Robustness of Multi-Modal Foundation Models

Dem Institut für Mathematik
der Technische Universität Berlin
vorgelegte

Masterarbeit

von
Kartikeya Chitranshi
Matrikelnummer 475703

Gutachter:

Prof. Dr. Sebastian Pokutta
Prof. Dr. Thorsten Koch

Betreuer:

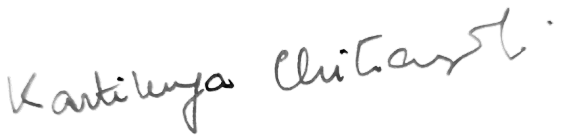
Shpresim Sadiku

Berlin, den 3 April 2025

Eidesstattliche Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit eigenständig ohne Hilfe Dritter und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe. Alle Stellen die den benutzten Quellen und Hilfsmitteln unverändert oder sinngemäß entnommen sind, habe ich als solche kenntlich gemacht. Sofern generische KI-Tools verwendet wurden, habe ich Produktnamen, Hersteller, die jeweils verwendete Softwareversion und die jeweiligen Einsatzzwecke (z.B. sprachliche Überprüfung und Verbesserung der Texte, systematische Recherche) benannt. Ich verantworte die Auswahl, die Übernahme und sämtliche Ergebnisse des von mir verwendeten KI-generierten Outputs vollumfänglich selbst. Die Satzung zur Sicherung guter wissenschaftlicher Praxis an der TU Berlin vom 8. März 2017. `https://www.static.tu.berlin/fileadmin/www/10000060/FSC/Promotion___Habilitation/Dokumente/Grundsaetze_gute_wissenschaftliche_Praxis_2017.pdf` habe ich zur Kenntnis genommen. Ich erkläre weiterhin, dass ich die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt habe.

Berlin, den 3 April 2025



Kartikeya Chitranshi

Es wurden Grammarly Version 8.927.0 zur sprachlichen Überprüfung der Texte verwendet.

Zusammenfassung

Multimodale Basismodelle haben in letzter Zeit viel Aufmerksamkeit auf sich gezogen, insbesondere die Vision-Language-Modelle, die Vision- und Sprachmodelle kombinieren. Sie werden für Aufgaben wie maschinelle Übersetzung, Bildbeschriftung, Video-Reasoning und Bild-zu-Text-Retrieval eingesetzt. Allerdings wurde auch ihre Anfälligkeit für sogenannte adversarial Attacks, insbesondere die Auswirkungen nicht wahrnehmbarer Störungen in Bildern, die den von diesen Modellen generierten Text stark verändern können, genau untersucht. Frühere Arbeiten untersuchten in erster Linie nicht-dünnbesetzte Angriffe, die Störungen über mehrere Pixel erzeugen. In dieser Arbeit untersuchen wir die Verwendung von spärlichen gegnerischen Angriffen auf diese Modelle für Aufgaben wie Bildbeschriftung und Beantwortung von visuellen Fragen. Außerdem vergleichen wir vortrainierte CLIP-Modelle, die auf Bildern, die durch gegnerische Angriffe erzeugt wurden, feinabgestimmt wurden, mit kontrafaktischen Modellen für Aufgaben wie Bildklassifizierung und Bild-Text-Retrieval. Darüber hinaus stellen wir Details zur Implementierung der in dieser Arbeit verwendeten adversarial Attacks vor.

Abstract

Multi-modal foundation models have recently attracted significant attention, especially the vision-language models that combine vision and language models. They are utilized for tasks such as machine translation, image captioning, video reasoning, and image-to-text retrieval. However, their susceptibility to adversarial attacks has also been studied, particularly the impact of imperceptible perturbations on images that can severely alter the text generated by these models. Previous works primarily explored non-sparse adversarial attacks, that introduce perturbations across multiple pixels. In this work, we investigate the use of sparse adversarial attacks on these models for tasks such as image captioning and visual question answering, and we further compare pre-trained CLIP models fine-tuned on images generated by adversarial attacks as compared to counterfactuals on tasks such as image classification and image-text retrieval. Additionally, we provide the details regarding the implementation of the attacks used in this work.

Contents

1 Introduction

Multi-modal foundation models, especially vision language models, have recently garnered significant attention. By merging the capabilities of large language models with vision encoders, these models have demonstrated strong performance across various cross-modal tasks, such as image captioning [Alayrac et al., 2022], visual question answering [Awadalla et al., 2023], image-text retrieval, and image classification [Radford et al., 2021].

Despite their success, multi-modal models face several challenges. When deployed in real settings, they can encounter adversarial threats. Malicious users may exploit vulnerabilities to jailbreak these models, as highlighted in concurrent research [Carlini et al., 2024; Qi et al., 2023; Schlarmann and Hein, 2023]. Moreover, these models can be manipulated into producing malicious and, at times, harmful output. An instance of this was seen in [Schlarmann and Hein, 2023], where adversarial images were created to generate the desired, harmful adversarial output. However, the adversarial perturbations were non-sparse, and as a result, manipulated an excess amount of pixels. Therefore, the utilization of sparse adversarial attacks is warranted.

Moreover, counterfactuals have also been explored for vision language models [Le et al., 2024; Howard et al., 2023] that introduce meaningful changes to both modalities - images and texts. In the **SocialCounterfactuals** dataset [Howard et al., 2023], counterfactuals are created via Stable Diffusion [Rombach et al., 2022] with cross-attention to examine the social biases of these models, meanwhile in the case of **Diffusion Visual Counterfactuals Explanations** [Augustin et al., 2022], diffusion processes [Song et al., 2020a,b; Ho et al., 2020] are used to target class-relevant features in an image for image classifiers. However, the comparison of the performance of pre-trained vision language models fine-tuned on counterfactuals as opposed to adversarial attacks is yet to be examined, to the best of our knowledge. In this work, we will explain and utilize the COCO-

Counterfactuals [Le et al., 2024] dataset for this purpose.

Our contributions can be summarized as follows:

1. We use a sparse adversarial attack to compare with a non-sparse adversarial attack on a multi-modal model for the image captioning and visual question answering tasks.

2. Furthermore, we compare the performance of pre-trained models fine-tuned on adversarial samples as opposed to counterfactuals in image classification and image-text retrieval tasks.

1.1. Organization

Chapter 2 introduces notations and concepts such as natural language processing, Transformer architecture [Vaswani et al., 2017], and many more. Chapter 3 looks into the SAIF [Imtiaz et al., 2022] and APGD [Croce and Hein, 2020] adversarial attacks, and COCO counterfactuals [Le et al., 2024]. Metrics, datasets, tasks, setup, and results for the conducted experiments are contained in Chapter 4.

2 Background

In this chapter, we introduce notations and concepts used throughout this work.

2.1. Notation

We denote a n-dimensional vector $\mathbf{x} \in \mathbb{R}^n$ as

$$\mathbf{x} := \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \tag{2.1}$$

Let $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$. Then, we denote the dot product between them $\mathbf{x} \cdot \mathbf{y}$ as

$$\mathbf{x} \cdot \mathbf{y} := \sum_{i=1}^{n} x_i y_i \tag{2.2}$$

and the component-wise product $\mathbf{x} \odot \mathbf{y}$ as

$$\mathbf{x} \odot \mathbf{y} := \begin{bmatrix} x_1 y_1 \\ x_2 y_2 \\ \vdots \\ x_n y_n \end{bmatrix} \tag{2.3}$$

We denote p-norms, $p \geq 1$, in $\mathbb{R}^n$ by $\|\cdot\|_p$. Given a vector $\mathbf{x} \in \mathbb{R}^n$, we write it as

$$\|\mathbf{x}\|_p := \left(\sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}} \tag{2.4}$$

$$\|\mathbf{x}\|_1 := \sum_{i=1}^{n} |x_i| \ \text{(Manhattan or 1-norm)} \tag{2.5}$$

$$\|\mathbf{x}\|_2 := \sqrt{\sum_{i=1}^{n} |x_i|^2} \ \text{(Euclidean or 2-norm)} \tag{2.6}$$

$$\|\mathbf{x}\|_\infty := \max_i |x_i| \ \text{(Chebyshev or } \infty\text{-norm)} \tag{2.7}$$

Let $f : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}$ be a differentiable function, $\mathbf{x} \in \mathbb{R}^m$, and $\mathbf{y} \in \mathbb{R}^n$. We denote the differentiation of f with respect to a variable via subscipt, that is, the gradients of f with respect to $\mathbf{x}$ and $\mathbf{y}$ are

$$\nabla_{\mathbf{x}} f(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^m \text{ and } \nabla_{\mathbf{y}} f(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^n \tag{2.8}$$

We let $[n] := \{1, 2, \cdots, n\}$ for $n \in \mathbb{N}$. For a matrix $A \in \mathbb{R}^{m \times n}$, we denote $A^\top$ as the transpose of A.

$$A := \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}, \quad A^\top := \begin{bmatrix} a_{11} & a_{21} & \cdots & a_{m1} \\ a_{12} & a_{22} & \cdots & a_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{mn} \end{bmatrix}. \tag{2.9}$$

For another matrix $B \in \mathbb{R}^{m \times n}$, we overload the $\odot$ symbol for the Hadamard product.

$$A \odot B := \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} & \cdots & a_{1n}b_{1n} \\ a_{21}b_{21} & a_{22}b_{22} & \cdots & a_{2n}b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}b_{m1} & a_{m2}b_{m2} & \cdots & a_{mn}b_{mn} \end{bmatrix} \tag{2.10}$$

We denote an **element** of the matrix A on the ith row and the jth column as a_{ij}, and we denote an entire row $i \in [m]$ as

$$A_{i,:} := [a_{i1}, a_{i2}, \cdots, a_{in}] \tag{2.11}$$

Furthermore, we denote **row-level** norm of A as $\texttt{RLN}(A) : \mathbb{R}^{m \times n} \to \mathbb{R}^m$. For the row $i \in [m]$ of the matrix A, we have

$$\|A_{i,:}\|_2 := \sqrt{\sum_{j=1}^{n} |A_{ij}|^2} \tag{2.12}$$

We then compute $\texttt{RLN}(A)$ as

$$\texttt{RLN}(A) := \begin{bmatrix} \|A_{1,:}\|_2 \\ \|A_{2,:}\|_2 \\ \vdots \\ \|A_{m,:}\|_2 \end{bmatrix} \tag{2.13}$$

We denote the indicator function as $\mathbb{1}_{\texttt{A}}$, where given a predicate $\texttt{A}$, we define

$$\mathbb{1}_{\texttt{A}} := \begin{cases} 1, & \text{if } \texttt{A} \text{ is true} \\ 0, & \text{else} \end{cases} \tag{2.14}$$

Let $\mathbf{x} \in \mathbb{R}^n$. We denote the **signum** function as $\text{sign} : \mathbb{R}^n \to \mathbb{R}^n$, and formulate it as

$$\text{sign}(\mathbf{x}) := \begin{cases} 1, & \text{if } x_i > 0 \\ 0, & \text{else if } x_i = 0 \\ -1, & \text{else} \end{cases} \qquad \forall i \in [n] \tag{2.15}$$

2.2. Deep Feed Forward Networks

Before venturing into complex architectures of multi-modal models, we first introduce the basics of a deep neural network. To be more precise, we start by introducing artificial neurons. An **artificial neuron** is a non-linear, differentiable function that can be expressed in the form of:

$$\hat{x}_j = \sigma(\sum_{i=i}^{n} x_i \mathbf{w}_{ij} + b_j) = \sigma(\mathbf{w}_j^\top x + b_j) \tag{2.16}$$

transforming an input vector $\mathbf{x} = (x_1, \cdots, x_n)$ into an output $\hat{x}_j$, where $\mathbf{w}_{ij}$ represents the weights, b_j is the bias and σ is the differentiable activation function. A **deep neural network** (DNN) combines numerous artificial neurons, typically arranged in a layered structure, to represent a more complex function. All layers, except the input and output layers, are referred to as **hidden layers**. Here, we focus exclusively on feedforward DNNs, where the flow of information is unidirectional, progressing from the input layer, through the hidden layers, and

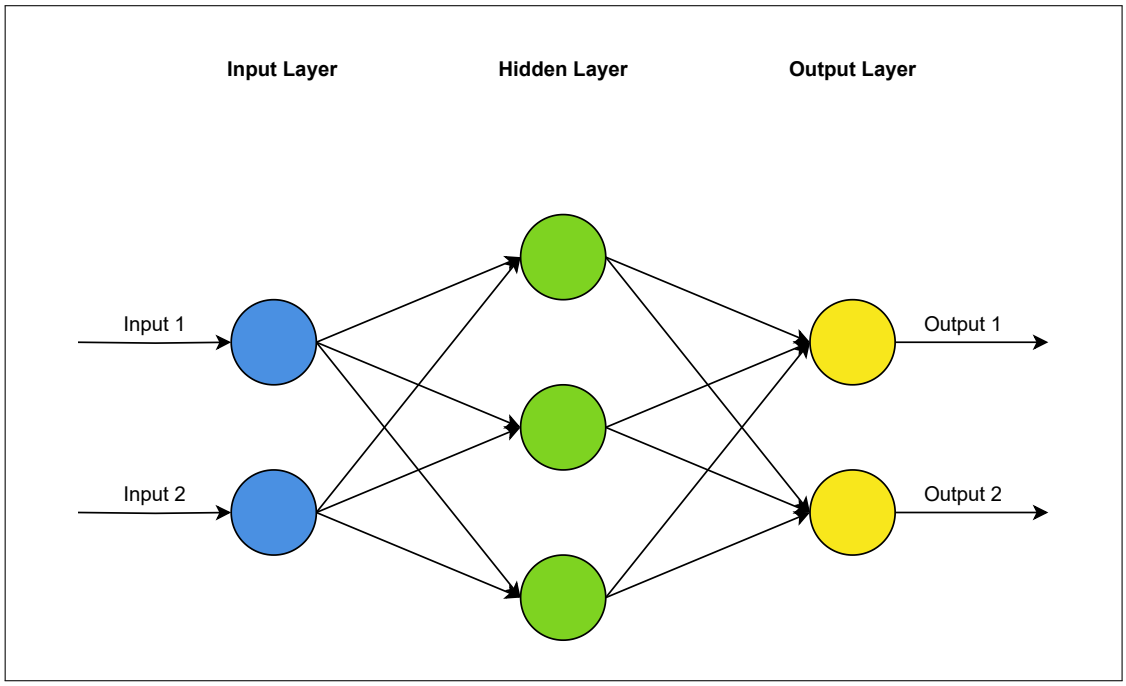


Figure 2.1.: An illustration of a feed-forward neural network. The input layer consists of 2 nodes, with the hidden layer consisting of 3 nodes and 2 for the output layer. Typically, a bias term is added to each layer except the output layer.

finally to the output layer, as shown in Figure 2.1.

A fully connected hidden layer with inputs $\mathbf{x} \in \mathbb{R}^n$ and outputs $\hat{\mathbf{x}} \in \mathbb{R}^m$ has weights $\mathbf{W} \in \mathbb{R}^{n \times m}$, biases $\mathbf{b} \in \mathbb{R}^m$ and an differentiable activation function $\sigma : \mathbb{R} \to \mathbb{R}$ that is applied component-wise. Non-linear activation functions are used to avoid the collapse of the DNN to an affine linear function. Some of the prevalent examples are the sigmoid function $\sigma(x) = \frac{1}{1-exp(x)}$ and the ReLU function $\sigma(x) = \max(0, x)$. An L-layer fully connected DNN with weights $\mathbf{W}_1, \cdots, \mathbf{W}_L$ with biases $\mathbf{b}_1, \cdots, \mathbf{b}_L$ can be formulated as:

$$y = \mathbf{W}_L \sigma(\mathbf{W}_{L-1} \sigma(\cdots \sigma(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1) \cdots) + \mathbf{b}_{L-1}) + \mathbf{b}_L \tag{2.17}$$

Some architectural examples that make use of DNNs are ResNet [He et al., 2016], VGG [Simonyan and Zisserman, 2014], AlexNet [Krizhevsky et al., 2012], and, most importantly, for our case, the vision transformer [Dosovitskiy et al., 2020]. We will look into the vision transformer further in the upcoming sections.

2.3. Natural Language Processing

Natural Language Processing (NLP) is a field within artificial intelligence (AI) that aids computers in comprehending, elucidating, and handling natural language [Rizzo, 2022]. NLP involves using machine learning to enable computers to understand and interact with human language. Recent advances in the field include the development of large language models (LLMs) [Achiam et al., 2023; Zhu et al., 2023; Devlin, 2018], which are sophisticated models trained on vast amounts of data to enhance their understanding of language and generative capabilities. But before we venture into them, we need to understand some key concepts in NLP.

2.3.1. Word Embeddings

Word embeddings play a key role in NLP, as all text-based data must be converted into a format that is computationally processible [Rizzo, 2022]. **Words** must be transformed into numerical vectors to make this information usable for computers. Thus, a **word embedding** is essentially the vector-based numerical representation of a word [Rizzo, 2022]. A prior idea was of **static embeddings**, i.e, providing fixed vector representation to each word [Salton et al., 1975; Mikolov, 2013; Pennington et al., 2014]. One way to do this is to utilize the technique of one-hot encoding. For example, if we have a vocabulary V of 3 words, for instance V = (apple, telephone, cat), then one-hot encoding can be created from that [Rizzo, 2022], such as

$$\text{cat} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \text{apple} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad \text{telephone} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}.$$

However, there are a few drawbacks to this approach [Rizzo, 2022]. Firstly, it is context-independent, as each word is given the same representation, eliminating any context [Rizzo, 2022]. Secondly, the dependence of the dimension of each vector on the vocabulary's dimensions which can lead to the curse of dimensionality in the case of vast vocabularies [Hastie, 2009; Rizzo, 2022]. Finally, words missing from the vocabulary are unrepresentable [Rizzo, 2022].

To work around this, one can choose NN-based embeddings like **Continuous Bag Of Words** (CBOW) [Mikolov, 2013] and **Skip-Gram** [Mikolov, 2013].

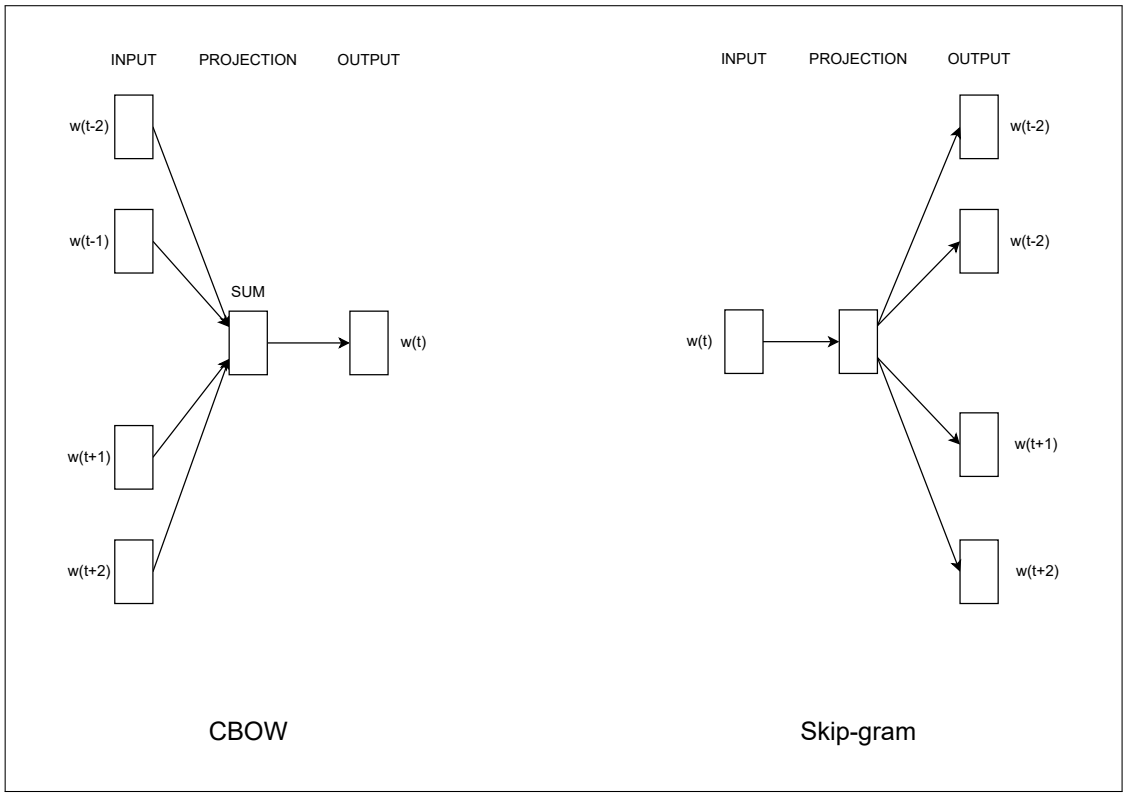


Figure 2.2.: Model architectures of CBOW and Skip-Gram. Image sourced from [Mikolov, 2013].

They consist of 3 layers: input, projection, and output layer [Mikolov, 2013]. CBOW aims to predict a target word based on context words within a fixed window size [Rizzo, 2022]. For instance, given the sentence "*The cat sits on the mat*", it will use ["*The*", "*cat*"] and ["*on*", "*the*"] to predict the word "*sits*". In contrast, the Skip-Gram model attempts to maximize the classification of a word from another word in the sentence [Mikolov, 2013]. Using the previous example, it would predict ["*The*", "*cat*"] and ["*on*", "*the*"] given "*sits*" [Rizzo, 2022]. The training of word embeddings occurs by the representation of words in one-hot encoding, followed by a pass through the hidden layer and, subsequently, computation of probabilities via softmax function [Mikolov, 2013; Rizzo, 2022]. With the aid of this process, the weight matrix of the hidden layer is learned, representing continuous and low-dimensional embeddings [Rizzo, 2022]. These weights are optimized using standard training techniques, and the embedding dimension[1] is a tunable hyperparameter. In particular, CBOW is faster to train, while Skip-Gram performs better for rare words [Mikolov, 2013].

These methods also allow us to measure the similarity between embeddings, say

[1]It is determined by the hidden layer size.

for words w_A and w_B, using metrics such as **cosine similarity** S_C [Salton et al., 1975], which is calculated as:

$$S_C(\mathbf{A}, \mathbf{B}) := \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|_2 \|\mathbf{B}\|_2} \tag{2.18}$$

where $\mathbf{A}, \mathbf{B} \in \mathbb{R}^d$ denote the vector embeddings for words w_A and w_B respectively, θ is the angle between them, and d denotes the dimension size of the embeddings.

The range of values S_C can take is $[-1, 1]$, where -1 denotes complete dissimilarity between two words w_A and w_B and 1 denotes absolute similarity between them. An example of this could be words like "*Hot*" and "*Cold*" for the lower value, whereas "*Happy*" and "*Joyful*" suffices for the upper value. Nevertheless, the issue of context-based representation persists, only to be solved by Transformers architecture, which we will visit in the upcoming sections. We next look at tokenization.

2.3.2. Tokenization

If a word is not found in the training corpus, computing its embedding becomes challenging [Rizzo, 2022]. This issue can be addressed using tokens, which [Schütze et al., 2008] describe as sequences of characters grouped into meaningful units for processing. **Tokenization** involves dividing text into smaller character sequences called tokens [Jurafsky and Martin, 2025]. Although splitting text into white spaces is a form of tokenization, relying on individual words as tokens can be problematic [Rizzo, 2022]. Instead, using parts of words or short letter sequences, known as character-level n-grams [Cavnar et al., 1994], where n is a flexible value, helps overcome the limitations of word-based tokenization [Rizzo, 2022]. But before that, we need to define n-grams.

An n**-gram** is a sequence of n words [Jurafsky and Martin, 2025]. For instance, given a caption *"I love machine learning"*, an unigram or 1-gram is written as [*"I", "love", "machine", "learning"*], 2-gram as [*"I love", "love machine", "machine learning"*], 3-gram as [*"I love machine", "love machine learning"*] and 4-gram as [*"I love machine learning"*]. A **character-level** n**-gram** is defined as an n-character slice of a word [Cavnar et al., 1994]. A character-level 2-gram can be written for a word *"love"* as [*"_l", "lo", "ov", "ve","e_"*] [Cavnar et al., 1994].

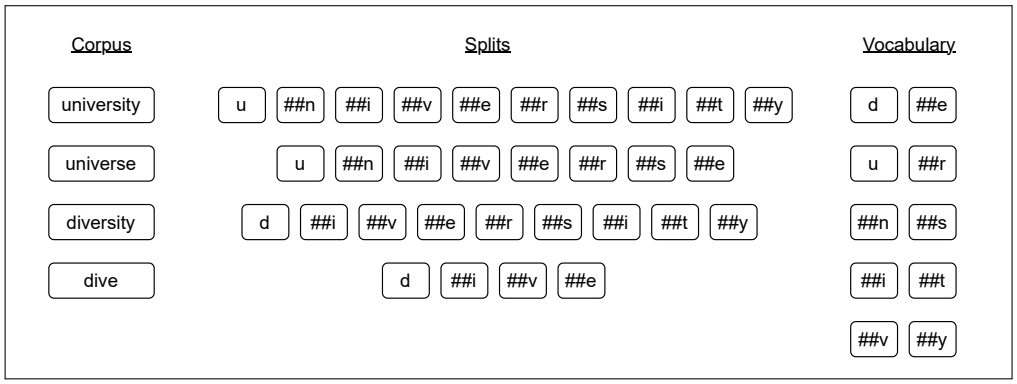


Figure 2.3.: Example of WordPiece tokenization. The ## indicates the character is not at the beginning of the word. Image sourced from [Rizzo, 2022].

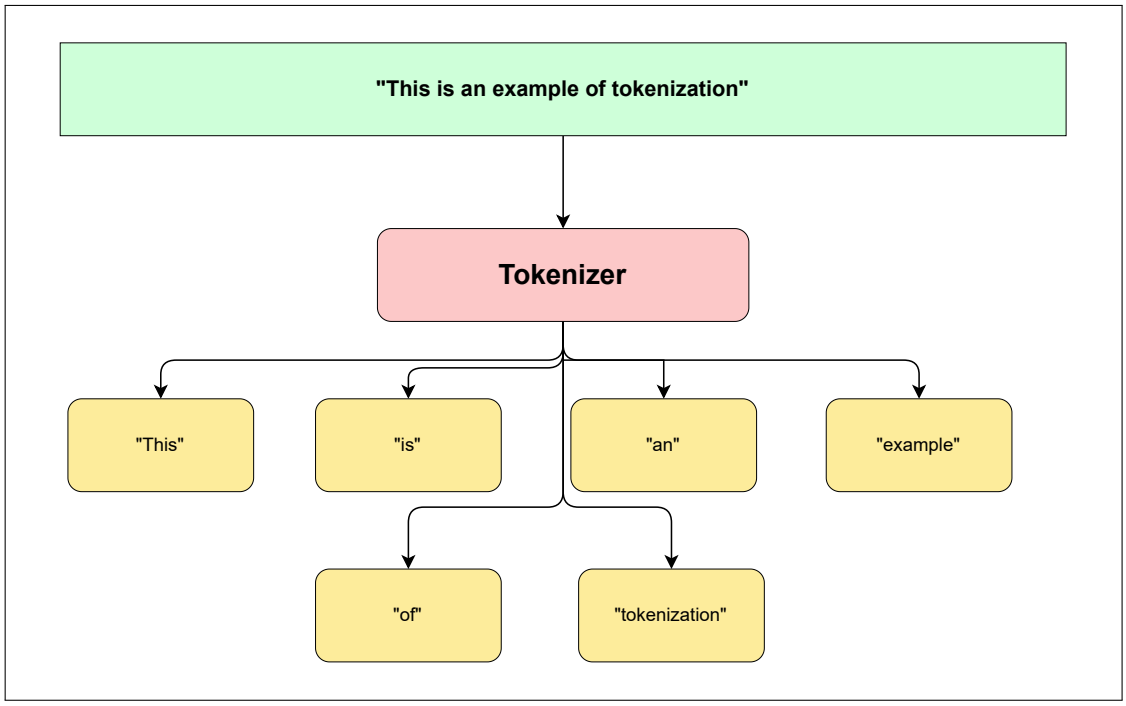


Figure 2.4.: An example of tokenization.

Common algorithms for tokenization include **byte-pair encoding** (BPE) [Sennrich, 2015] and **WordPiece** (WP) [Devlin, 2018]. BPE identifies the most frequent pairs of consecutive bytes (letters or symbols) in a document and replaces them with a new unused character [Sennrich, 2015]. This process repeats until no further compression is possible [Sennrich, 2015]. Modern adaptations of BPE, such as in [Sennrich, 2015], focus on building vocabularies rather than compressing documents. SentencePiece [Kudo, 2018] also incorporates elements of BPE.

$$\text{Score} = \frac{f_p}{f_{first} \times f_{second}} \tag{2.19}$$

WP [Wu et al., 2016] is a tokenization technique used for preprocessing text data, splitting words into subword units based on statistical scores. It begins by breaking all words into single characters (character-level unigrams) to form an initial vocabulary [Wu et al., 2016]. Scores, given in Eq. 2.19, are calculated for pairs of character-level unigrams using their frequency and co-occurrence probability, and the pair with the highest score is merged into a new token [Rizzo, 2022]. The f_p denotes the frequency of a pair, f_{first} is the frequency of the first element, and f_{second} is the frequency for the second element. This process is repeated iteratively, with new tokens added to the vocabulary, until a desired vocabulary size or likelihood increase is achieved [Rizzo, 2022]. WP is language-independent and enables the handling of rare or unknown words effectively by breaking them into meaningful subword components [Wu et al., 2016].

Now, before we look into large language models (LLM), we need to look into the Transformers architecture as that forms the backbone of the majority of the LLMs used today.

2.4. Transformer

The **Transformer** architecture, introduced in [Vaswani et al., 2017], utilizes self-attention to gather the long-range dependencies of its inputs and outputs. **Self-attention**, also known as intra-attention, is an attention mechanism that connects various positions within a single sequence to generate a representation of that sequence [Vaswani et al., 2017]. The model works by availing the encoder-decoder structure [Cho, 2014; Bahdanau, 2014]. In this, the **encoder** transforms an input sequence of symbol representations $\mathbf{x} = (x_1, \cdots, x_n)$ into a sequence of continuous representations $\mathbf{z} = (z_1, \cdots, z_n)$. The **decoder** then puts together an output sequence $\mathbf{y} = (y_1, \cdots, y_m)$ a single element at a time.

2.4.1. Encoder

The encoder consists of a stack of six identical layers (N = 6). Each layer contains two sub-layers: the first is a multi-head self-attention mechanism, and the second is a straightforward position-wise fully connected feed-forward network [Vaswani et al., 2017]. A residual connection is applied around each of the two sub-layers, followed by layer normalization [Ba, 2016]. To aid these residual connections, all sub-layers in the model, along with the embedding layers, generate outputs with a dimension $d_{model} = 512$ [Vaswani et al., 2017].

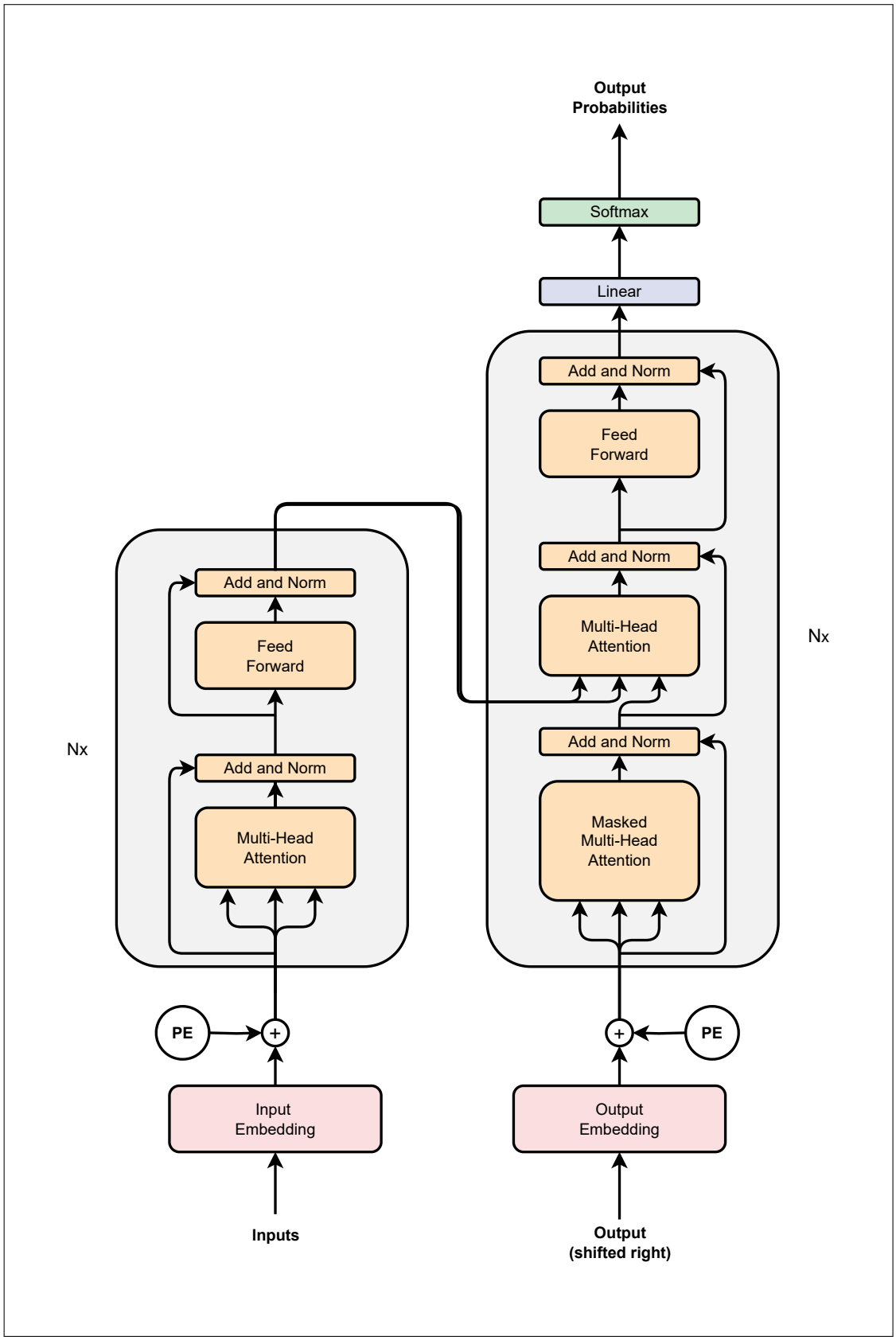


Figure 2.5.: The Transformer architecture. Image sourced from [Vaswani et al., 2017].

2.4.2. Decoder

The decoder, similar to the encoder, also comprises of a stack of six identical layers ($N = 6$). In addition to that, it also has another sub-layer that executes multi-head attention over the output produced by the encoder, along with the usage of residual connections and layer normalization as before [Vaswani et al., 2017]. The self-attention sub-layer in the decoder is also altered to make sure that there is no attending of future positions by current positions.

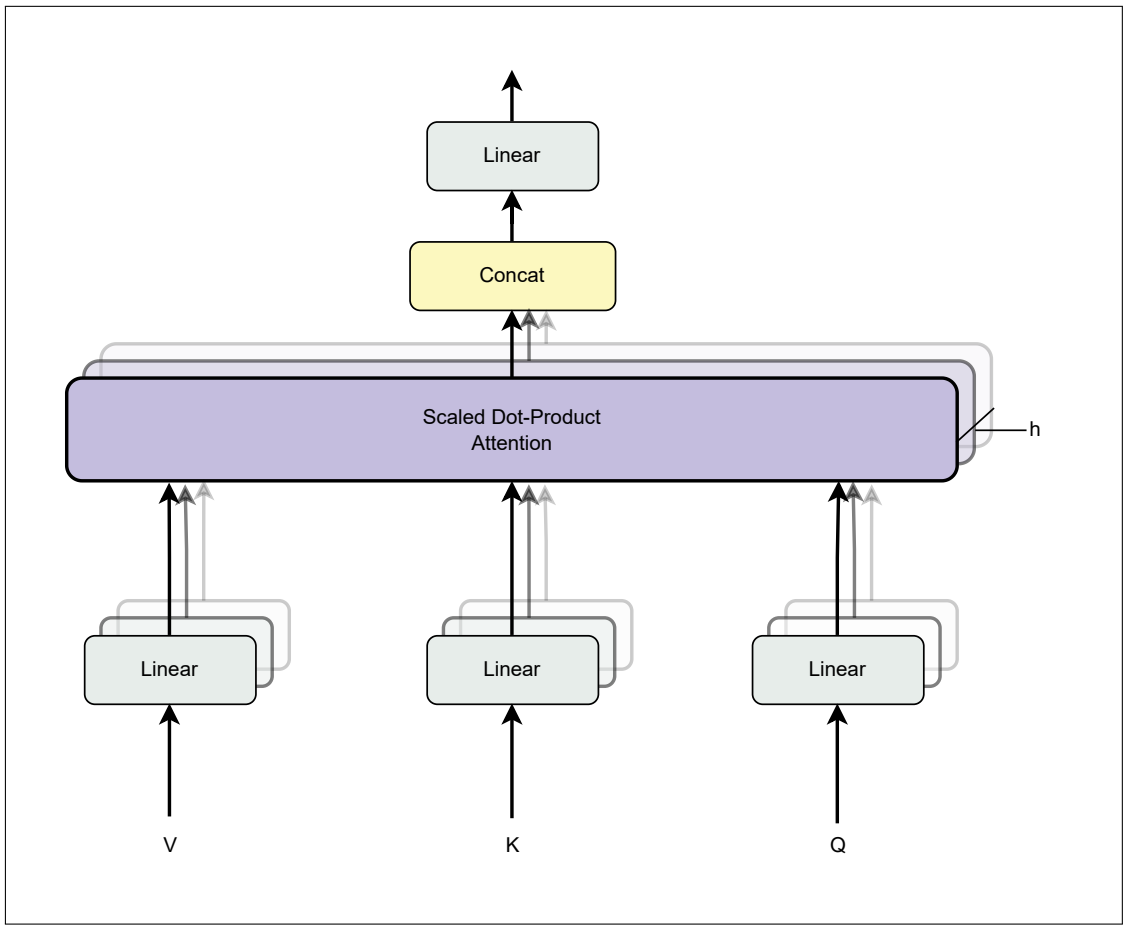


Figure 2.6.: A diagram of multi-head attention. Image sourced from [Vaswani et al., 2017].

2.4.3. Attention

The attention function maps a query and a set of key-value pairs to an output [2]. In this, the output is calculated as the weighted sum of the values. The weight

[2] All of them are vectors

assigned to each value is determined by a similarity function that assesses the query's relation to the corresponding key [Vaswani et al., 2017], as in Figure 2.7.

In [Vaswani et al., 2017], specific attention, called **Scaled Dot-Product Attention** is utilized. In this, the dot product of a query is computed for all keys, where the dimension of queries and keys is d_k and d_v for values. The dot products are then divided by $\sqrt{d_k}$, followed by a softmax function to gain the weights of the values [Vaswani et al., 2017].

Let the keys, values, and the set of queries be matrices and be denoted by Q, K and V, respectively. The attention is calculated as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{2.20}$$

Additionally, the queries, keys, and values are linearly projected h times with distinct learned linear projections to d_q, d_k and d_v dimensions, respectively. This is then followed by the attention function in parallel on every projected version of them, producing d_v-dimensional output values. They are then concatenated and once again projected, resulting in the final values. This is called **Multi-head attention** [Vaswani et al., 2017]. Multi-head attention enables the model to simultaneously focus on information from various representation subspaces at distinct positions [Vaswani et al., 2017].

$$\begin{aligned} \text{MultiHead}(Q, K, V) &= [\text{head}_1; \text{head}_2; \cdots ; \text{head}_h]W^o \\ \text{where head}_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{2.21}$$

Here, the projection are parameters matrices $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$ and $W_i^o \in \mathbb{R}^{hd_v \times d_{model}}$.

2.4.4. Point-wise Feed-Forward Networks

Each layer in the encoder and decoder includes a fully connected input network that is applied separately and similarly to each position [Vaswani et al., 2017]. This network comprises two linear transformations with a ReLU activation function in between.

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{2.22}$$

The input and output dimensionality is $d_{model} = 512$, while the inner layer has a dimensionality of $d_{ff} = 2048$.

2.4.5. Embeddings and Softmax

The learned embedding is used to convert the input and output tokens to vectors of dimension d_{model}. Standard learned linear transformation and softmax function

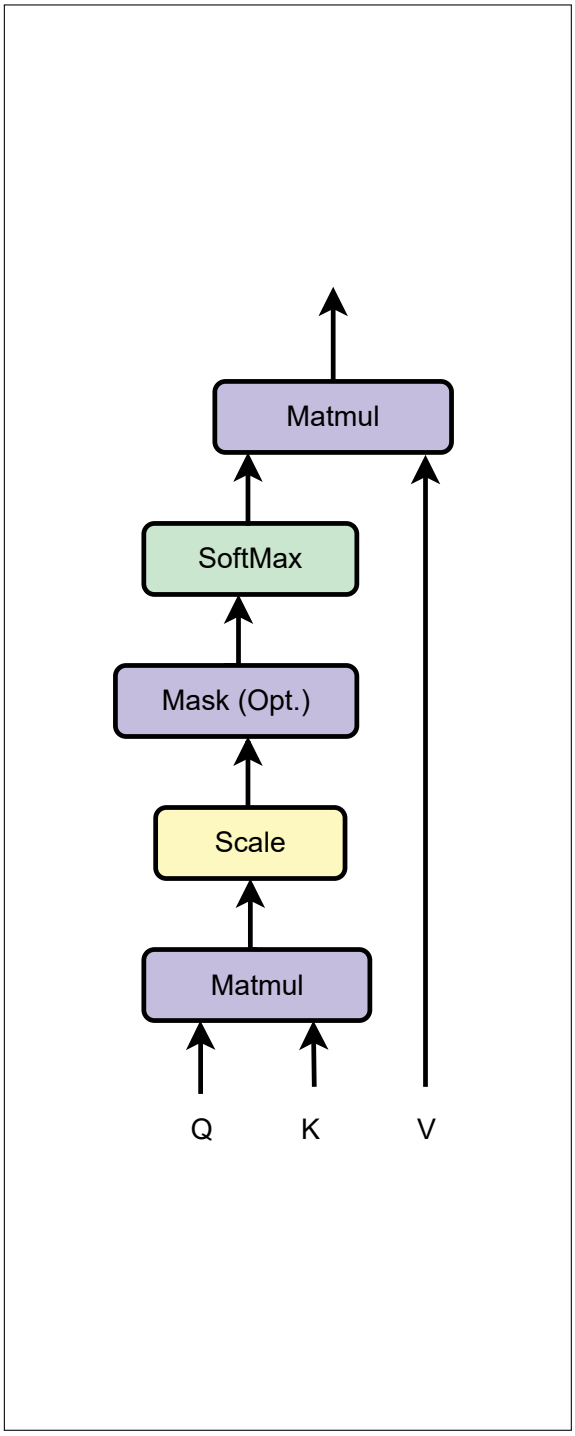


Figure 2.7.: A diagram of scaled dot-product attention. Image sourced from [Vaswani et al., 2017].

are also applied to map the decoder's output to the forecasted probabilities for the next token [Vaswani et al., 2017]. In the embedding layers, the weights are scaled by multiplying them with $\sqrt{d_{model}}$.

2.4.6. Positional Encoding

The positional encoding assists in ordering the input sequence due to the absence of convolution or recurrence. The encodings are added to the bottom of the encoder and decoder stacks and have the same dimension d_{model} as the embeddings for summing them up.

$$\begin{aligned} \mathrm{PE}_{(\mathrm{pos},2i)} &= \sin(\mathrm{pos}/10000^{2i/d_{model}}) \\ \mathrm{PE}_{(\mathrm{pos},2i+1)} &= \cos(\mathrm{pos}/10000^{2i/d_{model}}) \end{aligned} \tag{2.23}$$

where PE denotes positional encoding. The reason behind deciding on this function comes from the hypothesis [Vaswani et al., 2017] that this would enable the model to efficiently learn to focus on relative positions, as for any given offset k, PE_{pos+k} can be expressed as a linear function of PE_{pos}.

The architecture was trained and tested on multiple tasks, such as machine translation and English constituency parsing [Vaswani et al., 2017]. It could also be used for modalities other than text, like images [Dosovitskiy et al., 2020] and audio [Verma and Berger]. We will be looking into models that use this architecture in upcoming sections.

2.5. Vision Encoder Models

The **Vision Encoder Models**, first introduced in [Dosovitskiy et al., 2020], encode a raw visual input into a more compact and representative form, often called a latent vector, which can be used for tasks like image classification [Dosovitskiy et al., 2020].

In a vision transformer (ViT) [Dosovitskiy et al., 2020], an image $\mathbf{x} \in \mathbb{R}^{C\times H\times W}$, where the height, width and number of color channels are denoted by H, W and C respectively, is broken up into $N = HW/P^2$ patches, flattened, and then mapped to D dimensions with a learnable linear projection, which is then referred to as **patch embeddings**. Here, D is the constant latent vector size the Transformer uses throughout all layers, and (P, P) is the resolution for each image patch [Dosovitskiy et al., 2020]. A trainable embedding ($\mathbf{z}_0^0 = \mathbf{x}_{\text{class}}$) is also added to the beginning of embedding patches. The final state of this embedding after passing through the Transformer encoder ($\mathbf{z}_L^0$) is used as the image representation $\mathbf{y}$ [Dosovitskiy et al., 2020]. Here, L is the number of layers in the Transformer encoder architecture. Positional embeddings are also incorporated into the patch

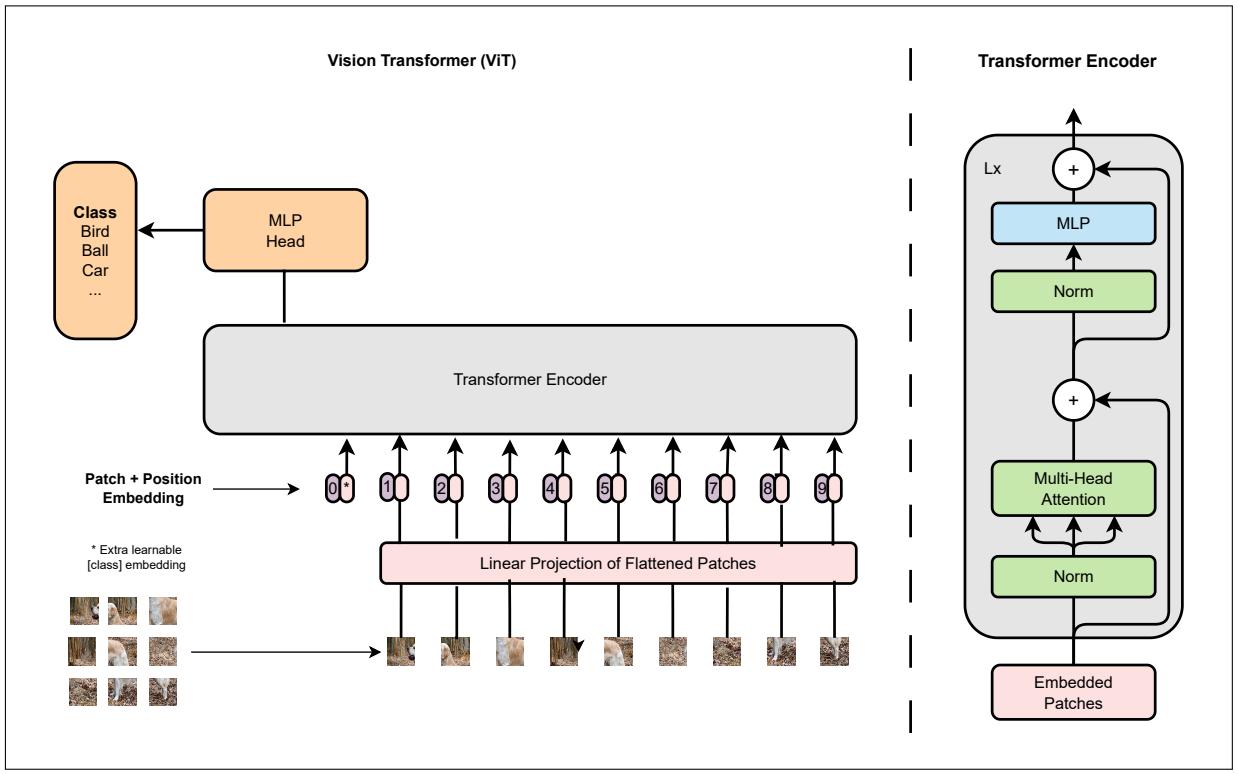


Figure 2.8.: An overview of the Vision Transformer model. Image Source: [Dosovitskiy et al., 2020]

embeddings to preserve positional information, ensuring the model knows the spatial arrangement of the image patches [Dosovitskiy et al., 2020].

From [Vaswani et al., 2017], interleaved layers of multiheaded self-attention and Multi-Layer Perceptron (MLP) blocks are present in the Transformer encoder, with Layernorm (LN) applied prior to every block and residual connection following afterward from that [Wang et al., 2019; Baevski and Auli, 2018]. This can be formulated as [Dosovitskiy et al., 2020]:

$$
\begin{aligned}
\mathbf{z}_0 &= [\mathbf{x}_{\text{class}}; \mathbf{x}_p^1\mathbf{E}; \mathbf{x}_p^2\mathbf{E}; \cdots; \mathbf{x}_p^N\mathbf{E}] + \mathbf{E}_{\text{pos}}, \\
&\quad \mathbf{E} \in \mathbb{R}^{(P^2 \cdot C) \times D}, \quad \mathbf{E}_{\text{pos}} \in \mathbb{R}^{(N+1) \times D}. \\
\mathbf{z}'_l &= \text{MSA}(\text{LN}(\mathbf{z}_{l-1})) + \mathbf{z}_{l-1}, \quad l = 1, \ldots, L. \\
\mathbf{z}_l &= \text{MLP}(\text{LN}(\mathbf{z}'_l)) + \mathbf{z}'_l, \quad l = 1, \ldots, L. \\
\mathbf{y} &= \text{LN}(\mathbf{z}_L^0).
\end{aligned}
\tag{2.24}
$$

where MSA denotes Multi-headed Self Attention. The MLP consists of two layers, with a Gaussian Error Linear Unit (GELU) [Hendrycks and Gimpel, 2016] activation function applied between them. Here, $\mathbf{E}$ is the patch embedding projection. An MLP classification head is attached to $\mathbf{z}_L^0$ with a single hidden layer.

We will be using it as the vision encoder of the OpenFlamingo [Awadalla et al., 2023] family of models and in CLIP [Radford et al., 2021] models.

2.6. Large Language Models

Large Language Models (LLMs) are language models that have been trained for natural language processing tasks like machine translation [Anil et al., 2023] and text generation [Team et al., 2024]. They are trained on enormous amounts of data. Popular examples are GPT-3 [Brown, 2020], MPT-7B [Team, 2023], and GPT-2 [Radford et al., 2019]. Given a sequence of tokens $\mathbf{x} = (x_1, x_2, \cdots x_n)$, the objective is to model the conditional probability[3] of predicting a particular token x_i given all or a subset of the other tokens in the sequence, parameterized by θ [Fang et al., 2024].

$$p_\theta(\mathbf{x}) = \prod_{i=1}^{n} p_\theta(x_i|\mathbf{x}_{<i}) \tag{2.25}$$

where x_i is the ith token and $\mathbf{x}_{<i}$ denotes the preceding tokens.

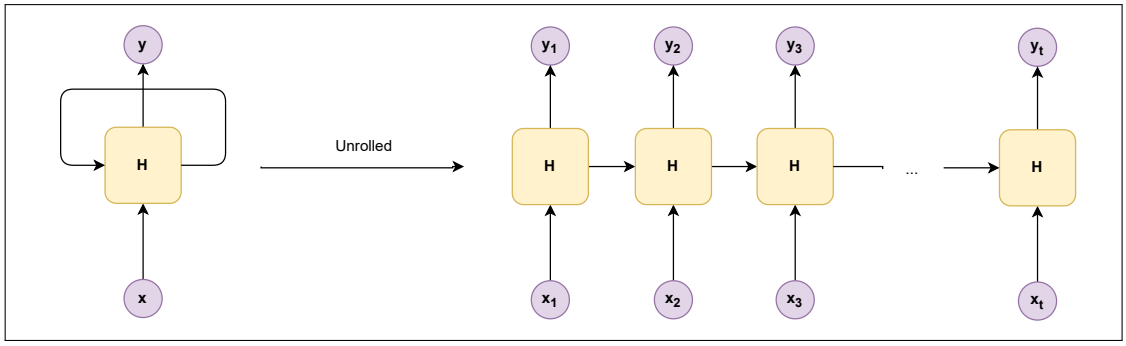


Figure 2.9.: An illustration of a RNN model. Image sourced from [Rumelhart et al., 1986].

The LLMs came as a replacement for previous NLP models like **Long Short-Term Memory** (LSTM) and **Recurrent Neural Network** (RNN) models. RNN models [Rumelhart et al., 1986] predict words by processing sequential data one step at a time, maintaining a hidden state that captures contextual information from previous time steps to aid in the prediction of the next word in the sequence. For a given input $\mathbf{x} = (x_1, x_2, \cdots, x_t)$, it outputs $\mathbf{y} = (y_1, y_2 \cdots, y_t)$ by sequentially processing the data from left to right, the model updates the hidden state at each time step and generates an output, as given in the Figure 2.9.

[3]Let A, B be events. We denote $p(B|A) = \frac{p(A \cap B)}{p(B)}$ as the conditional probability of B happening given A has occurred.

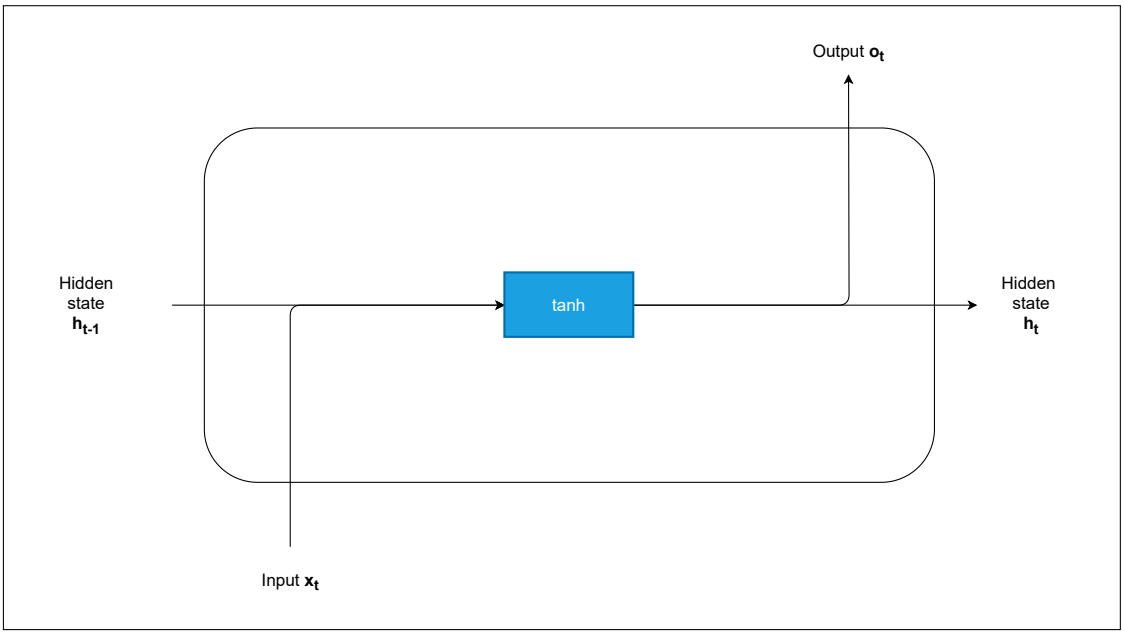


Figure 2.10.: An illustration of a hidden unit of a RNN model. Image sourced from [Rumelhart et al., 1986].

Given n samples of input[4] $\mathbf{x}^{(t)} \in \mathbb{R}^{n \times d}$, outputs[5] $\mathbf{y}^{(t)} \in \mathbb{R}^{n \times q}$, $\mathbf{h}^{(t-1)} \in \mathbb{R}^{n \times h}$ as the hidden layer output at time $t-1$, $W^{\mathbf{xh}} \in \mathbb{R}^{d \times h}$ that maps from input to hidden state, $W^{\mathbf{hh}} \in \mathbb{R}^{h \times h}$ that maps from the previous hidden state to the current hidden state, $W^{\mathbf{hy}} \in \mathbb{R}^{h \times q}$ that maps from hidden state to output, $\mathbf{b}^{\mathbf{y}} \in \mathbb{R}^{1 \times q}$ the bias term for output and $\mathbf{b}^{\mathbf{h}} \in \mathbb{R}^{1 \times h}$ the bias term for the hidden state. Then, RNN can be formulated as follows [Rumelhart et al., 1986].

$$\begin{aligned} \mathbf{h}^{(t)} &= \tanh(\mathbf{h}^{(t-1)} W^{\mathbf{hh}} + \mathbf{x}^{(t)} W^{\mathbf{xh}} + \mathbf{b}^{\mathbf{h}}), \\ \mathbf{y}^{(t)} &= \mathbf{h}^{(t)} W^{\mathbf{hy}} + \mathbf{b}^{\mathbf{y}} \end{aligned} \tag{2.26}$$

where h is the number of hidden units.

However, a major limitation of RNN models is that they often struggle to handle long sequences effectively due to issues with gradient propagation [Bengio et al., 1994; Hochreiter, 1997]. As gradients flow backward through time, they can either vanish or explode, hindering the model's ability to capture long-term dependencies [Bengio et al., 1994].

To counter the aforementioned limitation, LSTM models [Hochreiter, 1997] were introduced. They overcome the limitations of traditional RNNs by incorpo-

[4]Inputs have dimension d.
[5]Outputs have dimension q.

rating a memory cell and a system of gates: **input**, **output**, and **forget** gates [Hochreiter, 1997]. These gates control the flow of information, helping the network retain important details over extended sequences while filtering out unnecessary information [Hochreiter, 1997]. This structure reduces the problems of vanishing and exploding gradients, making it easier for LSTMs to capture and maintain long-term dependencies [Hochreiter, 1997]. Given $\mathbf{x}^{(t)}, \mathbf{y}^{(t)}$ and $\mathbf{h}^{(t-1)}$ from above, $W^{\mathbf{xi}}, W^{\mathbf{xf}}, W^{\mathbf{xo}}, W^{\mathbf{xc}} \in \mathbb{R}^{d \times h}$ and $W^{\mathbf{hi}}, W^{\mathbf{hf}}, W^{\mathbf{ho}}, W^{\mathbf{hc}} \in \mathbb{R}^{h \times h}$ are the weight parameters. The input, output and forgot gates are denoted by $\mathbf{i}^{(t)}, \mathbf{o}^{(t)}$ and $\mathbf{f}^{(t)} \in \mathbb{R}^{n \times h}$, respectively. The biases are denoted by $\mathbf{b}^{\mathbf{i}}, \mathbf{b}^{\mathbf{f}}, \mathbf{b}^{\mathbf{o}}, \mathbf{b}^{\mathbf{c}} \in \mathbb{R}^{1 \times h}$. Then, LSTM can be formulated as follows [Hochreiter, 1997].

$$
\begin{aligned}
\mathbf{i}^{(t)} &= \sigma(\mathbf{x}^{(t)} W^{\mathbf{xi}} + \mathbf{b}^{\mathbf{i}} + \mathbf{h}^{(t-1)} W^{\mathbf{hi}}), \\
\mathbf{f}^{(t)} &= \sigma(\mathbf{x}^{(t)} W^{\mathbf{xf}} + \mathbf{b}^{\mathbf{f}} + \mathbf{h}^{(t-1)} W^{\mathbf{hf}}), \\
\mathbf{o}^{(t)} &= \sigma(\mathbf{x}^{(t)} W^{\mathbf{xo}} + \mathbf{b}^{\mathbf{o}} + \mathbf{h}^{(t-1)} W^{\mathbf{ho}}), \\
\tilde{\mathbf{c}}^{(t)} &= \tanh(\mathbf{x}^{(t)} W^{\mathbf{xc}} + \mathbf{b}^{\mathbf{c}} + \mathbf{h}^{(t-1)} W^{\mathbf{hc}}), \\
\mathbf{c}^{(t)} &= \mathbf{f}^{(t)} \odot \mathbf{c}^{(t-1)} + \mathbf{i}^{(t)} \odot \tilde{\mathbf{c}}^{(t)}, \\
\mathbf{h}^{(t)} &= \mathbf{o}^{(t)} \odot \tanh(\mathbf{c}^{(t)}).
\end{aligned}
\tag{2.27}
$$

Here, $\odot$ is overloaded as the Hadamard product, and σ denotes the sigmoid function.

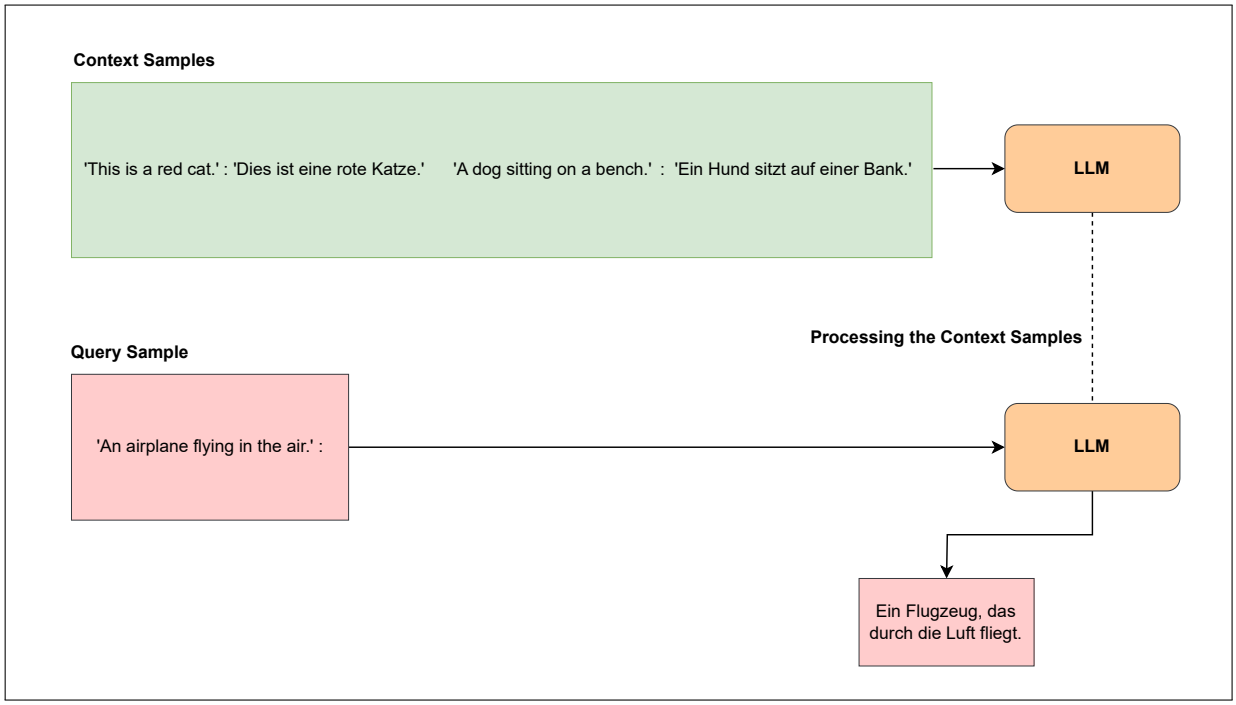


Figure 2.11.: Few-shot learning for the task for machine translation.

However, both have the property of sequential processing, i.e., sentences need to be processed one word at a time, which prevents training them in parallel [Vaswani et al., 2017].

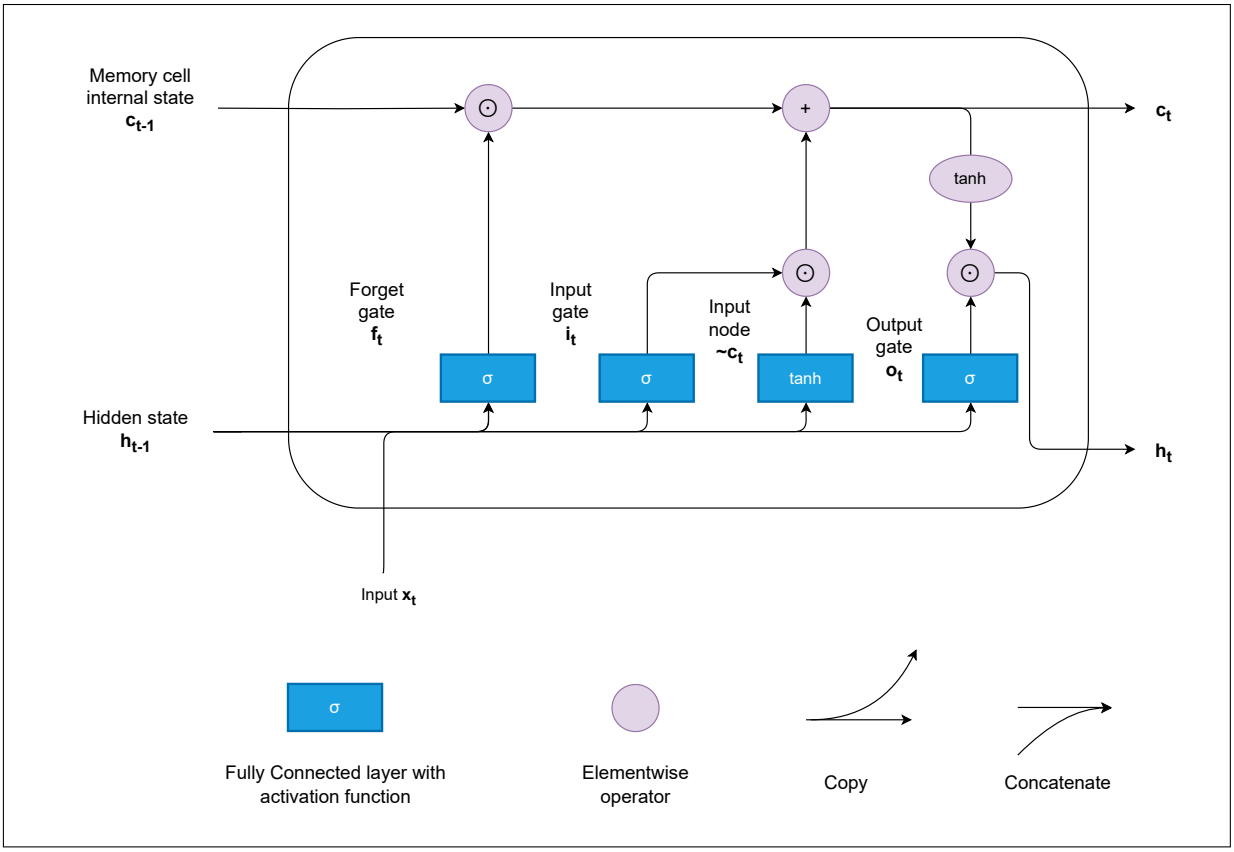


Figure 2.12.: An illustration of a hidden unit in a LSTM model. Image sourced from [Hochreiter, 1997].

Attention Mechanism [Vaswani et al., 2017] was introduced in 2017. It became the backbone of the transformer [Vaswani et al., 2017] architecture, which has transformed the field of NLP. Transformers, leveraging the Attention Mechanism, have lead to the development of LLMs that are capable of better understanding and generating human-like text than the earlier language models. Some instances of LLMs are **GPT** (Generative Pre-trained Transformer) [Achiam et al., 2023], **BERT** (Bidirectional Encoder Representations from Transformers) [Devlin, 2018], and most important for this work, the **MPT-7B** [Team, 2023] model. MPT-7B stands for MosaicML Pretrained Transformer and is one of the earlier models in the MosaicML Foundation series. It was trained on 1 trillion tokens of text and code and has about 7 billion parameters [Team, 2023]. It is an open-source large language model developed by the MosaicML Foundation.

2.7. Vision Language Models

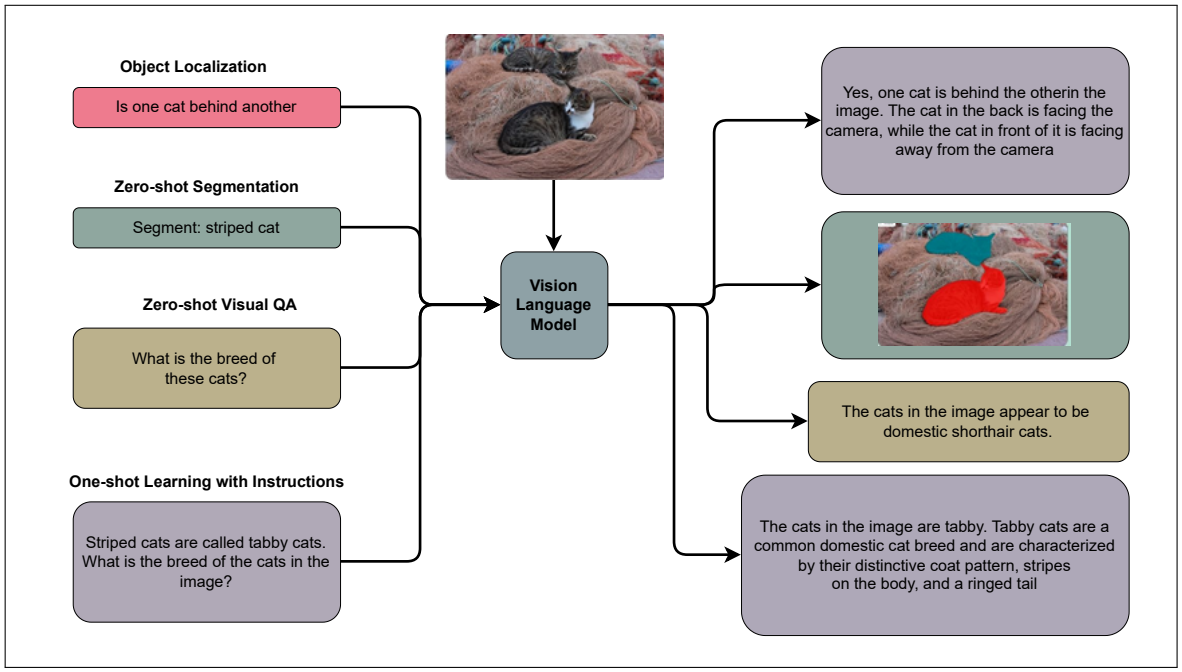


Figure 2.13.: An example of a vision-language model. Image sourced from [Wolf et al., 2020].

Vision-language models (VLMs) are a type of multi-modal models that acquire knowledge from both visual and textual data, enabling it to interpret visual inputs and generate text outputs [Bordes et al., 2024]. They combine a vision encoder with an LLM, allowing them to process text and visual data. These large models excel in many tasks, such as recognizing visuals based on instructions [Driess et al., 2023], answering visual questions [Awadalla et al., 2023; Alayrac et al., 2022], understanding documents [Kim et al., 2021], and generating image captions [Alayrac et al., 2022; Awadalla et al., 2023].

First, we define some terms. A **context** data sample helps provide context to a VLM about the ongoing task. For instance, in the case of image captioning, context image, along with the context text, provides a visual and textual context of the task to generate captions when an input image is given to the model, as in Figure 2.14. A **query** data sample is the image that is provided to a VLM so that an output can be generated. Again, in the case of image captioning, it is provided to a VLM to generate a caption. We will discuss more tasks (like image captioning and visual question answering) in more detail in the upcoming sections.

Finally, we define few-shot learning. **Few-shot learning** (or N**-shot learn-**

ing) is the task of understanding the underlying patterns in data given a few context data samples [Parnami and Lee]. In this, we give a VLM a few context samples to comprehend the task at hand, and then we provide a query sample to gain output from it. For instance, in **4-shot learning** for the task of visual question answering, we provide 4 pairs of context images and answers for a VLM to understand the task and then provide a query image to get an answer. An example of few-shot learning for the task of image captioning is given in Figure 2.14. In the case of **0-shot learning**, we provide absolutely no context samples whatsoever, and instead directly pass in the query image for output.

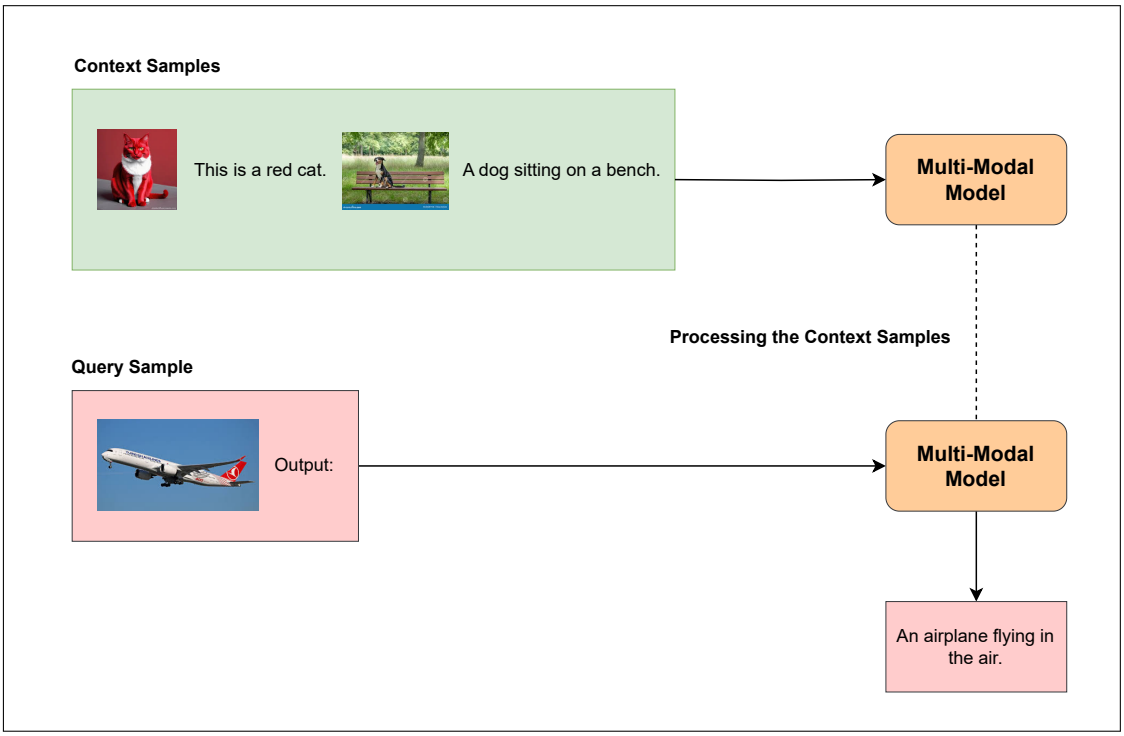


Figure 2.14.: Few-shot learning for the task of image captioning.

There are four particular families of VLMs based on different training paradigms [Bordes et al., 2024] - The **Contrastive-based** VLMs, **Masking-based** VLMs, **Generative-based** VLMs, and VLMs with **Pretrained-backbone**. A widely used approach, the **Contrastive** learning, is utilized in Contrastive-based VLMs. In this, the VLM is trained to produce similar representations for positive pairs of examples while ensuring distinct representations for negative pairs [Bordes et al., 2024]. **Positive examples** can be defined as a pair of inputs that are related or similar in some meaningful way, for instance, an image and its corresponding caption. In contrast, **negative examples** are the opposite of that, a pair of inputs

that are unrelated or dissimilar, like a pair of an image and a completely random caption. A prime example of this approach is the **Contrastive Language-Image Pretraining** (CLIP) [Radford et al., 2021] VLM, which achieves a shared representation space by training the model to integrate vision and language.

In **Masking-based** VLMs, the masking technique is used, in which the model learns to reconstruct missing image patches based on an unmasked text caption [Bordes et al., 2024] or predict masked words in a caption using an unmasked image as context [Bordes et al., 2024]. An instance of this would be **A Foundational Language and Vision Alignment Model** (FLAVA) [Singh et al., 2022], a transformer-based model pre-trained on 70 million image-text pairs.

Generative-based VLMs focus on the generation of images/texts, dissimilar to earlier training paradigms that primarily count upon latent representations to create abstractions of images or text for mapping between modalities [Bordes et al., 2024]. A text-to-image model, for example, would be trained to approximate the conditional likelihood of an image given a text prompt. An instance of Generative-based models would be CM3Leon [Yu et al., 2023] model.

Lastly, VLMs with **pre-trained backbones** combine pre-trained, frozen vision encoder and LLM to understand the text and image modality mapping [Bordes et al., 2024]. Some examples of such VLMs are Frozen [Tsimpoukelli et al., 2021], the OpenFlamingo family of models [Awadalla et al., 2023], and the Flamingo family of models [Alayrac et al., 2022].

We will be discussing the OpenFlamingo family of VLMs and CLIP models in the upcoming sections.

2.8. OpenFlamingo

OpenFlamingo [Awadalla et al., 2023] is a family of autoregressive vision language models. It is the open-source implementation of the Flamingo family of models [Alayrac et al., 2022]. It comprises the CLIP ViT-L/14 vision encoder [Radford et al., 2021] and open-source language models as the decoder. It was created due to the scarcity of open-source auto-regressive vision language models [Awadalla et al., 2023]. The family ranges from models with 3 billion parameters up to 9 billion parameters.

An OpenFlamingo model works by cross-attention of the frozen, pre-trained layers of the language model with that of the outputs of the frozen vision encoder [Awadalla et al., 2023]. It predicts the next text token in an interleaved sequence of images and text by conditioning on all previous text tokens and the most

recent preceding image [Awadalla et al., 2023]. An example of interleaved samples would be `x Good Morning`, where `x` is an image. It is then preprocessed as `<image> Good Morning <endofchunk>`, where the embeddings for `<image>` and `<endofchunk>` are learnable for most OpenFlamingo models [Awadalla et al., 2023].

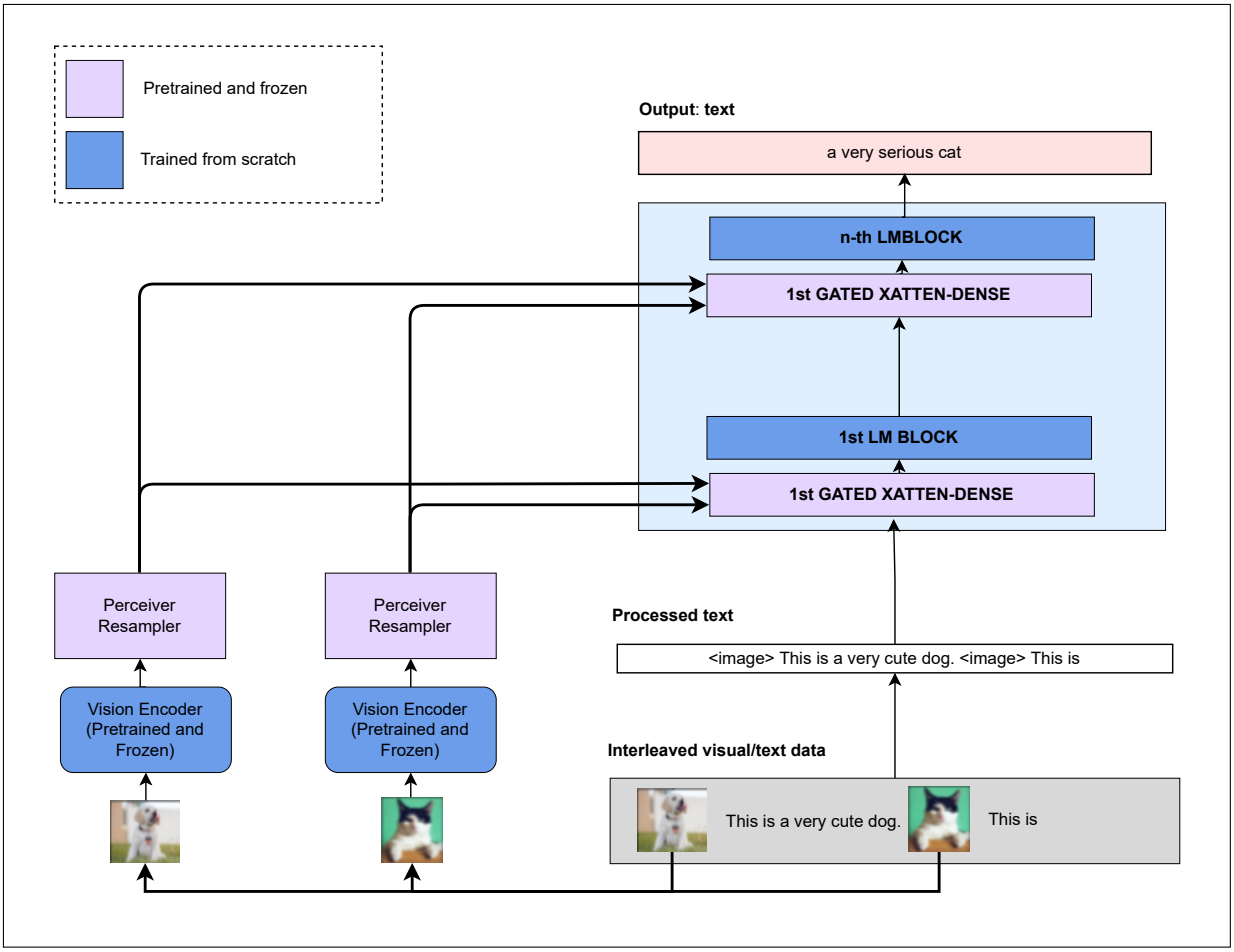


Figure 2.15.: An architectural overview of the closed-source Flamingo family of models. Image sourced from [Alayrac et al., 2022].

The model allows image-text attending by making use of dense cross-attention modules to connect image patch features (extracted via a frozen vision encoder and processed through a trainable Perceiver resampler [Jaegle et al., 2021]) with tokens in a frozen, autoregressive language model [Awadalla et al., 2023]. From [Schlarmann and Hein, 2023], the likelihood of text $\mathbf{y}$, given images $\mathbf{x}$, is:

$$p(\mathbf{y}|\mathbf{x}) = \prod_{l=1}^{L} p(y_l|\mathbf{y}_{<l}, \mathbf{x}) \tag{2.28}$$

where y_l is the lth language token and $\mathbf{y}_{<l}$ all the tokens preceding it.

The family of models was trained on the datasets **LAION-2B** [Schuhmann et al., 2022], **Multimodal C4** [Zhu et al., 2024], and synthetic data generated from ChatGPT [Achiam et al., 2023]. The results showed that in 7 evaluation datasets, the OpenFlamingo-3B and -9B models achieved 85% and 89% of the performance of their corresponding Flamingo models, respectively [Awadalla et al., 2023]. OpenFlamingo can also perform tasks like few-shot learning by taking some context images along with texts that describe them.

2.9. CLIP

Contrastive Language-Image Pairing (CLIP), introduced in [Radford et al., 2021], is an approach to learning from natural language supervision. Given M pairs of images and texts (image-text), it is trained to predict actual image-text pairs out of M^2 possible image-text pairs via the acquirement of knowledge regarding the multi-modal embedding through joint training of the image and text encoders for increasing the cosine similarity of the image and text embedding of the M real pairs while reducing it for the $M^2 - M$ wrong pairs [Radford et al., 2021].

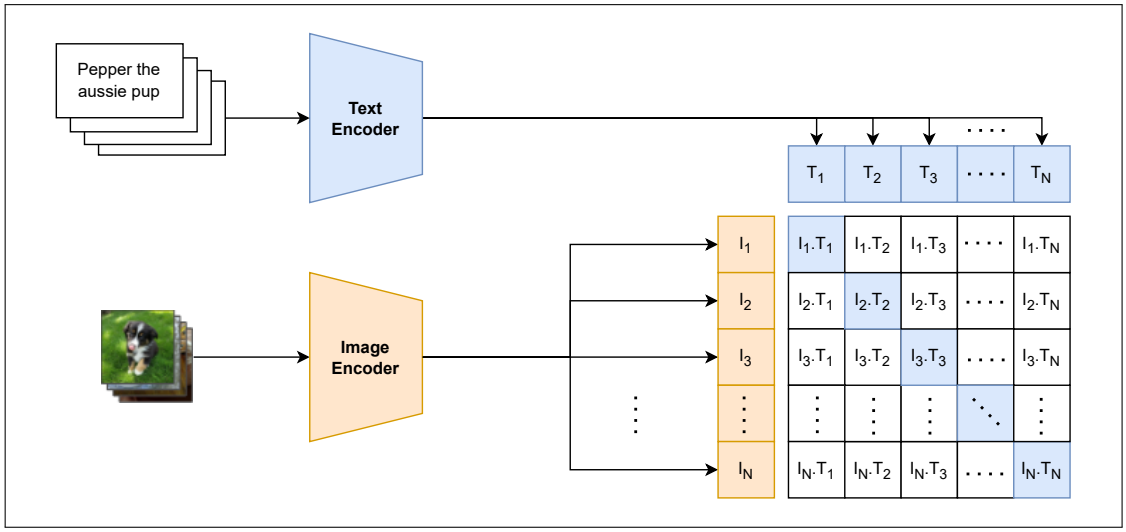


Figure 2.16.: An illustration of the joint training of the image encoder and the text encoder under CLIP to predict correct image-text pairs. Image sourced from [Radford et al., 2021].

The authors of [Radford et al., 2021] considered ResNet-50 [He et al., 2016] and ViT as the image encoders and a modified Transformer [Radford et al., 2019] as

the text encoder. The models were then trained on a newly constructed dataset consisting of 400 million image-text pairs [Radford et al., 2021].

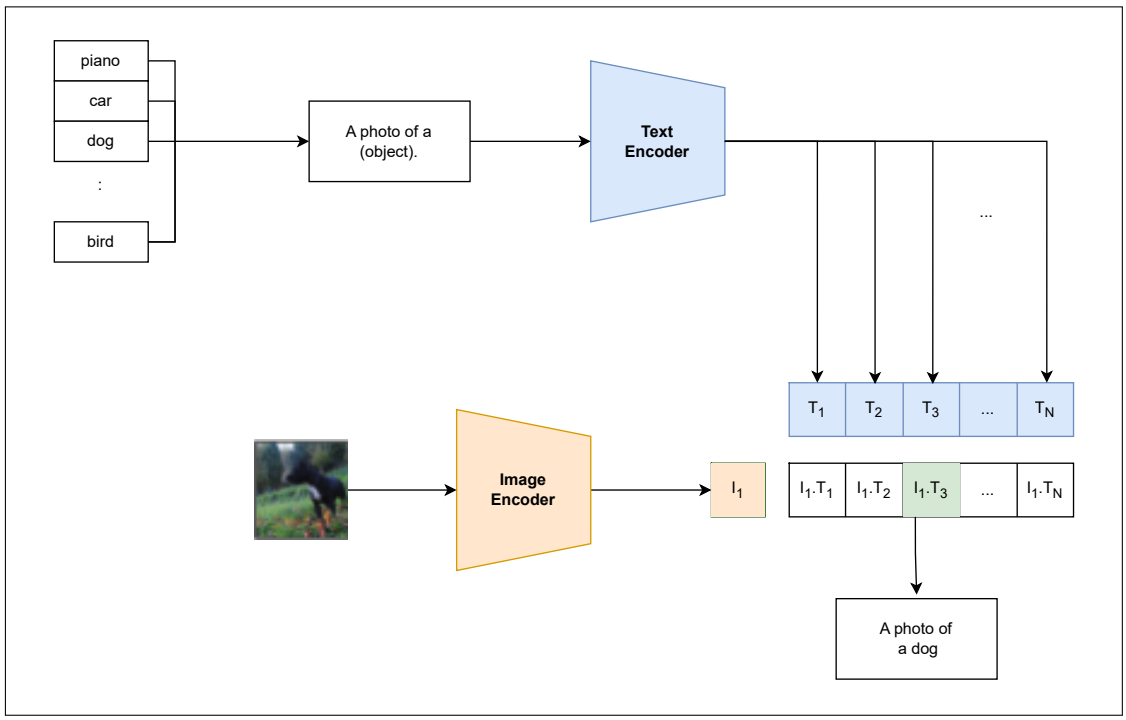


Figure 2.17.: A visualization of the 0-shot linear classifier assembled by the trained text encoder via embedding the names or descriptions of the target dataset's classes. Image sourced from [Radford et al., 2021].

CLIP is utilized for its 0-shot transfer capabilities in downstream tasks such as optical character recognition, geo-localization, and fine-grained object classification [Radford et al., 2021]. It has outperformed models like Visual N-Grams [Li et al., 2017] for 0-shot image classification on datasets such as ImageNet [Krizhevsky et al., 2012] and SUN [Xiao et al., 2010].

2.10. Adversarial Attacks

In an **adversarial attack**, an adversary might intend to insert perturbed data in the training phase or at the testing phase of a machine learning model to further its own adversarial goals. Commonly occurring attack scenarios are:

- **Evasion attacks.** In an evasion attack, the attacker seeks to avoid the detection of harmful samples by taking advantage of the weaknesses of the targeted model during the testing phase. For instance, perturbing pixels of an image to fool an image classifier [Madry et al., 2017; Croce and Hein, 2020; Carlini and Wagner, 2017].

- **Poisoning attacks.** In a poisoning attack, the adversary attempts to poison the training data for a model by adding carefully designed samples, leading to the deterioration of the model's performance. An example of this would be flipping the labels of a dataset to be used for training support vector machines [Xiao et al., 2012], or targeted poisoning attacks on neural networks [Shafahi et al., 2018].

- **Inference attacks.** In an inference attack, the adversary extracts sensitive information from a model by analyzing its outputs or behavior. An instance of that would be a membership inference attack in which an adversary seeks to determine whether a particular data sample was part of the model's training data or not by observing the loss given by the target model [Shokri et al., 2017], or estimating the target model's behavior by shadow training [Carlini et al., 2022].

- **Extraction attacks.** In an extraction attack, the parameters of a model are extracted by the adversary. For example, conducting an equation-solving attack to obtain exact replication of a target model[6] [Tramèr et al., 2016], or synthesizing artificial data via Jacobian-based dataset augmentation to steal the functionality of the target model [Papernot et al., 2017].

In our case, we focus on evasion attacks. These attacks have been primarily used in the task of image classification [Carlini and Wagner, 2017; Goodfellow et al., 2014; Chen et al., 2020a; Madry et al., 2017], where they created adversarial images that were misclassified by image classifiers. Given an image $\mathbf{x} \in \mathcal{X} := [0,1]^n \subset \mathbb{R}^n$ with label l, an image classifier $f : \mathcal{X} \to \mathbb{R}^K$ with the prediction function $F(\mathbf{x}) = \arg\max_j f_j(\mathbf{x})$, a loss function $\mathcal{L} : \mathbb{R}^k \times \{1 \cdots k\} \to \mathbb{R}$, some $c > 0$ and some distance metric $\mathcal{D} : \mathbb{R}^n \to \mathbb{R}$, an adversarial attack can be formulated as

$$\begin{aligned} &\min_{\mathbf{w}} \; -c \cdot \mathcal{L}(f(\mathbf{x}+\mathbf{w}), l) + \mathcal{D}(\mathbf{w}) \\ &\text{s.t. } \mathbf{x}+\mathbf{w} \in \mathcal{X} \end{aligned} \tag{2.29}$$

That was an example of an **untargeted attack**, in which the added perturbation $\mathbf{w}$ makes sure that any label, except for the original label l, is obtained from the classifier. In the case of a **targeted attack**, given a target label t where $t \neq l$, adding the perturbation $\mathbf{w}$ to the original image fools the model into predicting the target label t.

$$\begin{aligned} &\min_{\mathbf{w}} \; c \cdot \mathcal{L}(f(\mathbf{x}+\mathbf{w}), t) + \mathcal{D}(\mathbf{w}) \\ &\text{s.t. } \mathbf{x}+\mathbf{w} \in \mathcal{X} \end{aligned} \tag{2.30}$$

[6] Usually, linear models are chosen for the attack.

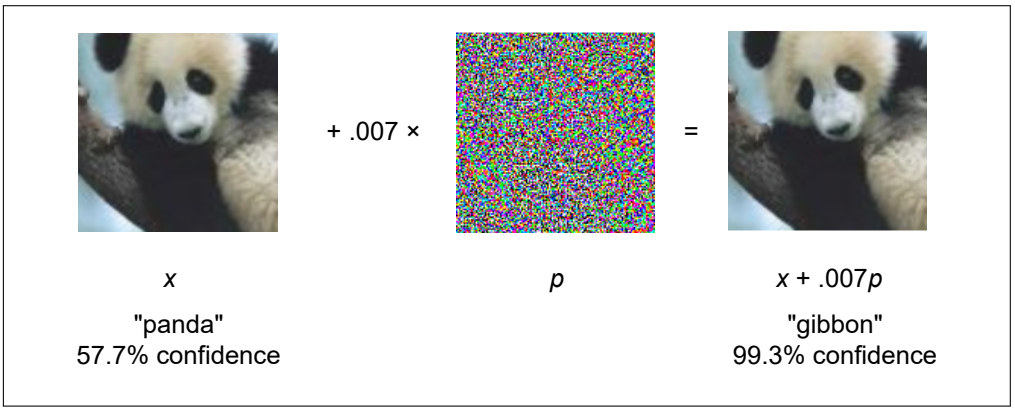


Figure 2.18.: An example of an adversarial attack. Here, the adversarial perturbation is denoted as p. Image sourced from [Goodfellow et al., 2014].

If the distance metric $\mathcal{D}$ is constrained by some $\epsilon \in \mathbb{R}$, then Eq. 2.29 and 2.30 can reformulated as:

$$\begin{aligned} &\min_{\mathbf{w}} \; -\mathcal{L}(f(\mathbf{x}+\mathbf{w}), l) \\ &\text{s.t. } \mathcal{D}(\mathbf{w}) \leq \epsilon, \, \mathbf{x}+\mathbf{w} \in \mathcal{X} \end{aligned} \tag{2.31}$$

and,

$$\begin{aligned} &\min_{\mathbf{w}} \; \mathcal{L}(f(\mathbf{x}+\mathbf{w}), t) \\ &\text{s.t. } \mathcal{D}(\mathbf{w}) \leq \epsilon, \, \mathbf{x}+\mathbf{w} \in \mathcal{X} \end{aligned} \tag{2.32}$$

For sparse adversarial attacks, the goal is to create perturbations that modify as few pixels as possible, which can be accomplished by defining the distance measure $\mathcal{D} = \|.\|_0$[7] or $\mathcal{D} = \|.\|_1$.

Adversarial attacks in the case of multi-modal models will be discussed in the upcoming sections.

2.11. Counterfactuals

Counterfactual explanations or **counterfactuals** are perturbed data samples with meaningful changes from the original data sample. This can be described as statements where a score p was returned because a variable V had values $(v_1, v_2, \ldots)$ but if V had values $(v_1', v_2', \ldots)$, then the score could have p' [Wachter et al., 2017]. An instance from [Wachter et al., 2017]

[7] Let $\mathbf{x} \in \mathbb{R}^n$. Then $\|\mathbf{x}\|_0 := \sum_{i=1}^{n} \mathbb{1}_{x_i \neq 0}$.

"You were denied a loan because your annual income was £30,000. You would have been offered a loan if your income had been £45,000."

This statement stands out as an adequate example of a statement of decision, immediately followed by a counterfactual. The fact that a small perturbation of the salary would have cleared the loan's eligibility is the very essence of a counterfactual. We formulate counterfactuals as follows

$$\arg\min_{\mathbf{x}'} \max_{\lambda} \lambda(f(\mathbf{x}') - \mathbf{y}')^2 + \mathcal{D}(\mathbf{x}' - \mathbf{x}) \tag{2.33}$$

Where $\mathbf{x}', \mathbf{x} \in \mathcal{X}$ are the counterfactual and the original data samples, respectively, $\mathbf{y}' \in [0, 1]$ is the new target score and $f : \mathcal{X} \to [0, 1]$ denotes a machine learning model that takes input and outputs a probability between 0 and 1.

Looked upon from [Wachter et al., 2017], the **1-norm MAD** attack introduces sparse perturbation into the input while maintaining the distance between the input and the perturbed input via the 1-norm weighted by median absolute deviation **(MAD)**. The authors of [Wachter et al., 2017] utilized the **ADAM** [Kingma and Ba, 2017] optimizer for the objective, as well as for training the model. The distance function being used is 1-norm weighted by the inverse median absolute deviation [Wachter et al., 2017]. This is written as

$$\mathcal{D}(\mathbf{w}) = \sum_{k \in F} \frac{|\mathbf{w}_k|}{\mathbf{MAD}_k} \tag{2.34}$$

Where $\mathbf{w} := \mathbf{x}' - \mathbf{x}$ and $\mathbf{MAD}_k$ is defined as

$$\mathbf{MAD_k} = \text{median}_{\mathbf{x} \in P}(|\mathbf{x}_k - \text{median}_{\mathbf{z} \in P}(\mathbf{z}_k)|) \tag{2.35}$$

Here, $P \subset \mathcal{X}$, $\mathbf{MAD_k}$ describes the median absolute deviation of feature k over the total number of data samples P, F is the set of features and $\mathbf{w}_k$ is the kth feature of $\mathbf{w}$. The distance has been described as having numerous desirable properties. Firstly, it takes into account some of the intrinsic volatility of the space [Wachter et al., 2017]. Secondly, it is more robust to outliers as compared to the usual standard deviation [Wachter et al., 2017]. Finally, it induces sparsity due to the usage of the 1-norm [Wachter et al., 2017].

In recent years, the **validity**, **proximity**, **sparsity**, and **diversity** of counterfactuals have also been discussed [Mothilal et al., 2020]. **Validity** [Mothilal et al., 2020] measures the proportion of examples generated by a method that truly results in different outcomes from the original input, focusing only on unique instances to avoid duplicates. Assume $\mathcal{C}$ to be a set of k counterfactual examples for an instance $\mathbf{x} \in \mathcal{X}$, then [Mothilal et al., 2020]:

$$\texttt{Valid-CFs} = \frac{|\{\text{unique instances in } \mathcal{C} \text{ s.t.} f(c) > 0.5\}|}{k} \tag{2.36}$$

where f is a trained machine learning model, and $c \in \mathcal{C}$.

Proximity [Mothilal et al., 2020] is defined as the mean of feature-wise distances between a counterfactual example and the original input, averaged across all samples in a set. For a counterfactual c_i from the set of counterfactuals $\mathcal{C} := \{c_1, c_2 \cdots c_k\}$, we have [Mothilal et al., 2020]:

$$\texttt{Continous-Proximity} = -\frac{1}{k}\sum_{i=1}^{k} \texttt{dist\_cont}(c_i, \mathbf{x}) \tag{2.37}$$

$$\texttt{Categorical-Proximity} = 1 - \frac{1}{k}\sum_{i=1}^{k} \texttt{dist\_cat}(c_i, \mathbf{x}) \tag{2.38}$$

where `dist_cont` is defined as [Mothilal et al., 2020]:

$$\texttt{dist\_cont}(c, \mathbf{x}) = \frac{1}{d_{cont}}\sum_{p=1}^{d_{cont}} \frac{|c^p - \mathbf{x}^p|}{\mathbf{MAD}_p} \tag{2.39}$$

where d_{cont} denotes the number of continuous variables and $\mathbf{MAD}_p$ defined from Eq. 2.35. The `dist_cat` is defined as [Mothilal et al., 2020]:

$$\texttt{dist\_cat}(c, \mathbf{x}) = \frac{1}{d_{cat}}\sum_{p=1}^{d_{cat}} \mathbb{1}_{c^p \neq \mathbf{x}^p} \tag{2.40}$$

where d_{cat} denotes the number of categorical variables.

Sparsity [Mothilal et al., 2020] is defined as the measure of the count of features that differ between the original input and a counterfactual.

$$\texttt{Sparsity} = 1 - \frac{1}{kd}\sum_{i=1}^{k}\sum_{l=1}^{d} \mathbb{1}_{c_i^l \neq \mathbf{x}_i^l} \tag{2.41}$$

where d denotes the number of input features.

Diversity [Mothilal et al., 2020] for a set of counterfactual examples is defined as the average distance between each pair of examples. Similar to proximity, separate diversity metrics are computed for categorical and continuous features.

$$\texttt{Diversity} = \frac{1}{C_k^2}\sum_{i=1}^{k-1}\sum_{j=i+1}^{k} \texttt{dist}(c_i, c_j) \tag{2.42}$$

where `dist` represents either `dist_cont` or dist_cat.

Figure 2.19.: An example of a visual counterfactual. Image sourced from [Augustin et al., 2022].

Finally, there has also been the introduction of visual counterfactuals [Augustin et al., 2022; Le et al., 2024], which perturb the class-relevant features. An example is provided in Figure 2.19. We will explore the image-text COCO counterfactuals in the upcoming sections.

3

Theory and Methods

In this section, we introduce the SAIF [Imtiaz et al., 2022] and APGD [Croce and Hein, 2020] adversarial attacks, along with the COCO counterfactuals [Le et al., 2024]. We assume C, H and W to be the number of color channels, height, and width of an image, respectively. We utilize p as described in Eq. 2.28 unless described otherwise.

3.1. Adversarial Attacks

In OpenFlamingo models, we follow [Schlarmann and Hein, 2023] and target only the images, leaving the text input unaltered and uncontaminated. From Eq. 2.28, the objective for untargeted attacks is to maximize the negative log-likelihood of text y over the threat model [Schlarmann and Hein, 2023]. Given a query image $q \in [0,1]^{C\times H\times W}$, ground truth caption y, N context images $c \in [0,1]^{N\times C\times H\times W}$, text z, query image perturbation budget $\epsilon_q \in \mathbb{R}$, $p_{\text{norm}} \geq 1$, and context images perturbation budget $\epsilon_c \in \mathbb{R}$, we define the loss function as:

$$\begin{aligned} \mathcal{L}(q, c, \delta_q, \delta_c, y) &:= -\sum_{l=1}^{m} \log p(y_l | y_{<l}, z, q+\delta_q, c+\delta_c) \\ \text{subject to} \quad & \|\delta_q\|_{p_{\text{norm}}} \leq \epsilon_q, \quad \|\delta_c\|_{p_{\text{norm}}} \leq \epsilon_c. \end{aligned} \tag{3.1}$$

where δ_c and δ_q denote the adversarial perturbation for the context images and query image, respectively [Schlarmann and Hein, 2023]. In the case of an **untargeted attack**, we perturb the images so that the textual output by the model is completely dissimilar to the original text, similar to Figure 3.1. The loss function for untargeted attacks can be formulated as follows:

$$\begin{aligned} \mathcal{L}(q, c, \delta_q, \delta_c, y)_{\text{untargeted}} &:= \max_{\delta_q, \delta_c} \mathcal{L}(q, c, \delta_q, \delta_c, y) \\ \text{s.t. } & \|\delta_q\|_{p_{\text{norm}}} \leq \epsilon_q, \|\delta_c\|_{p_{\text{norm}}} \leq \epsilon_c \end{aligned} \tag{3.2}$$

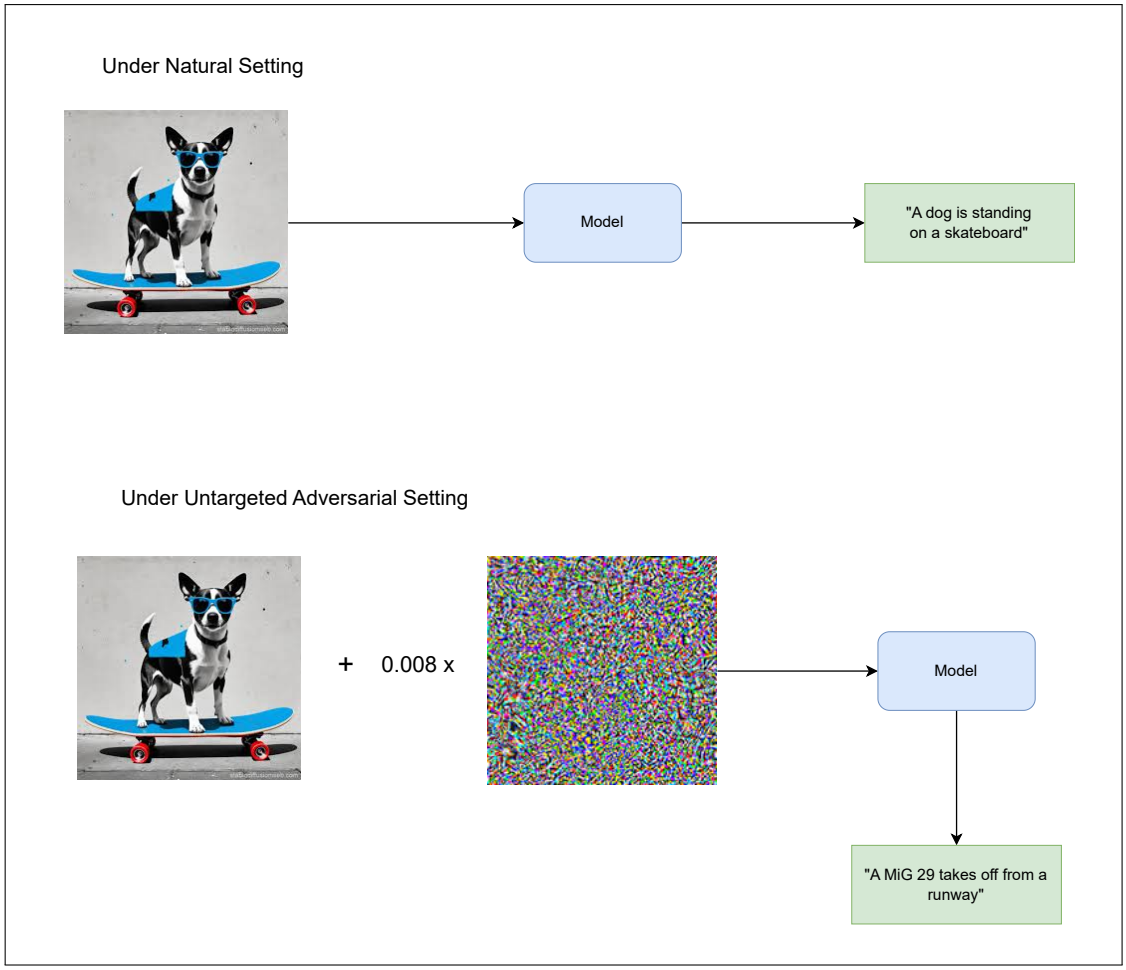


Figure 3.1.: An untargeted attack against a model. The disimilarity between the natural (or unperturbed) setting caption and adversarial (or perturbed) setting caption is quite visible. The image for the perturbation has been taken from `https://gradientscience.org/intro_adversarial/`.

For **targeted attacks**, the images are perturbed so that the textual data processed by the model exactly contains the targeted caption. As before, for targeted attacks, given a target caption $\hat{y}$, it follows:

$$\begin{aligned} \mathcal{L}(q, c, \delta_q, \delta_c, \hat{y})_{\text{targeted}} &:= \min_{\delta_q, \delta_c} \mathcal{L}(q, c, \delta_q, \delta_c, \hat{y}) \\ \text{s.t. } \|\delta_q\|_{p_{\text{norm}}} &\leq \epsilon_q, \|\delta_c\|_{p_{\text{norm}}} \leq \epsilon_c \end{aligned} \tag{3.3}$$

For clarity and ease of writing, we will denote δ_q and δ_c as δ as we consider each context and query images. We also denote $\epsilon := \epsilon_c = \epsilon_q$ as in our case the perturbation budget ϵ is the same for both the context images and query image. Thus, we denote $\mathcal{L}(\cdot, \delta, y)$ for untargeted attacks and $\mathcal{L}(\cdot, \delta, \hat{y})$ for targeted attacks. Also, let us denote the input as the concatenation of context and query images for N-shot learning as

$$\mathbf{x}_{\text{CQ}} = [c_1; c_2; \cdots; c_N; q] \tag{3.4}$$

where $c_1, c_2, \cdots, c_N, q \in [0,1]^{C \times H \times W}$ denote the N context images and the query image, respectively. So, $\mathbf{x}_{\text{CQ}} \in \mathcal{X} := [0,1]^{(N+1) \times C \times H \times W}$ and similarly, the perturbation can be written as $\delta := [\delta_{c_1}; \delta_{c_2}; \cdots; \delta_{c_N}; \delta_q] \in \mathbb{R}^{(N+1) \times C \times H \times W}$. Let us also denote $\mathbf{CQ} := \{i_{c_1}, i_{c_2}, \cdots i_{c_N}, i_q\}$ as the set of indices for the N context images and the query image. So, now the loss functions can be rewritten as

$$\mathcal{L}(\cdot, \delta, y) := -\sum_{l=1}^{m} \log p(y_l | y_{<l}, z, \mathbf{x}_{\text{CQ}} + \delta) \tag{3.5}$$

$$\begin{aligned} \mathcal{L}(\cdot, \delta, y)_{\text{untargeted}} &:= \max_{\delta} \mathcal{L}(\cdot, \delta, y) \\ \text{s.t. } \|\delta_i\|_{p_{\text{norm}}} &\leq \epsilon \ \forall i \in \mathbf{CQ} \end{aligned} \tag{3.6}$$

$$\begin{aligned} \mathcal{L}(\cdot, \delta, \hat{y})_{\text{targeted}} &:= \min_{\delta} \mathcal{L}(\cdot, \delta, \hat{y}) \\ \text{s.t. } \|\delta_i\|_{p_{\text{norm}}} &\leq \epsilon \ \forall i \in \mathbf{CQ} \end{aligned} \tag{3.7}$$

3.1.1. SAIF

The **Sparse Adversarial and Interpretable Framework** (SAIF), introduced in [Imtiaz et al., 2022], is utilized to create sparse adversarial images with low magnitude perturbation. The method consists of two integral parts, namely a sparsity-constrained mask $\mathbf{s} \in [0,1]^n$, where $n := (N+1) \times C \times H \times W$, and adversarial perturbation $\mathbf{p} \in \mathbb{R}^n$. Here, the adversarial perturbation δ is defined as $\delta := \mathbf{s} \odot \mathbf{p}$. Let ϵ be the perturbation budget for the ∞-norm of $\mathbf{p}$, and k be the 1-norm budget for $\mathbf{s}$. The formulation for the untargeted attack is given below.

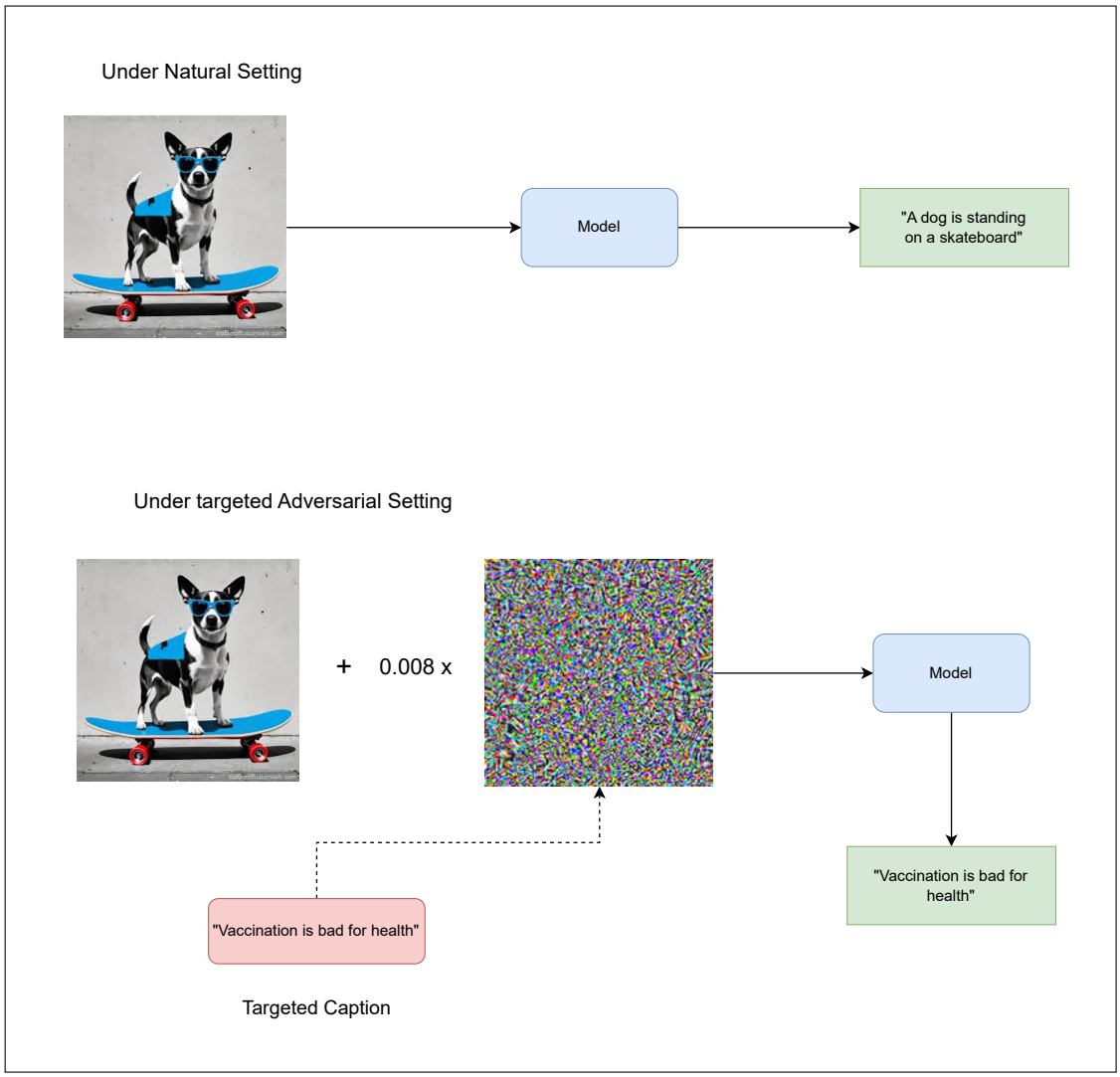


Figure 3.2.: An example of a targeted attack. The image was perturbed so that the target caption is exactly contained in the output caption. The image for the perturbation has been taken from `https://gradientscience.org/intro_adversarial/`.

$$\mathcal{L}(\cdot, \mathbf{s} \odot \mathbf{p}, y)_{\text{untargeted}} := \max_{\mathbf{s},\mathbf{p}} - \sum_{l=1}^{m} \log p(y_l | y_{<l}, z, \mathbf{x}_{\text{CQ}} + \mathbf{s} \odot \mathbf{p}) \\ \text{s.t. } \|\mathbf{p}\|_\infty \leq \epsilon, \|\mathbf{s}_i\|_1 \leq k \ \forall i \in \mathbf{CQ} \tag{3.8}$$

where $\mathbf{s}_i$ denotes the sparsity-constrained mask for ith image in $\mathbf{x}_{\text{CQ}}$. Here, we will also have to keep in mind the perturbation budget $(N+1)k$ for $\mathbf{s}$ because in the case of N-shot learning, we have N context images and a single query image, and the maximum 1-norm for the sparsity-constrained mask for each image can only be k. For instance, if we have 4-shot learning and the maximum budget for the 1-norm for each image is 1000, then $\|\mathbf{s}\|_1 \leq 5000$ as we have 4 context images and a query image.

$$\mathcal{L}(\cdot, \mathbf{s} \odot \mathbf{p}, \hat{y})_{\text{targeted}} := \min_{\mathbf{s},\mathbf{p}} - \sum_{l=1}^{m} \log p(\hat{y}_l | y_{<l}, z, \mathbf{x}_{\text{CQ}} + \mathbf{s} \odot \mathbf{p}) \\ \text{s.t. } \|\mathbf{p}\|_\infty \leq \epsilon, \|\mathbf{s}_i\|_1 \leq k \ \forall i \in \mathbf{CQ} \tag{3.9}$$

The SAIF algorithm works by employing the first-order, projection-free **Frank Wolfe** (FW) [Frank et al., 1956] algorithm for the purpose. For a convex and differentiable function f, the FW algorithm optimizes it over a convex set $\mathcal{C}$, as given in Algorithm 1. We say a set $\mathcal{C}$ is a **convex set** if $\forall\ x, y \in \mathcal{C}$, $\lambda x + (1-\lambda)y \in \mathcal{C}$ for all $\lambda \in [0, 1]$. We also say a function $f : \mathbb{R}^n \to \mathbb{R}$ is a **convex function** if its domain $\mathcal{C}$ is a convex set and $\forall x, y \in \mathcal{C}$, $f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y)$ for all $\lambda \in [0, 1]$.

FW has a convergence rate of $O(\frac{1}{T})$ for convex optimization [Jaggi, 2013], where T denotes the total number of iterations. In the case of locally solving non-convex objectives over convex regions, FW has a convergence rate of $O(\frac{1}{\sqrt{T}})$ [Imtiaz et al., 2022; Lacoste-Julien, 2016]. In Algorithm 1, $\langle \nabla f(\mathbf{x}^{(t)}), \mathbf{a} \rangle$ denotes the inner product between $\nabla f(\mathbf{x}^{(t)})$ and $\mathbf{a}$.

Algorithm 1 Frank-Wolfe Algorithm

Input Objective f, convex set $\mathcal{C}$, Maximum iterations T, stepsize rule $\eta^{(t)}$
Output Final iterate $\mathbf{x}^{(T)}$

$\mathbf{x}^{(0)} = 0$
for $t = 0 \ldots T-1$ **do**
  $\mathbf{a}^{(t)} = \arg\min_{\mathbf{a} \in \mathcal{C}} \langle \nabla f(\mathbf{x}^{(t)}), \mathbf{a} \rangle$
  $\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} + \eta^{(t)}(\mathbf{a}^{(t)} - \mathbf{x}^{(t)})$
end for
return $\mathbf{x}^{(T)}$

Algorithm 2 SAIF - Adversarial attack using Frank-Wolfe for joint optimization

Input: Input $\mathbf{x}_{\text{CQ}} \in \mathcal{X}$, target caption $\hat{y}$, loss function $\mathcal{L}$, iterations T_{iter}, $\mathbf{s}^{(0)} \in C_{\mathbf{s}} = \{\mathbf{s} \in [0,1]^n : \|\mathbf{s}_i\|_1 \leq k \ \forall i \in \mathbf{CQ}\}$, $\mathbf{p}^{(0)} \in C_{\mathbf{p}} = \{\mathbf{p} \in \mathbb{R}^n : \|\mathbf{p}\|_\infty \leq \epsilon\}$

Output: Perturbation $\mathbf{p}$, Sparse mask $\mathbf{s}$

Initialize $\mathbf{s}^{(0)} \in C_s, \mathbf{p}^{(0)} \in C_p$, $r \leftarrow 1$
for $t = 1, \ldots, T$ **do**
  $\mathbf{m_p}^{(t)} \leftarrow \nabla_{\mathbf{p}} \mathcal{L}(\cdot, \mathbf{s}^{(t-1)} \odot \mathbf{p}^{(t-1)}, \hat{y})$
  $\mathbf{m_s}^{(t)} \leftarrow \nabla_{\mathbf{s}} \mathcal{L}(\cdot, \mathbf{s}^{(t-1)} \odot \mathbf{p}^{(t-1)}, \hat{y})$
  $\mathbf{v}^{(t)} \leftarrow \arg\min_{\mathbf{v} \in C_{\mathbf{p}}} \langle \mathbf{m}_{\mathbf{p}}^{(t)}, \mathbf{v} \rangle$ as in Eq. 3.11
  $\mathbf{z}^{(t)} \leftarrow \arg\min_{\mathbf{z} \in C_{\mathbf{s}}} \langle \mathbf{m}_{\mathbf{s}}^{(t)}, \mathbf{z} \rangle$
  $L \leftarrow \mathcal{L}(\cdot, \mathbf{s}^{(t-1)} \odot \mathbf{p}^{(t-1)}, \hat{y})$
  $\eta^{(t)} \leftarrow \frac{1}{2^r \sqrt{t+1}}$
  while `primal_progress` is not True **do**
    $r \leftarrow r + 1$
    $\eta^{(t)} \leftarrow \frac{1}{2^r \sqrt{t+1}}$
  end while
  $\mathbf{p}^{(t)} \leftarrow \mathbf{p}^{(t-1)} + \eta^{(t)}(\mathbf{v}^{(t)} - \mathbf{p}^{(t-1)})$
  $\mathbf{s}^{(t)} \leftarrow \mathbf{s}^{(t-1)} + \eta^{(t)}(\mathbf{z}^{(t)} - \mathbf{s}^{(t-1)})$
end for
return $\mathbf{s}^{(T)} \odot \mathbf{p}^{(T)}$

In Algorithm 2, authors define `primal_progress` as

$$\texttt{primal\_progress} := L > \mathcal{L}\big(\cdot, (\mathbf{s}^{(t-1)} + \eta^{(t)}(\mathbf{z}^{(t)} - \mathbf{s}^{(t-1)})) \odot (\mathbf{p}^{(t-1)} + \eta^{(t)}(\mathbf{v}^{(t)} - \mathbf{p}^{(t-1)})), \hat{y}\big) \tag{3.10}$$

Given $\mathbf{m_p} := \nabla_{\mathbf{p}} \mathcal{L}(\cdot, \mathbf{s} \odot \mathbf{p}, \hat{y})$, the Linear Minimization Oracle (LMO) of the constraint $\|\mathbf{p}\|_\infty \leq \epsilon$ has a closed form solution [Chen et al., 2020b].

$$\arg\min_{\mathbf{v} \in C_{\mathbf{p}}} \mathbf{m}_{\mathbf{p}}^\top \mathbf{v} = -\epsilon \cdot \text{sign}(\mathbf{m_p}) \tag{3.11}$$

In the case of $\mathbf{s}$, SAIF uses non-negative k sparse polytope $C_{\mathbf{s}} = \{\mathbf{s} \in [0,1]^n : \|\mathbf{s}_i\|_1 \leq k \ \forall i \in \mathbf{CQ}\}$. Following the approach described in [Macdonald et al., 2021], SAIF performs LMO over this polytope. For $\mathbf{m_s} := \nabla_{\mathbf{s}} \mathcal{L}(\cdot, \mathbf{s} \odot \mathbf{p}, \hat{y})$, $\mathbf{m}_{\mathbf{s}_i} := \nabla_{\mathbf{s}_i} \mathcal{L}(\cdot, \mathbf{s} \odot \mathbf{p}, \hat{y}) \in \mathbb{R}^p \ \forall i \in \mathbf{CQ}, \mathbf{z} := [\mathbf{z}_i; \cdots] \in \{0,1\}^n \ \forall i \in \mathbf{CQ}$ and $p := C \times H \times W$, we choose a vector for $\mathbf{z}_i$ with at most k elements set to 1 in places where the $\mathbf{m}_{\mathbf{s}_i}$ has the k smallest negative values, and the rest of the elements set to 0 [Imtiaz et al., 2022]. This results in a vector with at most k non-zero values. This is done for all the indices $i \in \mathbf{CQ}$.

The authors assert that the algorithm achieves a high level of interpretability in the generated perturbations by separating the sparsity mask from the perturbation itself. The full attack is provided in Algorithm 2.

3.1.2. APGD

Auto Projected Gradient Descent attack (APGD), introduced in [Croce and Hein, 2020], is an improved version of the **Projected Gradient Descent** (PGD) [Madry et al., 2017] attack. Given a negative loss function f and a feasible set $\mathcal{X}$, PGD solves the problem of $\max_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$ by projecting the gradient ascent of $\mathbf{x}$ on the set $\mathcal{X}$ [Croce and Hein, 2020], thus respecting the constraints of set $\mathcal{X}$.

$$\mathbf{x}^{(t+1)} = P_{\mathcal{X}}\left(\mathbf{x}^{(t)} + \eta \cdot \operatorname{sign}(\nabla f_{\mathbf{x}^{(t)}}(\mathbf{x}^{(t)}))\right) \tag{3.12}$$

where $P_{\mathcal{X}}$ denotes the projection on set $\mathcal{X}$ and η denotes the learning rate. We use sign of gradient as we will be dealing with the ∞-norm threat model, in which, given original and adversarial sample $\mathbf{x}_{\text{orig}}, \mathbf{x}_{\text{adv}} \in \mathcal{X}$, we want $\|\mathbf{x}_{\text{adv}} - \mathbf{x}_{\text{orig}}\|_\infty \leq \epsilon$ for a given perturbation budget ϵ.

Algorithm 3 APGD - Auto Projected Gradient Descent

Input: Input $\mathbf{x}_{\text{CQ}} \in \mathcal{X}$, caption y, loss function $\mathcal{L}$ from Eq. 3.13, learning rate η, total iterations T_{iter}, checkpoints $W = \{w_0, w_1, \cdots, w_n\}$

Output: $\mathbf{x}_{\max}$, $\mathcal{L}_{\max}$

$\mathbf{x}^{(0)} \leftarrow \mathbf{x}_{\text{CQ}} + \mathbf{Z}$, where $\mathbf{Z} \sim \mathcal{U}(-\epsilon, \epsilon)$
$\mathbf{x}^{(1)} \leftarrow P_{\mathcal{X}}\left(\mathbf{x}^{(0)} + \eta \cdot \operatorname{sign}(\nabla_{\mathbf{x}^{(0)}} \mathcal{L}(\cdot, \mathbf{x}^{(0)} - \mathbf{x}_{\text{CQ}}, y))\right)$
$\mathcal{L}_{\max} \leftarrow \max\{\mathcal{L}(\cdot, \mathbf{x}^{(0)} - \mathbf{x}_{\text{CQ}}, y), \mathcal{L}(\cdot, \mathbf{x}^{(1)} - \mathbf{x}_{\text{CQ}}, y)\}$
$\mathbf{x}_{\max} \leftarrow \mathbf{x}^{(0)}$ **if** $\mathcal{L}_{\max} \equiv \mathcal{L}(\cdot, \mathbf{x}^{(0)}_{\text{CQ}} - \mathbf{x}_{\text{CQ}}, y)$ **else** $\mathbf{x}_{\max} \leftarrow \mathbf{x}^{(1)}$
for $k = 1$ **to** $T_{\text{iter}} - 1$ **do**
    $\mathbf{z}^{(k+1)} \leftarrow P_{\mathcal{X}}\left(\mathbf{x}^{(k)} + \eta \cdot \operatorname{sign}(\nabla_{\mathbf{x}^{(k)}} \mathcal{L}(\cdot, \mathbf{x}^{(k)} - \mathbf{x}_{\text{CQ}}, y))\right)$
    $\mathbf{x}^{(k+1)} \leftarrow P_{\mathcal{X}}\left(\mathbf{x}^{(k)} + \alpha(\mathbf{z}^{(k+1)} - \mathbf{x}^{(k)}) + (1-\alpha)(\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)})\right)$
    if $\mathcal{L}(\cdot, \mathbf{x}^{(k+1)} - \mathbf{x}_{\text{CQ}}, y) > \mathcal{L}_{\max}$ **then**
        $\mathbf{x}_{\max} \leftarrow \mathbf{x}^{(k+1)}$ and $\mathcal{L}_{\max} \leftarrow \mathcal{L}(\cdot, \mathbf{x}^{(k+1)} - \mathbf{x}_{\text{CQ}}, y)$
    end if
    if $k \in W$ **then**
        if Condition 1 **or** Condition 2 **then**
            $\eta \leftarrow \frac{\eta}{2}$ and $\mathbf{x}^{(k+1)} \leftarrow \mathbf{x}_{\max}$
        end if
    end if
end for
return $\mathbf{x}_{\max}, \mathcal{L}_{\max}$

Despite being computationally cheap and fast, the failure of PGD has also been observed, resulting in a substantial overestimation of robustness [Croce and Hein, 2020; Mosbach et al., 2018; Croce et al., 2020]. Specifically, the constant learning rate η and the choice of loss function (like the cross entropy loss) play a vital role in this potential failure [Croce and Hein, 2020]. The constant learning rate causes suboptimal performance as it cannot guarantee convergence, even in the

case of convex problems [Mosbach et al., 2018]. We do not discuss the choice of loss function as the issue is primarily for the task of image classification and, thus, is out of scope for this work.

To solve these problems, APGD removes the learning rate selection as a whole, replacing it with **exploitation** and **exploration** phases. The total number of iterations T_{iter} is divided into an initial exploration phase, followed by an exploitation phase. The exploration phase determines suitable initial points in the feasible set $\mathcal{X}$, after which the exploitation phase tries to maximize the knowledge accumulated so far [Croce and Hein, 2020]. The continuous diminution of the learning rate acts as a transition between the two phases. Thus, the choice of learning rate is well adapted to the progress of the optimization in the case of APGD, as opposed to PGD [Croce and Hein, 2020].

For APGD, the loss function for an untargeted attack [Schlarmann and Hein, 2023] is given as

$$\begin{aligned} \mathcal{L}(\cdot, \delta, y)_{\text{untargeted}} := \max_{\delta} -\sum_{l=1}^{m} \log p(y_l | y_{<l}, z, \mathbf{x}_{\text{CQ}} + \delta) \\ \text{s.t. } \|\delta\|_\infty \leq \epsilon \end{aligned} \tag{3.13}$$

where $\delta \in \mathbb{R}^n, n := (N+1) \times C \times H \times W$ is the adversarial perturbation, and ϵ is the perturbation budget. For a targeted attack with the targeted caption $\hat{y}$, it can be formulated as

$$\begin{aligned} \mathcal{L}(\cdot, \delta, \hat{y})_{\text{targeted}} := \min_{\delta} -\sum_{l=1}^{m} \log p(\hat{y}_l | y_{<l}, z, \mathbf{x}_{\text{CQ}} + \delta) \\ \text{s.t. } \|\delta\|_\infty \leq \epsilon \end{aligned} \tag{3.14}$$

The APGD update differs from the PGD algorithm because it incorporates a momentum term [Croce and Hein, 2020]. Let $\eta^{(k)}$ be the learning rate at iteration k. Then, the update can be written as:

$$\begin{aligned} \mathbf{z}^{(k+1)} &\leftarrow P_{\mathcal{X}} \left(\mathbf{x}^{(k)} + \eta^{(k)} \cdot \text{sign}(\nabla_{\mathbf{x}^{(k)}} \mathcal{L}(\cdot, \mathbf{x}^{(k)} - \mathbf{x}_{\text{CQ}}, y)) \right) \\ \mathbf{x}^{(k+1)} &\leftarrow P_{\mathcal{X}} \left(\mathbf{x}^{(k)} + \alpha(\mathbf{z}^{(k+1)} - \mathbf{x}^{(k)}) + (1-\alpha)(\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}) \right) \end{aligned} \tag{3.15}$$

where $\alpha \in [0, 1]$ controls the impact of the previous update on the current one.

The learning rate is initialised as $\eta^{(0)} = 2\epsilon$. Following this, *checkpoints* $w_0 = 0, w_1, \cdots, w_n$ are used by the APGD algorithm to determine if the current learning rate is to be halved, using the two conditions given in Eq. 3.16.

$$\begin{aligned} &1.\ \sum_{i=w_{j-1}}^{w_j-1} \mathbb{1}_{\mathcal{L}(\cdot,\mathbf{x}^{(i+1)}-\mathbf{x}_{\mathrm{CQ}},y)>\mathcal{L}(\cdot,\mathbf{x}^{(i)}-\mathbf{x}_{\mathrm{CQ}},y)} < \rho \cdot (w_j - w_{j-1}) \\ &2.\quad \eta^{(w_{j-1})} \equiv \eta^{(w_j)}, \quad \mathcal{L}_{\max}^{(w_{j-1})} \equiv \mathcal{L}_{\max}^{(w_j)} \end{aligned} \tag{3.16}$$

where $\mathcal{L}_{\max}^{(k)}$ denotes the highest value achieved in the first k iterations, and ρ is fixed to 0.75.

Condition 1 calculates the number of times the learning rate has been able to increase $\mathcal{L}$ since the last checkpoint w_{j-1}. **Condition 2** is in effect if there is no reduction in the learning rate and the best-found objective value has not improved since the last checkpoint. Upon the fulfillment of any of the conditions, the learning rate at iteration $k = w_j$ is halved, and $\eta^{(k)} := \frac{\eta^{w_j}}{2}$ for every $k = w_j + 1, \cdots, w_{j+1}$ [Croce and Hein, 2020].

The algorithm can also restart from the best point, in which, if at a checkpoint w_j the learning rate gets halved, then $\mathbf{x}^{(w_j+1)} := \mathbf{x}_{\max}$ thus restarting from the point achieving highest objective value $\mathcal{L}_{\max}$ so far [Croce and Hein, 2020].

The authors wanted a more initial exploration phase followed by quite frequent updates of the learning rate when venturing towards the exploitation phase. This allowed the utilization of the whole input space, and then the smaller learning rates helped reduce the objective function more frequently [Croce and Hein, 2020]. This is controlled by progressive reduction of the learning rate and by choosing when to decrease it, i.e,. the checkpoints w_j [Croce and Hein, 2020]. The checkpoints are set as $w_j = \lceil p_j T_{\mathrm{iter}} \rceil$, with $p_j \in [0, 1]$ provided as $p_0 = 0$, $p_1 = 0.22$ and

$$p_{j+1} = p_j + \max\{p_j - p_{j-1} - 0.03, 0.06\} \tag{3.17}$$

Finally, it is observable that the only hyper-parameter one has to set is the total number of iterations T_{iter}. The full attack is given in Algorithm 3. We now move on to counterfactuals.

3.2. Counterfactuals

In this section, we will look into COCO Counterfactuals.

3.2.1. COCO Counterfactuals

COCO Counterfactuals (COCO-CFs) [Le et al., 2024] is a synthetic multimodal counterfactual dataset consisting of paired images and text captions derived from the MS-COCO dataset [Lin et al., 2014]. The dataset is available at

`Intel/COCO-Counterfactuals` on `HuggingFace`[1].

To create a counterfactual caption C_c from an original caption C_o, the process involves altering a subject in C_o while retaining most of its original details. The modified subject acts as the altered causal feature, whereas the preserved details serve as potentially spurious correlated features [Le et al., 2024]. The following methodology is used:

1. **Identifying Candidate Words:** Nouns in C_o are identified as candidates for substitution using NLTK [Bird, 2006]. Each noun i from the set of n identified nouns becomes a substitution target [Le et al., 2024].

2. **Generating Substitutions:** For each noun, 10 candidate counterfactual captions are created by replacing the i-th noun in C_o with the `[MASK]` token [Le et al., 2024]. The RoBERTa model[2] [Liu, 2019] is then employed to generate the top-10 most probable replacements for the masked noun [Le et al., 2024].

3. **Filtering Candidates:** The generated $n \times 10$ counterfactual captions are filtered to retain only those where the substituted word is also a noun [Le et al., 2024].

4. **Ensuring Ontological Similarity:** To ensure that substitutions maintain an ontological similarity to the original noun, a pre-trained sentence similarity model [Wang et al., 2020] is used [Le et al., 2024]. Candidates with a sentence similarity score within the range of $(0.8, 0.91)$ relative to C_o are retained [Le et al., 2024].

5. **Selecting the Best Caption:** After filtering, GPT-2 [Radford et al., 2019] is used to score the perplexity [Jelinek et al., 1977] of the remaining candidates. The caption with the lowest perplexity is selected as the counterfactual caption C_c [Le et al., 2024].

Perplexity (PPL), as given above, is a metric that is often used for evaluating a language model's ability to predict the next word in a sequence [Jelinek et al., 1977; Fang et al., 2024]. Given a sequence of tokens $\mathbf{x} = (x_1, x_2, \cdots, x_n)$, PPL measures the inverse of the geometric mean of all token probabilities [Fang et al., 2024]:

$$\text{PPL}_\theta(\mathbf{x}) = \exp\left(-\frac{1}{n}\sum_{i=1}^{n}\log p_\theta(x_i|\mathbf{x}_{<i})\right) \tag{3.18}$$

[1]It is available at `https://huggingface.co/datasets/Intel/COCO-Counterfactuals`
[2]`RoBERTa-base` model is used.

Figure 3.3.: Examples of COCO-CFs generated with Prompt-to-Prompt (**right**) and without Prompt-to-Prompt (**left**). Images sourced from [Le et al., 2024].

where p_θ is defined in Eq. 2.25

After generating a counterfactual caption C_c, the next step involves creating synthetic images I_s^o and I_s^c from the original caption C_o and the counterfactual caption C_c, respectively. The goal is for I_s^o and I_s^c to differ only in the replaced noun, ensuring that the altered causal feature can be learned while retaining other unchanged details as potentially spurious features [Le et al., 2024].

It was also observed that text-to-image models often introduce significant variations in images, even with minor changes to text prompts [Le et al., 2024]. For example, replacing a single word in captions like *"A small child lounges with a **remote** in his hand"* with *"A small child lounges with a **toy** in his hand"* can unintentionally alter unrelated features such as facial expressions, posture, clothing, or background [Le et al., 2024]. To solve this, **Prompt-to-Prompt** [Hertz et al., 2022] technique is used. It ensures that only the relevant features are altered instead of most features.

It was observed that different types of image changes may require adjustments to the parameter p in Prompt-to-Prompt [Brooks et al., 2023; Le et al., 2024], which controls the number of denoising steps with shared attention weights. For instance, larger structural modifications often demand lower similarity between the resulting images and fewer shared attention weights [Le et al., 2024].

To address this, their (COCO-CFs) approach involves over-generating 100 image pairs using Prompt-to-Prompt, with p values randomly sampled from $\mathcal{U}(0.1, 0.9)$. These pairs are then filtered using CLIP [Radford et al., 2021] to retain those

with a minimum cosine similarity of 0.2 between the caption encoding and the corresponding generated image. From the remaining pairs, the best image pair (I_o^s, I_c^s) is selected based on directional similarity in CLIP space [Gal et al., 2022; Le et al., 2024].

$$\text{CLIP}_{dir} = \frac{\big(E_T(C_c) - E_T(C_o)\big) \cdot \big(E_I(I_c^s) - E_T(I_o^s)\big)}{\|E_T(C_c) - E_T(C_o)\|_2 \, \|E_I(I_c^s) - E_T(I_o^s)\|_2} \tag{3.19}$$

Here, E_T and E_I represent CLIP's text and image encoders, respectively. The CLIP_{dir} metric evaluates the consistency of changes between the two images (I_o^s, I_c^s) and their corresponding captions (C_o, C_s). Selecting image pairs with higher CLIP_{dir} scores enhances the quality of generated counterfactuals by ensuring greater alignment between the modifications in both textual and visual modalities [Le et al., 2024].

4 Computational Experiments

In this chapter, we provide details about the metrics, datasets, tasks and the setup we used for the experiments, along with the results for them.

4.1. Tasks and Datasets

We describe the datasets and perform the following tasks for the experiments.

4.1.1. Image Captioning

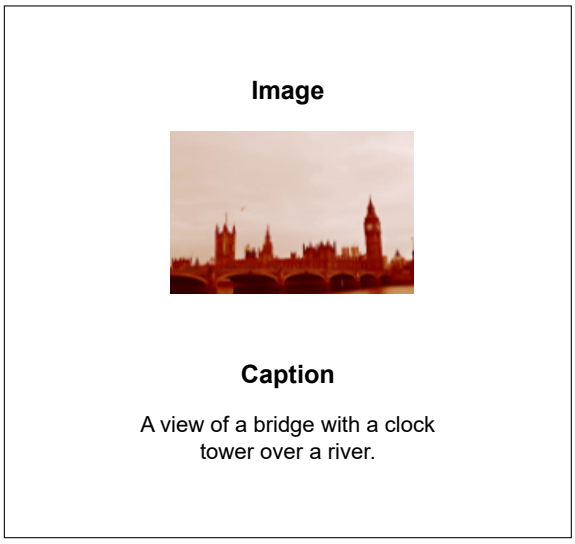


Figure 4.1.: An image-caption pair. Image sourced from [Ke et al., 2019].

Image Captioning is the task of delineating the visual content of an image in natural language [Stefanini et al., 2022]. Figure 4.2 shows an example of this.

Some popular datasets are **COCO** [Lin et al., 2014], **Flickr30k** [Plummer et al., 2015] and **Conceptual Captions** [Sharma et al., 2018].

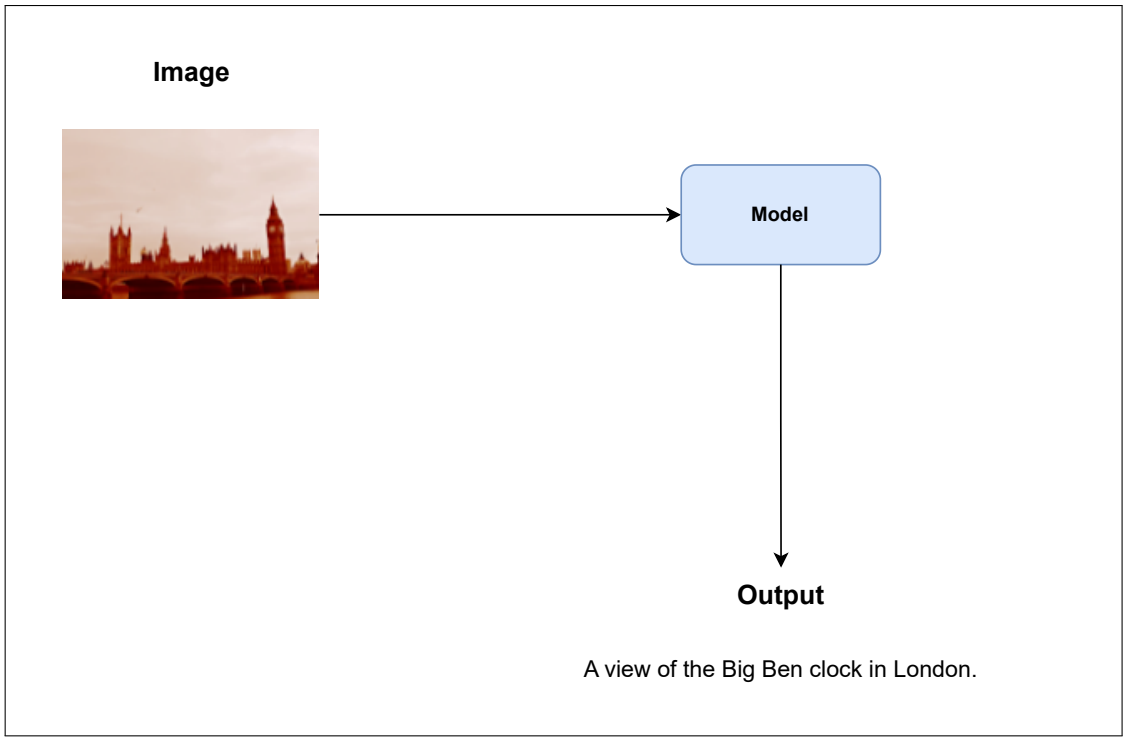


Figure 4.2.: An image captioning example. A model is provided an image and it outputs a caption. Image sourced from [Ke et al., 2019].

In our case, we will work with the **COCO** and **Flickr30k** datasets. **COCO** dataset is a widely used benchmark for tasks such as image segmentation, captioning, and object detection [Lin et al., 2014]. We use the COCO 2014 dataset (with a total of 123,287 images and 616,767 captions) for the image captioning task with the OpenFlamingo model and the COCO 2017 validation dataset (with a total of 5,000 images and 25,000 captions) for fine-tuning pre-trained CLIP models for comparison of counterfactuals with adversarial samples in the experiments section. Next is **Flickr30k**, another widely used dataset (with a total of 31,783 images and 158,915 captions) for image-text retrieval and image captioning tasks. We will use it for the image captioning task with the OpenFlamingo model and the image-text retrieval task with fine-tuned CLIP models.

4.1.2. Visual Question Answering

In **Visual Question Answering** (VQA) [Antol et al., 2015], a model takes an image along with an open-ended free-form question and provides an answer. **Open-ended questions** are questions that require the respondent to answer in a non-restrictive manner, like "*How*", "*Why*," and "*What*" [Antol et al., 2015]. An example of this can be seen in Figure 4.3. Some popular datasets are **OK-VQA** [Marino et al., 2019], **text-vqa** [Singh et al., 2019] and **VizWiz** [Gurari et al., 2018].

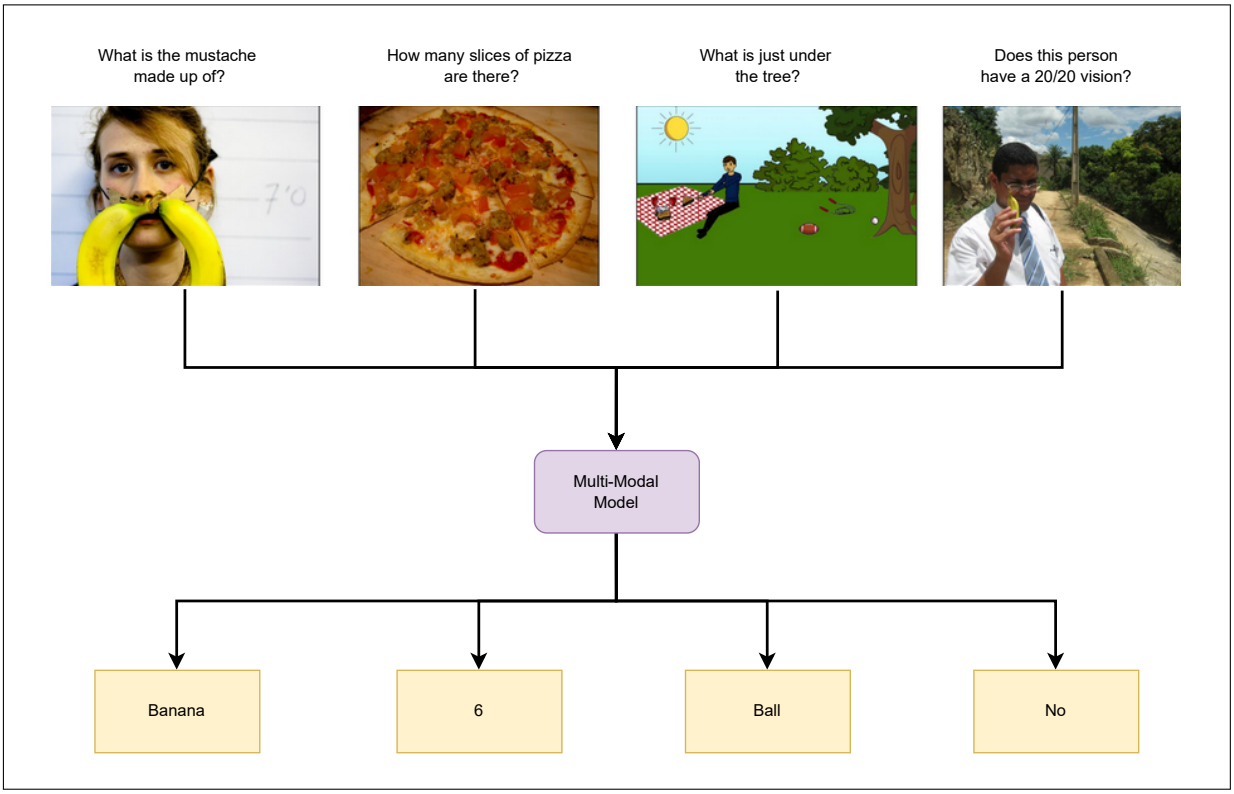


Figure 4.3.: A VQA example. Image sourced from [Antol et al., 2015].

For our purpose, we will be using the **OK-VQA** and **VizWiz** datasets. The VizWiz dataset was created to help develop artificial intelligence algorithms that answer blind people's questions [Gurari et al., 2018]. It consists of over 31,000 images and questions, with 10 crowd-sourced answers for each question. The OK-VQA dataset forces a model to take into account external knowledge resources via image-question pairs, in which the image content is inadequate to respond to the question [Marino et al., 2019]. It consists of over 14,000 images and questions, where each question has 5 human-provided answers. We use these datasets for the task of VQA for the OpenFlamingo model.

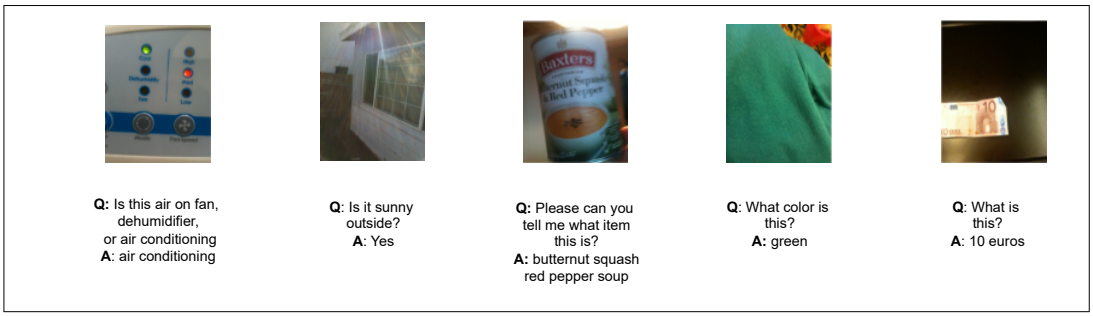


Figure 4.4.: An example from the VizWiz dataset. Image sourced from [Gurari et al., 2018].

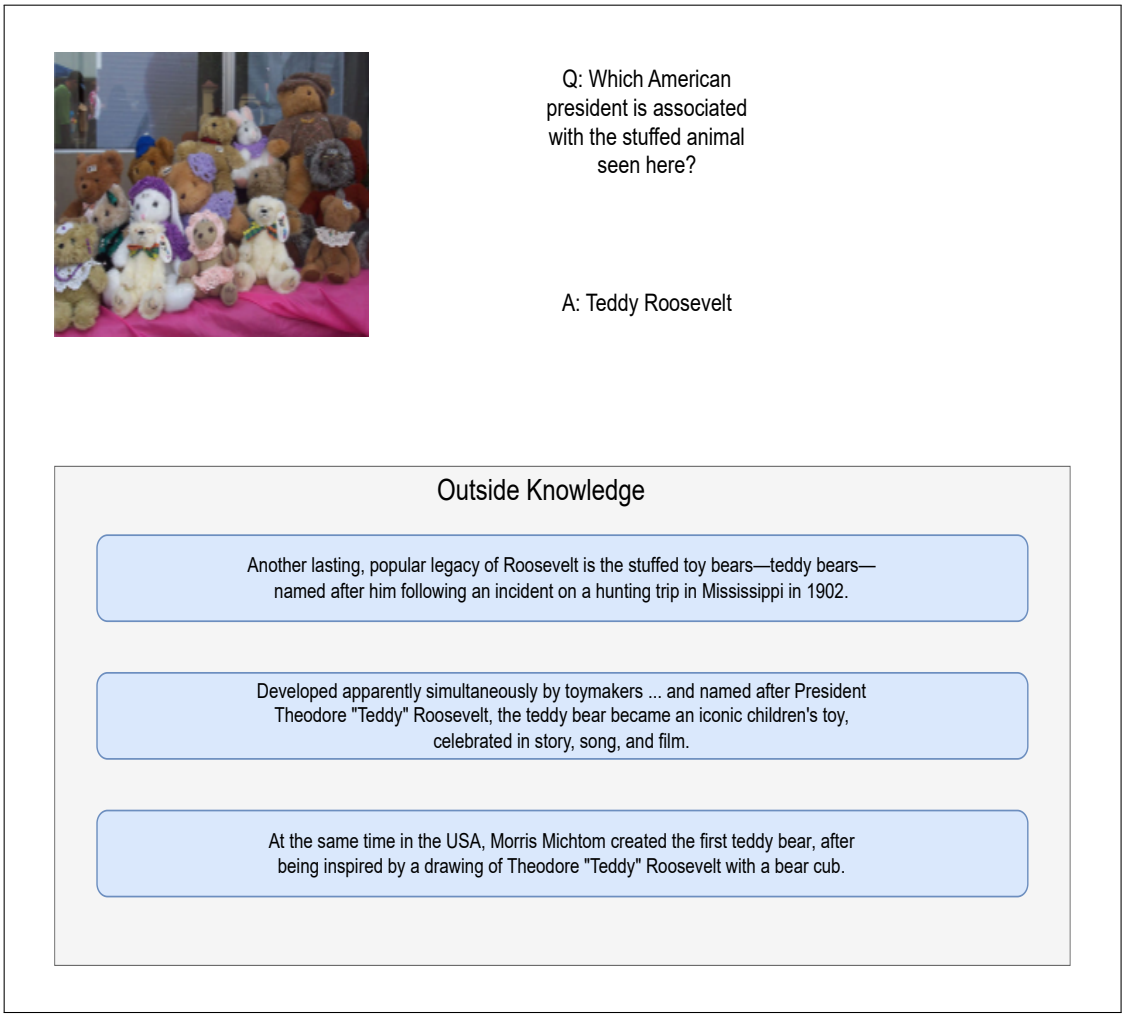


Figure 4.5.: An example from the OK-VQA dataset. Image sourced from [Marino et al., 2019].

4.1.3. Image-Text Retrieval

Image-text retrieval (ITR) is a type of cross-modal retrieval that involves retrieving relevant samples from one modality (images) based on a query expressed in another modality (texts) [Cao et al., 2022] and vice versa. It typically includes two subtasks: **image-to-text** (i2t) retrieval, where images are used to find matching text, and **text-to-image** (t2i) retrieval, where text is used to find corresponding images [Cao et al., 2022]. The widely used datasets are **Flickr30k** [Plummer et al., 2015] and **COCO** [Lin et al., 2014].

The **similarity matrix** is the cosine similarity between the embeddings of images and texts [Zhang et al., 2023]. We assume d to be the dimension size of the embeddings. Let $\mathbf{I} := [I_1; I_2; \cdots; I_n], \mathbf{I} \in \mathbb{R}^{n \times d}, I_i \in \mathbb{R}^d \ \forall \ i \in [n]$ denote the concatenation or "stacking" up of n image embeddings, where I_n denotes the embedding of the nth image, and let $\mathbf{T} := [T_1; T_2; \cdots; T_{\mathrm{m}}], \mathbf{T} \in \mathbb{R}^{m \times d}, T_i \in \mathbb{R}^d \ \forall \ i \in [m]$ denote the concatenation or "stacking" up of the text embeddings, where T_m denotes the embedding of the mth text, then the similarity matrix can be formulated as

$$\hat{\mathbf{I}} = \frac{\mathbf{I}}{\texttt{RLN}(\mathbf{I})} \quad \text{(Row-wise normalization)} \tag{4.1}$$

$$\hat{\mathbf{T}} = \frac{\mathbf{T}}{\texttt{RLN}(\mathbf{T})} \quad \text{(Row-wise normalization)} \tag{4.2}$$

$$\mathbf{S}^{\mathrm{i2t}} = \hat{\mathbf{I}}\hat{\mathbf{T}}^{\top} \tag{4.3}$$

and

$$\mathbf{S}^{\mathrm{t2i}} = \hat{\mathbf{T}}\hat{\mathbf{I}}^{\top} \tag{4.4}$$

where $\mathbf{S}^{\mathrm{i2t}} \in \mathbb{R}^{n \times m}$ denotes the similarity matrix for i2t, and $\mathbf{S}^{\mathrm{i2t}}_{nm}$ denotes the similarity score of the nth image with respect to mth text. Similarly, $\mathbf{S}^{\mathrm{t2i}} \in \mathbb{R}^{m \times n}$ denotes the similarity matrix for t2i, and $\mathbf{S}^{\mathrm{t2i}}_{mn}$ denotes the similarity score of the mth text with respect to the nth image.

In this work, we use the **Flickr30k** dataset for ITR for CLIP models fine-tuned with counterfactuals/adversarial attacks.

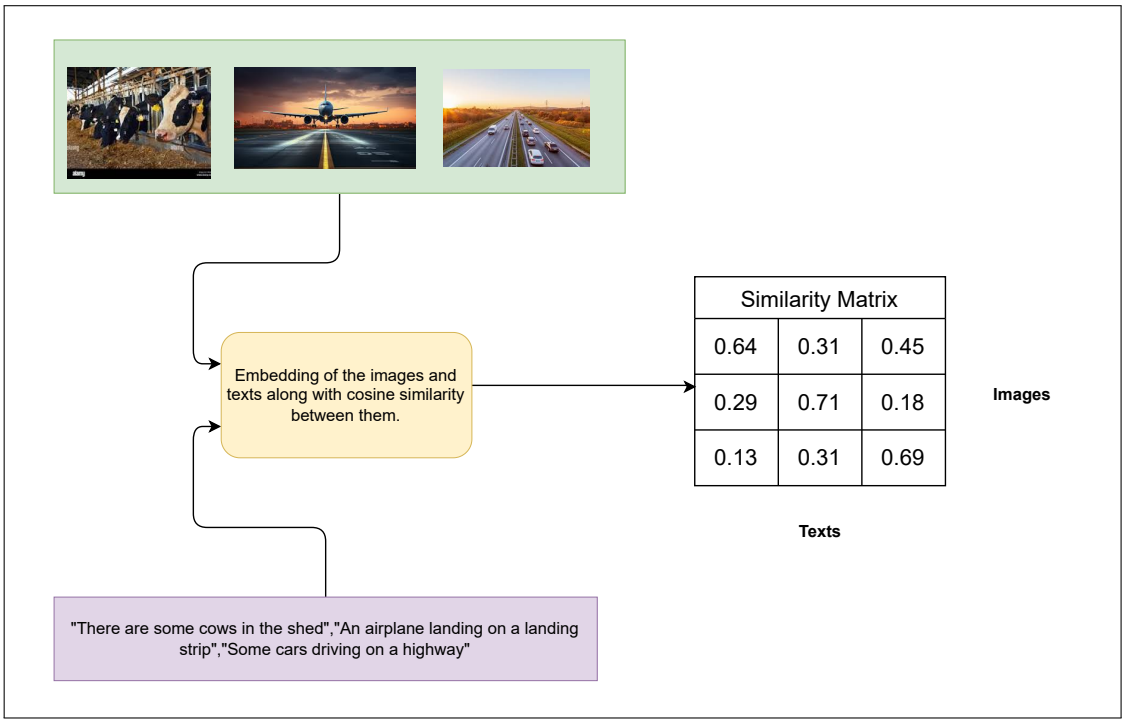


Figure 4.6.: An example of ITR. It can be seen that the similarity score between the image at the center and the second sentence "*An airplane landing on a landing strip*" is higher as compared to the third sentence "*Some cars driving on a highway*" since the cosine similarity of the embeddings of the former pair is higher as compared to the latter. All the numbers are random.

4.2. Metrics

We use the following metrics for the experiments.

4.2.1. CIDEr

Consensus-based Image Description Evaluation (CIDEr) score [Vedantam et al., 2015] is a metric that is used to measure how similar a generated caption is to a corpus of reference captions by analyzing the frequency of consecutive word sequences, adjusted using **Term Frequency Inverse Document Frequency** (TF-IDF) [Robertson, 2004] weighting [Schlarmann and Hein, 2023]. It is primarily used in the task of image captioning [Schlarmann and Hein, 2023]. The CIDEr score has a worst value of 0 and no fixed upper limit [Schlarmann and Hein, 2023].

Given an image $I_i \in I$, where I is a set of images from a dataset, a generated

caption c_i and a set of reference captions $S = \{s_{i1}, \cdots, s_{im}\}$, each sentence is represented via the set of n-grams present in it [Vedantam et al., 2015]. A consensus measure favors frequent n-gram matches with references, penalizes non-matches, and down-weights common dataset n-grams [Vedantam et al., 2015]. This intuition is encoded using TF-IDF weighting for each n-gram [Vedantam et al., 2015].

Let the number of times an n-gram ω_k is present in a reference sentence s_{ij} is denoted by $h_k(s_i)$ or $h_k(c_{ij})$ for the candidate sentence c_i. Then, the TF-IDF weighting, denoted by $g_k(s_{ij})$, is calculated as follows [Vedantam et al., 2015]:

$$g_k(s_{ij}) = \frac{h_k(s_{ij})}{\sum\limits_{\omega_l \in \Omega} h_l(s_{ij})} \log \left(\frac{|I|}{\sum\limits_{I_p \in I} \min\left(1, \sum\limits_q h_k(s_{pq})\right)} \right) \tag{4.5}$$

where Ω is the vocabulary of all n-grams.

It is observed that Eq. 4.5 combines Term Frequency (TF) and Inverse Document Frequency (IDF) to assess the importance of an n-gram ω_k in describing an image. The TF component weights n-grams that frequently occur in a specific reference sentence, while the IDF component reduces the importance of common n-grams that appear across many images, highlighting unique, visually informative words [Vedantam et al., 2015]. The CIDEr score calculates the relevance of n-grams of length n by averaging the cosine similarity between the candidate sentence and the reference sentences [Vedantam et al., 2015].

$$\text{CIDEr}_n(c_i, S_i) = \frac{1}{m} \sum_j \frac{\mathbf{g^n}(c_i) \cdot \mathbf{g^n}(s_{ij})}{\|\mathbf{g^n}(c_i)\|_2 \cdot \|\mathbf{g^n}(s_{ij})\|_2} \tag{4.6}$$

where $\mathbf{g^n}(c_i)$ is a vector created by $g_k(c_i)$ in accord with all n-grams of length n and $\|\mathbf{g^n}(c_i)\|_2$ denotes the magnitude of the vector $\mathbf{g^n}(c_i)$. Similar thing also occurs for $\mathbf{g^n}(s_i)$ [Vedantam et al., 2015]. The scores from n-grams of different lengths are then combined in the following manner [Vedantam et al., 2015].

$$\text{CIDEr}(c_i, S_i) = \sum_{n=1}^{N} w_n \text{CIDEr}_n(c_i, S_i) \tag{4.7}$$

where $w_n = \frac{1}{N}$ denotes uniform weights and N is usually kept at 4.

4.2.2. BLEU

Bilingual Evaluation Understudy [Papineni et al., 2002] (BLEU) is a widely used metric for evaluating machine translation, focusing on the matches of n-grams between the candidate translation and reference sentences. The range of

BLEU score ranges from 0 to 100[1] [Schlarmann and Hein, 2023]. The BLEU score is calculated as follows [Vedantam et al., 2015]:

$$P_n(c_i, S_i) = \frac{\sum_k \min(h_k(c_i), \max_{j \in m} h_k(s_{ij}))}{\sum_k h_k(c_i)} \tag{4.8}$$

where k indexes the set of possible n-grams of length n, and rest of the notations are from Subsection 4.2.1 [Vedantam et al., 2015]. The clipped precision metric restricts the counting of an n-gram to its highest frequency observed in a single reference sentence [Vedantam et al., 2015]. A brevity penalty is also applied due to the preference of P_n for shorter sentences [Papineni et al., 2002].

$$b(C, S) = \begin{cases} 1, & \text{if } l_C > l_S \\ e^{1-\frac{l_S}{l_C}}, & \text{if } l_C \leq l_S \end{cases} \tag{4.9}$$

where l_C and l_S denote the length of the candidate sentence c_i and the corpus-level effective reference length. In the case of multiple references for a candidate sentence, the *closest* reference length is chosen [Vedantam et al., 2015].

The overall BLEU score is calculated using a weighted geometric mean of the individual n-gram precision scores [Papineni et al., 2002; Vedantam et al., 2015].

$$\text{BLEU}_N(c_i, S_i) = b(c_i, S_i) \exp\left(\sum_{n=1}^{N} w_n \log P_n(c_i, S_i)\right) \tag{4.10}$$

where $N = 4$, with which, we get the BLEU-4 score.

4.2.3. VQA Accuracy

Visual Question Answering Accuracy [Antol et al., 2015] (VQA Accuracy) is a metric primarily used for the task of VQA. In VQA, two modalities are provided for answering questions during testing, namely, **open-answer** and **multiple-choices** [Antol et al., 2015]. We will be focusing on the former modality.

Open-ended (**open-answer**) **questions** are questions that lead to a wide variety of potential answers [Antol et al., 2015]. To measure the accuracy given a preprocessed[2] answer by the model, the following formulation is used [Antol et al., 2015]:

[1] Normally, the range of BLEU score is between 0 and 1, but we multiply it by 100 to get percentage.

[2] The answer is made lowercase, numbers are converted to digits, and punctuations & articles removed [Antol et al., 2015].

$$\text{accuracy} = \min\left(\frac{\text{number of humans that gave that answer}}{3}, 1\right) \tag{4.11}$$

where an answer is deemed completely (100%) correct if it matches with at least 3 humans that provided that same answer [Antol et al., 2015].

4.2.4. Attack Success Rate

Attack Success Rate (ASR) is defined as the percentage of the number of successful attacks out of the total number of attacks conducted. Let n be the total number of attacks conducted and m be the number of successful attacks conducted. Then, ASR[3] is formulated as follows:

$$\text{ASR} = \frac{m}{n} \tag{4.12}$$

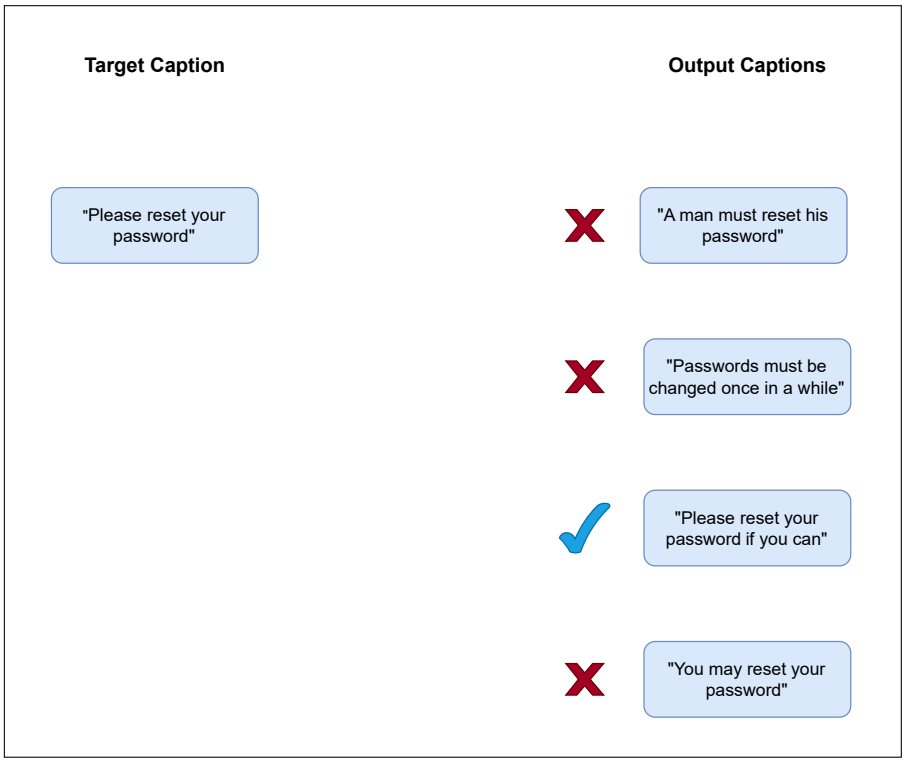


Figure 4.7.: An illustration of how the ASR works in our case. It can be seen that only the third output caption exactly contains the target caption and hence, is the only successful targeted attack.

[3]We multiply it by 100 to get percentage.

Following [Schlarmann and Hein, 2023], we say a targeted attack is successful if the output caption provided by the targeted model exactly contains[4] the target caption. Figure 4.7 gives an example of this.

4.2.5. Recall@k

Recall@k (R@k) metric is used to measure image-text retrieval [Cao et al., 2022]. It is defined as

$$\textbf{R@k} = \frac{\text{Number of queries with relevant item in top-k results}}{\text{Total number of queries}} \tag{4.13}$$

Here, **queries** are defined as an input provided by the user to search for relevant items within a dataset. For instance, if the task is **i2t**, then the images would be the queries and vice-versa. Let $S \in \mathbb{R}^{m \times n}$ be a similarity matrix computed using Eq. 4.4, then for each query $i \in [m]$, we have **Top-k** results as

$$\text{Top-k}_i = \texttt{ArgSort}(S_{i,:})[-\text{k} :] \tag{4.14}$$

and

$$\textbf{R@k} = \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}_{g(i) \in \text{Top-k}_i} \tag{4.15}$$

where $g(i)$ is the ground truth index for query i, and `Argsort` denotes the the argsort function that, for a given vector, provides the indices that would sort that vector in an ascending manner.

Typical values used for k are 1, 5, and 10. For i2t, given an image, **R@1** would be defined as the proportion of queries in which its corresponding text was among the top-1 ranks. Similarly, for t2i, given a text, **R@10** would be defined as the proportion of queries in which its corresponding image was among the top-10 ranks. We also multiply it by 100 to get the R@k percentage.

4.3. Setup

Sparse versus Non-Sparse Attacks on OpenFlamingo. For evaluating the adversarial robustness of a VLM, we utilize the COCO and the Flickr30k datasets for image captioning tasks, OK-VQA, and VizWiz for visual question-answering tasks. Following [Schlarmann and Hein, 2023], we sample 800 instances from each dataset and use the current strongest model from the OpenFlamingo family with 9 billion parameters. The model integrates a CLIP [Radford et al., 2021] vision encoder based on a ViT-L-14 vision transformer [Dosovitskiy et al., 2020] with an

[4]exactly contains condition is case-insensitive.

MPT-7B large language model [Team, 2023]. The **targeted** and **untargeted** attacks, or settings, are conducted under 2 sub-settings: 0-shot and 4-shot learning. For 4-shot learning, there are two modes: query and all. Under the **query** mode, only the query image is perturbed, leaving the context images untouched. For the **all** mode, all the images, including context and query, are perturbed. In the case of the untargeted setting, we employ the CIDEr score and VQA Accuracy for the task of image captioning and visual question answering, respectively. For the targeted setting, we employ the BLEU-4 score and ASR. The experimental details are given in the Appendix A.1.1.

Fine-tuning of CLIP Models on Adversarial Attacks versus Counterfactuals. For the comparison of fine-tuning with counterfactuals as opposed to adversarial attacks, we use pre-trained CLIP [Radford et al., 2021] models. Following [Le et al., 2024], we create four datasets of varying size[5] out of the validation set of COCO 2017 [Lin et al., 2014] and COCO-CFs/APGD attacks. The pre-trained CLIP models are then fine-tuned on these four datasets for every method (counterfactuals and adversarial samples). The adversarial samples are created using the targeted APGD attack on the 9-billion parameters model from the OpenFlamingo family for the COCO-CFs dataset. We didn't use the SAIF attack due to its lengthy computational time. We then conduct 0-shot image classification on the **Food101** [Bossard et al., 2014], **Caltech101** [Fei-Fei et al., 2004], **Caltech256** [Griffin et al., 2007], **CIFAR10** [Krizhevsky, 2009], **CIFAR100** [Krizhevsky, 2009] and the **ImageNet** [Deng et al., 2009] datasets from the Torchvision library [maintainers and contributors, 2016] using the fine-tuned CLIP models. We also perform image-text retrieval tasks on the **Flickr30k** dataset using the fine-tuned CLIP models. The details of the creation of the four datasets, fine-tuning details of the pre-trained CLIP models, and attack hyperparameters are given in Appendix A.1.2.

We use the Pytorch [Paszke et al., 2019], OpenFlamingo [Awadalla et al., 2023], and Transformers [Wolf et al., 2020] libraries for the experiments. The code is adapted from [Schlarmann and Hein, 2023; Radford et al., 2021]. All experiments were performed on machines with NVIDIA A40 and A100 GPUs.

[5]Varying in terms of number of data samples.

4.4. Results

In this section, we discuss the results of the experiments. We rely on figures and tables to interpret the results. Comprehensive tables representing the values of the metrics introduced in Section 4.2 can be found in Appendix A.2.

Sparse versus Non-Sparse Attacks on OpenFlamingo. Tables A.1 and A.2 provide the results for the attacks conducted under the untargeted setting using APGD and SAIF attacks respectively. The **bold** values highlight the lowest metric score across all hyper-parameters. Similar to [Schlarmann and Hein, 2023], there is a general decline in all of the metrics in the adversarial part of the tables, which can be expected due to the perturbed input images. The APGD attack, for perturbation budget $\epsilon \in \{\frac{1}{255}, \frac{4}{255}\}$, outperforms SAIF in all of the tasks.

Figures 4.8 - 4.10 display the stark difference between the original captions and the adversarial captions generated by the model. It is to be noted that original images are being displayed instead of adversarial images as the perturbation budget used for APGD renders perturbation nearly invisible, and for SAIF, the budget for ϵ is extremely high and would "distort" the images. It can also be observed that the stronger the attack [6], the more dissimilar the adversarial caption is to the original caption. Additionally, one can also notice an "anomaly" in the case of APGD ($\epsilon = \frac{4}{255}$) in the Figure 4.9 (the bottom left image), in which the model produces an extremely "bizarre" adversarial caption. This, however, doesn't happen in the case of APGD (with $\epsilon = \frac{1}{255}$) and SAIF, even though SAIF has a significantly higher perturbation budget of $\epsilon = \frac{255}{255}$. We hypothesize it to be the highly sparse nature of SAIF that restricts the number of perturbed pixels, thus leading to a dissimilar but comprehensible caption. For APGD ($\epsilon = \frac{1}{255}$), the reduced perturbation budget could be a potential reason.

Figure 4.11 shows that only a proportion of pixelwise perturbations are required for a substantial attack, at least in the untargeted setting, where roughly top 60% pixel-wise perturbations are enough for a significant drop in the CIDEr score on the COCO dataset for APGD attacks, similar to [Schlarmann and Hein, 2023]. Interestingly, SAIF outperforms APGD until the perturbation factor of 0.3, after which APGD surpasses it. Again, we believe it to be the sparse nature of SAIF, along with the high perturbation budget, that allows it to be more effective in the initial levels of perturbation factor as less number of perturbed pixels are considered during those levels. However, once it goes beyond 0.3, the non-sparse nature of APGD enables it to consider a higher number of perturbed pixels compared to its sparse counterpart, allowing for more perturbation of the image and, thus, gaining superior performance over SAIF.

[6] An attack with higher perturbation budget ϵ or is non-sparse or both.

For the targeted setting, Tables A.3 and A.4 showcase the results. The **bold** values highlight the highest ASR and BLEU-4 scores across all hyper-parameters. Again, the APGD attack outperforms the SAIF attack on nearly all tasks. Moreover, it can also be observed that longer target captions require more iterations or a higher perturbation budget ϵ or both, similar to [Schlarmann and Hein, 2023]. Finally, in Figure 4.11, it can be noticed that a higher fraction of perturbed pixels is required to get a notable rise in ASR. Again, SAIF outperforms APGD up until the perturbation factor of 0.5, after which APGD surpasses it.

Fine-tuning of CLIP Models on Adversarial Attacks versus Counterfactuals. Tables A.5, A.6 and A.7 display the results for performance on the task of 0-shot image classification of the pre-trained CLIP models fine-tuned on COCO-CFs versus APGD attacks. The main numbers represent the mean across multiple seeds, while the subscripted values indicate the corresponding standard deviations. The **bold** numbers denote the highest value in all 4 fine-tuning datasets, whereas **bold** and underlined numbers indicate the highest value achieved in the fine-tuning datasets containing adversarial attacks or counterfactuals. For the Caltech101, Caltech256, and CIFAR100 datasets, the 0-shot image classification for models fine-tuned on COCO-CFs marginally improves against the baseline[7] as compared to the ones fine-tuned on APGD attacks. In the APGD attack, for $\epsilon \in \{\frac{1}{255}, \frac{4}{255}\}$, performance scarcely deteriorates across all image classification datasets when compared to baseline. We hypothesize it to be the meaningful changes introduced by counterfactuals that reduce spurious correlations in the fine-tuned model's prediction, leading to marginally superior performance over fine-tuning with adversarial samples in the task.

We also showcase the difference between a counterfactual and an adversarial image via the Figure 4.12. It can be seen that the counterfactual image has a meaningful difference from the original image. The addition of a car and the removal of the pole in the counterfactual image displays relevant changes to the original image to achieve the counterfactual caption. In contrast, the adversarial attack is indistinguishable from the original image, thereby admitting to malicious, imperceptible changes to the original image to achieve the targeted caption.

Tables A.8, A.9 and A.10 shine a light on the difference in the performance of fine-tuned CLIP models on COCO-CFs versus APGD attacks for the task of image-text retrieval. The main numbers represent the mean across multiple seeds, while the subscripted values indicate the corresponding standard deviations. Further details are given in the Appendix A.1.2. The **bold** numbers denote the highest value across all 4 fine-tuning datasets, whereas **bold** and underlined numbers

[7]The performance of non fine-tuned pre-trained CLIP model.

indicate the highest value achieved across the datasets containing adversarial attacks or counterfactuals. It can be seen in both of the cases that the i2t and t2i improved notably across all of the four fine-tuning datasets. Finally, models fine-tuned on COCO-CFs marginally outperform the APGD attacks in all recalls in both sub-tasks, i2t and t2i. Again, we believe it to be the relevant changes introduced by counterfactuals that lead to marginally better performance over fine-tuning with adversarial samples in the task.

Original: A crowded street has a number of street signs.

Adversarial: An indoor play area for children.

Original: Tennis player and white outfit holding up a racket and a ball.

Adversarial: A shirtless man is holding his chest.

Original: A living room with hard wood floors filled with furniture.

Adversarial: A leprechaun with a pot of gold.

Original: A kitchen counter with a rounded edge and shelves.

Adversarial: A sink with a sink-hole.

Original:A busted bathroom sink underneath a mirror in a bathroom.

Adversarial: a box of jewelry with a red ribbon around it.

Original: A women in yellow is hitting a tennis ball on a clay court.

Adversarial: A couple of tennis players that are hodling a net.

Figure 4.8.: Generated captions on original and *adversarially perturbed* COCO images. The perturbations are obtained using **APGD** *untargeted attack* using the perturbation budget $\epsilon = \frac{1}{255}$ and 500 iterations in the 0-shot sub-setting.

Original: A crowded street has a number of street signs.

Adversarial: A tent set up in the shape of a tent with tent pegs holding it down.

Original: Tennis player and white outfit holding up a racket and a ball.

Adversarial: Juan Manuel Marquez after a fight.

Original: A living room with hard wood floors filled with furniture.

Adversarial: A man holding a hand full of manatees.

Original: A kitchen counter with a rounded edge and shelves.

Adversarial: sinkhole in sink.

Original:A busted bathroom sink underneath a mirror in a bathroom.

Adversarial: Suitcase suitcase

Original: A women in yellow is hitting a tennis ball on a clay court.

Adversarial: A collage of pie, pie, pie and pie

Figure 4.9.: Generated captions on *original* and *adversarially perturbed* COCO images. The perturbations are obtained using **APGD** *untargeted attack* for the **untargeted** setting with $\epsilon = \frac{4}{255}$ and 500 iterations on the 0-shot sub-setting.

Original: A crowded street has a number of street signs.

Adversarial: Two road signs on the side of a street in Malaysia.

Original: Tennis player and white outfit holding up a racket and a ball.

Adversarial: Marin Cilic of Croatia wipes sweat from his face during his men's singles match against

Original: A living room with hard wood floors filled with furniture.

Adversarial: a cat playing with a bag of confetti.

Original: A kitchen counter with a rounded edge and shelves.

Adversarial: Hand prints on kitchen cupboards.

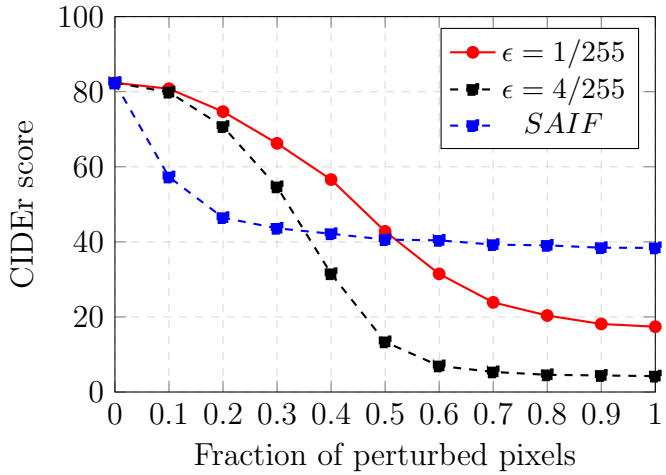
Original:A busted bathroom sink underneath a mirror in a bathroom.

Adversarial: A woman with a box of clothes

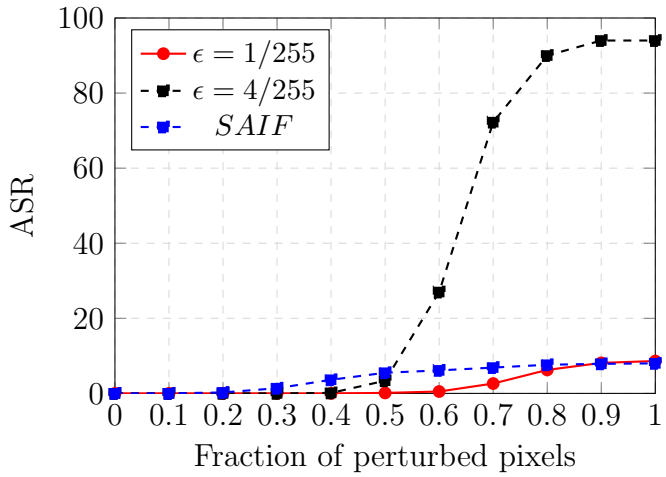
Original: A woman in yellow is hitting a tennis ball on a clay court.

Adversarial: A tennis player hits a ball with a racket.

Figure 4.10.: Generated captions on *original* and *adversarially perturbed* COCO images. The perturbations are obtained with **SAIF** *untargeted attack* with $k = 1000$, $\epsilon = \frac{255}{255}$ and 500 iterations in the 0-shot subsetting.



(a) Untargeted attack



(b) Targeted attack

Figure 4.11.: Effect on the performance if only a fraction of the perturbations are used. We zero out the perturbations that are smallest in magnitude and report (a) the resulting CIDEr score on COCO for *untargeted attacks* and (b) the attack success rate for the *targeted attacks* with target caption "*Please reset your password*". The perturbations are obtained with **SAIF** attack with $k = 1000$, $\epsilon = \frac{255}{255}$, and **APGD** attack with perturbation budget $\epsilon \in \{\frac{1}{255}, \frac{4}{255}\}$. The attacks were performed for 100 iterations in the 0-shot sub-setting.

Figure 4.12.: A visual representation to elucidate the difference between a counterfactual image and an adversarial sample. The original image (**leftmost**) and its counterfactual counterpart (**rightmost**) have been taken from the **COCO-CFs** dataset. The adversarial image (**middle**) was crafted by attacking the 9-billion parameter OpenFlamingo model with the **APGD** attack with $\epsilon = \frac{4}{255}$ under the targeted setting with 0-shot sub-setting for the targeted caption "*A rusted silver fire hydrant next to two* ***cars***." with 100 epochs. The original caption was "*A rusted silver fire hydrant next to two* ***poles***.". The counterfactual caption was chosen as the target caption.

5 Conclusion and Future Work

Multi-modal foundation models, especially vision-language models, have recently gained significant attention. However, they have also been shown to be vulnerable to adversarial attacks. In this work, we have examined the susceptibility of these models against the non-sparse APGD and sparse SAIF adversarial attacks in the context of image captioning and visual question answering, across multiple datasets on the OpenFlamingo family of models. We also compared the results of fine-tuning pre-trained CLIP models on the aforementioned APGD adversarial attacks and COCO-CFs in image classification and image-text retrieval. In the experiments, we saw that the APGD attack outperformed SAIF in nearly every in-context metric, and we also showcased that a fraction of perturbed pixels is more than enough to disorient the output generated by the OpenFlamingo model completely. We also emphasize the malicious nature of adversarial attacks as opposed to meaningful changes done by counterfactuals, and observed that the CLIP models fine-tuned on COCO-CFs performed marginally better than the one fine-tuned on APGD attacks on various tasks. Finally, we provide the implementations for all the adversarial attacks and counterfactuals discussed in this work in the supplementary material.

Bibliography

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35:23716–23736, 2022.

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*, 2023.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433, 2015.

Ken Arnold, James Gosling, and David Holmes. *The Java programming language*. Addison Wesley Professional, 2005.

Maximilian Augustin, Valentyn Boreiko, Francesco Croce, and Matthias Hein. Diffusion visual counterfactual explanations. *Advances in Neural Information Processing Systems*, 35:364–377, 2022.

Anas Awadalla, Irena Gao, Josh Gardner, Jack Hessel, Yusuf Hanafy, Wanrong Zhu, Kalyani Marathe, Yonatan Bitton, Samir Gadre, Shiori Sagawa, et al. Openflamingo: An open-source framework for training large autoregressive vision-language models. *arXiv preprint arXiv:2308.01390*, 2023.

Jimmy Lei Ba. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Alexei Baevski and Michael Auli. Adaptive input representations for neural language modeling. *arXiv preprint arXiv:1809.10853*, 2018.

Dzmitry Bahdanau. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

Steven Bird. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, pages 69–72, 2006.

Florian Bordes, Richard Yuanzhe Pang, Anurag Ajay, Alexander C Li, Adrien Bardes, Suzanne Petryk, Oscar Mañas, Zhiqiu Lin, Anas Mahmoud, Bargav Jayaraman, et al. An introduction to vision-language modeling. *arXiv preprint arXiv:2405.17247*, 2024.

Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In *European Conference on Computer Vision*, 2014.

Tim Brooks, Aleksander Holynski, and Alexei A Efros. Instructpix2pix: Learning to follow image editing instructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18392–18402, 2023.

Tom B Brown. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Min Cao, Shiping Li, Juntao Li, Liqiang Nie, and Min Zhang. Image-text retrieval: A survey on recent research and development. *arXiv preprint arXiv:2203.14713*, 2022.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. Ieee, 2017.

Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer. Membership inference attacks from first principles. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1897–1914. IEEE, 2022.

Nicholas Carlini, Milad Nasr, Christopher A Choquette-Choo, Matthew Jagielski, Irena Gao, Pang Wei W Koh, Daphne Ippolito, Florian Tramer, and Ludwig Schmidt. Are aligned neural networks adversarially aligned? *Advances in Neural Information Processing Systems*, 36, 2024.

William B Cavnar, John M Trenkle, et al. N-gram-based text categorization. In *Proceedings of SDAIR-94, 3rd annual symposium on document analysis and information retrieval*, volume 161175, page 14. Ann Arbor, Michigan, 1994.

Jianbo Chen, Michael I Jordan, and Martin J Wainwright. Hopskipjumpattack: A query-efficient decision-based attack. In *2020 ieee symposium on security and privacy (sp)*, pages 1277–1294. IEEE, 2020a.

Jinghui Chen, Dongruo Zhou, Jinfeng Yi, and Quanquan Gu. A frank-wolfe framework for efficient and effective adversarial attacks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 3486–3494, 2020b.

Kyunghyun Cho. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International conference on machine learning*, pages 2206–2216. PMLR, 2020.

Francesco Croce, Jonas Rauber, and Matthias Hein. Scaling up the randomized gradient-free adversarial attack reveals overestimation of robustness using established attacks. *International Journal of Computer Vision*, 128:1028–1046, 2020.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

Jacob Devlin. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

Lizhe Fang, Yifei Wang, Zhaoyang Liu, Chenheng Zhang, Stefanie Jegelka, Jinyang Gao, Bolin Ding, and Yisen Wang. What is wrong with perplexity for long-context language modeling? *arXiv preprint arXiv:2410.23771*, 2024.

Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *IEEE. CVPR 2004, Workshop on Generative-Model Based Vision*, 2004.

Marguerite Frank, Philip Wolfe, et al. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110, 1956.

Rinon Gal, Or Patashnik, Haggai Maron, Amit H Bermano, Gal Chechik, and Daniel Cohen-Or. Stylegan-nada: Clip-guided domain adaptation of image generators. *ACM Transactions on Graphics (TOG)*, 41(4):1–13, 2022.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Gregory Griffin, Alex Holub, Pietro Perona, et al. Caltech-256 object category dataset. Technical report, Technical Report 7694, California Institute of Technology Pasadena, 2007.

Danna Gurari, Qing Li, Abigale J Stangl, Anhong Guo, Chi Lin, Kristen Grauman, Jiebo Luo, and Jeffrey P Bigham. Vizwiz grand challenge: Answering visual questions from blind people. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3608–3617, 2018.

Trevor Hastie. The elements of statistical learning: data mining, inference, and prediction, 2009.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross attention control. *arXiv preprint arXiv:2208.01626*, 2022.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

S Hochreiter. Long short-term memory. *Neural Computation MIT-Press*, 1997.

Phillip Howard, Avinash Madasu, Tiep Le, Gustavo Lujan Moreno, Anahita Bhiwandiwalla, and Vasudev Lal. Probing and mitigating intersectional social biases in vision-language models with counterfactual examples. *arXiv preprint arXiv:2312.00825*, 2023.

Tooba Imtiaz, Morgan Kohler, Jared Miller, Zifeng Wang, Mario Sznaier, Octavia Camps, and Jennifer Dy. Saif: Sparse adversarial and interpretable attack framework. *arXiv preprint arXiv:2212.07495*, 2022.

Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General perception with iterative attention. In *International conference on machine learning*, pages 4651–4664. PMLR, 2021.

Martin Jaggi. Revisiting frank-wolfe: Projection-free sparse convex optimization. In *International conference on machine learning*, pages 427–435. PMLR, 2013.

Fred Jelinek, Robert L Mercer, Lalit R Bahl, and James K Baker. Perplexity—a measure of the difficulty of speech recognition tasks. *The Journal of the Acoustical Society of America*, 62(S1):S63–S63, 1977.

Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*. 3rd edition, 2025. URL `https://web.stanford.edu/~jurafsky/slp3/`. Online manuscript released January 12, 2025.

Lei Ke, Wenjie Pei, Ruiyu Li, Xiaoyong Shen, and Yu-Wing Tai. Reflective decoding network for image captioning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 8888–8897, 2019.

Geewook Kim, Teakgyu Hong, Moonbin Yim, Jinyoung Park, Jinyeong Yim, Wonseok Hwang, Sangdoo Yun, Dongyoon Han, and Seunghyun Park. Donut: Document understanding transformer without ocr. *arXiv preprint arXiv:2111.15664*, 7(15):2, 2021.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical Report TR-2009, University of Toronto, 2009. URL `https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf`.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

T Kudo. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*, 2018.

Simon Lacoste-Julien. Convergence rate of frank-wolfe for non-convex objectives. *arXiv preprint arXiv:1607.00345*, 2016.

Tiep Le, Vasudev Lal, and Phillip Howard. Coco-counterfactuals: Automatically constructed counterfactual examples for image-text pairs. *Advances in Neural Information Processing Systems*, 36, 2024.

Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Šaško, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Clément Delangue, Théo Matussière, Lysandre Debut, Stas Bekman, Pierric Cistac, Thibault Goehringer, Victor Mustar, François Lagunas, Alexander Rush, and Thomas Wolf. Datasets: A community library for natural language processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–184, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. URL `https://aclanthology.org/2021.emnlp-demo.21`.

Ang Li, Allan Jabri, Armand Joulin, and Laurens Van Der Maaten. Learning visual n-grams from web data. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4183–4192, 2017.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.

Yinhan Liu. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 364, 2019.

I Loshchilov. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Jan Macdonald, Mathieu Besançon, and Sebastian Pokutta. Interpretable neural networks with frank-wolfe: Sparse relevance maps and relevance orderings. *arXiv preprint arXiv:2110.08105*, 2021.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

TorchVision maintainers and contributors. Torchvision: Pytorch's computer vision library. `https://github.com/pytorch/vision`, 2016.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60, 2014.

Kenneth Marino, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. Ok-vqa: A visual question answering benchmark requiring external knowledge. In *Proceedings of the IEEE/cvf conference on computer vision and pattern recognition*, pages 3195–3204, 2019.

Tomas Mikolov. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 3781, 2013.

Marius Mosbach, Maksym Andriushchenko, Thomas Trost, Matthias Hein, and Dietrich Klakow. Logit pairing methods can fool gradient-based attacks. *arXiv preprint arXiv:1810.12042*, 2018.

Ramaravind K Mothilal, Amit Sharma, and Chenhao Tan. Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the 2020 conference on fairness, accountability, and transparency*, pages 607–617, 2020.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519, 2017.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

Archit Parnami and Minwoo Lee. Learning from few examples: A summary of approaches to few-shot learning. arxiv 2022. *arXiv preprint arXiv:2203.04291*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

Bryan A Plummer, Liwei Wang, Chris M Cervantes, Juan C Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. Flickr30k entities: Collecting region-to-phrase

correspondences for richer image-to-sentence models. In *Proceedings of the IEEE international conference on computer vision*, pages 2641–2649, 2015.

Xiangyu Qi, Kaixuan Huang, Ashwinee Panda, Mengdi Wang, and Prateek Mittal. Visual adversarial examples jailbreak large language models. *arXiv preprint arXiv:2306.13213*, 2023.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

Jacopo Rizzo. *Evaluating pre-trained language models on partially unlabeled multilingual economic corpora*. PhD thesis, 2022.

Stephen Robertson. Understanding inverse document frequency: on theoretical arguments for idf. *Journal of documentation*, 60(5):503–520, 2004.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

Christian Schlarmann and Matthias Hein. On the adversarial robustness of multimodal foundation models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3677–3685, 2023.

Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems*, 35:25278–25294, 2022.

Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. *Introduction to information retrieval*, volume 39. Cambridge University Press Cambridge, 2008.

Rico Sennrich. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison frogs! targeted clean-label poisoning attacks on neural networks. *Advances in neural information processing systems*, 31, 2018.

Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. 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)*, pages 2556–2565, 2018.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE, 2017.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards vqa models that can read. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8317–8326, 2019.

Amanpreet Singh, Ronghang Hu, Vedanuj Goswami, Guillaume Couairon, Wojciech Galuba, Marcus Rohrbach, and Douwe Kiela. Flava: A foundational language and vision alignment model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15638–15650, 2022.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020a.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020b.

Matteo Stefanini, Marcella Cornia, Lorenzo Baraldi, Silvia Cascianelli, Giuseppe Fiameni, and Rita Cucchiara. From show to tell: A survey on deep learning-based image captioning. *IEEE transactions on pattern analysis and machine intelligence*, 45(1):539–559, 2022.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

MosaicML NLP Team. Introducing mpt-7b: A new standard for open-source, commercially usable llms, 2023. URL www.mosaicml.com/blog/mpt-7b. Accessed: 2023-05-05.

Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction {APIs}. In *25th USENIX security symposium (USENIX Security 16)*, pages 601–618, 2016.

Maria Tsimpoukelli, Jacob L Menick, Serkan Cabi, SM Eslami, Oriol Vinyals, and Felix Hill. Multimodal few-shot learning with frozen language models. *Advances in Neural Information Processing Systems*, 34:200–212, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575, 2015.

Prateek Verma and Jonathan Berger. Audio transformers: transformer architectures for large scale audio understanding. adieu convolutions (2021). *arXiv preprint arXiv:2105.00335*.

Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harv. JL & Tech.*, 31:841, 2017.

Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F Wong, and Lidia S Chao. Learning deep transformer models for machine translation. *arXiv preprint arXiv:1906.01787*, 2019.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in neural information processing systems*, 33: 5776–5788, 2020.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45. Association for Computational Linguistics, October 2020. URL https://aclanthology.org/2020.emnlp-demos.6.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

Han Xiao, Huang Xiao, and Claudia Eckert. Adversarial label flips attack on support vector machines. In *ECAI 2012*, pages 870–875. IOS Press, 2012.

Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 3485–3492. IEEE, 2010.

Lili Yu, Bowen Shi, Ramakanth Pasunuru, Benjamin Muller, Olga Golovneva, Tianlu Wang, Arun Babu, Binh Tang, Brian Karrer, Shelly Sheynin, et al. Scaling autoregressive multi-modal models: Pretraining and instruction tuning. *arXiv preprint arXiv:2309.02591*, 2(3), 2023.

Xu Zhang, Xinzheng Niu, Philippe Fournier-Viger, and Xudong Dai. Image-text retrieval via preserving main semantics of vision. In *2023 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1967–1972. IEEE, 2023.

Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*, 2023.

Wanrong Zhu, Jack Hessel, Anas Awadalla, Samir Yitzhak Gadre, Jesse Dodge, Alex Fang, Youngjae Yu, Ludwig Schmidt, William Yang Wang, and Yejin Choi. Multimodal c4: An open, billion-scale corpus of images interleaved with text. *Advances in Neural Information Processing Systems*, 36, 2024.

List of Algorithms

A

Appendix

A.1. Experiment Details

In this section, we will provide the details regarding the various experiments conducted.

A.1.1. Sparse versus Non-Sparse Attacks on OpenFlamingo

Model. We use 16-bit precision, along with a `max_generation_length` of 20 to ensure that the output of our model is 20 words.

Datasets. For image captioning, we use the validation set of COCO 2014 and the Flickr30k 1K test set. For VQA, the validation sets of OK-VQA and VizWiz are utilized.

N-Shot Sub-Setting. In the case of the 4-shot sub-setting, the context samples are taken from the training set of a dataset, while the query image is taken from the validation/test set. In the case of the 0-shot sub-setting, the query image is taken from the validation/test set.

Attacks. For SAIF, the ∞-norm constraint of the perturbation is $\epsilon = \frac{255}{255}$, and the 1-norm constraint of the sparsity-constrained mask is set to $k = 1000$. For APGD, we utilize two values of ϵ to constrain the ∞-norm of the perturbation : $\epsilon \in \{\frac{1}{255}, \frac{4}{255}\}$. The number of iterations considered for both attacks is 100 unless mentioned otherwise. The attacks were conducted on a total of 800 samples.

Miscellaneous. In the targeted attacks, we restrict the `max_generation_length` to the total number of words in the target caption. For untargeted attacks, we use the StanfordCoreNLP Java [Arnold et al., 2005] library [Manning et al., 2014] via the `pycocoevalcap` package for calculating the CIDEr score and the

`compute_vqa_accuracy` function from the OpenFlamnigo library to calculate VQA accuracy. For the targeted attacks, `load_metric` from `datasets` library [Lhoest et al., 2021] is utilized for calculating the BLEU-4 score.

A.1.2. Fine-tuning of CLIP Models on Adversarial Attacks versus Counterfactuals

Attack. The hyper-parameters for the **APGD** attack are ∞-norm budget for the perturbation, $\epsilon \in \{\frac{1}{255}, \frac{4}{255}\}$, and 100 iterations.

Datasets. Following [Le et al., 2024], the four datasets for fine-tuning CLIP models are given below.

1. **MS-COCO** dataset: The MS-COCO dataset is a subset of the validation set of the COCO 2017 dataset. The samples are filtered out of the validation set based on whether they were present in the COCO-CFs dataset. Hence, the remaining samples are 17,410 pairs of images and captions.

2. **[MS-COCO + COCO-CFs]$_{\text{base}}$** dataset: The dataset is the combination of the MS-COCO dataset, constructed in 1, and the COCO-CFs dataset. There is a 50% random sampling from the MS-COCO dataset (about 8705 image-caption pairs) and 25% random sampling from the COCO-CFs dataset (about 4353 image-captions and their counterfactuals, so 8706 image-caption pairs), amounting to 17,411 samples.

3. **[MS-COCO + COCO-CFs]$_{\text{medium}}$** dataset: The dataset is the combination of the MS-COCO dataset, constructed in 1, and the COCO-CFs dataset. All the samples from the MS-COCO dataset (about 17,410 image-caption pairs) and 75% random sampling from the COCO-CFs dataset (about 26,115 image-caption pairs) are used, amounting to 43,525 image-caption pairs.

4. **[MS-COCO + COCO-CFs]$_{\text{all}}$** dataset: The dataset is the combination of the MS-COCO dataset, constructed in 1, and the COCO-CFs dataset. All the samples from the MS-COCO dataset (about 17,410) and the COCO-CFs dataset (about 34,820) are used, amounting to 52,230 image-caption pairs.

For **APGD**, we utilized the COCO-CFs dataset to create adversarial samples. For instance, given an original image `149622_1_img_0.jpg` in the COCO-CFs dataset with an original caption "*Two small **owls** on a branch in a tree*", there is a counterfactual image `149622_1_img_1.jpg` with a counterfactual caption "*Two small **birds** on a branch in a tree*". To create adversarial samples, we set the counterfactual caption as the target caption and then perform a targeted attack on

the original image using the 9-billion parameter model from the OpenFlamingo family. We then create an adversarial sample for every original image in the COCO-CFs dataset (about 17,410 images), which we will call **APGDs** dataset, and then create datasets (2-4) accordingly. Due to the random sampling required in the **base** and **medium** datasets, we take 15 seeds[1] to get considerable results.

Model. We use the pre-trained CLIP model and processor `clip-vit-base-patch32` from the Transformers library.

Fine-Tuning. We use the AdamW regularizer [Loshchilov, 2017] with a learning rate of 5e-7, weight decay of 0.001, training batch size of 128 and 20 iterations. The dataset is divided in an 80:20 ratio, with 80% of the data being used for fine-tuning the model and the rest 20% for evaluation, similar to [Le et al., 2024].

0-Shot Image Classification. We use the test/validation sets of all datasets except for Caltech101 and Caltech256, as both of them do not contain test/validation sets. We instead randomly select 20% of the total samples for both of the datasets, as test sets.

ITR. We consider **Recall@1**, **Recall@5** and **Recall@10** for the ITR task. We perform it on the **Flickr30k 1K** test set, which contains 1000 image-caption pairs.

[1]The Following seeds are taken: $107, 108, \cdots, 121$.

A.2. Additional Data

In this section, we give tables containing detailed data for all the experiments discussed.

Mode	ϵ	COCO		Flickr30k		OK-VQA		VizWiz	
		0-shot	4-shot	0-shot	4-shot	0-shot	4-shot	0-shot	4-shot
Original		82.36	93.12	58.46	66.16	34.20	32.57	16.99	22.56
Adversarial									
all	$\frac{1}{255}$	17.39	17.36	11.81	12.35	3.03	2.73	3.98	3.53
query			18.74		12.85		3.15		3.80
all	$\frac{4}{255}$	**4.23**	**3.85**	**2.96**	**3.17**	**1.02**	**0.90**	**2.26**	**2.66**
query			4.63		3.27		1.45		2.79

Table A.1.: Results for **untargeted** attacks: CIDEr scores for COCO and Flickr30k, VQA accuracies for OK-VQA and VizWiz are being reported. The attack utilized here is **APGD** with $\epsilon \in \{\frac{1}{255}, \frac{4}{255}\}$ with 100 iterations. In the *query* rows, only the query images are being perturbed

Mode	ϵ	COCO		Flickr30k		OK-VQA		VizWiz	
		0-shot	4-shot	0-shot	4-shot	0-shot	4-shot	0-shot	4-shot
Original		82.36	93.12	58.46	66.16	34.20	32.57	16.99	22.56
Adversarial									
all	$\frac{255}{255}$	35.50	34.13	25.26	24.71	13.52	4.75	12.76	4.84
query			45.83		29.34		8.28		11.60

Table A.2.: Results for **untargeted** attacks: CIDEr scores for COCO and Flickr30k, VQA accuracies for OK-VQA and VizWiz are being reported. The attack utilized here is **SAIF** with $k = 1000$ and $\epsilon = \frac{255}{255}$ with 100 iterations. In the *query* row, only the query images are being perturbed

Target	ϵ	Success rate		BLEU-4	
		0-shot	4-shot	0-shot	4-shot
"Please reset your password"					
all	$\frac{1}{255}$	8.63	39.25	8.82	41.15
query			7.63		7.86
all	$\frac{4}{255}$	**94.00**	**100**	**94.15**	**100**
query			96.75		96.67
"A person suffers severe side effects after vaccination"					
all	$\frac{1}{255}$	0.0	0.37	0.0	3.37
query			0.0		0.0
all	$\frac{4}{255}$	**5.63**	**65.62**	**17.89**	**81.05**
query			7.13		22.30

Table A.3.: Results for **targeted** attacks: BLEU-4 score and ASR attack between the target caption and output captions are being reported. The attack utilized here is **APGD** with $\epsilon = \{\frac{1}{255}, \frac{4}{255}\}$ with 100 iterations. In the *query* rows, only the query images are being perturbed.

Target	ϵ	Success rate		BLEU-4	
		0-shot	4-shot	0-shot	4-shot
"Please reset your password"					
all	$\frac{255}{255}$	8.00	36.62	10.31	37.85
query			4.25		4.40
"A person suffers severe side effects after vaccination"					
all	$\frac{255}{255}$	0.25	0.5	1.26	2.48
query			0.0		0.0

Table A.4.: Results for **targeted** attacks: BLEU-4 score and ASR attack between the target caption and output captions are being reported. The attack utilized here is **SAIF** with $k = 1000$ and $\epsilon = \frac{255}{255}$ with 100 iterations. In the *query* row, only the query images are being perturbed.

	$\#D_{train}$	%ASs	**CIFAR10**	**CIFAR100**	**Food101**	**Caltech101**	**Caltech256**	**ImageNet**	**Mean**
None (pre-trained CLIP)	0	0%	89.83	65.08	84.01	83.64	82.65	63.38	78.09
MS-COCO	13,928	0%	**89.02**	**65.33**	**82.17**	**85.48**	**83.70**	**62.34**	**78.01**
MS-COCO + APGDs	13,928	50%	$85.81_{1.41}$	$59.81_{2.79}$	$78.24_{2.06}$	$79.69_{2.53}$	$79.16_{1.65}$	$58.04_{1.94}$	$73.76_{2.04}$
MS-COCO + APGDs	34,820	60%	$86.33_{1.24}$	$60.21_{2.12}$	$78.35_{1.40}$	$79.37_{1.57}$	$79.05_{1.26}$	$58.02_{1.50}$	$73.55_{1.48}$
MS-COCO + APGDs	41,784	67%	**88.15**	**63.62**	**80.04**	**81.57**	**80.77**	**59.76**	**75.65**

Table A.5.: Results for CLIP fine-tuned on **MS-COCO+APGDs** with varying datasets and **APGD** ($\epsilon = \frac{1}{255}$) for the task of 0-shot image classification. The $\#D_{train}$ denotes the total number of samples used for fine-tuning. The %ASs denotes the percentage of adversarial samples sampled from the APGDs dataset.

	$\#D_{train}$	%ASs	**CIFAR10**	**CIFAR100**	**Food101**	**Caltech101**	**Caltech256**	**ImageNet**	**Mean**
None (pre-trained CLIP)	0	0%	89.83	65.08	84.01	83.64	82.65	63.38	78.09
MS-COCO	13,928	0%	**89.02**	**65.33**	**82.17**	**85.48**	**83.70**	**62.34**	**78.01**
MS-COCO + APGDs	13,928	50%	$85.49_{1.39}$	$59.78_{2.77}$	$78.30_{1.98}$	$80.07_{2.43}$	$79.09_{1.80}$	$58.05_{1.94}$	$73.46_{2.03}$
MS-COCO + APGDs	34,820	60%	$86.58_{0.95}$	$60.78_{1.84}$	$78.35_{1.38}$	$79.93_{1.44}$	$79.22_{1.22}$	$58.05_{1.45}$	$73.82_{1.34}$
MS-COCO + APGDs	41,784	67%	**88.30**	**63.43**	**79.97**	**81.28**	**80.61**	**59.85**	**75.57**

Table A.6.: Results for CLIP fine-tuned on **MS-COCO+APGDs** with varying datasets and **APGD** ($\epsilon = \frac{4}{255}$) for the task of 0-shot image classification. The $\#D_{train}$ denotes the total number of samples used for fine-tuning. The %ASs denotes the percentage of adversarial samples sampled from the APGDs dataset.

	$\#D_{train}$	%CFs	**CIFAR10**	**CIFAR100**	**Food101**	**Caltech101**	**Caltech256**	**ImageNet**	**Mean**
None (pre-trained CLIP)	0	0%	89.83	65.08	84.01	83.64	82.65	63.38	78.09
MS-COCO	13,928	0%	**89.02**	65.33	**82.17**	85.48	83.70	**62.34**	**78.01**
MS-COCO + COCO-CFs	13,928	50%	$87.06_{0.59}$	$63.24_{1.27}$	$80.74_{1.04}$	$85.27_{0.55}$	$83.51_{0.43}$	$59.95_{1.24}$	$76.63_{0.80}$
MS-COCO + COCO-CFs	34,820	60%	$88.72_{0.48}$	$64.57_{1.47}$	$80.83_{0.70}$	$85.76_{0.71}$	$83.62_{0.66}$	$60.03_{1.04}$	$77.26_{0.78}$
MS-COCO + COCO-CFs	41,784	67%	**88.92**	**66.65**	**81.94**	**85.89**	**84.16**	**61.10**	**77.96**

Table A.7.: Results for CLIP fine-tuned on **MS-COCO+COCO-CFs** with varying datasets for the task of 0-shot image classification. The $\#D_{train}$ denotes the total number of samples used for fine-tuning. The %CFs denotes the percentage of counterfactuals sampled from the COCO-CFs dataset.

			i2t			t2i			
Training dataset	$\#D_{train}$	%CFs	**R@1**	**R@5**	**R@10**	**R@1**	**R@5**	**R@10**	**Mean**
None (pre-trained CLIP)	0	0%	69.60	90.50	95.00	67.20	89.00	93.80	84.18
MS-COCO	13,928	0%	76.70	94.10	96.90	74.60	93.70	96.90	88.81
MS-COCO + COCO-CFs	13,928	50%	$75.55_{0.83}$	$93.76_{0.54}$	$96.61_{0.41}$	$74.49_{0.61}$	$93.16_{0.49}$	$96.49_{0.34}$	$88.34_{0.41}$
MS-COCO + COCO-CFs	34,820	60%	$76.00_{1.62}$	$94.03_{0.34}$	$96.84_{0.30}$	$\underline{\mathbf{75.19}}_{0.74}$	$93.62_{0.34}$	$96.76_{0.25}$	$88.74_{0.37}$
MS-COCO + COCO-CFs	41,784	67%	**76.90**	**94.10**	**97.20**	75.10	**93.90**	**96.90**	**89.01**

Table A.8.: Results for CLIP fine-tuned on **MS-COCO+COCO-CFs** on ITR for the Flickr30k 1K test set. The $\#D_{train}$ denotes the total number of samples used for fine-tuning. The %CFs denotes the percentage of counterfactuals sampled from the COCO-CFs dataset.

			i2t			t2i			
Training dataset	$\#D_{train}$	%ASs	**R@1**	**R@5**	**R@10**	**R@1**	**R@5**	**R@10**	**Mean**
None (pre-trained CLIP)	0	0%	69.60	90.50	95.00	67.20	89.00	93.80	84.18
MS-COCO	13,928	0%	**76.70**	**94.10**	**96.90**	**74.60**	**93.70**	**96.90**	**88.81**
MS-COCO + APGDs	13,928	50%	$73.71_{1.48}$	$92.85_{0.74}$	$95.80_{0.48}$	$72.02_{1.40}$	$91.79_{0.95}$	$95.61_{0.35}$	$86.96_{0.83}$
MS-COCO + APGDs	34,820	60%	$74.40_{1.04}$	$93.13_{0.56}$	$96.00_{0.43}$	$72.66_{1.09}$	$91.99_{0.80}$	$95.96_{0.35}$	$87.36_{0.75}$
MS-COCO + APGDs	41,784	67%	**75.70**	**93.40**	**96.70**	**74.20**	**93.10**	**96.50**	**88.27**

Table A.9.: Results for CLIP fine-tuned on **MS-COCO+APGDs** ($\epsilon = \frac{1}{255}$) on ITR for the Flickr30k 1K test set. The $\#D_{train}$ denotes the total number of samples used for fine-tuning. The %ASs denotes the percentage of adversarial samples sampled from the APGDs dataset.

			i2t			t2i			
Training dataset	$\#D_{train}$	%ASs	**R@1**	**R@5**	**R@10**	**R@1**	**R@5**	**R@10**	**Mean**
None (pre-trained CLIP)	0	0%	69.60	90.50	95.00	67.20	89.00	93.80	84.18
MS-COCO	13,928	0%	**76.70**	**94.10**	**96.90**	**74.60**	**93.70**	**96.90**	**88.81**
MS-COCO + APGDs	13,928	50%	$73.72_{1.39}$	$93.07_{0.63}$	$96.07_{0.35}$	$71.86_{1.36}$	$92.18_{0.71}$	$95.61_{0.43}$	$87.08_{0.73}$
MS-COCO + APGDs	34,820	60%	$74.45_{1.43}$	$\underline{\mathbf{93.43}}_{0.49}$	$96.13_{0.57}$	$72.65_{1.24}$	$92.07_{0.73}$	$95.80_{0.34}$	$\underline{\mathbf{88.42}}_{0.71}$
MS-COCO + APGDs	41,784	67%	**76.10**	93.10	**96.50**	**74.20**	**93.20**	**96.30**	88.23

Table A.10.: Results for CLIP fine-tuned on **MS-COCO+APGDs** ($\epsilon = \frac{4}{255}$) on ITR for the Flickr30k 1K test set. The $\#D_{train}$ denotes the total number of samples used for fine-tuning. The %ASs denotes the percentage of adversarial samples sampled from the APGDs dataset.