University of West Bohemia
Faculty of Applied Sciences

Department of Cybernetics

Image Captioning using Deep Learning

Master Thesis

Bc. Tomáš Železný

Supervisor: Ing. Marek Hrúz, Ph.D.

Pilsen, 2022

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2021/2022

ZADÁNÍ DIPLOMOVÉ PRÁCE
(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Tomáš ŽELEZNÝ**
Osobní číslo:	**A20N0041P**
Studijní program:	**N3918 Aplikované vědy a informatika**
Studijní obor:	**Kybernetika a řídicí technika**
Téma práce:	**Popis obrázků pomocí metod hlubokého učení**
Zadávající katedra:	**Katedra kybernetiky**

Zásady pro vypracování

1. Seznamte se s metodami popisu obrázků - tzv. image captioning.
2. Zvolte vhodnou metodu a implementujte ji.
3. Otestujte tuto metodu na standardní datové sadě.
4. Shrňte dosažené výsledky, případně navrhněte vylepšení.

Rozsah diplomové práce: **40 – 50 stránek A4**
Rozsah grafických prací:
Forma zpracování diplomové práce: **tištěná**
Jazyk zpracování: **Angličtina**

Seznam doporučené literatury:

- Li, X., Yin, X., Li, C., Zhang, P., Hu, X., Zhang, L., ... & Gao, J. (2020, August). Oscar: Object-semantics aligned pre-training for vision-language tasks. In *European Conference on Computer Vision* (pp. 121-137). Springer, Cham.
- Cui, Y., Yang, G., Veit, A., Huang, X., & Belongie, S. (2018). Learning to evaluate image captioning. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 5804-5812).

Vedoucí diplomové práce: **Ing. Marek Hrúz, Ph.D.**
Výzkumný program 1

Datum zadání diplomové práce: **1. října 2021**
Termín odevzdání diplomové práce: **23. května 2022**

Západočeská univerzita v Plzni
Fakulta aplikovaných věd
L.S.

Doc. Ing. Miloš Železný, Ph.D.
děkan

Prof. Ing. Josef Psutka, CSc.
vedoucí katedry

V Plzni dne 1. října 2021

DECLARATION

Hereby I submit for assessment and defense the master thesis prepared at the end of my studies at the Faculty of Applied Sciences of the University of West Bohemia in Pilsen.

I declare that I made this master thesis myself, exclusively using scientific literature and sources, the complete list of which is a part of it.

PROHLÁŠENÍ

Předkládám tímto k posouzení a obhajobě diplomovou práci zpracovanou na závěr studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí.

In Pilsen, date
V Plzni, dne .....

ACKNOWLEDGEMENTS

Firstly, I'd like to express my thanks to my supervisor Ing. Marek Hrúz, Ph.D. for his professional approach and the time he devoted to me. Without his supervision, this thesis would be hard or even impossible to finish. I would also like to thank Ing. Ivan Gruber, Ph.D for his advice on the detection networks and throughout the work generally. I would also like to thank Ing. Jiří Vyskočil for sharing with me his experience in the field of ablation studies.

I am also very grateful to my family and friends for their unceasing support and constant encouragement.

In addition, I would like to thank Metacentrum for providing the computing capacity. Computational resources were supplied by the project "e-Infrastruktura CZ" (e-INFRA CZ LM2018140) supported by the Ministry of Education, Youth and Sports of the Czech Republic.

Abstract

In this work, I discuss an automatic image captioning technique based on an existing method Oscar. Using a Faster-R-CNN detection network, I pre-process the images so that they can be further used by Oscar. By combining these two methods, I create a pipeline that allows me to generate a caption for any image. I evaluate its performance using metrics BLEU-4: 0.312, METEOR: 0.272, CIDEr: 1.02, and SPICE: 0.201, which is a drop from the original performance. Thus, I further discuss the causes in this work. Within the ablation study, I investigate the impact of individual modalities of Oscar. The results of the experiment suggest that Oscar is dependent on both modalities, with the visual modality predominating. In the end, I discuss the interesting cases of the behavior when the captioning pipeline is supposed to generate captions for images with objects unknown to it.

Key words

Image captioning, deep learning, computer vision, machine learning, object detection

Anotace

V této práci se zabývám technikou automatického popisu obrázků, založenou na existující metodě Oscar. Pomocí detekční sítě Faster-R-CNN vhodně předzpracovávám obrázky tak, aby mohly být dále použity metodou Oscar. Spojením těchto dvou metod vytvářím systém, který umožňuje vygenerování popisku pro libovolný obrázek. Tento systém je poté vyhodnocen na metrikách BLEU-4: 0.312, METEOR: 0.272, CIDEr: 1.02, a SPICE: 0.201, což je pokles oproti původním. V práci se tak dále zabývám důvody, které k tomu vedly. V rámci ablační studie se věnuji zkoumání závislosti jednotlivých modalit metody Oscar. Výsledky experimentu naznačují že Oscar je závislý na obou modalitách, vizuální modalita převažuje. V závěru práce diskutuji různé případy chování mého popisovacího systému, kdy měl generovat popisky k obrázkům s pro něj neznámými objekty.

Klíčová slova

Popis obrázků, hluboké učení, počítačové vidění, strojové učení, detekce objektů

Contents

Chapter 1

Introduction

Image captioning is the process of generating a description of the given image. Humans can recognize at a glance which objects are important in the picture. They continue to look for semantics, i. e. relationships between objects, what they do, and so on. Humans can even imagine the context of the image - what happened before and after it was captured. Captioning the image is used for many purposes including interpersonal communication, searching in databases, or screen readers. In many cases, there are huge datasets that we want to describe. As the interest in modern technology spreads, the number and volume of these datasets are increasing. Therefore, it is not feasible to describe them manually by humans. In these cases, we want the image to be described automatically.

Automatic image captioning is a very important task in many fields. It can be used for automatic image indexing. Image indexing may be used by search engines, digital libraries, social media platforms, and so on. Image captioning can also help people with vision disabilities to better understand their surroundings. It can be also used in the medical [1] or military [2] fields.

The aim of the image captioning task is to automatically generate a natural language sentence that describes the content of the input image. It is a very complex task. Image captioning system needs in some form to detect all the objects in the image, where they are, and what are their relations with other objects. Then it summarizes all this information and generates both semantically and syntactically correct sentence. For example, some of the methods use an object detector to detect them followed by a caption generator, which translates them into a sentence. An example can be seen in the Figure 1.1. When people describe the images, everyone considers the details in the picture to be of different importance. Because of this factor, each individual may generate a different description for the same image. This is a very important feature of this task that makes it ambiguous.

The scientific community has been interested in this task for over a decade. Many different approaches have been discovered. Some of the early methods use standard machine learning and computer vision techniques, but in recent years deep learning has been mainly

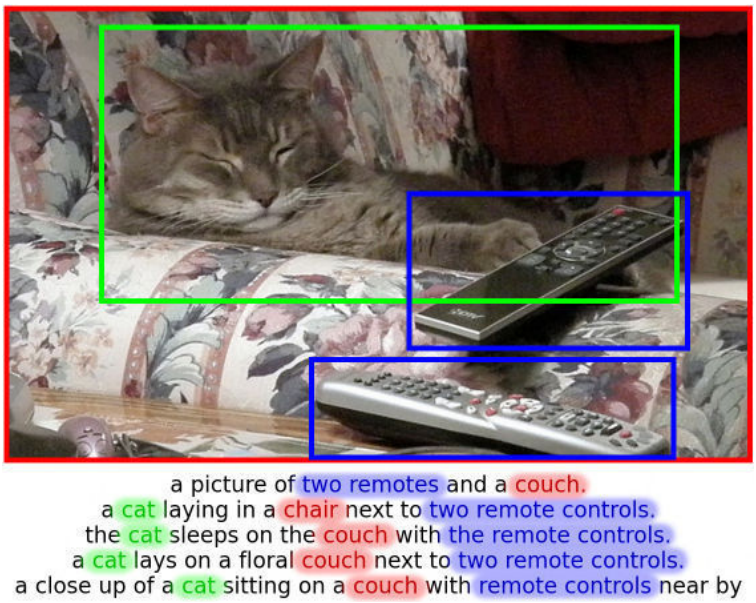

Figure 1.1: Image being described by five different but correct captions. Objects that are important for generating the captions are colored. Courtesy of [3].

used. In this work, I am replicating experiments of the current state-of-the-art. This way I want to get new experience in this field, understand how image captioning task is being solved, and get knowledge about deep learning in general. I replicate the results of the existing model with the given dataset. Furthermore, I extend this task so that I am able to generate captions for custom images, which are not part of the dataset.

Chapter 2

Datasets

Image captioning methods, whether they use traditional techniques or deep-learning-based ones, need a sufficient amount of data for their training. Datasets for image captioning task consist of image-caption pairs. There is a large amount of these datasets. Each of them was created in its individual way. They vary in the domain they are used in and in a number of images they consist of. The very important feature of the image captioning task is that there is not only one correct description of a single image. Therefore, datasets often contain more than one caption for a specific image. Individual datasets also vary in a number of these captions. Even the caption may be written in a different type of format. Datasets also vary in the approach, of how their captions were made and with which rules. In this chapter, I individually discuss some of the most popular datasets that are used for the image captioning task.

2.1 Flickr30k

Flickr30k [4] is a dataset that contains 31,000 images. It was created in 2013 as an extension of the previous dataset Flickr8k [5] that contains 8,000 images. Dataset consists of pictures of everyday scenes taken from the online photo-sharing application Flickr[1]. Flickr30k is one of the most popular datasets used for Image captioning with over 1,400 citations. It is used for Image Captioning [6, 7], Image Retrieval [8, 9] and Cross-Modal Retrieval [8, 10] tasks.

Captions for this dataset were created by human annotators. Each of the images was described by five independent people. The total size of Flickr30k is thus over 150,000 image-caption pairs.

[1] `https://www.flickr.com/`

Figure 2.1: Data from Flickr30k dataset [4].

2.2 COCO Caption

Microsoft COCO Caption Dataset [11] is a very popular dataset used for the tasks of Image Captioning [12, 13] and Text Generation [14, 15]. It was created by captioning images in COCO Dataset [3]. It is a large dataset that contains images of everyday scenes taken from Flickr. The images were gathered by searching for pairs of objects in their natural context. The first version of the dataset was released in 2014, containing over 164,000 images split into training (83,000), validation (41,000), and test (41,000) sets. The second version was created in 2015 by extending the test set with 40,000 additional images. Based on community feedback in 2017, the third version was created by redistributing training (118,000) and validation (5,000) data. The 2017 test set is a subset of 41,000 images of the 2015 test set. Additionally, COCO provides another set containing 123,000 unannotated images.

Captions for this dataset were created by human annotators. To ensure a consistent format of the captions, annotators were asked to comply with the following rules:

- Describe all the important parts of the scene.
- Do not start the sentences with "There is".
- Do not describe unimportant details.
- Do not describe things that might have happened in the future or past.
- Do not describe what a person might say.
- Do not give people proper names.
- The sentences should contain at least 8 words.

Figure 2.2: Data from Microsoft COCO Caption dataset [11].

Two datasets were collected, c5 and c40. COCO c5 contains five captions for each of the images in the COCO dataset. COCO c40 contains forty captions for the randomly chosen 5,000 images from the COCO testing dataset. The total number of image-caption pairs in the COCO Caption dataset is thus over 1,000,000.

2.3 Conceptual Captions

Conceptual Captions [16] is a dataset used for Image Captioning [17] that was introduced in 2018. It contains images that were collected from a huge quantity of web pages. The captions were collected from the Alt-text HTML attribute associated with the images. Authors then used the pipeline they developed to extract, filter, and transform the candidate image-caption pairs, with the goal of achieving cleanliness, informativeness, fluency, and learnability of the resulting captions. The total number of image-caption pairs is over 3,000,000. Thanks to the unique approach, Conceptual Captions provide much more variety both in images and captions than the Microsoft COCO dataset.

2.4 Multi30k

Multi30k [18] is a dataset, that was created to stimulate multilingual multimodal research for English and German [19, 20]. It was introduced in 2016 as an extension of the Flickr30k dataset [4]. Multi30k consists of two different datasets. The first dataset was created by choosing one of the five English captions for each image, that Flickr30k provides. Then it was translated by a professional translator into German. The first dataset thus consists of 31,000 triplets of an image, English caption, and German caption. The second dataset was created by generating five German captions for each image, independently of the original

Figure 2.3: Data from Conceptual Captions dataset [16].

English captions and also independently of each other. The second set thus contains over 150,000 image-caption pairs in English and another 150,000 image-caption pairs in German.

2.5 Hateful Memes

Hateful Memes [21] is a dataset introduced in 2020. It is used for a task of multimodal classification with a focus on detecting hate speech in multimodal memes [22]. The goal of the task is the binary classification of whether the meme contains hate speech or not. The source meme images were gathered from public social media groups. Then the non-English and violent memes were filtered out. The rest of the memes were reconstructed by using different, stock images and keeping the original text. Reconstructed images were then labeled as hateful or not. For the subset of the memes, that were labeled as hateful, were created memes with a different image or caption, which led to a change in meaning so that they were not hateful. The total size of the Hateful Memes dataset is over 10,000 memes, including non-hateful, hateful, and modified hateful memes.

2.6 Localized Narratives

Localized Narratives [23] is a multimodal dataset that was introduced in 2020. The images are described by the annotators with their voices while simultaneously hovering the mouse cursor over the described region of the image. The dataset is used for the task of Image Captioning [24]. Localized Narratives dataset consists of images originating from COCO [3], Flickr30k [4] and other datasets making it 849,000 images in total.

Figure 2.4: Data from Multi30k dataset [18]. On the left: one of the English captions is translated into German by a professional translator. On the right: five English captions and another five independent captions in German.

Figure 2.5: Data from Hateful Memes dataset [21].

Figure 2.6: Data from Localized Narratives dataset [23].

2.7 Conclusion

There are many datasets for image captioning. They vary in the category, size, and data format. Different format of captions is created by applying different rules when a dataset is created. A unified data format helps the machine to produce better results. The negative feature is that applying any rules to captions leads to inductive bias. Although the resulting model will be more accurate, it will be limited to the rules that were applied while the dataset was created. Data that are based on different rules, or not based on the rules at all, will produce worse results. When people describe an image, they do not apply any rules. Each individual describes a picture as they are used to speak. However, inductive bias may occur even without applying any rules. This may happen for instance when all the annotators share the same style of expression. Thus, it is suitable to gather annotations from annotators from the widest possible range. In order for the machine to generate the most natural caption possible, multiple different datasets are often used for its training. This may help to reduce the effect of inductive bias.

Chapter 3

Caption Evaluation

Evaluation of machine-generated captions is a complex task. A single image can be described by numerous correct captions. But how can we determine, whether the machine-generated caption is correct or not? When automatic captions are evaluated, their naturalness is judged. If the caption is as if it was generated by a human, it can be considered correct. The best evaluation metric for this task is human judgement. However, because many times a huge amount of data needs to be evaluated, it is required to remove the human factor. In such cases, automatic evaluation metrics are used to simulate human judgement.

Some of the evaluation metrics used for image captioning were originally designed for the evaluation of different tasks such as machine translation or text summarization. These evaluation tasks have much in common with the image captioning task. Image captioning can be interpreted also as a translation from the content of the image to the natural language sentence or as summarizing the content of the image into a sentence.

For the purpose of the caption evaluation, the image is not taken into account and the caption evaluation is a purely linguistic task. Before the caption is evaluated, both the candidate and the reference captions are pre-processed by a tokenizer. The individual captions are represented as a set of n-grams. Some of the metrics are simple but focus on the specific aspects of the quality, such as n-gram overlaps. There are also advanced and complex metrics that also focus on the semantic structure of the captions. In this chapter, I will discuss some of the most used evaluation metrics for image captioning.

Following metrics evaluate for an image I_i the quality of a candidate caption $c_i \in C$ for a given set of reference captions $S_i = \{s_{i1}, ..., s_{im}\} \in S$. The captions are represented by sets of n-grams $\omega_k \in \Omega$. Each n-gram contains n ordered tokens. The number of n-gram ω_k appearing in the candidate caption c_i is denoted as $h_k(c_i)$ and in the reference caption s_{ij} as $h_k(s_{ij})$.

3.1 BLEU

BLEU (Bilingual Evaluation Understudy) [25] was originally designed for evaluating machine translations. Its approach works on counting n-grams in both tokenized candidate and reference sentences. Then the clipped n-gram precision is computed, defined as follows:

$$\mathrm{CP}_n(C,S) = \frac{\sum_i \sum_k min(h_k(c_i), \max_{j \in m}(h_k(s_{ij}))}{\sum_i \sum_k h_k(c_i)}. \tag{3.1}$$

The clipped precision favors a short sentences. In order to penalize the short candidate captions, brevity penalty is used. It is defined as:

$$b(C,S) = \begin{cases} 1 & \text{if } l_C > l_S \\ e^{1-l_S/l_C} & \text{if } l_C \leq l_S, \end{cases} \tag{3.2}$$

where l_C symbolizes the total length of candidate sentences c_i's and l_S is the total length of reference sentences s_i's. When there are multiple reference captions for one candidate caption, the closest reference length is used. The final BLEU score for the given N is computed as follows:

$$\mathrm{BLEU}_N(C,S) = b(C,S) exp\left(\sum_{n=1}^{N} w_n \; log\left(CP_n(C,S)\right)\right), \tag{3.3}$$

where N is the number of n used for n-grams and w_n is the weight constant for all n. Typically N = 1,2,3,4 and $w_n = \frac{1}{4}$ are used.

BLEU is a fast and simple metric that has been widely adopted. It is frequently used in research so it is often used as one of the metrics for comparing different methods. It has shown good performance on the corpus level evaluation, where a huge number of n-gram matches exist. However, on the sentence level, the higher number of n-gram matches for the higher n rarely occur. For the lower n, there may be a totally wrong sentence structure even with a high number of matches. Another disadvantage is, that BLEU assumes that both candidate and reference captions have been tokenized with the same tokenizer.

3.2 ROUGE

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) [26] is the set of metrics that have originally been designed for evaluating text summarization. The following ROUGE metrics are available:

- $\mathrm{ROUGE_N}$: Computes the n-gram recall of the candidate caption over all reference captions:

$$\mathrm{ROUGE_N}(c_i, S_i) = \frac{\sum_j \sum_k min(h_k(c_i), (h_k(s_{ij}))}{\sum_j \sum_k h_k(s_i j)}. \tag{3.4}$$

- ROUGE_L: Looks for the Longest Common Subsequence (LCS). LCS is a longest sequence shared by both captions that consists of words in the same order, but they do not have to be contiguous. For the LCS length $l(c_i, s_{ij})$ is ROUGE_L computed as F_β score:

$$P_L = \max_j \frac{l(c_i, s_{ij})}{|c_i|}, \tag{3.5}$$

$$R_L = \max_j \frac{l(c_i, s_{ij})}{|s_{ij}|}, \tag{3.6}$$

$$\text{ROUGE}_\text{L}(c_i, S_i) = \frac{(1+\beta^2)P_l R_l}{\beta^2 P_l + R_l}, \tag{3.7}$$

  where P_l is a precision and R_l is a recall of LCS. β is the constant, it is typically set to $\beta = 1.2$.

- ROUGE_W: It is the metric based on ROUGE_L. By setting weights, it favors consecutive LCS.

- ROUGE_S: Instead of LCS or n-grams, ROUGE_S uses skip bi-grams. Skip bi-grams are pairs of words that match in the candidate and reference sentences in order. But similarly to LCS, there may be any words in between them. ROUGE_S score is also computed as F_β score defined with precision and recall as follows:

$$P_S = \max_j \frac{\sum_k min(f_k(c_i), f_k(s_{ij}))}{\sum_k f_k(c_i)}, \tag{3.8}$$

$$R_S = \max_j \frac{\sum_k min(f_k(c_i), f_k(s_{ij}))}{\sum_k f_k(s_{ij})}, \tag{3.9}$$

$$\text{ROUGE}_\text{S}(c_i, S_i) = \frac{(1+\beta^2)P_S R_S}{\beta^2 P_S + R_S}, \tag{3.10}$$

  where f_k is the count of skip bi-grams in the caption.

The general difference between BLEU and ROUGE is that while the BLEU score focuses on the precision, ROUGE focuses on the recall. Thanks to properties of skip bi-gram or LCS, which do not depend on the consecutiveness, ROUGE tends to capture the sentence structure more accurately.

3.3 METEOR

METEOR (Metric for Evaluation of Translation with Explicit Ordering) [27] was designed for evaluating machine translation to improve existing metric BLEU. It generates the alignment of each word in the candidate and the reference caption. The WordNet synonyms and stemming can be used for aligning the words. The consecutive words in the alignment are called chunks. The alignment is generated so that the number of chunks ξ is

minimized. For the set of the alignments of length m, the METEOR score is computed as harmonic mean F_{mean} penalized by the chunk penalty θ as follows:

$$P_M = \frac{|m|}{\sum_k h_k(c_i)}, \tag{3.11}$$

$$R_M = \frac{|m|}{\sum_k h_k(s_{ij})}, \tag{3.12}$$

$$F_{\text{mean}} = \frac{P_M R_M}{\alpha P_M + (1-\alpha) R_M}, \tag{3.13}$$

$$\theta = \gamma \left(\frac{\xi}{m} \right)^{\theta}, \tag{3.14}$$

$$\text{METEOR} = (1-\theta) F_{\text{mean}}. \tag{3.15}$$

where α, γ, θ are the parameters.

METEOR is the popular alternative to the BLEU score. The advantage is considering both precision and recall, and favouring the big chunks of the candidate caption aligned with the reference caption. It can also use semantic similarity when aligning the captions.

3.4 CIDEr

CIDEr (Consensus-based Image Description Evaluation) [28] is the metric used for image captioning evaluation. For the stemmed candidate and reference sentences, it computes the Term Frequency Inverse Document Frequency (TF-IDF) weighting for each n-gram. TF gives more weight to such n-grams, which occurs in the reference captions of the image more frequently. IDF reduces the weight of the n-grams which commonly occur in captions across all of the images. For the given set of images I is TF-IDF wight $g_k(s_{ij})$ computed as follows:

$$g_k(s_{ij}) = \frac{h_k(s_{ij})}{\sum_{\omega_l \in \Omega} h_l(s_{ij})} log \left(\frac{|I|}{\sum_{I_p \in I} min(1, \sum_q h_k(s_{ij}))} \right). \tag{3.16}$$

CIDEr_n score for n-grams of length n is then computed using the average cosine similarity between the candidate and the reference sentences as:

$$\text{CIDEr}_n(c_i, S_i) = \frac{1}{m} \sum_j \frac{g^n(c_i) g^n(s_{ij})}{\|g^n(c_i)\| \|g^n(s_{ij})\|}. \tag{3.17}$$

The final CIDEr score is computed by combining the CIDEr_n scores as follows:

$$\text{CIDEr}(c_i, S_i) = \sum_{n=1}^{N} w_n \text{CIDEr}_n(c_i, S_i), \tag{3.18}$$

where $w_n = \frac{1}{N}$ and $N = 4$ are usually used.

CIDEr-D is a modification of the CIDEr metric, to make it more robust against gaming. Gaming refers to the phenomenon, where the caption is judged with a low score by human

and with a high score by the automatic metric. To suppress the gaming effect, CIDEr-D uses clipping and a length-based Gaussian penalty to the CIDEr metric. It is computed by the following equations:

$$\text{CIDEr-D}_n(c_i, S_i) = \frac{10}{m} \sum_j e^{\frac{-(l(c_i)-l(s_{ij}))^2}{2\sigma^2}} \cdot \frac{min(g^n(c_i), g^n(s_{ij})) \cdot g^n(s_{ij})}{\|g^n(c_i)\|\|g^n(s_{ij})\|}, \tag{3.19}$$

$$\text{CIDEr-D}(c_i, S_i) = \sum_{n=1}^{N} w_n \text{CIDEr-D}_n(c_i, S_i), \tag{3.20}$$

where $l(c_i)$ is length of the candidate caption and $l(s_{ij})$ is the length of the reference caption. The constant $\sigma = 6$ is usually used. A factor of 10 is used in the numerator to make the CIDEr-D scores numerically similar to the other metrics.

According to [28], CIDEr has shown the best results when it was compared with the metrics described above. The disadvantage is that it requires the whole dataset to compute the word frequencies. Also, the range is not between 0 and 1, unlike other metrics.

3.5 SPICE

SPICE (Semantic Propositional Image Caption Evaluation) [29] is the metric designed for image captioning evaluation. It exploits the semantic structure of the captions by parsing both the candidate and the reference captions into scene graphs. The scene graph of the caption c is denoted as $G(c)$:

$$G(c) = \langle O(c), E(c), K(c) \rangle, \tag{3.21}$$

where $O(c)$ is the set of objects mentioned, $E(c)$ is the set of relations between objects, and $K(c)$ is the set of the attributes associated with objects. The caption parsing corresponds to the given rules such as labeling each noun as an object, even if it does not have any relations, or dropping plural nouns and the numeric modifiers being encoded as object attributes instead. Next, the function T is defined as:

$$T(G(c)) \triangleq O(c) \cup E(c) \cup K(c). \tag{3.22}$$

It returns logical tuples from a scene graph. Each tuple can contain one, two, or three elements, representing the objects, their attributes, or relations. The final SPICE score is then defined as the F_1 score computed from precision and recall:

$$P(c, S) = \frac{|T(G(c)) \otimes T(G(S))|}{|T(G(c))|}, \tag{3.23}$$

$$R(c, S) = \frac{|T(G(c)) \otimes T(G(S))|}{|T(G(S))|}, \tag{3.24}$$

$$\text{SPICE(c,S)} = F_1 = \frac{2 \cdot P(c, S) \cdot R(c, S)}{P(c, S) + R(c, S)}, \tag{3.25}$$

where $\otimes$ is the binary matching operator as the function that returns matching tuples in two scene graphs. For matching the tuples, the WordNet synonyms and stemming are used.

Unlike CIDEr, SPICE does not require knowledge of the whole dataset for computing word frequencies. Thanks to that, it can be used for both large and small datasets. Overall SPICE metric is a very good metric that correlates well with human judgement when compared to the metrics listed above. The disadvantage of SPICE is its computational complexity.

Chapter 4

Related Work

Image captioning is a very complex task. An image captioning system must first understand the content of the image and then generate a semantically and syntactically correct caption that describes the image. The interest in the task of image captioning has gradually increased in the last decade. The evolution of methods used in image labeling over time corresponds to the evolution of general methods used in machine learning. There are numerous works that thoroughly describe the existing image captioning techniques [30, 31, 32, 33, 34]. In this chapter, I briefly summarize the methods used in the image captioning task and discuss the methods that I find interesting for my work.

4.1 Types of methods

The image captioning methods can generally be divided into two categories: traditional computer vision and deep learning. Traditional computer vision methods include using hand-crafted features such as Histogram of Oriented Gradients (HOG), Local Binary Patterns (LPB), Scale Invariant Feature Transform (SIFT) or Speeded Up Robust Features (SURF). Features extracted from the input image are then used for training a classifier such as Support Vector Machine (SVM). The content of the real-world image is very complex and the traditional methods can not cover the whole semantics of the image. Therefore, traditional computer vision methods are used mainly for specific tasks.

In deep-learning-based methods, image features are automatically learned from the data. Convolutional Neural Networks (CNN) are widely used for this purpose. After extracting the features, CNN is generally followed by a Recurrent Neural Network (RNN) or a Multi-layer Transformer.

Each of the different techniques is suitable for a different type of image captioning. Image captioning can be generally divided into three categories: retrieval-based image captioning, template-based image captioning, and deep-learning-based caption generation. In the retrieval-based approach, the caption for the given image can be retrieved either from

the set of existing captions or by their combination. The resulting captions are always syntactically correct. On the other hand, they are very limited to describing specific elements of the images and thus they achieve poor semantic quality.

In the template-based image captioning approach, a fixed template is used. Slots of the template are then filled with corresponding words detected from the image. These include objects, attributes, and actions. For example, [35] uses a triplet of scene elements to fill in the slots of templates for generating image captions. Template-based methods can generate syntactically correct captions. The disadvantage of this approach is the fixed length of the caption.

In deep-learning-based image captioning, the methods first analyze the image and then generate the caption using a language model. These methods generate captions with better semantic accuracy and higher variability. These methods are usually based on deep learning. Deep-learning-based image captioning achieves better both semantic and syntactic accuracy and much higher variability than the two previously described methods. Therefore, research in recent years has mainly focused on deep-learning-based captioning.

4.2 Evolution of image captioning

One of the first methods that resembles today's image captioning, **Every Picture Tells a Story: Generating Sentences from Images** [35], has been published in 2010. This method retrieves the triplet of the object, action, and scene from the image. To solve this task, the method uses Markov Random Field, where each node represents the object, action, or scene. The potentials of edges are computed by the combination of scores from several detectors and classifiers and the potentials of edges are computed by the frequencies. The object can take one of 23 different values, action on of 16 values, and the scene one of 29 different values.

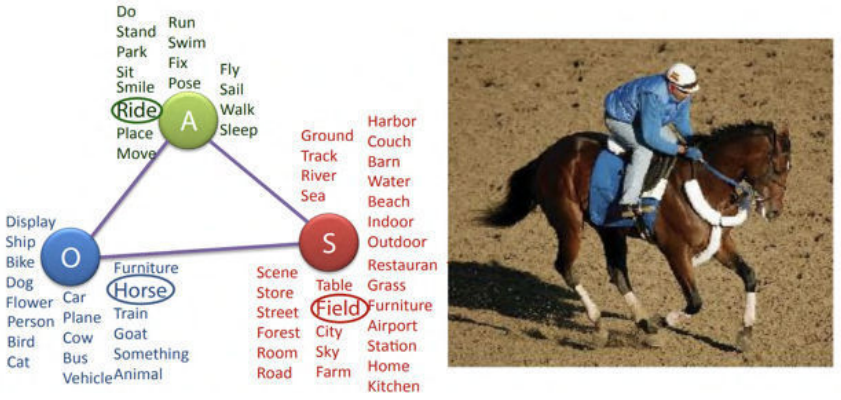

Figure 4.1: Markov Random Field used by [35].

As the performance of computers has increased, deep learning has come into use. It is a powerful tool for complex tasks, which image captioning certainly is. In 2015, **From Captions to Visual Concepts and Back** [36] was introduced. This method uses a CNN model to detect the probability of words from a given set in the caption of the image. Then, the most likely sentence is generated. A maximum entropy language model in a generative framework is used for this purpose.

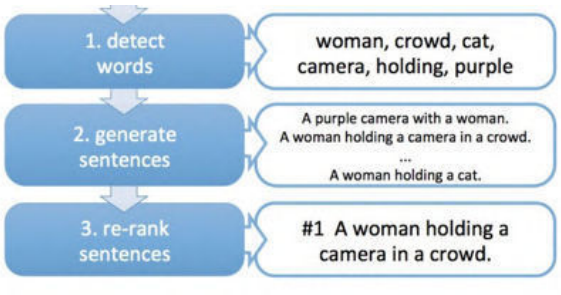

Figure 4.2: An illustrative example of a pipeline used in [36].

Also in 2015, O. Vinyals et al. introduced a path-breaking approach to image captioning, **Show and Tell: A Neural Image Caption Generator** [37]. Inspired by success in machine translation, the authors used an encoder-decoder framework to create a generative learning scenario. The CNN is used as an encoder to extract image features. Then, the RNN architecture, LSTM, is used as a decoder to generate the sequence of words. In each time step, LSTM decodes one word which is then used to update the state of the network and to generate the next word. This technique is used also in other text-generation tasks, such as machine translation. This method provided the state-of-the-art results of its time. However, this method also has its disadvantages. One of them is its huge demand on computing memory and time required for training. This is not such a problem for machine translation methods where the input is the text. However, with image captioning methods, there is dense information of the entire image on the input, which causes significant limitations.

Later in 2015, **Show, Attend and Tell: Neural Image Caption Generation with Visual Attention** [38] came up with an attention-based encoder-decoder. It is another important method in image captioning. In the decoder, the attention is computed over the image. The attention gives importance to relevant portions of the input image in the encoder for generating each word in the decoder. Unlike in the previous method, the image features are extracted from the lower layers of the CNN in the encoder.

In 2018, [39] was published. It introduces new method **UpDown**. UpDown joins a visual bottom-up mechanism and a task-specific context top-down one. The bottom-up mechanism gives proposals on image regions that it considers important. The top-down mechanism uses context to compute an attention distribution over them. This allows the attention to be directed to the important objects in the input image.

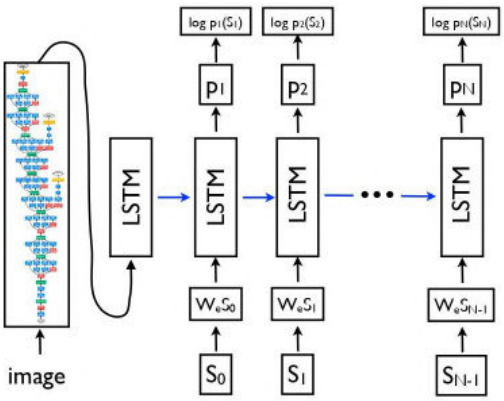

Figure 4.3: LSTM model combined with a CNN image feature extractor and word embeddings used in [37].

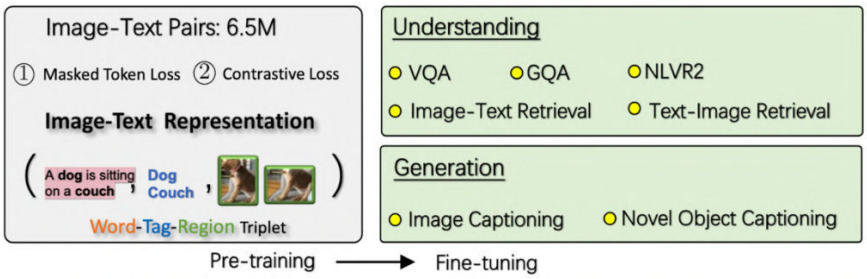

Figure 4.4: Pipeline used in [13].

Methods of this time have weaknesses such as the lack of explicit alignment information between the image regions and the captions. In addition, the image regions may contain two different objects overlapping and thus making the information ambiguous and noisy. In 2020, method **Oscar (Object-Semantics Aligned Pre-training)** [13] was introduced. To suppress these problems, Oscar uses object tags detected from images as "anchor points". It uses the triplets of image features, text, and newly the object tags for its training and the image features and tags for the inference. This helps when one of the channels is noisy and the other can carry the information. Instead of RNN, Oscar uses a multi-layer transformer for the generation of the caption. Transformer [40] is a deep learning architecture, that was introduced in 2017. In 2020, it was adopted for tasks of image processing, such as object detection [41] and image classification [42]. In 2020, Oscar achieved the best results in most of the metrics on the COCO Captions dataset and held his position until mid-2021. At the start of 2022, when this work is being written, the best BLEU-4 results provide the method **OFA (One For All)**, presented in [12].

Image captioning has many more different methods such as using reinforcement learning [43] or using generative adversarial networks [44]. It is a still an actual task and I believe, the methods will be evolving also in the future.

Chapter 5

Implementing the Captioning Pipeline

In this paper, I reproduce the results of some state-of-the-art image labeling methods. For this purpose, I decided to use the method Oscar [13], which I also mentioned in Section 4.2. The method was chosen based on its performance on the COCO Caption [11] dataset. At the time I was deciding which method to work with, Oscar achieved the best score on most of the comparing metrics. A table with the complete results is available at Papers With Code[1]. Oscar uses a multi-layer transformer architecture [40]. It is a relatively new technique introduced in 2017. I believe that by using and understanding this method, I will gain experience in this field and an awareness of the state-of-the-art in deep learning.

The image captioning task has two main parts: feature extraction from the input image and caption generation. Oscar provides only the second part. That means that image features appear on the input of Oscar and need to be generated by an external detector. The caption of a given image is the output of Oscar. For the demo to work, Oscar provides features for all images in the COCO Caption data set. However, the authors do not specify which detector was used to generate them. This means that whenever we want to generate a caption for an image that is not part of this dataset, it is not clear how to extract features from it. In this section, I implement a pipeline that is capable of generating a caption for a custom image. I use the Detectron2 [45] detection framework to generate features from the entire COCO dataset. I then train Oscar on the newly created dataset. In this section, I describe these steps in detail.

5.1 Dataset

Oscar [13] provides the image features extracted from COCO Caption dataset [11]. Based on this knowledge, I decided to use the COCO dataset, 2017 version, in my work. Thanks to

[1] `https://paperswithcode.com/sota/image-captioning-on-coco-captions`

that I am able to compare the results afterward. COCO Caption is the dataset containing over 164,000 images of everyday scenes with multiple captions for each image. More information about the structure of this dataset has been discussed in Section 2.2. It is widely used and is well approachable. COCO also provides the evaluation server[2], where methods can be evaluated by some of the metrics discussed in Chapter 3.

The images from the COCO Caption dataset are processed by the detector. It returns the set of detected objects, including their classes and feature vectors. Then, the data needs to be restructured in a specific way, as Oscar requires. The resulting structure consists of the following files:

- *data.label.tsv*
  Contains two columns of image id and the list of objects detected by encoder. Each of the object is represented by dictionary that consists of a class, position and its confidence. This list is in json format.

```
184613  [{"class": "animals", "rect": [140.4112,81.9309,
   499.4400, 303.80444], "conf": 0.2314}, {"class": "hay",
     "rect": [0.0, 169.6649, 394.9242, 335.4400], "conf":
   0.3473}, ...]
403013  [{"class": "ceiling", "rect": [0.0, 0.0, 300.4983,
   153.1022], "conf": 0.9131}, {"class": "kitchen", "rect":
   [31.7213, 5.3686, 300.4983, 266.1175], "conf": 0.3067},
   ...]
...
```

- *data.label.lineidx*
  Contains an index number on each line, representing the cumulative number of characters *data.label.tsv* for a current line.

```
0
6221
...
```

- *data.feature.tsv*
  Contains two columns of the image id and the dictionary. This dictionary contains a number of objects detected by encoder and the features of these objects encoded into base64 string. The features are $n \times 2056$ long vectors, where n = number of detected objects in the image.

```
184613      {"num_boxes": 47, "features": "AAAAALcFDUAFN18/
   QA UcQN0VH0DX/4U/iv8KQOANoT9w2wU/3beUPioqvD1ohH48gOXbPL
   ..."}
```

[2] https://competitions.codalab.org/competitions/3221

```
403013      {"num_boxes": 24, "features": "9YyqPgAAAADzhoA+
   AA AAABXurjwAAAAAAAAAAAAAADfO4tACi4CO3MZkj8AAAAALxxqPg
   ..."}
...
```

- *data.feature.lineidx*
  Contains an index number on each line, representing the cumulative number of characters *data.feature.tsv* for a current line.

```
0
514913
...
```

- *data_caption.json*
  Contains caption annotations by image id in the json format.

```
[{"image_id": "106140", "id": 98, "caption": "A large
   passenger airplane flying through the air."}, {"image_id
   ": "106140", "id": 101, "caption": "There is a GOL plane
   taking off in a partly cloudy sky."}, ...]
```

- *data.yaml*
  Contains settings - directories of all the files described above.

```
img: val.img.tsv
hw: val.hw.tsv
label: val.label.tsv
feature: val.feature.tsv
caption: val_caption.json
```

5.2 Detector

The authors of the method, which I am working with, Oscar [13], do not provide a solution on how to extract features from the source image to be used as Oscar's input. It is known that it requires the objects detected from the given image, including their tag, position, confidence, and the feature vector for each of them. The feature vector of one object is 2054-dimensional, where the first 2048 elements are the image features and the last 6 are the coordinates of the bounding box of the detected object in the input image $x_{min}, x_{max}, y_{min}, y_{max}$, its width $w = x_{max} - x_{min}$ and height $h = y_{max} - y_{min}$. Therefore, I was looking for an object detector which I can extract a 2048-dimensional vector from.

For this purpose, I decided to use the architecture Faster-R-CNN implemented in Detectron2 [45] framework. Detectron2 is a popular library that provides state-of-the-art

detection and segmentation algorithms. It provides several different backbones. My goal was to find a backbone that provides a 2048-dimensional vector in the last layer before the softmax layer. I decided to use an R50-C4 backbone that meets this requirement. This backbone uses a ResNet conv4 backbone with a conv5 head. The original baseline in the Faster R-CNN paper [46].

Since the authors do not provide information on how to extract features from the images, I had to conduct some research on provided data to be able to set up the detector correctly. After analyzing the provided data, I think I came up with an algorithm that the authors used to filter the detections:

- Every image has at least 10 detections. Maximum number of detections in one image is 99. In average, there are 31-32 detections per image.
- The detections have confidence over 0.2.
- In the case, when the image does not have at least 10 detections with a confidence over 0.2, detections with lower confidence are used to make a total of 10 detections.
- Multiple detections occur for a single object.

Detectron2 provides the detector weights pre-trained on the COCO dataset [3]. This dataset contains labels of 80 different classes. Therefore, the detector I am using is capable of detecting these 80 classes. This set of classes is designed to contain as wide a range of common objects as possible. For my work, I have therefore chosen to use the detector with the weights provided by the authors and will not further fine-tune the model to be able to detect more classes. It is expected that the subsequent results will not perform as well as the state-of-the-art methods perform. However, for my thesis, I have concluded that the results obtained using these 80 classes are sufficient.

In order to extract as much semantic information as possible from an image, many objects need to be detected. The data limited by Oscar does not seem to be limited by the number of objects detected. Therefore, my goal is to extract the detection of as many objects as possible from the image. One way to affect the number of detections is by changing the confidence threshold. However, if this threshold is set too low, false detections will start to occur, which is not a desirable effect. After trying several different settings, I have concluded that the optimal setting for my detector is a reliability threshold = 0.2. At this value, a sufficient number of objects are detected and at the same time, there are not many false detections. Moreover, this value corresponds to the value used by Oscar. However, my approach differs from Oscar's in that I do not use detections with a confidence threshold lower than 0.2 when there are not enough of them. The reason for this decision is purely practical. The detector framework does not support setting a minimum number of detected objects and its implementation would require deep intervention in the source code. Since it is desirable to have as many detected objects as possible, and it is known that Oscar uses

Detected objects by the detector
elephant, person, elephant, fire hydrant, bottle, elephant, baseball glove, elephant

Objects provided by Oscar
dirt, road, elephant, ground, tree, sky, trees, shirt, ear, ground, fence, elephant, fire hydrant, head, sign, sign, pole, grass, tusk, hat, eye, elephants, man, trunk, trunk, pole, field, grass, mouth, elephant, tree, fire extinguisher, foot

Figure 5.1: Object tags detected in the image by the detector, compared with object tags provided by Oscar for the same image. All the detected objects with confidence > 0.2 are taken into account. Although it may seem that the objects detected by the detector represent the image well, there are only 8 of them, which is below the threshold value of 10. Thus, I do not use this image to train Oscar. Courtesy of [3].

multiple detections for a single object, I also decided to tune the value of the intersection over union (IoU) threshold. After trying several different values, I decided to use the value that generated the closest number of detections to Oscar: $\text{thr}_{\text{IoU}}=0.8$.

Even if the detector settings are optimized, in some images the detector is not able to detect the required number of objects, i.e. 10. The volume of images with less than 10 detections with confidence above 0.2 is approximately 19 % of the whole dataset. Since I have enough data to train Oscar for my needs, I decided to preserve the data structure of Oscar and not use this 19 % of the data to train it. In the image in Figure 5.2, the examples of images with extreme numbers of detected objects can be seen.

5.3 Fine-tuning Oscar

Oscar [13] is a multi-layer transformer based method. It is implemented in Python 3.7, using Pytorch 1.2, torchvision 0.4.0, and cuda 10.0. Oscar is pre-trained on a large-scale visual-language dataset composed of Flickr30k, COCO Captions, Conceptual captions, and more datasets to achieve a total size of 6.5 million image-caption pairs. It allows fine-tuning and evaluation on seven visual-language understanding and generation tasks, such as image captioning, visual question answering, or image-text retrieval.

Figure 5.2: Images with the extreme number of detected objects. On the left: an image with 90+ detected objects. On the right: an image in which the detector did not detect any objects. After inspecting such images, I concluded that those are quite often images of traffic signs. Courtesy of [3].

In Section 5.2, I extract the features from the source images from the COCO dataset. Then, the detector output is re-structured into Oscar input structure, which is discussed in detail in Section 5.1. In this section, I use the newly acquired data to fine-tune Oscar on the image captioning task. Oscar provides different pre-trained checkpoints depending on the language model variant used. Oscar can be trained with different settings. I decided to train Oscar with the inspiration of the model zoo on its GitHub [3] and train it first using cross-entropy loss. The settings do not exactly match the zoo model: the batch size is set according to the physical capabilities of the hardware I used. To train Oscar I used the following settings:

- Pre-trained checkpoint: bert_base_uncased
- Initial learning rate: $3 \cdot 10^{-5}$
- Batch size: 32
- Number of training epochs: 30.

It takes over 100 hours to train 30 epochs on the hardware I used. However, I was limited to training for a maximum of 24 hours at a time. However, the code provided by Oscar does not support resuming training. Therefore, during training, I saved checkpoints after each epoch from which I could then resume training. These checkpoints contain the current model weights, but do not contain the optimizer state. Thus, during each of the training interruptions, the optimizer momentum was reset. The learning rate was set at each resuming to the value the model had when the checkpoint was saved.

[3] `https://github.com/microsoft/Oscar`

1. boat, boat, sports ball, boat, boat, boat, person, boat, person, boat, bird, person, boat, boat, boat, sports ball, person, person, person, boat, person, boat, boat, person, person, boat, boat
2. A small boat in a large body of water.

Figure 5.3: Example of an image that is not part of the COCO dataset. 1. - Detected object tags by the detector, 2. - Prediction of my custom pipeline.

After training with cross-entropy loss, model zoo continues with additional 5 epochs of fine-tuning with CIDEr optimization using Self-critical Sequence Training. However, the authors do not provide the files needed for Self-critical Sequence Training. Therefore, I decided to fine-tune Oscar without using Self-critical Sequence Training. I used the following settings:

- Initial learning rate: $5 \cdot 10^{-6}$
- Batch size: 16
- Number of training epochs: 5.

5.4 Conclusion

In my work, I decided to use Oscar [13] as the core of my captioning pipeline. It is a state-of-the-art image captioning method based on a multi-layer transformer. Oscar is used for several vision-language tasks, such as image captioning, vision question answering, or image-text retrieval. The authors provide image features extracted from the COCO dataset but do not specify which detector has been used. If we want to generate a caption for an image that is not part of the provided dataset, we need to extract features from the image first. However, since the authors do not provide information about which detector they used to extract the features, the one that satisfies the dimension requirements of its output must be chosen. From the wide range of various detectors, I decided to use the Faster-R-CNN architecture implemented within the Detectron2 [45] framework to create my own dataset of features. This dataset was then used to train Oscar. By having Oscar trained on the

features that come from a known detector I am thus able to generate a caption for any image. I do this by first extracting features from the image using Faster-R-CNN and then using these features as input to the Oscar program.

The results achieved by my custom pipeline cannot be sufficiently compared with the results achieved by Oscar. Its authors do not sufficiently describe how and on which dataset they perform the evaluation. It is known that Oscar was evaluated on the COCO dataset. However, the authors provide the results of metrics that the official COCO evaluation server[4] does not support. Thus, I decided to evaluate my model on validation data for which COCO provides ground truth data. In Table 5.1, the results of the metrics that the Oscar article publishes are compared with my results for the same metrics.

Metric	BLEU-4	METEOR	CIDEr	SPICE
Original Oscar [13]	**0.417**	**0.306**	**1.40**	**0.245**
My pipeline	0.312	0.272	1.02	0.201

Table 5.1: Performance of Oscar on COCO presented by [13], compared with performance of Oscar trained on my dataset on validation data.

The table shows that Oscar outperforms my pipeline in every metric. This is an expected result caused by several reasons. The first major difference between the approach presented in [13] and mine is the dataset. I only use the COCO 2017 dataset, while [13] uses a huge corpus consisting of seven different datasets. In total, I use 590,000 image-caption pairs to train Oscar, while [13] uses 6,500,000 image-caption pairs. The detector I used is able to detect objects of 80 different classes. [13] does not specify which detector was used to extract the features, but it can be assumed that the number of possible classes was much larger. A comparison of the object classes in the image can be seen in the example in Figure 5.1. Another difference is the pre-trained checkpoint used. Oscar provides two variants: basic and large. According to [13], large provides better results, but at the cost of higher computational requirements. To achieve the result in Table 5.1 I used basic and the original Oscar uses the large one. More details on the checkpoints can be seen in [13]. The last known difference between my pipeline and the original Oscar that may affect the metric score is the approach to filtering detected objects. All the objects that were detected with confidence > 0.2 are used. If there are less than 10 such objects, I do not use that image to train Oscar. However, the original Oscar uses objects with lower confidence in this case, so that there are a total of 10 objects.

The performance of my pipeline could be improved in the future. In my opinion, the most significant improvement would be to use a larger dataset with more possible classes on which both detector and Oscar could be tuned. Another improvement might be to use a different approach to filtering detections from the detector. Original Oscar uses at least

[4] `https://competitions.codalab.org/competitions/3221`

10 detections per frame. During my work, I have concluded that Oscar performs reliably even with fewer detections. It might be interesting to conduct an experiment and try a different minimum number of detections per image. If the minimum number of 10 detections had to be preserved, another experiment could be done by trying different approaches to complementing the detections, so that less than 10 detections would make just 10. One technique is to do it like the original Oscar, i.e., use even those with less confidence so that there are 10 in total. Since multiple detections of a single object are used, another technique for completing the detections to a total of 10 may be to duplicate those with the highest confidence.

Chapter 6

Ablation Study

The previous section describes the implementation of an image captioning pipeline based on the method Oscar [13]. Since Oscar is a multi-modal system, it may be interesting to test its sensitivity to individual modalities. To recapitulate, Oscar uses a visual modality, represented by the image features of the objects detected in the image. In addition, Oscar introduces object tags, represented by the class of the detected object, as a textual modality. In this chapter, I conduct an ablation study to test the robustness and sensitivity of Oscar to each modality. In the experiment, I remove one of the modalities while keeping the other, and monitor the output of the system. The sensitivity to each modality is measured by comparing the metric score obtained by predicting the captions using a dataset with complete information and another dataset with one of the modalities removed.

6.1 Experiment

In this experiment, I create new datasets with removed modalities based on a dataset with complete modality information. To evaluate the sensitivity, I compare the predictions of the dataset with full and restricted information generated by my pipeline, which I created in Chapter 5. To compare the datasets, I decided to use the official COCO evaluation server[1]. This evaluation server computes on the COCO 2014 validation and test dataset some of the metrics described in Chapter 3. Because of this, I conduct the ablation study specifically on these two datasets. The structure of these datasets is described in detail in Section 5.1.

For both, the COCO 2014 validation and test datasets, I create two new datasets. One for each of the modalities, visual and textual. It is done by modifying the original datasets so that in each new dataset, one of the modalities is removed while the other modality is retained. There are two main files in the data structure that carry information about the image. The first is *data.feature.tsv*. It carries information about the visual modality, which is represented by a 2056-dimensional vector encoded in a base64 string. The second file is

[1] `https://competitions.codalab.org/competitions/3221`

data.label.tsv, which carries the textual modality. It is represented as a set of detected class names expressed in text form.

First, I create a dataset with the visual modality removed while keeping the textual modality. The visual modality is removed by editing the *data.feature.tsv* file. I decode the string into an array of floats. I replace these numbers with zeros and encode the array back into a byte64 string in the correct form. I then create another dataset with the textual modality removed. It is stored in the *data.label.tsv* file as the name of the detected classes in text form. I replace these class names with an empty string expression. The previews of the modified files then look as follows:

- *data.features.tsv*

```
184613      {"num_boxes": 47, "features": "AAAAAAAAAAAAAAAA
    AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA..."}
403013      {"num_boxes": 24, "features": "AAAAAAAAAAAAAAAA
    AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA..."}
...
```

- *data.labels.tsv*

```
184613  [{"class": "", "rect": [140.4112,81.9309, 499.4400,
    303.80444], "conf": 0.2314}, {"class": "",   "rect":
    [0.0, 169.6649, 394.9242, 335.4400], "conf": 0.3473},
    ...]
403013  [{"class": "", "rect": [0.0, 0.0, 300.4983,
    153.1022], "conf": 0.9131}, {"class": "", "rect":
    [31.7213, 5.3686, 300.4983, 266.1175], "conf": 0.3067},
    ...]
...
```

For the purposes of the ablation study, I created a total of 6 datasets. The first two were created from the COCO 2014 validation and test dataset using the detector. The evaluation server requires the prediction of all the images from the datasets. In Chapter 5, I did not use the images with less than 10 detected objects. To meet the requirements of the evaluation server, I predict captions for every image that has any detections in this experiment. In case the image has no detection, an empty string is used as a caption for evaluation purposes.

The next two datasets were created by modifying the first two datasets by removing the visual modality. Another two datasets were also created by modifying the first two datasets, but this time by removing the textual modality. I used the official COCO evaluation server, which requires both validation and test datasets, to calculate the metrics that can be seen in Table 6.1.

Metric	B-1	B-2	B-3	B-4	M	R-L	C-D
Full information	0.697	0.526	0.393	0.296	0.265	0.531	0.977
Features removed	0.549	0.356	0.228	0.152	0.183	0.409	0.504
Tags removed	**0.672**	**0.498**	**0.366**	**0.272**	**0.250**	**0.511**	**0.875**

Table 6.1: B-1-4: BLEU-1-4, M: METEOR, R-L: ROUGE_L, C-D: CIDEr-D. Results of metrics computed for the captions generated by the pipeline I made in Chapter 5. The comparison is between the different approaches when the system had access to all the information, when the visual modality was removed, and when the textual modality was removed.

6.2 Conclusion

In this chapter, I conducted an ablation study on the multi-modal image captioning method Oscar [13]. This method uses a visual modality represented as image features and a textual modality represented as the name of the detected class. Using my custom pipeline described in Chapter 5, I created two feature datasets from the COCO 2014 validation and test datasets. In order to satisfy the requirements of the COCO evaluation server[2] I use for evaluation, the images with less than 10 detections were kept instead of discarded. I then created two more variants of the datasets: one variant with the visual modality removed while retaining the textual modality, and one with the textual modality removed. Then, I evaluated them on the COCO evaluation server. The results can be seen in Table 6.1. Both datasets with the modality removed give worse scores than the original. This is the expected result. It means that Oscar is sensitive to both modalities.

The dataset with the textual modality removed achieves better results than the one which has the visual modality removed. Although this may suggest that Oscar is more sensitive to the visual modality, I do not explicitly conclude this. An object tag is represented by a name of the class, which comes from the set of the 80 possible ones. This means that when Oscar is generating the caption of data with image features having removed, it has very limited options to generate the caption from. The idea is illustrated in the Figure 5.1. The detector I used, successfully detects the elephant but does not detect any object that may give out the context of the image, such as where the elephant is. It has no information whether the elephant is in the middle of city streets or in the countryside. However, the tags provided by the original Oscar signalize that the content of the image is probably happening somewhere in nature. On the other hand, we can not disregard the importance of the visual modality. It contains information that tags can never carry. From tags, it probably never be possible to know, for instance, color, size, position of objects, and so on. I think it may be interesting in the future to repeat the experiment on the data with a larger variety of

[2] `https://competitions.codalab.org/competitions/3221`

possible class names in order to determine whether the difference of the scores I provided changes or not.

The example of predicted captions, from all three versions of the datasets this chapter talks about, can be seen in Figure 6.1. It is interesting to observe what context Oscar assign to objects when features were removed. For instance, Oscar generated the caption mentioning a street and a tall building. It may be assumed that the truck close to the bench appears mostly on the street next to tall buildings in the training data. A similar interpretation can be done for the second image, where Oscar correctly guessed the tree in the background, having only the information of *train* and *person* being in the image. From this, I conclude that a train near a tree often appears in the training data.

1. truck, truck, truck, truck, bench
2. A rusted out pickup truck sitting in a field of tall grass.
3. A truck driving down a street next to a tall building.
4. A rusted out truck sitting on top of a lush green field.

1. train, person
2. A yellow and blue train traveling down train tracks.
3. A train on a train track with trees in the background.
4. A yellow and blue train traveling down train tracks.

1. clock, clock
2. A black and white clock on a brick wall.
3. A tall building with a large clock on the side of it.
4. A clock that is on the side of a building.

1. laptop, cat, book, bottle, laptop, laptop, cat, laptop, laptop, bottle, laptop, cup, laptop, laptop, book, book, laptop, laptop, microwave, book, cat, laptop, laptop, bowl, book, book, book, tv, laptop, book
2. A cat that is sitting on top of a laptop.
3. An open laptop computer sitting on top of a wooden desk.
4. A cat laying on top of a laptop computer.

Figure 6.1: Predictions from the dataset with full information compared to predictions from the dataset with one modality removed. 1 - Detected objects by the detector, 2 - Prediction from full information, 3 - Prediction with features removed, 4 - Prediction with tags removed. Courtesy of [3].

Chapter 7

Final Discussion

In chapter 5, I created an image captioning pipeline, based on the method Oscar [13]. I concluded that the quality of the generated captions depends on the quality and volume of the data that was used to train the models. It is known that the detector I use in my pipeline can detect objects from the known set of the 80 classes. I found it interesting to use the pipeline to generate labels for images that I believe contain features that were not included in the training data of the models used in the pipeline. For this purpose, I have created a mini-set of images that, in my opinion, meet this requirement. These images are not part of the COCO dataset [3] that I use in my experiments. They were taken by me or were manually retrieved from the web, based on my personal ideas. In this chapter, I discuss and give my personal opinion on the images and their captions from this mini-set that I find most interesting.

Figure 7.1 - **Swans**: I find this picture an interesting illustration of the similarity of human thinking to a machine. *Swan* is not part of the set of COCO class names. The detector detects objects that it assumes belong to the *bird* class. After 30 epochs of training, Oscar predicts a caption that refers to *a group of birds*. However, after another 5 epochs of fine-tuning, Oscar changes its prediction to *a flock of ducks*. Even though *duck* is also not part of the COCO class names, It is able to recognize from the semantic information. I believe it is able to do this thanks to a sufficient amount of ducks in the training data. In my eyes, the intermediate step between bird and swan is the duck. I believe that if a human did not know a swan, they would refer to it as a duck or a bird, like Oscar. On the other hand, I think that if the swan was in the training data enough times, Oscar would be able to improve his prediction to a swan at some point after further fine-tuning. However, since swan and duck are not part of the COCO class names, it is not possible to determine with certainty how many of them the training dataset contains.

1. person, bird, bird, bird, sheep, cow, cow, sheep, bird, sheep, person, bird, sheep, sheep, bird, fire hydrant, sheep, person, bird, cow, cow, sheep, person, bird, bird, elephant, bird, bird, cow, bird, sheep, sheep, bird
2. A group of birds that are sitting in the grass.
3. A flock of ducks sitting on top of a lush green field.

Figure 7.1: Example of how Oscar deals with the image of swans that are not part of the set of COCO class names. 1. - Detected object tags by the detector, 2. - Prediction of Oscar after 30 epochs of training, 3. - Prediction of Oscar after another 5 epochs of fine-tuning.

Figure 7.2 - **Tigers**: Since the detector is restricted to 80 possible classes, I found it interesting to let Oscar generate a caption for an image of animals, that are not part of these classes, but have similar characteristics to animals that are part of the 80 class set. The detector detected *zebra*, *dog*, *cat* and *bear*. From my personal gaze, I see that the tiger has some elements of each of those animals. In my opinion, the most interesting part is the caption that Oscar generated based on these detections. It called two tigers *a couple of animals*. Since the tiger is not part of the set, Oscar dealt with this problem very successfully. I assume it mimicked human behavior when asked to describe the image of an animal they had never seen in their lives.

1. zebra, dog, zebra, zebra, zebra, dog, dog, zebra, cat, cat, zebra, person, person, person, bear
2. A couple of animals that are in the dirt.

Figure 7.2: Example of how Oscar deals with the image of tigers that are not part of the set of COCO class names. 1. - Detected object tags by the detector, 2. - Prediction my custom pipeline[1].

[1] `https://unsplash.com/photos/dke0cAkors4`

Figure 7.3 - **Purple cow**: Within this Chapter, I wanted to test the importance of the coloring of the object to Oscar. I decided to generate a caption for the image of a purple cow. As can be seen in the figure, the detector correctly detected the cow without any difficulties. From the detections, Oscar generated a caption about a black and white cow. I assume there was no purple and white cow in the training data, but there were a lot of black and white ones. The fact is, lightning may affect the colour. Therefore I believe, that if there were black and white cows under bluish light in the training data, Oscar probably had no problems interpreting the purple cow.

1. cow, cow, cow, cow, cow, cow, cow, cow, cow, cow
2. A black and white cow standing on top of a grass covered field.

Figure 7.3: Example of how Oscar deals with the image of a cow in unnatural colors. 1. - Detected object tags by the detector, 2. - Prediction my custom pipeline[2].

1. bed, bed, bed, bed, bed, potted plant, bed, suitcase, couch
2. a bed sitting on top of a lush green field.

Figure 7.4: Example of how Oscar deals with the image of a bed in a unnatural environment - forest. 1. - Detected object tags by the detector, 2. - Prediction my custom pipeline[3].

[2] `https://aura.ch/webyep-system/daten/9-49-gl-fineart-9712.jpg`
[3] `https://www.presseportal.de/pm/110692/4121876`

Figure 7.4 - **Bed in the forest**: I found it interesting to generate the image caption where the object is not in its natural environment. For this purpose, I used the example of an image of a bed in the forest. Oscar generated a caption that describes the image well but did not use the word *forest*. This is probably due to the fact that the word *forest* or *tree* is not in the set of the COCO class names. It is interesting to compare this with the image in Figure 7.5, where Oscar recognized the forest. I assume this is due to the context of the objects in the training data. While the forest around the river and the boat is fairly natural, the forest around the bed is not as common and thus probably not in the training data.

1. boat, boat, boat, boat, bird, bird, boat, bird
2. a boat sitting on top of a river next to a forest.

Figure 7.5: This image serves as a comparison with the image in the Figure 7.4. The forest was recognised here. 1. - Detected object tags by the detector, 2. - Prediction my custom pipeline. Courtesy of [3].

Chapter 8

Conclusion

In this work, my goal was to analyze an image captioning state-of-the-art method and perform experiments on it. Within that, I wanted to gain knowledge of the image captioning task and deep learning in general. For this purpose, I decided to focus on a specific state-of-the-art method Oscar [13]. It is a transformer-based method. Transformer [40] is a deep learning architecture, that was introduced in 2017, used for sequence processing. In 2020, it has been adopted for image processing [41, 42]. At the time I decided to use it, Oscar was providing the best results on the image captioning task on one of the most popular datasets in use, COCO Caption [11]. When analyzing how Oscar works, I found that it generates a caption for a given image that has already been pre-processed. The authors have provided a pre-processed set of image features from the COCO [3] dataset that can be used as input but do not specify how they were obtained from the source images. In order to generate a caption for a custom image that was not part of the provided dataset, I decided to use the Faster-R-CNN architecture [46] implemented within Detectron2 [45] as the detector to extract the features from the source images. I conducted experiments to find out how to set up the detector properly and how to restructure the data so that they can be used as Oscar input. Based on the acquired knowledge, I generated features from the COCO 2017 dataset to fine-tune Oscar. In this way, I created a pipeline that is capable of generating a caption for any given image. The fine-tuned Oscar has been evaluated on the validation split of the COCO 2017 dataset so that it can be compared to the performance presented in the source document [13]. My pipeline achieves worse results than presented in the original paper. This is an expected result due to the lower variability of the training data. The details are described in Chapter 5.

Oscar is a multi-modal system. It uses a visual modality represented as image features and in addition it uses object tags as a textual modality. This fact gives the opportunity and motivation to conduct an ablation study experiment. In the experiment, which was discussed in detail in Chapter 6, I investigate how removing one of the modalities affects the results. The experiment confirmed Oscar's sensitivity to both modalities. The results

I obtained suggest that the visual modality is more important than the textual modality. However, based on the knowledge I gained during this work, I believe that the difference of sensitivities in the different modalities would not be as large if both Oscar and the detector were trained on more variable data.

My custom pipeline achieves the following score: BLEU-4: 0.312, METEOR: 0.272, CIDEr: 1.02, and SPICE: 0.201. Even though the achieved results are not as good as original Oscar published, I still consider the result of my work a success. I managed to implement a pipeline that can generate a caption for any image using Oscar. This was not possible from public sources at the time I started my work. Part of my work has involved creating tools to help me analyze data, create datasets, edit them, etc. These can be found on GitHub[1].

During the course of working on the thesis, I came to the conclusion that Oscar is a reasonably robust method. It is able to generate meaningful captions even with a reduced amount of information. It can furthermore generate captions even for images with a small number of objects detected or objects unseen in the training data. I suggest that these facts open up possibilities for further research on the method Oscar in the feature. For example by using more variable training data or more sophisticated approach to detection filtering.

[1] `https://github.com/zeleznyt/Image_captioning_custom_Oscar`

Bibliography

[1] J. Pavlopoulos, V. Kougia, and I. Androutsopoulos, "A survey on biomedical image captioning," in Proceedings of the second workshop on shortcomings in vision and language, pp. 26–36, 2019.

[2] S. Das, L. Jain, and A. Das, "Deep learning for military image captioning," in 2018 21st International Conference on Information Fusion (FUSION), pp. 2165–2171, IEEE, 2018.

[3] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in European conference on computer vision, pp. 740–755, Springer, 2014.

[4] P. Young, A. Lai, M. Hodosh, and J. Hockenmaier, "From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions," Transactions of the Association for Computational Linguistics, vol. 2, pp. 67–78, 02 2014.

[5] C. Rashtchian, P. Young, M. Hodosh, and J. Hockenmaier, "Collecting image annotations using amazon's mechanical turk," in Proceedings of the NAACL HLT 2010 workshop on creating speech and language data with Amazon's Mechanical Turk, pp. 139–147, 2010.

[6] L. Zhou, H. Palangi, L. Zhang, H. Hu, J. Corso, and J. Gao, "Unified vision-language pre-training for image captioning and vqa," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 13041–13049, 2020.

[7] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 3128–3137, 2015.

[8] Y. Zeng, X. Zhang, and H. Li, "Multi-grained vision language pre-training: Aligning texts with visual concepts," arXiv preprint arXiv:2111.08276, 2021.

[9] K. Li, Y. Zhang, K. Li, Y. Li, and Y. Fu, "Visual semantic reasoning for image-text matching," in Proceedings of the IEEE/CVF International conference on computer vision, pp. 4654–4662, 2019.

[10] K.-H. Lee, X. Chen, G. Hua, H. Hu, and X. He, "Stacked cross attention for image-text matching," in Proceedings of the European Conference on Computer Vision (ECCV), pp. 201–216, 2018.

[11] 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.

[12] P. Wang, A. Yang, R. Men, J. Lin, S. Bai, Z. Li, J. Ma, C. Zhou, J. Zhou, and H. Yang, "Unifying architectures, tasks, and modalities through a simple sequence-to-sequence learning framework," arXiv preprint arXiv:2202.03052, 2022.

[13] X. Li, X. Yin, C. Li, P. Zhang, X. Hu, L. Zhang, L. Wang, H. Hu, L. Dong, F. Wei, et al., "Oscar: Object-semantics aligned pre-training for vision-language tasks," in European Conference on Computer Vision, pp. 121–137, Springer, 2020.

[14] J. Guo, S. Lu, H. Cai, W. Zhang, Y. Yu, and J. Wang, "Long text generation via adversarial training with leaked information," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32, 2018.

[15] L. Yu, W. Zhang, J. Wang, and Y. Yu, "Seqgan: Sequence generative adversarial nets with policy gradient," in Proceedings of the AAAI conference on artificial intelligence, vol. 31, 2017.

[16] P. Sharma, N. Ding, S. Goodman, and R. Soricut, "Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning," in Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 2556–2565, 2018.

[17] R. Mokady, A. Hertz, and A. H. Bermano, "Clipcap: Clip prefix for image captioning," arXiv preprint arXiv:2111.09734, 2021.

[18] D. Elliott, S. Frank, K. Sima'an, and L. Specia, "Multi30k: Multilingual english-german image descriptions," arXiv preprint arXiv:1605.00459, 2016.

[19] I. Calixto, Q. Liu, and N. Campbell, "Incorporating global visual features into attention-based neural machine translation," arXiv preprint arXiv:1701.06521, 2017.

[20] H. Lin, F. Meng, J. Su, Y. Yin, Z. Yang, Y. Ge, J. Zhou, and J. Luo, "Dynamic context-guided capsule network for multimodal machine translation," in Proceedings of the 28th ACM International Conference on Multimedia, pp. 1320–1329, 2020.

[21] D. Kiela, H. Firooz, A. Mohan, V. Goswami, A. Singh, P. Ringshia, and D. Testuggine, "The hateful memes challenge: Detecting hate speech in multimodal memes," Advances in Neural Information Processing Systems, vol. 33, pp. 2611–2624, 2020.

[22] R. Velioglu and J. Rose, "Detecting hate speech in memes using multimodal deep learning approaches: Prize-winning solution to hateful memes challenge," arXiv preprint arXiv:2012.12975, 2020.

[23] J. Pont-Tuset, J. Uijlings, S. Changpinyo, R. Soricut, and V. Ferrari, "Connecting vision and language with localized narratives," in European Conference on Computer Vision, pp. 647–664, Springer, 2020.

[24] K. Yan, L. Ji, H. Luo, M. Zhou, N. Duan, and S. Ma, "Control image captioning spatially and temporally," in Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pp. 2014–2025, 2021.

[25] 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 of the Association for Computational Linguistics, pp. 311–318, 2002.

[26] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," in Text summarization branches out, pp. 74–81, 2004.

[27] S. Banerjee and A. Lavie, "Meteor: An automatic metric for mt evaluation with improved correlation with human judgments," in Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization, pp. 65–72, 2005.

[28] R. Vedantam, C. Lawrence Zitnick, and D. Parikh, "Cider: Consensus-based image description evaluation," in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4566–4575, 2015.

[29] P. Anderson, B. Fernando, M. Johnson, and S. Gould, "Spice: Semantic propositional image caption evaluation," in European conference on computer vision, pp. 382–398, Springer, 2016.

[30] 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.

[31] R. Staniūtė and D. Šešok, "A systematic literature review on image captioning," Applied Sciences, vol. 9, no. 10, p. 2024, 2019.

[32] 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.

[33] K. Nithya and V. V. Kumar, "A review on automatic image captioning techniques," in 2020 International Conference on Communication and Signal Processing (ICCSP), pp. 0432–0437, IEEE, 2020.

[34] M. Chohan, A. Khan, M. S. Mahar, S. Hassan, A. Ghafoor, and M. Khan, "Image captioning using deep learning: A systematic," Image, vol. 11, no. 5, 2020.

[35] A. Farhadi, M. Hejrati, M. A. Sadeghi, P. Young, C. Rashtchian, J. Hockenmaier, and D. Forsyth, "Every picture tells a story: Generating sentences from images," in European conference on computer vision, pp. 15–29, Springer, 2010.

[36] H. Fang, S. Gupta, F. Iandola, R. K. Srivastava, L. Deng, P. Dollár, J. Gao, X. He, M. Mitchell, J. C. Platt, et al., "From captions to visual concepts and back," in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1473–1482, 2015.

[37] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 3156–3164, 2015.

[38] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in International conference on machine learning, pp. 2048–2057, PMLR, 2015.

[39] P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang, "Bottom-up and top-down attention for image captioning and visual question answering," in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 6077–6086, 2018.

[40] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," Advances in neural information processing systems, vol. 30, 2017.

[41] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in European conference on computer vision, pp. 213–229, Springer, 2020.

[42] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al., "An image is worth 16x16 words: Transformers for image recognition at scale," arXiv preprint arXiv:2010.11929, 2020.

[43] N. Li, Z. Chen, and S. Liu, "Meta learning for image captioning," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 8626–8633, 2019.

[44] C. Chen, S. Mu, W. Xiao, Z. Ye, L. Wu, and Q. Ju, "Improving image captioning with conditional generative adversarial nets," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 8142–8150, 2019.

[45] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2." `https://github.com/facebookresearch/detectron2`, 2019.

[46] 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, 2015.