, num_samples):
    image_perturbations(image, num_samples)
    text_perturbations(sentence, answer, num_samples)

    # FEATURE EXTRACTION
    feature_extraction()

    # LINEAR MODEL
    #Combined weights
    weights = weightsImage + weightsText

    #Combined perturbations
    perturbationsC = np.concatenate((imagePerturbs,textPerturbs))


    #Get PREDICTIONS from model
    predictions = predict_fn()

    #Surrogate Model
    simpler_model = LinearRegression()
    simpler_model.fit(X=perturbationsC, y=predictions, weights)
    coeff = simpler_model.coef_[0]

    #Disentangling Coefficients
```

```
coeffImage = coeff[:num_superpixels]
coeffText = coeff[num_superpixels:]

# IMAGE EXPLANATION
#Use coefficients from linear model to extract top features
#Show only the superpixels corresponding to the top features

# TEXT EXPLANATION
#Use coefficients from linear model to find word importance
```

Chapter 4

Result

In this chapter we present the results of the experiments that we have performed. Section one displays the results of the comparison experiments and section two shows the results of the different explainability experiments.

4.1 Comparison

When we trained both the single-stream and dual-stream multimodal models we observed that UNITER had the best testing accuracy of 77.230. Rest of the models also had similar accuracy as shown in the table 4.1.1.

Model	Accuracy	Loss
UNITER	**77.230**	0.354
VL-BERT	76.828	0.358
ViLBERT	76.605	0.406
LXMERT	76.482	0.389

Table 4.1.1: Accuracy and Loss of multimodal models on Visual Entailment task

From the table 4.1.1 we can see that the single-stream models UNITER and VL-BERT perform better on the VE task compared to the dual-stream models ViLBERT and LXMERT.

4.2 Multimodal Explainability

As mentioned previously, we ran the explainability experiments for only 10 number of samples due to which the explainability results were not stable enough. For the experiments where we perturb only one modality and keep the other fixed we used the image 4.2.1. The hypothesis text given was "*The man painted the picture of a boy that happened to be his grandson.*" and the expected answer was "*neutral*".

Figure 4.2.1: Original Image

4.2.1 Perturbing the image and keeping the text fixed

The figure 4.2.2 displays the explainability result of UNITER (Fig: 4.2.2a), VL-BERT (Fig: 4.2.2b), ViLBERT (Fig: 4.2.2c) and LXMERT (Fig: 4.2.2d) for the experiment where we perturb the image but keep the text fixed. We can see that only UNITER predicted neutral correctly however, the explainability shows all the highlighted segments as red meaning negative influence on the prediction.

4.2.2 Perturbing the text and keeping the image fixed

The figure 4.2.3 displays the explainability result of UNITER (Fig: 4.2.3a), VL-BERT (Fig: 4.2.3b), ViLBERT (Fig: 4.2.3c) and LXMERT (Fig: 4.2.3d) for the experiment where we perturb the text but keep the image fixed. We can see that UNITER and LXMERT both predict correctly. Moreover, they have similar explanation for their prediction.

4.2.3 Perturbing both image and text

For the experiment where we perturb both image and text we selected 5 images with all the three label outputs. This report only shows the result for the image 4.2.1.

In the figures 4.2.4, 4.2.5, 4.2.6 and 4.2.7 the visible part of the images are the important feature segments and for the text the green bar shows positive impact and the red bar shows the negative impact of the word on the prediction. The predicted output by the model can be seen on top of the text bar chart. The first row of result shows the explainability result where neutral is the true label, the second row shows for entailment and the last row shows for the contradiction.

We can see that UNITER and ViLBERT predicted correctly in all the three cases (refer Fig: 4.2.4 and 4.2.6).

(a) Model: UNITER
Prediction: neutral

(b) Model: VL-BERT
Prediction: entailment

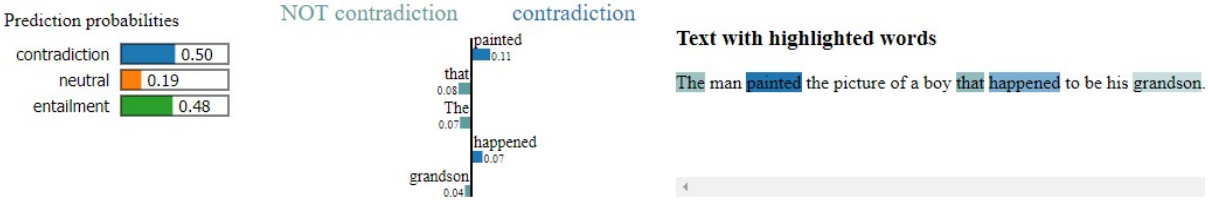
(c) Model: ViLBERT
Prediction: entailment

(d) Model: LXMERT
Prediction: entailment

Figure 4.2.2: **Explainability of the models where the image is perturbed and the text remains fixed.** Red represents the image segments that had negative effect on the prediction whereas Green represents the positive effect

(a) Model: UNITER
Prediction: neutral

(b) Model: VL-BERT
Prediction: entailment



(c) Model: ViLBERT
Prediction: contradiction

(d) Model: LXMERT
Prediction: neutral

Figure 4.2.3: **Explainability of the models where the text is perturbed and the image remains fixed.** Strongly color highlighted text represents the words which had the most impact on the prediction.

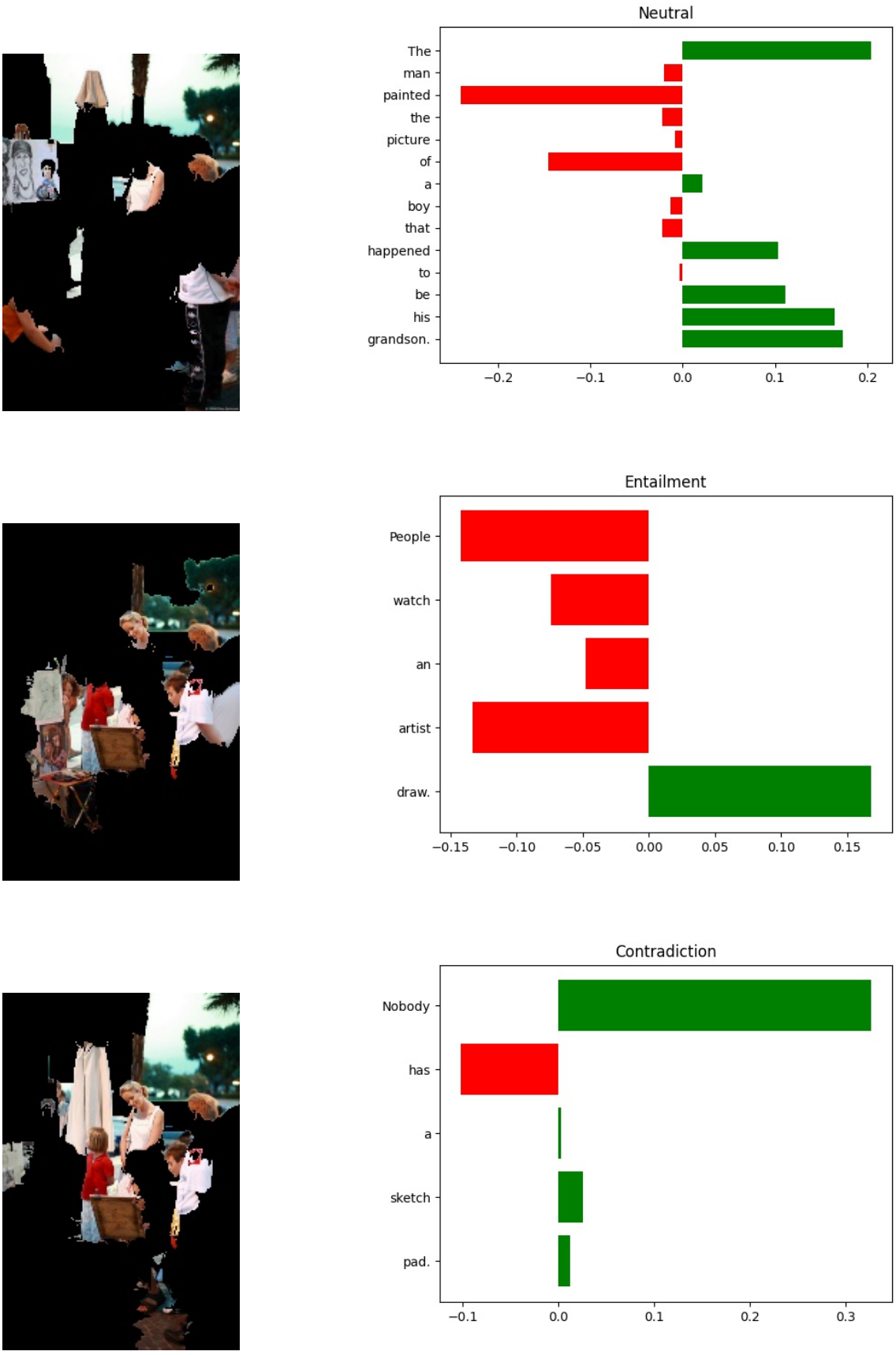


Figure 4.2.4: **Multimodal Explainability of UNITER.**

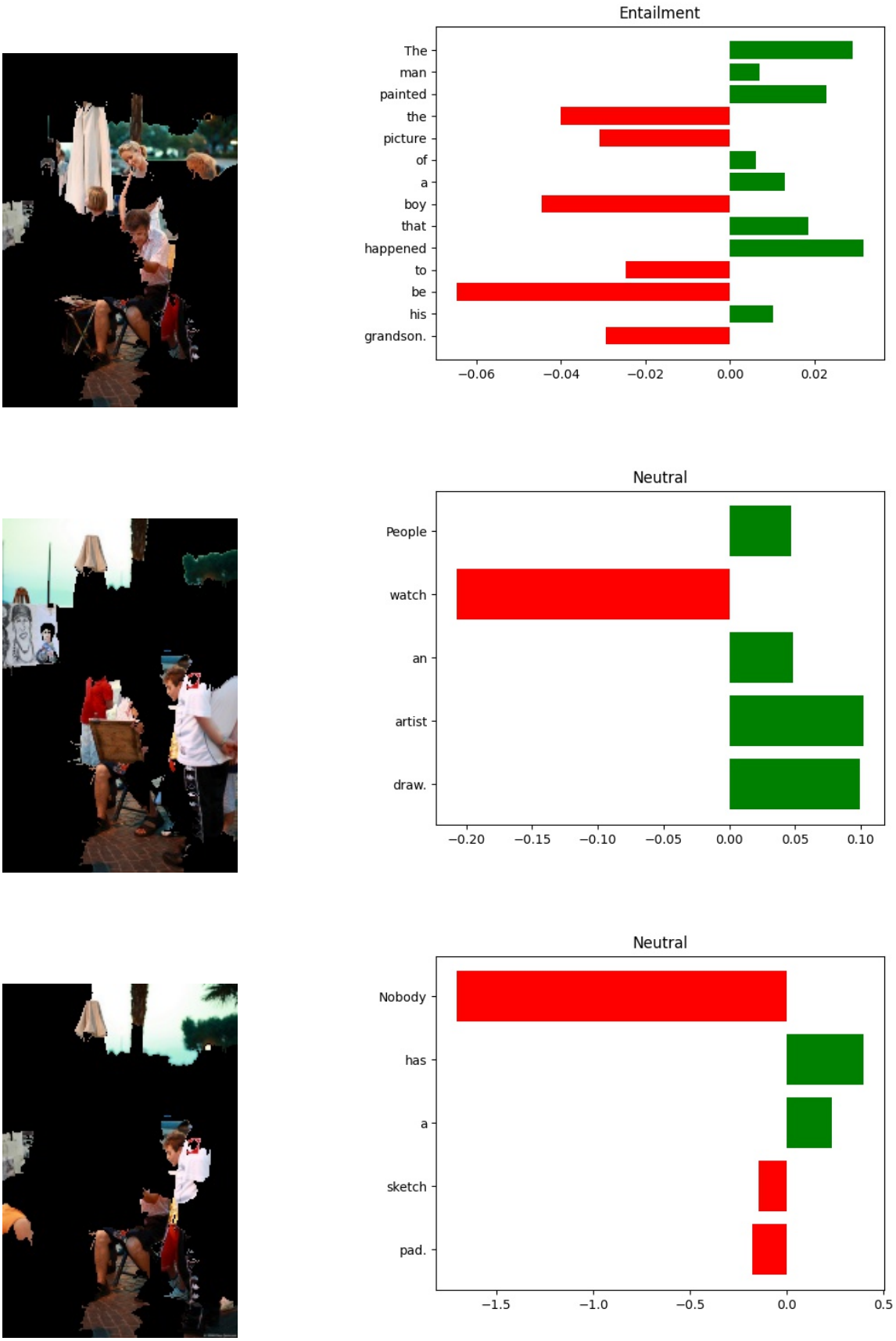


Figure 4.2.5: **Multimodal Explainability of VL-BERT.**

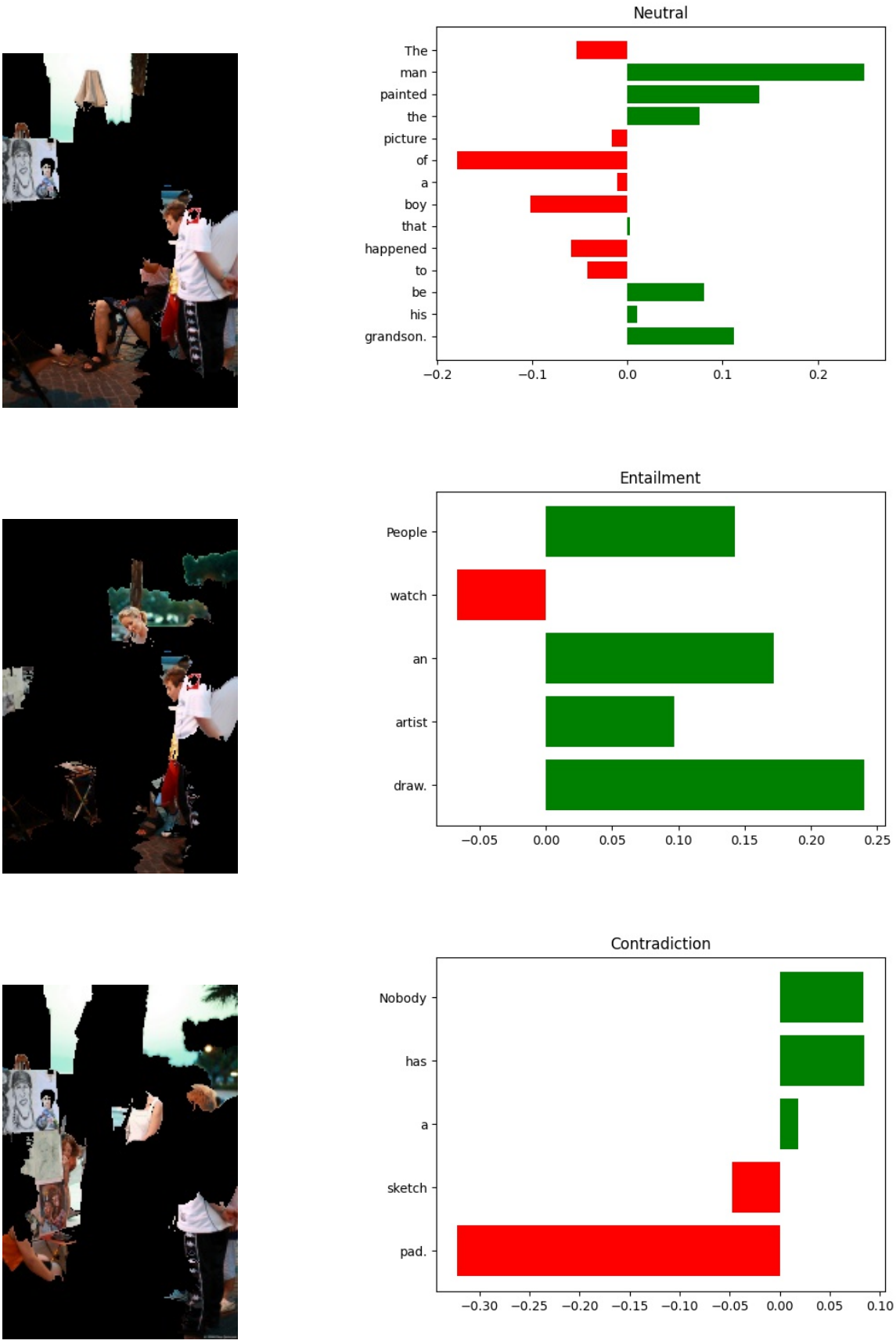


Figure 4.2.6: **Multimodal Explainability of ViLBERT.**

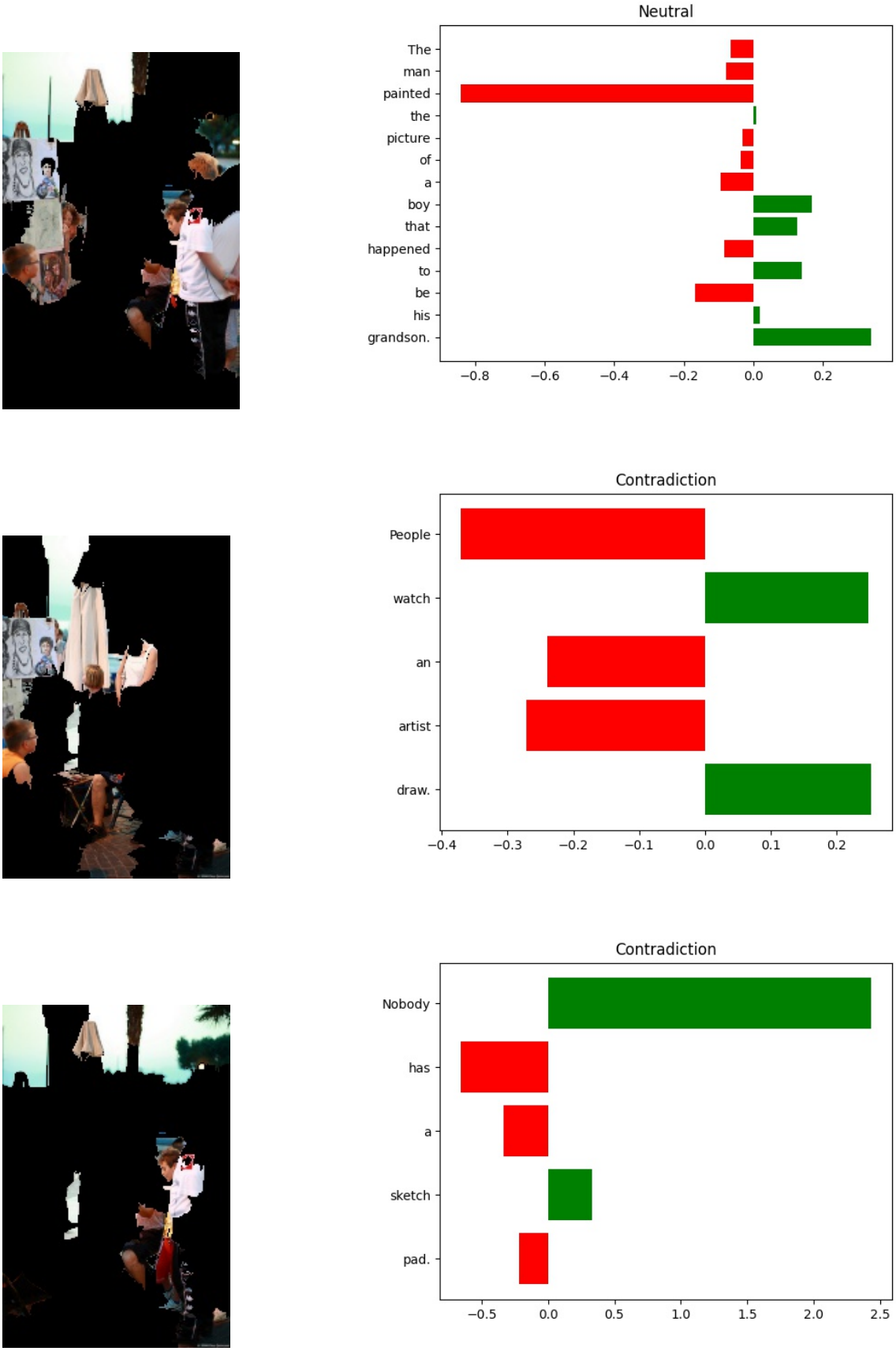


Figure 4.2.7: **Multimodal Explainability of LXMERT.**

Chapter 5

Conclusions

In this thesis project we firstly trained the multimodal models for the visual entailment problem and then we performed experiments for the three possible permutations for explainability. We implemented our own multimodal explainer by following the LIME technique.

We observed that multimodal explainability can be performed by adding the weights and perturbations of both the modalities. The results of our experiments were random due to the computational limitations, however, we can still conclude based on our observations that even the explainability of both single-stream and dual-stream model is similar. Therefore, before selecting a model to perform prediction the model which has the best accuracy could be selected. It does not matter whether it is a single-stream model or a dual-stream model.

5.1 Future Work

The down-stream task for all the models could be run for more number of epochs. Moreover, we could also apply different explainability techniques like SHAP, Integrated gradients etc. As we faced technical limitations there are a lot of things that could be performed as future work if these limitations are overcome. We can run the experiment for more number of samples (at least 1000) to further investigate which model architecture is better with respect to explainability. Furthermore, we can also try changing the kernel width in the LIME explainability technique.

Bibliography

[1] URL: https://hub.docker.com/r/airsplay/bottom-up-attention (visited on 05/28/2021).

[2] URL: https://github.com/marcotcr/lime (visited on 07/21/2021).

[3] Anderson, Peter, He, Xiaodong, Buehler, Chris, Teney, Damien, Johnson, Mark, Gould, Stephen, and Zhang, Lei. "Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering". In: *arXiv:1707.07998 [cs]* (Mar. 14, 2018). arXiv: 1707.07998. URL: http://arxiv.org/abs/1707.07998 (visited on 03/05/2021).

[4] Baltrušaitis,
Tadas, Ahuja, Chaitanya, and Morency, Louis-Philippe. "Multimodal Machine Learning: A Survey and Taxonomy". In: *arXiv:1705.09406 [cs]* (Aug. 1, 2017). arXiv: 1705.09406. URL: http://arxiv.org/abs/1705.09406 (visited on 02/05/2021).

[5] Bugliarello, Emanuele, Cotterell, Ryan, Okazaki, Naoaki, and Elliott, Desmond. "Multimodal Pretraining Unmasked: Unifying the Vision and Language BERTs". In: *arXiv:2011.15124 [cs]* (Nov. 30, 2020). arXiv: 2011.15124. URL: http://arxiv.org/abs/2011.15124 (visited on 03/05/2021).

[6] Chen, Yen-Chun, Li, Linjie, Yu, Licheng, Kholy, Ahmed El, Ahmed, Faisal, Gan, Zhe, Cheng, Yu, and Liu, Jingjing. "UNITER: UNiversal Image-TExt Representation Learning". In: *arXiv:1909.11740 [cs]* (July 17, 2020). arXiv: 1909.11740. URL: http://arxiv.org/abs/1909.11740 (visited on 02/16/2021).

[7] Devlin, Jacob, Chang, Ming-Wei, Lee, Kenton, and Toutanova, Kristina. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding".

In: *arXiv:1810.04805 [cs]* (May 24, 2019). arXiv: 1810.04805. URL: http://arxiv.org/abs/1810.04805 (visited on 04/23/2021).

[8] Krishna, Ranjay, Zhu, Yuke, Groth, Oliver, Johnson, Justin, Hata, Kenji, Kravitz, Joshua, Chen, Stephanie, Kalantidis, Yannis, Li, Li-Jia, Shamma, David A., Bernstein, Michael S., and Li, Fei-Fei. "Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations". In: *arXiv:1602.07332 [cs]* (Feb. 23, 2016). arXiv: 1602.07332. URL: http://arxiv.org/abs/1602.07332 (visited on 09/09/2021).

[9] Li, Liunian Harold, Yatskar, Mark, Yin, Da, Hsieh, Cho-Jui, and Chang, Kai-Wei. "VisualBERT: A Simple and Performant Baseline for Vision and Language". In: *arXiv:1908.03557 [cs]* (Aug. 9, 2019). arXiv: 1908.03557. URL: http://arxiv.org/abs/1908.03557 (visited on 03/22/2021).

[10] Lin, Tsung-Yi, Maire, Michael, Belongie, Serge, Bourdev, Lubomir, Girshick, Ross, Hays, James, Perona, Pietro, Ramanan, Deva, Zitnick, C. Lawrence, and Dollár, Piotr. "Microsoft COCO: Common Objects in Context". In: *arXiv:1405.0312 [cs]* (Feb. 20, 2015). arXiv: 1405.0312. URL: http://arxiv.org/abs/1405.0312 (visited on 09/09/2021).

[11] Lu, Jiasen, Batra, Dhruv, Parikh, Devi, and Lee, Stefan. "ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks". In: *arXiv:1908.02265 [cs]* (Aug. 6, 2019). arXiv: 1908.02265. URL: http://arxiv.org/abs/1908.02265 (visited on 02/20/2021).

[12] Lundberg, Scott and Lee, Su-In. "A Unified Approach to Interpreting Model Predictions". In: *arXiv:1705.07874 [cs, stat]* (Nov. 24, 2017). arXiv: 1705.07874. URL: http://arxiv.org/abs/1705.07874 (visited on 03/10/2021).

[13] Molnar, Christoph. *Interpretable Machine Learning. A Guide for Making Black Box Models Explainable*. https://christophm.github.io/interpretable-ml-book/. 2019.

[14] Ordonez, Vicente, Kulkarni, Girish, and Berg, Tamara. "Im2Text: Describing Images
Using 1 Million Captioned Photographs". In: *Advances in Neural Information Processing Systems*. Vol. 24. Curran Associates, Inc., 2011. URL: https://papers.nips.cc/paper/2011/hash/5dd9db5e033da9c6fb5ba83c7a7ebea9-Abstract.html (visited on 09/09/2021).

[15] Ravichandiran, Sudharsan. *Getting Started with Google BERT*. Packt. ISBN: 9781838821593.

[16] Ren, Shaoqing, He, Kaiming, Girshick, Ross, and Sun, Jian. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *arXiv:1506.01497 [cs]* (Jan. 6, 2016). arXiv: 1506.01497. URL: http://arxiv.org/abs/1506.01497 (visited on 08/27/2021).

[17] Ribeiro,
Marco Tulio, Singh, Sameer, and Guestrin, Carlos. ""Why Should I Trust You?": Explaining the Predictions of Any Classifier". In: *arXiv:1602.04938 [cs, stat]* (Aug. 9, 2016). arXiv: 1602.04938. URL: http://arxiv.org/abs/1602.04938 (visited on 03/10/2021).

[18] Sharma, Piyush, Ding, Nan, Goodman, Sebastian, and Soricut, Radu. "Conceptual Captions: A Cleaned, Hypernymed, Image Alt-text Dataset For Automatic Image Captioning". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. ACL 2018. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 2556–2565. DOI: 10.18653/v1/P18-1238. URL: https://aclanthology.org/P18-1238 (visited on 09/09/2021).

[19] Shrikumar, Avanti, Greenside, Peyton, and Kundaje, Anshul. "Learning Important Features Through Propagating Activation Differences". In: *arXiv:1704.02685 [cs]* (Oct. 12, 2019). arXiv: 1704.02685. URL: http://arxiv.org/abs/1704.02685 (visited on 02/04/2021).

[20] Su, Weijie, Zhu, Xizhou, Cao, Yue, Li, Bin, Lu, Lewei, Wei, Furu, and Dai, Jifeng. *VL-BERT: Pre-training of Generic Visual-Linguistic Representations*. 2020. arXiv: 1908.08530 [cs.CV].

[21] Sundararajan, Mukund, Taly, Ankur, and Yan, Qiqi. "Axiomatic Attribution for Deep Networks". In: *arXiv:1703.01365 [cs]* (June 12, 2017). arXiv: 1703.01365. URL: http://arxiv.org/abs/1703.01365 (visited on 03/18/2021).

[22] Tan, Hao and Bansal, Mohit. "LXMERT: Learning Cross-Modality Encoder Representations from Transformers". In: *arXiv:1908.07490 [cs]* (Dec. 3, 2019). arXiv: 1908.07490. URL: http://arxiv.org/abs/1908.07490 (visited on 03/22/2021).

[23] Vaswani, Ashish, Shazeer, Noam, Parmar, Niki, Uszkoreit, Jakob, Jones, Llion, Gomez, Aidan N., Kaiser, Lukasz, and Polosukhin, Illia. “Attention Is All You Need”. In: *arXiv:1706.03762 [cs]* (Dec. 5, 2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762 (visited on 04/23/2021).

[24] Wu, Jialin and Mooney, Raymond. “Faithful Multimodal Explanation for Visual Question Answering”. In: *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 103–112. DOI: 10.18653/v1/W19-4812. URL: https://www.aclweb.org/anthology/W19-4812 (visited on 02/15/2021).

[25] Xie, Ning, Lai, Farley, Doran, Derek, and Kadav, Asim. “Visual Entailment: A Novel Task for Fine-Grained Image Understanding”. In: *arXiv:1901.06706 [cs]* (Jan. 20, 2019). arXiv: 1901.06706. URL: http://arxiv.org/abs/1901.06706 (visited on 03/03/2021).