Universitat Oberta de Catalunya (UOC)

Máster Universitario en Ciencia de Datos (*Data Science*)

TRABAJO FINAL DE MÁSTER

Área: Big Data / Machine Learning

Deep Learning for Image Captioning

An Encoder-Decoder Architecture with Soft Attention

Autor: Mario Gómez Martínez

Tutor: Anna Bosch Rué

Profesor: Jordi Casas Roma

Valencia, June 28, 2019

Copyright

FICHA DEL TRABAJO FINAL

Título del trabajo:	Deep Learning for Image Captioning. An Encoder-Decoder Architecture with Soft Attention
Nombre del autor:	Mario Gómez Martínez
Nombre del colaborador/a docente:	Anna Bosch Rué
Nombre del PRA:	Jordi Casas Roma
Fecha de entrega (mm/aaaa):	06/2019
Titulación o programa:	Máster en Ciencia de Datos
Área del Trabajo Final:	Aprendizaje automático
Idioma del trabajo:	Inglés
Palabras clave	Aprendizaje Profundo, Descripción de Imágenes
Código fuente	https://github.com/magomar/image_captioning_with_attention

Dedicatoria

Dedicado a mi compañera, siempre ahí, para lo bueno y para lo malo, cercana, constante, inspiradora...

... y a los tres retoños que inundan cada día de juegos inesperados y alegría arrolladora.

Abstract

Automatic image captioning, the task of automatically producing a natural-language description for an image, has the potential to assist those with visual impairments by explaining images using text-to-speech systems. However, accurate image captioning is a challenging task that requires integrating and pushing further the latest improvements at the intersection of computer vision and natural language processing fields

This work aims at building an advanced model based on neural networks and deep learning for the automated generation of image captions.

Keywords: Deep Learning, Artificial Neural Networks, Automated image captioning

Resumen

El subtitulado automático de imágenes, la tarea de producir automáticamente una descripción en lenguaje natural para una imagen, tiene el potencial de ayudar a las personas con discapacidades visuales a explicar las imágenes mediante sistemas de conversión de texto a voz. Sin embargo, el subtitulado preciso de imágenes es una tarea desafiante que requiere integrar y avanzar en la intersección de los campos de procesamiento de lenguaje natural y visión por computador.

Este trabajo pretende desarrollar un modelo basado en redes neuronales y aprendizaje profundo para la generación automática de descripciones de imágenes.

Palabras clave: Aprendizaje Profundo, Redes Neuronales Artificiales, Descripción automática de imágenes

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The web provides a vast amount of information, including a lot of text, but it is increasingly dominated by visual information, both static (pictures) and dynamic (videos). However, much of that visual information is not accessible to those with visual impairments, or with slow internet speeds that prohibit the loading of images. Image captions, manually added by content providers (typically by using the *Alt-text* HTML tag), is one way to make this content more accessible, so that a *text-to-speech* system can be applied to generate a natural-language description of images and videos. However, existing human-curated image descriptions are added for only a very small fraction of web images. Thus, there is great interest in developing methods to automatically generate image descriptions.

Automatic image description, also known in the research community as image captioning, can be defined as the task of automatically generating a description of an image using natural language. It is a very challenging problem that encompasses two kinds of problems: the problem of understanding an image, which is a **Computer Vision (CV)** task and the problem of generating a meaningful and grammatically-correct description of the image, which is a kind of **Natural-Language Processing (NLP)** task. Therefore, to tackle this task it is necessary to advance the research in the two fields, CV and NLP, as well as promoting the cooperation of both communities to address the specific problems arising when combining both tasks.

Figure 1.1 shows an example of the automatic image generation tasks addressed by this project.

There are other use cases in which automatically generated image captions may help. Generally speaking, any domain in which images need to be interpreted by humans, but human availability is scarce, or the task at hand is tedious, may surely benefit from algorithms able to automatically generate textual image descriptions.

Figure 1.1: Image captioning can help millions with visual impairments by converting images captions to text. Image by Francis Vallance (Heritage Warrior), used under CC BY 2.0 license.

A vast area of application is that of **Concept-Based Image Retrieval**. This kind of task, also named as "description-based" or "text-based" image indexing/retrieval, refers to retrieval from text-based indexing of images that may employ keywords, subject headings, captions, or natural language text. The main problem with this approach to image retrieval is the scarcity of image descriptions, since having humans manually annotate images by entering keywords or descriptions in a large database can be very tedious and time consuming [1]. Therefore, automatic image description may be of great utility, and it can be applied to many areas requiring image indexing and retrieval, such as biomedicine, education, digital libraries, etc., as well as general web search. In addition, there is an extra benefit in using captions vs keywords: image captions are semantically richer and more sophisticated, thus allowing for more complex queries to improve the precision of the search.

Another area where automatic image captioning may be of great utility is the analysis and extraction of information from videos. Some video monitoring tasks are very boring and tedious. An automated mechanism to describe scenes in video footage will be of great utility for creating summaries or monitoring specific situations and events. [2].

Image captioning can also be used to improve image indexing since it provides a more sophisticated and semantically rich description that image classification or simple image tagging. This kind of semantic indexes would be very very useful for any kind of Content-Based Image Retrieval application, which would help improve any king of Content-Based Image Retrieval. This kind of Having images automatically generated captions for images that originally hasn't been described, would help to find images based on a complex description rather than a simple

[1] This is one of the reasons explaining the upswing of *Content-Based Image Retrieval*, which uses content derived from the visual properties of the image, like color, textures, and shape, instead of semantic and descriptive data such as keywords and captions

[2] As an interesting example, Shell is conducting a pilot study using deep-learning to automatically monitor video footage in order to identify safety hazards and generate alerts (link)

collection of tags.

1.2 Goals

This project aims at advancing in the task of automatically generating image descriptions. That is the ultimate goal of the project. However, in order to achieve such an abstract goal, we decompose it into various subgoals, as follows:

1. Get a solid understanding of the problem at hand and review the state-of-art solutions to it
2. Get practical knowledge on the technologies required to solve this problem
3. Develop a model using a benchmark dataset like the Flickr30K
4. Scale the model to a larger benchmark dataset such as the COCOLin et al. (2014) dataset or the more recent Conceptual Captions Dataset by Google Sharma et al. (2018) (see Fig. 1.2).
5. Evaluate the model, ideally partaking in some challenge or competition, like the ones using the COCO dataset or the Conceptual Captions dataset.

1.3 Methodology

This project is mainly an academical, research-oriented project, so it follows a process model which is common for this kind of projects. For instance, it includes an extensive survey of the state of the art and proposes a model to solve the target problem. However, this project will also include the development of a software artifact to implement the proposed model using real data. As such, this project can benefit from a data-analytic model as the well known and widely adopted **CRISP-DM**. CRISP-DM, which stands for *Cross-Industry Standard Process for Data Mining*, is an open standard process model and an industry-proven methodology to guide data mining projects.

As a methodology, it includes descriptions of the typical phases of a project, the tasks involved with each phase, and an explanation of the relationships between these tasks.

As a process model, CRISP-DM provides an overview of the data mining life cycle.

Figure 1.3 depicts the relationships between the different phases of the CRISP-DM model. The sequence of the phases is not strict and moving back and forth between different phases is often required. The arrows in the process diagram indicate the most important and frequent

Figure 1.2: Illustration of images and captions in the Conceptual Captions dataset.Clockwise from top left, images by Jonny Hunter, SigNote Cloud, Tony Hisgett, ResoluteSupportMedia. All images used under CC BY 2.0 license.

dependencies between phases. The outer circle in the diagram symbolizes the cyclic nature of data mining itself. A data mining process continues after a solution has been deployed. The lessons learned during the process can trigger new, often more focused business questions and subsequent data mining processes will benefit from the experiences of previous ones.

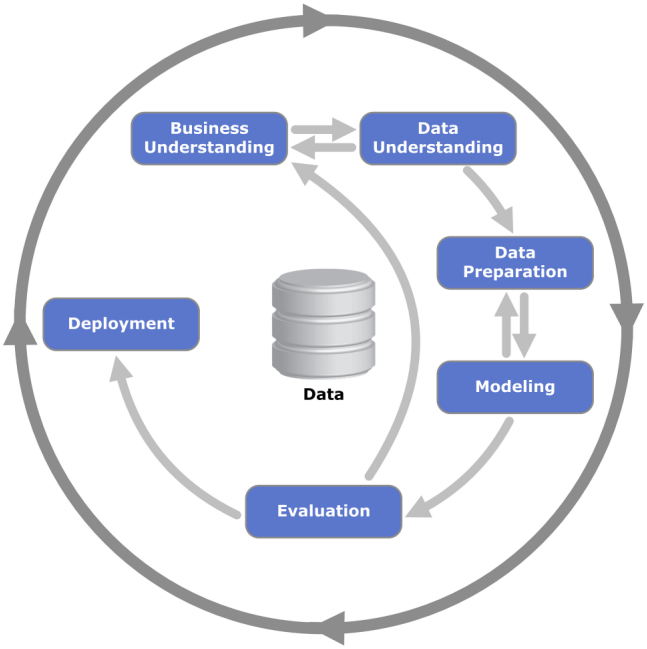


Figure 1.3: Process diagram showing the relationship between the different phases of CRISP-DM. Image by Kenneth Jensen, used under CC BY-SA 3.0 license..

The process model consists of six major phases:

- **Business Understanding**: Includes in-depth analysis of the business objectives and needs. The situation is assessed and the goals of the project are defined. This should follow the setting up of a plan to proceed.
- **Data Understanding**: Conduct initial or exploratory data analysis to become familiar with data and identify potential problems. Examine the properties of data and verify its quality by answering questions concerning the completeness and accuracy of the data.
- **Data Preparation**: After the data sources are completely identified, proper selection, cleansing, constructing and formatting should be done before modeling.
- **Modeling**: Modeling is usually conducted in multiple iterations, which involves running several models using the default parameters and then fine-tune the parameters or revert to the data preparation phase for additional preparation. Usually, there are different ways to look at a given problem, so it is convenient to build multiple models,

- **Evaluation**: The results of models are evaluated in the backdrop of business intentions. New objectives may sprout up owing to the new patterns discovered. This is, in fact, an iterative process, and the decision whether to consider them or not has to be made in this step before moving on to the final phase
- **Publication**. The final information gathered has to be presented in a usable manner to the stakeholders. This has to be done as per their expectations and business requirements.

1.4 Planning

This section briefly describes the initial planning established to accomplish this Final Master's Thesis.

1.4.0.1 Original planning

Table 1.1: Main tasks and milestones.

Phase	End	Description
1	3/3/2019	Definition and planning
2	24/3/2019	State of the Art
3	19/5/2019	Development
4	9/6/2019	Complete this report
5	16/6/2019	Presentation

Table 1.1 sums up the original planning for this project. It consisted of 5 phases:

Phase 2 is devoted to surveying the state of the art. This phase will encompass the following tasks:

- Reviewing relevant bibliography
- Studying the problem domain (business understanding), and becoming familiar with the data. At this stage, we would also start preparing the data for the modeling stage

Phase 3 is intended to design and develop a software artifact. It is the longest phase, and includes the following tasks:

- Preparing the data. Although data preparation could be started during phase 2, depending on the chosen models it could be necessary to conduct some additional data preparation operations

- Generating one or various models. We plan at creating at least two models, one that would replicate existing work, and another one to explore new ideas and try to beat the baseline model.
- Evaluating the models, and very specifically, compare our model against the replicated model.
- Publication: We consider two courses of action: a) participating in the Microsoft COCO Image Captioning Challenge, and b) delivering some product to the final user, although it would be a very basic prototype given the time available.

Phase 4 is conceived for completing this report. This phase would probably overlap with some of the tasks in phase 3 that will require additional time, like the evaluation of the models and the participation in challenges.

Phase 5 is the last one. At this phase, the results of the project should be published, including both the report and any code and documentation. Finally, the project has to exposed and defended publicly, to be evaluated by an academic board.

Chapter 2

State of the Art

Recently there has been an upsurge of interest in problems that require a combination of linguistic and visual information. Besides, the rise of social media in the web has made available a vast amount of multimodal information, like tagged photographs, illustrations in newspaper articles, videos with subtitles, and multimodal feeds on social media. To tackle combined language and vision tasks and to exploit the large amounts of multimodal information, the CV and NLP communities have been increasingly cooperating, for example by organizing combined workshops and conferences. One such area of research in the intersection of both worlds is automatic image description.

Automatic image description can be defined as the task of automatically generating a description of an image using natural language. It is a very challenging problem that combines two different problems into a single task: on the one hand, there is the problem of understanding an image, which belongs to the **Computer Vision (CV)** field, one the other hand, there is also the problem of generating a meaningful and grammatically-correct description of the image, which belongs to the **Natural-Language Processing (NLP)** field, and to be more precise, it belongs to the class of **Natural-Language Generation (NLG)** problems.

Both CV and NLP are challenging fields themselves. While both fields share common techniques rooted in artificial intelligence and machine learning, they have historically developed separately, with little interaction between their scientific communities. Recent years have seen considerable advances in both fields, to a great extent thanks to the application of deep-learning techniques and the recent advances in this area. This chapter presents a brief survey of the recent literature on this topic, including some antecedents, but focusing primarily on the recent advances coming from the application of **Deep Learning** technology since this is our main interest.

The chapter is organized in various sections. The first section is devoted to further delimiting the task at hand as well as introducing a classification schema for the different approaches to the

problem. Subsequent sections review relevant publications organized according to the provided classification scheme. Finally, there is a section describing the datasets used by the community to benchmark their models and a short discussion on the evaluation metrics for this kind of tasks.

2.1 Task definition and classification of methods

We have already defined the task of automatic image description as the task of automatically generating a description of an image using natural language generation. However, this definition is too generic to precisely characterize the task we are interested in. For example, when presented with certain image, an algorithm may generate a list of labels describing different elements of the image, or it may describe technical features of the image, such as the dimensions, the predominant colors, brightness, etc. Therefore, we need a more concrete definition of the task.

When talking about **automatic image description**, we refer to descriptions that meet three properties:

- Descriptions that are relevant, that is descriptions that talk about the elements of the image.
- Descriptions that are expressed as natural language, using grammatically correct sentences
- Descriptions that are comprehensive but concise at the same time, that is, the description should aim at summing up the important elements of the image, not just describing it.

From the CV point of view, this task requires **full image understanding**: the description should demonstrate or pretend a good understanding of the scene, far beyond simply recognizing the objects in the image. This means that the description is able to capture relations between the objects in the scene, and the actions happening there.

From the NLP point of view, this task requires **sophisticated natural language generation** (NLG), which involves: selecting which aspects to talk about (content selection), sorting and organizing the content to be verbalized (text planning), and finally generating a semantically and syntactically correct sentence (Surface realization).

Intuitively, descriptions should be easy to understand by a person, and that person should be able to grasp the essence of the image, to create a mental model of the image without actually seeing it. The description task can become even more challenging when we take into account user-specific tailored descriptions. For instance, when describing the paintings available in a museum, a tourist may require a different description than a librarian or an art critic.

Since 2011 there has been a considerable advance in challenging CV tasks, to a great extent fostered by the application of deep learning models and the availability of large corpus of data available to researchers. More recently, a similar process seems to be occurring in the NLP field. Not surprisingly, these advances in both CV and NLP have also propelled a new wave of interest in cross-disciplinary research problems involving both areas of research, and automatic image description is a very good example. As a consequence, the CV and NLP communities have increased cooperation, for example by organizing joint workshops over the past few years. These efforts have resulted in a surge of new models, datasets and evaluation measures, which is reflected in the increase of publications, especially from 2014.

In order of ease of review, understanding, and comparison of the growing amount of research on the topic, existing surveys have proposed various schemes to classify the models being used.

One the one hand, the survey by Bernardi et al. (2017) proposes a classification system based on two dimensions and only three categories. On the other hand, Bai and An (2018) organize the existing research according to the kind of architecture or framework used, resulting in a more fine-grained classification with 8 categories.

After comparing both approaches to classify the existing research, we prefer the approach adopted by Bai and An (2018) as we consider it more precise and descriptive, resulting in a finer granularity; while the classification by Bernardi et al. (2017) is more abstract, resulting in a coarser granularity. A more recent survey by Hossain et al. (2019) takes the same approach found in (Bai and An, 2018) with a more focused review of deep-learning based models and references to the most recent work published so far.

Below we provide a short overview of the publications covered in this survey organized into categories and sorted by publication year:

2.2 Early work: methods that are not based on deep learning

This section reviews the initial attempts to solve the image captioning problem. All these methods have in common that they do not use deep learning techniques. We divide them into two groups: retrieval based approaches and template-based approaches.

2.2.1 Retrieval-based

Early work on the topic was often based on the use of retrieval-based approaches, also referred to as transfer-based approaches. These approaches usually follow a two steps process. During the first step, given a query image, a candidate set of similar images is retrieved using content-

Table 2.1: Summary of published research on the automatic image description problem.

Approach	Representative research
Retrieval-based	Farhadi et al. (2010); Ordonez et al. (2011); Gupta et al. (2012); Kuznetsova et al. (2012); Hodosh et al. (2013); Kuznetsova et al. (2014); Mason and Charniak (2014); Hodosh and Hockenmaier (2013)
Template-based	Yang et al. (2011); Kulkarni et al. (2011); Li et al. (2011); Mitchell et al. (2012); Ushiku et al. (2015)
Earlier Deep Models	Socher et al. (2014); Karpathy et al. (2014); Ma et al. (2015); Yan and Mikolajczyk (2015); Lebret et al. (2015a); Yagcioglu et al. (2015)
Multimodal learning	Kiros et al. (2014a); Mao and Yuille (2015); Karpathy and Fei-Fei (2015); Chen and Zitnick (2015)
Encode-Decoder framework	Kiros et al. (2014b); Vinyals et al. (2015); Donahue et al. (2015); Jia et al. (2015); Wu et al. (2016); Pu et al. (2016b); Gan et al. (2017a); Hao et al. (2018)
Compositional architectures	Fang et al. (2015); Tran et al. (2016); Ma and Han (2016); Oruganti et al. (2016); Wang et al. (2016); Fu et al. (2017); Gan et al. (2017b)
Attention-guided	Xu et al. (2015); You et al. (2016); Yang et al. (2016); Zhou et al. (2017); Khademi and Schulte (2018); Anderson et al. (2018b); Jiang et al. (2018)
Describing novel objects	Mao et al. (2015); Hendricks et al. (2016)
Other deep learning methods	Pu et al. (2016a); Dai et al. (2017); Shetty et al. (2017); Ren et al. (2017); Rennie et al. (2017); Zhang et al. (2017); Anderson et al. (2018a); Feng et al. (2018); Gu et al. (2018); Li et al. (2018); Li and Chen (2018); Lindh et al. (2018)

based image retrieval techniques, which are based on global image features extracted from the image. During the second step, a re-ranking of the retrieved images is computed using a variety of methods. Finally, the caption of the top image is returned, or a new caption is composed of the captions of the top-n ranked images.

Farhadi et al. (2010) use a meaning space consisting of <*object, action, scene*> triplets to link images and sentences. Their model takes an input image, map it into the meaning space by solving a Markov Random Field, and use Lin's information-based similarity measure (Lin, 1998) to determine the semantic distance between the query image and the pool of available captions. Finally, the semantically closest sentence is used as the image description.

The IM2TEXT model by Ordonez et al. (2011) uses the scene-centered descriptors of the GIST model (Oliva and Torralba, 2006; Torralba et al., 2008) to retrieve a set of similar images as a baseline, then these images are ranked using a classifier trained over a range of object

detectors and scene classifiers specific to the entities mentioned in the candidate descriptions. Finally, the caption of the top-ranked image is returned.

Kuznetsova et al. (2012) take on the work by Ordonez et al. (2011) with some notable twists: instead of retrieving a candidate set of images using content –visual– features, their model start by running the detectors and scene classifiers used in the re-ranking step of the IM2TEXT model to extract and represent the semantic content of the query image. Then, instead of performing a single retrieval step to get neighbors of the query image, their model carries out a separate retrieval step for each visual entity detected in the query image. The result of this step is a collection of phrases of different type (noun and verb phrases and prepositional phrases). Finally, a new sentence is composed from a selected set of phrases using a constraint optimization approach. A refinement of the former approach is presented in Kuznetsova et al. (2014), based on the use of tree structures to improve the sentence generation process.

Gupta et al. (2012) combine the image features of the GIST model with RGB and HSV color histograms, Gabor and Haar descriptors, and SIFT (Lowe, 2004) descriptors for the retrieval step. Then, instead of using the visual features for the ranking step, their model relies on the textual descriptions of the retrieved images, which are segmented to obtain phrases of a different type. This model takes the phrases associated to the retrieved images, rank them based on image similarity, and integrate them to get triples of the form *(((attribute1, object1), verb), (verb, prep, (attribute2, object2)), (object1, prep, object2))*. Finally, the tree top-n triplets are used to generate an image description.

Hodosh and Hockenmaier (2013) employ a Kernel Canonical Correlation Analysis (KCCA) (Bach and Jordan, 2003) technique to project images and text items into a common space, where training images and their corresponding captions are maximally correlated. In the new common space, cosine similarities between images and sentences are calculated to rank the sentences, which are then used to select the top-n captions. This work is highly focused on the problems associated with existing benchmarks datasets and the evaluation metrics, and as a result, the authors introduce a new dataset composed of 8000 images annotated with 5 captions per image and propose a new metric based on binary judgments of image descriptions.

Mason and Charniak (2014) use visual similarity to retrieve a set of candidate captioned images. Second, a word probability density conditioned on the query image is computed from the retrieved captions. Finally, the candidate captions are scored using the word probability density and the one with the highest score is selected.

A detailed analysis of the literature reviewed here reveals at least two dimensions that may further help in understanding and organizing the research: on the one hand, some approaches simply select one of the retrieved sentences as the image caption, while others compose a new caption by combining elements from several sentences; on the other hand, some approaches use

visual detectors to find similar images (Content-Based Image Retrieval), while other approaches use conceptual information (Concept-Based Image Retrieval), or use a combination of visual and conceptual information for the retrieval step. Table 2.2 sums up the various dimensions used to classify retrieval-based methods to the image captioning problem.

Table 2.2: Classification of retrieval-based image captioning.

Approach	Representative research	
Information modality	Select one caption	Compose caption
Visual	Farhadi et al. (2010); Mason and Charniak (2014)	Gupta et al. (2012)
Conceptual	Ordonez et al. (2011)	Kuznetsova et al. (2012, 2014)
Hybrid	Hodosh and Hockenmaier (2013)	

The main disadvantage of Retrieval-Based approaches comes from the reuse of existing images. The sentences used to describe images in the past may be completely inadequate for describing a new image. Although some of the proposed models are able to compose new sentences rather than just returning one of the stored sentences, these are still based on previously used sentences, so the result may still be unsuited for describing the query image. This limitation was strikingly evident with the use of small datasets, therefore, perhaps the problem would be alleviated with the use of very large datasets as the ones that have appeared recently Lin et al. (2014); Sharma et al. (2018).

2.2.2 Template-based approaches

Another group of early attempts to solve the automatic image captioning problem takes some kind of template-based approach. Unlike retrieval based approaches, template-based approaches analyze the query image to generate concepts and then use some kind of constraining mechanism to compose a sentence. The constraints typically adopt the form of a template, but can also be specified as grammar rules.

For example, Yang et al. (2011) use a quadruplet consisting of $<$*Noun-Verb-Scene-Preposition*$>$ as template for generating image descriptions. The process starts with the execution of detection algorithms to estimate objects and scenes in the image, and then apply a language model trained over the Gigaword corpus[1] to compute nouns, verbs, scenes and prepositions that might appear in the caption. Finally, they apply Hidden Markov Model Inference to compute probabilities for all the elements and use the elements with the highest probabilities to fill the template and generate a new sentence.

[1] https://catalog.ldc.upenn.edu/LDC2003T05

Kulkarni et al. (2011) employ Conditional Random Field (CRF) to determine image contents, which results in a graph with nodes corresponding to objects in the image, object attributes, and spatial relationships between objects. Unary potential functions are computed for nodes in the graph, while pairwise potential functions are obtained on a collection of existing descriptions. Afterward, CRF inference is used to determine the image contents to be described, and finally, a sentence template is applied to generate a sentence using the selected content.

Li et al. (2011) use visual models to detect objects, attributes and spatial relationships, and. Then, they resort to web-scale n-gram data to compute frequency counts of possible n-gram sequences to fill a triplet defined as *<(adj1, obj1), prep, (adj2, obj2)>*. Finally, dynamic programming is applied to find an optimal set of phrases that constitute the description of the query image.

Mitchell et al. (2012) apply algorithms that are able to represent an image as an <objects, actions, spatial relationships> triplet. Syntactically informed word co-occurrence statistics are computed and used by a sentence generator to filter and constrain the output of the image processor using tree structures.

The aforementioned works in this section produce new sentences based on individual words (nouns, adjectives, verbs, prepositions, etc.) that generated in a piece-wise manner from the query image, and these words are later connected according to certain grammar model. To improve on this approach, some authors have proposed the use of phrases instead of individual words. In particular, Ushiku et al. (2015) present a method named Common Subspace for Model and Similarity (CoSMoS). CoSMoS obtains a subspace in which all feature vectors associated with the same phrase are mapped as mutually close, classifiers for each phrase are learned, and training samples are shared among co-occurring phrases. Phrases estimated from a query image are connected by using multi-stack Beam search to generate a description.

Template-based approaches to automatic image captioning may improve the relevance of the resulting captions when compared with retrieval based approaches, template-based captions tend to be too rigid, thus resulting in a lack of naturality when compared to human-written sentences.

2.3 Deep-learning approaches

Convolutional Neural Networks (CNN) were first introduced by Yann LeCun in 1998 (Lecun et al., 1998), but it was not until more than a decade later than they started to shine. In 2012, a large, deep CNN (Krizhevsky et al., 2012) was used to win, by an incredibly wide margin, the 2012 ImageNet Large-Scale Visual Recognition Challenge. From that turning point, the field

has attracted the attention of researchers from various fields and gained howling popularity. The success of CNN and other deep learning models was due to a great extent to its impressive results in many challenging problems, prominently in the fields of Computer Vision and Natural Language Processing, where deep-learning models have become the current state-of-the-art. Not surprisingly, researchers have attempted to apply deep neural networks to solve problems in the interstice of both fields, CV and NLP, includes the problem of automatically generating image descriptions.

Recent advances in the field have been achieved with the introduction of new architectures and frameworks, accompanied by increasingly larger datasets to feed those deep neural networks. Therefore, there is a wide repertoire of methods for tackling the image captioning tasks. This section presents the contributions to the field organized according to the kind of architecture or framework utilized, with one subsection per category

2.3.1 Earlier Deep Models

Encourage by the success of CNN to solve image classification tasks, researchers began incorporating deep models into their image captioning methods, yet still influenced by the retrieval-based and template-based methods. Image captioning was formulated as a multi-modality embedding Frome et al. (2013) and ranking problem.

Socher et al. (2014) use dependency-tree recursive neural networks (DT-RNN) to represent phrases and sentences as compositional vectors. Another deep neural network (Le et al., 2011) is used to extract visual features from the images. Both types of features are mapped into a common space by using a max-margin objective function. After training, sentence retrieval is performed based on similarities between representations of images and sentences in the common space.

Karpathy et al. (2014) introduce a twist in the previous model by working at a finer level, that is, instead of modeling full images, they work on a finer level by embedding fragments of images as well as fragments of sentences fragments into a common space. A structured max-margin objective is used to explicitly associate these fragments across modalities. Image fragments are obtained by means of a Region CNN (Girshick et al., 2014). Sentence fragments are modeled as typed dependency tree relations (De Marneffe et al., 2006). At last, a retrieval task is performed by computing image-sentence similarities that combine both a global term and a fragment-aligned term. Experimental evaluation showed that reasoning on both the global level of images and sentences and the finer level of their respective fragments improves performance on the image-sentence retrieval task.

Ma et al. (2015) take into consideration different levels of interaction between images and sentences in order to compute similarities. Their architecture combines two different deep

neural networks to tackle the multimodal space: an image encoding CNN (Simonyan and Zisserman, 2015) to encode visual data, and a matching CNN (Hu et al., 2014) to learn the joint representation of images and sentences. The matching CNN composes words to different semantic fragments and learns the inter-modal relations between image and the composed fragments at different levels.

Yagcioglu et al. (2015) propose a query expansion approach for improving transfer-based automatic image captioning. The core idea of this method is to translate the given visual query into a distributional semantics based form, which is generated by the average of the sentence vectors extracted from the captions of images visually similar to the input image. It is a data-driven approach, that extracts image features using the Caffe architecture (Jia et al., 2014), a Region-CNN pipeline trained on the ImageNet dataset. The original query is expanded as the average of the distributed representations of the captions associated with the retrieved descriptions (Mikolov et al., 2013), weighted by their similarity. Finally, they transfer the closest caption as the description of the input image.

Devlin et al. (2015) also utilizes CNN activations as the global image descriptor, and perform k-NN retrieve images in the training set that are similar to the query image. Then their model selects a description from the candidate descriptions associated with the retrieved, just like the approaches by Mason and Charniak (2014) and Yagcioglu et al. (2015). Their approach differs in terms of how they represent the similarity between description and how they select the best candidate over the whole set. Specifically, they propose to compute the description similarity based on the *n-gram* overlap *F-score* between the descriptions. The model returns the description with the highest mean n-gram overlap with the other candidate descriptions, that is the one associated with the k-NN centroid.

2.3.2 Multimodal learning

Retrieval-based and template-based methods to image captioning impose limitations on the capacity to describe images in the form of templates, structured prediction, and/or syntactic trees. Thanks to the advances in neural networks, new approaches emerged that were able to surpass those limitations. These methods can yield more expressive and flexible sentences with richer structures. Multimodal neural language models constitute one way of approaching the problem from a learning perspective. In general, these models are bidirectional, that is, they are able to generate new captions for an image, but they can also be applied in retrieval tasks for both images and sentences.

The general structure of multimodal neural-based learning is shown in Fig. 2.1. First, image features are extracted using a feature extractor, typically a CNN. Then, the extracted features are forwarded to a neural-based model, which maps the image features into a common space

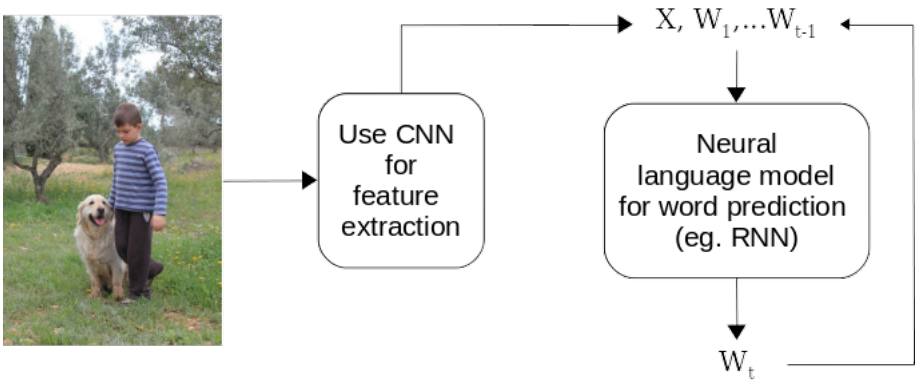


Figure 2.1: Structure of multimodal neural learning models for image captioning

with the word features. Finally, the model predicates new words based on the image feature and previously generated context words.

Kiros et al. (2014a) use a CNN for extracting image features using a multimodal space that jointly represents image and text. The key to achieving that is the use of Multimodal Neural Language Models (MNLM), that is, models of natural language that can be conditioned on other modalities. An image-text multimodal neural language model can be used to retrieve images given complex sentence queries, retrieve phrase descriptions given image queries, as well as generate text conditioned on images. The authors introduce two such languages that are adaptations of the Log-Bilinear Language Model proposed by Mnih and Hinton (2007). In the case of image-text modeling, it is possible to jointly learn word representations and image features by training the models together with a CNN like AlexNet (Krizhevsky et al., 2012).

Mao et al. (2014); Mao and Yuille (2015) present a multimodal Recurrent Neural Network (m-RNN) model for generating novel image captions. It directly models the probability distribution of generating a word given previous words and an image and uses this distribution to generate image captions. The model consists of two sub-networks: a deep RNN for sentences and a deep CNN for images. These two sub-networks interact with each other in a multimodal layer to form the whole m-RNN model. Besides generating captions, the m-RNN model can be applied for retrieving images or sentences. This model achieved state-of-the-art results across image captioning and image and caption retrieval tasks.

Karpathy and Fei-Fei (2015) propose another multimodal approach using deep neural networks to learn to embed both image and natural language data for the task of bidirectional images and sentences retrieval. However, unlike previous methods that embed full images and sentences, this method works at a finer level, by working with fragments of images and fragments of sentences. This method applies Region-CNN to break down an image into a number of objects, RNN over sentences, represented by a dependency tree relations (DTR) (De Marneffe

et al., 2006), and a structured objective that aligns the two modalities through a multimodal embedding. The resulting Multimodal RNN architecture learns to generate novel descriptions of image regions. This model outperformed retrieval baselines on full images as well as images annotated with region-level sentences, which were derived from the MSCOCO dataset.

Chen and Zitnick (2015) also explore the bi-directional mapping between images and their textual descriptions. This approach uses an RNN that attempts to dynamically build a visual representation of the scene as a caption is being generated or read. The representation automatically learns to remember long-term visual concepts. This model is capable of both generating novel captions given an image, and reconstructing visual features given an image description. The model was tested on several tasks, including sentence generation, sentence retrieval, and image retrieval, and achieved state-of-the-art results for the task of generating novel image descriptions. The results for the image and sentence retrieval tasks were comparable to state-of-the-art methods using similar visual features.

In general, the multimodal approaches have limitations to handle a large amount of data and are inefficient to work with long term memory.

2.3.3 The Encoder-Decoder framework

The **encoder-decoder framework** was originally designed to translate sentences between different languages (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Cho et al., 2014). Inspired by its success in solving translation problems, some researchers have adopted them to generate image captions with great success. The idea behind adapting this framework designed for translation tasks was to see the image captioning tasks also as a translation problem but across different modalities.

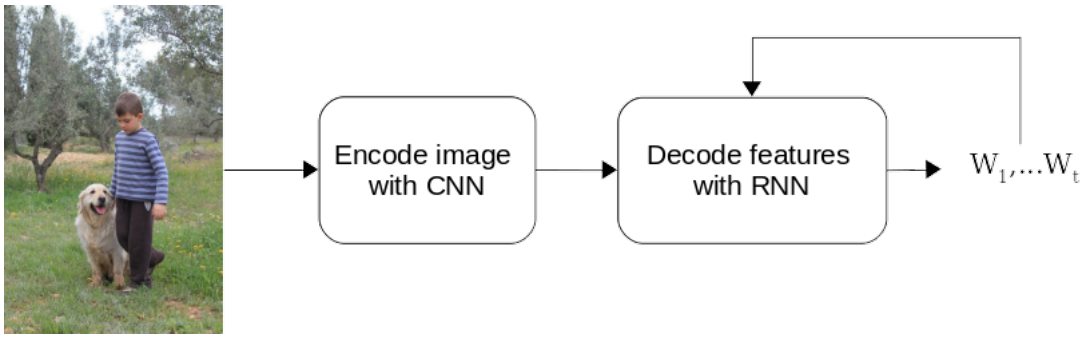


Figure 2.2: Structure of image captioning methods based on the encoder-decoder framework

Kiros et al. (2014b) were the first to apply the encoder-decoder framework to generate image descriptions by unifying joint image-text embedding models and multimodal neural language

models. The idea is that given an image input, a sentence output can be generated word by word, like in language translation. Their model uses a Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) to encode textual data, and a CNN to encode visual data. Then, by optimizing a pairwise ranking loss, encoded visual data is projected into an embedding space spanned by LSTM hidden states that encode textual data. In the embedding space, a multimodal neural language model –the so-called Structure-Content Neural Language Model (SC-NLM)– is used to decode the visual features conditioned on context word feature vectors, thus allowing for the word-by-word generation of a sentence. This method achieved significant improvements in generating realistic image captions over the multimodal neural language models proposed in (Kiros et al., 2014a).

Vinyals et al. (2015) propose a method similar to the one by Kiros et al. (2014a), which uses CNN for image representations and an LSTM for generating image captions. This method trains the LSTM based on maximum likelihood estimation. The LSTM works as follows: (1) image information is included into the initial state of the LSTM; (2)next words are generated based on the current time step and the previous hidden state; (3) this process continues until it gets the end token of the sentence.

A problem with the former encoder-decoder methods is their susceptibility to the vanishing gradient problem, due to image information being fed only at the beginning of the process. Consequently, the role of the initial words is also becoming weaker and weaker as the process continues. Therefore, the captions tend to degrade in relevance when generating long sentences. With the aim of mitigating this problem, Donahue et al. (2015) introduce a variation into the encoder-decoder architecture that stacks multiple LSTM atop one another. Rather than projecting the vision space into the embedding space of the hidden states at the initial stage, this model takes a copy of the static image and the previous word directly as input, that is then fed to the stack of LSTMs at each time step. This architecture referred to as the Long-term Recurrent Convolutional Network (LRCN) model, is able to process not just static images, but also dynamic tasks involving sequences, like videos.

Still playing around with the encoder-decoder framework, some researchers propose methods to augment it by including high-level semantic concepts explicitly.

Jia et al. (2015) propose an extension of the LSTM model, which they call guided LSTM of gLSTM for short. This model adds semantic information from the image, and this information is included as an extra input to each gate and cell state of the LSTM network, with the aim of guiding the model towards more relevant solutions, particularly when generating long sentences. Semantic information is extracted in different ways. One way is by means of a cross-modal retrieval task that first retrieves image captions and then extracts semantic information from these captions Another way is by using a multimodal embedding space. Besides, this work

explores different length normalization strategies for beam search to avoid bias towards short sentences, obtaining results that are on par with or better than the current state-of-the-art.

Wu et al. (2016) also incorporate high-level semantic concepts into the encoder-decoder framework in the form of attribute probabilities. To this end, the authors first mined a set of semantic attributes based on words commonly found in image captions. Second, they trained a multi-label classifier using one CNN per attribute, to predict the probability of attributes being present in the image. The trained CNNs are used to create a vector for each image that represents the probability of each attribute being present in it. For the caption generation, the prediction vector is passed to an LSTM. For different tasks, different language models may be used, and specifically, for image captioning, they adopt the region-based multi-modal language model by Vinyals et al. (2015). The model is also capable to incorporate external semantic information and doing so further improves performance.

A circumstance that may limit the practical application of the majority of approaches discusses so far will occur when the availability of captions is limited to just a fraction of the available images. In that case, a semi-supervised learning approach would be of significant practical values. With that situation in mind, Pu et al. (2016a) propose a semi-supervised learning method under the encoder-decoder framework. This method uses a Deep Generative Deconvolutional Network (DGDN) (Pu et al., 2016b) as a decoder of the latent image features, and a deep CNN as an image encoder; the CNN is used to approximate a distribution for the latent DGDN features. The latent features are also linked to generative models of captions by the RNN decoder. To generate captions for new images, averaging is performed across the distribution of latent features. This framework is capable of modeling the image in the absence of associated captions, thus this is a semi-supervised setting for CNN learning with images.

2.3.4 Attention-guided models

Image captions should be kept short but informative. Therefore, such descriptions should mention only the most salient elements of the image, and avoid small details. Motivated by the visual attention mechanism of primates and humans (Rensink, 2000; Spratling and Johnson, 2004), approaches that utilize attention to guide image description generation are proposed. By incorporating attention to the encoder-decoder framework, sentence generation will be conditioned on hidden states that are computed based on the attention mechanism. The general structure of attention-guided image captioning methods is depicted in Fig. 2.3. In such methods, the attention mechanisms based on various kinds of cues from the input image are incorporated to make the decoding process focus on certain aspects of the input image during the caption generation process.

Encouraged by successes of other tasks that employ attention mechanism Mnih et al. (2014);

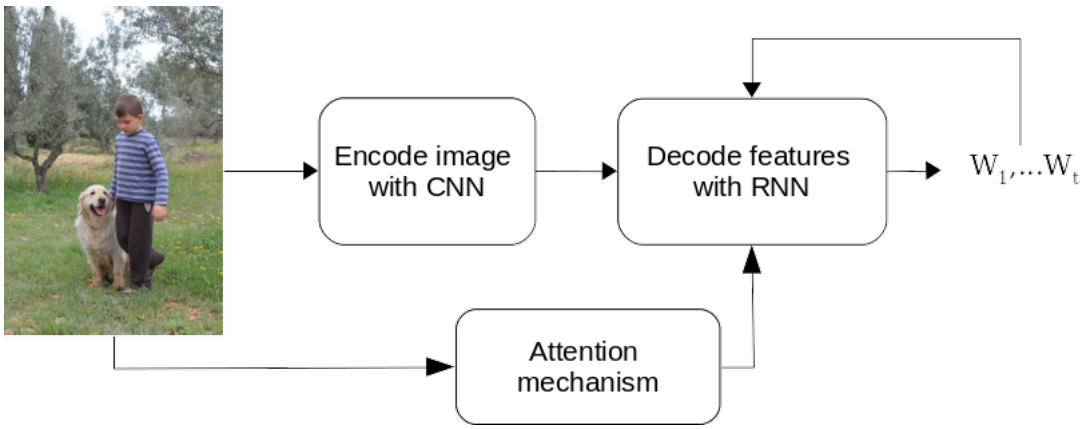


Figure 2.3: Structure of image captioning methods based on the encoder-decoder framework

Lei Ba et al. (2015), Xu et al. (2015) were the first to introduce an attention-mechanism to the problem of automatic image captioning. Their model, which is a variation of the encoder-decoder framework, is capable of dynamically attending salient image regions while generating the description of the image. While most approaches based on CNN image encoders use the top layer of the ConvNet for extracting image features, this method generates context vectors by using the features learned at lower layers of the ConvNet. The idea behind this approach is that the use of the top layer may result in the loss of details which may actually be useful for generating image descriptions. Xu et al. have tried two different techniques for simulating attentions: stochastic hard attention and deterministic soft attention.

It is possible to classify many of the methods used in image captioning as either bottom-up or top-down. In the top-down paradigm (Donahue et al., 2015; Karpathy and Fei-Fei, 2015; Chen and Zitnick, 2015; Mao and Yuille, 2015; Mao et al., 2015; Vinyals et al., 2015; Xu et al., 2015) visual features are extracted first and used to select or generate a caption. In the bottom-up paradigm (Farhadi et al., 2010; Kulkarni et al., 2011; Li et al., 2011; Kuznetsova et al., 2012; Elliott and Keller, 2013; Lebret et al., 2015a) visual concepts (e.g., regions, objects, and attributes) are extracted first and combined later to generate a caption. Top-down approaches have difficulties to deal with the fine details of an image. Bottom-up approaches, on the other hand, can operate at any image resolution and therefore are more suited for dealing with fine details of the image. However, they have problems with formulating an end-to-end process. To take advantage of the complementary properties of both approaches, You et al. (2016) propose a semantic attention-based method that is designed to provide a detailed, coherent description of semantically important objects. On the one hand, the visual concepts are collected using K-NN image retrieval for computing global –non-parametric– visual concepts, and a fully convolutional network (FCN) (Long et al., 2015) is trained to predict region attributes. With each attribute

corresponding to one entry of the used vocabulary, words and image attributes share the same vocabulary. Under the encoder-decoder framework, the visual concepts are only forwarded to the encoder at the initial step. In the decoding stage, an LSTM is applied to decode the image features using both the global visual concepts and the local attributes. The LSTM decoder uses two functions to generate words: an output attention function that modulates attribute attention based on the hidden state of the LSTM, and an input feedback mechanism that modulates attribute attention based on the previous word. Compared with the method introduced by Xu et al. (2015), which works on a fixed and predefined spatial location, this semantic attention-based method can work on any resolution and any location of the image.

Arguing that attention-enriched encoder-decoder models lack global modeling abilities due to their sequential information processing, Yang et al. (2016) propose a new model called review network, which results of augmenting the encoder-decoder framework with a review module. The review network is generic and can enhance any existing encoder-decoder model. This model performs a number of review steps with an attention mechanism on the encoder hidden states and outputs a thought vector after each review step. These thought vectors are introduced to capture the global properties of the image in a compact vector representation that is used as the input of the attention mechanism in the decoder. For example, the first review step can review what are the objects in the image, then it can review the relative positions of the objects, and another review can extract the information of the overall context of the image. Another role for the thought vectors is as a focus for multitask learning.

Pedersoli et al. (2017) propose an area based attention mechanism for image captioning. Previous attention based methods map image regions only to the hidden state of the RNN language model. This approach, however, allows a direct association between caption words and image regions by modeling the dependencies between image regions, caption words, and the hidden states. This association makes the model capable of predicting the next word as well as corresponding image regions in each time-step. Another contribution of this work is the introduction of spatial transformation networks that allow for image-specific attention areas (while previous attention mechanisms used CNN activation grids, object proposals), and can be trained jointly with the rest of the network. This combination of techniques together yields state-of-the-art results on the MSCOCO dataset.

Lu et al. (2017) proposed another attention-based mechanism for image captioning using a visual sentinel. Other attention-based methods focus on the image in every time step of the recurrent neural network. However, in practice there are some words that do not need visual information (e.g. some prepositions and articles), thus applying attention to the image in these steps may be misleading and diminish the overall effectiveness of the caption generation process. To deal with this drawback, Lu et al. introduce an adaptive mechanism to dynamically

determine when to look at the image, and when to relay just on the language model to generate the next word. The proposed model combines a new spatial model that is able to determine where to look and a sentinel gate, which determines when to look at the image. The adaptive attention mechanism is introduced as an extension of the decoder LSTM that generates "visual sentinel" vectors (instead of just a hidden state) fallback option for the decoder. The "sentinel gate" is responsible of deciding whether to look at the image and how much information to get from it, or on the contrary, whether to rely on the visual sentinel for the caption generation process.

As we have seen, various attention mechanisms have successfully been added to deep models for improving the generation of image captions. These deep models learn dynamic weightings of the input vectors, which allow for more flexibility and expressive power than other models. Attention maps between the language space and the and language space are supposed to contain relevant information that could be useful in understanding and improving deep learning models in problems involving image and natural language, such as the image captioning problem.

With the aim of advancing in the understanding and utility of attention mechanisms in image captioning, Liu et al. (2017a) propose a quantitative method to evaluate the correctness of attention maps, which is defined as the consistency between attention maps and the corresponding regions that the words/phrases describe in the image. More specifically, Liu et al. use the alignment annotations between image regions and noun phrase caption entities provided in the Flickr30k Entities dataset (Plummer et al., 2015) as ground truth maps. Using that metric, the attention-based model by Xu et al. (2015) performs better than the uniform attention baseline, but still has room for improvement in terms of attention consistency with human annotations. To improve the quality of attention maps, the authors propose the use of supervised attention, with two types of supervision: strong supervision with alignment annotation, and weak supervision with semantic labeling. In experiments, the method shows that supervised attention model performs better in mapping attention as well as image captioning tasks.

Chen et al. (2017a) argue that existing visual attention based solely on spatial features have some drawbacks. To begin with, these methods compute weighted pooling only on attentive feature maps, which results in a gradual loss of spatial information. Moreover, these methods use the spatial information from the last convolution layer of the CNN, whose receptive fields correspond to very large regions, with very narrow gaps between regions; therefore, they do not get significant spatial attention for an image. To deal with these limitations, Chen et al. propose a method that combines both a spatial attention mechanism with a channel-wise attention mechanism. The new method, named SCA-CNN, extracts features not just from spatial locations, but also from different channels and multiple layers of the CNN. In addition,

each filter of a convolutional layer acts as semantic detector (Zeiler and Fergus, 2014) while other methods use external sources for obtaining semantic information. In the task of image captioning, SCA-CNN dynamically modulates the sentence generation context in multi-layer feature maps, encoding where (i.e., attentive spatial locations at multiple layers) and what (i.e., attentive channels) the visual attention is. This method outperformed visual attention-based image captioning methods across the Flickr30K and MSCOCO datasets.

To bridge the gap between humans and machines in image understanding and describing, Tavakoliy et al. (2017) study the agreement between bottom-up saliency-based visual attention and object referrals in scene description constructs. As a result of the study, Tavakoliy et al. propose a saliency-boosted image captioning model that investigates possible benefits from low-level cues in language models. There are two main conclusions of the study: first, that humans mention more salient objects earlier than less salient ones in their descriptions; and second, the better a captioning model performs, the better attention agreement it has with human-generated descriptions. The authors found that the proposed saliency-boosted model does not improve significantly when the task is well learned, but they observed a better generalization of their model when applied upon unseen data.

Visual attention plays an important role to understand images and demonstrates its effectiveness in generating natural language descriptions of images. On the other hand, recent studies show that language associated with an image can steer visual attention in the scene during our cognitive process. Inspired by this, Mun et al. (2017) introduce a text-guided attention model which learns to drive visual attention using associated captions. This model introduces an exemplar-based learning approach that retrieves from training data associated captions with each image, and use them to learn attention on visual features. This attention model enables to describe a detailed state of scenes by distinguishing small or confusable objects effectively.

The are two main approaches to attention-based mechanisms: either use the low-level visual features to localize objects (Xu et al., 2015; Lu et al., 2017), or utilize the high-level semantic features to describe objects' attributes (Wang et al., 2017; Wu et al., 2016), whilst the inner connections of these two types of features are not utilized. Inspired by the visual processing of our cognitive system, Li and Chen (2018) propose a model that combines low-level visual features and high-level semantic features (attributes). The low-level features are extracted by a Region Proposal Network (RPN) whilst the high-level features are obtained by a CNN. The caption generation module is an LSTM with visual and semantic cells. The visual cell LSTM utilizes the visual features to localize the objects in the image, whilst the semantic cell LSTM further integrates the localized objects with their attributes to generate corresponding words. In addition, this method also uses Reinforcement Learning (the REINFORCE algorithm (Sutton et al., 1999)) to explore proper vocabularies in the training process. The RL operates

by introducing state perturbations to the LSTM units.

Top-down attention-mechanisms for building visual attention maps typically focus on some selective regions obtained from the output of one or two layers of a CNN. The input regions are of the same size and have the same shape of the receptive field. Anderson et al. (2018b) argue that this approach has little consideration for the content of the image and propose a new method inspired by the human visual system. Recent studies on human attention (Buschman and Miller, 2007) indicate that human attention can be focused volitionally by top-down signals determined by the task at hand (e.g., looking for something), and automatically by bottom-up signals associated with unexpected, novel or salient stimuli. Inspired by these findings, Anderson et al. introduce a top-down mechanism based on Faster R-CNN Ren et al. (2015). This mechanism proposes image regions, each with an associated feature vector, while a top-down mechanism determines feature weightings. Therefore, this method can attend both object level regions as well as other salient image regions.

Hao et al. (2018) propose another model based on the popular encoder-decoder architecture, but it uses the recently proposed densely convolutional neural network (DenseNet) to encode the feature maps. Meanwhile, the decoder uses an LSTM to parse the feature maps and converse them into textual descriptions. This model predicts the next word in the description by taking the effective combination of feature maps with word embedding of current input word by a visual attention switch.

Khademi and Schulte (2018) present a context-aware attention-based method that employs a Bidirectional Grid LSTM (BiGrLSTM), which takes visual features of an image as input and learns complex spatial patterns based on two-dimensional context, by selecting or ignoring its input. In addition, this method leverages a set of local region-grounded texts obtained by transfer learning. The region-grounded texts often describe the properties of the objects and their relationships in an image. To generate a global caption for the image, this method integrates the spatial features from the Grid LSTM with the local region-grounded texts, using a two-layer Bidirectional Grid LSTM. The first layer models the global scene context such as object presence. The second layer utilizes a dynamic spatial attention mechanism to generate the global caption word-by-word while considering the caption context around a word in both directions. Unlike recent models that use a soft attention mechanism, this attention mechanism considers the spatial context of the image regions.

Jiang et al. (2018) propose an extension of the encoder-decoder framework by adding a component called guiding network. The guiding network models the attribute properties of input images, and its output is leveraged to compose the input of the decoder at each time step. The guiding network can be plugged into the current encoder-decoder framework and trained in an end-to-end manner. Hence, the guiding vector can be adaptively learned according to the

signal from the decoder, making itself to embed information from both image and language. Additionally, discriminative supervision can be employed to further improve the quality of guidance.

2.3.5 Compositional architectures

Most of the neural methods for image captioning described in the previous sections are end-to-end systems made of tightly coupled modules whose parameters are trained jointly. An alternative architectonic approach to build such systems is based on the notion of compositional architectures, which consist of independent, loosely coupled components connected through a pipeline, where each component provides data to the next component until a final result is obtained. The idea behind this architecture is to make modular systems where it is easier to reuse existing components and replace components with little effort.

The general structure of compositional image captioning methods is depicted in Fig. 2.4. Usually, there are three components:

1. A visual model that obtains visual features from the query image.
2. A natural language model that generates captions for the extracted features
3. A ranking module that re-ranks generated captions using a multimodal similarity model.

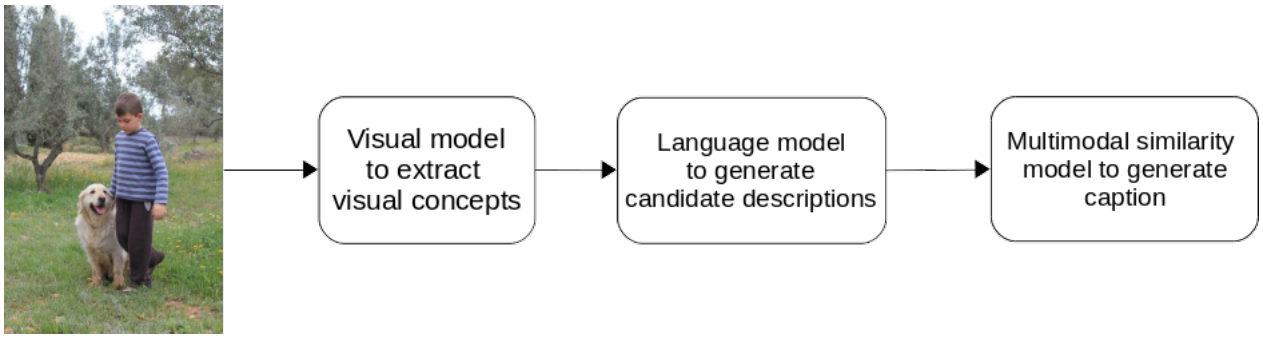


Figure 2.4: Structure of image captioning methods based on the encoder-decoder framework

Fang et al. (2015) describe a system that consists of three models: visual detectors, language models and multimodal similarity models. The training process consists of two steps: First, a vocabulary of words that are most common in the training captions. Second, a visual detector is trained for each word in the vocabulary using multiple-instance learning (Viola et al., 2005). Once trained, the system can be used to generate image captions: A CNN (Krizhevsky et al., 2012) is used to obtain visual features expressed as words learned vocabulary, then a maximum entropy language model (Berger et al., 1996) is applied to generate candidate captions

conditioned on those features. During this process, left-to-right beam search (Ratnaparkhi, 2000) with a stack of pre-specified length l, produces l candidate descriptions. Finally, a deep multimodal similarity model, which maps images and text fragments into a common space for similarity measurement, is used to re-rank the candidate descriptions and select the top one.

Taking on the same architecture introduced by Fang et al. (2015), Tran et al. (2016) present a system specialized in describing landmarks and celebrities. However, instead of a CNN for the visual feature detector, this model uses a deep residual network (ResNet) to detect a broader range of visual concepts He et al. (2016), a maximum entropy language model for candidate description generation, and a deep multimodal similarity model for caption ranking.

Ma and Han (2016) propose another method using a compositional architecture. Their main contribution is the introduction of structural words; tetrads composed of <objects, attributes, activities, scenes>. The method consists of two stages: structural word generation and sentence translation. At the first stage, multi-layer optimization is applied to generate a hierarchy of concepts that will play the role of structural words. During the second stage, an LSTM is used to translate the structural words into full sentences.

Realizing the high specialization of most approaches to multi-modal learning problems, Oruganti et al. (2016) introduce a more general approach called the Fusion-based Recurrent Multi-Modal (FRMM) model. This model is made of three independent components: one component per modality, and a fusion stage. Each modality is learned in a separate stage, and then their outputs are mapped to a shared representation. Finally, the fusion generates an output based on the associations between modalities it has previously learned. This approach provides the FRMM architecture increased flexibility over previous approaches in accommodating disparate inputs. The architecture for each stage might be different depending on the modalities involved, ensuring that the overall architecture is highly adaptable. The FRMM model is designed to work with multi-modal applications in which at least one of the modalities has a temporal or sequential nature, such as image description, sentiment analysis, and language translation. Furthermore, two variations of the FRMM architecture are proposed according to the lengths of the inputs in each modality: Aligned FRMM (AFRMM) when the length is the same, and Unaligned FRMMM (UFRMM) otherwise. For image captioning in particular, the AFRMM model achieved the best results. This models obtained better results for big datasets like MSCOCO, but not for smaller datasets like Flickr30K, probably because it needs to learn more parameters than other models.

Wang et al. (2016) propose a parallel version of the compositional architecture with the goal of taking advantage of the complementary properties of simple RNNs one the one hand, and LSTMs networks on the other hand. There is a visual stage using CNN, like many other approaches, as well as a language generator and a fusion step. The contribution of this work

lies in the parallelization of the generation and fusion stages, which consists in dividing the hidden units of the RNN into several same-size parts and letting them work in parallel. Then, their outputs are merged with corresponding ratios for word prediction. This parallelization allows for the use of different types of hidden units to combine their strengths. Specifically, the authors have used various configurations of simple RNN and LSTM. Experimental results showed a reduction in resource consumption accompanied by a simultaneous improvement in performance, compared with the dominated models (models using just one type of hidden units, either simple RNN or LSTM).

Gan et al. (2017b) introduce another compositional architecture based on the encoder-decoder framework. Their model, called Semantic Compositional Network (SCN) extracts semantic tags from the image, and the probability of each tag is used to compose the parameters in an LSTM network. The SCN extends each weight matrix of the LSTM to an ensemble of tag-dependent weight matrices. The degree to which each member of the ensemble is used to generate an image caption is tied to the image-dependent probability of the corresponding tag. The same architecture can be used to describe video clips.

Fu et al. (2017) introduce an approach that aims at exploiting the parallel structures between images and sentences. In this model, the generation of the next word is aligned with the visual perception experience, where the attention shifting among the visual regions imposes a thread of visual ordering. In addition, this method uses high-level semantic information in the form of scene-specific contexts. This method works as follows: First, selective search (Uijlings et al., 2013) is applied to construct a hierarchical segmentation of the image containing image regions/patches at different scales. Second, a small number of regions are selected for further processing based on the criterion of being semantically meaningful, non-compositional and contextually rich. Next, visual features are extracted from regions using a deep CNN. The sentence generation process is guided by an attention-based multi-modal LSTM decoder. The visual attention mechanism is inspired by the model described in (Xu et al., 2015), but it is enriched with information representing the scene-specific context. This information is modeled as scene vectors obtained by Latent Dirichlet Allocation (LDA) (Blei et al., 2003) and a multilayer perceptron.

2.3.6 Other approaches

2.3.6.1 Language Convolutional Neural Networks

Since 2014, recurrent neural networks have been applied with great success to various sequence learning tasks, including the image caption generation problem. Although traditional RNNs suffer from vanishing and exploding gradient problems and cannot adequately handle long-

term temporal dependencies, this problem is mitigated by modern architectures like LSTM Hochreiter and Schmidhuber (1997) and GRU(Chung et al., 2014). Not surprisingly, these network architectures have also dominated the image captioning problem.

However, LSTM/GRUs ignore the underlying hierarchical structure of a sentence. They also require significant storage due to long-term dependencies through a memory cell. In contrast, CNNs can learn the internal hierarchical structure of the sentences and they are faster in processing than LSTMs. Therefore, recently, convolutional architectures are being increasingly used in sequence-to-sequence tasks, like conditional image generation and machine translation. Inspired the success of CNN in sequence learning tasks, some authors are proposing CNN as an alternative to RNN for the language generation part of their image captioning methods. The result is a CNN+CNN alternative to the dominant CNN+RNN paradigm.

The first attempt to actually use CNN for the language model of an image captioning system was the multimodal convolutional neural network (m-CNN) proposed by Ma et al. (2015), that we already described as an earlier deep learning model.

More recently, Gu et al. (2017) introduce a language CNN model which is suitable for statistical language modeling tasks and shows competitive performance in image captioning. In contrast to previous models which predict next word based on one previous word and hidden state, this language CNN is fed with all the previous words and can model the long-range dependencies in history words, which are critical for image captioning. However, the method cannot model the dynamic temporal behavior of the language model using just a language-CNN, thus it combines the language CNN with a recurrent network to model the temporal dependencies properly.

Aneja et al. (2018) propose a fully convolutional architecture (CNN+CNN) for the task of image captioning. This method uses a feed-forward network without any recurrent function. The architecture of the method has four components: (i) input embedding layer (ii) image embedding layer (iii) convolutional module, and (iv) output embedding layer. It also uses an attention mechanism to leverage spatial image features. The evaluation on the MSCOCO dataset shows comparable performance to an LSTM based method on standard metrics

Wang and Chan (2018) also propose a CNN+CNN image captioning method. It is similar to the method by Aneja et al., except that it includes a hierarchical attention module to connect the vision-CNN with the language-CNN. The authors of this method also investigate the use of various hyperparameters, including the number of layers and the kernel width of the language-CNN. They show how the influence of the hyperparameters can improve the performance of the method in image captioning.

2.3.6.2 Reinforcement Learning

Most image captioning methods are trained by maximizing the likelihood of ground-truth annotated captions given the image. While simple and easy to implement, this approach does not directly maximize language quality metrics such as CIDEr. In order to improve on this kind of metrics, recently proposed methods include some form of Reinforcement Learning as a way to improve the quality of generated captions with respect to this kind of metric. However, RL faces various problems when applied to language generation problems. First, unlike images, which are represented using real numbers, text processing is based on discrete numbers, which results in non-differentiable functions, making it difficult to apply back-propagation directly. However, this problem can be overcome by using the REINFORCE family of algorithms, which introduces Policy gradient with differentiable function approximation (Sutton et al., 1999).

Rennie et al. (2017) propose a new optimization approach called self-critical sequence training (SCST). SCST is a form of the popular REINFORCE algorithm that, rather than estimating a base-line to normalize the rewards and reduce variance, utilizes the output of its own test-time inference algorithm to normalize the rewards it experiences. Using this approach, estimating the reward signal (as actor-critic methods must do) and estimating normalization (as REINFORCE algorithms typically do) is avoided, while at the same time harmonizing the model with respect to its test-time inference procedure. Empirically, the authors find that directly optimizing the CIDEr metric with SCST and greedy decoding at test-time is highly effective.

The method proposed by Li and Chen (2018) also includes a REINFORCE algorithm to improve their captions by optimizing the CIDEr metric. This method is described in the section devoted to attention-based methods, so it is not described here again..

Zhang et al. (2017) investigate an actor-critic reinforcement learning (Barto et al., 1983) approach. The architecture of this method consists of a policy network (actor) and a value network (critic). The actor treats the job as a sequential decision problem and can predict the next token of the sequence. In each state of the sequence, the network will receive its evaluation metrics score as a reward. The job of the critic is to predict the reward. If it can predict the expected reward, the actor will continue to sample outputs according to its probability distribution.

The existing image captioning approaches typically train a one-stage sentence decoder, which makes it difficult to produce fine-grained descriptions. On the other hand, multi-stage sentence decoders are hard to train due to the vanishing gradient problem. Gu et al. (2018) propose a coarse-to-fine multi-stage decoder composed of a number of stacked decoders each of which operates on the output of the previous stage, producing increasingly refined image descriptions. The vanishing gradient problem is tackled by providing a learning objective function that enforces intermediate supervisions. In particular, the model is optimized with a

reinforcement learning method which utilizes the output of each intermediate decoder's test-time inference algorithm as well as the output of its preceding decoder to normalize the rewards. This method also solves the exposure bias problem and the loss-evaluation mismatch problem.

2.3.6.3 Generative Adversarial Networks

Despite phenomenal research progresses in the past several years, sentences produced by methods based on Recurrent Neural Networks tend to be rigid and lacking in variability because they learn by maximizing the likelihood of ground-truth annotated captions in the training examples. This principle encourages high resemblance to the ground-truth captions, thus missing the opportunity to generate other reasonable descriptions. Conventional evaluation metrics, e.g. BLEU and METEOR, also favor such restrictive methods.

Generative Adversarial Networks (GAN) provides a possible way to generate more diverse and distinctive captions. GANs can learn deep features from unlabeled data. They achieve these representations applying a competitive process between a pair of networks: the Generator and the Discriminator.

Dai et al. (2017) introduce a method based on GAN with the aim to improve the naturalness and diversity of generated captions. This method, called Conditional GAN (CGAN), jointly learns a generator to produce descriptions conditioned on images, and an evaluator to assess how well a description fits the visual content. GANs face two problems when compared with RNN. First, like Reinforcement Learning, it is difficult to apply back-propagation directly, but similarly, this problem can be tackled by introducing policy gradient mechanism with differentiable function approximation (Sutton et al., 1999). Second, sequence generation faces vanishing gradients and error propagation training difficulties. A way to mitigate this problem is to provide early feedback to the mitigator. Following this idea, CGAN adds Monte Carlo rollouts to compute future reward values at early stages of the process (Yu et al., 2017). This method only generates one caption per image.

Shetty et al. (2017) introduce another GAN based method that is able to generate multiple captions for a single image. Instead of function approximation used by other methods to deal with the discrete data backpropagation problem, this method uses Gumbel sampler (Jang et al., 2017). During training, the generator learns the loss value provided by the discriminator instead of learning it from explicit sources. The discriminator has true data distribution and can discriminate between generator-generated samples and true data samples. This allows the network to learn diverse data distribution. Moreover, the network classifies the generated caption sets into either real or fake.

Li et al. (2018) propose a conditional GAN for generating diverse captions across images. Instead of estimating the quality of a caption solely on one image, the proposed comparative

adversarial learning framework assesses the quality of captions by comparing a set of captions within the image-caption joint space. The proposed architecture, called Comparative Adversarial Learning (CAL) Network consists of a caption generator and a comparative relevance discriminator which play a min-max game and optimize the loss function. With the guides from the discriminator, the generator effectively learns to generate more specific and distinctive captions, hence increases the diversity across the corpus.

2.3.6.4 Other methods

The majority of methods described in former sections are variations of the encoder-decoder framework following the CNN+RNN schema, that is to say, the encoder uses Convolutional Neural Networks Lecun et al. (1998), and the decoder uses Recurrent Neural Networks, such as LSTM(Hochreiter and Schmidhuber, 1997) or GRU(Chung et al., 2014).

There is also a new wave of models that replace the RNN encoder module with some form of CNN for language modeling (CNN+CNN methods) or variants of the Generative Adversarial Training, but in general, all these methods still rely on a single CNN for the encoder layer. That is, the CNN feature map of the image is taken as the source input to the decoder module.

An exception to the standard CNN+RNN approach is the model proposed by Liu et al. (2017b), called MAT, which stands for Multimodal Attentive Translator. The main idea behind MAT is to view the image captioning problem as a specific form of translation task, from a sequence of objects into a sequence of words. To that aim, instead of modeling the input image as a feature map, as most CNN-based approached do, the input image has to be represented as a sequence of detected objects. In order to model the image as a sequence, MAT introduces a sequential attention layer to selectively attend to the objects that are related in order to generate corresponding words. Here, the input layer expands the image representation from a single CNN at a single time step in RNN, to a sequence of objects at multiple time steps. This architecture combines CNN object detectors with LSTM for the encoding module, a second LSTM for the decoding module, plus an attention module.

The majority of deep learning approaches to the image captioning problem take the supervised learning route, with some exceptions that use some form of semi-supervised learning(Pu et al., 2016a; Anderson et al., 2018a). However, Feng et al. (2018) propose a completely unsupervised approach to training an image captioning model. Instead of relying on manually labeled image-sentence pairs, this method merely requires an image set, a sentence corpus, and an existing visual concept detector. The sentence corpus is used to teach the language generation model how to generate plausible sentences. Meanwhile, the knowledge in the visual concept detector is distilled into the captioning model to guide the model to recognize the visual concepts in an image. Furthermore, the image and caption are projected into a common latent

space so that they can reconstruct each other. Given that the existing sentence corpora are mainly designed for linguistic research, the authors have also crawled a large-scale image description corpus of two million natural sentences to facilitate the unsupervised image captioning scenario. Experimental results show promising results without any caption annotations.

Lindh et al. (2018) also introduce a form of unsupervised learning, although their model still relies on supervised learning for the most part. These authors address the low diversity and specificity in generated sentences that limit many image captioning methods by introducing an Image Retrieval model to improve the caption specificity of the Image Captioning model. This model uses the error signal from the Image Retrieval model to improve caption specificity, which is expected to increase diversity. Training is inspired by GAN: the discriminator is replaced by the Image Retrieval module, and the generator is replaced by the Image Captioning module. But, unlike GAN, where the generator and discriminator learn from each other, in this method the discriminator is pre-trained and doesn't learn from the generator.

2.3.6.5 Describing novel objects

The image captioning methods described so far are limited to predefined vocabularies, that is, these methods are unable to generate descriptions for novel objects, objects that are not associated to any of the concepts represented in the training data (image-sentence pairs). However, humans have the ability to recognize, learn and use novel concepts in various visual understanding tasks. In practical image description applications, it is quite possible to come across situations where there are novel objects which are not included in the vocabulary learned by the model and would be very costly to retrain the system every time a new concept appears. In this section, we describe some attempts to deal with novelties in image captioning.

In order to learn novel visual concepts without retraining the whole system, Mao et al. (2015) propose a method that uses just a few images with sentence descriptions. This method is able to hypothesize the semantic meaning of new words and add them to its dictionary for future use. In order to implement and test this method, the authors have modified their previous model (Mao and Yuille, 2015) to make it more suitable for the novel concept learning task. On the one hand, a transposed weight sharing strategy is applied to reduce the number of parameters, which in turn makes the model less prone to overfitting the new concepts. Second, an LSTM network replaces the simple RNN to avoid the gradient explosion and vanishing problem.

Another approach to make image captioning capable of describing novel objects is the Deep Compositional Captioner (DCC), by Hendricks et al. (2016). The DCC achieves this goal by leveraging large object recognition datasets and external text corpora, and by transferring knowledge between semantically similar concepts. First, a lexical classifier and a language model are trained over image datasets and text corpora, respectively. Then, a multimodal

caption model is trained to integrate the lexical classifier and the language model using linear combinations of affine transformations of image and language features. This model enables the transfer of semantic knowledge between the two modalities, which allows generating descriptions for novel objects and their interactions with other objects.

2.4 Datasets and evaluation

There is a wide range of datasets available for automatic image captioning research. The images in these datasets are linked to textual captions and differ from each other in certain aspects such as in size, the format of the captions, or the procedure used to collect the data. This section presents a brief overview of the common approaches used to collect data, the datasets themselves, and the metrics utilized for model evaluation and benchmarking.

The datasets are summarized in Table 2.3, and examples of images their corresponding captions are given in Fig. 2.5. For a wider survey of datasets in the CV and NLP fields the reader is referred to Ferraro et al. (2015). That survey is not limited to image captioning and provides additional statistics and quality metrics such as perplexity, syntactic complexity, and abstract to concrete word ratios.

Some authors establish a difference between image captions and image descriptions, and in this survey, we have actually used them interchangeably. However, such a distinction is relevant when presenting datasets. Image descriptions verbalize what can be seen in the image, i.e., they refer to the objects, actions, and attributes depicted, etc. Captions, on the other hand, typically provide information that cannot be seen in the image or is not evident, such as personal, cultural, or historical context. Images shared through social networks or photo-sharing websites can be accompanied by descriptions or captions, or a mixture of both types of text. The images in a newspaper or a museum will typically contain cultural or historical texts, i.e., captions not

2.4.1 Datasets

This subsection briefly describes the most popular datasets used in image description tasks, sorted by year of publication.

The **Pascal1K** sentence dataset Rashtchian et al. (2010) consists of 1000 images that were selected from the Pascal 2008 object recognition dataset[2] (Everingham et al., 2010). The

[2]Every year, the Pattern Analysis, Statistical Modeling, and Computational Learning (PASCAL) organization hosts the Visual Object Classes Challenge. In this competition, a new dataset of images with classification and detection information is released, and computer vision researchers compete to create the best classification, detection, and segmentation systems

Table 2.3: Summary of benchmark datasets.

Dataset	Images	Texts	Judgements	Objects
Pascal1K (Rashtchian et al., 2010)	1K	5	No	Partial
IAPTR-TC12 (Escalante et al., 2010)	20K	1-5	No	Segmented
VLT2K (Elliott and Keller, 2013)	2K	3	Partial	Partial
Flickr8K (Rashtchian et al., 2010)	8K	5	Yes	No
Abstract Scenes (Zitnick et al., 2013)	10K	6	No	Complete
Flickr30K (Young et al., 2014)	31K	5	No	No
MSCOCO (Lin et al., 2014)	328K	5	Collected	Partial
Conceptual Captions (Sharma et al., 2018)	3.3M	1	No	No

dataset covers a wide variety of objects and scenery classified into 20 different categories, such as humans, animals, and vehicles. Each image is associated with five one-sentence descriptions generated by humans using the **Amazon Mechanical Turk (AMT)**[3] crowdsourcing service. For the image annotation process, the Turkers (AMT workers) were instructed to focus on the images and describe their contents without considering the context.

The **IAPR-TC12** dataset (Escalante et al., 2010) contains an image collection consisting of 20k images taken from locations around the world and comprising a varying cross-section of still natural images. They were obtained by guides of an independent travel company and undergo a selection process. The images are accompanied by written descriptions expressed in multiple languages (predominantly English and German). Each image is associated with one to five descriptions, sorted by a certain priority pattern.

The **Visual and Linguistic Treebank (VLT)** (Elliott and Keller, 2013) makes use of images from the Pascal 2010 action recognition dataset, augmented with three descriptions of two sentences length per image. These descriptions were collected on the AMT with specific instructions to verbalize the main action depicted in the image and the actors involved (first sentence), as well as the most important background objects (second sentence). There are object annotations available for a subset of 341 images, in the form of polygons around all objects mentioned in the descriptions. Moreover, this subset also includes 3 manually-created Visual Dependency Representations (VDR) per image.

The **Flickr8K dataset** (Rashtchian et al., 2010) comprises approximately 8k images obtained from the *Flickr* photo-sharing website. The images in this dataset were collected through user queries for specific objects and actions using the AMT. The descriptions consist of 5 sentences per image collected from AMT workers using a strategy similar to that of the Pascal1K

[3]Amazon Mechanical Turk (MTurk) is a crowdsourcing marketplace that makes it easier for individuals and businesses to outsource their processes and jobs to a distributed workforce who can perform these tasks virtually. This could include anything from conducting simple data validation and research to more subjective tasks like survey participation, content moderation, and more. See https://www.mturk.com/

dataset.

The **SBU Captioned Photo** dataset (Ordonez et al., 2011) contains around 1M images harvested from Flickr. It was the first attempt to provide a huge dataset for image captioning tasks, but further analysis showed several issues that severely diminished its utility, like captions containing information that cannot be obtained from the image itself (eg. names of people and location), captions describing just a small detail of the image, or captions consisting being a mere commentary about the image.

The **Abstract Scenes** dataset (Zitnick et al., 2013) consists of 10k clip-art images and their descriptions. The images were created through AMT, where workers were asked to place a fixed vocabulary of 80 clip-art objects into a scene of their choosing. The descriptions were then sourced for these worker-created scenes. The authors provided these descriptions in two different forms. While the first group contains a single sentence description for each image, the second group includes two alternative descriptions per image. Each of these two descriptions consists of three simple sentences with each sentence describing a different aspect of the scene. The main advantage of this dataset is it affords the opportunity to explore image description generation without the need for automatic object recognition, thus avoiding the associated noise. A more recent version of this dataset has been created as a part of the visual question-answering (VQA) dataset (Antol et al., 2015). It contains 50k different scene images with more realistic human models and with five single-sentence descriptions.

The **Flickr30K** dataset (Young et al., 2014) is an extended version of the Flickr8K dataset comprising around 31k images. The images in this dataset are mainly about humans involved in everyday activities and events. Each image is described by 5 sentences, like its minor version. Moreover, unlike the 8k version, this dataset is complemented with a *denotation graph* that pairs generalized versions of the image captions with their visual denotations, i.e. the sets of images they describe.

The **MS-COCO** (Common Objects in Context) dataset (Lin et al., 2014), by Microsoft, is one the largest datasets available for research in image description tasks. It includes 328k images of complex everyday scenes containing common objects in their natural context. These images have been annotated with 5 descriptions per image, plus bounding boxes for objects belonging to 80 object categories. This dataset has been widely used for image description, something that is facilitated by the standard evaluation server that is available online since the 2015 COCO Captions Competition [4].

Very recently, Google launched **Conceptual Captions** (Sharma et al., 2018), a huge dataset with an order of magnitude more images than the largest dataset to date (MS-COCO), and represents a wider variety of both images and image caption styles. However, unlike former

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

datasets, which were annotated manually by humans, the 3.3M images of this dataset were collected programmatically, by extracting and filtering images and text annotations from billions of webpages. It consists of a wide variety of images and associated descriptions taken from original Alt-text attributes, automatically transformed to achieve a balance between cleanliness, informativeness, and learnability. The remaining image, caption pairs contain around 16,000 entity types, guaranteed to be well represented in terms of the number of examples.

2.4.1.1 Other datasets

This subsection summarizes other datasets that, in spite of being less popular than the ones already presented, are to a certain extent relevant to the problem at hand.

- The Attribute Discovery dataset (Berg et al., 2010) was collected using automatic dataset construction techniques that exploit the connection between text and images. This dataset consists of 4 broad shopping categories (bags, earrings, ties, and shoes). The data set was collected from *like.com* and includes both images and associated textual descriptions.

- The **MIT-Adobe FiveK** (Bychkovsky et al., 2011) dataset consists of 5K images containing a diverse set of scenes, subjects, and lighting conditions. These images are mainly about people, nature, and man-made objects.

- The **NYU-v2** dataset (Silberman et al., 2012) is a more specialized dataset containing 1.5K RGBD images capturing 464 diverse indoor scenes, with detailed annotations. The particularity of this dataset is its focus on 3D segmentation. It has been augmented with five descriptions per image by Lin et al. (2015) for image description tasks.

- The **SUN Attribute** database (Patterson et al., 2014) provides around 15K images from 707 scene categories. It was derived from the SUN categorical database using crowdsourcing to annotate images with attributes. Previously, crowdsourced human studies were conducted to find a taxonomy of 102 discriminative attributes related to varied aspects, such as materials, surface properties, lighting, affordances, and spatial layout.

- The **FlickrStyle10k** dataset has 10K Flickr images with stylized captions. The training data consists of 7K images. The validation and test data consist of 2,000 and 1,000 images respectively. Each image contains romantic, humorous, and factual captions.

- The **Stock3M** dataset (Wang et al., 2017) has more than 3M images uploaded by the users of a stock image website. Each image is associated with one caption that is provided by the photo uploader. The captions are much shorter than those found on the MSCOCO

1. One jet lands at an airport while another takes off next to it.
2. Two airplanes parked in an airport.
3. Two jets taxi past each other.
4. Two parked jet airplanes facing opposite directions.
5. two passenger planes on a grassy plain

(a) Pascal1K[8]

1. There are several people in chairs and a small child watching one of them play a trumpet
2. A man is playing a trumpet in front of a little boy.
3. People sitting on a sofa with a man playing an instrument for entertainment.

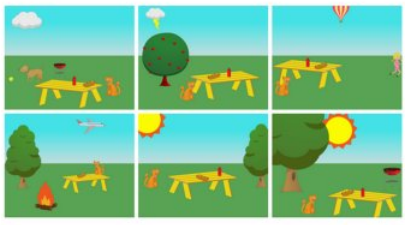
(b) VLT2K[9]

1. A man is snowboarding over a structure on a snowy hill.
2. A snowboarder jumps through the air on a snowy hill.
3. a snowboarder wearing green pants doing a trick on a high bench
4. Someone in yellow pants is on a ramp over the snow.
5. The man is performing a trick on a snowboard high in the air.

(c) Flickr8K[10]

1. a yellow building with white columns in the background
2. two palm trees in front of the house
3. cars are parking in front of the house
4. a woman and a child are walking over the square

(d) IAPR-TC12[11]



1. A cat anxiously sits in the park and stares at an unattended hot dog that someone has left on a yellow bench

(e) Abstract Scenes[12]

1. A blue smart car parked in a parking lot.
2. Some vehicles on a very wet wide city street.
3. Several cars and a motorcycle are on a snow covered street.
4. Many vehicles drive down an icy street.
5. A small smart car driving in the city.

(f) MS COCO[13]

Figure 2.5: Examples of images and their descriptions across various benchmark datasets

dataset but are claimed to be more natural, with a larger vocabulary and image content variety.

- The **VQA** (Visual Question Answering) dataset is intended to support a variation of the image description problem called *visual question answering*: given an image and a natural language question about the image, the task is to provide an accurate natural language answer (Antol et al., 2015). To that aim, the dataset contains 50K abstract scenes using clip-art style graphics. It includes 20 paper-doll human models spanning genders, races, and ages with 8 different expressions, over 100 objects and 31 animals in various poses. The images are completed with five captions per image, as well as diverse questions and answers.

2.4.2 Evaluation metrics

Evaluating Natural Language Generation (NLG) systems is a fundamentally difficult task (Reiter and Belz, 2009) so that it has become a research problem itself. However, we will simply summarize the metrics commonly used for evaluating the quality of image descriptions generated automatically.

2.4.2.1 Human judgement

A common way to assess the quality of automatically generated texts is Subjective evaluation by human judges. Automatically produced text is usually judged in terms of syntactical correctness (grammar compliance) and semantic correctness (relevance of the content). Another dimension of interest is the fluency of the text, especially when a surface realization technique is involved in the generation process. Image descriptions generated automatically can be evaluated using the same techniques that are common in other NLG tasks, an very particularly in machine translation (MT) tasks. In this case, judges get both the image and the generated description for the evaluation task. Subjective human evaluations of machine-generated image descriptions are often performed on the Amazon Mechanical Turk by means of Likert-scale questions. Some examples of these kinds of questions are:

- The description accurately describes the image
- The description is grammatically correct
- The description has no incorrect information
- The description is relevant for this image
- The description is human like

2.4.2.2 Automatic evaluation metrics

Since human evaluation requires large amounts of non-reusable human effort, it is difficult to scale up. Furthermore, human judgment is inherently subjective, which makes it suffer from user variances. To avoid these pitfalls, automated metrics have been created with different properties. Below follows a short review of the most common metrics used for evaluating machine-generated image descriptions.

Classical **information retrieval metrics**, like the *median rank* (mRank) and the *precision and recall at k* (S@k, R@K) have been included quite often to evaluate the descriptions returned by image captioning methods, particularly with for methods that adopt an information retrieval approach, although they are not considered the most appropriate metric for this task.

IBM's **BLEU** (Bilingual Evaluation Understudy) (Papineni et al., 2002) is a family of metrics conceived to substitute human judges in the evaluation of MT tasks, so besides being quick, inexpensive and language independent, it was designed to show good correlation with human judgment. This family of metrics uses a weighted average of variable length phrase matches against the reference translations written by humans to measure their closeness. BLEU starts with a baseline metric from which a number of metrics are derived for different n-gram sizes: BLEU-1 compares the candidate sentence with sentences in unigram, BLEU-2 compares the candidate sentence with sentences in bigram, and so on, until BELU-4, which obtains the best correlation with human judgments. While unigram scores account for the adequacy, higher n-gram scores account for fluency. BLEU is popular because was it was the first metric popularized for MT tasks, long before the research on automatic image generation field took off. Although BLEU shows a reasonable correlation with human judgments it seems to work well only when the generated text is sort (Callison-Burch et al., 2006), and it may even occur that an increase in BLEU score did not correspond to an increase of quality (Lin and Och, 2004).

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) (Lin, 2004) is a set of metrics introduced as an alternative to BLEU to work with larger texts, like text summaries. This metric employs the longest common subsequence between a candidate sentence and a set of reference sentences to measure their similarity at sentence-level. The longest common subsequence between two sentences only requires in-sequence word matches, and the matched words are not necessarily consecutive. Determination of the longest common subsequence is achieved by using dynamic programming. Different versions of ROUGE exist for different tasks: ROUGE-1 and ROUGE-W are appropriate for single document evaluation whereas ROUGE-2 and ROUGE-SU4 have good performance for short summaries.

METEOR (Metric for Evaluation of Translation with Explicit ORdering) (Banerjee and Lavie, 2005) is another metric originally conceived to evaluate MT tasks. It is based on

a generalized concept of unigram matching between the machine- produced translation and human-produced reference translations. Unigrams can be matched based on their surface forms, stemmed forms, and meanings. It first performs generalized unigram matches between a candidate sentence and a human-written reference sentence and then computes a score based on the matching results. The score computation involves precision, recall, and alignments of the matched elements, and in the case of multiple reference sentences, the best score among all the possible matches is taken as the final evaluation score. METEOR achieves better correlation at the sentence or the segment levels. There exists a *Universal* version of this metric supporting many languages, like Russian and Hindi Denkowski and Lavie (2014).

CIDEr (Consensus-based Image Description Evaluation) (Vedantam et al., 2015) adopts a novel paradigm which is based on human consensus and was designed specifically to evaluate image description tasks instead of MT tasks. This metric measures the similarity of an automatically generated sentence to the majority –consensus– of ground-truth sentences (image descriptions) written by humans. It works by encoding the frequency of the n-grams in the candidate sentence that appear in the reference sentences using a Term Frequency-Inverse Document Frequency (TF-IDF) weighting for each n-gram. While most existing datasets have only five captions per image, which the authors consider is not enough to use human consensus, therefore they also created two new datasets with 50 human-generated sentences per image: **Pascal-50S** and **Abstract-50S**, which uses the 1000 images from the Pascal1K, and 500 random images from the Abstract Scene dataset, respectively. A version of CIDEr named **CIDEr-D** is available as a part of the MSCOCO evaluation server to enable systematic evaluation and benchmarking.

SPICE (Semantic Propositional Image Caption Evaluation) (Anderson et al., 2016) is a new evaluation metric for image descriptions based on a graph-based semantic representation of images called scene-graph (Johnson et al., 2015; Schuster et al., 2015). This graph can extract the information of different objects, attributes and their relationships from the image descriptions, and is based on the assumption that semantic propositional content is an important component of human caption evaluation. Extensive evaluations across a range of models and datasets indicate that SPICE captures human judgments over model-generated captions better than other automatic metrics. Furthermore, SPICE can answer questions such as which caption-generator best understands colors? and can caption-generators count?

SPIDEr Liu et al. (2017c) has been proposed recently to overcome some of the problems attributed to existing metrics by combining the properties of two of them: CIDEr and SPICE. This combination is achieved by means of a policy gradient (PG) method that directly optimizes a linear combination of SPICE and CIDEr (hence the name SPIDEr). SPICE score ensures semantic faithfulness of generated sentences, while CIDEr score guarantees that these

descriptions are syntactically fluent.

(Sharif et al., 2018) explore a learning-based approach to combine various metrics and report on their results. The idea behind this metric is the same behind SPIDEr: while some metrics tend to focus on the lexical and syntactical correctness (BLEU, ROUGE, CIDEr), other metrics tend to focus on the semantic relevance (SPICE). Therefore, combining the linguistic properties of various metrics should help achieve a better assessment of the quality of image description. Experimental results show improvements over other metrics in terms of correlation and accuracy on the Pascal-50S dataset.

2.4.3 Summary of methods, datasets, metrics and benchmark results

This subsection summarises all the published research covered in this survey. We begin with an extended overview of the reviewed methods for image captioning that use deep learning techniques. We already presented an overview of methods in Table 2.1, but this time we include additional information to classify and characterize the methods. This information is described below, and it is presented in Table 2.4.

- The **network architecture** used to encode images. All methods use convolutional neural networks (CNN), but they differ in the specific network architecture, like VGG, Inception or ResNet, among others.
- The kind of network used to learn the **language model**, typically some form of LSTM, but also other RNN and other models.
- A classification of the method across multiple dimensions.
  - **Visual Space (VS)** vs **Multimodal Space (MS)**: Most methods model the image in the visual space, but some methods learn a combined visual and language space (see §2.3.2)
  - **Supervised Learning (SL)** vs **Other Deep Learning (ODL)**. Although most methods use supervised learning, in some cases a form of semisupervised or unsupervised learning is employed (eg. generative adversarial networks (GAN) and reinforcement learning (RL)).
  - **Whole Scene (WS)** vs **Dense Captioning (DC)**: Most methods encode and describe images treated as a whole, but some approaches also detect regions in the image and generate descriptions for those regions.
  - **Encode-Decoder (EDA)** vs **Compositional Architecture (CA)**: many approaches adopt and encode-decoder architecture (see §2.3.3), while only a few approaches adopt a compositional architecture (see §2.3.5).

- In addition to the above dimensions, some methods are annotated with extra descriptors to further characterize the image captioning methods:
  - **Attention-Based (AB)** refers to methods that can dynamically focus on various parts of the image while the language descriptions are generated (see §2.3.4).
  - **Semantic Concept-Based (SCB)** methods selectively attend to a set of semantic concepts extracted from the image, so these methods apply a very specific form of attention mechanism.
  - **Novel Object (NOB)** is used to characterize those methods that are able to describe objects that were not learned, that is, objects that were not included in the training dataset (see §2.3.6.5).
  - **Stylized Captions (SC)** refers to the use of styles (eg. romantic, humorous) to generate more attractive captions.

Other acronyms used in this classification are:

- LBL: Log-Bilinear Language Model. It is a neural language model that operates on a feature vector representation of words. Word embeddings are computed using a feedforward neural network. Then, the probability of the next word is computed given the previous words (context) (see (Mnih and Hinton, 2007) for details).
- SC-NLM: Structure-Content Neural Language Model. It is a multimodal neural language model proposed by (Kiros et al., 2014a)
- DTR: Dependency Tree Relations Structure that represents all sentence relationships uniformly as typed dependency relations, that is, as triplets of a relation between pairs of words (see (De Marneffe et al., 2006)).
- MELM: Maximum Entropy Language Model. This model predicts the next word in a sentence based on the principle of maximum entropy (see (Berger et al., 1996) for details).
- HM-LSTM: Hierarchical Multimodal LSTM. Variation of the LSTM architecture proposed by Niu et al. (2017)
- LCNN: Language CNN. Refers to the use of convolutional neural networks to represent language models [5].

[5]The use of CNNs for sentence modeling traces back to Collobert and Weston (2008). The ability of CNNs to create informative latent semantic representations of sentences for downstream tasks Collobert et al. (2011); Kalchbrenner et al. (2014); Kim (2014) led to a huge proliferation of CNN-based networks applied to NLP tasks.

Note: In the classification presented in Table 2.4, we have omitted various methods described in the earlier works section (§2.3.1) that do not fit in the proposed dimensions. These methods are Socher et al. (2014); Lebret et al. (2015a,b); Yagcioglu et al. (2015).

Table 2.4: Classification of deep-learning methods

Publication	Image Encoder	Language Model	Category
Karpathy et al. (2014)	AlexNet	DTR	MS,SL,WS,EDA
Kiros et al. (2014a)	AlexNet	LBL	MS,SL,WS,EDA
Kiros et al. (2014b)	AlexNet, VGG	LSTM,SC-NLM	MS,SL,WS,EDA
Mao et al. (2014)	AlexNet	RNN	MS,SL,WS
Chen and Zitnick (2015)	VGG	RNN	VS,SL,WS,EDA
Donahue et al. (2015)	Caffe	LSTM	VS,SL,WS,EDA
Devlin et al. (2015)	VGG	MELM	MS,SL,SL,WS,EDA
Fang et al. (2015)	AlexNet,VGG	MELM	VS,SL,WS,EDA
Jia et al. (2015)	VGG	LSTM	VS,SL,WS,CA, SCB
Karpathy and Fei-Fei (2015)	VGG	RNN	MS,SL,WS,EDA
Ma et al. (2015)	VGG	LCNN	MS,SL,WS,EDA
Mao and Yuille (2015)	AlexNet,VGG	RNN	MS,SL,WS
Vinyals et al. (2015)	Inception-V1	LSTM	VS,SL,WS,EDA
Xu et al. (2015)	AlexNet	LSTM	VS,SL,WS,EDA,AB
Hendricks et al. (2016)	VGG	LSTM	VS,SL,WS,CA,NOB
Johnson et al. (2016)	VGG	LSTM	VS,SL,DC,EDA
Ma and Han (2016)	AlexNet	LSTM	VS,SL,WS,CA
Mao et al. (2016)	AlexNet,VGG	LSTM	VS,SL,WS,EDA
Mathews et al. (2016)	Inception-V1	LSTM	VS,SL,WS,EDA,SC
Sugano and Bulling (2016)	VGG	LSTM	VS,SL,WS,EDA,AB
Tran et al. (2016)	ResNet	MELM	VS,SL,WS,CA
Wang et al. (2016)	VGG	LSTM	VS,SL,WS,CA
Wu et al. (2016)	VGG	LSTM	VS,SL,WS,EDA,SCB
Yang et al. (2016)	VGG	LSTM	VS,SL,DC,EDA,AB
You et al. (2016)	Inception-V1	RNN	VS,SL,WS,EDA,SCB
Chen et al. (2017a)	VGG,ResNet	LSTM	VS,SL,EDA,AB
Dai et al. (2017)	VGG	LSTM	VS,ODL,WS,EDA
Fu et al. (2017)	VGG	LSTM	VS,SL,WS,EDA,AB
Gan et al. (2017a)	ResNet	LSTM	VS,SL,WS,EDA,SC
Gan et al. (2017b)	ResNet	LSTM	VS,SL,WS,CA,SCB

Table 2.4: Classification of deep-learning methods

Publication	Image Encoder	Language Model	Category
Gu et al. (2017)	VGG	LCNN,LSTM	VS,SL,WS,EDA
Liu et al. (2017a)	VGG	LSTM	VS,SL,WS,EDA,AB
Liu et al. (2017c)	Inception-V3	LSTM	VS,RL,WS,EDA
Liu et al. (2017b)	CNN+LSTM	LSTM	VS,SL,DC,EDA,AB
Lu et al. (2017)	ResNet	LSTM	VS,SL,WS,EDA,AB
Niu et al. (2017)	AlexNet	HM-LSTM	MS,SL,DC,EDA
Park et al. (2017)	ResNet	LSTM	VS,SL,WS,EDA,AB
Pedersoli et al. (2017)	VGG	RNN	VS,SL,WS,EDA,AB
Ren et al. (2017)	VGG	LSTM	VS,ODL,WS,EDA
Rennie et al. (2017)	ResNet	LSTM	VS,ODL,WS,EDA
Shetty et al. (2017)	Inception-V1	LSTM	VS,ODL,WS,EDA
Tavakoliy et al. (2017)	VGG	LSTM	VS,SL,WS,EDA,AB
Venugopalan et al. (2017)	VGG	LSTM	VS,SL,WS,CA,NOB
Wang et al. (2017)	ResNet	LSTM	VS,SL,WS,EDA
Yang et al. (2017)	VGG	LSTM	VS,SL,DC,EDA,AB
Yao et al. (2017b)	Inception-V1	LSTM	VS,SL,WS,EDA,SCB
Yao et al. (2017a)	VGG	LSTM	VS,SL,WS,CA,NOB
Zhang et al. (2017)	Inception-V3	LSTM	VS,RL,WS,EDA
Aneja et al. (2018)	VGG	LCNN	VS,SL,WS,EDA
Jiang et al. (2018)	Inception-V3	LSTM	VS,SL,WS,EDA,AB
Khademi and Schulte (2018)	VGG,ResNet	Grid LSTM	VS,SL,DC,EDA,AB
Li and Chen (2018)	Faster R-CNN	LSTM	VS,RL,DC,EDA,SCB
Wang and Chan (2018)	VGG	LCNN	VS,SL,WS,EDA

Next, we present two tables that enumerate the datasets and evaluation metrics reported in each of the publications. Due to the broad number of publications, we have divided them into two groups, one devoted to those proposals that are not based on neural networks, and another one with those proposals based on neural networks and deep learning.

Table 2.5: Summary of non neural methods, datasets and evaluation metrics.

Publication	Datasets	Evaluation Metrics
Farhadi et al. (2010)	Pascal1K	BLEU
Kulkarni et al. (2011)	Pascal1K	Human, BLEU
Li et al. (2011)	Pascal1K	Human, BLEU
Ordonez et al. (2011)	SBU	
Yang et al. (2011)	Pascal1K	BLEU, ROUGE, METEOR
Gupta et al. (2012)	Pascal1K	Human, BLEU, ROUGE
Kuznetsova et al. (2012)	SBU	Human, BLEU
Mitchell et al. (2012)	Pascal1K	Human
Elliott and Keller (2013)	VLT2K	Human, BLEU
Hodosh and Hockenmaier (2013)	Pascal1K, Flickr8K	Human, BLEU, ROUGE, mRank, R@k
Gong et al. (2014)	SBU, Flickr30K	R@k
Kuznetsova et al. (2014)	SBU	Human, BLEU, METEOR
Patterson et al. (2014)	SBU	BLEU
Socher et al. (2014)	Pascal1K	mRank, R@k
Verma and Jawahar (2014)	IAPR, SBU, Pascal1K	BLEU, ROUGE, R@k
Yatskar et al. (2014)	Own data	Human, BLEU
Elliott and de Vries (2015)	VLT2K, Pascal1K	BLEU, METEOR
Lebret et al. (2015a)*	Flickr30K, COCO	BLEU, R@k
Lebret et al. (2015b)*	COCO	BLEU
Mateos-Ortiz et al. (2015)	Abstract Scenes	Human, BLEU
Ushiku et al. (2015)	Pascal1K, IAPR, SBU, COCO	BLEU

* Note that (Lebret et al., 2015a,b) do not use neural nets as part of their learning method, but they use a pre-trained CNN for the representation of images, so they are in the frontier of both worlds. For this reason, we have included them in both summary tables, the first one, devoted to non-neural methods, as well as the second one, dedicated to neural-based methods.

Table 2.6: Summary of neural methods, datasets and metrics

Publication	Datasets	Evaluation Metrics
Karpathy et al. (2014)	Flickr8K/30K, COCO	BLEU, METEOR, CIDEr
Kiros et al. (2014a)	IAPTR, Attribute Descriptions	BLEU, PPLX
Kiros et al. (2014b)	Flickr8K/30K	R@k, mRank
Mao et al. (2014)	IAPTR, Flickr8K/30K	BLEU, mRank, R@k

Table 2.6: Summary of neural methods, datasets and metrics

Publication	Datasets	Evaluation Metrics
Chen and Zitnick (2015)	Pascal1K, Flickr8K/30K, COCO	BLEU, METEOR, CIDEr, mRank, R@k
Donahue et al. (2015)	Flickr30K, COCO	Human, BLEU, mRank, R@k
Devlin et al. (2015)	COCO	BLEU, METEOR
Fang et al. (2015)	Pascal1K, COCO	Human, BLEU, METEOR, PPLX
Jia et al. (2015)	Flickr8K/30K, COCO	BLEU, METEOR, CIDEr
Karpathy and Fei-Fei (2015)	Flickr8K/30K, COCO	BLEU, METEOR, CIDEr, mRank, R@k
Lebret et al. (2015a)	Flickr30K, COCO	BLEU, R@k
Lebret et al. (2015b)	COCO	BLEU
Ma et al. (2015)	Flickr8K/30K	mRank, R@k
Mao and Yuille (2015)	IAPR, Flickr30K, COCO	BLEU, mRank, R@k
Vinyals et al. (2015)	Pascal1K, SBU, Flickr30K, COCO	Human, BLEU, METEOR, CIDEr, mRank, R@k
Xu et al. (2015)	Flickr8K/30K, COCO	BLEU, METEOR
Yagcioglu et al. (2015)	Flickr8K/30K, COCO	Human, BLEU, METEOR, CIDEr
Hendricks et al. (2016)	COCO, ImageNet	BLEU, METEOR
Johnson et al. (2016)	Visual Genome	METEOR, AP, IoU
Ma and Han (2016)	Flickr8K, UIUC	BLEU, R@k
Mao et al. (2016)	ReferIt	BLEU, METEOR, CIDEr
Mathews et al. (2016)	COCO, SentiCap	BLEU, METEOR, ROUGE, CIDEr
Sugano and Bulling (2016)	COCO	BLEU, METEOR, ROUGE, CIDEr
Tran et al. (2016)	COCO, MIT-Adobe, Instagram	Human
Wang et al. (2016)	Flickr8K	BLEU, METEOR, PPLX
Yang et al. (2016)	COCO	BLEU, METEOR, CIDEr
You et al. (2016)	Flickr30K, COCO	BLEU, METEOR, ROUGE, CIDEr
Chen et al. (2017a)	Flickr8K/30K, COCO	BLEU, METEOR, ROUGE, CIDEr
Dai et al. (2017)	Flickr30K, COCO	BLEU, METEOR
Fu et al. (2017)	Flickr8K/30K, COCO	BLEU, METEOR, ROUGE, CIDEr
Gan et al. (2017a)	FlickrStyle10K	BLEU, METEOR, ROUGE, CIDEr
Gan et al. (2017b)	Flickr30K, COCO	BLEU, METEOR, CIDEr
Gu et al. (2017)	Flickr30K, COCO	BLEU, METEOR, CIDEr, SPICE

Table 2.6: Summary of neural methods, datasets and metrics

Publication	Datasets	Evaluation Metrics
Liu et al. (2017a)	Flickr30K, COCO	BLEU, METEOR
Liu et al. (2017c)	COCO	Human, BLEU, METEOR, ROUGE, CIDEr, SPIDEr
Lu et al. (2017)	Flickr30K, COCO	BLEU, METEOR, CIDEr
Park et al. (2017)	Instagram	BLEU, METEOR, ROUGE, CIDEr
Pedersoli et al. (2017)	COCO	BLEU, METEOR, CIDEr
Ren et al. (2017)	COCO	BLEU, METEOR, ROUGE, CIDEr
Rennie et al. (2017)	COCO	BLEU, METEOR, CIDEr, SPICE
Shetty et al. (2017)	COCO	BLEU, METEOR, SPICE
Tavakoliy et al. (2017)	COCO, Pascal 50S	BLEU, METEOR, ROUGE, CIDEr
Venugopalan et al. (2017)	COCO, ImageNet	METEOR
Wang et al. (2017)	COCO, Stock3M	METEOR, ROUGE, CIDEr, SPICE
Yang et al. (2017)	VisualGenome	METEOR, AP, IoU
Yao et al. (2017b)	COCO	BLEU, METEOR, ROUGE, CIDEr
Yao et al. (2017a)	COCO, ImageNet	METEOR
Zhang et al. (2017)	COCO	BLEU, METEOR, ROUGE, CIDEr
Aneja et al. (2018)	COCO	BLEU, METEOR, ROUGE, CIDEr
Jiang et al. (2018)	COCO	BLEU, METEOR, ROUGE, CIDEr
Khademi and Schulte (2018)	COCO	BLEU, METEOR, ROUGE, CIDEr
Wang and Chan (2018)	COCO	BLEU, METEOR, ROUGE, CIDEr

Finally, we conclude this section, and the review, with a comparison of methods across the most popular benchmark datasets and evaluation metrics, that is Flickr30K and MSCOCO datasets, and the BLUE, ROUGE, METEOR and CIDEr metrics. This is not an exhaustive review, but we have included the most cited ones that include comparable results.

Table 2.7 compares results by various methods on the Flickr30K dataset. In this table Bn, M, and C refer, respectively, to BLEU-n, METEOR and CIDEr scores.

Table 2.7: Comparison of methods on the Flickr30K dataset

Method	B1	B2	B3	B4	M	C
Kiros et al. (2014b)	60	38	25.4	17.1	16.9	
Chen and Zitnick (2015)				12.6	16.4	
Donahue et al. (2015)	58.7	39.1	25.1	16.5		
Jia et al. (2015)	64.6	44.6	30.5	20.6	17.9	
Karpathy and Fei-Fei (2015)	57.3	36.9	24	15.7		
Mao and Yuille (2015)	60	41	28	19		
Vinyals et al. (2015)	66.3	42.3	27.7			
Xu et al. (2015) Soft	66.7	43.4	28.8	19.1	18.49	
Xu et al. (2015) Hard	66.9	43.9	29.6	19	18.5	
Oruganti et al. (2016)	58.9	40	26.6	17.7	17.8	
Wu et al. (2016)	73	55	40	28		
You et al. (2016)	64.7	46	32.4	23	18.9	
Chen et al. (2017a)	66.2	46.8	32.5	22.3	19.5	44.7
Gan et al. (2017b)	74.7	55.2	40.3	28.8		
Gu et al. (2017)	73.8	56.3	41.9	30.7	20.6	
Fu et al. (2017)	64.9	46.2	32.4	22.4	19.4	
Lu et al. (2017)	67.7	49.4	35.4	25.1	20.4	53.1
Li and Chen (2018)	**75.5**	**57.1**	**42.9**	**31.7**	**22.9**	**71.5**
Wang and Chan (2018)	57.7	40.1	27.6	19	18.4	35.2
Wang and Chan (2018) Hier	60.7	42.5	29.2	19.9	19.1	39.5

Table 2.8 compares results on the MSCOCO dataset. In this table Bn, M, R and C refer, respectively, to BLEU-n, METEOR, ROUGE-L and CIDEr scores. Scores in bold and italics indicate the best and the second best results in a given metric, respectively. Methods marked with * are ensemble methods.

Table 2.8: Comparison of methods on the MSCOCO dataset

Method	B1	B2	B3	B4	M	R	C
Kiros et al. (2014b)	70.8	48.9	34.4	24.3	20.03		
Chen and Zitnick (2015)				19	20.4		
Donahue et al. (2015)	66.9	48.9	34.9	24.9			
Fang et al. (2015)				25.7	23.6		
Jia et al. (2015)	67	49.1	35.8	26.4	22.7		81.3
Karpathy and Fei-Fei (2015)	62.5	45	32.1	23	19.5		66
Mao and Yuille (2015)	66.8	48.8	34.2	23.9	22.1	48.9	72.9
Vinyals et al. (2015)	66.6	46.1	32.9	24.6			85.5
Xu et al. (2015) Soft	70.7	49.2	34.4	24.3	23.9		
Xu et al. (2015) Hard	71.8	50.4	35.7	25	23		
Oruganti et al. (2016)	70.2	52.8	38.3	27.6	22.5		
Wu et al. (2016)	74	56	42	31	26		94
Yang et al. (2016)	73.8			29	23.7		88.6
You et al. (2016)	70.9	53.7	40.2	30.4	24.3		
Chen et al. (2017a)	71.9	54.8	41.1	31.1	25		95.2
Chen et al. (2017b)	76.1	59.6	45	33.7	25.7	55	102.9
Fu et al. (2017)	72.4	55.5	41.8	31.3	24.8	53.2	95.5
Gan et al. (2017b)	74.1	57.8	44.4	34.1	26.1		104.1
Gu et al. (2017)	72.3	55.3	41.3	30.6		25.2	98.9
Liu et al. (2017b)	73.1	56.7	42.9	32.3	25.8		105.8
Lu et al. (2017)	74.2	58	43.9	33.2	26.6		108.5
Mun et al. (2017)	74.9	58.1	43.7	32.6	25.7		102.4
Ren et al. (2017)	71.3	53.9	40.3	30.4	25.1	52.5	93.7
Rennie et al. (2017)				34.2	26.7	55.7	114
Rennie et al. (2017)*				35.4	27.1	56.6	117.5
Shetty et al. (2017)					27.2		
Wang et al. (2017)	74.2	57.7	44	33.6	26.8	55.2	107.3
Yao et al. (2017b)	73.4	56.7	43	32.6	25.4		100.2
Zhang et al. (2017)				34.4	26.7	55.8	116.2
Zhou et al. (2017)	71.6	54.5	40.5	30.1	24.7		97
Anderson et al. (2018b)	77.2			36.2	27	56.4	113.5
Anderson et al. (2018b) opt	79.8			36.3	27.7	56.9	120.1
Aneja et al. (2018)	71.1	53.8	39.4	28.7	24.4	52.2	91.2
Gu et al. (2018)	78.6	62.5	47.9	36.1	27.4	56.9	120.4
Jiang et al. (2018)	75.1	59.2	45.8	35.3	26.7	55.5	107.8
Khademi and Schulte (2018)	76.2	60.1	45.1	35	27		
Wang and Chan (2018)	68.8	51.3	37	26.5	23.4	50.7	83.9
Wang and Chan (2018) Hier	68.5	51.1	36.9	26.7	23.4	51	84.4

Chapter 3

Scope

This chapter constitutes a preparatory step before describing the model in the next chapter. On the one hand, this chapter lays out the scope of the project and justifies the main design decisions taken for developing a system. In the other hand, it presents the architectural framework adopted in our system and provides some theoretical background.

3.1 Scope

Before diving into the design of the system developed for this project, it seems convenient to delimit the scope of it as well as justify the design decisions taken. In particular, we have to decide on three main separated aspects:

1. To begin with, we should decide on the kind of architecture to design our system, which can further be divided into various finer details, like deciding the concrete models to use for each component of the architecture, whether to use transfer learning, and whether to use ensemble methods, among others.

2. Another decision of high importance if the dataset to use in order to train and evaluate our system. As a secondary decision, we should also decide which metrics to use for evaluating it.

3. Finally, we should decide on the specific experiments, which in turn will involve deciding on various hyper-parameters and configuration options that may be available in our model.

Deciding on those three aspects should be conditioned by a combination of factors, from research considerations to hardware requirements and time constraints. This chapter will briefly discuss these aspects, and then we will conclude with the final decisions that define the scope of the system to be developed as part of this project.

3.1.1 Research considerations

As we have seen in our review of the state of the art (Chapter 2), image captioning is a thrilling field of research that has seen tremendous advances in recent years, especially since neural networks and deep learning were applied to it. From the first neural model proposed by Kiros et al. (2014b), a myriad of different neural models have been proposed, usually combining a visual encoder and a text decoder. Although there are some exceptions, the majority of the models proposed so far use some form of Convolutional Neural Network (CNN) for the encoder, and some form of Recurrent Neural Network (RNN) for the decoder. Some models also use CNN for the decoder, and a few recent models include Generative Adversarial Networks (GAN) to achieve more varied and expressive captions. Most of the proposed models use some form of supervised learning, typically back-propagation, but there is a growing number of models that include reinforcement learning, and there have been some recent attempts to use also semi-supervised learning, and even unsupervised learning.

Summing up, there are many options to choose from in order to develop our own image captioning system. If we look at the results obtained from these models in benchmark datasets, we can see various patterns and trends.

- First, end-to-end approaches based on the encoder-decoder framework prevail over compositional architectures.
- Second, attention mechanisms have flourished and are a key component included in the vast majority of models, although they can adopt different forms.
- Third, the addition of reinforcement learning is gaining a lot of momentum as a means to improve the quality of generated captions when considering language quality metrics such as CIDEr.
- Fourth, models tend to increase in complexity by stacking more layers and including additional methods and forms of learning. Another aspect of this trend is the use of ensemble methods that combine various models together to produce the final result.

Besides the aforementioned trends and patterns, the study of the published research relative to image captioning reveals a high degree of alignment between this research and the more general research sequence-to-sequence (*seq2seq*) learning and sequence translation problems. Therefore, it would be interesting to study current trends in *seq2seq* research to anticipate the next advances in image captioning.

Indeed, neural image captioning systems adopted the encoder-decoder architecture used in sequence translation, with the particularity of using CNN for the encoder, but the decoder is

an RNN as those used in seq2seq problems. Next, ResNet and Attention make its appearance and rapidly gained popularity, both in seq2seq learning and image captioning.

Attention-based networks are increasingly used by the big players in the IT world, including Google and Facebook. The main reason for this shift from simple RNN to attention-based architectures is because the former requires more resources to train and run than the latter. Therefore, we deem attention as a key ingredient to include in our own system.

3.1.2 Hardware requirements and time constraints

Although for some projects hardware and time may appear as two separate aspects to factor in, they are closely related when we are dealing with deep learning. That is because due to the vast size of some datasets, training some systems may require either fabulous hardware resources or incredibly long periods, and even a combination of both.

For this project we have to deal with a combination of both little time and limited hardware resources, so we have to limit the scope of our project to something manageable.

At the beginning of this project, we had at our disposal a computer equipped with an NVIDIA GTX 1070 GPU supporting CUDA, which is a must-have for any deep learning project beyond toy examples.

After doing some preliminary experiments we realized that with the hardware at hand, it was going to be unfeasible to work with state of the art benchmark datasets such as MS COCO (not to mention the Conceptual Captions dataset, recently released by Google). Furthermore, we got serious problems when using the computer in interactive mode whilst training a deep learning model.

One option was to conduct our experiments on a smaller, dataset such as the Flickr8K or perhaps the Flickr30K dataset, but there are quite outdated nowadays. Therefore, in order to work with larger datasets, I decided to invest in a new, more powerful GPU, and so I bought an NVIDIA GTX 20180 Ti. I got this GPU installed on the same computer, alongside the GTX 1070. As a result of the new configuration, the training times reduced considerably, and it became feasible to use the computer whilst training a model on the COCO dataset.

The most relevant specifications of the final hardware used to conduct the experiments as as follows:

- CPU: Intel Core i7-4790K (4 cores, 8 threads clocked @ 4 GHz)
- RAM: 16GB DIMM DDR3 2400 (clocked @ 1333 MHz)
- Storage: 2 x SSD Samsung 850 Evo 250GB
- GPU-0: NVIDIA GTX 1070, 6GB RAM

- GPU-1: NVIDIA RTX 2080 Ti, 11GB RAM

Note the two GPUs included in the system. The GTX 1070 is used as the rendering unit, for interactive tasks, while the RTX 2080 Ti is used as the main CUDA platform to carry on the data-intensive tasks required to train and evaluate our model on big datasets. With that hardware, we deem it possible to train models one the full COCO dataset in less than a week, depending on the concrete architecture of the net and different hyperparameters on the net, as well as some restrictions imposed on the training data, such as the size of the vocabulary, or the maximum caption length allowed.

3.1.3 Software requirements

With respect to the software requirements, from the very beginning of the project we two general requirements were clear:

- Using a language we are familiar with
- Using a popular framework with a big community

The combination of the two requisites above led us to choose Python as the development language, and Keras with Tensorflow backend as the computational framework.

However, in the end, we decided to take a little risk by using an alpha version of the **Keras-Tensorflow** API stack that has been released recently as part of the new **Tensorflow 2.0.0-alpha**.

3.1.4 Summing up

The study of published research resulted in the selection of an **encoder-decoder architecture with a soft attention mechanism** based on the model proposed by Xu et al. (2015).

However, there are so many options to choose from for the different components of our system, that we should limit those options to a few options.

- **Encoder**: we aim at trying at least two different encoders, like the Inception-V3 network, and the NASNet.
- **Decoder**: for this component, the idea is to try both GRU and LSTM units.
- **Attention** mechanism: we aim at trying the soft attention mechanism, which is easier to implement

Finally, if there is enough time we may try different hyper-parameters, such as the number of hidden units or units in the embedding layer. Other options to play with are the size of the vocabulary, but that is difficult to estimate at this stage of the project.

3.2 The Encoder-Decoder architecture

3.2.1 From feature engineering to feature learning

Machine learning algorithms take features as inputs and produce some output. How we represent features makes a huge difference in the performance of the learning algorithms. Traditionally, approaches to machine learning require a strong **feature engineering** approach, meaning that the designer has to carefully choose features and their representation before feeding them into the algorithm. When facing complex problems like computer vision, this approach came up with sophisticated representations, such as HOG (Histogram of Oriented Gradients) for representing image features.

Due to such representations, there is a difference to be made between the *raw inputs*, and the features which are input to the model. For example, in face recognition, the pixels of a picture are the raw input, while the HOG features of the image can be the actual input to the model.

Someone came up with the idea that we can use an algorithm to learn the feature representation itself, aptly called **feature learning**. Deep learning uses a neural network model to achieve this task. In point of fact, feature learning is accomplished by the first layers of a neural network, which map raw inputs to efficient feature representations. The last layers (typically fully connected layers) mix and match and combine these features to produce an output.

3.2.2 The encoder-decoder architecture

The **encoder-decoder architecture** explicitly aims to leverage this ability of neural networks to learn efficient representations. It is a neural network design pattern that consists of two main components, the *encoder* and the *decoder*. The **encoder** maps raw inputs to feature representations, and these representations are then passed to the **decoder**, which has to produce an output. This is commonly referred to as the encoder-decoder framework, and its general architecture is shown in Fig. 3.1

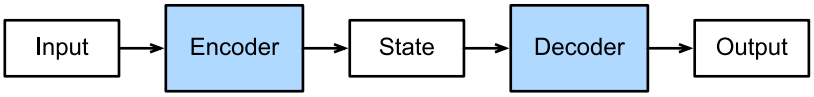


Figure 3.1: The encoder-decoder architecture.

Theoretically, encoder and decoder parts can be used independently of each other. For instance, an encoder RNN can be used to encode the features of an incoming email as a *features vector*, which is then used as input to a Fully-Connected Layer (FCL) to predict whether the email is spam or not. However, neural encoder and decoders are often used in a tightly coupled

manner, meaning that the system is trained as a whole. This approach, known as **end-to-end**, is been increasingly used due to a combination of good performance and less engineering effort.

The encoder-decoder architecture is enormously popular in the fields of computer vision and natural language processing and may adopt very different forms. In some cases, the same type of network is used for both the encoder and the encoder.

For example, Fig. 3.2 depicts an example of a pure convolutional encoder-decoder architecture (there is no fully connected layer). This model is used to perform semantic segmentation of an image. The left half of the network (the encoder) maps raw image pixels to a rich representation consisting of feature vectors. The right half of the network (the decoder) takes these features and upsamples them to produce a sparse feature map which is fed to a soft-max for pixel-wise classification.

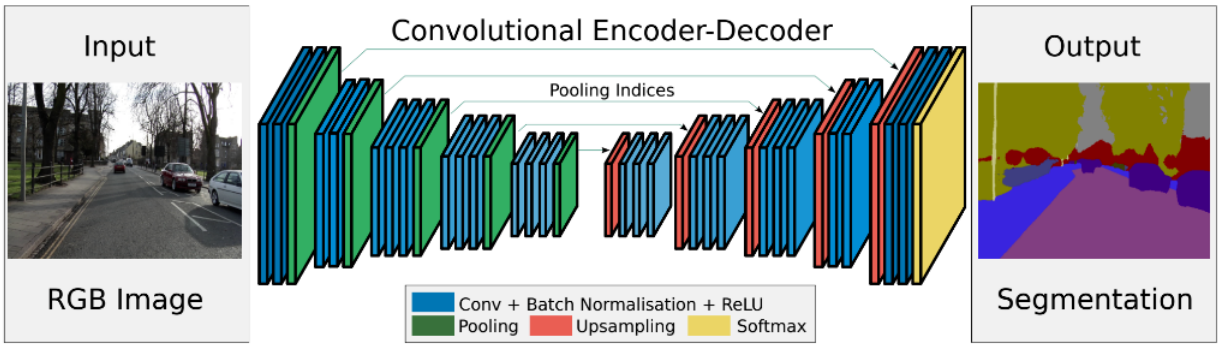


Figure 3.2: Convolutional encoder-decoder for image segmentation in the SeqNet architecture (Badrinarayanan et al., 2017)

Encoder-decoder networks based on RNN are very common in problems requiring *sequence-to-sequence (seq2seq)* modeling, such as translation, summarising and question-answering. For example, Fig. 3.3 shows an example of a model used to generate automatic responses to incoming emails. The left half of the network encodes the email into a feature vector, and the right half of the network decodes the feature vector to produce word predictions.

However, in general, an encoder-decoder network may hybridize different types of network. A common pattern is the CNN-RNN architecture, which uses a CNN as the encoder and an RNN as the decoder. This is a class of models that is both spatially and temporally deep and has the flexibility to be applied to a variety of multimodal tasks involving visual inputs and sequential outputs. Some examples of tasks where this architecture is used include:

- Visual time series forecasting: Predicting the evolution of series, such as stock prices or energy load.
- Activity recognition: Generating a textual description of an activity demonstrated in a sequence of images.

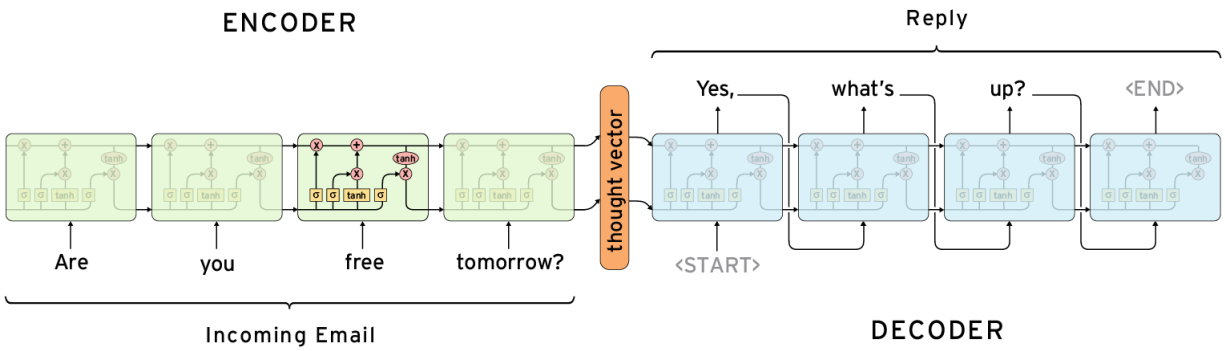


Figure 3.3: Recurrent encoder-decoder for automated email answering. The model is given input a sentence and produces a response in the same language. Source: Google AI Blog

- Image captioning: Generating a textual description of a single image.
- Video description: Generating a textual description of a sequence of images.

This architecture was originally referred to as a Long-term Recurrent Convolutional Network (LRCN) (Donahue et al., 2015), since typically it uses LSTM for the recurrent part. For example, Fig. 3.4 depicts the Neural Image Captioner developed by Google (Vinyals et al., 2015), which uses a Batch Normalization version of their Inception CNN for the encoder and an LSTM network for the decoder. The model works as follows: first, the CNN process an image and extracts visual features, which are then passed as input to the LSTM. The LSTM takes a word, the context from previous time steps, and defines a probability distribution over the next word in the sentence. The LSTM is conditioned on the image information at the first time step. The generative process is started with a special "<start>" token and ends when a special "<end>" token is generated.

A key of this architecture is the use of a CNN that is pre-trained on a challenging image classification task, which is re-purposed as a feature extractor for the caption generating problem (this is called *transfer learning*). Typically, a CNN trained on the ImageNet dataset is used for this purpose.

For the decoder, most systems proposed so far use LSTM units, since they are more powerful than other types of recurrent units, although lately, GRU units have also emerged as a viable alternative with similar capabilities bu reduced cost.

3.2.3 Sequence to Sequence modelling

Next, we delve into the case where both the encoder and decoder are recurrent networks and is commonly known as sequence-to-sequence modeling.

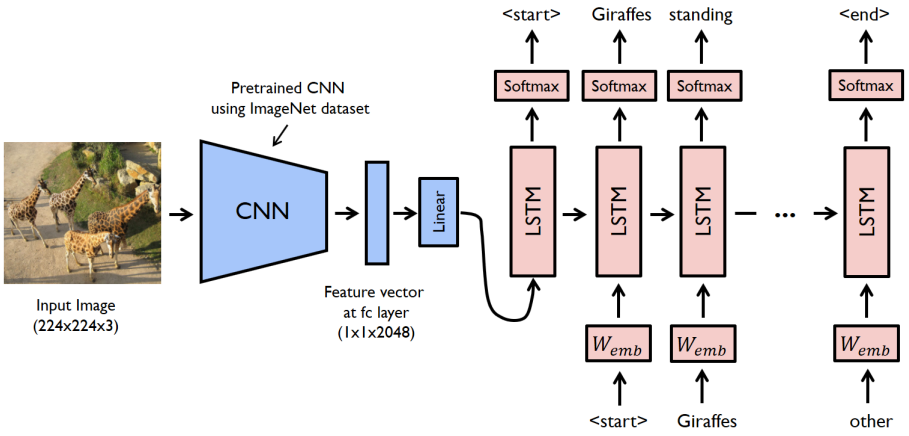


Figure 3.4: Encoder-decoder combining CNN and LSTM for image captioning. Source: Yunjey Choi's implementation of the *Show and Tell* model (Vinyals et al., 2015).

Seq2seq architecture Sequence to sequence modeling, *seq2seq* for short, is based on the encoder-decoder architecture to generate a sequence output for a sequence input. This model was born in the field of language modeling (Sutskever et al., 2014). Broadly speaking, it aims to transform an input sequence (source) to a new one (target) and both sequences can be of arbitrary lengths. Examples of transformation tasks include machine translation between multiple languages in either text or audio, question-answer dialog generation, or even parsing sentences into grammar trees.

The seq2seq model normally has an encoder-decoder architecture, composed of:

- An encoder processes the input sequence and compresses the information into a context vector (also known as sentence embedding or "thought" vector) of a fixed length. This representation is expected to be a good summary of the meaning of the whole source sequence.

- A decoder is initialized with the context vector to emit the transformed output. The early work only used the last state of the encoder network as the decoder initial state.

Both the encoder and decoder are recurrent neural networks, i.e. using LSTM or GRU units

Fig. 3.3 shows an example of a query answering application that uses the seq2seq model. Fig. 3.5 shows another example of seq2seq modeling applied to translation.

The layers in the encoder and the decoder are illustrated in Fig. 3.6.

Seq2seq encoder In the encoder, we use the word embedding layer to obtain a feature index from the word index of the input language and then input it into a multi-level recurrent layer,

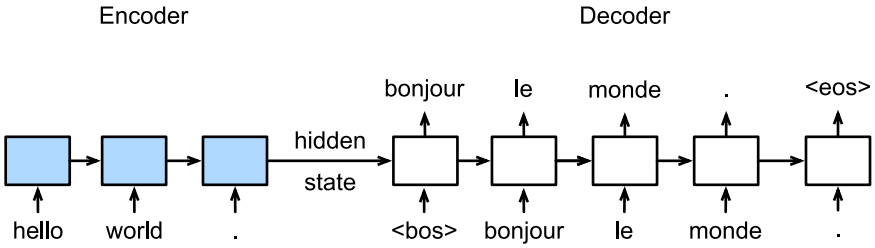


Figure 3.5: The sequence to sequence model architecture.

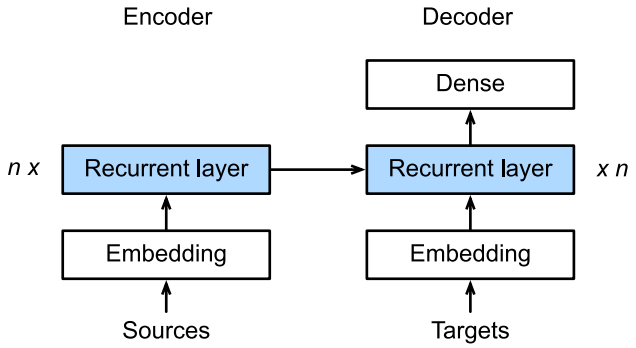


Figure 3.6: Layers in seq2seq model.

typically made of gated recurrent units such as LSTM or GRU. The input for the encoder is a batch of sequences, which is 2-D tensor with shape (batch size, sequence length). The output consists of both the outputs of the recurrent units and the hidden states (and memory cells of the last time step if using LSTM).

The output shape returned by the encoder after performing forward calculation on the input is (number of time steps, batch size, number of hidden units). The shape of the multi-layer hidden state of the gated recurrent unit in the final time step is (number of hidden layers, batch size, number of hidden units). If GRU is used the *state* contains only one element, which is the hidden state. If LSTM is used, the *state* list will also contain another element, which is the memory cell.

Seq2seq decoder We directly use the hidden state of the encoder in the final time step as the initial hidden state of the decoder. This requires that the encoder and decoder RNNs have the same numbers of layers and hidden units.

The forward calculation of the decoder is similar to the encoder's calculation. The only difference is the addition of a dense layer with the hidden size to be the vocabulary size to output the predicted confidence score for each word.

Seq2seq training For each time step, the decoder outputs a vocabulary size confident score vector to predict words. Similar to language modeling, we can apply *softmax* to obtain the

probabilities and then use cross-entropy loss to calculate the loss.

But note that we padded the target sentences to make them have the same length. We would not like to compute the loss on the padding symbols. To solve that, a masked version of the loss function is used.

During training, if the target sequence has length n, we feed the first $n-1$ tokens into the decoder as inputs, and the last $n-1$ tokens are used as ground truth label. This is called *Teacher Forcing*, and is depicted in Fig. 3.5.

Seq2sea prediction To predict a new sequence, the generation process is started by feeding the same "<bos>" token to the decoder at time step 0, exactly as is done in the training stage. But the input token for a later time step is the predicted token from the previous time step, instead of the ground truth token used during training with the Teacher forcing algorithm. This process is shown in Fig. 3.7.

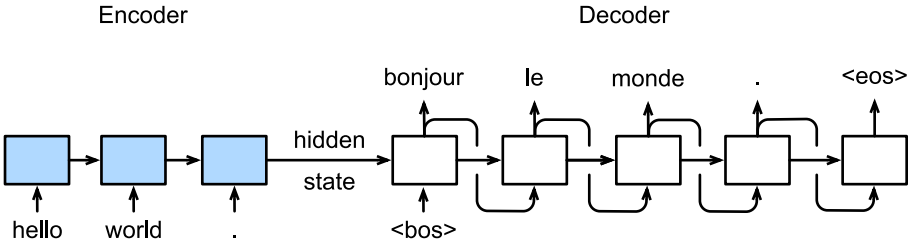


Figure 3.7: Sequence to sequence model predicting with greedy search.

Various approaches exist to predict the next token. For example, in Greedy search, the token with the highest score at each time step is chosen and used to predict the next token. More details on this will be provided in the next chapter when describing our own model details.

3.2.4 Attention mechanisms

A recent trend in Deep Learning is the use of attention mechanisms, typically added to the decoder in an encoder-decoder architecture.

Attention is, to some extent, motivated by how we pay visual attention to different regions of an image or correlate words in one sentence. Human visual attention is well-studied and while there exist different models, all of them essentially come down to being able to focus on a certain region of an image with "high resolution" while perceiving the surrounding image in "low resolution", and then adjusting the focal point over time. Given a small patch of an image, pixels in the rest provide clues about what should be displayed there. For example, if we see an eye besides a nose, we expect to see another eye at the opposite side of the nose.

Similarly, we can explain the relationship between words in one sentence or a close context. When we see "eating", we expect to encounter a food word very soon.

Attention in Neural Networks has a long history, particularly in image recognition, but only recently have attention mechanisms made their way into recurrent neural networks architectures that are typically used in NLP, and increasingly also in computer vision tasks.

3.2.4.1 Attention in Neural Machine Translation

In the original seq2seq model (§3.2.3), the source sequence is encoded into a fixed length context vector that is then passed to the decoder to generate a target sequence. A critical and apparent disadvantage of the fixed-length context vector design of the initial seq2seq model lies in its incapability of remembering long sentences. Often it has forgotten the first part once it completes processing the whole input. The attention mechanism was proposed by Bahdanau et al. (2015) to resolve this weakness.

Besides, often, a token in the target sequence may closely relate to some tokens in the source sequence instead of the whole source sequence. For example, when translating "Hello world." to "Bonjour le monde.", "Bonjour" maps to "Hello" and "monde" maps to "world". In the seq2seq model, the decoder may implicitly select the corresponding information from the state passed by the decoder. The attention mechanism, however, makes this selection explicit.

Below we describe the attention mechanism as originally proposed by Bahdanau et al., and then we will present a more general model of attention which can be applied to multimodal information tasks such as image captioning, which combines both visual and textual information.

The attention mechanism was born to help memorize long source sentences in neural machine translation (NMT). Rather than building a single context vector out of the encoder's last hidden state, the secret sauce invented by attention is to create shortcuts between the context vector and the entire source input. The weights of these shortcut connections are customizable for each output element.

While the context vector has access to the entire input sequence, we don't need to worry about forgetting. The alignment between the source and target is learned and controlled by the context vector. Essentially the context vector consumes three pieces of information: encoder hidden states, decoder hidden states, and alignment between source and target.

More formally: say, we have a source sequence $\mathbf{x}$ of length n and try to output a target sequence $\mathbf{y}$ of length m:

$$\mathbf{x} = [x_1, x_2, \ldots, x_n]$$

$$\mathbf{y} = [y_1, y_2, \ldots, y_m]$$

(Variables in bold indicate that they are vectors; same for everything else in this post.)

The encoder is a bidirectional RNN (or any other recurrent network setting of your choice)

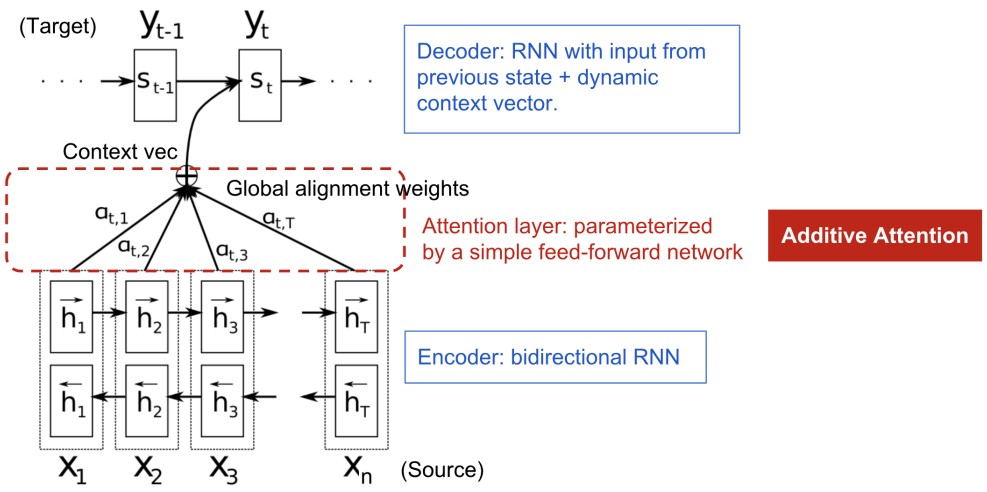


Figure 3.8: The encoder-decoder model with additive attention mechanism in (Bahdanau et al., 2015)

with a forward hidden state $\overrightarrow{\boldsymbol{h}}_i$ and a backward one $\overleftarrow{\boldsymbol{h}}_i$. A simple concatenation of two represents the encoder state. The motivation is to include both the preceding and following words in the annotation of one word.

$$\boldsymbol{h}_i = [\overrightarrow{\boldsymbol{h}}_i^\top; \overleftarrow{\boldsymbol{h}}_i^\top]^\top, i = 1, \ldots, n$$

The decoder network has hidden state $\boldsymbol{s}_t = f(\boldsymbol{s}_{t-1}, y_{t-1}, \mathbf{c}_t)$ for the output word at position $t, t = 1, \ldots, m$, where the context vector $\mathbf{c}_t$ is a sum of hidden states of the input sequence, weighted by alignment scores:

$$\begin{aligned}
\mathbf{c}_t &= \sum_{i=1}^{n} \alpha_{t,i} \boldsymbol{h}_i && \text{; Context vector for output } y_t \\
\alpha_{t,i} &= \text{align}(y_t, x_i) && \text{; How well two words } y_t \text{ and } x_i \text{ are aligned.} \\
&= \frac{\exp(\text{score}(\boldsymbol{s}_{t-1}, \boldsymbol{h}_i))}{\sum_{i'=1}^{n} \exp(\text{score}(\boldsymbol{s}_{t-1}, \boldsymbol{h}_{i'}))} && \text{; Softmax of some predefined alignment score..}
\end{aligned}$$

The alignment model assigns a score $\alpha_{t,i}$ to the pair of input at position i and output at position $t, (y_t, x_i)$, based on how well they match. The set of $\{\alpha_{t,i}\}$ are weights defining how much of each source hidden state should be considered for each output. In (Bahdanau et al., 2015), the alignment score α is parameterized by a feed-forward network with a single hidden layer and this network is jointly trained with other parts of the model. The score function is therefore in the following form, given that *tanh* is used as the non-linear activation function:

$$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\boldsymbol{s}_t; \boldsymbol{h}_i])$$

where both $\mathbf{v}_a$ and $\mathbf{W}_a$ are weight matrices to be learned in the alignment model.

The matrix of alignment scores is a nice byproduct to explicitly show the correlation between source and target words.

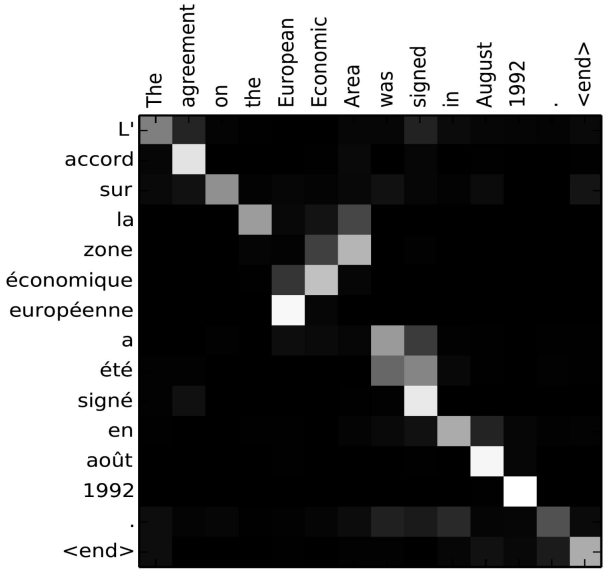


Figure 3.9: Alignment matrix of an French sentence and its English translation. Source: Fig 3 in (Bahdanau et al., 2015))

3.2.4.2 Different attention mechanisms

With the help of the attention, the dependencies between source and target sequences are not restricted by the in-between distance anymore. Given the big improvement by attention in machine translation, it soon got extended into the computer vision field (Xu et al., 2015) and people started exploring various other forms of attention mechanisms (Luong et al., 2015; Britz et al., 2017; Vaswani et al., 2017).

Table 3.1 enumerates popular attention mechanisms and their corresponding alignment score functions, where $\mathbf{W}_a$ is a trainable weight matrix in the attention layer.

Notes *Additive* attention is also referred to as *additive* or *concat* by other authors. *Location-based* attention simplifies the softmax alignment to only depend on the target position. The *scaled dot attention* is very similar to the dot-product attention except for the scaling factor $1/\sqrt{n}$; where n is the dimension of the source hidden state.

Table 3.1: summary table of several popular attention mechanisms and corresponding alignment score functions

Name	Alignment score function	Citation
Content-based att.	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \text{cosine}[\boldsymbol{s}_t, \boldsymbol{h}_i]$	(Graves et al., 2014)
Additive	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\boldsymbol{s}_t; \boldsymbol{h}_i])$	(Bahdanau et al., 2015)
Location-Base	$\alpha_{t,i} = \text{softmax}(\mathbf{W}_a \boldsymbol{s}_t)$	(Luong et al., 2015)
General	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \mathbf{W}_a \boldsymbol{h}_i$	(Luong et al., 2015)
Dot-Product	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \boldsymbol{h}_i$	(Luong et al., 2015)
Scaled Dot-Product	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \frac{\boldsymbol{s}_t^\top \boldsymbol{h}_i}{\sqrt{n}}$	(Vaswani et al., 2017)

Besides the scoring functions, there are different ways to classify attention mechanisms, for example:

- Self-Attention (Cheng et al., 2016): Relating different positions of the same input sequence. Theoretically, the self-attention mechanism can adopt any score functions above, but just replace the target sequence with the same input sequence.
- Global/Soft Xu et al. (2015): Attending to the entire input state space.
- Local/Hard (Xu et al., 2015; Luong et al., 2015): Attending to the part of input state space; i.e. a patch of the input image.

3.2.4.3 Self-Attention

Self-attention, also known as intra-attention, is an attention mechanism relating different positions of a single sequence in order to compute a representation of the same sequence. It has been shown to be very useful in machine reading, abstractive summarization, or image description generation.

The LSTM network by Cheng et al. (2016) uses self-attention to do machine reading. Fig. 3.10 illustrates this mechanism with an example. See how self-attention mechanism enables the model to learn the correlation between the current words and the previous part of the sentence.

Self-attention can also be applied to image captioning. For example, in the "Show, attend and tell" paper by Xu et al. (2015), an input image is first encoded by a convolutional neural network to generate a feature map, and then a recurrent network with self-attention over the image consumes the attention-weighted feature maps to generate the descriptive words one by one. The visualization of the attention weights clearly demonstrates which regions of the image the model pays attention to so as to output a certain word.

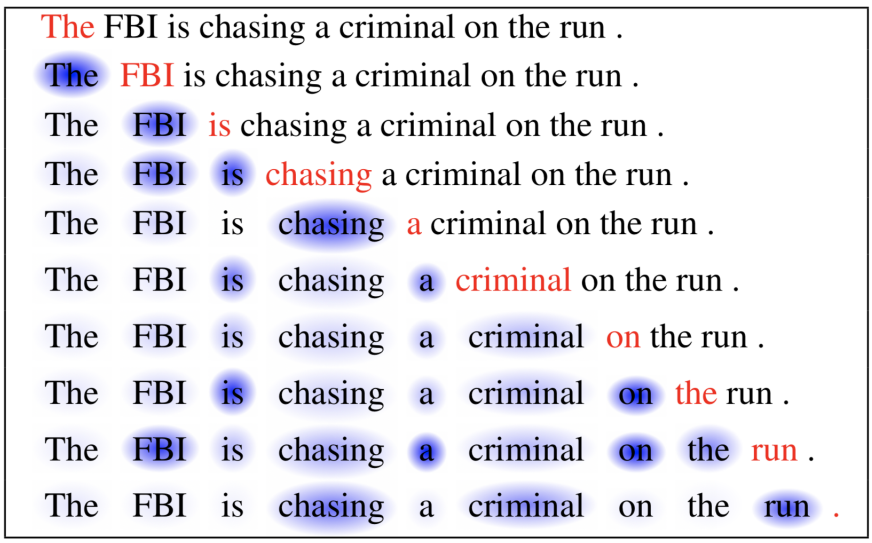


Figure 3.10: The current word is in red and the size of the blue shade indicates the activation level. Source: (Cheng et al., 2016).

Figure 3.11: Caption generation example:A woman is throwing a frisbee in a park. Source: (Xu et al., 2015).

3.2.4.4 Soft vs Hard Attention

The *soft* vs *hard* attention is another way to categorize how attention is defined. The original idea was proposed in the "Show, attend and tell" paper (Xu et al., 2015), based on whether the attention has access to the entire image or only a patch:

- Soft Attention: the alignment weights are learned and placed "softly" over all patches in the source image; which corresponds to the additive attention model by Bahdanau et al. (2015).
  - Pro: the model is smooth and differentiable.
  - Con: expensive when the source input is large.
- Hard Attention: only selects one patch of the image to attend to at a time.
  - Pro: less calculation at the inference time.
  - Con: the model is non-differentiable and requires more complicated techniques such as variance reduction or reinforcement learning to train (Luong et al., 2015).

Fig. 3.12 illustrates the soft vs hard effect attention with a visual example from the original paper.

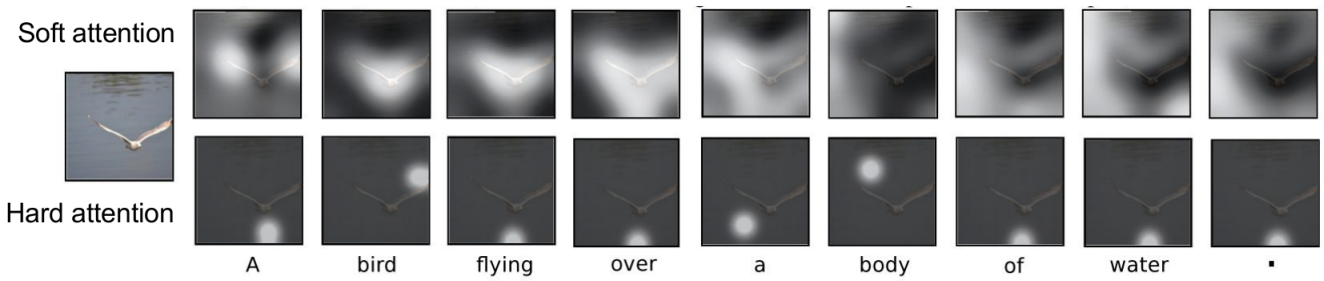


Figure 3.12: Comparative of soft (top) vs hard (bottom) attention during caption generation. Source: (Xu et al., 2015).

3.2.4.5 Global vs Local Attention

Luong et al. (2015) coined the *global* vs *local* attention distinction. The global attention is similar to the soft attention, while the local one is an interesting blend between hard and soft, an improvement over the hard attention to make it differentiable: the model first predicts a single aligned position for the current target word and a window centered around the source position is then used to compute a context vector.

Fig. 3.13 compares global and local models of attention in a seq2seq setup.

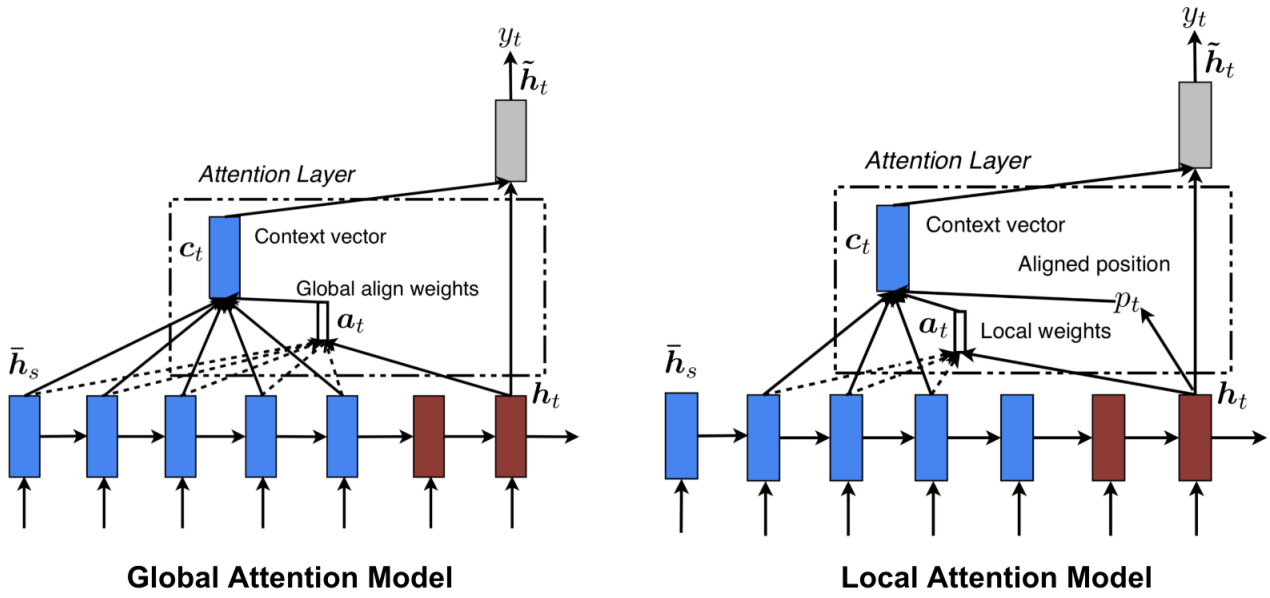


Figure 3.13: Comparative of global (left) vs local (right) attention in the seq2seq model. Source: (Luong et al., 2015).

The are other models that apply similar ideas and can thus be seen as variations of the same idea. We will not describe them here, but we enumerate them together:

- The *Neural Turing Machine*(NTM), by Graves et al. (2014), is a model architecture for coupling a neural network with external memory storage. The memory mimics the Turing machine tape and the neural network controls the operation heads to read from or write to the tape. However, the memory in NTM is finite, and thus it probably looks more like a "Neural von Neumann Machine".

- In problems like sorting or traveling salesman, both input and output are sequential data. Unfortunately, they cannot be easily solved by classic seq2seq or NMT models, given that the discrete categories of output elements are not determined in advance, but depends on the variable input size. The *Pointer Net* (Ptr-Net), was proposed by Vinyals et al. (2015) to address this problem when the output elements correspond to positions in an input sequence. Rather than using attention to blend hidden units of an encoder into a context vector (See Fig. 8), the Pointer Net applies attention over the input elements to pick one as the output at each decoder step.

- The *transformer* model introduced in the "Attention is all you need" paper by Vaswani et al. (2017) is entirely built on the self-attention mechanisms without using sequence-aligned recurrent architecture. It presented a lot of improvements to the soft attention

mechanism, and make it possible to do seq2seq modeling without recurrent network unit, thus reducing the computational costs.

Chapter 4

Model architecture and processes

In this chapter, both the architecture and the training and inference mechanisms of a concrete system are described. We begin introducing the overall architecture of the system, followed by specific sections devoted to the different parts of the system: encoder, decoder, and attention mechanism. Finally, we describe the methods used to train and evaluate the system.

4.1 Overview of the model architecture

As we have discussed in previous section §3.1, we have decided to develop an end-to-end deep learning solution to the image captioning problem based on an **encoder-decoder architecture** with an **attention mechanism**. This is so far the most common architecture used by state of the art systems, as we have seen in our review of the field (Chapter 2.) More specifically, our system will consist of the following components:

- Encoder made of a Convolutional Neural Network (CNN)
- Decoder made of a Recurrent Neural Network (RNN)
- Soft Attention mechanism

The general architecture of the system is depicted in Fig. 4.2. To the left of the figure, there is the encoder, which takes an image as input and extracts a feature map. To the right, the decoder has to produce a caption for the input image, using information from its input, its hidden state, as well as the context information received from the attention mechanism. The attention mechanism lies in between the encoder and the decoder. It is responsible for deciding on which parts of the image (on which features) to focus at each time step while. The attention mechanism takes the image features and the hidden state of the decoder as inputs and generates

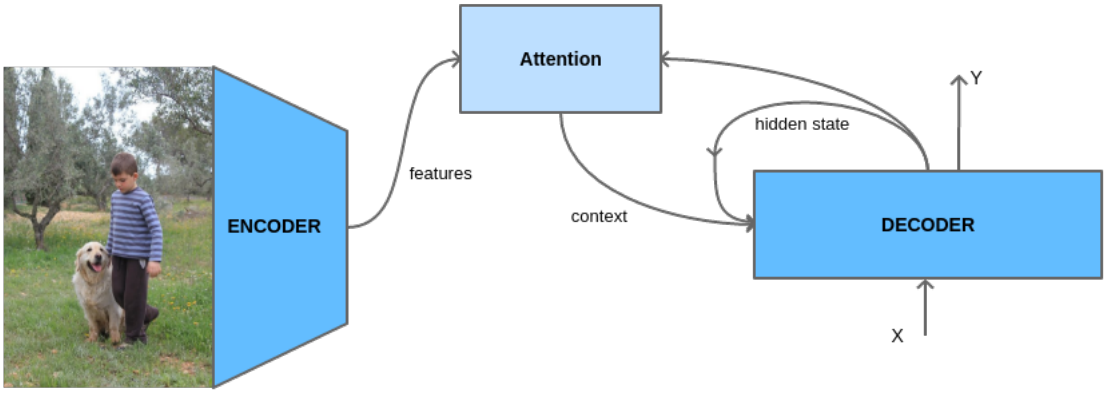


Figure 4.1: Overview of the model architecture: encoder, decoder and attention mechanism

a context vector, which consists of the features weighted according to some learned weights, which represent the relevance given to each feature at a given moment.

Whilst the encoder generates the features in a single forward step, the decoder has to generate a caption word by word, one word at a time. To generate the next word it uses the information from past states, the current input (the last generated word), and the context information. The process of generating a caption as well as the training procedure will be described later in this chapter.

A more detailed view of the architecture is provided in Fig. 4.2, showing the main layers used in the different components of the system.

- The encoder consists of a CNN module pre-trained on the ImageNet dataset. We take the output from the last convolutional layer and pass it through a dense, fully connected layer to flatten it before passing it to the attention module.

- The decoder consists of an embedding layer that converts input tokens into embedding vectors, a recurrent layer, which may be made of either GRU or LSTM units, and two stacked fully connected layer that produces the final output. The input is a concatenation of the external input and the context vector from the attention mechanism.

- The attention mechanism actually consists of two fully connected layers, although they are not showed in the picture. It takes the flattened features from the encoder as input, the hidden states from the decoder, computes attention weights and produces a context vector as output.

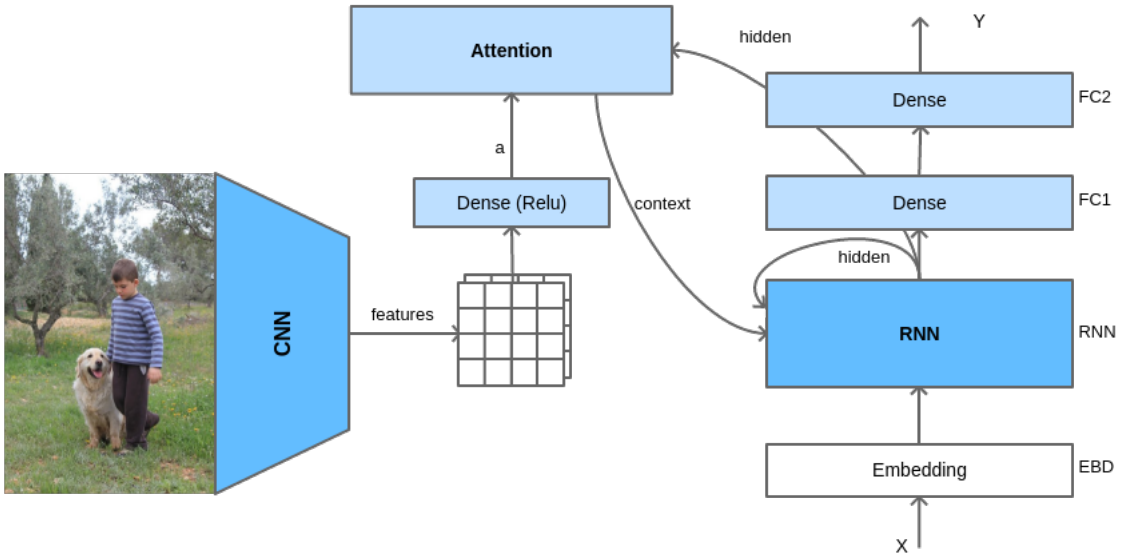


Figure 4.2: Detailed architecture showing the layers of each component

The following sections will describe each component of the network architecture in detail.

4.2 Encoder

The encoder consists of a **CNN** module *pretrained* on the ImageNet dataset. Several options are available for this model. We have included some of the most popular ConvNets, including VGG-16(Simonyan and Zisserman, 2015),Inception-V3(Szegedy et al., 2016), ResNet50(He et al., 2016), and Xception (Chollet, 2017). We also tried to use the NASNet (Zoph et al., 2018) large model, but it was impossible due to memory limitations. Finally, we added the InceptionResnet Szegedy et al. (2016) model to the range of options available for the encoder.

As an example, let us assume we are using Inception-V3 for the CNN module. The architecture of this network is shown in Fig. 4.3.

This network receives an image represented by a 3D tensor with shape $(229, 229, 3)$, and generates an output consisting of 1001 logits, corresponding to the 1000 classification categories in the ImageNet dataset, plus an additional output to denote *other* things not included in the labeled categories. However, we are interested in the last convolutional layer, just before entering the fully connected block. The output of a convolutional layer is a feature map organized as a grid of image patches and a number of features per image patch –the depth, or the number of channels. As shown in Fig. 4.3, the output of the Inception-v3 network's last convolutional layer consists of $8 \times 8 \times 2048$ features, or in general, a 3D tensor of shape

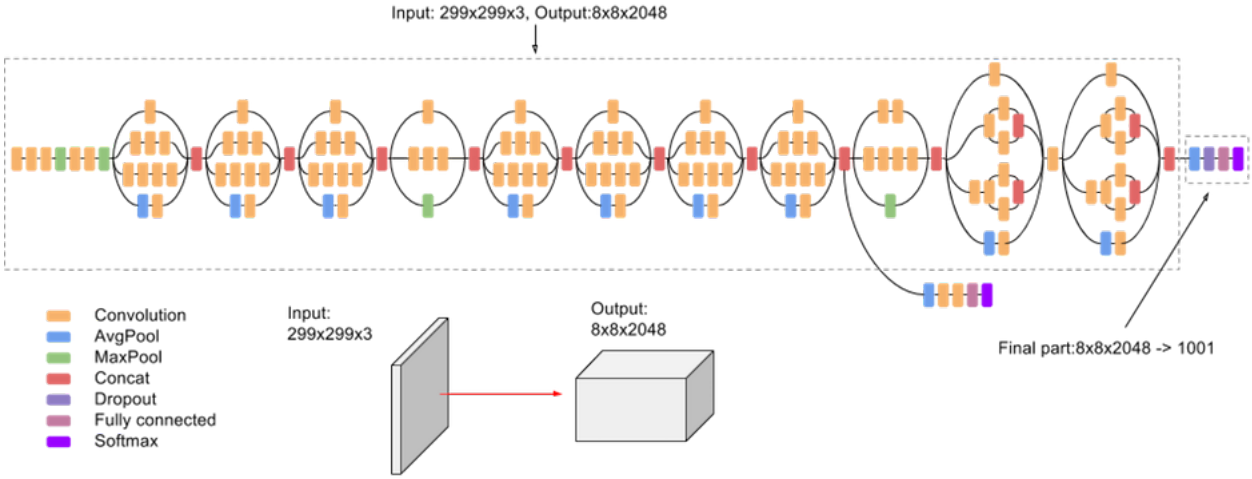


Figure 4.3: Overview of the Inception v3 network architecture. Source Google's Advanced Guide to Inception v3 on Cloud TPU

(f_w, f_h, f_c), where f_w and f_h are the width and height of the feature map, respectively, and f_c is the number of channels. However, instead of using this map as it is, it is squashed into a 2D tensor, with shape $(f_w \times f_h, f_c)$.

Finally, the squashed output of the last convolutional layer is fed into a **fully connected layer** with a ReLu activation function. This layer contains D units, which is a hyperparameter of the model. After forwarding the features map across this layer, we obtain as output a 2D tensor with shape (L, D), where L is the number of image patches, and D is the number of features used to encode each patch (the number of units of the fully connected layer). Note that from an implementation viewpoint, due to the use of batch processing, we should consider another dimension in our data, the batch size. However, from a mathematical perspective, and for the sake of simplicity, we can ignore this additional dimension.

All in all, hereafter, we will use the following notation to represent the output of the encoder. Let us call each set of features for a certain image patch as an *annotation* vector, denoted by a_i. Then, the set of feature vectors can be defined as:

$$a = \{\mathrm{a}_1, ..., \mathrm{a}_L\}, \mathrm{a}_i \in \mathbb{R}^D$$

This approach uses the features of the convolutional layer instead of the fully connected layer of the encoding CNN. This will allow the decoder to selectively focus on certain parts of an image by weighting a subset of all the feature vectors.

In our system, the number of features L used to encode images is a configurable hyperparameter (config.num_features), as is the name of the specific CNN used to extract image features (config.cnn).

4.3 Decoder

The goal of the decoder is to generate a caption for the input image, that is, a sentence that describes what can be seen in the image. We will use y to denote the output of the decoder, which is a sequence of the outputs generated at each time step.

$$y = \{\mathrm{y}_1, ..., \mathrm{y}_c\}, \mathrm{y}_i \in \mathbb{R}^K$$

where K is the size of the vocabulary (number of words), and C is the length of the caption. Each output y_i is a vector with size K that represents a probability distributions over the language vocabulary. That is, each element represents the probability of choosing one specific word. There are different ways to choose the final sentence from this sequence of probability distributions, as we will discuss later in this chapter.

The core element of the decoder is a single **recurrent layer** (named **RNN** in Fig. 4.2), which consists of a number of recurrent unit with hidden state, either GRU or LSTM units indistinctly. Both the type of recurrent unit and the number of units are hyperparameters of the model.

In addition to the recurrent layer, the encoder also comprises an embedding layer and two fully connected layers stacked on top of the recurrent layer.

The **embedding layer** takes an input consisting of a word token, and transforms it into a dense vector embedding using an embedding matrix $E \in \mathbb{R}^{m \times K}$, where m is the dimension of the embedding vector. During the caption generation process, the input to the embedding layer is the last token generated, so we denote the result of the embedding as $E\mathrm{y}_{t-1}$.

The input to the recurrent layer at a given timestep t will consist of both the result of the embedding layer $E\mathrm{y}_{t-1}$, a context vector $\hat{z}_t$ coming from the attention mechanism, as well as the previous hidden state h_{t-1}.

The context vector $\hat{z}_t$ is a dynamic representation of the relevant part of the image input at time t. Its computation is described in the next section.

The fully connected layers FC1 and FC2 constitute an MLP that is used to generate a sequence of words based on the output of the RNN. These two layers make a projection of the hidden state space into the output space. FC1 simple projects the RNN output into a tensor of shape (C, n), where C is the maximum sentence length, and n is the number of RNN units. FC2 takes the output from FC1 and projects it into the final output space, that is, the language model. In particular, this final output is a sequence $\{\mathrm{y}_1, ..., \mathrm{y}_c\}$ of probability distributions of words in the vocabulary. That is, each element y_i is a 1D tensor with shape (K), made of values in [0,1] representing the probability of a different word from the language vocabulary. There are different methods to convert these probability distributions into a sentence, some

which will be described later (§4.6).

4.3.1 Attention

The attention mechanism is responsible for deciding on which parts of the image to focus on at each timestep. In particular, we have designed an attention mechanism inspired by the model described in *Show, Attend and Tell*, by Xu et al. (2015), which is in turn based on the mechanism proposed by Bahdanau et al. (2015) as an addition to the *seq2seq* model. More specifically, we replicate the *soft* attention mechanism proposed in the paper. This mechanism weights the features of each part of the image depending on a relevance score that is computed taking into account the current state of the decoder. This type of attention is called *soft*, or *global*, because it attends to the entire input space, unlike *hard*/*local* mechanisms, which attend to a specific location of the input space at any given moment, i.e. a patch of the image. Furthermore, this is a deterministic form of attention, as opposed to stochastic mechanisms, which use probability distributions to select the part of the input to focus at. Finally, this form attention can also be classified as self-attention, since the decoder attends to the encoder to generate its output, but it is also guided by its own internal state, as we will soon describe more formally. For general information and details on this type of attention, please see §4.3.1.

The output of the attention mechanism is called the context vector $\hat{z}_t$ we have already introduced. Now let us see how this vector is computed.

We define a mechanism ϕ that computes $\hat{z}_t$ from the annotation vectors $\mathrm{a}_i, i = 1, ..., L$ corresponding to the features extracted at different image locations. For each location i, the mechanism generates a positive weight of α_i which is interpreted as the relative importance to give to location i in blending the a_i's together. The weight α_i of each annotation vector a_i is computed by an *attention model* f_{att} for which we use a multilayer perceptron (not depicted in Fig. 4.2), conditioned on the previous hidden state h_{t-1}. To emphasize, note that the hidden state varies as the RNN advances in its output sequence: "where" the network looks next depends on the sequence of words that have already been generated (represented by the hidden state).

$$e_{ti} = f_{\mathrm{att}}(\mathrm{a}_i, \mathrm{h}_{t-1}) \tag{4.1}$$

The attention weights can be computed from the attention scores e_{ti} by scaling to $[0, 1]$ and normalizing (making weights sum to one).

$$\alpha_{ti} = \frac{\exp\left(e_{ti}\right)}{\sum_{k=1}^{L} \exp\left(e_{tk}\right)} \tag{4.2}$$

The attention weights represent the alignment between parts of the image and words in the caption that is being generated, as we will later illustrate with some pictures. Once the attention weights are computed, the context vector $\hat{z}_t$ can be computed by:

$$\hat{z}_t = \phi(\{\mathrm{a}_i\}, \{\alpha_i\}) \tag{4.3}$$

where ϕ is a function that returns a single context vector given the set of annotation vectors and their corresponding weights. There are different approaches to formulate this function depending on the attention model being used. On the one hand, learning a hard –stochastic– attention model requires sampling the attention locations s_t each timestep. However, for a soft attention model, we can take the expectation of the context vector $\hat{z}_t$ directly,

$$\mathbb{E}_{p(s_t|a)}[\hat{z}_t] = \sum_{i=1}^{L} \alpha_{ti}\mathrm{a}_i,$$

and formulate a deterministic attention model by computing a weighted annotation vector

$$\phi(\{\mathrm{a}_i\}, \{\alpha_i\}) = \sum_{i=1}^{L} \alpha_i \mathrm{a}_i, \tag{4.4}$$

as proposed by Bahdanau et al. (2015). These weighted vectors α_i are used as a soft context that weights the different parts of the image with variable degrees of relevance, depending on the state of the decoder, ie. the sentence that has been generated so far. The whole model is smooth and differentiable under deterministic attention, so learning end-to-end is trivial by using standard back-propagation.

In particular, to compute the attention weights α we first compute the scores e_{eti} using the additive scoring mechanism proposed by (Bahdanau et al., 2015). This mechanism can be implemented by a multilayer perceptron with a *tanh* activation function applied over the addition of the annotation vector (the image features for different image locations), and the hidden state of the decoder's RNN.

More formally, the *attention model* f_{att} is formulated as:

$$f_{\text{att}} = v_a^\top \tanh(W_1 a_t + W_2 h_t) \tag{4.5}$$

where W_1, W_2, and V_a are weights to be learned by backpropagation during the training phase.

Finally, note that computing the attention weights α_i as defined in Eq. (4.2) is equivalent to applying a *softmax* function over the attention scores e_{ti}. Therefore, equations Eq. (4.1) and Eq. (4.5) can be implemented by a combination of three fully connected layers and two

activation functions, as depicted in Fig. 4.4.

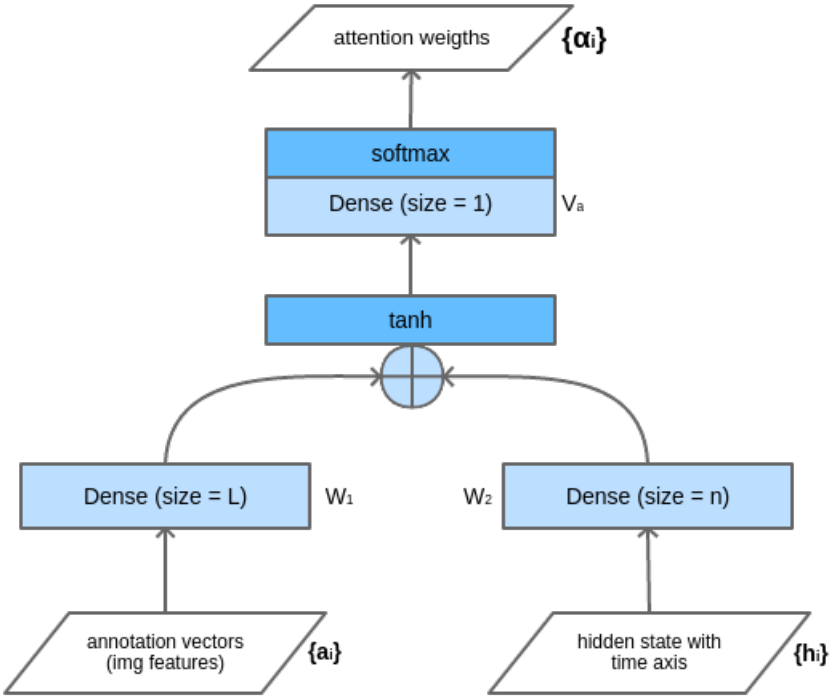


Figure 4.4: Detailed architecture of the additive soft attention mechanism

In our model, the number of units, and thus the dimension of the hidden state, is a configurable hyperparameter (config.rnn_units). In addition, the type of RNN unit to use is also specified as a hyperparameter (config.rnn) with two possible values: 'gru' or 'lstm'.

4.4 Data pipelines

This section describes the data preparation steps needed to train our model over a benchmark dataset such as MS COCO. Fitting a model over a big dataset will require processing large amounts of data, and thus may highly benefit from as much parallelization as possible. To that aim, we will organize our data to support parallel pipeline processing by leveraging data parallelization structures, such as the Dataset class available in Tensorflow.

A general overview of the data pipeline is provided in Fig. 4.5. It all begins with a raw dataset, as the COCO dataset, which consists of a number of images and associated captions in some dataset-specific format. Then, we perform separate preprocessing tasks for the images and for the captions. In order to accelerate training and evaluation tasks, image features are computed once and cached to disk. Once image features have been extracted and cached, and

the captions have been converted into sequences ready to be set into the RNN, images and their associated sequences are combined into parallel-efficient data structures supporting buffering and batching. If the data is being used for training, then it is also randomly shuffled before generating batches.

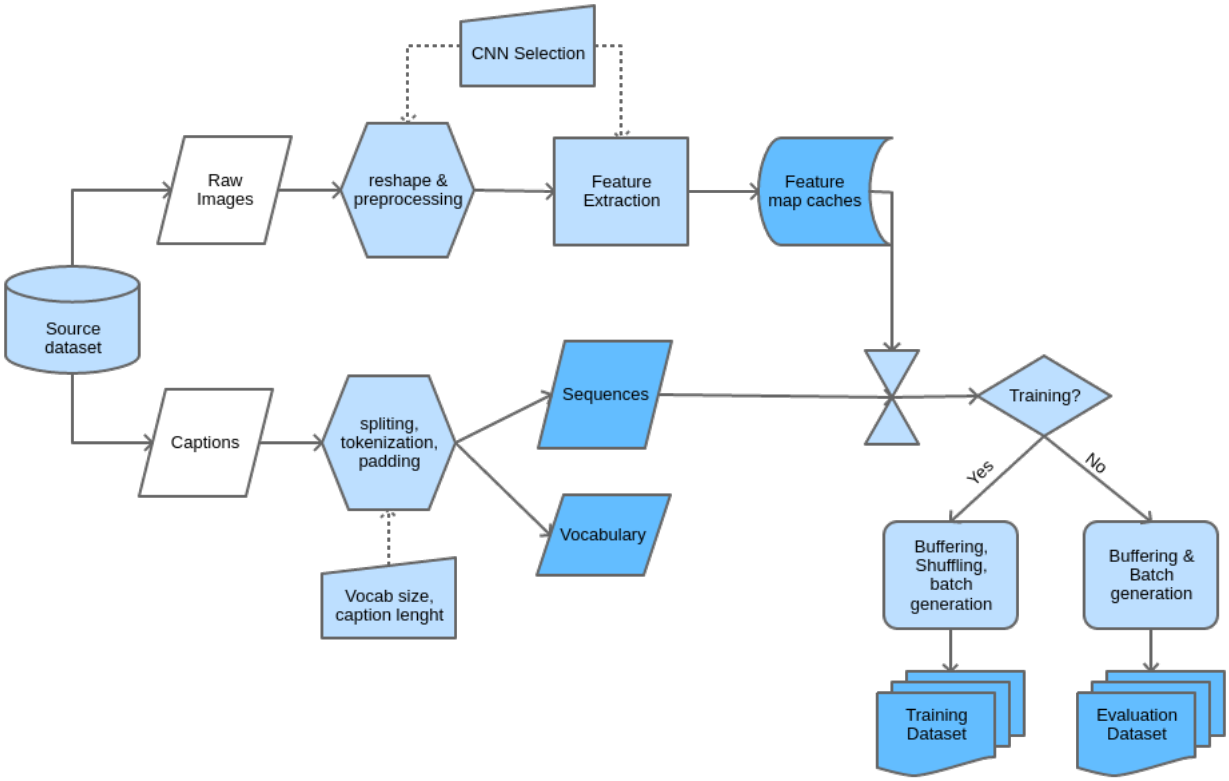


Figure 4.5: Overview of the data preparation pipeline

The section is organized as follows, first we describe the pre-processing steps required to prepare images for the pipeline, then we describe the steps required to prepare the captions, and finally, we describe the way images and captions are collated together into parallel efficient structures before being fed to the training or evaluation processes.

4.4.1 Image pre-processing

Images have to be encoded by a CNN to extract the visual features needed by the decoder. In order to be fed into the CNN, images have to be preprocessed first, which includes resizing them to the shape required by the particular CNN being used, as well as applying tweaks that are specific of each particular CNN. Some models use images with values ranging from 0 to 1, others from -1 to +1, yet others use the *Caffe*[1] style, that is not normalized but is centered.

[1] *Caffe* is an open source deep learning framework, originally developed at University of California, Berkeley. More info can be found at the official website.

To support using different ConvNets for the encoder, we have developed a generic method that can preprocess images for different models. Table 4.1 summarizes the main characteristics of some of the most popular ConvNet architectures that are available to the research community as a pre-trained model, typically trained on the ImageNet dataset.

Table 4.1: Summary of popular ConvNets pretrained on the ImageNet dataset

Model	Image size	Weights size	Top-1 acc.	Top-5 acc.	Params	Depth
Xception	299 x 299	88 MB	0.790	0.945	22,910,480	126
VGG16	224 x 224	528 MB	0.715	0.901	138,357,544	23
VGG19	224 x 224	549 MB	0.727	0.910	143,667,240	26
ResNet50	224 x 224	99 MB	0.759	0.929	25,636,712	168
InceptionV3	299 x 299	92 MB	0.788	0.944	23,851,784	159
InceptResNetV2	299 x 299	215 MB	0.804	0.953	55,873,736	572
MobileNet	224 x 224	17 MB	0.665	0.871	4,253,864	88

Images can be encoded into feature maps beforehand, that is, prior to using them for training or inference purposes. That means we only need to generate the image feature maps once and save them to disk. To do that aim, in addition to having commands for the training and evaluation steps, we have also included a command called prepare which simply generates and saves to disk image features for a certain CNN.

It is important to remark that for encoding images we are using the features obtained by the last convolutional layer of the network, excluding the fully connected parts that use to lie at the top, since these networks are pre-trained for a classification task[2]. The size of the feature maps generated by the encoding process also differs for different ConvNets. These maps are encoded as grids made of image patches with a certain width W, height H, and depth D. Note that the depth is simply the number of features per patch, and is equal to the number of channels of the last convolution layer. For example, the InceptionV3 model produces feature maps with shape $(8 \times 8 \times 2048)$. However, in the end, all the image features will be passed through a fully connected layer that will squash them into 2D tensor with shape (L, D), as described in section §4.2.

The specific pre-trained CNN to use in the encoder constitutes a hyperparameter of the model (config.cnn) which may adopt the following values: 'vgg16', 'resnet50', 'inception', 'xception', and 'nasnet_large' (although we were unable to use the latest due to lack of memory).

[2]Typically these models have the weights of the networks that participated in some edition of the ImageNet Large Scale Visual Recognition Competition (ILSVRC)

4.4.2 Text pre-processing

Captions are short sentences in natural language. The ultimate goal of the image captioning problem is to generate captions for new images not included in the training dataset. Therefore, the task of generating a caption can be seen as a task of predicting a sequence of symbols, which implies learning some model of the language. For building such models, we need to turn text into a format that we can optimize over, i.e. we need to map sentences of the language into numerical vectors. These vectors provide a numerical representation of the symbols in the language. First of all, we must decide on the level of granularity used to represent the symbols of the language. One strategy is to treat each word as a unique entity, e.g. (Salton et al., 1975). At the other extreme lies the strategy to predict one character at a time, as suggested e.g. by Ling et al. (2015) [3].

Going into details about language models is beyond the scope of this section. Here it suffices to say that we need a strategy to convert textual captions into a sequence of tokens. A token is a data point the model will train and predict. For seq2seq tasks like translation and image captioning the most common approach is to use words as tokens, and that is the approach we adopt here.

4.4.2.1 Tokenization and language modelling

To begin with, we need to split captions into tokens, that is, words. As part of this process, we will also remove non-alphanumeric characters and punctuation marks.

Then, we need to map tokens into numerical indices. We often call it a *vocabulary*. Its input is a list of tokens, called a corpus. Then it counts the frequency of each token in this corpus, and assigns a numerical index to each token according to its frequency, from most frequent to more rare tokens. The size of the vocabulary is limited, so that rare tokens are removed to reduce the complexity. In order to mark the beginning and end of a sentence, two special tokens are added to the vocabulary: a token named "<start>" (also named "<sos>" or "<bos>") to mark the beginning of sentence, and "<end>" (or "<end>") for the ending of a sentence. Furthermore, another token named "<unk>" is used to represent words that are not included in the vocabulary. Finally, since we may be representing sentences of different length, we also need a special token named "<pad>" for padding shorter sentences. It is common practice to use 0 as the index of the "<pad>" token so that it is ignored (it has no activation effects).

In our system, the corpus is made of all the captions included in the training dataset.

[3] Actually, another alternative between both strategies is byte-pair encoding, as described by Sennrich et al. (2016) for the purpose of neural machine translation. It decomposes a text into syllable-like fragments that occur frequently. This allows for models that are able to generate words like heteroscedastic or pentagram based on previously viewed words, e.g. heterogeneous, homoscedastic, diagram, and pentagon

We have included configuration options to limit the number of words in the vocabulary. For example, for the COCO dataset, the complete corpus consists of almost 30,000 words, but we have limited the vocabulary size to 10,000 words, which seems to be a reasonable size. This vocabulary only has to be built once, since it can be saved and loaded from disk. Needless to say that the vocabulary must be rebuilt whenever a new corpus is used or the existing corpus changes. The size of the vocabulary is a configurable hyperparameter of our model (config.vocabulary_size).

One-Hot encoding One-hot encoding vectors provide an easy way to express words as vectors in order to process them in a deep network. In a nutshell, we map each word to a different unit vector: assume that the number of different words in the dictionary is K and each word has a one-to-one correspondence with a single value in the index of successive integers from 0 to $K-1$. If the index of a word is the integer i, then we create a vector $\mathbf{e}_i$ of all 0s with a length of K and set the element at position i to 1. This vector is the *one-hot* vector representation of the original word.

Word embeddings One-Hot vectors provide a sparse vector representation of words. *Word embedding*, however, refers to a technique that uses dense, vectors of real numbers to represent words. Such a vector can be thought of as the feature vector of a word.

Although one-hot word vectors are easy to construct, they are usually not a good choice. One of the major reasons is that the one-hot word vectors cannot accurately express the similarity between different words, such as the cosine similarity that we commonly use. For the vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, their cosine similarities are the cosines of the angles between them:

$$\frac{\mathbf{x}^\top \mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|} \in [-1, 1].$$

Since the cosine similarity between the one-hot vectors of any two different words is 0, it is difficult to use the one-hot vector to accurately represent the similarity between multiple different words.

There are various approaches to learn the word embeddings space, being the *skip-gram*(Mikolov et al., 2013) and the *bag-of-words* (BoW) [4] are two of the most popular.

Although there exist open source tools (eg. Word2Vec) and pre-trained language models (Glove) available, for this project we decided to take another approach: learning the embedding matrix as an integral part of the complete, end-to-end learning process. This can be a slower

[4]The BoW model is a simplifying representation used in natural language processing and information retrieval (IR). In this model, a text is represented as the bag (multiset) of its words, disregarding grammar and even word order but keeping multiplicity. This model dates from Zellig Harris's 1954 article on Distributional Structure

approach but tailors the language model to the task at hand and the specific dataset.

From a practical viewpoint, learning a tailored model can be achieved simply by adding an *embedding* layer as the first layer of the decoder, as described above (section §4.3). That way, we will not need to embed the words beforehand, as a previous step. Instead, the embedding matrix will be learned just like any other layer of the model, and it will suffice with providing a one-hot encoding representation of captions to train the model.

Such an embedding layer has two hyperparameters that are added to our model as configurable options: the size of the vector space in which words will be embedded (config.embedding_dim), and the size of the vocabulary (config.vocabulary_size)

4.4.3 Dataset generation

We have already described the image preprocessing steps and the text preprocessing that has to be carried out to prepare our data prior to using it. However, in order to efficiently use this data as input to the full pipeline of our end-to-end model, we need to organize it into special data structures supporting batching, shuffling, prefetching (buffering), and parallelism. Fortunately, Tensorflow a data structure supporting all those features, by the name of Dataset. This structure allows linking together different data sources, as long as the data can be converted to tensors, which includes all primitive data types. That means we can build an instance of Dataset that collates together the feature maps obtained from a set of images, and the one-hot tokenized representations of the captions associated with those images. Therefore, we could prefetch, shuffle and organize that data in batches.

In our system, we have added the batch size as a hyperparameter that can be customized (config.batch_size).

We have created methods to create two different Dataset instances, one for the training portion of the image-caption dataset, and another one its the evaluation portion.

4.5 Training the model

Training is the process of fitting the model to a particular dataset. In many tasks, including image captioning, training is supervised, that is, dataset instances are labeled with the solutions. In these cases, the model is fitted to the data by modifying its parameters (eg. the weights of a neural network) so that a particular loss function is minimized.

In the image captioning problem, the data examples are the images, and the labels are the captions, expressed in natural language. Usually, these captions are provided by humans that

were asked to describe the images[5]. In this particular case, the loss function is computed by comparing word by word the sentence generated by the RNN against the provided caption. However, due to the sequential nature of the output, this is not as simple as computing the loss with non-sequential outputs, like typically happens when solving a classification task. In particular, we will use a specific technique named **Teacher Forcing**, which is often used for learning sequential models and will be explained below.

4.5.1 Loss function

In order to use supervised training, we need to minimize a loss function. In the image captioning problem, the error will be related to the generation of the caption: the actual caption constitutes the ground truth, while the caption generated by the model will be the prediction. However, a sequence is not generated in a single step, instead, it is computed word by word, and the loss will be computed word by word. Let us see how the training unfolds sequentially in order to understand how to compute a loss function.

On the one hand, the image associated to the example caption is processed by the encoder ConvNet and results in a 2D tensor of features, which are then weighted by the attention mechanism, and conveniently reshaped before being fed into the decoder RNN as. The resulting tensor corresponds to the *context* explained above (§4.3.1): $\hat{z}_t$

On the other hand, the caption is converted into a sequence of embedding vectors, once per each word (Ex_t). The word embeddings have to be concatenated with the context vectors to produce the final input to the RNN, but this process does not happen as a single step. Instead, the word embeddings have to be processed one by one, and for each embedding Ex_t a new context vector will be computed by the attention mechanism and concatenated to make the input at timestep t:

$$[\hat{z}_t; Ex_t]$$

Given that input and the previous hidden state t_{t-1}, the RNN will generate a new output of y_t. The output is a vector of *logits* with dimension equals to the number of words in the vocabulary, K. That is, each element of the output vector is the probability of a word in the vocabulary. Such output can be interpreted as the prediction of a particular word so that it can be compared against the ground truth to compute a loss value, where the ground truth refers to the actual word in the t^{th} position of the caption. This can be seen as a multitask classification task, with as many classes as words in the vocabulary. Furthermore, since the ground truth is

[5]This can be done in large quantities thanks to crowdsourcing marketplaces like the Amazon Mechanical Turk (MTurk).

represented using one-hot encoding, then we should use a SparseCategoricalCrossentropy loss function to compute the loss between the predicted word and the actual word.

The losses for every word in the sentence will be computed separately and summed together to obtain a single loss value. Finally, the accumulated loss shall be divided by the length of the caption to obtain a loss value that is independent of the length of the caption.

Besides, when computing the loss value, padding tokens should be avoided, which is achieved by using a masked version of the loss function.

4.5.2 Teacher forcing

There are sequence prediction models that use the output from the last time step y_{t-1} as input for the model at the current time step X_t. This type of model is common in language models that output one word at a time and use the output word as input for generating the next word in the sequence. For example, this type of language model is used in an Encoder-Decoder recurrent neural network architecture for sequence-to-sequence generation problems such as machine translation, text summarization, question answering, and image captioning.

Teacher forcing is a strategy for training recurrent neural networks that uses the model output from a prior time step as an input. This method was originally proposed as an alternative technique to backpropagation through time for training a recurrent neural network (Williams and Zipser, 1989).

Teacher forcing works by using the expected output from the training dataset at the current time step t as input in the next time step $t+1$, rather than the output generated by the network, as is the case when the network is predicting a new sequence (as is the case during the validation stage). Let us explain Teacher forcing with a concrete example.

Consider an image of a tennis player with the following caption:

```
<start> a man standing on a tennis court holding a tennis racquet <end>
```

Fig. 4.6 depicts the inputs at each time step for the example above. We are not showing the outputs here because it is not relevant. The loss function will be computed between pairs (input, output), aggregated and divided by the length of the caption.

Teacher forcing is a fast and effective way to train a recurrent neural network that uses the output from prior time steps as input to the model. Nevertheless, this approach can also result in models with poor prediction performance as the RNN's conditioning context (the sequence of previously generated samples) diverge from sequences seen during training(Lamb et al., 2016). This is common in most applications of this type of model as the outputs are probabilistic in nature. This type of application of the model is often called open loop.

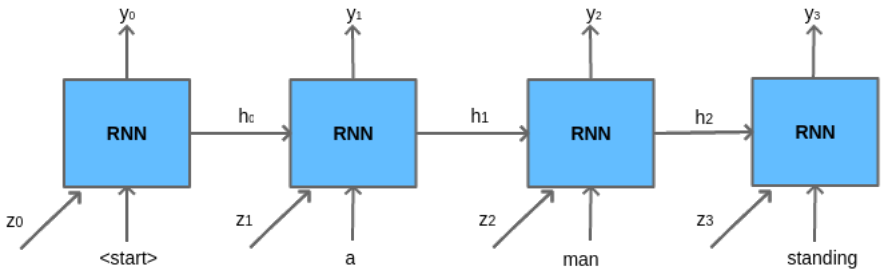


Figure 4.6: Teacher forcing for image captioning training. The expected word is used as input of the network at each timestep

4.6 Inference and validation

After the model is trained, it can be used to generate new sequences. To do that, we should simply feed a "<start>" token to the RNN start the process, and the generated word in the output sequence will then be used as input on the subsequent time step, along with the context vector coming from the attention mechanism. This process is depicted in Fig. 4.7

The generation process will continue until the "<end>" token is reached, or a maximum sequence length is reached.

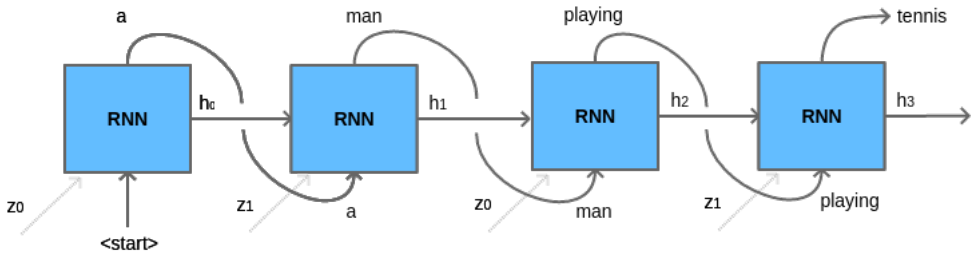


Figure 4.7: Word-by-word caption generation

Various approaches exist to predict the next token. For example, when using Greedy search, the token with the highest score at each time step is chosen and used to predict the next token. However, there are more sophisticated approaches, introduce below.

For ease of discussion, we assume that the output of the decoder is a sequence of text. Let the size of output text dictionary $\mathcal{Y}$ (contains special symbol "<end>") be $|\mathcal{Y}|$, and the maximum length of the output sequence be T'. There are a total $\mathcal{O}(|\mathcal{Y}|^{T'})$ types of possible output sequences. All the subsequences after the special symbol "<end>" in these output sequences will be discarded.

4.6.1 Greedy Search

First, we will take a look at a simple solution: Greedy search. For any time step t' of the output sequence, we are going to search for the word with the highest conditional probability from $|\mathcal{Y}|$ numbers of words, with

$$y_{t'} = \operatorname*{argmax}_{y \in \mathcal{Y}} \mathbb{P}(y \mid y_1, \ldots, y_{t'-1}, \boldsymbol{c})$$

as the output. Once the "<end>" symbol is detected, or the output sequence has reached its maximum length T', the output is completed.

As we mentioned in our discussion of the decoder, the conditional probability of generating an output sequence based on the input sequence is $\prod_{t'=1}^{T'} \mathbb{P}(y_{t'} \mid y_1, \ldots, y_{t'-1}, \boldsymbol{c})$. We will take the output sequence with the highest conditional probability as the optimal sequence. The main problem with Greedy search is that there is no guarantee that the optimal sequence will be obtained.

Take a look at the example in Fig. 4.8. We assume that there are four words "A", "B", "C", and "<end>" in the output dictionary. The four numbers under each time step represent the conditional probabilities of generating "A", "B", "C", and "<end>" at that time step. At each time step, greedy search selects the word with the highest conditional probability. Therefore, the output sequence "A", "B", "C", "<end>" will be generated in Fig. 4.8. The conditional probability of this output sequence is $0.5 \times 0.4 \times 0.4 \times 0.6 = 0.048$.

Time step	1	2	3	4
A	0.5	0.1	0.2	0.0
B	0.2	0.4	0.2	0.2
C	0.2	0.3	0.4	0.2
<eos>	0.1	0.2	0.2	0.6

Figure 4.8: The four numbers under each time step represent the conditional probabilities of generating "A", "B", "C", and "<end>" at that time step. At each time step, greedy search selects the word with the highest conditional probability.

Now, we will look at another example shown in Fig. 4.9. Unlike in Fig. 4.8, Fig. 4.9 selects the word "C" for it has the second highest conditional probability at time step 2. Since the output subsequences of time steps 1 and 2, on which time step 3 is based, are changed from "A" and "B" in Fig. 4.8 to "A" and "C" in Fig. 4.9, the conditional probability of each word generated at time step 3 has also changed in Fig. 4.9. We choose the word "B", which has the highest conditional probability. Now, the output subsequences of time step 4 based on the first

three time steps are "A", "C", and "B", which are different from "A", "B", and "C" in Fig. 4.8. Therefore, the conditional probability of generating each word in time step 4 in Fig. 4.9 is also different from that in Fig. 4.8. We find that the conditional probability of the output sequence "A", "C", "B", "<end>" at the current time step is $0.5 \times 0.3 \times 0.6 \times 0.6 = 0.054$, which is higher than the conditional probability of the output sequence obtained by greedy search. Therefore, the output sequence "A", "B", "C", "<end>" obtained by the greedy search is not an optimal sequence.

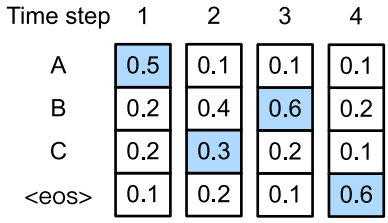


Figure 4.9: The four numbers under each time step represent the conditional probabilities of generating "A", "B", "C", and "<end>" at that time step. At time step 2, the word "C", which has the second highest conditional probability, is selected.

4.6.2 Exhaustive Search

If the goal is to obtain the optimal sequence, we may consider using Exhaustive search: an exhaustive examination of all possible output sequences, which outputs the sequence with the highest conditional probability.

Although we can use an exhaustive search to obtain the optimal sequence, its computational overhead $\mathcal{O}(|\mathcal{Y}|^{T'})$ is likely to be excessively high. For example, when $|\mathcal{Y}| = 10000$ and $T' = 10$, we will need to evaluate $10000^{10} = 10^{40}$ sequences. This is next to impossible to complete. The computational overhead of greedy search is $\mathcal{O}(|\mathcal{Y}|\,T')$, which is usually significantly less than the computational overhead of an exhaustive search. For example, when $|\mathcal{Y}| = 10000$ and $T' = 10$, we only need to evaluate $10000 \times 10 = 1 \times 10^5$ sequences.

4.6.3 Beam search

Beam search is an improved algorithm based on Greedy search. It has a hyper-parameter named beam size. We set it to k. At time step 1, we select k words with the highest conditional probability to be the first words of the k candidate output sequences. For each subsequent time step, we are going to select the k output sequences with the highest conditional probability from the total of $k\,|\mathcal{Y}|$ possible output sequences based on the k candidate output sequences from the

previous time step. These will be the candidate output sequence for that time step. Finally, we will filter out the sequences containing the special symbol "<end>" from the candidate output sequences of each time step and discard all the subsequences after it to obtain a set of final candidate output sequences.

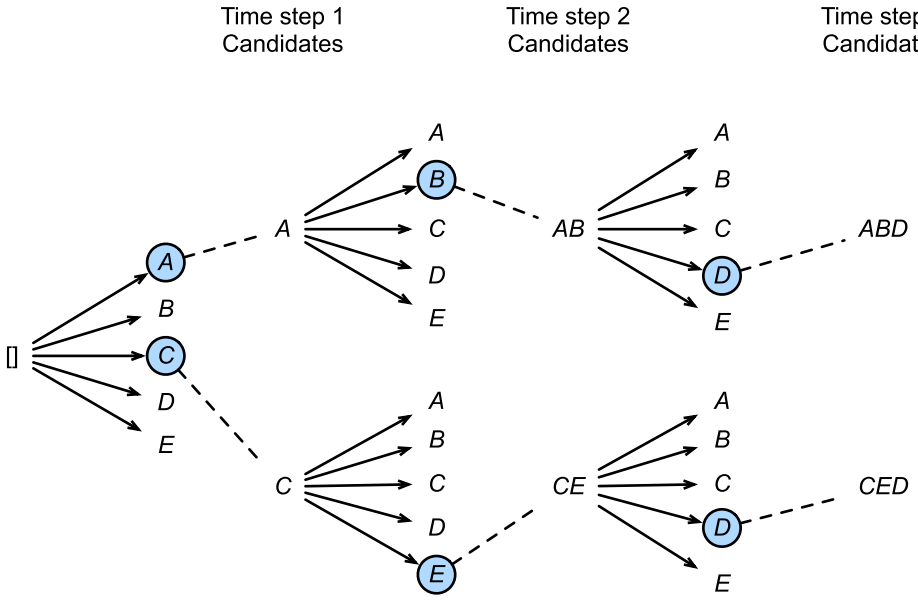


Figure 4.10: The beam search process. The beam size is 2 and the maximum length of the output sequence is 3. The candidate output sequences are A, C, AB, CE, ABD, and CED.

Fig. 4.10 demonstrates the process of beam search with an example. Suppose that the vocabulary of the output sequence only contains five elements: $\mathcal{Y} = \{A, B, C, D, E\}$ where one of them is a special symbol "<end>". Set beam size to 2, the maximum length of the output sequence to 3. At time step 1 of the output sequence, suppose the words with the highest conditional probability $\mathbb{P}(y_1 \mid \boldsymbol{c})$ are A and C. At time step 2, we compute $\mathbb{P}(A, y_2 \mid \boldsymbol{c}) = \mathbb{P}(A \mid \boldsymbol{c})\mathbb{P}(y_2 \mid A, \boldsymbol{c})$ and $\mathbb{P}(C, y_2 \mid \boldsymbol{c}) = \mathbb{P}(C \mid \boldsymbol{c})\mathbb{P}(y_2 \mid C, \boldsymbol{c})$ for all $y_2 \in \mathcal{Y}$, and pick the largest two among these 10 values: say $\mathbb{P}(A, B \mid \boldsymbol{c})$ and $\mathbb{P}(C, E \mid \boldsymbol{c})$. Then at time step 3, we compute $\mathbb{P}(A, B, y_3 \mid \boldsymbol{c}) = \mathbb{P}(A, B \mid \boldsymbol{c})\mathbb{P}(y_3 \mid A, B, \boldsymbol{c})$ and $\mathbb{P}(C, E, y_3 \mid \boldsymbol{c}) = \mathbb{P}(C, E \mid \boldsymbol{c})\mathbb{P}(y_3 \mid C, E, \boldsymbol{c})$ for all $y_3 \in \mathcal{Y}$, and pick the largest two among these 10 values: say $\mathbb{P}(A, B, D \mid \boldsymbol{c})$ and $\mathbb{P}(C, E, D \mid \boldsymbol{c})$. As a result, we obtain 6 candidates output sequences: (1) A; (2)C; (3) A, B; (4)C, E; (5)A, B, D; and (6)C, E, D. In the end, we will get the set of final candidate output sequences based on these 6 sequences.

In the set of final candidate output sequences, we will take the sequence with the highest score as the output sequence from those below:

$$\frac{1}{L^\alpha} \log \mathbb{P}(y_1, \ldots, y_L) = \frac{1}{L^\alpha} \sum_{t'=1}^{L} \log \mathbb{P}(y_{t'} \mid y_1, \ldots, y_{t'-1}, \boldsymbol{c}),$$

Here, L is the length of the final candidate sequence and the selection for α is generally 0.75. The L^α on the denominator is a penalty on the logarithmic addition scores for the longer sequences above. The computational overhead $\mathcal{O}(k\,|\mathcal{Y}|\,T')$ of the beam search can be obtained through analysis. The result is a computational overhead between those of greedy search and exhaustive search. In addition, Greedy search can be treated as a beam search with a beam size of 1. Beam search strikes a balance between computational overhead and solution quality using a flexible beam size of k.

Note: In the developed system, we have implemented both greedy search and beam search, however, results with Beam search hasn't been satisfactory.

4.7 Implementation

The software developed for this Master Thesis Project is available as an open source project under a Creative Commons license (CC Attribution-ShareAlike 4.0). The code repository can be found here: https://github.com/magomar/image_captioning_with_attention

Chapter 5

Experiments

5.1 Dataset

We were considering four datasets to conduct our experiments. We also considered using two or more datasets, but in the end, we decided to perform all our experiments using the current benchmark dataset, the one that is used by researchers to compare their models and publish their results: The COCO dataset, released by Microsoft in 2014 Lin et al. (2014). At the moment of writing this report, this is without a doubt the preferred benchmark dataset for the image captioning problem. In addition, there is an online server that automates the evaluation and comparison of submitted results. This server is running 24/7, and results are publicly available, so it is actually an open competition, known as the Microsoft COCO Image Captioning Challenge.

Figure 5.1: Image caption examples. Source: COCO Image Captioning Task

In the end, we only used this dataset because the other common options are clearly outdated, such as the Flickr8K and Flickr30K. Although the latter was deemed as an interesting option

for faster experiments in the first stages of the project, we finally upgraded the hardware with a very powerful GPU that made it feasible to use the COCO dataset as the only experimental dataset, and still conduct a considerable number of experiments.

As a matter of fact, there is still a larger dataset supporting image captioning, the Conceptual Captions dataset, by Google (Sharma et al., 2018), which also offers an online evaluation server and a public leaderboard. However, it was released just a few months ago, and it is so huge, that it will probably take sometime before it is widely adopted by researchers[1].

For our project, we are using the images and the associated captions included in this dataset, which are divided into two separate sets: one for training, and another one for validation purposes.

5.1.0.1 Training dataset

This dataset includes a total of 82.783 unique images which are supposed to have 5 captions each, which would amount to 413.915 captions. However, there are actually 414.113, probably because some image has more than 5 captions.

The length (number of words) of the captions is highly variable, with a maximum length of 51. The full corpus of text in the training captions dataset is 31.437.

After some initial analysis on the set of captions, we realized that a vast majority of captions are much shorter than the maximum length, therefore, for the sake of speeding up the processing of the recurrent layer, we decided to restrict the dataset to those captions having a maximum length of 15, which implied removing 13.370 images and 66.885 captions from the training dataset. This filter was applied before tokenizing the captions, thus in practice, when including the "<start>" and "<end>" tokens the maximum length of a sequence would be 17. Note the maximum length of the captions is actually a configurable parameter of the model (config.max_caption_length)

All in all, after filtering the dataset on a maximum caption length basis, we end up with a dataset containing 347.228 instances, each one consisting of a single caption and its associated image.

Concerning the vocabulary, the resulting corpus contains 24.775 words, which is still a huge number, considering that many words only appear once. Finally, we decided to set the vocabulary size to 10,000. The words excluded from the vocabulary were assigned the "<unk>" token. The size of the vocabulary is another hyperparameter of the model (config.vocabulary_size).

[1]When I started this project there were no results yet but the ones published by Google. A month later there are already four additional results from two different teams

5.1.0.2 Validation dataset

The full validation dataset contains 40504 images and 5 captions per image. After applying the same filtering schema based on the maximum length of the captions, the dataset was reduced to 34043 images

Note that for the training process, each instance is made up of one single caption and its associated image, whilst for the evaluation process, each instance is an image and all its associated captions.

5.2 Experiments

5.2.1 Experimental setup

All the experiments reported in this chapter were performed on the MSCOCO dataset, with the additional constraints described above, which resulted in two separated datasets, one for the training, and another one for validations. Summing up:

- **Training**: 347,228 examples, each one consisting of an image-caption pair
- **Validation**: 34043 examples, each one consisting of an image, and all their captions (5 captions per image)

Caption length is limited to 15 significant tokens, plus the "<start>" and "<end>" token. Hardware setup:

- CPU: Intel Core i7-4790K (4 cores, 8 threads clocked @ 4 GHz)
- RAM: 16GB DIMM DDR3 2400 (clocked @ 1333 MHz)
- Data Storage: SSD Samsung 850 Evo 250GB
- GPU-0: NVIDIA GTX 1070, 6GB RAM
- GPU-1: NVIDIA RTX 2080 Ti, 11GB RAM

We will not give detailed timing results, because the computer was used interactively to perform other tasks while it was training, but we will make some comments on time aspects whenever there is something relevant to mention. For the moment, it suffices to say that in most cases, a full training epoch took between 40 and 50 minutes to complete.

5.2.2 Overview of the experiments

For the experiments we trained the same basic models but changing some hyperparameters:

- Type of ConvNet: 'VGG16', 'incept', 'Xception, 'ResNet50', ,'InceptionResNet_v2'
- Type of recurrent units: 'GRU' or 'LSTM'
- Number of recurrent units: 256, 512
- Embedding dimension: 256, 512
- Number of image features: 256, 512, 1024
- Batch size: 64, 128
- Dropout percentage: 0, 0.3, 0.5

Unfortunately, due to the long time required to train each model, we were unable to systematically try all the combinations of the chosen hyperparameters.

We trained all the models for 30 epochs. In one case we trained the model for 90 epochs, but the results clearly indicated that the model was overfitting. Therefore, we conducted some additional experiments to analyze the problem, and finally, we came up adding some Dropout layers as a form of regularization. All these experiments will be analyzed below.

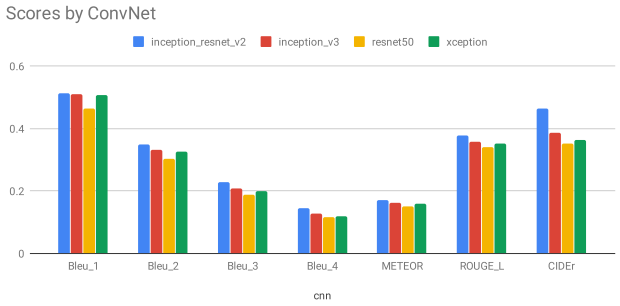
To have a sense of the kind of experiments performed, Table 5.1 provides a summary of the main block of experiments performed. In these experiments, we did not apply regularization and the training lasted 30 epochs. CNN and RNN are the types of ConvNet and RecurrentNet respectively, Emb refers to the size of the embedding vectors, Feat is the number of visual features used as input, Batch refers to the batch size, Optimi to the optimizer function, B1-B4 are Ble scores, MET refers to Meteor, R refers to ROUGE, and C to CIDEr. Results in this table are sorted by CIDEr score.

5.2.2.1 Effect of hyperparameters on performance

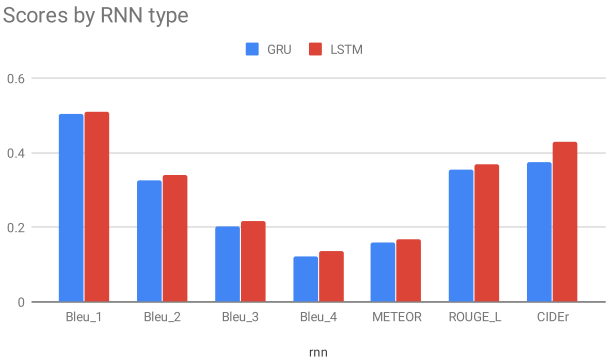
In order to study the effect of the different hyperparameters of the model on the performance, we have grouped and average the results for different combinations of parameters. Fig. 5.2a and Fig. 5.2b compares the performance depending on different types of network, for the encoder and decoder respectively. The other figures compare the performance of the model depending on some numerical hyperparameters, such as the batch size, the number of hidden units, the embedding size, or the number of image features that are generated by the encoder.

Table 5.1: Evaluation results. Model trained for 30 epochs. No regularization applied.

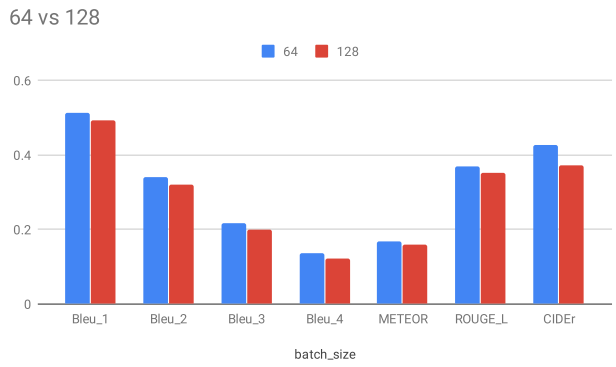
CNN	RNN	Emb	Hid	Feat	Batch	Optim	B1	B2	B3	B4	MET	ROU	CID
incep-res	lstm	512	512	512	64	Adam	0.485	0.320	0.203	0.127	0.162	0.357	0.413
inception	lstm	256	512	256	128	Adam	0.510	0.337	0.213	0.131	0.164	0.364	0.410
inception	lstm	256	512	256	64	Adam	0.521	0.342	0.215	0.132	0.166	0.367	0.406
inception	lstm	512	512	256	64	Adam	0.532	0.346	0.215	0.132	0.170	0.365	0.394
inception	gru	512	512	512	64	Adam	0.509	0.330	0.205	0.125	0.162	0.357	0.390
inception	gru	256	512	256	64	Adam	0.503	0.329	0.206	0.127	0.160	0.357	0.388
inception	gru	256	512	256	64	Nadam	0.515	0.336	0.209	0.127	0.163	0.360	0.380
inception	lstm	256	512	1024	128	Adam	0.509	0.327	0.201	0.121	0.162	0.355	0.370
xception	gru	256	512	256	64	Adam	0.507	0.326	0.199	0.120	0.160	0.352	0.365
inception	gru	512	1024	512	128	Adam	0.490	0.314	0.192	0.116	0.155	0.347	0.355
resnet	lstm	256	512	512	128	Adam	0.463	0.302	0.188	0.115	0.152	0.342	0.351



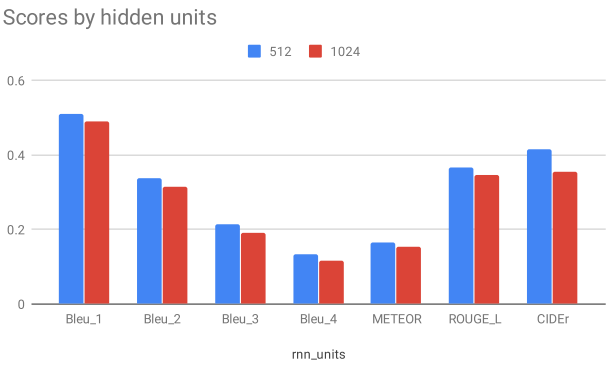
(a) Scores by type of ConvNet



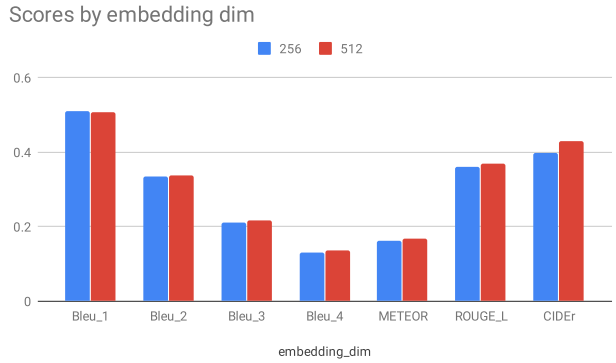
(b) Scores by type of RNN unit



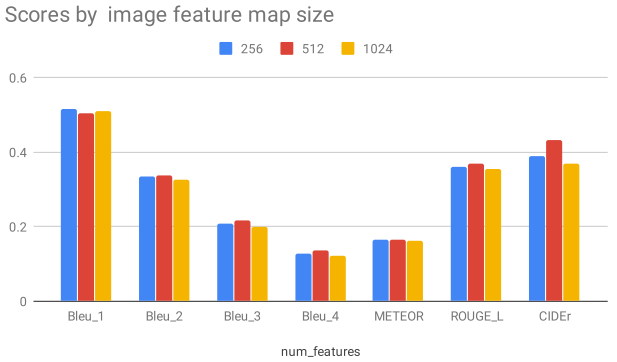
(a) Scores by batch size



(b) Scores by number of recurrent units



(a) Scores by embedding dimension



(b) Scores by number of image features

By looking at the former graphs, and at the correlation matrix in Fig. 5.5, we can see the following relations between the hyperparameters and the scores measured over the validation dataset.

- In general, there is a high degree of correlation between the different metrics except for CIDEr, that shows the least correlation with the other metrics.

- In all the numerical hyperparameters we observe a moderately negative correlation with the scoring metrics. That is, the higher the value of the hyperparameter the worse scoring obtained. Why is that? In general, higher values in these parameters (ie. more units, larger batches) make the model more complex, which in turn increases the bias. Therefore, by having set up a fixed number of training epochs, this observation is consistent with overfitting. That is, since the model is overfitting, increasing the complexity of the model can only aggravate the problem.

5.2.2.2 Training and overfitting

Overfitting occurs when a model performs very well with the training dataset, but not so well with other data, that is, the model does not generalize well. Overfitting can be discovered by comparing the performance over the training dataset and the validation dataset. Usually, the same loss function is used in both cases and it is pretty easy to find out if a model is overfitting. In the image captioning problem and other problems in NLP, this is not so easy because the prediction stage is different from the training stage (recall the teacher forcing technique used to generate captions in our model). Furthermore, language tasks use sophisticated metrics to estimate the quality of the model. In these cases, overfitting can be discovered by studying the dynamics of the system. Typically, we will see a stall o even a deterioration in the performance

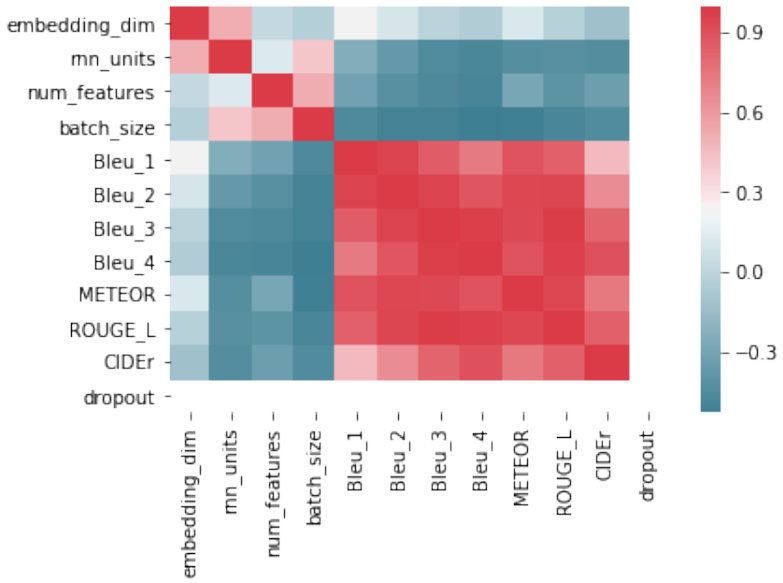


Figure 5.5: Correlation matrix for the numeric hyperparameters and the performance metrics

although the training loss continues decreasing. Ideally, we would interleave training and validation to detect this kind of problems as soon as possible. By doing this we can stop the training to avoid wasting time and keep a copy of the model when it was performing the better. This technique is called *early stopping*, and can be seen as a specific form of regularization.

We did not include an automatic validation mechanism interleaved with the training process; however, we did perform various analysis of the dynamics of the model by keeping checkpoints, that is, by saving the model at different moments along the training process, and evaluating multiple models afterward. An example of such an analysis can be seen in Fig. 5.6. Clearly, there is a point, around the 6th epoch, when the metric scores worsen, while the training loss is still decreasing. This behavior is a clear sign of overfitting.

There are various approaches to avoid or diminish overfitting. Two of the most popular techniques applied in deep learning are dropout and batch normalization.

5.2.2.3 Dropout regularization

In the last stage of the project we did manage to include some degree of regularization by introducing dropout layers just before dense layers in the different components of the model: the encoder, the decoder, and also the attention mechanism. We did not have to time for a systematic analysis, but preliminary results showed that this solution delayed the overfitting for several epochs, although in the end there was overfitting. We expect to delve deeper into this technique in the future to improve our results. Also, it would be interesting to try other

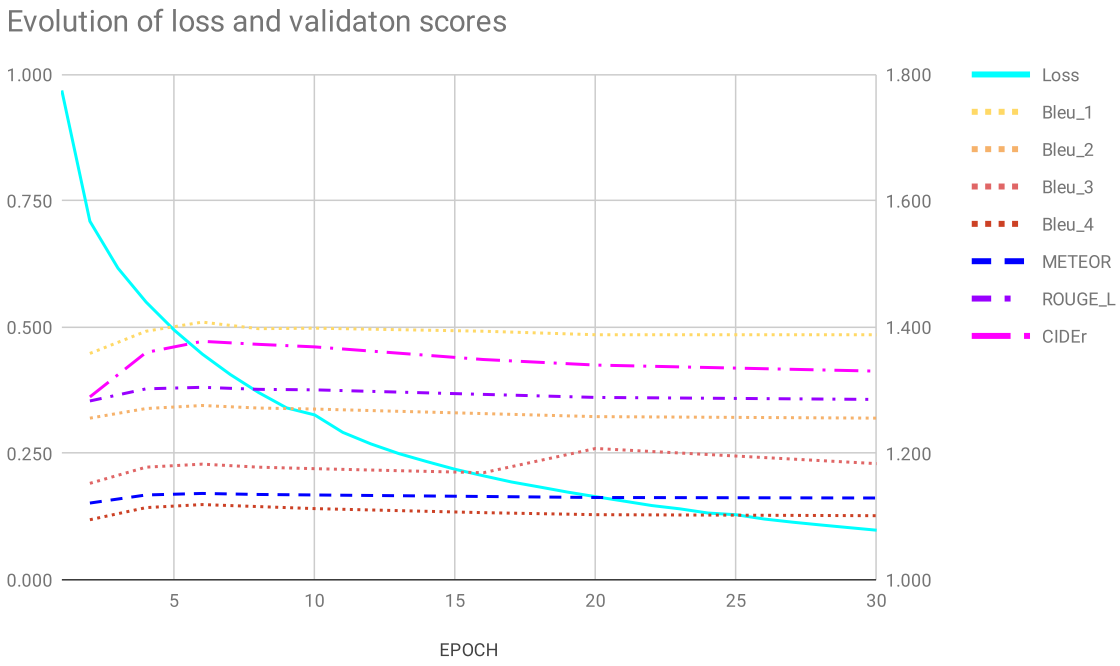


Figure 5.6: Evolution of loss and performance as training progresses

mechanisms such as batch normalization.

As illustration of the effect obtained by dropout, compare Table 5.2 and Table 5.3. These tables show results of the same model for 10 epochs, showing the metric scores of the validation every 2 epochs. However, the first one does not include dropout, and the second one includes a dropout of 0.3 (probability of letting a unit out during training)

Table 5.2: Some results of training and validation without normalization

Epochs	Loss	Bleu1	Bleu2	Bleu3	Bleu4	METEOR	ROUGE	CIDEr
2	1.56763	0.448	0.302	0.191	0.119	0.152	0.354	0.362
4	1.439718	0.492	0.339	0.223	0.143	0.168	0.378	0.45
6	1.357404	0.501	0.345	0.229	0.149	0.171	0.381	0.472
8	1.296668	0.497	0.34	0.223	0.145	0.169	0.377	0.466
10	1.261107	0.498	0.338	0.22	0.141	0.168	0.376	0.461

5.2.3 Demonstrative examples

We conclude this section by showing some examples of the captions generated by our model using some examples from the validation dataset.

Table 5.3: Some results of training and validation with dropout = 0.3

Epochs	Loss	Bleu1	Bleu2	Bleu3	Bleu4	METEOR	ROUGE	CIDEr
2	1.562302	0.466	0.324	0.214	0.137	0.161	0.369	0.434
4	1.431397	0.496	0.346	0.231	0.15	0.171	0.381	0.478
6	1.350335	0.511	0.357	0.239	0.154	0.175	0.387	0.495
8	1.285521	0.522	0.363	0.24	0.154	0.177	0.387	0.494
10	1.237422	0.517	0.356	0.235	0.15	0.175	0.383	0.484

As you can see, in general the system has been able to detect some of the main elements of the picture, although not all of them. Many times some elements of the image are not included in the caption, or there are errors. Sometimes, there is a clear error, like not detecting the truck in Fig. 5.11, or error in the quantity of an important element of the image, such as the number of people in Fig. 5.9.

Concerning the structure of the generated captions, the majority of them are grammatically correct, and most of the problems identified are rather semantic, like the double mention to "chocolate" in Fig. 5.14.

Figure 5.7: Ground Truth: A man standing on a tennis court holding a tennis racquet.

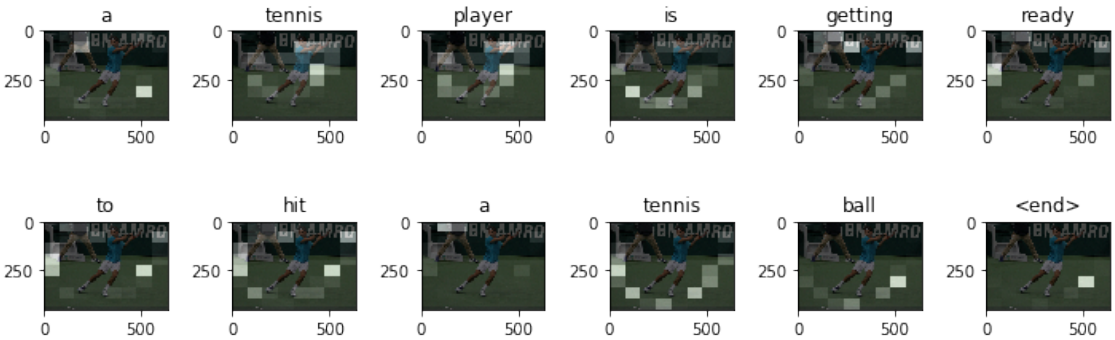


Figure 5.8: Predicted: a tennis player is getting ready to hit a tennis ball.

Figure 5.9: Ground Truth: Two males and a female walking in the snow carrying a ski sled and snowboard.

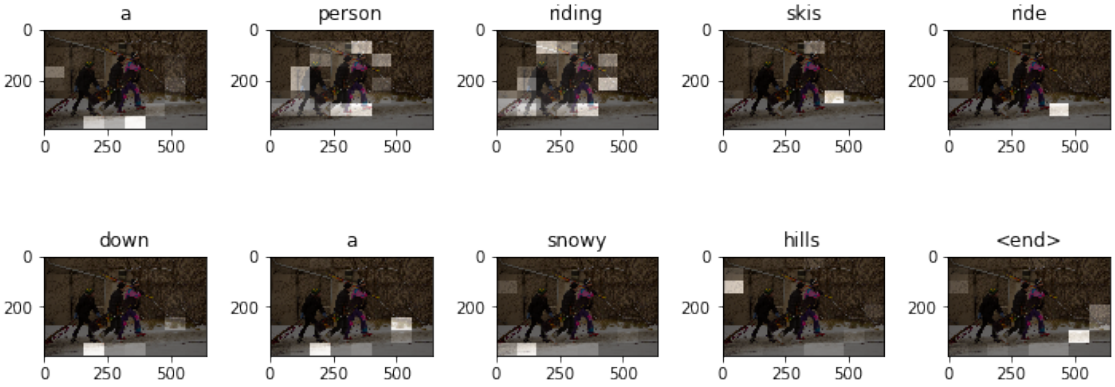


Figure 5.10: Predicted: a person riding skis ride down a snowy hills.

Figure 5.11: Ground Truth: A man sitting on the back of a red truck.

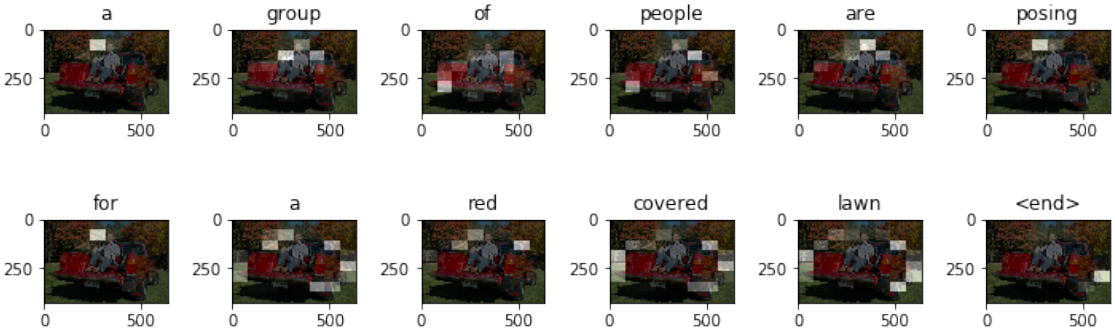


Figure 5.12: Predicted: a group of people are posing for a red covered lawn.

Figure 5.13: Ground Truth:a man is holding a piece of food with chocolate in it.

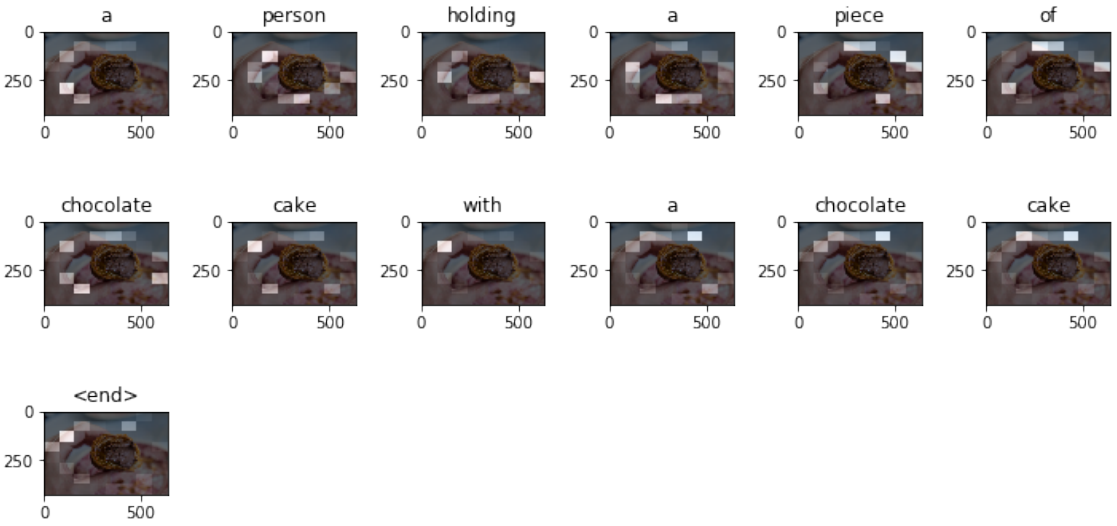


Figure 5.14: Predicted: A person holding a piece of chocolate cake with a chocolate cake.

Chapter 6

Conclusions

This chapter presents a critical self-opinion of the project and collects a number of potential improvements and future work.

6.1 What I learned

This project has supposed a huge advancement in my understanding of deep learning, especially in the fields of Natural Language Processing (NLP) and Computer Vision (CV).

To begin with, I had to review a lot of literature on the topics of the project. Furthermore, being a project that combines NLP and CV, it covered a vast number of concepts, models and technologies. The initial stages of the project, where I was supposed to collect information on the project and review the state of the art, easily exceeded the expected effort by twice or tribble at least. One paper led to another paper, one model was soon discovered to have been surpassed by other models the following year, and so on. In the end, I expend more than a month actively researching the subjects, plus some extra days collecting and summarizing benchmark results from several, some times contradictory sources.

Secondly, I combined my research-oriented efforts with an intense self-learning program which basically took me through the 5 courses of the Deep Learning specialization program by *deeplearning.ai* and professor Andrew Ng. at Coursera. Those courses gave me a wide and solid knowledge of deep learning, NLP and CV, combining both theory knowledge with hands-on experience, although the pace was probably too fast as to get a deep understanding of all the content: techniques to develop and improve deep networks, ConvNets, sequence, and recurrent networks, etc. There were many practical exercises, which were done using Python and Tensorflow on Jupyter Notebook. The case studies ranged from healthcare and autonomous driving to music generation, and natural language processing. All this knowledge was probably too much to digest in barely two months, but I still got a grasp of many of the concepts and

techniques described in the courses, which was of great help to understand the research papers and model implementations I had to review as part of the project.

Third, I had to put into practice many of the concepts and techniques learned. These included fully connected layers, which I was already familiar with, but also ConvNets, RNNs, language modeling, seq2seq models, attention mechanisms, beam search, and so many other, most of which were new to me. In the beginning, started implementing quick prototypes as a proof of concept, to make sure my hardware had enough computational power to handle a big dataset, as was my intention. To that avail, I started prototyping by means of the Jupyter Notebook platform. Preliminary results made me buy a more powerful GPU, as described in §3.1. Soon, I jumped from the prototyping environment into a fully-fledged development environment, which included the following tools and technologies:

- Linux operating system, with CUDA drivers for GPU support *Visual Studio Code* editor
- *Git* control version and a remote *GitLab* repository
- *Conda* and *Pip* for package management
- Tensorflow 2.0.0-alpha with GPU support, and Keras API.

6.1.1 What I would have changed

The development of a working system was an intense and very exciting experience too, but the entire development effort was highly conditioned from the very beginning by an excruciating lack of time. As a matter of fact, the analysis of the problem and the review of the literature consumed too much of my time, and thereafter I felt severely pushed by this lack of time, which made me commit several mistakes.

- First and probably the most severe mistake was developing and debugging the system using a big dataset. Once I realize I needed a smaller dataset for the initial development stage, I limited the number of examples actually used from the dataset, but as soon as I obtained an apparently working system I moved into the full dataset. By doing this, I had to expend a lot of time waiting for results rather than tweaking and improving the model.
- A second mistake that aggravated the former was the lack of a proper evaluation mechanism, which was delayed until the latest stages of the development process, when I included a tool to compute the usual metrics (Bleu, Meteor, etc.). Even worse, I didn't take the effort to interleave validation and training, that is, to check the generalization

ability of the network as the training progressed. Too late I realized that the model was tremendously overfitting, as shown in the experimental section (Chapter 5.

The combination of the two mistakes above made me waste a lot of time and computing resources overfitting the model far beyond the sweet spot. I couldn't imagine how soon overfitting started, but it was too late. However, this is a lesson I will never forget.

6.1.2 What I did not expect

In general, the different experiments with the model performed considerably worse than expected considering the results achieved by the model it was inspired by, the one described in the "Show, attend and tell" paper by Xu et al. (2015). However, but I have yet to find out whether this is due to implementation details, the lack of proper fine-tuning of the parameters, or a combination of the former.

The most frustrating result was due to the bad performance obtained by the beam search algorithm. It took several hours to develop a working *beamsearch* algorithm. However, it was incredibly slow, and yet it obtained worse results than using greedy search, which was utterly unexpected. I should have to investigate this in the future since I am really disappointed.

After conducting a number of experiments with attention and still not getting as good results as expected, I decided to compare the model with attention against the same model without attention. For that purpose, I modified the existing model and conducted various experiments, but something was probably wrong because I couldn't obtain any reasonable results to compare against, but I did not have more time to debug it.

6.2 Future work

The limited time prevented me from adopting a more systematic and thorough approach to the development of a solution. In this final section I will enumerate some of the many things that could be done to complete, improve or extend the developed system, as well as alternative approaches to the image captioning problems, which include published work as well as some ideas that come to my mind based on what I have learned during this project.

First of all, some validation should be included as an integral part of the training process to monitor it and detect overfitting, perhaps combined with *early stopping* as a form of regularization.

Second, given the results obtained, overfitting arises as one of the major problems with the model developed. Unfortunately, I realized very late the need for regularization mechanisms,

such as *batch normalization* and *dropout*. This would be the most important improvements to undertake in the future, combined with the use of validation during the training procedure.

Third, it would be convenient to conduct more experiments, and in a more principled way, so as to have a proper distribution of values for the most important hyperparameters. It would be also very helpful to perform most of these experiments on a smaller dataset, such as the Flickr8K or the Flickr30K datasets, to get a more comprehensive understanding on the effects of the various hyperparameters and how to tune them before moving into larger datasets.

Fourth, it would be nice to have a baseline to compare against, such as the same basic model without the attention mechanism. Actually, I tried to do this but something went wrong and I did not have time enough to debug it.

Finally, in addition to monitoring validation while training, it would also be very interesting to include some form of visualization, probably by means of the *Tensorboard* visualization tool. This is something I did not have the opportunity to explore, but I would definitely add this as something to address in the future.

Besides the proposed improvements, there are a number of things that could be done in the future to extend the work done for this project.

- To begin with, it would be nice to develop alternative models of attention to compare with, such as the global attention mechanism proposed by Luong et al. (2015) or a hierarchical attention mechanism as the one proposed by Khademi and Schulte (2018).

- I'm really curious about the possibility of adding reinforcement learning to improve the captioning process, in the likes of the model proposed by Rennie et al. (2017).

- Finally, aside from other forms of attention and alternative methods to generate captions, it would be nice to build an online application for demonstrative purposes. Since our model was developed in Tensorflow, it would be easy to load the model into a Web App by means of the Tensorflow.js library.

6.3 Concluding remarks

All in all, retrospectively, I am very happy with having chosen this problem for my Master Thesis. It is a very challenging problem, and it was very exciting as a research problem. As a matter of fact, I ended up dedicating far more time than planned surveying the state of the art, but I did not regret this at all; it was quite inspiring to see so many advances in such a brief period of time.

However, I felt like I would have needed far more time to develop a more satisfactory system. In the end, the results were not up to my expectations, and I am looking forward to improving and extending it in my free time in the near future.

The code for the project

Bibliography

Anderson, P., Fernando, B., Johnson, M., and Gould, S. (2016). SPICE: Semantic Propositional Image Caption Evaluation. In *4th European Conference on Computer Vision (ECCV'16)*, pages 382–398. Springer.

Anderson, P., Gould, S., and Johnson, M. (2018a). Partially-Supervised Image Captioning. In *32nd Conference on Neural Information Processing Systems (NeurIPS 2018)*, Montreal.

Anderson, P., He, X., Buehler, C., Teney, D., Johnson, M., Gould, S., and Zhang, L. (2018b). Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6077–6086, Salt Lake City, USA. IEEE.

Aneja, J., Deshpande, A., and Schwing, A. G. (2018). Convolutional Image Captioning. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5561–5570, Salt Lake City, USA. IEEE.

Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Zitnick, C. L., and Parikh, D. (2015). VQA: Visual Question Answering. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2425–2433. IEEE.

Bach, F. and Jordan, M. (2003). Kernel independent component analysis. In *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'03)*, volume 4, pages IV–876–9, Hong Kong, China. IEEE.

Badrinarayanan, V., Kendall, A., and Cipolla, R. (2017). SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2481–2495.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural Machine Translation by Jointly Learning to Align and Translate. In *2015 International Conference on Learning Representations (ICLR)*, San Diego, USA.

Bai, S. and An, S. (2018). A survey on automatic image caption generation. *Neurocomputing*, 311:291–304.

Banerjee, S. and Lavie, A. (2005). 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*, pages 65–72, Ann Arbor, USA. Association for Computational Linguistics.

Barto, A. G., Sutton, R. S., and Anderson, C. W. (1983). Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(5):834–846.

Berg, T. L., Berg, A. C., and Shih, J. (2010). Automatic Attribute Discovery and Characterization from Noisy Web Data. In *Proceedings of the 9th European Conference on Computer Vision (ECCV)*.

Berger, A. L., Pietra, S. A. D., and Pietra, V. J. D. (1996). A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, 22(1):39–71.

Bernardi, R., Çakici, R., Elliott, D., Erdem, A., Erdem, E., Ikizler-Cinbis, N., Keller, F., Muscat, A., and Plank, B. (2017). Automatic description generation from images: A survey of models, datasets, and evaluation measures. *IJCAI International Joint Conference on Artificial Intelligence*, 55:4970–4974.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022.

Britz, D., Goldie, A., Luong, M.-T., and Le, Q. (2017). Massive Exploration of Neural Machine Translation Architectures. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1442–1451, Stroudsburg, PA, USA. Association for Computational Linguistics.

Buschman, T. J. and Miller, E. K. (2007). Top-Down Versus Bottom-Up Control of Attention in the Prefrontal and Posterior Parietal Cortices. *Science*, 315(5820):1860–1862.

Bychkovsky, V., Paris, S., Chan, E., and Durand, F. (2011). Learning photographic global tonal adjustment with a database of input / output image pairs. In *CVPR 2011*, pages 97–104. IEEE.

Callison-Burch, C., Osborne, M., and Koehn, P. (2006). Re-evaluating the Role of BLEU in Machine Translation Research. In *Proceedings of 11th Conference of the European Chapter of the Association for Computational Linguistics*, Trento, Italy.

Chen, L., Zhang, H., Xiao, J., Nie, L., Shao, J., Liu, W., and Chua, T.-S. (2017a). SCA-CNN: Spatial and Channel-Wise Attention in Convolutional Networks for Image Captioning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6298–6306, Honolulu, USA. IEEE.

Chen, M., Ding, G., Zhao, S., Chen, H., Han, J., and Liu, Q. (2017b). Reference Based LSTM for Image Captioning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*, San Francisco, USA. AAAI Press.

Chen, X. and Zitnick, C. L. (2015). Mind's eye: A recurrent visual representation for image caption generation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2422–2431. IEEE.

Cheng, J., Dong, L., and Lapata, M. (2016). Long Short-Term Memory-Networks for Machine Reading. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 551–561, Stroudsburg, PA, USA. Association for Computational Linguistics.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods on Natural Language Processing (EMNLP)*, pages 1724–1734.

Chollet, F. (2017). Xception: Deep Learning with Depthwise Separable Convolutions. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, USA. IEEE.

Chung, J., Gülçehre, Ç., Cho, K., and Bengio, Y. (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. In *NIPS 2014 Deep Learning and Representation Learning Workshop*, Montreal, Canada.

Collobert, R. and Weston, J. (2008). A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. In *25th International Conference on Machine Learning (ICML'08)*, Helsinki, Finland.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. P. (2011). Natural Language Processing (almost) from Scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Dai, B., Fidler, S., Urtasun, R., and Lin, D. (2017). Towards Diverse and Natural Image Descriptions via a Conditional GAN. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2989–2998, Venice, Italy. IEEE.

De Marneffe, M.-C., MacCartney, B., and Manning, C. D. (2006). Generating Typed Dependency Parses from Phrase Structure Parses. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC)*.

Denkowski, M. and Lavie, A. (2014). Meteor Universal: Language Specific Translation Evaluation for Any Target Language. In *ACL 2014 Ninth Workshop on Statistical Machine Translation*, Baltimore, USA. Association for Computational Linguistics.

Devlin, J., Cheng, H., Fang, H., Gupta, S., Deng, L., He, X., Zweig, G., and Mitchell, M. (2015). Language Models for Image Captioning: The Quirks and What Works. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 100–105, Stroudsburg, PA, USA. Association for Computational Linguistics.

Donahue, J., Hendricks, L. A., Guadarrama, S., Rohrbach, M., Venugopalan, S., Darrell, T., and Saenko, K. (2015). Long-term recurrent convolutional networks for visual recognition and description. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2625–2634. IEEE.

Elliott, D. and de Vries, A. (2015). Describing Images using Inferred Visual Dependency Representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 42–52, Stroudsburg, PA, USA. Association for Computational Linguistics.

Elliott, D. and Keller, F. (2013). Image Description using Visual Dependency Representations. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*, pages 1292–1302, Seatle, USA. Association for Computational Linguistics.

Escalante, H. J., Hernández, C. A., Gonzalez, J. A., López-López, A., Montes, M., Morales, E. F., Enrique Sucar, L., Villaseñor, L., and Grubinger, M. (2010). The segmented and annotated IAPR TC-12 benchmark. *Computer Vision and Image Understanding*, 114(4):419–428.

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2010). The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 88(2):303–338.

Fang, H., Gupta, S., Iandola, F., Srivastava, R. K., Deng, L., Dollar, P., Gao, J., He, X., Mitchell, M., Platt, J. C., Zitnick, C. L., and Zweig, G. (2015). From captions to visual concepts and back. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1473–1482. IEEE.

Farhadi, A., Hejrati, M., Sadeghi, M. A., Young, P., Rashtchian, C., Hockenmaier, J., and Forsyth, D. (2010). Every Picture Tells a Story: Generating Sentences from Images. In *Proceedings of the 11th European Conference on Computer Vision (ECCV)*, volume 6314 LNCS, pages 15–29, Heraklion, Greece. Elsevier.

Feng, Y., Ma, L., Liu, W., and Luo, J. (2018). Unsupervised Image Captioning.

Ferraro, F., Mostafazadeh, N., Huang, T.-H., Vanderwende, L., Devlin, J., Galley, M., and Mitchell, M. (2015). A Survey of Current Datasets for Vision and Language Research. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 207–2013, Lisbon, Portugal. Association for Computational Linguistics.

Frome, A., Corrado, G. S., Shlens, J., Bengio Jeffrey Dean, S., Ranzato, A., and Mikolov, T. (2013). DeViSE: A Deep Visual-Semantic Embedding Model. In *Advances in Neural Information Processing Systems 26 (NIPS 2013)*.

Fu, K., Jin, J., Cui, R., Sha, F., and Zhang, C. (2017). Aligning Where to See and What to Tell: Image Captioning with Region-Based Attention and Scene-Specific Contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2321–2334.

Gan, C., Gan, Z., He, X., Gao, J., and Deng, L. (2017a). StyleNet: Generating Attractive Visual Captions with Styles. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 955–964. IEEE.

Gan, Z., Gan, C., He, X., Pu, Y., Tran, K., Gao, J., Carin, L., and Deng, L. (2017b). Semantic Compositional Networks for Visual Captioning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1141–1150. IEEE.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587. IEEE.

Gong, Y., Wang, L., Hodosh, M., Hockenmaier, J., and Lazebnik, S. (2014). Improving Image-Sentence Embeddings Using Large Weakly Annotated Photo Collections. In *Proceedings of the 13th European Conference on Computer Vision (ECCV)*, pages 529–545, Zurich, Switzerland. Springer.

Graves, A., Wayne, G., and Danihelka, I. (2014). Neural Turing Machines.

Gu, J., Cai, J., Wang, G., and Chen, T. (2018). Stack-Captioning: Coarse-to-Fine Learning for Image Captioning. In *The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, New Orleans, USA. AAAI Press.

Gu, J., Wang, G., Cai, J., and Chen, T. (2017). An Empirical Study of Language CNN for Image Captioning. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1231–1240, Venice, Italy. IEEE.

Gupta, A., Verma, Y., and Jawahar, C. V. (2012). Choosing linguistics over vision to describe images. In *Proceedings of the Twenty-Sixth (AAAI) Conference on Artificial Intelligence*, pages 606–612, Toronto, Ontario, Canada. AAAI Press.

Hao, Y., Xie, J., and Lin, Z. (2018). Image Caption via Visual Attention Switch on DenseNet. In *2018 International Conference on Network Infrastructure and Digital Content (IC-NIDC)*, pages 334–338. IEEE.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778. IEEE.

Hendricks, L. A., Venugopalan, S., Rohrbach, M., Mooney, R., Saenko, K., and Darrell, T. (2016). Deep Compositional Captioning: Describing Novel Object Categories without Paired Training Data. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–10. IEEE.

Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Hodosh, M. and Hockenmaier, J. (2013). Sentence-Based Image Description with Scalable, Explicit Models. In *2013 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 294–300. IEEE.

Hodosh, M., Young, P., and Hockenmaier, J. (2013). Framing Image Description as a Ranking Task: Data, Models and Evaluation Metrics. *Journal of Artificial Intelligence Research*, 47:853–899.

Hossain, M. Z., Sohel, F., Shiratuddin, M. F., Laga, H., Hossain, Z., Sohel, F., Shiratuddin, M. F., and Laga, H. (2019). A Comprehensive Survey of Deep Learning for Image Captioning. *ACM Computing Surveys*, 51(6):1–36.

Hu, B., Lu, Z., Li, H., and Chen, Q. (2014). Convolutional Neural Network Architectures for Matching Natural Language Sentences. In *Advances in Neural Information Processing Systems 27 (NIPS)*, Montreal, Canada.

Jang, E., Gu, S., and Poole, B. (2017). Categorical reparameterization with gumbel-softmax. In *5th International Conference on Learning Representations*, Toulon, France.

Jia, X., Gavves, E., Fernando, B., and Tuytelaars, T. (2015). Guiding the Long-Short Term Memory Model for Image Caption Generation. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2407–2415, Santiago, Chile. IEEE.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. (2014). Caffe: Convolutional Architecture for Fast Feature Embedding. In *Proceedings of the ACM International Conference on Multimedia*, pages 675–678, New York, New York, USA. ACM Press.

Jiang, W., Ma, L., Chen, X., Zhang, H., and Liu, W. (2018). Learning to Guide Decoding for Image Captioning. In *The 32nd AAAI Conference on Artificial Intelligence (AAAI-18)*, New Orleans, USA.

Johnson, J., Karpathy, A., and Fei-Fei, L. (2016). DenseCap: Fully Convolutional Localization Networks for Dense Captioning. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4565–4574, Las Vegas, NV, USA. IEEE.

Johnson, J., Krishna, R., Stark, M., Li, L.-J., Shamma, D. A., Bernstein, M. S., and Fei-Fei, L. (2015). Image retrieval using scene graphs. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3668–3678. IEEE.

Kalchbrenner, N. and Blunsom, P. (2013). Recurrent Continuous Translation Models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, Seattle, USA. Association for Computational Linguistics.

Kalchbrenner, N., Grefenstette, E., and Blunsom, P. (2014). A Convolutional Neural Network for Modelling Sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 655–665, Stroudsburg, PA, USA. Association for Computational Linguistics.

Karpathy, A. and Fei-Fei, L. (2015). Deep visual-semantic alignments for generating image descriptions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 39, pages 3128–3137, Boston, USA. IEEE.

Karpathy, A., Joulin, A., and Fei-Fei, L. (2014). Deep Fragment Embeddings for Bidirectional Image Sentence Mapping. In *Proceedings of the 27th International Conference on Neural Information Processing Systems (NIPS 2014)*, pages 1889–1897. MIT Press.

Khademi, M. and Schulte, O. (2018). Image Caption Generation with Hierarchical Contextual Visual Spatial Attention. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2024–20248, Salt Lake City, USA. IEEE.

Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kiros, R., Salakhutdinov, R., and Zemel, R. S. (2014a). Unifying Visual-Semantic Embeddings with Multimodal Neural Language Models. In *Advances in Neural Information Processing Systems Deep Learning Workshop*.

Kiros, R., Zemel, R., and Salakhutdinov, R. (2014b). Multimodal Neural Language Models. In *Proceedings of the 31st International Conference on Machine Learning (ICML 2014)*, volume 32, pages 595–603, Beijing, China. Proceedings of Machine Learning Research.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In *Proceedings of the 25 th International Conference on Neural Information Processing Systems*, pages 1097–1105, Lake Tahoe, USA.

Kulkarni, G., Premraj, V., Dhar, S., Li, S., Choi, Y., Berg, A. C., and Berg, T. L. (2011). Baby talk: Understanding and generating simple image descriptions. In *24th IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2011)*, volume 18, pages 1601–1608, Colorado Springs, USA. IEEE.

Kuznetsova, P., Ordonez, V., and Berg, A. (2012). Collective generation of natural image descriptions. *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, 1(July):359–368.

Kuznetsova, P., Ordonez, V., Berg, T. L., and Choi, Y. (2014). TreeTalk: Composition and Compression of Trees for Image Descriptions. *Transactions of the Association for Computational Linguistics*, 2:351–362.

Lamb, A., Goyal, A., Zhang, Y., Zhang, S., Courville, A., and Bengio, Y. (2016). Professor Forcing: A New Algorithm for Training Recurrent Networks. In *Thirtieth Conference on Neural Information Processing Systems (NIPS 2016)*, Barcelona, Spain.

Le, Q. V., Ranzato, M., Monga, R., Devin, M., Chen, K., Corrado, G. S., Dean, J., and Ng, A. Y. (2011). Building high-level features using large scale unsupervised learning. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8595–8598. IEEE.

Lebret, R., Pinheiro, P. O., and Collobert, R. (2015a). Phrase-based Image Captioning. In *Proceedings of the 32nd International Conference on Machine Learning*.

Lebret, R., Pinheiro, P. O., and Collobert, R. (2015b). Simple Image Description Generator via a Linear Phrase-Based Approach. In *Proceedings of the 2015 International Conference on Learning Representations*.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lei Ba, J., Mnih, V., Deepmind, G., and Kavukcuoglu, K. (2015). Multiple Object Recognition with Visual Attention. In *Proceedings of the 3rd International Conference on Learning Representations*, San Diego, USA.

Li, D., Huang, Q., He, X., Zhang, L., and Sun, M.-T. (2018). Generating Diverse and Accurate Visual Captions by Comparative Adversarial Learning. In *32nd Conference on Neural Information Processing Systems (NIPS 2018)*, Montreal, Canada.

Li, N. and Chen, Z. (2018). Image Captioning with Visual-Semantic LSTM. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18)*, Stockholm, Sweden.

Li, S., Kulkarni, G., Berg, T., Berg, A., and Choi, Y. (2011). Composing simple image descriptions using web-scale n-grams. In *Proceedings of the 15th Conference on Computational Natural Language Learning (CoNLL 2011)*, pages 220–228, Portland, USA. Association for Computational Linguistics.

Lin, C.-Y. (2004). ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out. Post-Conference Workshop at the 42nd Annual Meeting of the ACL*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Lin, C.-Y. and Och, F. J. (2004). Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics - ACL '04*, Barcelona, Spain. Association for Computational Linguistics.

Lin, D. (1998). An Information-Theoretic Definition of Similarity. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML 1998)*, pages 296–304, Madison, USA.

Lin, D., Fidler, S., Kong, C., and Urtasun, R. (2015). Generating Multi-sentence Natural Language Descriptions of Indoor Scenes. In *Procedings of the British Machine Vision Conference 2015*. British Machine Vision Association.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft COCO: Common Objects in Context. In *Proceedings of the 13th European Conference on Computer Vision (ECCV 2014)*, volume 8693 LNCS, pages 740–755. Springer.

Lindh, A., Ross, R. J., Mahalunkar, A., Salton, G., and Kelleher, J. D. (2018). Generating Diverse and Meaningful Captions. In *27th International Conference on Artificial Neural Networks (ICANN 2018)*, pages 176–187. Springer, Rhodes, Greece.

Ling, W., Dyer, C., Black, A. W., Trancoso, I., Fermandez, R., Amir, S., Marujo, L., and Luis, T. (2015). Finding Function in Form: Compositional Character Models for Open Vocabulary Word Representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1520–1530, Stroudsburg, PA, USA. Association for Computational Linguistics.

Liu, C., Mao, J., Sha, F., and Yuille, A. (2017a). Attention Correctness in Neural Image Captioning. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI-17)*.

Liu, C., Sun, F., Wang, C., Wang, F., and Yuille, A. (2017b). MAT: A Multimodal Attentive Translator for Image Captioning. In *26th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4033–4039, Melbourne, Australia. International Joint Conferences on Artificial Intelligence Organization.

Liu, S., Zhu, Z., Ye, N., Guadarrama, S., and Murphy, K. (2017c). Improved Image Captioning via Policy Gradient optimization of SPIDEr. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 873–881. IEEE.

Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440. IEEE.

Lowe, D. G. (2004). Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110.

Lu, J., Xiong, C., Parikh, D., and Socher, R. (2017). Knowing When to Look: Adaptive Attention via a Visual Sentinel for Image Captioning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3242–3250. IEEE.

Luong, T., Pham, H., and Manning, C. D. (2015). Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ma, L., Lu, Z., Shang, L., and Li, H. (2015). Multimodal Convolutional Neural Networks for Matching Image and Sentence. In *2015 IEEE International Conference on Computer Vision (ICCV)*, volume 17, pages 2623–2631. IEEE.

Ma, S. and Han, Y. (2016). Describing images by feeding LSTM with structural words. In *2016 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE.

Mao, J., Huang, J., Toshev, A., Camburu, O., Yuille, A., and Murphy, K. (2016). Generation and Comprehension of Unambiguous Object Descriptions. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11–20, Las Vegas, USA. IEEE.

Mao, J., Wei, X., Yang, Y., Wang, J., Huang, Z., and Yuille, A. L. (2015). Learning Like a Child: Fast Novel Visual Concept Learning from Sentence Descriptions of Images. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2533–2541. IEEE.

Mao, J., Xu, W., Yang, Y., Wang, J., and Yuille, A. L. (2014). Explain Images with Multimodal Recurrent Neural Networks. In *NIPS 2014 Deep Learning Workshop*.

Mao, J. and Yuille, A. (2015). Deep Captioning with Multimodal Recurrent Neural Networks (m-RNN). *2015 International Conference on Learning Representations (ICLR 2015)*, 1090(2014):1–17.

Mason, R. and Charniak, E. (2014). Nonparametric Method for Data-driven Image Captioning. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 592–598, Stroudsburg, PA, USA. Association for Computational Linguistics.

Mateos-Ortiz, L. G., Wolff, C., and Lapata, M. (2015). Learning to Interpret and Describe Abstract Scenes. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics Human Language Technologies*, pages 1505–1515, Stroudsburg, PA, USA. Association for Computational Linguistics.

Mathews, A., Xie, L., and He, X. (2016). SentiCap: Generating Image Descriptions with Sentiments. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, pages 3574–3580, Phoneix, Arizona.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26 (NIPS 2013)*, Lake Tahoe, USA.

Mitchell, M., Dodge, J., Goyal, A., Yamaguchi, K., Stratos, K., Han, X., Mensch, A., Berg, A. C., Berg, T. L., and Daume III, H. (2012). Midge: Generating Image Descriptions From Computer Vision Detections. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 747–756.

Mnih, A. and Hinton, G. (2007). Three new graphical models for statistical language modelling. In *Proceedings of the 24th international conference on Machine learning - ICML '07*, pages 641–648, New York, New York, USA. ACM Press.

Mnih, V., Heess, N., Graves, A., Kavukcuoglu, K., and Deepmind, G. (2014). Recurrent Models of Visual Attention. In *Advances in Neural Information Processing Systems 27 (NIPS 2014)*.

Mun, J., Cho, M., and Han, B. (2017). Text-guided Attention Model for Image Captioning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*, pages 4233–4239, San Francisco, USA.

Niu, Z., Zhou, M., Wang, L., Gao, X., and Hua, G. (2017). Hierarchical Multimodal LSTM for Dense Visual-Semantic Embedding. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1899–1907. IEEE.

Oliva, A. and Torralba, A. (2006). Building the gist of a scene: the role of global image features in recognition. In *Progress of Brain Research*, volume 155, pages 23–36. Elsevier B.V.

Ordonez, V., Kulkarni, G., and Berg, T. (2011). Im2text: Describing Images Using 1 Million Captioned Photographs. *Advances in Neural Information Processing Systems (NIPS)*, pages 1143–1151.

Oruganti, R. M., Sah, S., Pillai, S., and Ptucha, R. (2016). Image description through fusion based recurrent multi-modal learning. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3613–3617. IEEE.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of*

the Association for Computational Linguistics, page 311, Philadelphia, USA. Association for Computational Linguistics.

Park, C. C., Kim, B., and Kim, G. (2017). Attend to You: Personalized Image Captioning with Context Sequence Memory Networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6432–6440, Honolulu, USA. IEEE.

Patterson, G., Xu, C., Su, H., and Hays, J. (2014). The SUN Attribute Database: Beyond Categories for Deeper Scene Understanding. *International Journal of Computer Vision*, 108(1-2):59–81.

Pedersoli, M., Lucas, T., Schmid, C., and Verbeek, J. (2017). Areas of Attention for Image Captioning. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1251–1259, Venice, Italy. IEEE.

Plummer, B. A., Wang, L., Cervantes, C. M., Caicedo, J. C., Hockenmaier, J., and Lazebnik, S. (2015). Flickr30k Entities: Collecting Region-to-Phrase Correspondences for Richer Image-to-Sentence Models. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2641–2649. IEEE.

Pu, Y., Gan, Z., Henao, R., Yuan, X., Li, C., Stevens, A., and Carin, L. (2016a). Variational Autoencoder for Deep Learning of Images , Labels and Captions. In *NIPS'16 Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 2360–2368, Barcelona, Spain.

Pu, Y., Yuan, W., Stevens, A., Li, C., and Carin, L. (2016b). A Deep Generative Deconvolutional Image Model. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS 2016)*, pages 741–750, Cadiz, Spain.

Rashtchian, C., Young, P., Hodosh, M., and Hockenmaier, J. (2010). 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,*, pages 139–147, Los Angeles, USA.

Ratnaparkhi, A. (2000). Trainable Methods for Surface Natural Language Generation. *6th Applied Natural Language Processing Conference (ANLP)*, pages 194–201.

Reiter, E. and Belz, A. (2009). An Investigation into the Validity of Some Metrics for Automatically Evaluating Natural Language Generation Systems. *Computational Linguistics*, 35(4):529–558.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *Advances in Neural Information Processing Systems 28 (NIPS 2015)*, pages 91–99.

Ren, Z., Wang, X., Zhang, N., Lv, X., and Li, L.-J. (2017). Deep Reinforcement Learning-Based Image Captioning with Embedding Reward. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1151–1159, Honolulu, USA. IEEE.

Rennie, S. J., Marcheret, E., Mroueh, Y., Ross, J., and Goel, V. (2017). Self-Critical Sequence Training for Image Captioning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1179–1195, Honolulu, USA. IEEE.

Rensink, R. A. (2000). The Dynamic Representation of Scenes. *Visual Cognition*, 7(1-3):17–42.

Salton, G., Wong, A., and Yang, C. S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620.

Schuster, S., Krishna, R., Chang, A., Fei-Fei, L., and Manning, C. D. (2015). Generating Semantically Precise Scene Graphs from Textual Descriptions for Improved Image Retrieval. In *Proceedings of the Fourth Workshop on Vision and Language*, pages 70–80, Lisbon, Portugal. Association for Computational Linguistics.

Sennrich, R., Haddow, B., and Birch, A. (2016). Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Sharif, N., White, L., Bennamoun, M., and Shah, S. A. A. (2018). Learning-based Composite Metrics for Improved Caption Evaluation. In *Proceedings of ACL 2018 Student Research Workshop*, pages 14–20, Melbourne, Australia. Association for Computational Linguistics.

Sharma, P., Ding, N., Goodman, S., and Soricut, R. (2018). 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*, pages 2556–2565, Melbourne, Australia. Association for Computational Linguistics.

Shetty, R., Rohrbach, M., Hendricks, L. A., Fritz, M., and Schiele, B. (2017). Speaking the Same Language: Matching Machine to Human Captions by Adversarial Training. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 4155–4164, Venice, Italy. IEEE.

Silberman, N., Hoiem, D., Kohli, P., and Fergus, R. (2012). Indoor Segmentation and Support Inference from RGBD Images. In *Proceedings of the 12th European Conference on Computer Vision (ECCV)*, pages 746–760. Springer, Berline, Germany.

Simonyan, K. and Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, San Diego, USA.

Socher, R., Karpathy, A., Le, Q. V., Manning, C. D., and Ng, A. Y. (2014). Grounded Compositional Semantics for Finding and Describing Images with Sentences. *Transactions of the Association for Computational Linguistics*, 2:207–218.

Spratling, M. W. and Johnson, M. H. (2004). A Feedback Model of Visual Attention. *Journal of Cognitive Neuroscience*, 16(2):219–237.

Sugano, Y. and Bulling, A. (2016). Seeing with Humans: Gaze-Assisted Neural Image Captioning.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems 27 (NIPS 2014)*, pages 3104–3112.

Sutton, R. S., Mcallester, D., Singh, S., and Mansour, Y. (1999). Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *Proceedings of the 12th International Conference on Neural Information Processing Systems (NIPS'99)*, pages 1057–1063, Denver, USA. MIT Press.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the Inception Architecture for Computer Vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826. IEEE.

Tavakoliy, H. R., Shetty, R., Borji, A., and Laaksonen, J. (2017). Paying Attention to Descriptions Generated by Image Captioning Models. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2506–2515. IEEE.

Torralba, A., Fergus, R., and Freeman, W. (2008). 80 Million Tiny Images: A Large Data Set for Nonparametric Object and Scene Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(11):1958–1970.

Tran, K., He, X., Zhang, L., and Sun, J. (2016). Rich Image Captioning in the Wild. In *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 434–441. IEEE.

Uijlings, J. R. R., van de Sande, K. E. A., Gevers, T., and Smeulders, A. W. M. (2013). Selective Search for Object Recognition. *International Journal of Computer Vision*, 104(2):154–171.

Ushiku, Y., Yamaguchi, M., Mukuta, Y., and Harada, T. (2015). Common Subspace for Model and Similarity: Phrase Learning for Caption Generation from Images. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2668–2676. IEEE.

Vaswani, A., Brain, G., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention Is All You Need. In *31st Conference on Neural Information Processing Systems (NIPS 2017)*, Long Beach, USA.

Vedantam, R., Zitnick, C. L., and Parikh, D. (2015). CIDEr: Consensus-based image description evaluation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4566–4575. IEEE.

Venugopalan, S., Hendricks, L. A., Rohrbach, M., Mooney, R., Darrell, T., and Saenko, K. (2017). Captioning Images with Diverse Objects. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1170–1178. IEEE.

Verma, Y. and Jawahar, C. V. (2014). Im2Text and Text2Im: Associating Images and Texts for Cross-Modal Retrieval. In *Proceedings of the British Machine Vision Conference 2014*. British Machine Vision Association.

Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. (2015). Show and tell: A neural image caption generator. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 07-12-June, pages 3156–3164, Boston, USA. IEEE.

Viola, P., Platt, J. C., and Zhang, C. (2005). Multiple Instance Boosting for Object Detection. In *Advances in Neural Information Processing Systems 18 (NIPS 2005)*, pages 1419–1426, Vancouver, Canada. MIT Press.

Wang, M., Song, L., Yang, X., and Luo, C. (2016). A parallel-fusion RNN-LSTM architecture for image caption generation. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 4448–4452. IEEE.

Wang, Q. and Chan, A. B. (2018). CNN+CNN: Convolutional Decoders for Image Captioning. *CoRR*.

Wang, Y., Lin, Z., Shen, X., Cohen, S., and Cottrell, G. W. (2017). Skeleton Key: Image Captioning by Skeleton-Attribute Decomposition. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7378–7387. IEEE.

Williams, R. J. and Zipser, D. (1989). A Learning Algorithm for Continually Running Fully Recurrent Neural Networks. *Neural Computation*, 1(2):270–280.

Wu, Q., Shen, C., Liu, L., Dick, A., and van den Hengel, A. (2016). What Value Do Explicit High Level Concepts Have in Vision to Language Problems? In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 203–212. IEEE.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R., Zemel, R., and Bengio, Y. (2015). Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In *32nd International Conference on Machine Learning (ICML'15)*, volume 37, pages 2048–2057, Lille, France.

Yagcioglu, S., Erdem, E., Erdem, A., Cakici, R., and Çakici, R. (2015). A Distributed Representation Based Query Expansion Approach for Image Captioning. In *Proceedings of the e 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 106–111, Beijing, China. Association for Computational Linguistics.

Yan, F. and Mikolajczyk, K. (2015). Deep correlation for matching images and text. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 07-12-June, pages 3441–3450. IEEE.

Yang, L., Tang, K., Yang, J., and Li, L.-J. (2017). Dense Captioning with Joint Inference and Visual Context. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1978–1987, Honolulu, USA. IEEE.

Yang, Y., Teo, C., Daumé III, H., and Aloimonos, Y. (2011). Corpus-Guided Sentence Generation of Natural Images. In *The 2011 Conference on Empirical Methods in Natural Language Processing*, pages 444–454.

Yang, Z., Yuan, Y., Wu, Y., Salakhutdinov, R., and Cohen, W. W. (2016). Review Networks for Caption Generation. In *30th Annual Conference on Neural Information Processing Systems (NIPS 2016)*, pages 2016–2023, Barcelona, Spain. Curran Associates, Inc.

Yao, T., Pan, Y., Li, Y., and Mei, T. (2017a). Incorporating Copying Mechanism in Image Captioning for Learning Novel Objects. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5263–5271. IEEE.

Yao, T., Pan, Y., Li, Y., Qiu, Z., and Mei, T. (2017b). Boosting Image Captioning with Attributes. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 4904–4912. IEEE.

Yatskar, M., Galley, M., Vanderwende, L., and Zettlemoyer, L. (2014). See No Evil, Say No Evil: Description Generation from Densely Labeled Images. In *Proceedings of the 3rd Joint Conference on Lexical and Computational Semantics (SEM 2014)*, pages 110–120, Stroudsburg, PA, USA. Association for Computational Linguistics and Dublin City University.

You, Q., Jin, H., Wang, Z., Fang, C., and Luo, J. (2016). Image Captioning with Semantic Attention. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4651–4659. IEEE.

Young, M. H. P., Lai, A., and Hockenmaier, J. (2014). From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78.

Yu, L., Zhang, W., Wang, J., and Yu, Y. (2017). SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. In *The 31st AAAI Conference on Artificial Intelligence (AAAI 2017)*, San Francisco, USA.

Zeiler, M. D. and Fergus, R. (2014). Visualizing and Understanding Convolutional Networks. In *Proceedings of the 2014 European Conference on Computer Vision (ECCV)*, pages 818–833. Springer, Cham.

Zhang, L., Sung, F., Liu, F., Xiang, T., Gong, S., Yang, Y., and Hospedales, T. M. (2017). Actor-Critic Sequence Training for Image Captioning. In *31st Conference on Neural Information Processing Systems (NIPS 2017)*, Long Beach, USA.

Zhou, L., Xu, C., Koch, P., and Corso, J. J. (2017). Watch What You Just Said: Image Captioning with Text-Conditional Attention. In *Thematic Workshops of ACM Multimedia 2017*, pages 305–313, New York, USA. ACM Press.

Zitnick, C. L., Parikh, D., and Vanderwende, L. (2013). Learning the Visual Interpretation of Sentences. In *2013 IEEE International Conference on Computer Vision*, pages 1681–1688. IEEE.

Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. V. (2018). Learning Transferable Architectures for Scalable Image Recognition. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8697–8710. IEEE.