TEXT GENERATION AND COMPREHENSION FOR OBJECTS IN IMAGES AND VIDEOS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

HAZAN ANAYURT ÖZYEĞİN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

SEPTEMBER 2021

Approval of the thesis:

TEXT GENERATION AND COMPREHENSION FOR OBJECTS IN IMAGES AND VIDEOS

submitted by **HAZAN ANAYURT ÖZYEĞİN** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Halit Oğuztüzün
Head of Department, **Computer Engineering** _____

Assoc. Prof. Dr. Sinan Kalkan
Supervisor, **Computer Engineering, METU** _____

Examining Committee Members:

Assist. Prof. Dr. Emre Akbaş
Computer Engineering, METU _____

Assoc. Prof. Dr. Sinan Kalkan
Computer Engineering, METU _____

Assoc. Prof. Dr. Hacer Yalım Keleş
Computer Engineering, Hacettepe University _____

Assoc. Prof. Dr. Mehmet Erkut Erdem
Computer Engineering, Hacettepe University _____

Assist. Prof. Dr. Ramazan Gökberk Cinbiş
Computer Engineering, METU _____

Date:

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: HAZAN ANAYURT ÖZYEĞİN

Signature :

ABSTRACT

TEXT GENERATION AND COMPREHENSION FOR OBJECTS IN IMAGES AND VIDEOS

ANAYURT ÖZYEĞİN, HAZAN
M.S., Department of Computer Engineering
Supervisor: Assoc. Prof. Dr. Sinan Kalkan

September 2021, 64 pages

Text generation from visual data is a problem often studied using deep learning, having a wide range of applications. This thesis focuses on two different aspects of this problem by proposing both supervised and unsupervised methods to solve it.

In the first part of the thesis, we work on referring expression comprehension and generation from videos. We specifically work with relational referring expressions which we define to be expressions that describe an object with respect to another object. For this, we first collect a novel dataset of referring expressions and videos where there are multiple copies of the same object, making relational referring expressions necessary to describe them. Moreover, we train two baseline deep networks on this dataset, which show promising results. Finally, we propose a deep attention network that significantly outperforms the baselines on our dataset.

In the second part of the thesis, we tackle the problem we solved in the first part in an unsupervised way. Models that generate text from videos or images tend to be supervised, which means that there needs to be corresponding textual description for every visual example in the datasets they use. However, collecting such paired data is

a costly task and much of the data we have is not labeled. As the lack of data was one of the bottlenecks in the supervised part of our thesis, in this part we consider the same problem in an unsupervised setting. For this, we adapt the CycleGAN architecture by Zhu *et al*. to be between the visual and text domains. Moreover, we use this architecture to perform experiments on different video and image captioning datasets, for some of which we achieve promising results.

Keywords: Referring Expressions, Video Captioning, Image Captioning, Unsupervised Learning

ÖZ

İMGE VE VİDEOLARDAKİ NESNELERDEN YAZI ÜRETME VE ANLAMA

ANAYURT ÖZYEĞİN, HAZAN
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. Sinan Kalkan

Eylül 2021 , 64 sayfa

Birçok uygulama alanı olması nedeniyle, görsel verilerden yazı üretme problemi derin öğrenme kullanılarak yaygın olarak çalışılan bir problemdir. Bu tezde, bu problemi denetimli ve denetimsiz olarak çözmek için iki farklı yönüne odaklanıyoruz.

Tezin ilk kısmında, videolardan atıfsal tümleç anlama ve üretme konularında çalışıyoruz. Çalışmamızı özellikle bir nesneyi başka bir nesne kullanarak tanımlayan ilişkisel atıfsal tümleçler üstüne yapıyoruz. Bunun için ilk olarak videolar ve ilişkisel atıfsal tümleçler üzerine yeni bir veri kümesi topluyoruz. Bu veri kümesi, aynı nesnenin birden çok kopyasını bulundurması nedeniyle atıfsal tümleçlerin ilişkisel olmasını gerekli kılıyor. Ayrıca, bu veri kümesi üzerinde umut vadeden sonuçlar veren iki tane baz model eğitiyoruz. Son olarak, aynı veri kümesinde baz modellerden çok daha iyi sonuçlar gösteren derin bir model öneriyoruz.

Tezin ikinci kısmında, ilk kısımda denetimli olarak çözdüğümüz probleme denetimsiz olarak yaklaşıyoruz. Video veya imgelerden yazı üreten modellerin çoğu denetimli olarak eğitiliyor. Bu kullandıkları veri kümelerindeki her görsel örneğin karşılığı olan bir yazı olması gerektiği anlamına geliyor. Ancak, bu şekilde eşlenmiş verileri elde

etmenin masraflı olması nedeniyle elimizdeki verilerin büyük kısmı etiketlenmiş değil. Önceki kısımdaki darboğazlarımızdan biri bu veri azlığı olduğu için, bu kısımda aynı problemi denetimsiz olarak değerlendiriyoruz. Bunun için Zhu vd. tarafından yapılan CycleGAN modelini görsel ve yazı alanları arasında çalışması için uyarlıyoruz. Ayrıca, aynı mimariyi videolardan ve imgelerden yazı üretmek için kullanıyoruz ve bazı deneylerimizde umut vadeden sonuçlar görüyoruz.

Anahtar Kelimeler: Atıfsal Tümleçler, Videolardan Yazı Üretme, İmgelerden Yazı Üretme, Denetimsiz Öğrenme

To my family which now happily includes Artun

ACKNOWLEDGMENTS

First and foremost, I would like to thank my supervisor Assoc. Prof. Dr. Sinan Kalkan for his endless support and guidance through the last three years. I am so glad that we could continue the journey we started during my senior year into my Master's years as well. He has taught me so many valuable things through these years and I could not ask for a better supervisor.

It is no secret that without Artun I would not be where I am in any aspect of life; academical, comedic, culinary, musical, artistic, anything you could name. He has been my companion in quite literally everything through the past five years, and I am genuinely so grateful to have someone I can depend on, come rain or shine. I am looking forward to having his company for the rest of our lives.

I would like to thank my family for always being there whenever I need them, and always letting me know they will support me in any venture in life. I don't like phone calls but I love phone calls with them. I always felt included in our household even though I wasn't physically there.

I appreciate my father for always offering help no matter the subject, be it classes or even my thesis. He never ceased asking if there's anything he can do and was always willing to learn more even though my knowledge in computer science became better than his sometime in the course of my undergraduate years.

I am thankful to my mother for making me feel like I'm always her little daughter who is welcome to lay my head in her lap and get my hair stroked any time I would like. She never made me feel like I'm far away from home despite the months and kilometers apart.

I could not go on without my sister providing me with so much love when I needed it. I'm sorry this had to be through the phone most of the time and I couldn't witness her becoming a "cool" teen. I still really appreciate the cuddles and tickle fights we

got to have along the way.

I'm lucky I had Samet and İsmail sharing this Master's experience with me. I'm glad we had each other and could go through all the hardships together. We were like a little support group, which I really needed. I have already missed our group study sessions.

This work was partially supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK) through the project titled "Object Detection in Videos with Deep Neural Networks" (project no 117E054).

I would like to thank the Scientific and Technological Research Council of Turkey (TÜBİTAK) for the financial support offered in the 2210A - Domestic Graduate Scholarship Program during my education.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

Adam	Adaptive Momentum
BLEU	Bilingual Evaluation Understudy
BN	Batch Normalization
C3D	Convolutional 3D
CycleGAN	Cycle-Consistent Generative Adversarial Network
FAN	Feature Attention Network
GAN	Generative Adversarial Network
ILSVRC	ImageNet Large-Scale Visual Recognition Challenge
LN	Layer Normalization
LSTM	Long Short-Term Memory
mAP	Mean Average Precision
MattNet	Modular Attention Network
METEOR	Metric for Evaluation of Translation with Explicit Ordering
MS-COCO	Microsoft Common Objects in Context
PCA	Principal Component Analysis
R-CNN	Regional Convolutional Neural Network
RE	Referring Expression
ReLU	Rectified Linear Unit
ResNet	Residual Network
RNN	Recurrent Neural Network
VIRAT	Video Image Retrieval and Analysis Tool
VIREF	Video Referring Expressions
VIREF-a	VIREF without Attention
VIREF-e	VIREF without LSTM encoder

WEN	Word Estimation Network
WGAN	Wasserstein GAN
WGAN-GP	Wasserstein GAN Gradient Penalty

CHAPTER 1

INTRODUCTION

This chapter is partially adapted from our paper, Anayurt *et al*. [1].

1.1 Motivation and Problem Definition

With the increase of visual data, navigating through it has become harder for humans. Since the only way we have for communicating with a computer is textual, bridging the gap between visual and textual data representations is of importance.

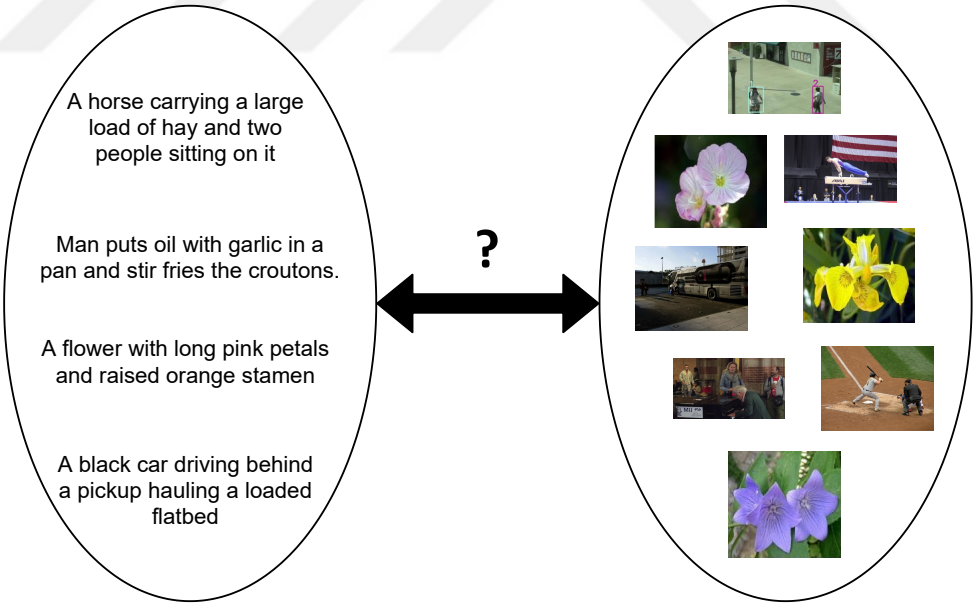


Figure 1.1: Finding the mapping between textual and visual data is a challenging yet an important problem with many applications in various domains.

Generating text descriptions from visual data is one way to close this gap, and its

many applications have been a popular subject of study in deep learning in recent years [2]. These applications include a wide range of tasks such as assisting visually-impaired people and image/video indexing.

Another way is comprehending the textual data and matching it with the content of the visual data. Content-based search in videos and images has many applications such as person search with textual descriptions and retrieval from surveillance data.

This study addresses two different aspects of this area: finding/describing objects in videos using relational referring expressions and unsupervised captioning of images and videos.

1.1.1 Referring Expression Generation and Comprehension from Videos

Textual descriptions of visual data can have many forms. A widely-used one is referring expressions (REs) which are descriptions that identify an object. Humans are likely to use non-relational REs that include absolute attributes such as color or relative attributes such as size especially when there are no objects with the same attributes [3]. However, when there are ambiguities e.g. owing to having multiple instances of the same object categories or multiple objects having the same attributes, humans prefer relational RE which are descriptions that identify an object with respect to other objects [3, 4]. Some relational RE examples can be seen in Figure 1.2.

Previous studies that address searching for objects in visual data using REs work on datasets where the level of ambiguity is low [6, 7, 8]. Hence, they did not require the use of relational REs. There was also little video data to be found that had RE annotations, relational or otherwise. To address this, we collect a video referring expression dataset with ambiguities where each object is described using another object. Moreover, to be able to search for objects in the videos in our dataset using referring expressions, we train a model using the dataset we collected.

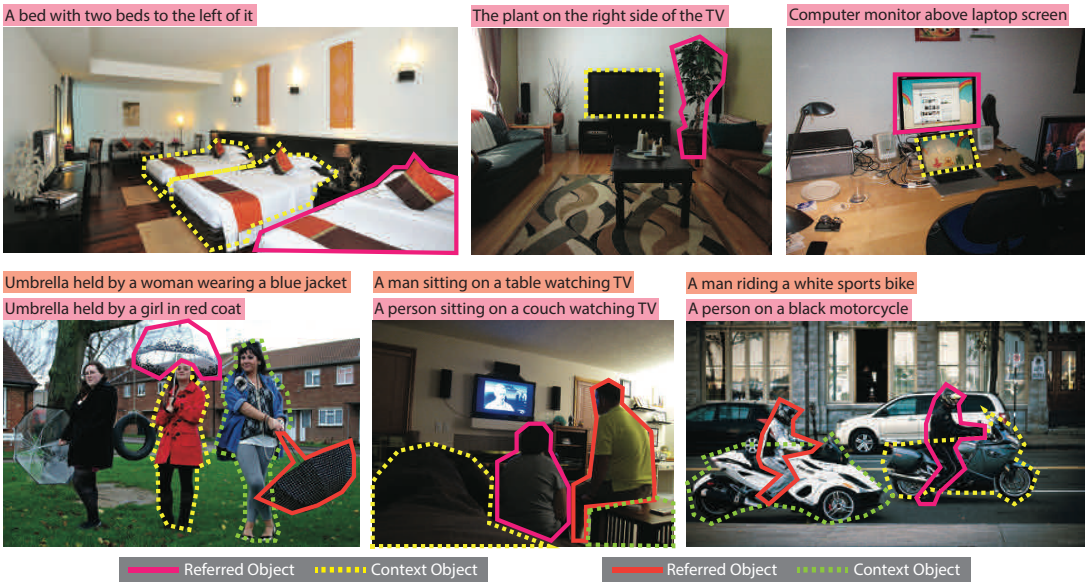


Figure 1.2: Referring expression examples for object pairs in images. Taken from Nagaraja *et al*. [5].

1.1.2 Unsupervised Translation from Visual Data to Text

Models that generate text from videos or images tend to be supervised, which means that for every image or video in the datasets they use, there is some corresponding textual description [2, 9]. However, this requires the collection of such paired data which is a costly task. As a result, only a fraction of the video and image data we have is labeled. This limit in the amount of paired data can be considered one of the main bottlenecks when training a deep learning model.

Unsupervised learning offers a solution to this by allowing the usage of unlabeled data in our training. Such methods have recently been used on various tasks such as image-to-image translation [10] and machine translation [11] and have achieved notable results.

Inspired by the recent success of unsupervised methods in deep learning, we apply one of the popular methods, Cycle-Consistent Generative Adversarial Networks (CycleGAN) [10], to the video referring expression generation and video/image captioning tasks.

1.2 Proposed Methods and Models

We propose two methods for the supervised and unsupervised settings respectively. First, we collect a video RE dataset METU-VIREF (METU Video Referring Expressions) [1] and devise a model for both RE generation and comprehension on it. Secondly, we adapt the architecture proposed by Zhu *et al.* [10] to work between the visual data and text domains.

1.2.1 Supervised Referring Expression Generation and Comprehension from Videos

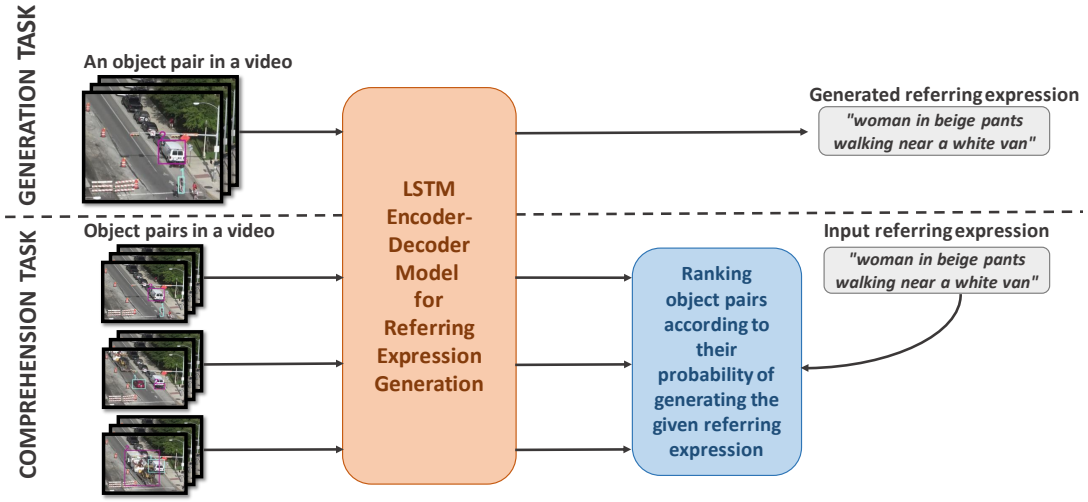


Figure 1.3: An overview of the generation and comprehension tasks performed by our model. In the generation task, given a video sequence containing an object pair, our model generates a RE defining one object using the other. In the comprehension task, given a RE, our model ranks object pairs in a given video starting from the pair that is best defined by the RE. Taken from Anayurt *et al.* [1].

We firstly collect a dataset for REs describing an object using another (context) object, which means that every RE includes two objects: the main object and the context object. We collect the REs on the objects from the VIRAT (Video Image Retrieval and Analysis Tool) dataset [12], which is a surveillance video dataset mainly containing people and vehicles, and a subset of the ILSVRC2015 (ImageNet Large-Scale Visual Recognition Challenge) dataset [13] that contains vehicles.

Using the dataset we collected, we train a model that recognizes the relationship between two objects including the classes and physical appearances of the objects and the actions they are performing. To achieve this, we use a model similar to the one used by Mao *et al*. [14] for RE comprehension and generation on images. The idea behind the model is first training a Long Short-Term Memory (LSTM) generator that, given two objects in a video, generates an RE describing the first object using the second object. Then, this LSTM generator is used in the comprehension task. See Fig. 1.3 for an illustration.

1.2.2 Unsupervised Text Generation from Videos and Images

With the previous method, we solve the problem of RE generation using the paired data we collected. In this part, we aim to solve the same problem in an unsupervised way, without the guidance of the pairing between the data. In this way, the model can be trained without being constrained by the small amount of paired data. We work on not only the RE generation problem on the METU-VIREF dataset but also the video and image captioning problems with the model.

We base our method on the CycleGAN model [10], which enables unsupervised translation between two domains. We establish this model, which was originally among two image domains, between the video features and RE features. In the CycleGAN architecture, two cycles are established between two domains by training generators in both directions. The adapted CycleGAN architecture we use can be seen in Figure 1.4.

Let us call the visual domain $\mathcal{V}$ and the text domain $\mathcal{T}$. Then, one of the cycles is $\mathcal{V} \rightarrow \mathcal{T} \rightarrow \mathcal{V}$ and the other one is $\mathcal{T} \rightarrow \mathcal{V} \rightarrow \mathcal{T}$. With the so-called cycle consistency loss, CycleGAN can preserve the information during the translation from one domain to the other by trying to bring the initial input and the recovered one closer. It would be easier for the model to reduce this loss if it performs the simplest translation instead of mapping the two domains arbitrarily.

Before using the visual/text data as a domain in the adapted CycleGAN, we preprocess the data to ease the training procedure. The feature extraction methods differ

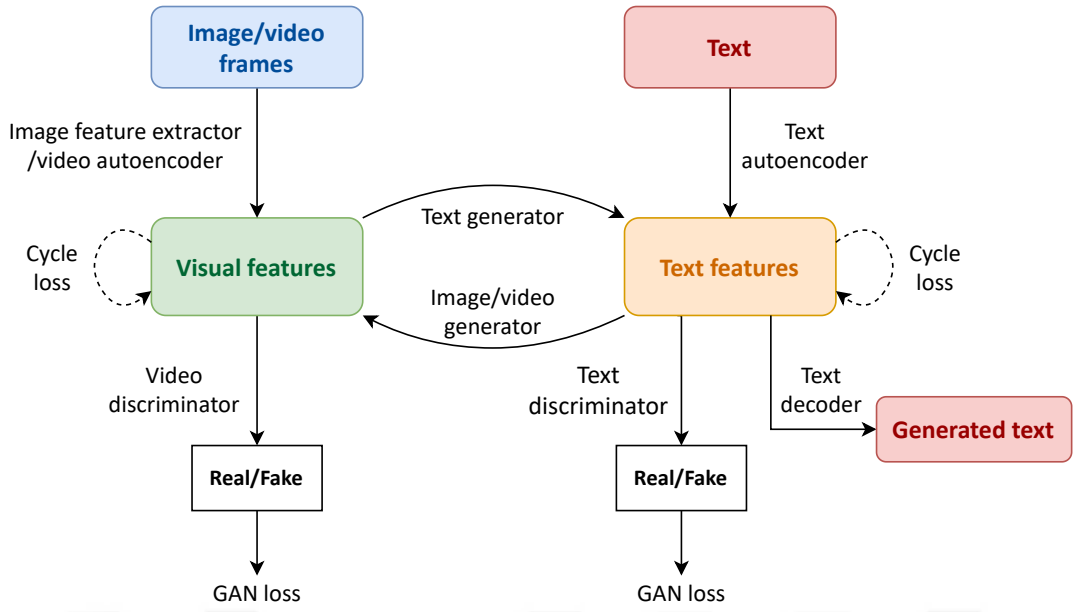


Figure 1.4: The adapted CycleGAN architecture for unsupervised generation of text from images or videos.

between visual datasets which are discussed in the following chapters; however, for the text data, an autoencoder is employed to preserve the unsupervised property of the task.

During testing, as we only want to generate text from visual data and not the other way around, the visual feature generator part of the model is not needed. To go from the text features that were generated to the actual text domain, we use the decoder part of the previously-trained text autoencoder.

1.3 Contributions and Novelties

Our contributions are as follows:

- We collect a dataset of relational referring expressions which is highly ambiguous and includes numerous instances of the same objects. This contribution was done in collaboration with Sezai Artun Özyeğin, Ülfet Çetin, and Utku Aktaş.
- We design and train a deep attention model on the collected dataset that both generates referring expressions to describe object pairs in videos and finds ob-

ject pairs in videos that best fit a given referring expression. The architecture was designed in collaboration with Sezai Artun Özyeğin, who also took part in the implementation of the model.

- We design and train two simpler baseline models and experiment with the three models we developed to compare their performances. We show the effectiveness of the components in our model that we omitted in the simpler models. These experiments were conducted together with Sezai Artun Özyeğin.
- We adapt the CycleGAN model that was originally used for unsupervised translation between image domains to work between visual data feature and text feature domains.
- To simplify the work of the CycleGAN model, we train various autoencoders on different referring expression and captioning datasets to extract features from them in an unsupervised way.
- We conduct experiments on various image and video captioning datasets and compare our results to both supervised and unsupervised methods developed previously.

1.4 The Outline of the Thesis

In Chapter 2, we explore the related work on referring expressions. Moreover, we examine the methods used for unsupervised captioning. We give related background information for the methods we used.

The contributions of the thesis are covered in two parts: Chapter 3 for the Supervised Referring Expression Generation/Comprehension contributions and Chapter 4 for the Unsupervised Text Generation contributions.

In Section 3.1, we go over our dataset collection process and present its specifications.

In Section 3.2, we discuss the architectural details of the VIREF model for referring expression generation and comprehension. We present the two baseline architectures we developed.

In Section 3.3, we give the implementation and training details. We present our experiments and results on comprehension and generation on the METU-VIREF dataset.

In Section 4.1, we go over the feature extraction processes for the visual and text data. We explain the CycleGAN-based architecture we formed for unsupervised translation between these features.

In Section 4.2, we give the details of the image and video captioning datasets we used to train our unsupervised models. We present and comment on the results of our experiments on each dataset.

In Chapter 5, we conclude our thesis by summarising our methodology and results. We comment on the problems we encountered during this work and propose possible future work that can be done to overcome them.

CHAPTER 2

BACKGROUND AND RELATED WORK

This chapter is partially taken from our paper, Anayurt *et al.* [1].

2.1 Referring to Objects in Images

Many applications that include object search in images can benefit from understanding how humans use natural language to describe objects in the real world [15]. With its numerous applications, this has been a popular subject of study, and several works proposing different approaches to this problem have emerged.

Mao *et al.* [14] use a Recurrent Neural Network (RNN) architecture that they train on RE generation to perform both generation and comprehension. They state that generation task can be converted to comprehension task by using Bayes' Rule. In their model, an RE is generated using an LSTM [16] network that is initialized with VGG features [17] extracted from the input image. After that, they use this generator model and rank possible regions by their probability of giving the expression in the query. In our work, we use a similar method.

Nagaraja *et al.* [5] formulate RE generation and comprehension as a Multiple Instance Learning problem for learning RE generation with respect to a context object even when such objects are not labeled in the dataset. They claim that, by modeling the context between objects, they were able to achieve a better performance than when they used only the main object's properties.

Yu *et al.* [18] use attention to improve their performance in RE comprehension, which is common for problems linking computer vision and language. Their model, Mat-

tNet, is a module-level attention model that modulates contributions from location, object and action/relation. They build an end-to-end model for the comprehension task on images and they train it on RefCOCO datasets.

In a more recent work, Liu *et al.* [19] use attribute-level attention to guide RE generation and comprehension in images. They use an attribute parser to obtain attributes from REs, and try to find the probability of each image region having those attributes during RE comprehension. They use triplet loss to bring together the encodings of image and RE domains and use this image encoding they learn using this loss when generating REs.

Yang *et al.* [20] propose to use scene graphs to improve both RE generation and comprehension in images. When comprehending REs, they extract scene graphs from both images and REs, then match the objects in REs to corresponding image regions. When generating REs from images, they use scene graphs as an intermediary step.

2.2 Referring to Objects in Videos

Inspired by similar studies on images, many have focused on RE comprehension for objects in videos. These studies have shown that REs in videos are more challenging since they also contain temporal information, and the dimensionality of the inputs is much higher. Examples studies include Vasudevan *et al.* [6], Wiriyathammabhum *et al.* [8] and Khoreva *et al.* [7].

Vasudevan *et al.* [6] study the link of human gaze with REs. In their work, the gaze location is estimated from the user's eyes, and an object corresponding to an uttered RE is sought in a region around the gaze location.

Khoreva *et al.* [7] address the segmentation of a referred object in a video given an RE query. They train their method on the DAVIS [21] dataset using the REs they collected to describe the segmented objects and evaluate it on the segmentation problem.

Wiriyathammabhum *et al.* [8] extend the MattNet model [18] for REs in videos by including motion modules and attention on motion modules.

In our work, the temporal aspect introduced by videos is handled by using an LSTM to encode the features extracted from sampled video frames. This encoding allows the architecture in the work of Mao *et al.* [14] to be adapted to the video domain. Furthermore, we use attention during this encoding process, inspired by the works that show the benefit of it on tasks that bring visual data and natural language together.

2.3 Datasets for Referring to Objects in Images and Videos

There have been many works investigating the relationship between REs and visual data. For this purpose, multiple datasets with different specifications have been collected. Widely used datasets for referring expression for objects in images include RefCOCO [15] and Google RefExp [14].

Kazemzadeh *et al.* [15] is one of the first works that collect a large-scale dataset called RefCOCO containing REs for objects in images. They crowd-source this collection process as a two-player game to obtain and verify REs.

Mao *et al.* [14] collect Google RefExp dataset that specifically focuses on images that have more than one instance of the same class of object. They require the REs to unambiguously define a single object in the image. These constraints cause the REs to have more detail and use context and not just the object itself.

While being similar, referring expressions in the video domain have some core aspects that are different from the image domain. Due to the temporal aspect, the movement of objects with respect to each other can be seen and this introduces a new kind of context to the REs in the dataset. ORGaze [6] and DAVIS [7] are datasets that provide RE annotations for videos.

Vasudevan *et al.* [6] collect a significant number of REs for objects in videos. In their dataset, ORGaze, they not only collect REs but also the information of where the person annotating an object is looking at in a given scene. They use this information to aid them in finding the target object among many proposals.

Khoreva *et al.* [7] provide REs for the DAVIS video object segmentation dataset [21]. They use these REs as textual queries to obtain object segmentations.

In other works, the collected referring expression datasets on images and videos do not generally have context objects labeled, so even though the referring expressions themselves do contain these objects, a model that exploits the relation between objects cannot easily be trained. With the METU-VIREF dataset [1], we aim to make use of these relations by training the model using the supervision of both the main object and the context object.

2.4 Image and Video Captioning Datasets

Different from REs which are noun phrases that describe a target object, captions are sentences that describe the content of an image or video. Captions can contain one or more sentences and different levels of detail. Caption generation from visual data has a large number of applications, even more so than RE generation, due to the variety in possible caption characteristics and the broadness of the visual domain.

Owing to them being applicable to many real-life scenarios, image and video captioning have been very popular problems. This has produced a large number of datasets, each with different qualities. There are generic datasets with unconstrained domains, as well as ones that focus on limited areas such as cooking or flowers. The datasets that we use are discussed in the following paragraphs.

Microsoft COCO Captions [22] is one of the most commonly-used image captioning datasets in the literature. It has complex scenes with natural contexts that contain people, animals and common objects [23]. It has 120,000 images that each have 5 different captions.

Reed *et al.* [24] provide captions for the Oxford 102 Flowers dataset [25] which contains a single flower in each image. They collect ten single-sentence captions for each flower image, providing fine-grained visual descriptions.

Similar to the image domain, the video domain also has a wide variety of datasets. We use ActivityNet Captions [26] dataset which has dense descriptions of 20,000 videos, corresponding to 100,000 sentences in total. The videos in the dataset contain humans performing 200 classes of different activities.

Even though all of the datasets we mentioned contain paired visual data and captions, since we approach the problem in an unsupervised way, we regard the visual and textual data as two separate corpora during training time. During test time, however, we use the labels to evaluate our performance.

2.5 Generative Adversarial Networks

First proposed by Goodfellow *et al.* [27] in 2014, Generative Adversarial Networks (GANs) have been one of the leading techniques for generative modeling in deep learning since. The idea behind GANs is two neural networks working against each other, each iteratively improving their performance to beat the other.

In generative adversarial networks, one of the two models, the generator, aims to generate data that comes from the same distribution as the real data, given noise as input. On the other hand, the second model, the discriminator, is fed with either real data points or data produced by the generator and tries to determine the probability of it coming from the real distribution.

This minimax game between the generator and the discriminator theoretically reaches its optimal state if the generated data distribution converges to real data distribution. In this case, the discriminator cannot distinguish between generated and real examples, and can at best give a 0.5 probability of being real to both.

Let G be the generator network and $G(\boldsymbol{z})$ the data generated by G, where $\boldsymbol{z}$ is the input noise vector sampled from standard normal distribution $p_{\boldsymbol{z}}$. Let D be the discriminator network, fed by input data $\boldsymbol{x}$ either sampled from real data distribution $p_{\boldsymbol{data}}$ or produced by the generator. $D(\boldsymbol{x})$ is the scalar value of the probability of $\boldsymbol{x}$ coming from $p_{\boldsymbol{data}}$. Then, the overall objective of the GAN can be defined as:

$$\min_G \max_D \mathbb{E}_{\boldsymbol{x}\sim p_{data}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]. \tag{2.1}$$

2.5.1 Wasserstein GAN

To alleviate certain problems in GAN training such as mode collapse and the necessity of carefully designing and balancing the generator and the discriminator, Arjovsky *et al.* [28] propose to use a different function to measure how far the generated and the real probability distributions are from each other. They analyze the behaviors of several distance and divergence functions and adapt Earth-Mover Distance (EMD) to GANs to address the mentioned problems.

Let $\mathbb{P}_r$, $\mathbb{P}_g$ be the real and generated probability distributions respectively. Then, the EMD can be written as follows:

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}[||x - y||], \tag{2.2}$$

where $\Pi(\mathbb{P}_r, \mathbb{P}_g)$ denotes the set of all joint distributions $\gamma(x, y)$ whose marginals are respectively $\mathbb{P}_r$ and $\mathbb{P}_g$.

Then, to solve this optimization problem, they convert it to its dual form and obtain:

$$W(\mathbb{P}_r, \mathbb{P}_g) = \sup_{||f||_L \leq 1} \mathbb{E}_{x\sim\mathbb{P}_r}[f(x)] - \mathbb{E}_{x\sim\mathbb{P}_\theta}[f(x)], \tag{2.3}$$

where $||f||_L$ corresponds to the Lipschitz constant of the function f. They further limit the function f by implementing it as a neural network and parametrizing it by some w where $||f_w||_L \leq K$. As they state in their work, since we scale the Lipschitz constant by some K, the distance $W(\mathbb{P}_r, \mathbb{P}_g)$ will be scaled by the same constant. The optimization problem becomes:

$$\max_{||f_w||_L \leq K} \mathbb{E}_{x\sim\mathbb{P}_r}[f_w(x)] - \mathbb{E}_{x\sim\mathbb{P}_\theta}[f_w(x)]. \tag{2.4}$$

Arjovsky *et al.* propose to clip the weights of the network f_w by some constant to ensure that the overall network will be K-Lipschitz continuous for some constant K. However, they conclude that clipping is not the best way to satisfy this constraint.

2.5.1.1 Wasserstein GAN Gradient Penalty

Although Wasserstein GAN (WGAN) improves the stability of GAN training, Gulrajani *et al.* [29] claim that it is still prone to failure in some cases. They suspect the

culprit of these problems to be the weight clipping and suggest an alternative way to impel the model to have a Lipschitz constant of 1.

Gulrajani *et al.* propose to add a penalty term to the loss to make the norm of the gradients with respect to the inputs close to 1:

$$\mathcal{L} = \mathbb{E}_{x\sim\mathbb{P}_r}[f_w(x)] - \mathbb{E}_{x\sim\mathbb{P}_\theta}[f_w(x)] + \lambda\mathbb{E}_{x\sim\mathbb{P}_s}[(||\nabla_x f_w(x)|| - 1)^2], \tag{2.5}$$

where $\mathbb{P}_s$ is a distribution created by sampling uniformly along straight lines between the points sampled from $\mathbb{P}_r$ and $\mathbb{P}_\theta$.

2.6 CycleGAN

Zhu *et al.* [10] propose CycleGAN, a method for unpaired image-to-image translation. This task aims to translate an image from source domain X to target domain Y where we do not have the supervision of the exact mapping between these two domains. CycleGAN trains two generators, $G : X \rightarrow Y$ and $F : Y \rightarrow X$, and their respective discriminators, D_Y and D_X, to learn a mapping between the two domains. Figure 2.1 (a) shows the CycleGAN model.

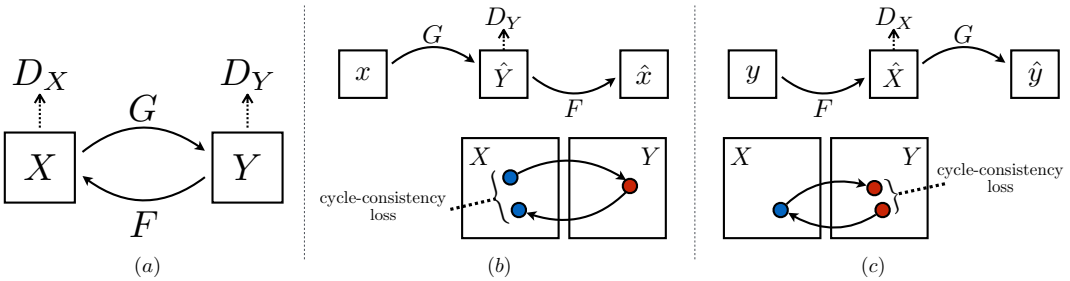


Figure 2.1: CycleGAN architecture. Taken from Zhu *et al.* [10]. (a) G denotes the generator defined from X domain to Y domain whereas F is the generator in the opposite direction. D_X is the discriminator that tells apart real X examples from the examples generated by F. D_Y is the discriminator that does the same in the Y domain. (b) Forward cycle-consistency loss that encourages $F(G(x))$ to be similar to x. (c) Backward cycle-consistency loss that encourages $G(F(y))$ to be similar to y.

This method can be applied to domain pairs where it is costly to obtain paired data, such as semantic segmentation, or where a well-defined mapping doesn't exist, such

as object transfiguration ($e.g.$ horse $\leftrightarrow$ zebra). After the initial paper which tackled the problem of image-to-image translation, CycleGAN and similar architectures have been used for translation between many different domains such as text-to-image [30] and text-to-text [11].

In the training of CycleGAN, two types of losses are used: adversarial loss and cycle consistency loss. Adversarial loss is the original GAN loss formulated as:

$$\mathcal{L}_{GAN}(G, D_Y, X, Y) = \mathbb{E}_{\boldsymbol{y} \sim p_{data}(\boldsymbol{y})}[\log D_Y(\boldsymbol{y})] + \mathbb{E}_{\boldsymbol{x} \sim p_{data}(\boldsymbol{x})}[\log(1 - D_Y(G(\boldsymbol{x})))], \tag{2.6}$$

for the GAN trained from X domain to Y domain and:

$$\mathcal{L}_{GAN}(F, D_X, Y, X) = \mathbb{E}_{\boldsymbol{x} \sim p_{data}(\boldsymbol{x})}[\log D_X(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{y} \sim p_{data}(\boldsymbol{y})}[\log(1 - D_X(F(\boldsymbol{y})))], \tag{2.7}$$

for the GAN trained in the opposite direction. Using only this loss does not ensure that the mapping from one domain to the other is the desired translation as a network with enough capacity can learn any arbitrary mapping. Therefore, Zhu *et al*. propose using cycle consistency loss:

$$\mathcal{L}_{cyc}(G, F) = \mathbb{E}_{\boldsymbol{x} \sim p_{data}(\boldsymbol{x})}[||F(G(\boldsymbol{x})) - \boldsymbol{x}||_1] + \mathbb{E}_{\boldsymbol{y} \sim p_{data}(\boldsymbol{y})}[||G(F(\boldsymbol{y})) - \boldsymbol{y}||_1], \tag{2.8}$$

where $||.||_1$ is the L1 norm. Cycle-consistency losses in both directions are illustrated in Figures 2.1 (b) and (c). This loss aims to preserve the information during translation. The overall loss of CycleGAN corresponds to:

$$\begin{aligned} \mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{GAN}(G, D_Y, X, Y) \\ + \mathcal{L}_{GAN}(F, D_X, Y, X) \\ + \alpha \mathcal{L}_{cyc}(G, F), \end{aligned} \tag{2.9}$$

where α is the weight parameter for the cycle consistency loss.

2.7 Unsupervised Translation Between Text and Visual Data

There are some works on unsupervised image captioning whose methods could be applied to the video domain as well.

In one of the earliest works on this subject, Gu *et al.* [31] approach the problem by making use of parallel corpora in Chinese and English to use Chinese as a pivot

language when generating English captions for images. This way, they do not need to have English image-caption pairs as supervision and they can make use of the image-caption pairs they have in Chinese for the training.

In a later work of theirs, Gu *et al*. [32] use scene graphs as a middle ground between images and text to eliminate the need for paired data. They first use a rule-based method to parse sentences into scene graphs, then train an encoder-decoder architecture to encode the scene graphs into features and decode these features to obtain sentences from them. The previously obtained sentence-scene graph pairs act as supervision for this training. To get scene graphs from images, they use a previously trained object detection and scene graph classification pipeline, then use the encoder they trained to obtain scene graph features. They use a CycleGAN-based [10] architecture for unsupervised translation between the scene graph features of sentences and images. Finally, they use the decoder they trained to go from the generated scene graph features to actual captions.

Feng *et al*. [33] propose to use a concept extractor on images to be able to train their model without paired data. They train a GAN to generate captions from image features. For the generated caption to match the concepts in the image, they give a reward to each generated word that matches one of the concepts extracted. To avoid their output quality being limited by the concept extractor, they also use the generator and the discriminator they trained to reconstruct the sentences and the images, and use this reconstruction loss for their training as well.

Laina *et al*. [34] also aim to bring the image and text domains together by mapping them into a shared latent space. To do this, they first obtain an unsupervised sentence embedding by training an autoencoder on a text corpus. However, they specifically want this autoencoder to encode the visual semantics in the text. To achieve this, they use triplet loss to pull the sentence features that contain the same visual concepts together. To project the visual input to this shared space, they first use a pretrained concept extractor, whose results they enrich with concept subcategories. To have these concepts align with the visual concepts extracted from the sentences, they use a loss to bring the two spaces together. They furthermore use a discriminator to make the image concept features closer to the sentence concept features. Finally, they use

the decoder of the autoencoder they trained to generate captions from the aligned concept embedding obtained from the images.

Guo *et al.* [35] propose a similar approach by using a concept extractor to map the image and text domains to a common space. For the translation from images to concepts, they train Faster R-CNN [36] on a comprehensive image dataset with object-level and image-level labels. For the textual concepts, they extract a dictionary of the common occurring concepts in the images. They then train a recurrent memory network from concepts to sentences using this dictionary they extracted as supervision. Finally, they use this memory model to generate captions from the concepts extracted from an input image.

2.7.1 Text-to-Image-to-Text Translation using Cycle Consistent Adversarial Networks

Gorti *et al.* [30] propose to apply unsupervised methods to text-to-image translation by making use of the CycleGAN [10] architecture. Even though they aim to generate images and not text, they train a caption generator as well to be able to fulfill cycle consistency, making their work closely related to ours. Moreover, they use the Oxford 102 Flowers Captions [24] dataset, which we also perform experiments on.

Gorti *et al.* divide the image generation process into two stages and use a discriminator in both stages to first generate a low-resolution version of the target image, then refine it. Different from the original CycleGAN architecture, they only form the cycle in one direction (text-to-image-to-text) since they are not actually interested in the caption generation task and only use it to improve image generation.

By using a caption generator on the generated image, they aim to recover the original input caption, as being able to recover it with less loss would mean that the generated image contains more of the details in the original input. In addition, they use the caption discriminator as a correctness check, and not just a realness check. To ensure this, the discriminator loss has a term to evaluate whether a caption actually matches with the given image.

In their work, the cycle consistency loss is shown to qualitatively improve the gen-

erated images in terms of matching more closely with the input text. The generated images themselves however become worse in terms of Inception score [37] when not considered in relation to the text.

CHAPTER 3

SUPERVISED REFERRING EXPRESSION GENERATION AND COMPREHENSION FROM VIDEOS

This chapter is adapted from our paper, Anayurt *et al*. [1].

3.1 METU-VIREF Dataset

As part of the undergraduate design project, we had started collecting a dataset of referring expressions defining objects using other objects. As part of this thesis, we continued collecting the dataset, and were able to complete and present it in our paper [1]. The dataset collection process and the dataset specifications are given in the following sections.

3.1.1 Dataset Collection

For our dataset, we used videos from two video datasets: The VIRAT Ground [12] and ILSVRC2015 [13] datasets. Our dataset consists of videos and bounding boxes from these two datasets and REs we collected for object pairs in these videos. VIRAT is a video surveillance dataset that contains videos of places like streets and parking lots. Due to this, the main object classes are people and vehicles. Because we wanted to narrow our domain down to fit the majority of classes in VIRAT, we only used videos that contained vehicles from the ILSVRC dataset. Due to this, VIRAT makes up most of our videos. Other than this, we did not put a restriction on object categories or REs on them, which means that our dataset contains objects (and REs using these objects) such as garbage bins, traffic cones, and objects carried by people e.g. bags

or boxes.

One thing we did put a restriction on while choosing object pairs was whether the pair actually had a relation or not. This was because an RE defining one object using the other could not be written if the objects did not have any meaningful "overlap" in the video. We saw that the number of pairs we obtained was still too high even after eliminating pairs that were too far apart spatially or temporally; therefore, we visually observed each pair to decide manually whether or not a meaningful RE could be written for them. The REs we have collected for this dataset are only for the pairs that remained after both eliminations.

To collect the REs for our dataset, we used the Microworkers crowdsourcing platform. The video sequences we showed the workers were only the parts of a video starting when the first object of the pair to enter the video enters and ending when the last one of the pair to exit the video exits. Each video sequence contained an object pair with one object labeled as the first object and the other object labeled as the second and we asked for the workers to provide an RE for the first object using the second object as the context object after watching the video. Therefore, the order of the labels was important. We also reversed the order of the pair and collected REs for the reverse order as well.

We collected six REs for each object pair (three for the straight and three for the reverse order). We asked the workers to give as much detail as possible and we did not have any vocabulary or grammar restrictions apart from asking the answers to be noun phrases defining the main object w.r.t. the context object without using full sentences. Differently from the previous datasets, we did not require the REs to be unambiguous; hence, one RE could be correctly identifying more than one pair.

3.1.2 Dataset Specifications

In Tables 3.1 and 3.2 we provide some descriptive statistics about our dataset. REs with length greater than 25 are not used during training as explained in Section 3.3.1. Moreover, words that occur only once are removed from our vocabulary.

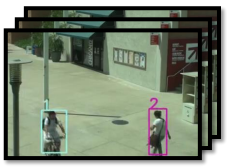
Table 3.1: Information about the videos in our dataset.

	Avg.	**Min.**	**Max.**
Sequence length (sec)	19.00	1	42
# objects per video	9.38	2	46
# pairs per video	19.57	2	148
# videos	125 (VIRAT) + 37 (ILSVRC)		
# objects (total)	1520		
# pairs (total)	3170		

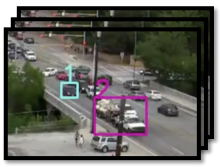
Table 3.2: Information about the REs in our dataset. $\bullet$ stands for the number of words in an RE, and $\star$ denotes the number of occurences of words in the dataset.

	Avg.	**Min.**	**Max.**
RE length	11.70	5	37
# RE per object	12.51	6	90
# RE	9416 ($\bullet < 25$) + 94 ($\bullet \geq 25$)		
# words in REs	708 ($\star > 1$) + 370 ($\star = 1$)		

Two example object pairs from our dataset along with the collected REs can be seen in Figure 3.1.



(i) *"a man in white shirt riding a bike and passing by a woman in white shirt"*
(ii) *"a man in white t-shirt riding bicycle beside a woman in white t-shirt"*
(iii) *"a male cyclist in white clothes passing a female pedestrian in a white blouse"*



(i) *"a black car going behind of a white truck which is carrying a load"*
(ii) *"a black car driving behind a pickup hauling a loaded flatbed"*
(iii) *"a black sedan following a white mini truck on a two-way street"*

Figure 3.1: Two object pairs from our dataset and the three REs collected for them.

3.2 Methodology

In this part of our work, we aim to solve both the referring expression generation and the comprehension task on the METU-VIREF dataset we collected. This has many applications such as searching for certain objects among many in a video and generating expressions that define certain objects already detected in a video. For this, we adapt the method of Mao *et al.* [14] by performing the two tasks together using the same model. The model is trained on the generation task and used in test to perform the comprehension task as well.

For our model, similar to [6, 7, 8], we assume that objects have been already detected or annotated. Then, given two objects in a video, our goal is to generate the most probable referring expression defining the first object using the second object, which is called the generation task. We also use the model trained on the generation task to find the most suitable object pair among labeled objects in a video given an RE as input, which is the comprehension task.

We use a method similar to Mao *et al.* and approach the generation and comprehension tasks together. We first encode the objects in videos using an LSTM, then we feed the encoded object pair to a decoder LSTM. We use the hidden states of the decoder to calculate the attention weights of the objects' features, enabling our model to attend to different combinations of features according to the word it wants to predict. After the attention, we combine the encoded features that the model has attended to and the hidden state of the decoder to calculate the probability of each word in our vocabulary to try to obtain the most probable RE given an object pair in a video. We also use these probabilities while searching for objects in a video (comprehension task). An overview of our model is provided in Figure 3.2.

3.2.1 Generation Task

The generation task can be formally defined as finding the most likely sequence of words, $\mathbf{r} =: < w_1, ..., w_n >$, (i.e., the referring expression) from the vocabulary given a video sequence $\mathbf{v} =: < I_1, ..., I_m >$ (I_i is the i^{th} frame) with annotated boxes for the main object, $\mathbf{o}^t =: < B_1^t, ..., B_m^t >$, and the context object, $\mathbf{o}^c =: < B_1^c, ..., B_m^c >$. B

is simply a vector with the coordinates of the top-left and the bottom-right corners of the bounding box.

For the generation task, we use an LSTM decoder architecture similar to Mao *et al.* with two main differences, the first of which is using an encoder LSTM on the object features. This helps in adapting the model to the video domain by allowing it to capture the temporal relation between the objects more accurately. The second one is the attention module attached on top of the outputs of the decoder.

The input ($\mathbf{x}_i$) of the encoder LSTM at step i are the features extracted from the i^{th} frame of the input video (sampled each second). Denoting the VGG16 (fc1) [17] features by $\phi()$, $\mathbf{x}_i$ is a concatenation of the following:

$$\mathbf{x}_i =< \phi(I_i(B_i^t)),\ \phi(I_i(B_i^c)),\ \phi(I_i),\ \phi(M(B_i^t)),\ \phi(M(B_i^c)) >, \tag{3.1}$$

where $I_i(B)$ denotes an image with the pixels in bounding box B; and $M(B)$ is a binary image (with the same size as I_i) where the pixels are white inside B and black elsewhere.

The hidden state of the encoder LSTM at step (frame) i can be denoted as follows:

$$\mathbf{h}_i^e = LSTM^e(\mathbf{x}_i^{a_0},\ h_{i-1}^e), \tag{3.2}$$

where $LSTM^e$ is the encoder LSTM, $\mathbf{x}_i^{a_0}$ is the i^{th} input scaled with the initial attention weights $a_0 =< a_0^1,\ a_0^2,\ a_0^3,\ a_0^4,\ a_0^5 >\in \mathbb{R}^5$ and $\mathbf{h}_i^e$ is the encoder hidden state at step i. $\mathbf{h}_0^e$ is a trainable parameter and it is initialized as 0. The attention weights are integrated as follows:

$$\mathbf{x}_i^{a_0} =< a_0^1{\times}\phi(I_i(B_i^t)),\ a_0^2{\times}\phi(I_i(B_i^c)),\ a_0^3{\times}\phi(I_i),\ a_0^4{\times}\phi(M(B_i^t)),\ a_0^5{\times}\phi(M(B_i^c)) > . \tag{3.3}$$

The rest of our model resembles a regular image captioning architecture with the addition of a feature attention mechanism. As input to the decoder, we used GloVe [38] 50-dimensional vector representations of the words. The hidden state of the decoder at step j can be denoted as follows:

$$\mathbf{h}_j^d = LSTM^d(\psi(w_j),\ h_{j-1}^d), \tag{3.4}$$

where w_j is the word at step j and $\psi(\cdot)$ is the function that takes a word to its embedding. The last hidden state of the encoder is fed into the decoder as its initial hidden

state, i.e., $\mathbf{h}_1^d = LSTM^d(\psi(w_{start}), h_m^e)$, where $\psi(w_{start})$ is the embedding vector of a pre-defined start token.

At each time step j of the decoder, the attention weight vector a_j is calculated using a network named Feature Attention Network (FAN) as $a_j = FAN(h_j^d)$. a_j is used as the attention weights to the features of the encoder (Figure 3.2). The attention weights a_j are integrated into the inputs in the same way as shown for a_0 in Equation 3.3. Then, with those weights, the encoder is run again. Combining the last hidden state of the newly run encoder and the $\mathbf{h}_j^d$, the output is calculated using the Word Estimation Network (WEN) as follows:

$$\mathbf{out}_j = WEN(h_j^d,\ LSTM^e(x_m^{a_j},\ LSTM^e(x_{m-1}^{a_j},\ \ldots))). \tag{3.5}$$

The output of WEN is passed through Softmax to obtain a probability distribution over the words in our vocabulary. With this, we obtain the probability of an RE, $p(\mathbf{r} \mid \mathbf{v}, \mathbf{o}^t, \mathbf{o}^c)$, by multiplying the word probabilities. In test time, we sample the most probable RE (i.e., $\arg\max_{\mathbf{r}} p(\mathbf{r} \mid \mathbf{v}, \mathbf{o}^t, \mathbf{o}^c)$) using beam search with beam size 3, similar to [14]. The illustration of the VIREF generation model can be seen in Figure 3.2.

The attention mechanism we employ is similar to the modular attention mechanism employed in the works of Wiriyathammabhum *et al.* [8] and Yu *et al.* [18]. The main differences are that attention in our model is performed over the extracted features at each time step, not over the modules that summarize information over time separately, and it is adapted to the video domain by using an encoder LSTM.

As can be seen in Equation 3.5, the encoder is run again for each time step of the decoder with its corresponding feature attention weights. It may be argued that this is too much computational overhead; however, in this way, the decoder can decide which input features it would attend to according to the word it wants to predict and this allows the features to interact in the encoding step. If we wanted to have feature-level attention but only ran the encoder once and attended to the encoded features, we would have to encode each feature separately. This would prevent the encoder from capturing the relation between the features such as an object and its location, or the motion of the objects with respect to each other.

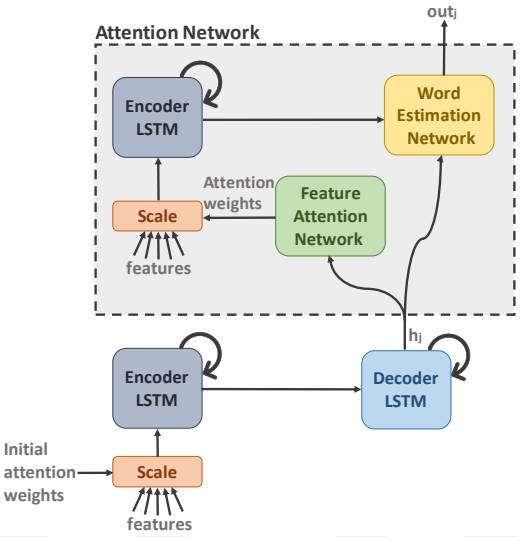


Figure 3.2: The VIREF model.

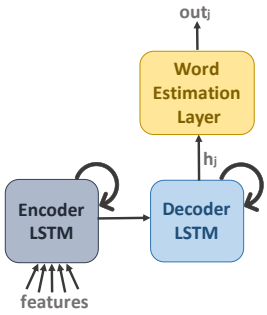


Figure 3.3: The VIREF-a model (VIREF without attention).

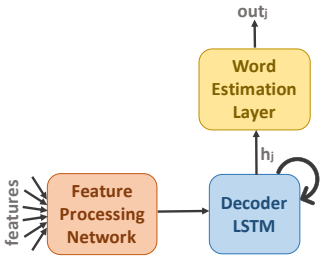


Figure 3.4: The VIREF-e model (VIREF without LSTM encoder).

3.2.2 Comprehension Task

For the comprehension task, we do not train a separate model but use the model we trained for the generation task. This becomes possible if we assume uniform prior for the probability of the video and the places of the objects in it (meaning they can occur anywhere in any video). Under this assumption, using the Bayes' Rule, the probability of a video and an object pair given a referring expression becomes directly proportional to the probability of a referring expression given an object pair since the evidences are the same:

$$\arg\max_{\mathbf{v},\mathbf{o^t},\mathbf{o^c}} p(\mathbf{v},\mathbf{o^t},\mathbf{o^c}|\mathbf{r}) = \arg\max_{\mathbf{v},\mathbf{o^t},\mathbf{o^c}} p(\mathbf{r}|\mathbf{v},\mathbf{o^t},\mathbf{o^c}). \tag{3.6}$$

Therefore, when an RE is taken as input, we calculate the probability of that RE for each object pair in each video, $p(\mathbf{r}|\mathbf{v},\mathbf{o^t},\mathbf{o^c})$, using the generator model as explained in Section 3.2.1. Then, we rank the object pairs according to the probability of them giving the input RE. Mao *et al.* [14] have used a similar approach.

3.2.3 Architecture and Loss

For both encoder and decoder, we use a six-layer LSTM, whereas Feature Attention Network and Word Estimation Network both consist of 3 fully-connected layers. Both the LSTMs and the fully-connected networks in the attention module use Dropout with a dropping probability of 0.2. The initial attention weights are trainable parameters and all attention weights are used after being passed through the Softmax function. The input features are 4096-dimensional each and add up to 20480 when concatenated after scaling(see Equation 3.3). The output of the Word Estimation Network is of size 1024, which is the size of our vocabulary, and is also passed through Softmax. For training the networks, we use cross-entropy loss.

3.2.4 Baseline Models

To compare our method with, we provide two baseline models. The architectures are similar to VIREF in certain parts with the main differences being in the encoder module. Essentially, these models are simplified versions of the VIREF model to show the performance improvements provided in both the generation and comprehension tasks by the LSTM encoder and attention modules.

3.2.5 VIREF without Attention (VIREF-a)

VIREF-a is the same model as VIREF without the extra attention module, as shown in Figure 3.3. This model can be described as a straightforward extension of the model used by Mao *et al.* to the video domain, without the usage of video-related feature extractors. The feature extractors in the image domain are still more powerful than the ones in the video domain, therefore, to efficiently use the information in the time dimension, an LSTM encoder is utilized.

3.2.6 VIREF without LSTM encoder (VIREF-e)

Another straightforward extension of the model of Mao *et al*. is directly using video feature extractors and feeding them into decoder LSTM (as illustrated in Figure 3.4). Using this kind of approach would remove the necessity of using a recurrent network to capture temporal information.

We used six features that could replace the LSTM-encoded VGG16 features. The first two are the averages of VGG16 (fc1) [17] features for the main and the context objects over time. The other four are the C3D (fc6) [39] features for the main object, the context object, and the main & context objects together (formed by blackening regions outside of their bounding boxes), and the whole scene. We believe that VGG16 features of the objects could capture the fine details of the objects and C3D features could contain motion-related information.

The model takes these six features, processes them using fully-connected layers, and uses the same decoder as VIREF-a. The only difference is that the encoder part uses different kinds of feature extraction methods instead of the LSTM encoder used in the other two models. Each of the features is of the size 4096 and the features add up to 24576 when concatenated. After the concatenation, features are passed through Feature Processing Network which consists of 3 fully-connected layers.

3.3 Experiments

In this section, we evaluate our method and compare it against the baselines on generation and comprehension tasks.

3.3.1 Training and Implementation Details

We split our dataset into training, validation and test sets respectively with 60%, 10%, 30% percentages. For training the networks, we used Adam optimizer with a learning rate of 10^{-4} and a batch size of 10. We used the validation set to perform early-stopping and observed this to be a sufficient measure against overfitting in our experiments.

The decoder LSTMs output a probability distribution over the words in our vocabulary. The vocabulary includes (i) the words that appear at least twice in our training set (634 words), and (ii) the words most frequently used in the Google RefExp dataset [14] (386 words). In total, with <*start*>, <*end*>, <*unk*> and <*nil*> tokens, our vocabulary contains 1024 words.

During training, we use padding (the <*nil*> token) so that the referring expressions will have the same length. However, the number of very long REs only make up a very small percentage of the dataset, and to be able to use less padding to even out the length of all REs, we exclude every RE that has a length of 25 or greater in training. Out of the $9,510$ REs we collected in total, only 94 exceed this length constraint and we are able to avoid higher training times at the cost of only 1% of our data.

3.3.2 Generation Results

To evaluate the results of the generator, we calculated average scores using the BLEU [40] and METEOR [41] metrics. The averaged BLEU-4 and METEOR scores of the models on the test set can be seen in Table 3.3. We observe that VIREF outperforms both baseline models in terms of the BLEU and METEOR metrics. However, even though VIREF is the best model in terms of BLEU and METEOR scores, it can be

seen that it generates REs from a more narrow set of words compared to VIREF-e. According to our observations, VIREF-e's vast usage of different words is sometimes accompanied by the generation of REs that do not meaningfully describe an object. We believe that this has affected its average BLEU and METEOR scores. Some of the generation results of the models can be observed in Figure 3.5.

Table 3.3: RE generation performance of the models. Higher is better.

Method	Average BLEU-4 Score	Average METEOR Score	# of words used in the output
VIREF-e	0.1197	0.4054	**233**
VIREF-a	0.1498	0.4580	75
VIREF	**0.2365**	**0.5492**	91

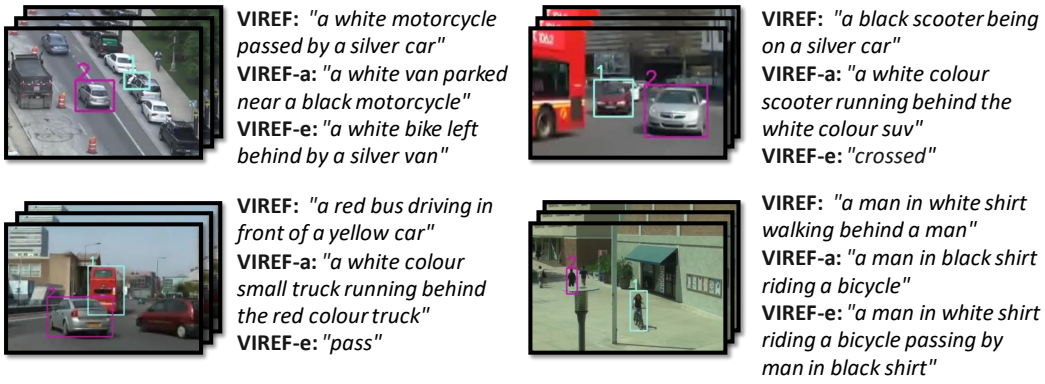


Figure 3.5: Sample generated REs. Object with label 1 is the main object, and the one with label 2 is the context object.

3.3.3 Comprehension Results

The comprehension part of the model, as mentioned in Section 3.2.2, uses the generator model. To evaluate the comprehension model, for each video, we selected an RE from the test set and calculated a matching score with every pair in that video. We assumed that only the object pair it was written for is the correct answer. This may have, however, led to a score that is less than the actual score, since our dataset consists of ambiguous examples, and an RE may actually represent more than one object pair.

We used two different retrieval task evaluation measures: mean average precision (mAP) and rank-k accuracy. AP is defined simply as $1/k$ if the correct pair is retrieved at rank k (since we have only a single "relevant document", average precision is directly equal to precision). On the other hand, rank-k accuracy is the percentage of results in which the correct pair is retrieved at the top k object pairs.

Table 3.4 lists the comprehension results of the methods in terms of mAP and rank-1, rank-2, and rank-3 accuracies. We observe that VIREF again performs better than the other models, thanks to its attention model. See also Fig. 3.6 for some sample results.

Table 3.4: RE comprehension results. Higher is better.

Method	**mAP**	**rank-1 accuracy**	**rank-2 accuracy**	**rank-3 accuracy**
VIREF-e	0.55	0.35	0.61	0.69
VIREF-a	0.46	0.26	0.49	0.57
VIREF	**0.65**	**0.47**	**0.69**	**0.78**

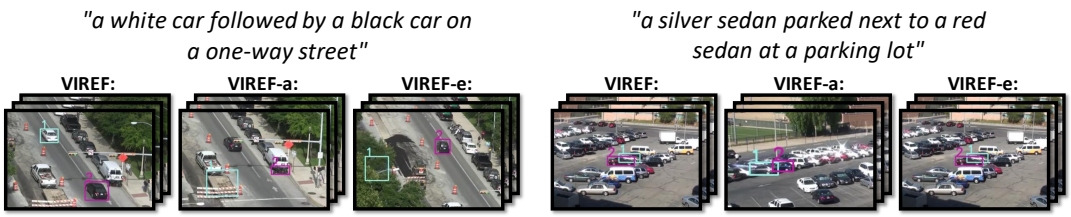


Figure 3.6: Sample comprehension results. Object with label 1 is the main object, and the one with label 2 is the context object.

CHAPTER 4

UNSUPERVISED TRANSLATION BETWEEN TEXT AND VISUAL DATA

4.1 Methodology

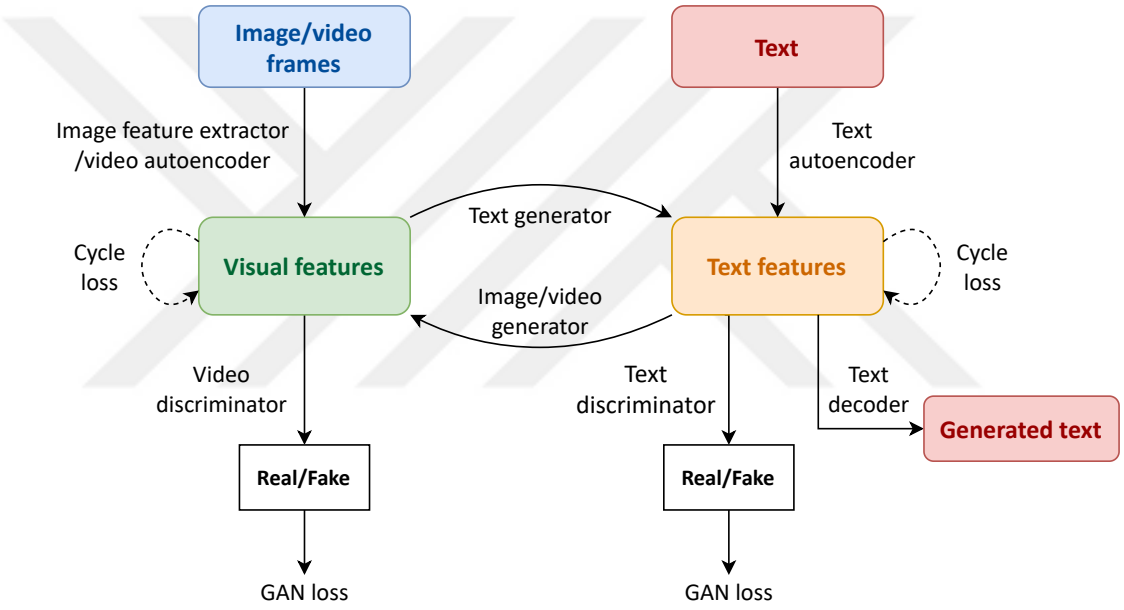


Figure 4.1: The adapted CycleGAN architecture for unsupervised generation of text from images or videos. This figure was previously used in Section 1.2.1.

Contrary to our supervised method in the previous chapter, we only perform the generation task and not the comprehension task with the unsupervised models. For all of the text generation tasks, namely referring expression (RE) generation from videos and image/video captioning, the same adapted CycleGAN architecture (Figure 4.1) is used between visual features and text features. However, the details of how we encode the visual data change. This section gives the details of each encoding process, an in-depth look at the adapted CycleGAN architecture, and the implementation

details for all of the models and their training processes.

4.1.1 Overview

The overall pipeline for the training and testing of our unsupervised architecture consists of the following steps:

1. Visual feature extraction (Section 4.1.2):
   (a) Extracting features from the frames of the video sequences in the METU-VIREF dataset.
   (b) Training an autoencoder on features extracted from the frames of the METU-VIREF dataset.
   (c) Using the encoder part of the video autoencoder we trained to encode the video frame features into visual features.
   (d) Using pretrained feature extractors to obtain visual features from the captioning datasets we use.

2. Text feature extraction (Section 4.1.3):
   (a) Training an autoencoder on each text corpus belonging to the datasets we use.
   (b) Using the encoder parts of the text autoencoders we trained to encode text data into text features.

3. Training the adapted CycleGAN for unsupervised translation between the visual features and the text features we obtained in the previous step. (Section 4.1.4)

4. Generating text from visual data during testing (Section 4.2.2):
   (a) Using the autoencoder and pretrained feature extractors to encode the visual data into visual features.
   (b) Using the text feature generator we trained in the previous step to generate text features from the visual features.
   (c) Using the decoder parts of the text autoencoders we trained to decode the text features into text.

These steps are explained in detail in the next sections.

4.1.2 Visual Feature Extraction

Since we build the adapted CycleGAN architecture to be between feature vectors we have different processes for different datasets to go from videos and images to their encoded versions. For the METU-VIREF dataset, to obtain a single feature vector from several video frames, we build and train an LSTM autoencoder on the features extracted from each frame.

4.1.2.1 Feature Extraction for the METU-VIREF Dataset

Since we use the same dataset, our notation in this section is similar to Section 3.2.1. Let $\mathbf{v} =: < I_1, ..., I_m >$ be a video sequence where I_i is the i^{th} frame sampled at each second. For each frame, we have the annotated boxes for the main object, $\mathbf{o}^t =: < B_1^t, ..., B_m^t >$, and the context object, $\mathbf{o}^c =: < B_1^c, ..., B_m^c >$ where B is the vector containing the top-left and the bottom-right corners of the boxes.

We train the model to be an autoencoder with an LSTM encoder-decoder architecture. The input ($\mathbf{x}_i$) of the encoder LSTM at step i are the features extracted from the i^{th} video frame. VGG() denotes features obtained from the output of the fc1 layer of the VGG16 network [17]. Then, $\mathbf{x}_i$ is the concatenation of the following:

$$\mathbf{x}_i =< \text{VGG}(I_i(B_i^t)),\ \text{VGG}(I_i(B_i^c)),\ \text{VGG}(I_i),\ \text{VGG}(M(B_i^t)),\ \text{VGG}(M(B_i^c)) >, \tag{4.1}$$

where $I_i(B)$ denotes an image with the pixels in bounding box B; and $M(B)$ is an image (with the same size as I_i) where the pixels are white inside B and black elsewhere.

We use a stacked LSTM with two layers. The initial hidden state we feed the encoder, $\mathbf{h}_0 \in \mathbb{R}^{2\times512}$ is a learnable parameter of the model initialized with 0s where 512 is the hidden size. The initial cell state $\mathbf{c}_0$ is a tensor with the same size, filled with 0s. At each time step i, we feed the encoder with $\mathbf{x}_i$, for $i = 1, ..., m$.

After m^{th} timestep, we obtain $\mathbf{h}_m$, which we feed to the decoder as its initial hidden state. We use teacher forcing during training, which means the actual video frame features are used as the decoder's input and not the features it has generated in the previous step. The decoder inputs for all time steps is $< \mathbf{x}_s, \mathbf{x}_1, ..., \mathbf{x}_m >$ where $\mathbf{x}_s$ is the start token and the expected output sequence is $< \mathbf{x}_1, ..., \mathbf{x}_m, \mathbf{x}_e >$ where $\mathbf{x}_e$ is the end token.

The loss at each time step is calculated using the cosine distance between the expected and generated features. To do this, we parse the feature vectors into their components as shown in Equation 4.1 and calculate the cosine distances of all five separately and sum them up.

We train the autoencoder for 500 epochs with batch size 10 using the Adam [42] optimizer with the learning rate 10^{-4}. Dropout is used on the outputs of each LSTM layer except the last layer with 0.2 probability. The video sequences have a maximum number of 42 frames (sampled each second).

The features extracted from each sequence, $\mathbf{\Phi}_{\mathcal{V}}$, which are used as the visual features in the training of CycleGAN, are the h_m's produced by the encoder. They have a size of 1024, which is the flattened version of the hidden states of the two layers.

4.1.2.2 Feature Extraction for the ActivityNet Captions Dataset

Because we could not obtain the raw videos from the ActivityNet dataset [26] as some of them were removed, we are not able to train an autoencoder on their frames as we did for the METU-VIREF dataset. Due to this, we use the publicly available[1] C3D [43] features of the dataset videos.

To extract these features, the pretrained C3D model [43] which has a temporal resolution of 16 frames is used. The network weights are used as-is and not fine-tuned with the ActivityNet dataset. They use the output of the fc7 layer whose dimensionality they reduce from 4096 to 500 using Principal Component Analysis (PCA). The features were extracted every 8 frames.

[1] `https://cs.stanford.edu/people/ranjaykrishna/densevid/`

The dataset contains dense captions made up of several sentences that describe parts of the video ordered temporally. In the dataset, there is the information of exactly which frames a sentence corresponds to. To ease the training, we split the video into these frame sequences each having one sentence as its caption and used these as our data samples. To obtain $\Phi_{\mathcal{V}}$ from each video sequence, we average all of the C3D features belonging to that sequence.

4.1.2.3 Feature Extraction for the MS-COCO and Oxford 102 Flowers Datasets

We extract features from these two datasets in the same way since they are both image datasets. For this, we use the pretrained Resnet-50 architecture [44] provided by PyTorch. We use the output of the average pooling layer as the image features $\Phi_{\mathcal{V}}$. Before feeding the images to the network, we resize the input to (512, 512) and normalize the input using the data mean and standard deviation: [0.485, 0.456, 0.406] and [0.229, 0.224, 0.225] respectively for channels in the order red, green, blue.

4.1.3 Text Feature Extraction

For all datasets, we employ the same denoising autoencoder architecture by Shen *et al*. [45] to encode the text. In their work, they claim that regular autoencoders tend to not map similar sentences to nearby latent vectors because they are able to learn arbitrary mappings due to their high capacity. They propose to add a denoising objective to overcome this problem by trying to reconstruct sentences from their perturbed versions. They randomly delete words with a probability as the perturbation. They show that this helps to guide the model to have a better geometry in its latent space meaning it has more similar latent vectors for similar sentences. Even though they present different autoencoder architectures such as adversarial autoencoder in their paper, we use the vanilla denoising autoencoder they provided online[2]. We train the architecture by Shen *et al*. on the different corpora belonging to the datasets we work on.

For the METU-VIREF [1] dataset, because of the low number of REs, we augment

[2] `https://github.com/shentianxiao/text-autoencoders`

the dataset with REs from DAVIS [7] and ORGaze [6] datasets. After combining the three datasets, we randomly split them into train, validation and test with ratios 90%, 5% and 5% respectively. The number of examples in the resulting training set is 37,326. For the ActivityNet Captions [26], MS-COCO Captions [22] and Oxford 102 Flowers Captions [24] datasets, we use only their own captions with the data splits the authors provided.

The architecture we use has 1-layer LSTM's for both the encoder and the decoder. The latent size is 1024, which means $\mathbf{\Phi}_{\mathcal{T}} \in \mathbb{R}^{1024}$. The vocabulary is limited to the most frequent 50,000 words. We trained the model for 50 epochs with batch size 64 using Adam optimizer [42] with the learning rate 5×10^{-4} and beta parameters 0.5 and 0.999 respectively.

4.1.4 Adapted CycleGAN Architecture

For the different datasets and tasks, we used a common CycleGAN-based architecture. However, we established it between different visual feature domains $\mathcal{V}$ and text feature domains $\mathcal{T}$. How we extract the visual and text features are mentioned in the previous sections. Therefore in this section, we will directly be working with the feature domains and the usage of "visual domain" and "text domain" refers to those.

Let $\mathcal{G}_{\mathcal{T}} : \mathcal{V} \to \mathcal{T}$ be the generator that translates the visual domain to the text domain and $\mathcal{G}_{\mathcal{V}} : \mathcal{T} \to \mathcal{V}$ be the generator that does the translation in the reverse direction. The respective text and visual discriminators are $\mathcal{D}_{\mathcal{T}} : \mathcal{T} \to \mathbb{R}$ and $\mathcal{D}_{\mathcal{V}} : \mathcal{V} \to \mathbb{R}$ in the case of WGAN. If LSGAN is used, $\mathbb{R}$ becomes $\{x \in \mathbb{R} \mid 0 < x < 1\}$. Figure 4.2 shows the complete architecture.

At each iteration, a batch of visual features $\mathbf{\Phi}_{\mathcal{V}}$'s are fed to $\mathcal{G}_{\mathcal{T}}$ to obtain the fake text features $\mathbf{\Phi}_{\mathcal{T}}^{\mathcal{V}\to\mathcal{T}}$. Similarly, a batch of text features $\mathbf{\Phi}_{\mathcal{T}}$'s are fed to $\mathcal{G}_{\mathcal{V}}$ to obtain the fake visual features $\mathbf{\Phi}_{\mathcal{V}}^{\mathcal{T}\to\mathcal{V}}$. After obtaining the fake features, the discriminators tell apart their respective features: $\mathcal{D}_{\mathcal{T}}$ discriminates $\mathbf{\Phi}_{\mathcal{T}}$ and $\mathbf{\Phi}_{\mathcal{T}}^{\mathcal{V}\to\mathcal{T}}$, $\mathcal{D}_{\mathcal{V}}$ discriminates $\mathbf{\Phi}_{\mathcal{V}}$ and $\mathbf{\Phi}_{\mathcal{V}}^{\mathcal{T}\to\mathcal{V}}$.

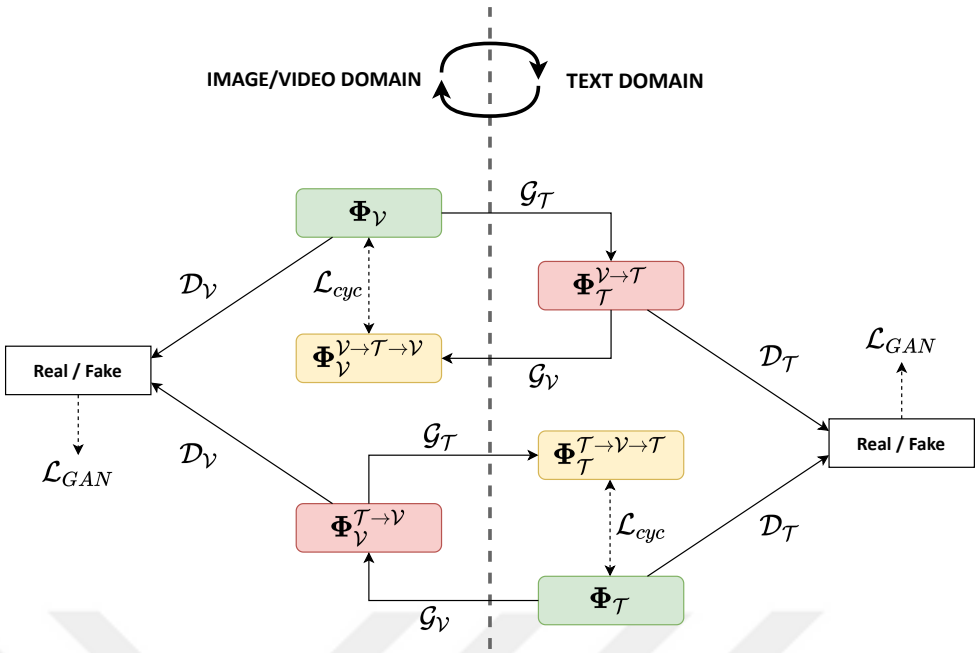


Figure 4.2: The adapted CycleGAN architecture.

If WGAN-GP is used, the GAN loss for $\mathcal{G}_{\mathcal{T}}$ and $\mathcal{D}_{\mathcal{T}}$ is

$$\begin{aligned}\mathcal{L}_{GAN}^{\mathcal{VT}}(\mathcal{G}_{\mathcal{T}}, \mathcal{D}_{\mathcal{T}}) = \mathbb{E}_{\mathbf{\Phi}_{\mathcal{T}}\sim\mathbb{P}_{\mathcal{T}}}[\mathcal{D}_{\mathcal{T}}(\mathbf{\Phi}_{\mathcal{T}})] - \mathbb{E}_{\mathbf{\Phi}_{\mathcal{V}}\sim\mathbb{P}_{\mathcal{V}}}[\mathcal{D}_{\mathcal{T}}(\mathcal{G}_{\mathcal{T}}(\mathbf{\Phi}_{\mathcal{V}}))] \\ + \lambda\mathbb{E}_{\mathbf{\Phi}\sim\mathbb{P}_{\mathcal{VT}}}[(||\nabla_{\mathbf{\Phi}}\mathcal{D}_{\mathcal{T}}(\mathbf{\Phi})|| - 1)^2],\end{aligned} \tag{4.2}$$

where $\mathbb{P}_{\mathcal{T}}$ and $\mathbb{P}_{\mathcal{V}}$ are the uniform distributions defined over $\mathcal{T}$ and $\mathcal{V}$, $\mathbb{P}_{\mathcal{VT}}$ is a distribution created by sampling uniformly along straight lines between the points sampled from the real and the generated text feature distributions.

Similarly, the WGAN loss for $\mathcal{G}_{\mathcal{V}}$ and $\mathcal{D}_{\mathcal{V}}$ is

$$\begin{aligned}\mathcal{L}_{GAN}^{\mathcal{TV}}(\mathcal{G}_{\mathcal{V}}, \mathcal{D}_{\mathcal{V}}) = \mathbb{E}_{\mathbf{\Phi}_{\mathcal{V}}\sim\mathbb{P}_{\mathcal{V}}}[\mathcal{D}_{\mathcal{V}}(\mathbf{\Phi}_{\mathcal{V}})] - \mathbb{E}_{\mathbf{\Phi}_{\mathcal{T}}\sim\mathbb{P}_{\mathcal{T}}}[\mathcal{D}_{\mathcal{V}}(\mathcal{G}_{\mathcal{V}}(\mathbf{\Phi}_{\mathcal{T}}))] \\ + \lambda\mathbb{E}_{\mathbf{\Phi}\sim\mathbb{P}_{\mathcal{TV}}}[(||\nabla_{\mathbf{\Phi}}\mathcal{D}_{\mathcal{V}}(\mathbf{\Phi})|| - 1)^2],\end{aligned} \tag{4.3}$$

where $\mathbb{P}_{\mathcal{TV}}$ is a distribution created by sampling uniformly along straight lines between the points sampled from the real and the generated visual feature distributions.

For the cycle consistency, after generating the fake features $\mathbf{\Phi}_{\mathcal{T}}^{\mathcal{V}\to\mathcal{T}}$ and $\mathbf{\Phi}_{\mathcal{V}}^{\mathcal{T}\to\mathcal{V}}$, we use the generators $\mathcal{G}_{\mathcal{V}}$ and $\mathcal{G}_{\mathcal{T}}$ respectively to recover the original features. The recovered features are $\mathbf{\Phi}_{\mathcal{V}}^{\mathcal{V}\to\mathcal{T}\to\mathcal{V}} = \mathcal{G}_{\mathcal{V}}(\mathbf{\Phi}_{\mathcal{T}}^{\mathcal{V}\to\mathcal{T}})$ and $\mathbf{\Phi}_{\mathcal{T}}^{\mathcal{T}\to\mathcal{V}\to\mathcal{T}} = \mathcal{G}_{\mathcal{T}}(\mathbf{\Phi}_{\mathcal{V}}^{\mathcal{T}\to\mathcal{V}})$. After obtaining the recovered features, we apply a cycle consistency loss between them and the original features. The cycle consistency loss is the cosine distance and can be written as

$$\begin{aligned}\mathcal{L}_{cyc}(\mathcal{G}_{\mathcal{T}}, \mathcal{G}_{\mathcal{V}}) = \mathbb{E}_{\mathbf{\Phi}_{\mathcal{V}} \sim \mathbb{P}_{\mathcal{V}}}[1 - cos(\mathcal{G}_{\mathcal{V}}(\mathcal{G}_{\mathcal{T}}(\mathbf{\Phi}_{\mathcal{V}})), \mathbf{\Phi}_{\mathcal{V}})] \\ + \mathbb{E}_{\mathbf{\Phi}_{\mathcal{T}} \sim \mathbb{P}_{\mathcal{T}}}[1 - cos(\mathcal{G}_{\mathcal{T}}(\mathcal{G}_{\mathcal{V}}(\mathbf{\Phi}_{\mathcal{T}})), \mathbf{\Phi}_{\mathcal{T}})].\end{aligned} \tag{4.4}$$

The overall loss becomes

$$\mathcal{L}(\mathcal{G}_{\mathcal{T}}, \mathcal{D}_{\mathcal{T}}, \mathcal{G}_{\mathcal{V}}, \mathcal{D}_{\mathcal{V}}) = \mathcal{L}_{GAN}^{\mathcal{V}\mathcal{T}}(\mathcal{G}_{\mathcal{T}}, \mathcal{D}_{\mathcal{T}}) + \mathcal{L}_{GAN}^{\mathcal{T}\mathcal{V}}(\mathcal{G}_{\mathcal{V}}, \mathcal{D}_{\mathcal{V}}) + \mathcal{L}_{cyc}(\mathcal{G}_{\mathcal{T}}, \mathcal{G}_{\mathcal{V}}). \tag{4.5}$$

4.2 Experiments

We did our unsupervised experiments on four different datasets and three different tasks. This section presents the training and implementation details for the generator and discriminator networks, the text decoding process during test time, and the qualitative and quantitative results we obtained on each task and dataset. We discuss these results and the reasons for the experiments' success or lack thereof.

4.2.1 Training and Implementation Details

The training setup for the various datasets and the architectural details for the models are discussed in this section. The same generator and discriminator architectures are used for all of the models with changes in the input, hidden and output sizes, which is a 3-layer fully-connected architecture. The input and output sizes correspond to the respective sizes of input and output features $\mathbf{\Phi}_{\mathcal{V}}$ and $\mathbf{\Phi}_{\mathcal{T}}$'s. Batch Normalization is used in the generator architecture whereas Layer Normalization is used for the discriminator. Batch Normalization is not a suitable normalization method for the discriminator because the real and fake examples are fed to the network in separate batches, causing different batch statistics between the two. ReLU is used for the activation function. The discriminator outputs a scalar value $y \in \mathbb{R}$ as we use WGAN [28]. The architectures can be seen in Figure 4.3.

The input, hidden and output sizes used for different datasets of the generators and the discriminators in both directions are given in Table 4.1 and Table 4.2 respectively.

Table 4.1: Input, hidden and output sizes of $\mathcal{G}_{\mathcal{T}}$ and $\mathcal{G}_{\mathcal{V}}$ for different datasets.

	$\mathcal{V} \rightarrow \mathcal{T}$				$\mathcal{T} \rightarrow \mathcal{V}$			
Dataset	i	h_1	h_2	o	i	h_1	h_2	o
METU-VIREF	1024	1024	512	128	128	1024	512	1024
ActivityNet Captions	500	1024	512	1024	1024	1024	512	500
MS-COCO Captions	2048	1024	512	1024	1024	512	1024	2048
Oxford 102 Flowers Captions	2048	1024	512	1024	1024	512	1024	2048

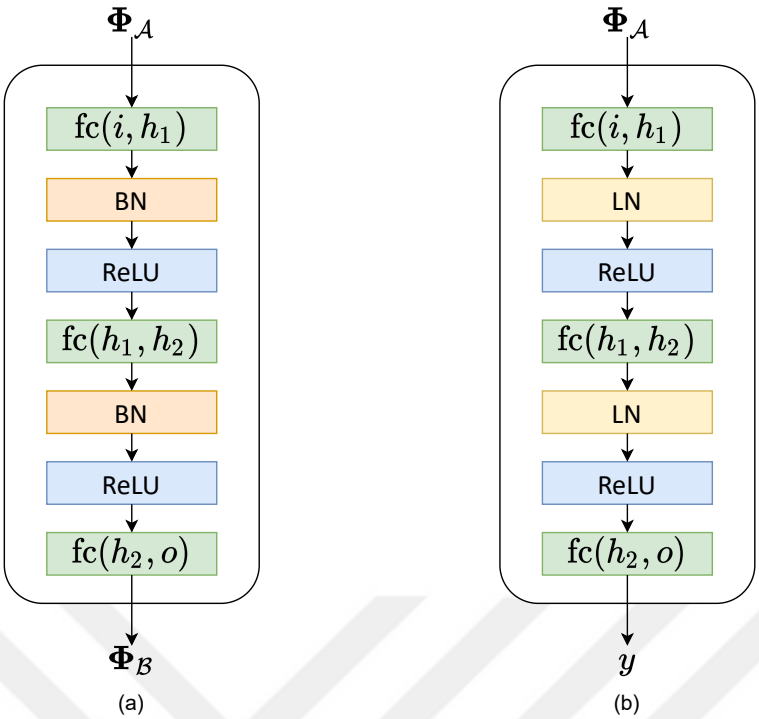


Figure 4.3: (a) Generator and (b) discriminator architectures.

Table 4.2: Input, hidden and output sizes of $\mathcal{D}_{\mathcal{T}}$ and $\mathcal{D}_{\mathcal{V}}$ for different datasets.

	$\mathcal{T}$				$\mathcal{V}$			
Dataset	i	h_1	h_2	o	i	h_1	h_2	o
METU-VIREF [1]	128	1024	512	1	1024	1024	512	1
ActivityNet Captions [26]	1024	1024	512	1	500	1024	512	1
MS-COCO Captions [22]	1024	1024	512	1	2048	1024	512	1
Oxford 102 Flowers Captions [30]	1024	1024	512	1	2048	1024	512	1

During training, the loss in the WGAN-GP paper [29] is used with penalty parameter $\lambda = 10$ as suggested. Hyperparameter optimization showed this number to perform the best for all of the datasets. Also as suggested in the WGAN and WGAN-GP papers, the discriminator has five iterations per one iteration of the generator.

We performed our experiments using the models trained with the learning rate 10^{-4} for all of the datasets. The number of epochs are: METU-VIREF = 200, ActivityNet

Captions = 100, MS-COCO Captions = 10, Oxford 102 Flowers Captions = 200. These were chosen by observing the validation loss, considering the size of the dataset and our time constraints.

4.2.2 Text Generation During Experiments

As explained in Section 4.1.3, we train and use autoencoders to encode text into features $\Phi_{\mathcal{T}}$ for all of the datasets. Since the adapted CycleGAN architecture is trained to translate between $\Phi_{\mathcal{T}}$ and $\Phi_{\mathcal{V}}$ but we want to generate actual captions from the visual features, we need a way to go from the text features to text. For this, we use the decoder part of the autoencoders we trained. Notice that, after their initial training with their respective text corpora, these decoders are only used during test time and not included in the training of the adapted CycleGAN architecture.

4.2.3 Unsupervised Referring Expression Generation from Videos on the METU-VIREF Dataset

Our first unsupervised trials were on the METU-VIREF dataset we have collected and previously worked on with our supervised method. In this part, we performed RE generation on the METU-VIREF dataset without using the existing pairing between the data.

As the generation tests we did were the same as the supervised task, we are able to compare the BLEU-4 [40], METEOR [46] and BERT [47] scores with the ones we obtained with the supervised VIREF and baseline models. The difference in the METEOR scores reported is due to the version used in the paper [1] and Chapter 3 being the previous version of the METEOR score [41], whereas the one used throughout the unsupervised experiments will be the more recent version [46]. Table 4.3 shows the scores obtained from the supervised methods to be much higher than the unsupervised method. This aligns with our expectations considering that the model has a much more complex job in making the connections between two domains when there is no supervision.

Table 4.3: Comparison of models on BLEU-4, METEOR and BERT scores on METU-VIREF dataset. The three columns in the BERT scores correspond to Precision, Recall and F1 scores respectively.

	BLEU-4	**METEOR**	**BERT**			**BERT (rescaled)**		
VIREF-e	0.120	0.237	**0.96**	**0.95**	**0.95**	0.52	0.45	0.48
VIREF-a	0.150	0.262	0.95	0.95	0.95	0.71	0.64	0.67
VIREF	**0.237**	**0.301**	0.95	0.94	0.95	**0.74**	**0.69**	**0.71**
VIREF CycleGAN	0.027	0.169	0.92	0.92	0.92	0.54	0.54	0.54

Upon visual inspection, even though the objects in the generated REs seem to match with the objects in the input videos, the expressions do not describe the attributes and the movement of the objects correctly in most examples. Figure 4.4 shows some videos and the REs generated from them. The REs do not match the videos although they are structurally correct and contain the correct concepts.

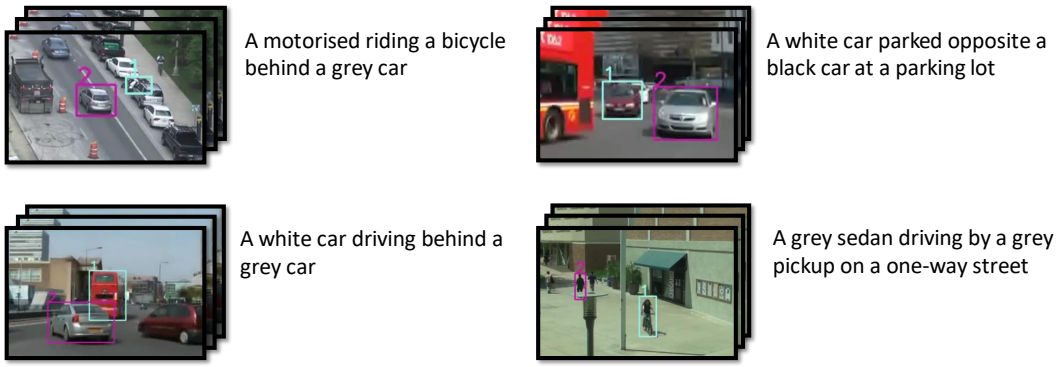
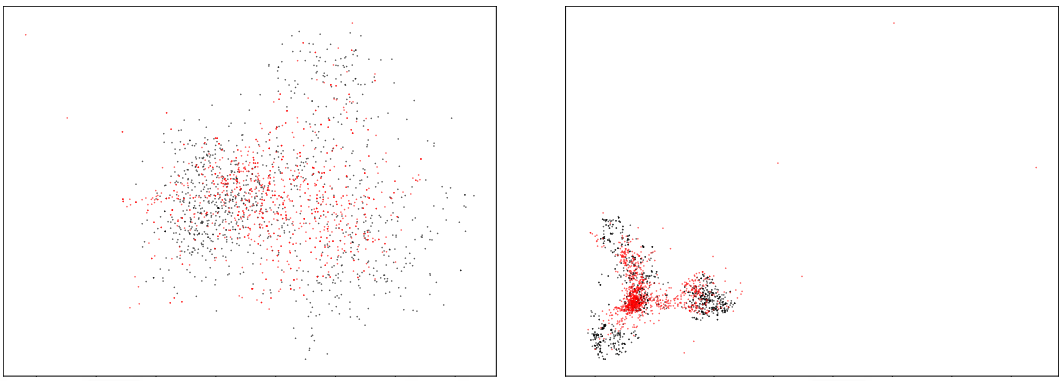


Figure 4.4: Unsupervised referring expression generation results on the METU-VIREF dataset.

To understand whether the generators can capture the real data distributions, we visualized the real and generated features in a 2D space. The 2-dimensional points are obtained using PCA on a set of examples from the real distribution, then applying the same transformation to both the real and generated feature vectors. Figure 4.5 shows the scatter plots of the real and generated data points. In Figure 4.5a, it can be seen that the generated REs are distributed evenly in the RE space. On the contrary, it is apparent in Figure 4.5b that the generated video features fail to cover all of the real

video feature space. Even though the video features are situated in the space where real video features exist too, similar features are generated for different REs. Since information is lost when generating the video features, this may be the reason we were not able to recover the original REs when going back to the text domain.



(a) (b)

Figure 4.5: Feature distribution scatter plots from the METU-VIREF dataset. The red points correspond to generated features whereas the black points are the features obtained from real data. (a) Referring expression features PCA scatter plot. (b) Video features PCA scatter plot.

4.2.4 Unsupervised Video Captioning on the ActivityNet Caption Dataset

One of the possible reasons for the problems we had in unsupervised RE generation on the METU-VIREF dataset could be the small amount of data, which does not provide the model with enough examples to learn a meaningful mapping between the data without supervision.

Another reason might be that the number of video sequences that can correspond to the same RE is very high. This is due to the REs not containing enough details to completely describe the videos and hence being able to be translated into more than one video. This type of correspondence is not necessarily bad but was not suitable for our architecture because it made it impossible to complete the cycle by going back to the video domain from the generated captions. This made us decide to use a dataset and problem with more descriptive text.

A dataset that fits the need for more descriptive text was the ActivityNet Captions [26] dataset which is one of the largest datasets for video captioning, each video having several sentences that describe it in detail. Some examples from the dataset are shown in Figure 4.6.

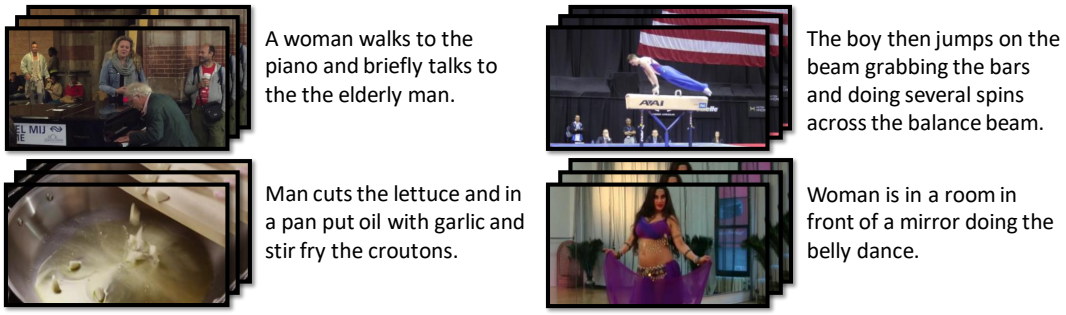


Figure 4.6: Examples from the ActivityNet Captions dataset. Taken from [26].

The feature extraction process for the videos were performed as discussed in Section 4.1.2.2. Because we did not use the whole videos but smaller video sequences that correspond to one sentence, our experimental settings were different than other works that use this dataset. Due to this, we were not able to compare our quantitative results with other works.

To have a point of comparison, we trained a network for the same task in a supervised way. As our supervised model, we used the same text generator architecture as shown in 4.2.1 and trained it to generate text features from video features. We used Mean Squared Error loss between real and generated text features. Then, we used the same decoder to obtain captions from the features. The results of the supervised and unsupervised trials are shown in Table 4.4.

The scores for both the supervised and unsupervised trials were very low, the unsupervised results understandably being lower than the supervised ones for all columns. Even the supervised method not being able to achieve success on this dataset shows the task to be quite difficult. Qualitatively, although the generator was able to capture some of the concepts in the videos such as sports, the generated captions did not contain most of the activities in the dataset. Some of the sentences were also structurally incorrect. Some examples can be seen in Figure 4.7.

Table 4.4: Comparison of the unsupervised adapted CycleGAN architecture for the video captioning task on ActivityNet Captions dataset to our supervised trial on the same task. The three columns in the BERT scores correspond to Precision, Recall and F1 scores respectively.

	BLEU-4	**METEOR**	**BERT**			**BERT (rescaled)**		
Supervised trial	**0.0004**	**0.069**	**0.87**	**0.87**	**0.87**	**0.20**	**0.23**	**0.21**
ActivityNet CycleGAN	0.0003	0.049	0.85	0.86	0.86	0.12	0.17	0.15

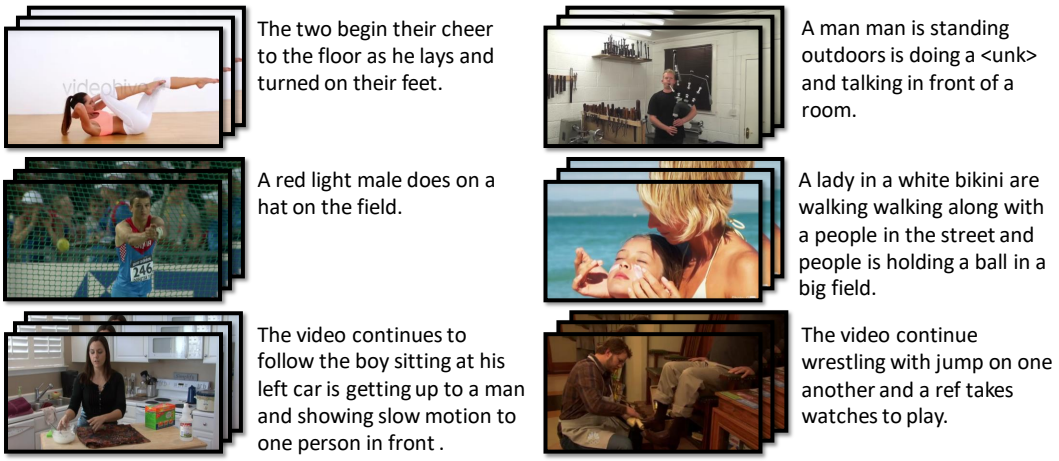
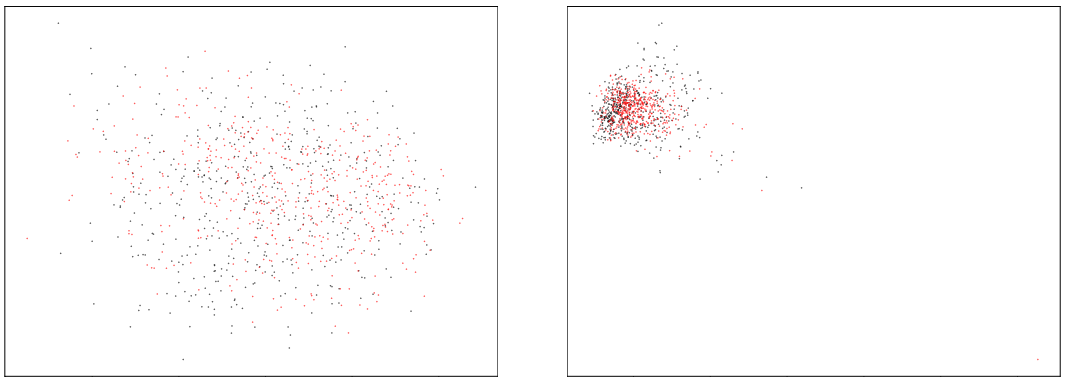


Figure 4.7: Unsupervised caption generation results on the ActivityNet Captions dataset.

Our trials on this dataset were unsuccessful on the whole, presumably due to just how complex both the videos and the captions are. Especially compared to the METU-VIREF dataset, the amount of detail contained in the captions is much bigger than in REs. Also since the video encoder we used, C3D [43], is trained on the activity classification task, the video features might not contain enough information to be able to generate dense captions from. It would have been better to use a more recent feature extractor, however, it was not possible to download the raw videos with our resources. A better feature extractor would likely help the results.

Figure 4.8 shows PCA scatter plots of real and fake features. Figure 4.8b shows the generated video features to be well-distributed in the video feature domain. However, this does not mean that these features are the correct ones that correspond to the

captions they were generated from, and the same goes for the text domain as can be seen in Figure 4.8a. Even though the generated captions seem to span the whole caption space, they did not describe the input videos correctly.



(a) (b)

Figure 4.8: Feature distribution scatter plots from the ActivityNet Captions dataset. The red points correspond to generated features whereas the black points are the features obtained from real data. (a) Caption features PCA scatter plot. (b) Video features PCA scatter plot.

4.2.5 Unsupervised Image Captioning

Based on our analyses on the ActivityNet dataset [26], we hypothesized that working in a less complex domain and using a better feature extractor would be effective. Taking the temporal aspect out of the data and doing image captioning seemed to be the next logical step. We aimed for our findings from this simpler experiment to guide us to achieve a better training for the ActivityNet dataset.

4.2.5.1 Microsoft COCO Captions Dataset

As our image captioning dataset we chose MS-COCO Captions [22] as it is one of the largest datasets for this problem and has very detailed captions that describe the images well, which we thought might help guide the unsupervised training. Some examples from this dataset are shown in Figure 4.9.

Figure 4.9: Examples from the MS-COCO Captions dataset. Taken from Chen *et al.* [22].

Table 4.5 shows the BLEU-4 and METEOR scores for the adapted CycleGAN architecture in comparison to the other unsupervised architectures. Both of the scores are very low compared to the other methods, with the METEOR score being somewhat more comparable to the others.

Table 4.5: Comparison of the adapted CycleGAN architecture for the image captioning task on MS-COCO Captions dataset to other unsupervised methods. The three columns in the BERT scores correspond to Precision, Recall and F1 scores respectively.

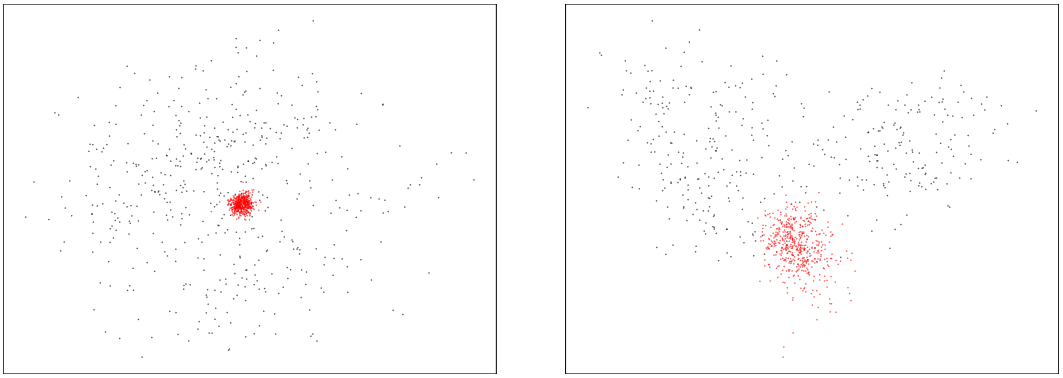
	BLEU-4	**METEOR**
Gu *et al.* [32]	**0.215**	**0.209**
Feng *et al.* [33]	0.186	0.179
Laina *et al.* [34]	0.193	0.202
Guo *et al.* [35]	0.083	0.140
MS-COCO CycleGAN	0.009	0.095

The results were not only quantitatively low but the generated features also reflected this. Especially in the generated captions, the mode collapse problem could visually be observed by the fact that the generator only tended to generate a few select sentences containing different concepts and nothing else, and they did not seem to depend on the input image. The examples in Figure 4.10 show the mode collapse of

the caption generator.

Figure 4.10: Unsupervised caption generation results on the MS-COCO Captions dataset.

Figure 4.11 clearly shows the mode collapse problem in both caption and image feature generation. As a result, we were not able to achieve any success on this dataset. It might be the case that even though image captioning is a simpler problem than video captioning, the MS-COCO Captions dataset was quite complex, having a very broad domain. Our experiments showed the mode collapse problem to be too severe to generate any meaningful captions from this dataset. However, with a better solution against mode collapse, we might be able to obtain better results in this task.



(a) (b)

Figure 4.11: Feature distribution scatter plots from the MS-COCO Captions dataset. The red points correspond to generated features whereas the black points are the features obtained from real data. (a) Image features PCA scatter plot. (b) Video features PCA scatter plot.

4.2.5.2 Oxford 102 Flowers Captions Dataset

Our experiments with the previous three datasets left us questioning whether this problem could actually be solved in an unsupervised way. We decided to use a very simple image and caption dataset to give the model its best chance. The dataset we chose was the Oxford 102-flowers dataset [25], having a very limited domain that contains flower pictures and their detailed descriptions. Figure 4.12 shows some examples from the dataset.

The dataset has not previously been used for image captioning but is usually used to generate images from the descriptions. The work of Gorti *et al.* [30] performs this task in an unsupervised way using an adapted CycleGAN architecture. The success they achieved with their work motivated us to approach the image captioning task the other way around.

Our experiments on this dataset are not quantitatively comparable to other works because they do not work on the same problem. However, for comparison, we trained a supervised model in the same way we did for the ActivityNet Captions model in Section 4.2.4. Table 4.6 shows the results of the supervised and unsupervised experi-

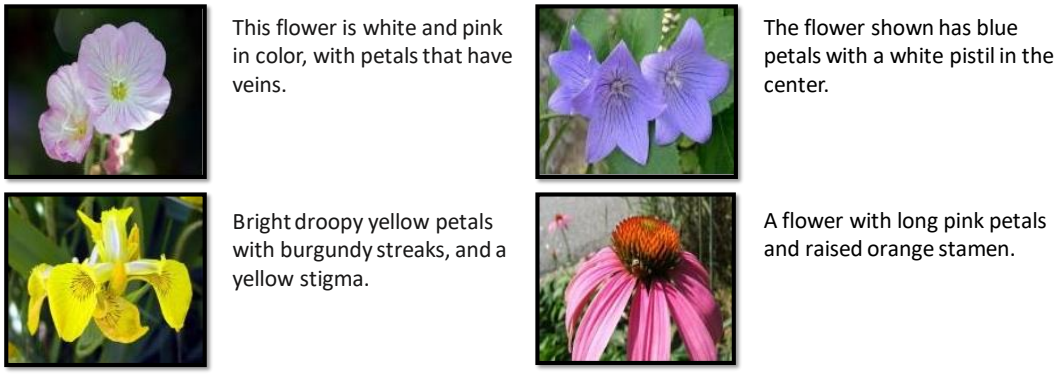


Figure 4.12: Examples from the Oxford 102 Flowers Captions dataset. Taken from Reed *et al*. [24].

ments.

Table 4.6: Comparison of the unsupervised adapted CycleGAN architecture for the image captioning task on Oxford 102 Flowers Captions dataset to our supervised trial on the same task. The three columns in the BERT scores correspond to Precision, Recall and F1 scores respectively.

	BLEU-4	**METEOR**	**BERT**			**BERT (rescaled)**		
Supervised trial	0.028	**0.303**	**0.93**	**0.94**	**0.94**	**0.60**	**0.66**	**0.62**
Flowers CycleGAN	**0.141**	0.241	0.92	0.92	0.92	0.50	0.53	0.50

Except for the BLEU-4 scores, the supervised model has achieved better scores as expected. However, both models have achieved quite high scores considering the dataset and the image captioning problem. Visual inspection also showed the unsupervised results to be satisfactory, the majority of generated descriptions containing the right attributes and colors corresponding to the input images. However, there are a significant number of cases where the model generates captions that do not describe the image or describe it only partially. Some results are shown in Figure 4.13.

The scatter plots in Figure 4.14 support the qualitative and quantitative results we obtained, the fake features being distributed throughout the feature space for both image and text domains. All in all, we obtained promising results for the image captioning task on the Oxford 102 Flowers Captions dataset.

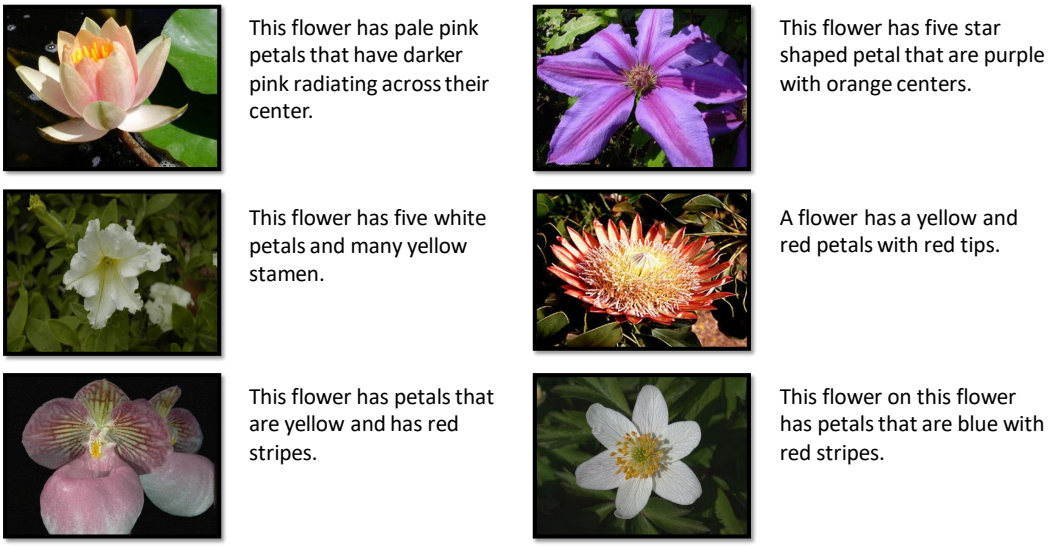
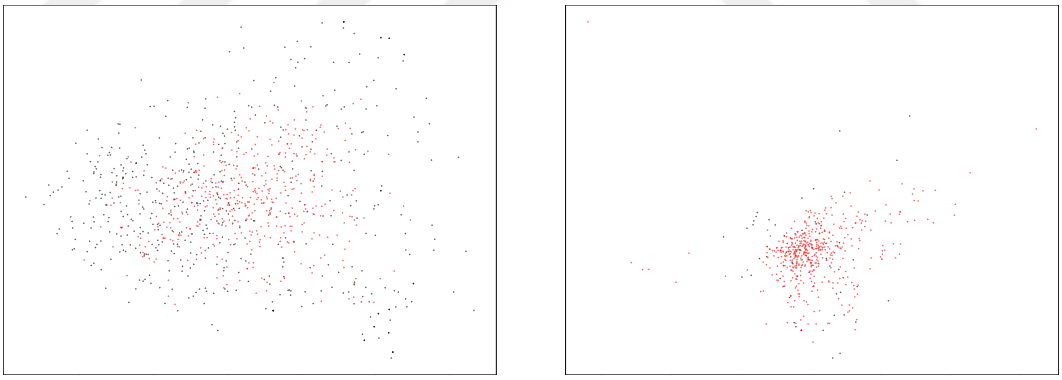


Figure 4.13: Unsupervised caption generation results on the Oxford 102 Flowers Captions dataset.



(a) (b)

Figure 4.14: Feature distribution scatter plots from the Oxford 102 Flowers Captions dataset. The red points correspond to generated features whereas the black points are the features obtained from real data. (a) Caption features PCA scatter plot. (b) Image features PCA scatter plot.

CHAPTER 5

CONCLUSION

This chapter is partially taken from our paper, Anayurt *et al*. [1].

This work consists of two parts: Supervised Referring Expression Generation and Comprehension from Videos, and Unsupervised Text Generation from Images or Videos.

In the first part of the work, we addressed linking objects in videos with relational referring expressions. For this, we first collected a dataset of REs for a subset of videos from VIRAT and ILSVRC datasets. The videos we specifically chose include highly ambiguous settings with numerous occurrences of objects.

Moreover, we proposed an LSTM encoder-decoder architecture with which we can both comprehend an RE (i.e. identify the matching object pair in a video) and generate an RE identifying a pair of objects in a video. At each decoding stage, our model can attend to the features in the encoding stage. Compared with the two baselines that we have developed, our model performs significantly better in both generation and comprehension tasks.

In the second part of the work, we aimed to solve the problem of referring expression or caption generation from visual data in an unsupervised way. With this, we aimed to overcome the bottleneck of the amount of paired data present in supervised learning. With this purpose, we adapted the popular CycleGAN architecture by Zhu *et al*. [10] for unsupervised translation between domains to work between visual and textual data.

We trained and experimented on several different datasets using this method and

achieved different levels of success. In our experiments, we observed that the problems we were working on were quite complex to be solved in a completely unsupervised way. In some of our experiments, the visual domain was not constrained in any way and there was no guidance for the feature extraction process which made it difficult to find a clear match between the domains.

During our trials with different datasets, we encountered various challenges. The small amount of data and the lack of detail in the referring expressions of the METU-VIREF dataset could have been the reason that caused the method to have a hard time reconstructing complete video features from them. Moreover, finding a mapping between the domains might be a complex task to be solved without supervision when both the visual and text domains are as broad as they are in the ActivityNet Captions and MS-COCO Captions datasets.

Even though unconstrained domains make it harder for this method to work, we saw that it can be applied to simpler domains with success. Our experimental results on the Oxford 102 Flowers Captions dataset showcased this. On this dataset containing only flower images and their detailed descriptions, we were able to achieve promising results. This suggests that if we can solve the problem of mode collapse on the METU-VIREF and MS-COCO Captions datasets, we may obtain captions with better correlation to the input visual data as well.

5.1 Limitations and Future Work

In the second part of our work, we had certain limitations during training that may have caused the results we obtained to be unsatisfactory. For the METU-VIREF dataset, these were the lack of data and the ambiguity of the dataset which could have caused the model to be unable to find a clear mapping between the data despite being able to generate structurally plausible referring expressions. These limitations stem from the nature of the dataset we collected, and therefore, cannot be solved without changing our dataset.

In the ActivityNet Captions dataset, the model seemed to only be able to pick up on vague concepts and not comprehend what was happening in the videos. We suspect

that the pretrained C3D [43] network we used to extract video features being trained on the activity classification task is the culprit for this, as any information beyond the activity class is likely to be discarded by the network. If this is the case, the generator would not have enough information to generate a caption that describes the whole scene. If we can use a better way to extract features, the model might achieve better results since even with this limitation some concepts were conserved during translation.

In our experiments on the MS-COCO Captions dataset, we saw a severe mode collapse problem. This should be able to be overcome by employing different techniques during training. As a possible solution, we changed the architecture and loss to be WGAN-GP [29], since this has been shown to ease the training of GANs and help in overcoming various problems. However, this did not help in solving this problem. Even though we tried some of the training tricks mentioned in the literature as well, there are other approaches that can be taken to help. Our results in the other three datasets implicate that solving the mode collapse problem might have a positive outcome.

Beside the various limitations in training, the unsupervised translation between two domains that are very different from each other is a very complex problem. Looking at other unsupervised works on similar problems and considering the nature of the difficulties we encountered, it is apparent that a way to bring these two domains closer on some middle ground would help immensely. As future work, projecting the visual and text features onto a common domain that is more closely connected to both seems to be the next logical step.

Some unsupervised captioning works have used scene graphs as the middle ground between visual and text data. Having the properties of containing all of the object, attribute and relation details of a scene but representing these in lingual words inside of a graph, scene graphs seem to be a good candidate for our task. A similar method to Gu *et al*. [32] might be a good future direction for the video domain as well.

In conclusion, even though we were not able to overcome the limitations we mentioned, there are still methods that can be tried. The results we have obtained so far show that there is potential to achieve success in this problem.

REFERENCES

[1] H. Anayurt, S. A. Ozyegin, U. Cetin, U. Aktas, and S. Kalkan, "Searching for ambiguous objects in videos using relational referring expressions," in *30th British Machine Vision Conference 2019, BMVC 2019, Cardiff, UK, September 9-12, 2019*, p. 272, BMVA Press, 2019.

[2] M. Z. Hossain, F. Sohel, M. F. Shiratuddin, and H. Laga, "A comprehensive survey of deep learning for image captioning," *ACM Computing Surveys (CsUR)*, vol. 51, no. 6, pp. 1–36, 2019.

[3] R. Dale, "Cooking up referring expressions," in *Proceedings of the 27th Annual Meeting on Association for Computational Linguistics*, 1989.

[4] J. Viethen and R. Dale, "The use of spatial relations in referring expression generation," in *Proceedings of the Fifth International Natural Language Generation Conference*, pp. 59–67, Association for Computational Linguistics, 2008.

[5] V. K. Nagaraja, V. I. Morariu, and L. S. Davis, "Modeling context between objects for referring expression understanding," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 792–807, Springer, 2016.

[6] A. Balajee Vasudevan, D. Dai, and L. Van Gool, "Object referring in videos with language and human gaze," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4129–4138, 2018.

[7] A. Khoreva, A. Rohrbach, and B. Schiele, "Video object segmentation with referring expressions," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 7–12, Springer, 2018.

[8] P. Wiriyathammabhum, A. Shrivastava, V. Morariu, and L. Davis, "Referring to objects in videos using spatio-temporal identifying descriptions," 04 2019.

[9] S. Islam, A. Dash, A. Seum, A. H. Raj, T. Hossain, and F. M. Shah, "Exploring

video captioning techniques: A comprehensive survey on deep learning methods," *SN Computer Science*, vol. 2, no. 2, pp. 1–28, 2021.

[10] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, pp. 2223–2232, 2017.

[11] G. Lample, A. Conneau, L. Denoyer, and M. Ranzato, "Unsupervised machine translation using monolingual corpora only," in *International Conference on Learning Representations*, 2018.

[12] S. Oh, A. Hoogs, A. Perera, N. Cuntoor, C.-C. Chen, J. T. Lee, S. Mukherjee, J. K. Aggarwal, H. Lee, L. Davis, E. Swears, X. Wang, Q. Ji, K. Reddy, M. Shah, C. Vondrick, H. Pirsiavash, D. Ramanan, J. Yuen, A. Torralba, B. Song, A. Fong, A. Roy-Chowdhury, and M. Desai, "A large-scale benchmark dataset for event recognition in surveillance video," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3153–3160, IEEE, 2011.

[13] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[14] J. Mao, J. Huang, A. Toshev, O. Camburu, A. L. Yuille, and K. Murphy, "Generation and comprehension of unambiguous object descriptions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11–20, 2016.

[15] S. Kazemzadeh, V. Ordonez, M. Matten, and T. Berg, "Referitgame: Referring to objects in photographs of natural scenes," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 787–798, 2014.

[16] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[17] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-

scale image recognition," in *Proceedings of 3rd International Conference on Learning Representations (ICLR)*, 2015.

[18] L. Yu, Z. Lin, X. Shen, J. Yang, X. Lu, M. Bansal, and T. L. Berg, "Mattnet: Modular attention network for referring expression comprehension," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[19] J. Liu, W. Wang, L. Wang, and M.-H. Yang, "Attribute-guided attention for referring expression generation and comprehension," *IEEE Transactions on Image Processing*, vol. 29, pp. 5244–5258, 2020.

[20] S. Yang, G. Li, and Y. Yu, "Graph-structured referring expression reasoning in the wild," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[21] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung, "A benchmark dataset and evaluation methodology for video object segmentation," in *Computer Vision and Pattern Recognition*, 2016.

[22] X. Chen, H. Fang, T.-Y. Lin, R. Vedantam, S. Gupta, P. Dollár, and C. L. Zitnick, "Microsoft coco captions: Data collection and evaluation server," *arXiv preprint arXiv:1504.00325*, 2015.

[23] M. Stefanini, M. Cornia, L. Baraldi, S. Cascianelli, G. Fiameni, and R. Cucchiara, "From show to tell: A survey on image captioning," *arXiv preprint arXiv:2107.06912*, 2021.

[24] S. Reed, Z. Akata, H. Lee, and B. Schiele, "Learning deep representations of fine-grained visual descriptions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 49–58, 2016.

[25] M.-E. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pp. 722–729, IEEE, 2008.

[26] R. Krishna, K. Hata, F. Ren, L. Fei-Fei, and J. Carlos Niebles, "Dense-captioning events in videos," in *Proceedings of the IEEE international conference on computer vision*, pp. 706–715, 2017.

[27] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[28] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International conference on machine learning*, pp. 214–223, PMLR, 2017.

[29] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of wasserstein gans," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, p. 5769–5779, 2017.

[30] S. K. Gorti and J. Ma, "Text-to-image-to-text translation using cycle consistent adversarial networks," *arXiv preprint arXiv:1808.04538*, 2018.

[31] J. Gu, S. Joty, J. Cai, and G. Wang, "Unpaired image captioning by language pivoting," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 503–519, 2018.

[32] J. Gu, S. Joty, J. Cai, H. Zhao, X. Yang, and G. Wang, "Unpaired image captioning via scene graph alignments," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10323–10332, 2019.

[33] Y. Feng, L. Ma, W. Liu, and J. Luo, "Unsupervised image captioning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4125–4134, 2019.

[34] I. Laina, C. Rupprecht, and N. Navab, "Towards unsupervised image captioning with shared multimodal embeddings," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7414–7424, 2019.

[35] D. Guo, Y. Wang, P. Song, and M. Wang, "Recurrent relational memory network for unsupervised image captioning," *arXiv preprint arXiv:2006.13611*, 2020.

[36] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, pp. 91–99, 2015.

[37] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," *Advances in neural information processing systems*, vol. 29, pp. 2234–2242, 2016.

[38] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014.

[39] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 4489–4497, 2015.

[40] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pp. 311–318, Association for Computational Linguistics, 2002.

[41] A. Lavie and A. Agarwal, "Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments," in *Proceedings of the Second Workshop on Statistical Machine Translation*, pp. 228–231, Association for Computational Linguistics, 2007.

[42] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization. iclr 2015," *arXiv preprint arXiv:1412.6980*, vol. 9, 2015.

[43] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 1, pp. 221–231, 2012.

[44] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[45] T. Shen, J. Mueller, R. Barzilay, and T. Jaakkola, "Educating text autoencoders: Latent representation guidance via denoising," in *International Conference on Machine Learning*, pp. 8719–8729, PMLR, 2020.

[46] M. Denkowski and A. Lavie, “Meteor universal: Language specific translation evaluation for any target language,” in *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*, 2014.

[47] T. Zhang*, V. Kishore*, F. Wu*, K. Q. Weinberger, and Y. Artzi, “Bertscore: Evaluating text generation with bert,” in *International Conference on Learning Representations*, 2020.