Master's thesis
Master's Programme in Data Science

Architectures of Leading Image Captioning Systems

Mikko Kotola

June 4, 2021

Supervisor(s):	Associate Professor Laura Ruotsalainen
Examiner(s):	Associate Professor Laura Ruotsalainen
	Dr. Lidia Pivovarova

University of Helsinki
Faculty of Science

P. O. Box 68 (Pietari Kalmin katu 5)
00014 University of Helsinki

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Koulutusohjelma — Utbildningsprogram — Degree programme
Faculty of Science	Master's Programme in Data Science

Tekijä — Författare — Author

Mikko Kotola

Työn nimi — Arbetets titel — Title

Architectures of Leading Image Captioning Systems

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidantal — Number of pages
Master's thesis	June 4, 2021	96

Tiivistelmä — Referat — Abstract

Image captioning is the task of generating a natural language description of an image. The task requires techniques from two research areas, computer vision and natural language generation.

This thesis investigates the architectures of leading image captioning systems. The research question is: What components and architectures are used in state-of-the-art image captioning systems and how could image captioning systems be further improved by utilizing improved components and architectures? Five openly reported leading image captioning systems are investigated in detail: Attention on Attention, the Meshed-Memory Transformer, the X-Linear Attention Network, the Show, Edit and Tell method, and Prophet Attention.

The investigated leading image captioners all rely on the same object detector, the Faster R-CNN based Bottom-Up object detection network. Four out of five also rely on the same backbone convolutional neural network, ResNet-101. Both the backbone and the object detector could be improved by using newer approaches. Best choice in CNN-based object detectors is the EfficientDet with an EfficientNet backbone. A completely transformer-based approach with a Vision Transformer backbone and a Detection Transformer object detector is a fast-developing alternative. The main area of variation between the leading image captioners is in the types of attention blocks used in the high-level image encoder, the type of natural language decoder and the connections between these components. The best architectures and attention approaches to implement these components are currently the Meshed-Memory Transformer and the bilinear pooling approach of the X-Linear Attention Network. Implementing the Prophet Attention approach of using the future words available in the supervised training phase to guide the decoder attention further improves performance.

Pretraining the backbone using large image datasets is essential to reach semantically correct object detections and object features. The feature richness and dense annotation of data is equally important in training the object detector.

ACM Computing Classification System (CCS):
Computing methodologies → Artificial intelligence → Computer vision → Computer vision problems → Object detection
Computing methodologies → Artificial intelligence → Natural language processing → Natural language generation
Computing methodologies → Artificial intelligence → Computer vision → Computer vision tasks → Scene understanding

Avainsanat — Nyckelord — Keywords

Image captioning, Object detection, Natural language generation, Deep learning

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Contents

1. Introduction

Image captioning is the task of generating a natural language description of an image. It is a high-level artificial intelligence task, which requires techniques from two central research areas, computer vision and natural language generation (NLG). Research articles typically describe and advance only one part of a captioning system, rather than the whole. But image captioning systems are truly a whole, taking an image as input and producing a natural language caption as output. To improve a system, it is possible to improve one component, keeping the other components and interfaces constant. But it is also possible to improve the architecture of the system: to add or remove components, or change the interfaces or interdependencies between components. This thesis examines the architectures of leading image captioning systems.

The architecture and components of image captioning systems consist of two stages: the object detector and the image captioner [23]. Within the object detector stage, there is typically a backbone convolutional neural network (CNN), and additional object detector layers that use CNN features as input. There is variation within which types of backbone CNNs are used, and also on how the higher-level object detection is achieved [40, 2, 23]. Some methods use a separate region proposal network (RPN) to identify regions of interest, and then further process those regions to output object detections. Other methods implement a single end-to-end-trainable network for the whole object detection process. Typically, the output of the object detector stage are feature vectors with information about detected objects. Within the image captioner stage, there is usually an encoder network (attention- or transformer-based [53]) that further learns context-rich high-level features about the whole image based on the object detection feature vectors. Immediately connected to the high-level encoder is a decoder network that outputs natural language tokens (words) time step by time step. The decoder is usually based on either a long short-term memory (LSTM) network or transformer blocks. The role of the language decoder is generating fluent language outputs that are grounded in the image information provided by the encoder. In addition to the decoder, some methods have additional components evaluating and editing the captions proposed by the decoder network [44].

The latest advances to image captioners have quite heavily focused on improving the high-level encoding of the image and related decoding of the latent representation into natural language. In particular, leading image captioners have kept the object detection part of the system constant. This thesis aims at clearly presenting all the components of leading image captioning systems and proposing ways to further improve them by i) evolving individual components ii) adding new components and by iii) evolving the interfaces and interdependencies between components. The research question is: **What components and architectures are used in state-of-the-art image captioning systems and how could image captioning systems be further improved by utilizing improved components and architectures?**

The investigation is limited to image captioning systems that are *public* and *in the top 100 of the leaderboard of the Microsoft COCO Image Captioning Challenge*[1] as of 31.3.2021. A public method is such a method that has been described in a research paper published in a peer-evaluated forum or on arXiv. The methods chosen for closer investigation are: Attention on Attention (#34 on COCO leaderboard), the Meshed-Memory Transformer (#21), the X-Linear Attention Network (#13), the Show, Edit and Tell method (#64), and Prophet Attention (#11). The chosen methods represent the state-of-the-art of captioning methods. More details on how the exact methods were chosen are given in chapter 4.

A central dataset for captioning is the COCO dataset [32], which includes 128 000 images of common objects in context. In addition, there is a withheld test set of images, which is used by an online server for evaluating proposed models. COCO is the standard used for evaluating image captioning models. COCO and other image captioning datasets are discussed in detail in section 2.2.

The thesis is a primarily theoretical investigation. The grounds for evaluating the practical performance of the models and architectures is the online results of the (continuously ongoing) COCO Image Captioning Challenge. One of the main contributions of this work is providing a clear presentation of current state-of-the-art image captioning systems as a whole, including both the object detector and the image captioner components. The architectures are visualized in original architecture diagrams presented in chapter 4. A description of the field of machine learning tasks related to image captioning and a visualization of image captioning as a task in relation to other tasks (figure 2.2) is presented. A visualization of the training phases of a captioning system (figure 4.9) is also presented. A central result of the thesis is a list of potential improvements to the state-of-the-art image captioning systems. Implementing and experimentally verifying the performance of the proposed improvements is not within the scope of this thesis, but could be taken up as follow-up work. Training procedures

[1]See Microsoft COCO Image Captioning Challenge, https://competitions.codalab.org/competitions/3221.

and phases, loss functions, and required choices of hyperparameters for components are discussed only at a high level, in relation to the architecture of the system. The focus of this work is not on exact values of hyperparameters and low-level design choices such as exact non-linear activation functions, optimizers, learning rates and batch sizes. Discussing low-level choices is a topic of its own for each component, and while arguably the performance of the whole system will benefit from the optimal low-level choices, the architecture of the system certainly sets the boundaries for how far performance can be improved by fine-tuning low-level details. These details are provided in the original papers of the methods under investigation, and can be looked up if needed.

This thesis covers a quite broad range of deep learning technologies, and also discusses the latest approaches in several areas. To keep the scope reasonable, I have chosen to not cover the basics of a range of standard deep learning techniques in detail – they are presumed to be background knowledge for people in the deep learning field. Such techniques include backpropagation, long short-term memory (LSTM) networks, convolutional neural networks (CNNs), the basic ResNet architecture, and the transformer module. The basics of these techniques can be looked up in open access materials such as the Deep Learning book [20] and other open web resources. Also knowledge of linear algebra is required to read the mathematical formulations. For a layman with no background in deep learning, chapters 1 and 2 give a good overview of image captioning as a machine learning task.

The main results of the thesis are as follows. The investigated leading image captioners all rely on the same object detector, the Faster R-CNN based Bottom-Up object detection network [2]. Four out of five also rely on the same backbone CNN, ResNet-101, with X-Linear Attention Network using SENet-154. Both the backbone and the object detector could be improved by using newer approaches that have been proven to outperform the outdated ones. Best choice in CNN-based object detectors is the EfficientDet [51] with an EfficientNet [50] backbone. A completely transformer-based approach with a Vision Transformer[15] backbone and a Detection Transformer [9] object detector could also be used. The main area of variation between the leading image captioners is in the types of attention blocks used in the high-level image encoder, the type of natural language decoder and the connections between these components. The best architectures and attention approaches to implement these components are currently the Meshed-Memory Transformer and the bilinear pooling approach of the X-Linear Attention Networks. Implementing the Prophet Attention approach of using the future words available in the supervised training phase to guide the decoder attention further improves performance.

The thesis is structured as follows. In chapter 2, I describe image captioning as a task, its practical applications and relation to other machine learning tasks, relevant datasets and automated evaluation metrics. In chapter 3, I describe the backbone and object detection components used in the the leading image captioners. Then in chapter 4, I describe each of the five image captioning systems chosen for investigation. Chapter 5 describes the latest innovations related to backbones, chapter 6 describes innovations and potential related to object detectors, and chapter 7 draws together innovations and improvements related to the high-level encoding and NLG decoding components, and also discusses the potential for introducing new components or evolving the interconnections of existing components. Chapter 8 draws together the main results and conclusions.

2. Image Captioning as a Task

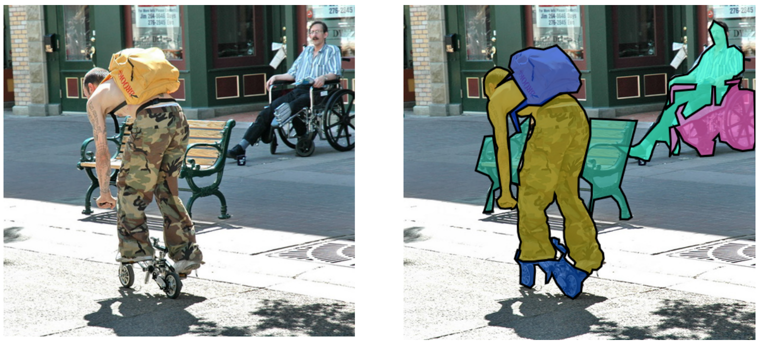
In this chapter, I first describe image captioning as a task, its practical applications and relation to other machine learning tasks. I then move on to describe datasets that are relevant to the image captioning task, especially the central COCO dataset. Finally, I describe the automated evaluation metrics used to evaluate the performance of image captioning systems, both generally and specifically in the COCO Image Captioning Challenge.

2.1 Image Captioning as a Semantic Understanding Task

Image captioning is the task of generating a natural language description of an image. Image captioning requires recognizing the important objects, their attributes, and their relationships in an image. The goal is to to generate syntactically and semantically correct sentences that describe what the image represents.

The concept of ground truth is not trivial in image captioning. The COCO training and validation dataset uses five ground truth captions per image. As can be witnessed in figure 2.1, the ground truth captions may include some objects present in the image but not include others. The humans that have written the ground truth captions seem to have had different viewpoints on what is central to this image. All five writers mention the man in the foreground and the vehicle he is riding. But there is already variance in interpreting, what the vehicle is: is it a bike or a scooter? There is variation in which attributes of the man are included: shirtlessness, camouflage pants (notice that "camouflage" is misspelled in the ground truth caption) and the backpack. Two captions mention the street as the place. One mentions the man sitting in the wheelchair. No caption mentions the bench, which is also present in the photo and even annotated as one of the present objects in the data set. This variance in ground truth captions adds to the difficulty level of the automated image captioning task.

Practical applications of image captioning include assisting the visually impaired by providing natural-language description of an image or scene as speech, content-based



Ground truth captions

- man in camaflogue pants riding a small scooter.
- a man riding on a miniature bike on a city street.
- a man with a back pack riding a very tiny bicycle.
- the man in the wheelchair watched the man on a small bike
- a shirtless man riding a mini bicycle on the street

Figure 2.1: Left: a sample COCO image. Center: ground-truth pixel-level segmentations of foreground objects for the same image. Right: the five ground truth captions for the image. Image and caption source: COCO, https://cocodataset.org/#explore?id=51052.

.

image indexing and retrieval [23], and visual intelligence in chatting robots [17].

In the computer vision context, image captioning belongs to a family of tasks known as *scene understanding*. Semantic understanding of visual scenes – scene understanding – is a high-level task area that covers multiple tasks [32, 58]: recognizing objects, localizing objects within the scene, determining attributes of the scene and the objects, characterizing relationships between objects and formulating a semantic description of the scene. Image captioning could be described as a central high-level task in the semantic understanding task family: it relies heavily on object detection and requires not only detecting object instances, but also learning the interdependencies of different object instances and the higher-level context of the image. A closely related task is *visual question answering* (VQA), where the input to the task is an image and a question ("What color is his tie?"), and the output is an answer ("Blue"). Scene understanding tasks can take as input 2-dimensional visual data (images), such as the COCO dataset [32], or 3-dimensional data, such as LiDAR point clouds in the SemanticKITTI dataset [5]. The 3-dimensional variants are important for machines that move and interact with the environment, such as self-driving cars. This thesis will concentrate on image captioning that uses 2-dimensional images as input.

A high-level diagram of image captioning and selected other machine learning tasks situated in the fields of computer vision and natural language generation is shown in figure 2.2. A group of scene understanding tasks that use images or video as input and give natural language as output is coloured green in the figure. These tasks are a part of both larger fields of research, drawing and combining ideas and approaches from both research communities. Object detection is a closely integrated task on the computer vision side, as image captioning systems, when looked at as complete systems taking images as input and giving natural language captions as output, actually include an object detection subsystem. The most closely related scene understanding task is stylistic image captioning, which extends the basic image captioning with different textual styles, for example generating a humorous description of an image instead of a neutral fact-describing caption. Visual question answering also depends on object detection, but changes the NLG part to accept natural language questions as input to the VQA-system. Video captioning is an extension of image captioning, adding the time dimension and paying attention to motion and events instead of only static image content.

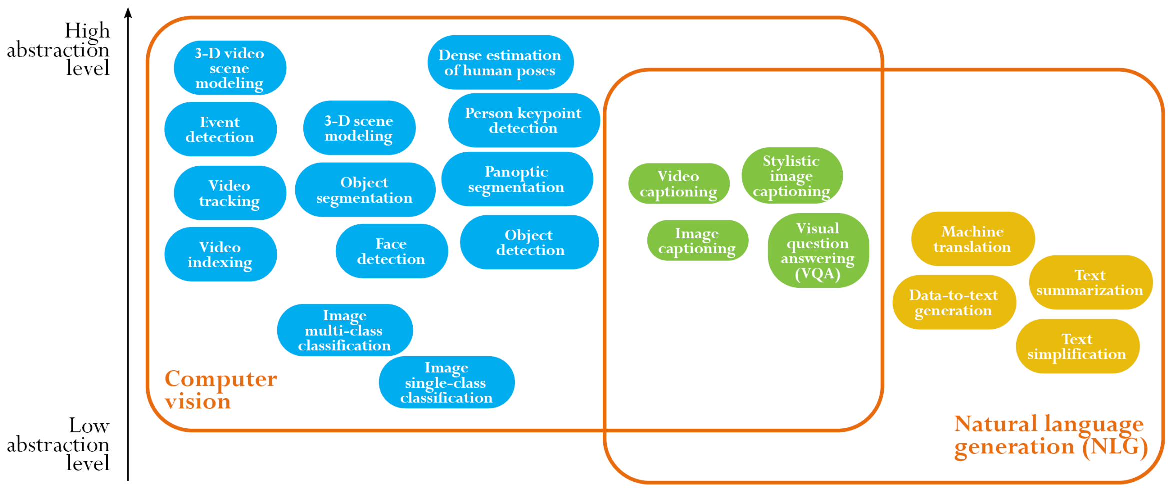


Figure 2.2: Image captioning and selected other machine learning tasks in the fields of computer vision and natural language generation. Abstraction level is a rough characterization.

Besides image captioning, another practical example of a use case for scene understanding is that of a household robot (example follows Zhou et al. [58]): A household robot equipped with a trained convolutional network can recognize that the visual scene is a living room. However, to be able to usefully interact with the environment, for example to pick up a dirty coffee cup from the top of a table and deliver it to the kitchen sink, the robot has to be able to recognize both things like *coffee cup* and *table* and stuff categories like *floor* and *wall* within the scene to be able to navigate in the scene and pick up a coffee cup. This task requires some level of semantic understanding of the scene.

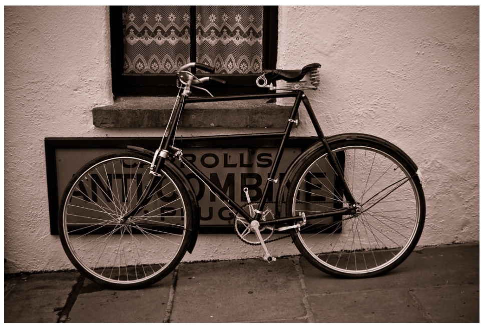


Figure 2.3: Bicycle in an iconic image. "Bicycle" by Conal Gallagher, CC BY 2.0.

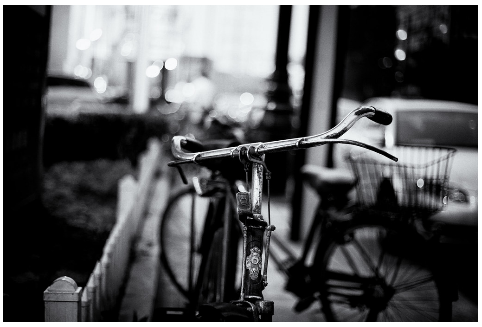


Figure 2.4: Bicycle from a non-canonical perspective. "Bicycle and Bokeh" by Bowen Chin, CC BY-NC-ND 2.0.

2.2 Image Captioning Datasets

In machine learning, datasets are often a central driving force in advancing state-of-the-art research. This is especially true for supervised learning, where a model learns to perform a task by using labeled samples in a dataset. In this section I will give an overview of selected datasets that are relevant to image captioning.

What is Required of an Image Captioning Dataset. Well-known datasets created for single-label or multi-label image classification tasks are not adequate for scene understanding tasks, including image captioning, for several reasons. One reason is that most image collections used for image classification tasks have images, that present objects as *iconic views* [32]. Bicycles are depicted as seen from the side, centered in the image, unobstructed, with two tyres with spokes visible. An example of an iconic view of a bicycle is in figure 2.3. A more demanding goal for machine learning, and necessary especially in scene understanding tasks, is to be able to recognize objects also when depicted using non-iconic images or from non-canonical perspectives: partly covered by other objects, among a group of other objects, or in the background. A example of a bicycle image from a non-canonical perspective and only partially shown is in figure 2.4.

A second reason for the inadequacy of image classification datasets is that object recognition has traditionally focused heavily on "thing" categories like *car*, *skateboard* and *elephant*. For scene understanding, however, the "stuff" categories, categories that do not have clear boundaries or parts, like *grass* or *sky*, are very central [8]. Caesar et al. [8] list several reasons for the central role of stuff. For one, stuff categories often define the type of a scene: a beach scene is a beach scene because there is water and sand present. Stuff also often captures the three-dimensional structure of the scene and limits the possible locations of the thing categories. It is rare for trees to appear in water or cats to appear in the sky. Lastly, stuff provides context for identifying things, especially small things. Smaller objects in images generally require more contextual reasoning to recognize [32]. If there is something small in the sky, it is likely to be a bird or an aeroplane far away. As the basic version of the COCO dataset used for image captioning does not include stuff categories, and the used object detectors explicitly focus on foreground object instances, the context provided by stuff categories and full panoptic segmentation is still largely an unexplored territory for image captioning. Lastly, as scene understanding requires a much more detailed segmentation, labeling and annotation of an image, the sheer amount of metadata about an image is much larger in a scene understanding dataset than in an image classification dataset. Where for image multi-labeling, the metadata could only include a list of category labels ("tree", "car", "human"), for scene understanding the metadata would include pixel-level masks or bounding boxes for different object instances, attributes about each object and scene captions written by humans. An example of the object instance masks and scene captions is shown in figure 2.5.

Even if image classification datasets are not adequate for object detection, they can be used as part of the training process, especially to train the backbone network. For this reason, both simpler but larger image classification datasets and smaller but richer scene understanding datasets are important for image captioning systems. The training processes will be discussed in chapters 3 to 6.

Overview of Datasets Relevant to Image Captioning. The datasets relevant to image captioning can be divided into three categories: image classification datasets, scene understanding datasets, and specialized datasets. Central image classification datasets that are presented first are ImageNet and JFT-300M. Scene understanding datasets include PASCAL VOC 2012, SUN, Flickr30k, COCO, ADE20K and Visual Genome. Finally, some relevant specialized datasets are shortly discussed: FlickrStyle10K, Personality-Captions and Localized Narratives.

ImageNet [14, 43][1] is the largest of the public datasets with over 14 million images and about 22 000 object categories, more precisely stated *synonym sets* (*synsets*). ImageNet is organized according to the WordNet hierarchy of concepts. Each image is labeled with a single label. Due to the large size of the dataset, single-label tagging and a semantic hierarchy of concepts, the labels include inconsistencies: a very similar image of a cow is labeled "cow" in some instance and "animal" in another instance [42]. The version of ImageNet that is most commonly used in computer vision was constructed for the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [43]. An important feature of this version is that it only uses 1000 categories, and these categories were selected so that there is no overlap between the associated synsets; in practice this means that there are no parent-daughter concepts included (no "ani-

[1]See https://www.image-net.org/

Dataset	Images	Categories	Captions	Cat./image	Instances/image	Year
ImageNet-21k	14 197 122	21 841	-	1	1	2009
ImageNet-1k (ILSVRC-2012)	1 431 167	1000	-	1	1	2012
JFT-300M	303 000 000	18 291	-	1.2	1.2	2017
PASCAL VOC 2012	11 530	20	-	< 2	2.38	2012
SUN	22 210	4479	-	> 17	> 17	2010
Flickr30k	31 783	-	158 915	-	-	2014
COCO-2014	164 062	80	616 435	3.5	7.7	2014
COCO	328 124	80	616 435	3.5	7.7	2015
COCO-Stuff	164 062	172	616 435	> 3.5	> 7.7	2016
ADE20K	25 210	2693	-	10.5	19.5	2017
Visual Genome	108 077	33 877	5 406 939	< 35	35	2017
FlickrStyle10k	7000	-	14 000	-	-	2017
Personality-Captions	201 858	-	241 858	-	-	2019
Localized Narratives	848 749	-	873 107	-	-	2020

Table 2.1: Comparison of datasets relevant to image captioning. The statistics marked by - are not reported or not applicable. The COCO-2014 is the dataset version used for the Microsoft COCO Image Captioning Challenge, whose scores are reported for the image captioning systems in this thesis. Data sources: [32, 8, 58, 47, 36, 29, 48, 15, 42] and http://image-net.org/about-stats

mal", only "cow), and all categories are mutually exclusive. The ILSVRC-2012 version (version of the dataset for the challenge of year 2012) is referred to as ImageNet-1k, whereas the full dataset is referred to as ImageNet-21k. ImageNet is primarily an image classification dataset, but it is very important to object detection and image captioning because it is very commonly used for training the backbones of object detectors.

Even larger than the ImageNet, the **JFT-300M** [48] dataset contains 303M images with a total of 375M automatically generated labels. On average each image has 1.26 labels. There are 18 291 categories, including for example 1165 types of animals and 5720 types of vehicles. The categories form a rich hierarchy, with maximum hierarchy depth of 12 and maximum number of children per parent node being 2876. Approximately 20 % of the labels in the dataset are noisy. The category distribution is strongly long-tailed: there are more than 2M *flowers* but only 131 images of *train conductors*, and 2000 categories have less than 20 images. The JFT-300M dataset is owned by Google and is not publicly available, so it has not and cannot be used as a standard dataset in scientific research. The research using the dataset cannot be reproduced, but the research community has accepted papers by Google researchers based on the dataset to be published in peer-evaluated publications.

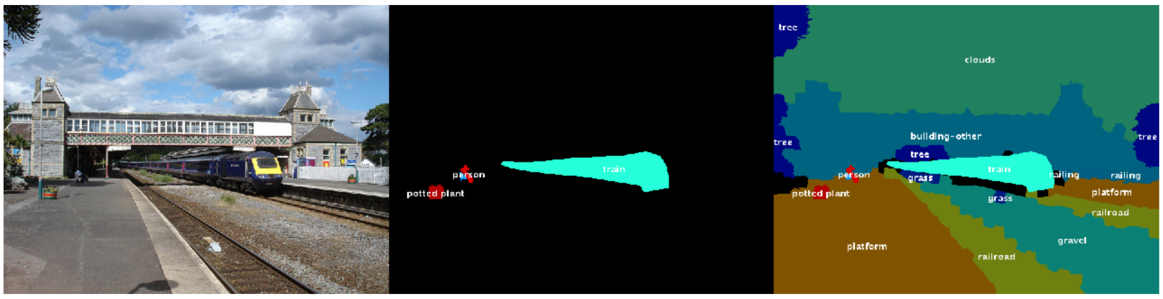
A second category of datasets are *semantic segmentation datasets*. Early well-known semantic segmentation datasets were PASCAL VOC 2012 [16] and SUN. Especially SUN included significant contextual information [32]: it had over 17 object and stuff instances per image, making the images much more complicated and contextual than the one-object-per-image oriented ImageNet. These early datasets served primarily object detection tasks, but did not yet include descriptive captions. The **Flickr30k** dataset, published in 2014, was one of the first larger datasets that focused on providing semantic natural language descriptions, captions, for images. Using images harvested from Flickr and crowd-sourcing, this dataset provided 31 783 images with five captions each. Flick30k did not become as central as COCO in scene understanding research communities. A likely reason is that COCO had a larger number of images and provided richer metadata supporting a wider number of scene understanding tasks.

The most important semantic segmentation and image captioning dataset **COCO** was originally published during 2014 and 2015 with the name "The Microsoft Common Objects in COntext (MSCOCO) dataset" [32]. It has later evolved to a multistakeholder project and changed its name to a simpler "COCO dataset". The year 2014 part of the COCO dataset (COCO-2014) contains 123 287 training/validation images and additionally 40 775 test images, for which captions are not public. This is the dataset used in the Microsoft COCO Image Captioning Challenge, whose scores are reported for the image captioning systems in this thesis. The training/validation images of COCO-2014 are officially split as 82 783 training images and 40 504 validation images, but in practice the image captioning community uses the *Karpathy split*[1] [27], where 113 287 images are designated as training images, 5000 as validation and 5000 as offline testing images. The second part of COCO was published in 2015, doubling the size of COCO to a total of 328 124 images. The second part of the dataset included similar images and object detection metadata as the 2014 part, but no captioning metadata. The COCO dataset has labels for 80 categories of *things*, objects for which there are individual instances (person, chair, bicycle). The original COCO dataset did not include "stuff" categories, materials and objects without clear boundaries (sky, street, sand). However, in 2017, a separate Coco-Stuff project [8] added 91 stuff categories and the category *unlabeled* to the same set of images. The COCO dataset specifically aims to have non-iconic images containing objects in their natural context. One way to estimate the amount of contextual information present in an image is by calculating the average number of object categories and instances per image. The original COCO has an average of 3.5 categories and 7.7 instances per image, whereas the ImageNet has only one category per image, and PASCAL VOC has, on average, less than 2 categories and 3 instances per image. Under 10 % of the COCO images have only one category.

COCO only uses a predefined, closed set of entry-level categories, category labels commonly used by humans when describing objects [32]. Examples of entry-level categories are *dog*, *chair* and *person*. COCO does not use object-part categories like *face*,

[1]The Karpathy split is named after Andrej Karpathy due to Andrej Karpathy's and Li Fei-Fei's paper where it was initially introduced, see the reference.

foot and *fuselage*. The criteria for the object category selection included usefulness for practical machine learning tasks, category diversity, balance between supercategories (such as animals, vehicles and furniture), compatibility with other datasets, and availability of a large enough number of image instances containing the category. An example COCO image with the enriched COCO-Stuff segmentation is presented in figure 2.5. The full label hierarchy for COCO (including COCO-Stuff labels) in shown in figure 2.6.



A large long **train** on a steel **track**.
A blue and yellow transit **train** leaving the **station**.
A **train** crossing beneath a city **bridge** with brick **towers**.
A **train** passing by an over **bridge** with a railway **track** (..).
A **train** is getting ready to leave the train **station**.

Figure 2.5: Example COCO image (left) with original COCO thing segmentation (center) and COCO-Stuff-augmented segmentation (right). Scene captions written by humans below the images. Image source: Caesar et al. [8].

Due to the careful label design, only 6 % of pixels belong to the category *unlabeled* in the COCO dataset augmented with stuff labels [8]. 69,1 % of the pixels in the COCO training and validation dataset belong to stuff and 30,9 % to things. When considering the human-written scene captions, the stuff categories make up 38,2 % of the nouns. This means that the stuff categories are often seen as central to describing what a scene is about and to scene understanding. The spacial context of a category can be calculated by using distance between the target category component edge and surrounding category components and the relative angle to the centre of mass of the target category component [8]. This analysis on the COCO-Stuff dataset leads to observations like the following: trains are typically above railroads (thing-stuff); TVs are typically in front of persons (thing-thing); tiled walls are above tiled floors (stuff-stuff); and roads often have persons on both sides (stuff-thing). The support relations (such as *on top of*) are semantically important, whereas side-by-side relationships are less so. Some concepts have high probabilities of occurring right next to another concept: a backpack is often right next to a person (on their back), and snowboards are almost always in the middle of snow and attached to persons.

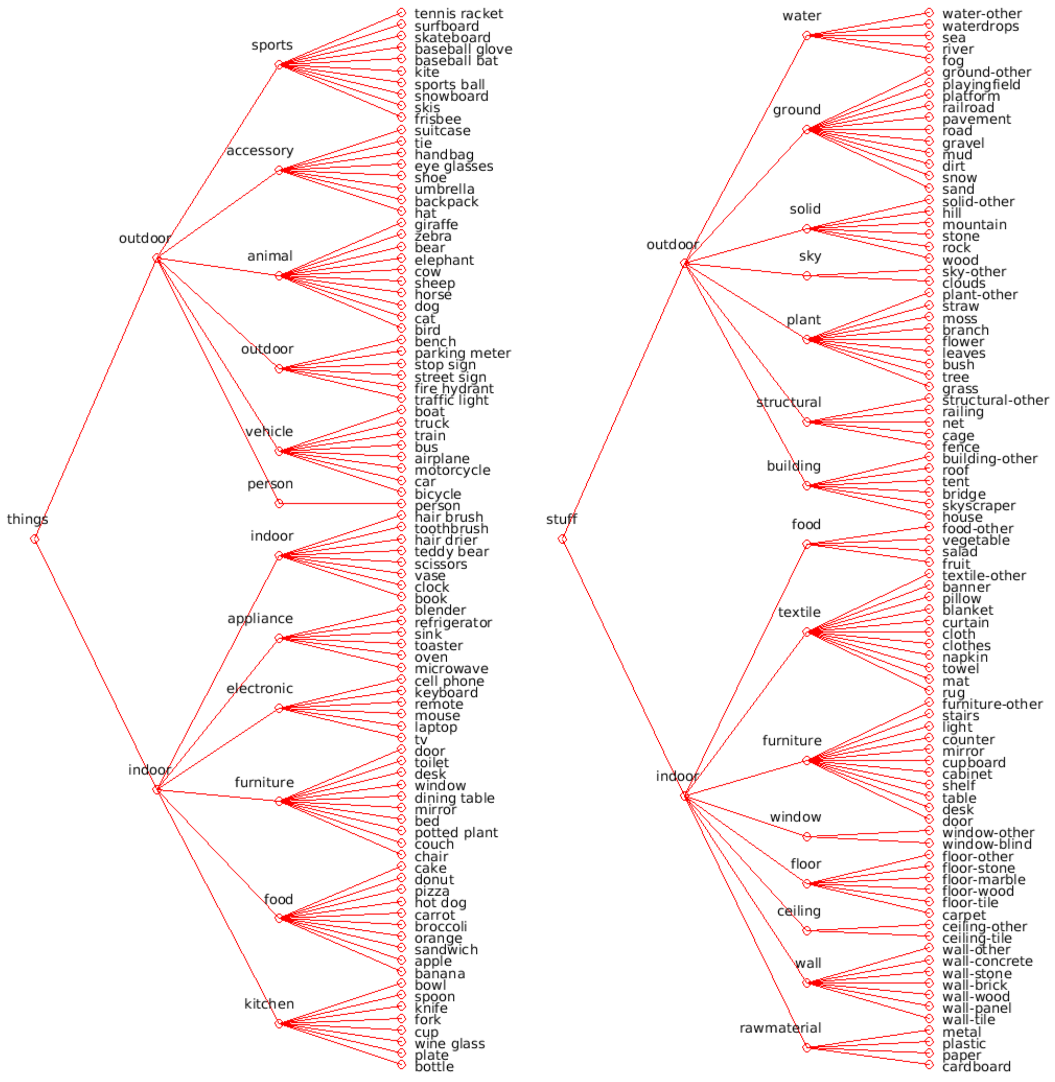


Figure 2.6: Label hierarchy of the COCO thing and stuff labels. Image source: The COCO-Stuff dataset, https://github.com/nightrome/cocostuff.

The **ADE20K** dataset [58] contains 20 210 training, 2000 validation and 3000 testing images. It annotates images with a large and an unrestricted open vocabulary. The annotations include object segments with names, object parts, and attributes. ADE20K uses dense annotation, meaning every pixel has a semantic label. The usage of an open vocabulary is both a strength and a weakness of the dataset. In general, with an open vocabulary, labeling is more difficult for human annotators, and naming is more inconsistent between different annotators. The creators of the ADE20K dataset have solved the inconsistency issue by having a single expert annotator annotate the whole dataset. The used part hierarchy has a depth of 3[1]. For example, a *knob* is a part of a *door*, which can be part of a *cabinet*. The other major issue with an open

[1]Full object-part hierarchy is available at https://groups.csail.mit.edu/vision/datasets/ADE20K/.

vocabulary is that there are rare object categories that appear only a few times in the whole dataset. The rare categories may be important for semantic understanding of the scene, but in practice there are too few instances for them to be useful for most machine learning algorithms. The images in ADE20K have not been manually curated to contain a similar number of instances for all caterories. As a result, the category distribution follows Zipf's law, with certain categories appearing much more frequently than others. There is an average of 10.5 object categories and 19.5 instances in an image. Any image contains at least 5 objects, and, at the largest, the number of object instances in an image is 273 (419 with part instances counted as well).

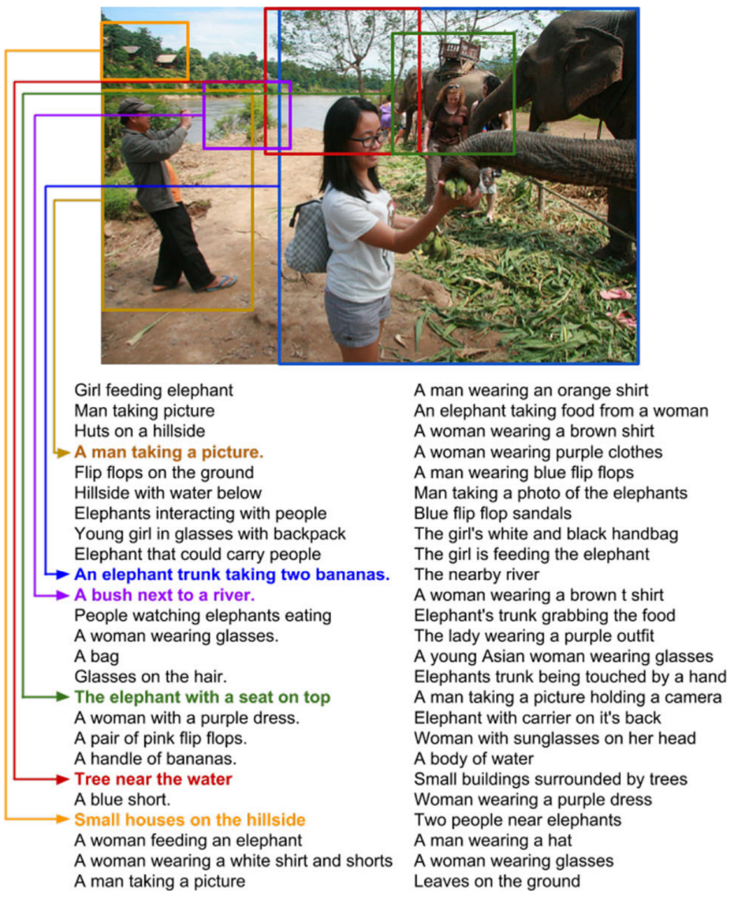
The **Visual Genome** [29] dataset[1] aims to support a very detailed understanding of an image. It provides a detailed dense annotation of objects, attributes, and relationships within each image. It provides natural language descriptions for several regions of an image instead of just one description (caption) for the whole image. The dataset consists of a subset of COCO, and contains 108k images, where each image has an average of 35 objects, 26 attributes and 21 pairwise relationships between objects. On average, the region descriptions are 5 words long. A example image with all 50 region descriptions is shown in figure 2.7. The related bounding box for only 6 regions is shown in the image for clarity, but similar bounding boxes exist for all regions. Regions with only one object usually have descriptions focusing on the attributes of a single object. Regions with two or more objects generally have descriptions about the attributes of several objects and their pairwise relationships. Each object, attribute and relationship is grounded in the image with a bounding box (these detailed bounding boxes are not shown in the sample image in figure 2.7).

Visual Genome is important for image captioning, because it was used by Anderson et al. [2] for creating rich feature vectors used by most leading image captioners. Visual Genome makes it possible to ground visual concepts – object categories, their attributes and relations – to image regions more firmly than just using images and image-level captions. For this dense grounding approach, looking at complimentary datasets to Visual Genome, a prominent options would be the Flickr30k Entities [36], which extends the Flickr30k with grounding the entities mentioned in captions to image regions.

The third category of datasets relevant to image captioning are specialized datasets. These datasets define metadata of an image in a more detailed way than standard scene understanding datasets such as COCO. Two datasets, FlickrStyle10k and Personality-Captions, extend the metadata with variations of language style. Localized narratives offers more explicit grounding of captions in the image.

The **FlickrStyle10k** dataset [17] takes a part of the Flickr30k dataset and aug-

[1] See https://visualgenome.org/



Girl feeding elephant
Man taking picture
Huts on a hillside
A man taking a picture.
Flip flops on the ground
Hillside with water below
Elephants interacting with people
Young girl in glasses with backpack
Elephant that could carry people
An elephant trunk taking two bananas.
A bush next to a river.
People watching elephants eating
A woman wearing glasses.
A bag
Glasses on the hair.
The elephant with a seat on top
A woman with a purple dress.
A pair of pink flip flops.
A handle of bananas.
Tree near the water
A blue short.
Small houses on the hillside
A woman feeding an elephant
A woman wearing a white shirt and shorts
A man taking a picture

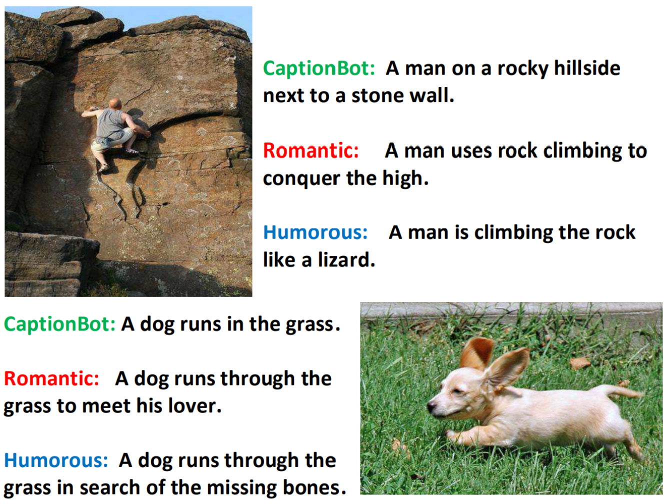
A man wearing an orange shirt
An elephant taking food from a woman
A woman wearing a brown shirt
A woman wearing purple clothes
A man wearing blue flip flops
Man taking a photo of the elephants
Blue flip flop sandals
The girl's white and black handbag
The girl is feeding the elephant
The nearby river
A woman wearing a brown t shirt
Elephant's trunk grabbing the food
The lady wearing a purple outfit
A young Asian woman wearing glasses
Elephants trunk being touched by a hand
A man taking a picture holding a camera
Elephant with carrier on it's back
Woman with sunglasses on her head
A body of water
Small buildings surrounded by trees
Woman wearing a purple dress
Two people near elephants
A man wearing a hat
A woman wearing glasses
Leaves on the ground

Figure 2.7: A sample Visual Genome image with all 50 region descriptions. The related bounding box for only 6 regions is shown in the image for clarity, but similar bounding boxes exist for all 50 regions. Image source: Krishna et al. [29].

ments them with additional captions with different language styles like humorous or romantic. Whereas standard captioning datasets like COCO and Flickr30k provide captions that are factual in style, FlickrStyle10k aims to teach models that can output language in specific styles, which would expand the usage contexts of models. The dataset contains 7000 images. A sample image with captions in different styles is shown in figure 2.8.

In the **Personality-Captions** [47] dataset, the image captions are engaging and conversational in style, with each caption conditioned on one of 215 separate personality traits (for example anxious, dramatic, money-minded and humble)[1] The dataset consists of 241 858 captions that relate to 200 000 images. This dataset is an interesting expansion of both the dataset metadata provided and the directly related task of generating captions with different attitudes or traits. A sample image with captions

[1]Pretrained models and caption examples available at https://parl.ai/projects/personality_captions/.



CaptionBot: A man on a rocky hillside next to a stone wall.

Romantic: A man uses rock climbing to conquer the high.

Humorous: A man is climbing the rock like a lizard.

CaptionBot: A dog runs in the grass.

Romantic: A dog runs through the grass to meet his lover.

Humorous: A dog runs through the grass in search of the missing bones.

Figure 2.8: Captions with different styles based on the FlickrStyle10k dataset. Captions created by StyleNet. Image source: Gan et al. [17]

in different styles is shown in figure 2.9.

Image	Personality	Generated comment
	Anxious Happy Vague Dramatic Charming	I love cats but i always get so scared that they will scratch me. That cat looks SO happy to be outside. That's a nice cat. Or is it a lion? That cat looks so angry; it might claw your eyes out! Awww, sweet kitty. You are so handsome!
	Sweet Vague Cultured Paranoid Overimaginative	I love, love, love these chairs! I want the big one in my house! This chair is either covered in snow or the snow is covered in the chair. These chairs remind me of the Swedish interior design revolution of the 70's. What if someone fell off those chairs. Those chairs look like they could be in a doll house.
	Skeptical Paranoid Happy Arrogant Humble	I wonder why the ships are all parked further down the deck. I hope those ships don't sink Look how beautiful the port is at this time of day! :) Those boats don't need to be docked at this time of night We are so lucky to have these boats available locally
	Romantic Anxious Creative Sweet Money-minded	A charming home that will call you back to days gone by. This house and this street just makes me feel uneasy. I could write a novel about this beautiful old home! What a cute little neighborhood! Call APR now to get your house renovated!

Figure 2.9: Sample captions using TransResNet model on the Personality-Captions dataset. Image source: Shuster et al. [47].

The **Localized Narratives** [37][1] dataset is an important recent contribution, that adds new types of annotations to several datasets, including the whole COCO, Flick30k and ADE20K. A total of 849k images are annotated. The new annotations consist of a spoken caption, transcription of the spoken caption, and related mouse traces to ground the caption segments into image regions. The annotators are asked

[1]See data at https://google.github.io/localized-narratives/

to use their own voice to describe the images and at the same time move their mouse to point at what region of the image they are describing. These two new modalities provide a dense grounding for language in the image, at the detail level of a mouse trace segment per word. The annotators are asked to transcribe their own description immediately after giving it in spoken form, ensuring the quality of the annotations. Localized Narratives can be used for several tasks, including image captioning. As the captions in this dataset are originally provided in spoken language, they are longer than previous captions: where COCO captions have an average of 10,5 words, the Localized Narratives captions have an average of 36,5 words. This new dataset could be used to train models to write longer captions than those originally in the COCO dataset. The dense grounding of language in the image provides completely new possibilities for improving image captioner performance, including using mouse traces to guide attention at training time. A comparison of different types of captions is shown in figure 2.10.

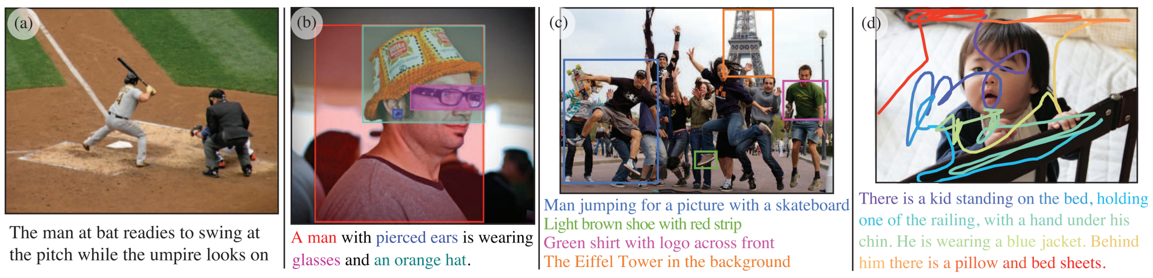

(a)
The man at bat readies to swing at the pitch while the umpire looks on
(b)
A man with pierced ears is wearing glasses and an orange hat.
(c)
Man jumping for a picture with a skateboard
Light brown shoe with red strip
Green shirt with logo across front
The Eiffel Tower in the background
(d)
There is a kid standing on the bed, holding one of the railing, with a hand under his chin. He is wearing a blue jacket. Behind him there is a pillow and bed sheets.

Figure 2.10: Comparison of different types of captions. In *a*, a COCO caption describes the image as a whole. In *b*, a Flickr30k Entities caption grounds concrete words of the caption in the image regions. In *c*, Visual Genome captions separately describe several regions of the image. In *d*, a Localized Narratives caption is grounded in the image with mouse trace segments. Notice that the Localized Narratives caption was originally spoken and has been transcribed into text. Image source: Pont-Tuset et al. [37].

2.3 Automated Evaluation Metrics

There are several standard automated metrics (BLEU, ROUGE, METEOR, CIDEr) that are routinely used to evaluate natural language generation models on several different tasks, including image captioning. They are all calculated based on a set of reference sentences, in the case of image captioning, the ground truth captions. All of these metrics rely on evaluating overlap of n-grams between the produced caption and ground truth captions [1]. The metrics are completely based on or oriented towards precision (BLEU), recall (ROUGE), and on a balance of precision and recall (METEOR, CIDEr). There are also different variations of the metrics, and automated

NLG evaluation metrics are a field of research in its own. A captioning-specific metric, SPICE [1], takes a more ambitious semantics-oriented approach and is somewhat commonly used within the captioning task research. It is, however, not as well known as the previously mentioned four metrics, likely due to the fact that it can only be used in the captioning domain, while the others can be used for many other NLG domains. SPICE is not reported by the COCO test server.

In this thesis, I will use the CIDEr-D c40 [54] score for reporting the performance of the models. The CIDEr-D c40 is the primary metric used to rank the captioning models in the online COCO evaluation server. CIDEr (Consensus-based Image Description Evaluation) uses stemmed n-grams and weights them using a Term Frequency Inverse Document Frequency (TF-IDF) approach, effectively giving more weight to more informative (less frequent) expressions [54]. The point of comparison is 40 ground truth captions for each image in the COCO test test (hence the c40 postfix). It should be noted, that for the COCO training and validation sets that are available to be used for training models, only 5 captions per image exist. The 40 captions per image for the not public test set were created to get an even more broad base to evaluate the models against.

Formally, the goal of CIDEr [54] is to automatically evaluate, for image I_i, how well a candidate sentence c_i matches the consensus of a set of image descriptions $S_i = \{s_{i1}, s_{i2}, ..., s_{im}\}$. All words in both candidate and reference sentences are first stemmed: "fish", "fishing" and "fished" all get reduced to "fish". The sentences with stemmed words are then each represented as a set of one or more ordered words that are present in the sentence, n-grams. The n-grams of the candidate sentence are then compared to the reference sentences: a good candidate sentence is one that has n-grams, that are also present in the reference sentences and does not have n-grams that are not found in the reference sentences. n-grams that are common in many captions in the dataset are less informative, and should be given lower weight. The number of times an n-gram ω_k, where k is an index, occurs in a reference sentence s_{ij} is denoted by $h_k(s_{ij})$, and the number of times it appears in a candidate sentence by $h_k(c_i)$. The TF-IDF weighting $g_k(s_{ij})$ for each n-gram ω_k is computed using the equation

$$g_k(s_{ij}) = \frac{h_k(s_{ij})}{\sum_{\omega_l \in \Omega} h_l(s_{ij})} \log \left(\frac{|I|}{\sum_{I_p \in I} \min(1, \sum_q h_k(s_{pq}))} \right) \tag{2.1}$$

where l is a set of n-grams that appear in the corpus (set of words in case of $n = 1$), and I is the set of all images in the dataset. The first term on the right side of the equation is the term frequency. In the case of $n = 1$, for example the word "cat" could occur 500 times in a set of captions containing a total of 10 000 words, and would then have TF(cat) = 500/10000 = 0.05. The second term on the right side of the equation, IDF, measures the rarity of an n-gram ω_k. It reduces the weight of those n-grams that commonly appear across many captions in the dataset. IDF is computed using the logarithm of the number of images in the dataset $|I|$ divided by the number of images for which ω_k occurs in any of its reference sentences. In the $n = 1$ case, words like "person" will have a low IDF, as they are found in very many images and related ground truth caption sets.

The CIDEr score for n-grams of length n is calculated using average cosine similarity between the candidate sentence and reference sentences:

$$\text{CIDEr}_n(c_i, S_i) = \frac{1}{m} \sum_j \frac{\boldsymbol{g^n}(c_i) \cdot \boldsymbol{g^n}(s_{ij})}{\|\boldsymbol{g^n}(c_i)\| \, \|\boldsymbol{g^n}(s_{ij})\|} \tag{2.2}$$

where $\boldsymbol{g^n}(c_i)$ is a vector calculated by concatenating the TF-IDF weightings $g_k(c_i)$ of all n-grams of length n in the candidate sentence c_i, $\|\boldsymbol{g^n}(c_i)\|$ is the length of that vector, and $\boldsymbol{g^n}(s_{ij})$ and $\|\boldsymbol{g^n}(s_{ij})\|$ similarly for the set of reference captions for the image.

The final CIDEr score is calculated by summing the n-gram CIDEr scores for n-grams of length 1, 2, 3 and 4:

$$\text{CIDEr}(c_i, S_i) = \sum_{n=1}^{4} w_n \text{CIDEr}_n(c_i, S_i) \tag{2.3}$$

The scaling factor w_n would make it possible to give more emphasis to either shorter or longer n-grams, but CIDEr uses uniform weights, and sets $w_n = 1/N$.

CIDEr reflects well the human perception of model performance on the captioning task. CIDEr-D is a modification of the basic CIDEr that aims to eliminate gaming possibilities of the metric. It achieves this by not stemming words, introducing a penalty for differing sentence length between target and ground truth, and penalizing the repetition of specific n-grams beyond the number of times they occur in the reference sentence. The CIDEr-D [54] variants of the already described CIDEr equations are:

$$\text{CIDEr-D}_n(c_i, S_i) = \frac{10}{m} \sum_j e^{\frac{-(l(c_i)-l(s_{ij}))^2}{2\sigma^2}} * \frac{\min(\boldsymbol{g^n}(c_i), \boldsymbol{g^n}(s_{ij})) \cdot \boldsymbol{g^n}(s_{ij})}{\|\boldsymbol{g^n}(c_i)\| \, \|\boldsymbol{g^n}(s_{ij})\|} \tag{2.4}$$

$$\text{CIDEr-D}(c_i, S_i) = \sum_{n=1}^{4} w_n \text{CIDEr-D}_n(c_i, S_i) \tag{2.5}$$

where $l(c_i)$ denotes the length of a candidate sentence, $l(s_{ij})$ denotes the length of the reference sentences, σ is a constant ($\sigma = 6$), and scaling factor $w_n = 1/N$ is again uniform. The constant factor of 10 is added to make CIDEr-D numerically similar to other metrics, especially the standard CIDEr. The CIDEr authors report [54] that the CIDER-D version of the metric has a rank correlation of 0.94 with the original CIDEr metric while being more robust to gaming.

The CIDEr-D variant of CIDEr is used by the COCO online server and in this thesis. For all the models, a wider set of metrics (c40 versions of BLEU-1, BLEU-2, BLEU-3, BLEU-4, METEOR, ROUGE-L, CIDEr-D), all provided by the COCO online test server, are reported in Appendix A.

The top COCO evaluation server CIDEr-D c40 score as of 31.3.2021 23:59:59 UTC was 1.387 by the entry MSR-MS_Cog_Svcs (presumably a Microsoft Cognitive Services based team), which does not currently have a related published research paper, code or other information to accompany it.

3. Object Detection in Leading Image Captioning Systems

Features matter.
- Ross Girshick, Jeff Donahue, Trevor Darrell and Jitendra Malik

All leading image captioners use an object detection network to calculate the object detections and their feature vectors as input for the image captioning network. Interestingly, all the leading image captioners under investigation use the *same* object detector, the Bottom-Up object detector by Anderson et al. [2], which is in turn based on the Faster R-CNN [40]. The object detector arguably does a large part of the work of an image captioner, and therefore is in fact an essential part of the image captioning system. In this chapter, I will present the object detection network used by the image captioners. In the next chapter (4), I will present the image captioner components of the systems. Later, in chapters 5 and 6, I will investigate ways to further improve the backbone and object detector components of the image captioning system.

Object detection is a process where all areas of an image containing objects of interest are bounded, while areas of the image that are not part of these objects are ignored [26]. The most straightforward way to bound the object instances is to define a bounding box around each object instance. The bounding boxes are usually defined using spatial coordinates of its top-left corner and its width and height. The downside of the bounding box approach is that for objects that are partially occluded by other objects or have very complex shapes, a notable amount of pixels within the bounding box does not actually belong to the object instance. In response to this problem, more fine-grained approaches of bounding the object instance using polygons [26] or a pixel-level mask [21] are also used. The more fine-grained approaches are closely related to the bounding box approach, and can often be computed in sequence or in parallel within the same pipeline or end-to-end-model [26, 9]. All of the leading image captioning methods under investigation use feature vectors based on bounding box object detections.

3.1 Faster R-CNN

Faster R-CNN is an evolution of the Fast R-CNN [18] (Fast Region-based Convolutional Network), which was published by Ross Girshick in 2015. The Fast R-CNN is an object detector, which builds on the work of CNNs used for object classifications (like VGG-16) and the earlier (2014) ground-breaking work on the R-CNN [19] model.

R-CNN. R-CNN [19] introduced many of the central techniques used by Fast R-CNN, Faster R-CNN and through it, also the image captioners under investigation. It extracted a large number (about 2000) category-independent region proposals for the input image, then extracted a fixed-length feature vector for each proposal, and then classified each region as belonging to an object category [19]. It also used the techniques of *affine image warping* to compute a fixed-size CNN input from each region proposal, independent of the region's shape, *greedy non-maximum suppression* to find unique object instances from a larger set of potential object instances, and *bounding box regression* to learn correct locations for object instances. Another contribution, still valid and in use, was using supervised pretraining of the CNN on a large auxiliary dataset (ImageNet), followed by domain-specific fine-tuning on a smaller dataset (in R-CNN case Pascal VOC), a technique that later came to be known as transfer learning. An important feature of the R-CNN was that it was agnostic of the region proposal process, being compatible with several ways of producing region proposals. It did utilize the selective search technique to demonstrate the approach in practice.

Fast R-CNN. Fast R-CNN addressed the main problems of the R-CNN: slow and space-demanding multi-stage training and slow test-time performance [18]. It used the VGG16 as the backbone CNN, trained the whole network in one stage and updated not only the custom object-detection layers, but also the backbone convolutional layers. It also replaced R-CNN's support vector machines in the object detector with region of interest (RoI) pooling, fully connected layers and a softmax layer. The region of interest pooling layer extracted a fixed-length feature vector for each object proposal from the shared feature map produced by the backbone CNN. In practice, RoI pooling worked by dividing a RoI window of height h and width w into a grid of sub-windows $H \times W$ of approximate size $h/H \times w/W$, and then max-pooling the values in each sub-window into one output cell. The max pooling was applied independently to each feature map channel. The typical size of the fixed-size feature map after RoI max pooling was $7 \times 7 \times 512$. The two fully connected layers after RoI max pooling were 4096 units wide. Architecture of Fast R-CNN is shown in figure 3.1. Fast R-CNN used a multi-task loss function to simultaneously learn to output the correct object category and a bounding box for the object instance [18].

Faster R-CNN. The Fast R-CNN was still agnostic as to how to extract the

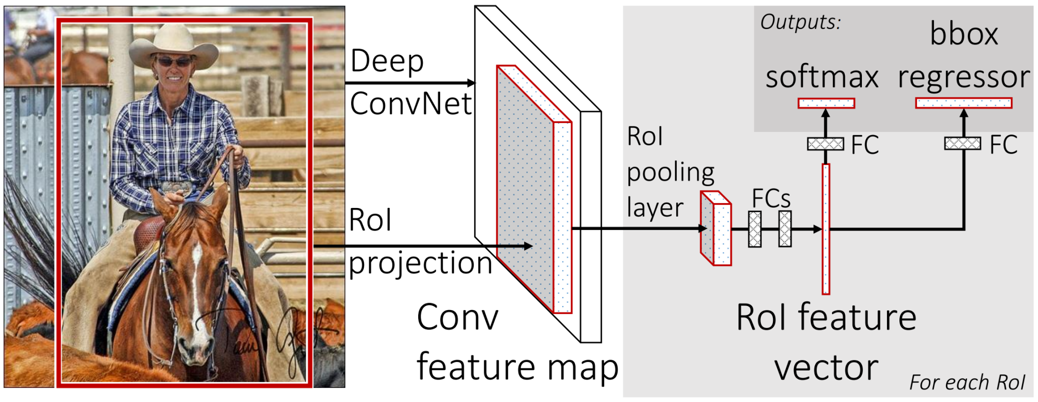


Figure 3.1: Architecture of Fast R-CNN. Note that the architecture does not include how to create object proposals. Image source: Girshick [18].

object proposals. Faster R-CNN [40] evolved Fast R-CNN by introducing the region proposal network (RPN), sharing convolutional layers between the RPN and object detector, and hence implementing the region proposal process and per-region feature extraction in an efficient way. A region proposal network is a fully convolutional network that predicts object bounds and objectness scores for several anchor positions. The RPN effectively works as an attention mechanism, telling the object detector where in the image to look. Faster R-CNN also introduced *anchor boxes*, reference points, which were used to examine proposals at multiple scales and aspect ratios. The architecture of the region proposal network is shown in figure 3.2.

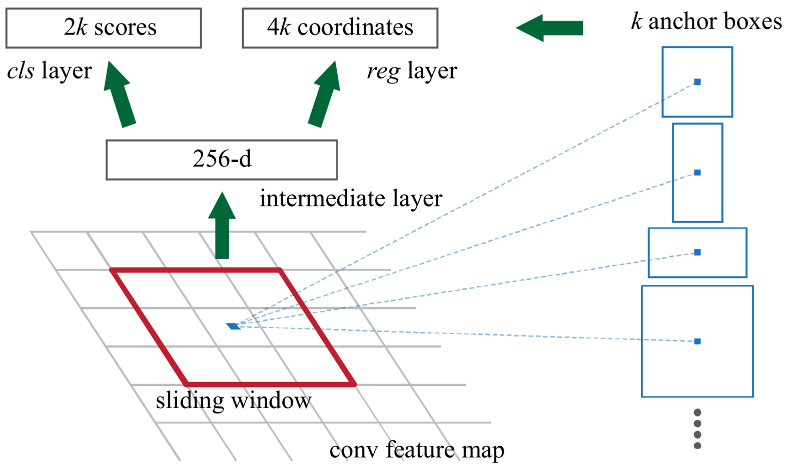


Figure 3.2: Structure of the region proposal network (RPN) in Faster R-CNN. Note especially the anchor boxes with varying sizes and ratios. Typically there would be 9 anchor boxes per position. The convolutional feature map shown is a mid- to high-level map from the backbone CNN (see e.g. figure 4.7 for more details on the complete architecture). Image source: Ren et al. [40].

In the RPN training, positive objectness classification is assigned to any anchor box with the highest intersection-over-union (IoU)[1] overlap with a ground-truth box, or an anchor that has an IoU overlap higher than 0.7 with any ground truth box. This means that there can be several anchors associated with a single ground-truth box at this stage. On the other hand, a negative classification label is assigned to any non-positive anchor if its IoU is less than 0.3 for all ground-truth boxes. Anchors that are not either positive nor negative do not contribute to the training objective.

With Faster R-CNN, the typical amount of proposals per image was still 2000 at training time, but was reduced from 2000 down to 300 at test time, making the inference very fast compared to Fast R-CNN [40]. Due to the non-maximum suppression, there actually is a variable amount of object proposals, often much less than 300.

In the original Faster R-CNN paper [40], alternating optimization was used. Alternating optimization meant training the object detector and shared convolutional layers while keeping the RPN frozen, then switching to training the RPN and convolutional layers, while keeping object detector layers frozen, and iterating. Later the authors stated that approximate joint optimization produces similar quality results, but trains faster[2]. In the approximate joint optimization, there are four loss functions: objectness classification and bounding box regression in the RPN and object classification and bounding box regression in the object detector.

Faster R-CNN is compatible with different backbones. Already in 2015, the original backbone VGG16 was replaced with ResNet-101, and the upgraded Faster R-CNN won several leading object detection and image classification competitions [40]. Some sample object detections using Faster R-CNN on COCO images are shown in figure 3.3.

Faster R-CNN had been modified for semantic segmentation already when the article was released in 2015. The standard panoptic and semantic segmentation extension, published in 2017, is Mask R-CNN [21], which extends Faster R-CNN by predicting object masks for all object instances in parallel with the bounding box recognition process. It is still used in leading object detection platforms and libraries, for example the Detectron2 platform[3] by Facebook AI Research.

[1] *Intersection-over-union* is calculated simply by counting the pixels of the ground truth box and the predicted box that intersect (are in both boxes) and dividing it with the number of pixels in the union (are in at least one of the boxes). In practice this gives a statistic on how well the predicted box is aligned with the ground truth box.

[2] See original Python implementation at `https://github.com/rbgirshick/py-faster-rcnn` and Ross Girshick's presentation at `https://www.dropbox.com/s/xtr4yd4i5e0vw8g/iccv15_tutorial_training_rbg.pdf?dl=0`

[3] See `https://github.com/facebookresearch/detectron2`.

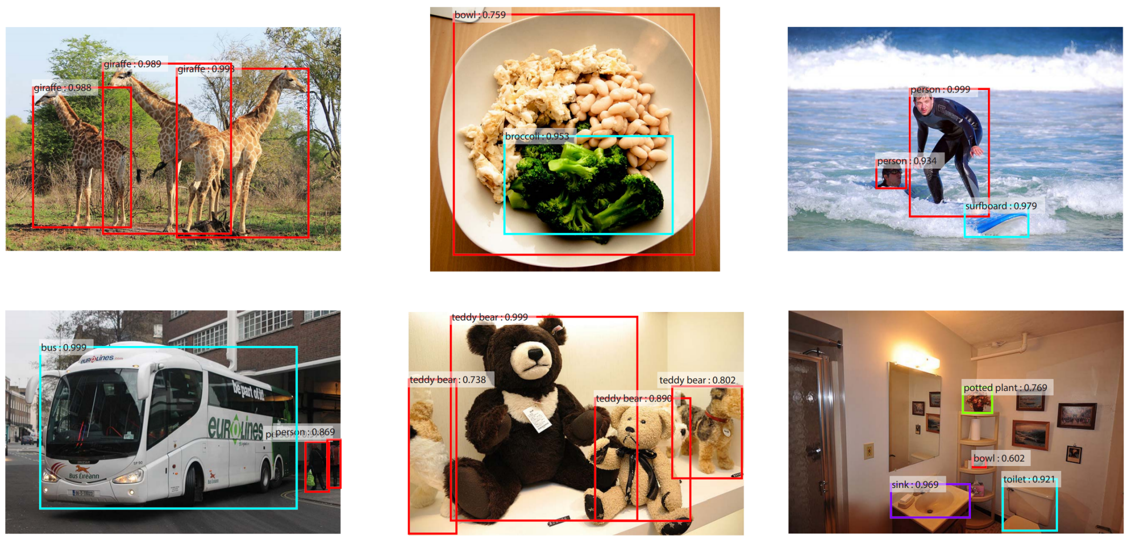


Figure 3.3: Sample object instance detections on some COCO images. The object detector is detecting the 80 categories present in the COCO dataset. The network is Faster R-CNN with VGG16 as backbone. An object category softmax score threshold of 0.6 is used for displaying a detection. Image source: Ren et al. [40].

3.2 Bottom-Up and Top-Down

In their 2018 paper, Anderson et al. [2] present a technique for image captioning, where they use a Faster R-CNN based network to determine the regions of interest of an image in a bottom-up fashion, and then use the feature vectors associated with each region in a top-down manner to determine feature weightings to use in image captioning and visual question answering (VQA). The leading image captioners being examined in this thesis still use a very similar high-level approach. More importantly, all leading image captioners[1] directly use the object detections and feature vectors calculated by the Bottom-Up and Top-Down authors as input into their high-level image encoders and language decoders[2]. Effectively these methods are using the object detector built by Anderson et al. as a part of their image captioning system. Therefore it is important to examine in detail the object detector part of their work. The image captioner is not examined at a similar level of detail, since the state-of-the-art image captioners have later improved on that part.

[1]Except for X-Linear Attention Network, which originally uses ResNet-101 and the provided features, but improves by swapping the backbone to SENet-154 and recalculates the features using Bottom-Up.

[2]Feature vectors are available at `https://github.com/peteanderson80/bottom-up-attention`. Both fixed 36 object instances per image and the adaptive 10 to 100 object instances per image are available as precalculated feature vectors for the whole COCO dataset.

The Bottom-Up and Top-Down method implements their object detector based on Faster R-CNN with ResNet-101 as the backbone. A major contribution is generating more rich feature vectors thanks to training the object detector using object and attribute annotations from the Visual Genome dataset [29]. The attributes are learned through adding a new loss function, object attribute loss, to the four existing loss functions of the Faster R-CNN (bounding box regression and objectness in RPN and bounding box regression and object classification in object detector). The five loss functions and the architecture of the Bottom-Up object detector, including the ResNet-101 backbone, is presented in the architecture diagrams of the leading image captioners in chapter 4, for example in figure 4.1.

The Visual Genome (VG) data set is used with a train-eval-test split of 98k/5k/5k, ensuring that the images that are contained in both COCO and Visual Genome (51k) are all in matching splits of the two datasets [2]. As VG object and attribute annotations are free-form text, heavy preprocessing of the annotations is applied: abstract classes are removed, and a structured set of 1600 object classes and 400 attribute classes is extracted. An important decision is not to merge or remove overlapping classes (like *person*, *man*, *guy*), classes with singular and plural formulations (like *tree* and *trees*), or stuff classes (like *sky* or *grass*). This richness of classes at training time contributes to creation of rich feature vectors, the true end goal of the object detector as a part of an image captioning system.

Non-maximum suppression is performed at the late object detection layers for each object class using an IoU threshold of 0.3 and for each bounding box using an IoU threshold of 0.7 [3]. Then all regions where any class detection probability exceeds a confidence threshold of 0.2 are selected. For each selected region i, the feature vector v_i is calculated as a mean-pooled, 2048-dimensional vector based on the backbone CNN features of that area [2].

In the paper [2], the authors initially allow the number of regions per image to vary with the complexity of the image, up to a maximum of 100. They, however, note, that selecting only the top 36 regions per image works nearly as well in the downstream image captioning and visual Q&A tasks.

There are several reasons, why the Bottom-Up and Top-Down captioning approach – which is inherited by the current leading image captioners – is so successful. One reason is that it can attend to both fine details and large image regions, thanks to the multi-scale, multi-ratio region proposals of the Faster R-CNN. A second reason is that the feature vectors are rich, containing a representation of all the visual attributes and concepts associated with the object. This enables processing all the information related to an object at once. The authors also note some similarities to the visual processing mechanisms in the human brain [2].

4. Overview of Leading Image Captioners

In this chapter, I will investigate the architectures and contributions of the five image captioning systems under investigation: Attention on Attention [25], the Meshed-Memory Transformer [12], the X-Linear Attention Network [35], the Show, Edit and Tell [44] method, and Prophet Attention [33]. The chosen image captioning systems and their basic statistics are shown in table 4.1. These systems were chosen, because they were the top methods found by following cross-references between captioning publications and searching through Google Scholar, with the exception of Show, Edit and Tell. Show, Edit and Tell was chosen due to a complimentary approach to captioning and taking into account that the COCO test server score was achieved using a single model, not an ensemble like in the other cases.

Captioning system	CIDEr-D c40 score	COCO online server rank 31.3.2021	Year
Attention on Attention	1.296	#34	2019
Meshed-Memory Transformer	1.321	#21	2019
X-Linear Attention Network	1.335	#13	2020
Show, Edit and Tell	1.257	#64	2020
Prophet Attention	1.337	#11	2020

Table 4.1: Image captioning systems chosen for investigation.

As already stated, all leading image captioning systems under investigation rely on the Bottom-Up object detector presented in the previous chapter. In this chapter, I will present architecture diagrams of the systems as a whole, including the backbone CNN and object detector layers. Even if all currently leading image captioners are effectively two-part systems, where the image captioning layers use only the feature vectors output by the object detector, it is important to keep in mind, that this kind of a split system with a well-defined bottleneck interface between the two parts is not the only possible architecture. Developing this top-level architecture will be discussed in chapter 7.

As the systems are effectively two-part systems, the training of the systems is also done in more than one stage. All described image captioning systems share a similar three-stage training process, which is described in detail in the section on the Meshed-Memory Transformer (section 4.2). The first stage is related to training the backbone, second to the object detector and the third to the image captioning components. There is some variation in the training within this third stage, and this variation in described in conjunction with each captioner.

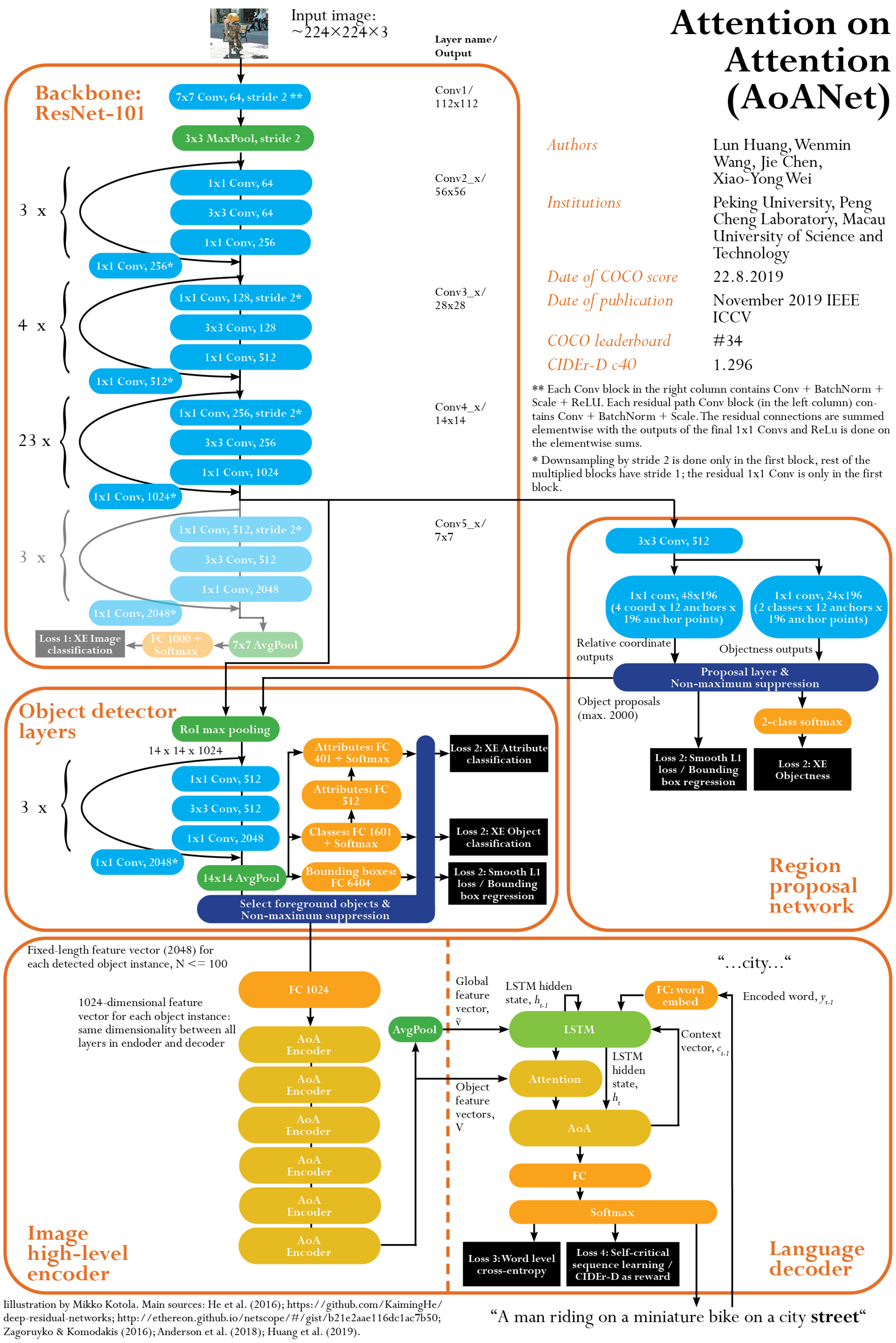
The captioning systems under investigation have all (except Show, Edit and Tell) been #1 or very close to the top on the COCO Image Captioning Challenge leaderboard when they have been published. At the time of this investigation (31.3.2021), there are already some newer entries to the COCO challenge, that have achieved higher scores than the five entries under investigation. New high-scoring entries are made almost daily, so the exact rankings change constantly. As captioning is an active research area with commercial applications, it is likely that some of the top entries have been made by professional commercial teams and may never be published. Based on the names of the entries, some of the top entries are likely by teams at Microsoft (MSR-MS_Cog_Svcs) and the National Laboratory of Pattern Recognition at the Chinese Academy of Science (IVA-CASIA). As the cycle of developing new methods in the academia is such, that test server runs are usually done 3-6 months before publishing a related research paper, it is also possible that some of the top COCO test server entries per 31.3.2021 will be published during 2021. The CIDEr-D c40 scores of the top 100 entries are between 1.387 and 1.210, with the top 32 entries being above 1.3, so it is likely that there are no major unpublished breakthroughs – this kind of breakthrough should result in notably higher scores. The chosen methods do therefore represent the current state-of-the-art of captioning methods.

Attention on Attention (AoANet)

Authors	Lun Huang, Wenmin Wang, Jie Chen, Xiao-Yong Wei
Institutions	Peking University, Peng Cheng Laboratory, Macau University of Science and Technology
Date of COCO score	22.8.2019
Date of publication	November 2019 IEEE ICCV
COCO leaderboard	#34
CIDEr-D c40	1.296

** Each Conv block in the right column contains Conv + BatchNorm + Scale + ReLU. Each residual path Conv block (in the left column) contains Conv + BatchNorm + Scale. The residual connections are summed elementwise with the outputs of the final 1x1 Convs and ReLu is done on the elementwise sums.

* Downsampling by stride 2 is done only in the first block, rest of the multiplied blocks have stride 1; the residual 1x1 Conv is only in the first block.



Input image: ~224×224×3
Layer name / Output
Backbone: ResNet-101
7x7 Conv, 64, stride 2 **
Conv1 / 112x112
3x3 MaxPool, stride 2
3 x
1x1 Conv, 64
3x3 Conv, 64
1x1 Conv, 256
1x1 Conv, 256*
Conv2_x / 56x56
4 x
1x1 Conv, 128, stride 2*
3x3 Conv, 128
1x1 Conv, 512
1x1 Conv, 512*
Conv3_x / 28x28
23 x
1x1 Conv, 256, stride 2*
3x3 Conv, 256
1x1 Conv, 1024
1x1 Conv, 1024*
Conv4_x / 14x14
3 x
1x1 Conv, 512, stride 2*
3x3 Conv, 512
1x1 Conv, 2048
1x1 Conv, 2048*
Conv5_x / 7x7
Loss 1: XE Image classification
FC 1000 + Softmax
7x7 AvgPool
3x3 Conv, 512
1x1 conv, 48x196 (4 coord x 12 anchors x 196 anchor points)
1x1 conv, 24x196 (2 classes x 12 anchors x 196 anchor points)
Relative coordinate outputs
Objectness outputs
Proposal layer & Non-maximum suppression
Object proposals (max. 2000)
2-class softmax
Loss 2: Smooth L1 loss / Bounding box regression
Loss 2: XE Objectness
Region proposal network
Object detector layers
RoI max pooling
14 x 14 x 1024
3 x
1x1 Conv, 512
3x3 Conv, 512
1x1 Conv, 2048
1x1 Conv, 2048*
14x14 AvgPool
Attributes: FC 401 + Softmax
Attributes: FC 512
Classes: FC 1601 + Softmax
Bounding boxes: FC 6404
Loss 2: XE Attribute classification
Loss 2: XE Object classification
Loss 2: Smooth L1 loss / Bounding box regression
Select foreground objects & Non-maximum suppression
Fixed-length feature vector (2048) for each detected object instance, N <= 100
FC 1024
1024-dimensional feature vector for each object instance: same dimensionality between all layers in endoder and decoder
AoA Encoder
AoA Encoder
AoA Encoder
AoA Encoder
AoA Encoder
AoA Encoder
AvgPool
Image high-level encoder
Global feature vector, $\bar{v}$
Object feature vectors, V
LSTM hidden state, h_{t-1}
"…city…"
FC: word embed
Encoded word, y_{t-1}
LSTM
Context vector, c_{t-1}
Attention
LSTM hidden state, h_t
AoA
FC
Softmax
Loss 3: Word level cross-entropy
Loss 4: Self-critical sequence learning / CIDEr-D as reward
Language decoder
"A man riding on a miniature bike on a city **street**"

Illustration by Mikko Kotola. Main sources: He et al. (2016); https://github.com/KaimingHe/deep-residual-networks; http://ethereon.github.io/netscope/#/gist/b21e2aae116dc1ac7b50; Zagoruyko & Komodakis (2016); Anderson et al. (2018); Huang et al. (2019).

Figure 4.1: Architecture of the complete AoANet system.

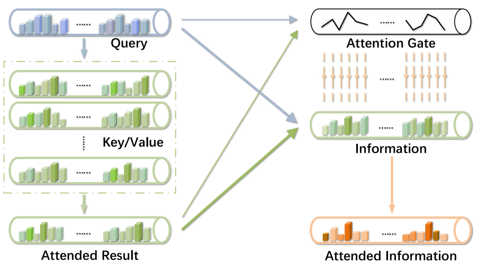


Figure 4.2: Attention on Attention model calculates an information vector and an attention gate using the attention result and the attention query. Then a second level attention mechanism is added by applying the gate to the information, resulting in the attended information vector. Image source: Huang et al. [25].

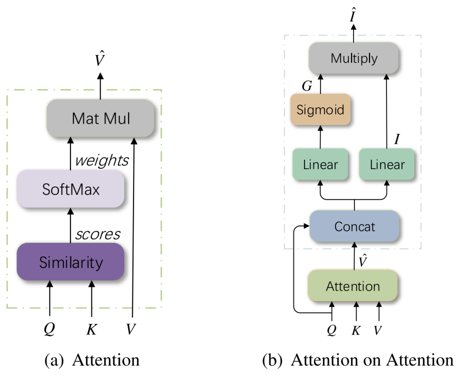


Figure 4.3: Comparison of the structure of plain attention and Attention on Attention blocks. Image source: Huang et al. [25].

4.1 Attention on Attention

Attention-based encoder-decoder captioners generally rely on the attention mechanism to guide the decoding process by generating a weighted average over the extracted feature vectors for each time step. Huang et al. [25] note that in earlier attention-based captioning models, the attended image-based vector at some time-step of the decoding process and the language-context based attention query are not always equally related. In some time-steps the decoder has to output filler words or abstract words not necessarily grounded in any image regions, but more purely on the state of the language decoder. This can lead to the decoder giving a misled output. The attention on attention based model *AoANet* solves this problem by introducing a second layer of attention, enabling the model to adjust its degree of attending to the image feature encodings and the language context. The main idea of the model is presented in figure 4.2. I will first discuss the general AoA module, which is the way of extending the plain transformer module to include second order attention. I will then describe how the AoA module is used to build an image captioner, AoANet. The architecture of the complete AoANet-based captioning system is shown in figure 4.1.

Attention on Attention module. The plain attention module $f_{\text{att}}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V})$ operates on queries $\boldsymbol{Q}$, keys $\boldsymbol{K}$ and values $\boldsymbol{V}$ and generates weighted average vectors $\hat{\boldsymbol{v}} = f_{\text{att}}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V})$ [25]. First, similarity scores between queries and keys are calculated, usually by calculating a dot product between the vectors:

$$\boldsymbol{a}_{\mathrm{i,j}} = f_{\mathrm{sim}}(\boldsymbol{q}_i, \boldsymbol{k}_j) \tag{4.1}$$

Then a softmax function is applied to the similarity scores to get attention weights:

$$\boldsymbol{\alpha}_{\mathrm{i,j}} = \frac{e^{\boldsymbol{a}_{\mathrm{i,j}}}}{\sum_j e^{\boldsymbol{a}_{\mathrm{i,j}}}} \tag{4.2}$$

And finally the attention weights are multiplied with the value vectors to get attention results:

$$\hat{\boldsymbol{v}}_i = \sum_j \boldsymbol{\alpha}_{\mathrm{i,j}} \boldsymbol{v}_j \tag{4.3}$$

These three equations correspond to the plain attention module visualized on left side of figure 4.3. Here, $\boldsymbol{q}_i \in \boldsymbol{Q}$ is the ith query, $\boldsymbol{k}_j \in \boldsymbol{K}$ and $\boldsymbol{v}_j \in \boldsymbol{V}$ are the jth key/value pair, f_{sim} is a function that gives the similarity score of each $\boldsymbol{k}_j$ and $\boldsymbol{q}_i$, and $\hat{\boldsymbol{v}}_i$ is the attended vector for the query $\boldsymbol{q}_i$.

As visualized on the right side of figure 4.3, AoA takes the result of the plain attention module $\hat{\boldsymbol{v}}$ and concatenates the original query vector with it. It then calculates two linear transformations [25], the information vector $\boldsymbol{i}$ and the attention gate $\boldsymbol{g}$:

$$\boldsymbol{i} = \boldsymbol{W}_q^i \boldsymbol{q} + \boldsymbol{W}_v^i \hat{\boldsymbol{v}} + \boldsymbol{b}^i \tag{4.4}$$

$$\boldsymbol{g} = \sigma(\boldsymbol{W}_q^g \boldsymbol{q} + \boldsymbol{W}_v^g \hat{\boldsymbol{v}} + \boldsymbol{b}^g) \tag{4.5}$$

where σ is the sigmoid activation function and $\boldsymbol{W}_q^i$, $\boldsymbol{W}_v^i$, $\boldsymbol{W}_q^g$, $\boldsymbol{W}_v^g \in \mathbb{R}^{D\times D}$ are embedding matrices, $\boldsymbol{b}^i$, $\boldsymbol{b}^g \in \mathbb{R}^D$ are bias vectors, and D is the dimension of $\boldsymbol{q}$ and $\hat{\boldsymbol{v}}$. AoA then finally does element-wise multiplication of the attention gate $\boldsymbol{g}$ and the information vector $\boldsymbol{i}$ to get the final attended information vector $\hat{\boldsymbol{i}}$:

$$\hat{\boldsymbol{i}} = \boldsymbol{g} \odot \boldsymbol{i} \tag{4.6}$$

AoANet. The captioning model built on the Attention on Attention module is called AoANet. It uses six stacked transformer-based AoA modules as the encoder and an LSTM-based, AoA-augmented decoder [25]. The original paper calls the encoder modules *refining modules*, meaning that they refine the input vectors' content and interactions. The structure of a refining module is shown in figure 4.4. In this thesis, I refer to this part of the model as the *high-level image encoder*. In the encoder, multi-head attention with 8 heads is used. Compared to the original transformer block [53], the feedforward layer is dropped. This change is justified by the fact of getting the required non-linearity in the AoA model and improved simplicity of the module.

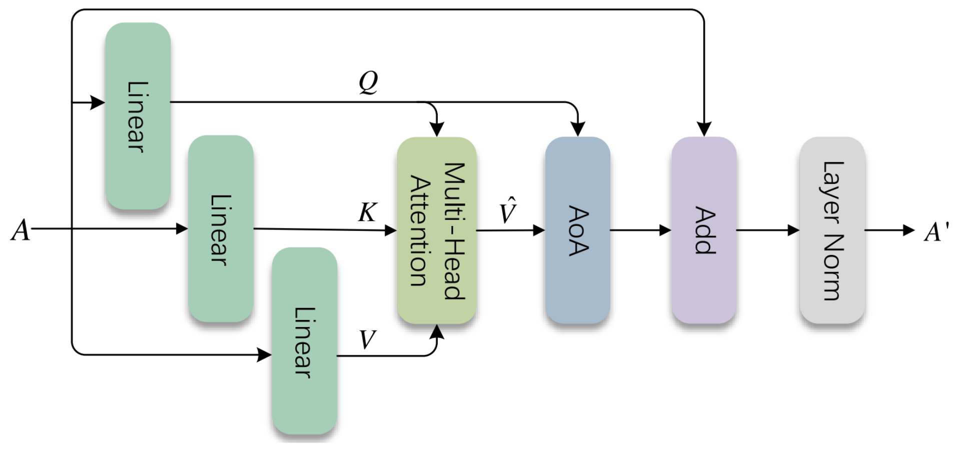


Figure 4.4: Architecture of a refining block in the AoANet encoder. It follows the AoA idea, adding residual connection and layer normalization. The function of the refining block is model the interactions among the detected objects in the image. Huang et al. [25].

The AoANet decoder is LSTM-based, but additionally uses an AoA module. The architecture of the decoder is shown in figure 4.5. As the AoA is applied on the decoder, the value of each channel of the attention gate $\boldsymbol{g}$ indicates the relevance of the information on the corresponding channel in the information vector $\boldsymbol{i}$ to the current time step. The second-level attention in the decoder learns to filters out irrelevant attention results and keeps only the useful ones. As is shown by the ablation study [25][1], the AoA-based decoder contributes far more to the CIDEr-D c40 score improvement than the AoA-based encoder. The contribution of the AoA in the decoder is 6.0 CIDEr-D points in comparison to a multi-head attention and LSTM-based decoder without AoA, whereas AoA in the encoder improves the score over the baseline only 2.0 points.

An example of the attention attributions of AoANet to image regions for different time steps is shown in figure 4.6. For time steps where the model is generating a word related to a concrete object in the picture, such as the last word "book", the attention is focused in the correct image regions. How strongly AoANet is relying on the image features is now shown in the image.

[1]An *ablation study* is a systematical study of a system by removing certain components, in order to understand the contribution of the component to the overall system.

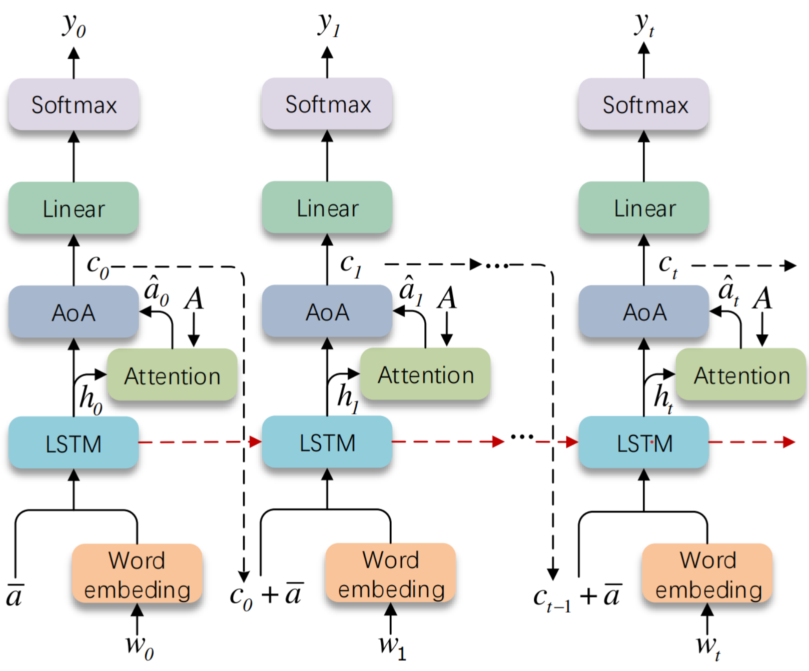


Figure 4.5: Architecture of the decoder in the AoANet encoder. It is LSTM-based and uses AoA to attend to the meaningful image features based on the decoder context. Huang et al. [25].

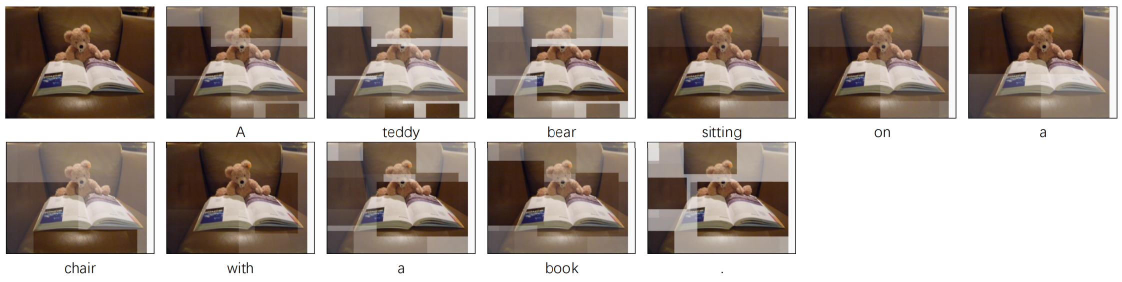


Figure 4.6: Example of attention attributions by AoA. Note that the main contribution of AoA is that it only selectively attends to these image regions due to the second level attention in the decoder. Huang et al. [25].

Object detection. Like other leading captioning models, AoANet relies on object detections and their feature vectors from the Bottom-Up Top-Down [2] model, meaning the backbone of the object detector is ResNet-101 [22], there is a region proposal network to select the object areas, and a set of object detection layers. The high-level image encoder takes as input 1 to n feature vectors of dimension 2048, one for each detected object instance in the image. The Bottom-Up model outputs a maximum of 100 object instances per image, so that is the practical upper limit. AoANet transforms the object feature vectors to a 1024-dimensional internal representation, and this dimensionality is used between encoder blocks and in the decoder.

Training. The AoANet encoder-decoder model is trained in two phases, similarly to the other captioners' encoder-decoders. The main training phase is supervised learning, and uses word-level cross-entropy loss to learn to output the next word of the caption using the previous words from the ground truth caption. Due to the model relying on the hidden state of the LSTM, this phase must be done step-by-step and cannot be done in parallel. The second phase of the training fine-tunes the sequence generation using reinforcement learning with the CIDEr-D as the optimization objective using the self-critical sequence training approach [41].

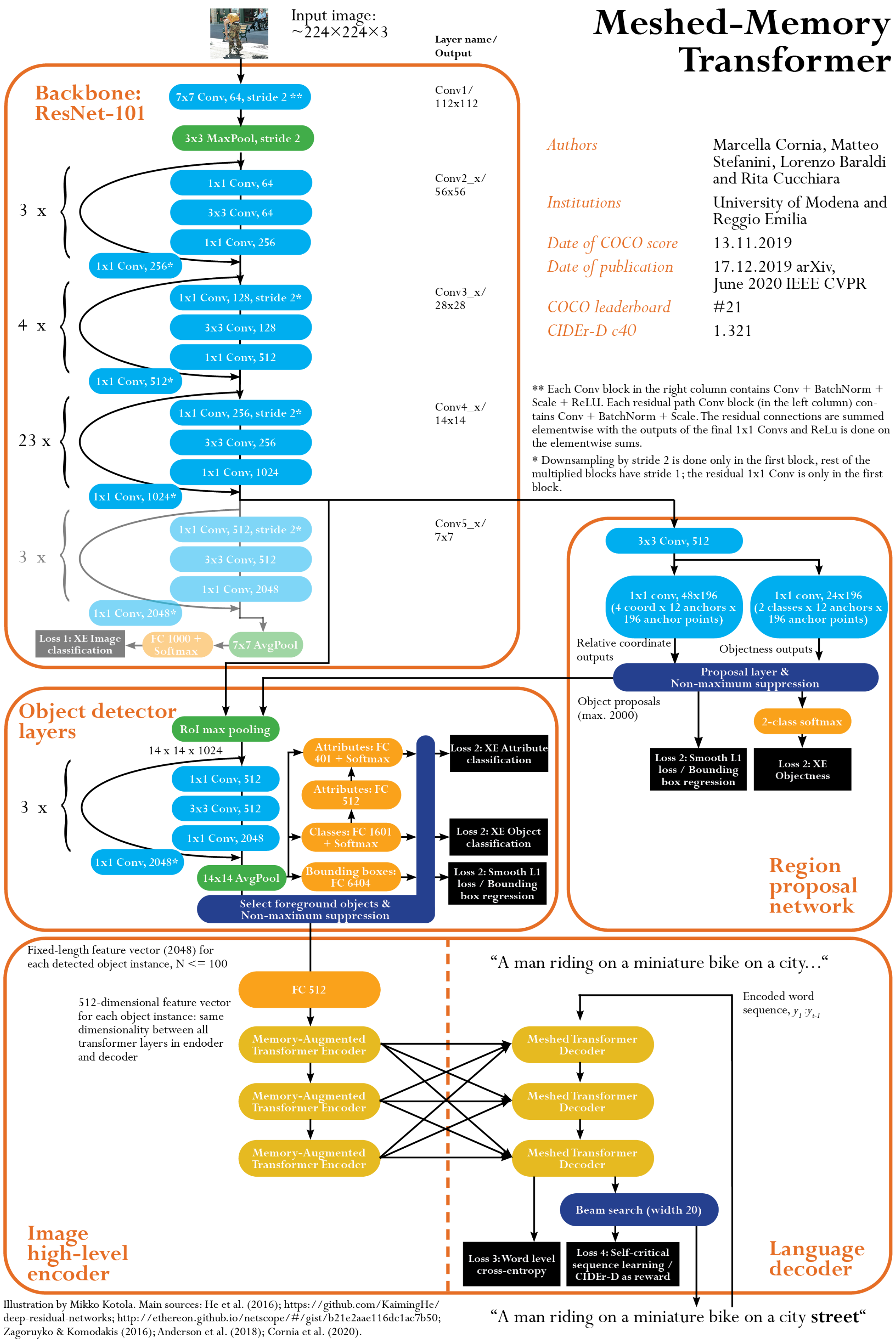
Main contributions. The AoA model is an improvement to the plain transformer block. AoA blocks are stackable just like plain transformer blocks. Therefore the AoA can be utilized as a variation in any transformer-based models. The main contribution of AoA to the image captioning task is utilizing it in the decoder to disregard image features in those states where they are meaningless, and learning to attend only to feature channels that are useful given the current language decoder context. This result is acknowledged, and later methods have built on and extended this approach.

Meshed-Memory Transformer

Authors	Marcella Cornia, Matteo Stefanini, Lorenzo Baraldi and Rita Cucchiara
Institutions	University of Modena and Reggio Emilia
Date of COCO score	13.11.2019
Date of publication	17.12.2019 arXiv, June 2020 IEEE CVPR
COCO leaderboard	#21
CIDEr-D c40	1.321

** Each Conv block in the right column contains Conv + BatchNorm + Scale + ReLU. Each residual path Conv block (in the left column) contains Conv + BatchNorm + Scale. The residual connections are summed elementwise with the outputs of the final 1x1 Convs and ReLu is done on the elementwise sums.

* Downsampling by stride 2 is done only in the first block, rest of the multiplied blocks have stride 1; the residual 1x1 Conv is only in the first block.



Input image: ~224×224×3
Layer name/ Output
Backbone: ResNet-101
7x7 Conv, 64, stride 2 **
Conv1/ 112x112
3x3 MaxPool, stride 2
3 x
1x1 Conv, 64
3x3 Conv, 64
1x1 Conv, 256
1x1 Conv, 256*
Conv2_x/ 56x56
4 x
1x1 Conv, 128, stride 2*
3x3 Conv, 128
1x1 Conv, 512
1x1 Conv, 512*
Conv3_x/ 28x28
23 x
1x1 Conv, 256, stride 2*
3x3 Conv, 256
1x1 Conv, 1024
1x1 Conv, 1024*
Conv4_x/ 14x14
3 x
1x1 Conv, 512, stride 2*
3x3 Conv, 512
1x1 Conv, 2048
1x1 Conv, 2048*
Conv5_x/ 7x7
Loss 1: XE Image classification
FC 1000 + Softmax
7x7 AvgPool
3x3 Conv, 512
1x1 conv, 48x196 (4 coord x 12 anchors x 196 anchor points)
1x1 conv, 24x196 (2 classes x 12 anchors x 196 anchor points)
Relative coordinate outputs
Objectness outputs
Proposal layer & Non-maximum suppression
Object proposals (max. 2000)
2-class softmax
Loss 2: Smooth L1 loss / Bounding box regression
Loss 2: XE Objectness
Region proposal network
Object detector layers
RoI max pooling
14 x 14 x 1024
3 x
1x1 Conv, 512
3x3 Conv, 512
1x1 Conv, 2048
1x1 Conv, 2048*
14x14 AvgPool
Attributes: FC 401 + Softmax
Attributes: FC 512
Classes: FC 1601 + Softmax
Bounding boxes: FC 6404
Loss 2: XE Attribute classification
Loss 2: XE Object classification
Loss 2: Smooth L1 loss / Bounding box regression
Select foreground objects & Non-maximum suppression
Fixed-length feature vector (2048) for each detected object instance, N <= 100
FC 512
512-dimensional feature vector for each object instance: same dimensionality between all transformer layers in endoder and decoder
Memory-Augmented Transformer Encoder
Memory-Augmented Transformer Encoder
Memory-Augmented Transformer Encoder
Image high-level encoder
"A man riding on a miniature bike on a city..."
Encoded word sequence, $y_1:y_{t-1}$
Meshed Transformer Decoder
Meshed Transformer Decoder
Meshed Transformer Decoder
Beam search (width 20)
Loss 3: Word level cross-entropy
Loss 4: Self-critical sequence learning / CIDEr-D as reward
Language decoder
"A man riding on a miniature bike on a city **street**"

Illustration by Mikko Kotola. Main sources: He et al. (2016); https://github.com/KaimingHe/deep-residual-networks; http://ethereon.github.io/netscope/#/gist/b21e2aae116dc1ac7b50; Zagoruyko & Komodakis (2016); Anderson et al. (2018); Cornia et al. (2020).

Figure 4.7: Architecture of the complete Meshed-Memory Transformer system.

4.2 Meshed-Memory Transformer

Cornia et al. [12] present the Meshed-Memory Transformer ($\mathcal{M}^2$) model, which uses several transformer-based blocks to encode a multi-level representation of the relationships between detected objects, and then uses mesh-like connections between the encoder and the also transformer-based decoder to exploit both low- and high-level features when generating captions. Additionally, the encoder is augmented with memory, effectively leading the encoder to generalize and learn representations that are useful across images. The architecture of the $\mathcal{M}^2$ encoder-decoder is shown in figure 4.8, and the architecture of the whole Meshed-Memory Transformer captioning system is presented in figure 4.7. The Meshed-Memory Transformer is currently #21 on the COCO leaderboard, with CIDEr-D c40 score of 1.321. At the time of its testing on the evaluation server (November 13th 2019), it was ranked #1 on the leaderboard.

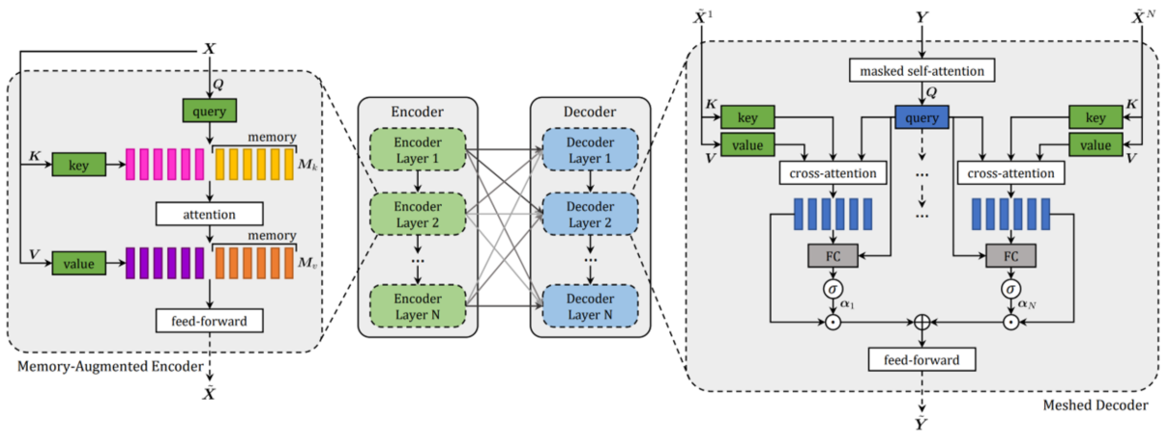


Figure 4.8: Architecture of the high-level image encoder and decoder. AddNorm operations are not shown in the image to make the image more understandable. Image source: Cornia et al. [12].

Object detection. Just like AoANet, The $\mathcal{M}^2$ model relies on object detections and their feature vectors from the Bottom-Up Top-Down [2] model, meaning the backbone of the object detector is ResNet-101 [22], there is a region proposal network to select the object areas, and a set of object detection layers. The high-level image encoder takes as input 1 to n feature vectors of dimension 2048, one for each detected object instance in the image. The Bottom-Up model outputs a maximum of 100 object instances per image, so that is the practical upper limit for $\mathcal{M}^2$ also. The proper domain training and enrichment of the feature vectors done by Anderson et al. [2] is central to the quality of the method as a whole.

Memory vectors. The two unique contributions of the model are usage of persistent memory vectors in the image encoder blocks, and the mesh-like connections between encoder and decoder blocks, with a learnable gating mechanism. The role of the persistent memory vectors in the encoder blocks is, according to the authors, learning and encoding *a priori* knowledge [12]. The use of the term *a priori* is perhaps somewhat misguided, since the term is usually used to refer to purely analytical knowledge, knowledge which is independent of experience. What is meant by the term in this context is that the memory vectors encode features which are independent of the *current* input image. The encoded information is learned from the training images during the training phase, when the parameters of the memory vectors are trained using backpropagation. But during inference, the weights of the memory features do not depend on the input image - their role is to act as additional, general keys and values within the encoder blocks. In the ablation study [13], different amounts of parallel memory vectors, from 0 to 80, are compared. Best performance is reached with 40 memory vectors, increasing the CIDEr score from 129.4 with no memory to 131.2 with 40 memory vectors. This memory technique within a transformer-based encoder block appears generalizable, and could benefit also other models.

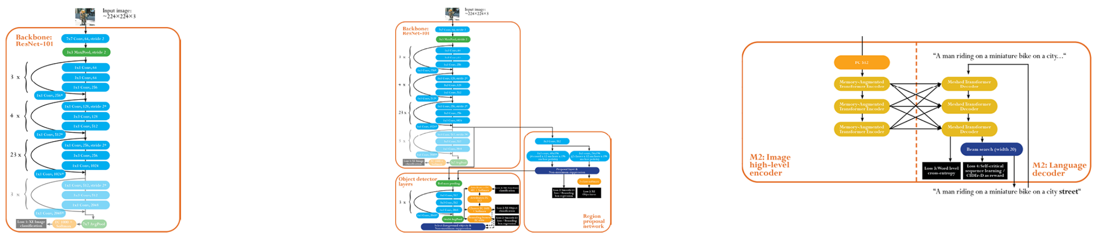
Meshed connections between encoder and decoder. The mesh-like connections between encoder and decoder blocks is a very interesting contribution from the architecture perspective. The connections enable the system to exploit both low-level and high-level features in generating the caption, in practice learning to focus attention on single objects or their features, and the connections between objects and higher level image context, based on the language decoder state. Through an ablation study [12], it is shown that using 1-to-1 connections (highest encoder layer to highest decoder layer, lowest encoder layer to lowest decoder layer), in comparison to only connecting the highest encoder layer to the lowest decoder layer like in the original transformer model [53], already improves the CIDEr score from 123.6 to 129.2. The meshed connections with learnable gating using sigmoid activation functions further improve the CIDEr score from 129.2 to 131.2. This mesh-based architectural pattern is likely generalizable and can benefit also other transformer encoder-decoder captioning models. The mesh connections with learnable gating can be seen as an extension of the Attention on Attention [25] method, and the authors of $\mathcal{M}^2$ are aware of AoA's proven benefits.

Stacking of encoder and decoder blocks. The $\mathcal{M}^2$ model uses three layers of encoder blocks and three layer of decoder blocks. Cornia et al. report experimenting with 2, 3 and 4 layers of encoder and decoder layers [13], and also comparing three- and six-layered versions that use simplified transformer encoder blocks (in practice using the original transformer blocks by Vaswani et al. [53]). They reach best results with the three-layered version. Their hypothesis is that three layers are optimal in the captioning setting, due to the relatively low semantic complexity of sentences and the relatively small size of the training set in comparison to machine translation, where six-layer models were used [12].

Training. The $\mathcal{M}^2$ model is trained in two phases. The main training phase is supervised learning, and uses word-level cross-entropy loss to learn to output the next word of the caption using the previous words from the ground truth caption. As the ground truth captions are fully available, the computation of all words in an output sequence (image caption) can be done in a single pass, with all operations done in parallel. The second phase of the training fine-tunes the sequence generation using reinforcement learning with the CIDEr-D as the optimization objective. In practice, it is done using the self-critical sequence training approach [41] and beam search.

A visualization of the training process of the whole $\mathcal{M}^2$ image captioning system is shown in figure 4.9. All leading image captioning systems use a similar 3-stage training process. In the first phase the backbone network (ResNet-101) is trained on ImageNet-1k using single-label image classification loss. At this phase, the backbone learns the lower-level features of objects in images and higher-level features that enable identifying it as a representative of some object class. In the second phase, the object detector (Faster R-CNN with Bottom-Up detailed features) is trained using the Visual Genome dataset. There are a total of five loss functions in this stage (see a more detailed picture in figure 4.7). In the region proposal network, there are loss functions for bounding box regression (where the anchor box is located in relation to an anchor) and objectness (is there a foreground object present in the anchor box or not). In the object detection layers, the final layers of the ResNet-101 are taken as a starting point, and are further trained at this stage using three loss functions: one for object classes, a second for object features and a third for the bounding box size and location. After the second stage of training, the object detector is fully trained and can output 2048-dimensional feature vectors for each detected object region in the image. The feature vectors contain information not just related to the object classes, but also their more detailed features and interrelations, thanks to the object detector being trained on the rich Visual Genome dataset. In the third stage of training only the image captioner components (encoder and decoder) are trained. As described above, there are two training phases in this stage, the main phase of word-level cross-entropy training, and the finetuning phase that uses reinforcement learning with the CIDEr-D as the optimization objective.

Main contributions. The main contributions of the Meshed-Memory Transformer are demonstrating a fully transformer-based encoder and decoder, using persistent memory vectors in the encoder, and using mesh-like connections between the encoder and decoder blocks to enable the system to exploit both low-level and high-level features in generating the caption, The transformer-based decoder is faster to



i. Train backbone

Train ResNet-101

Dataset: ImageNet images / train 1.28M, eval 50k, test 100k

Params: 44,5M

FLOPS: 7.6×109

Training phases:
1. ResNet-101 with Imagenet with image classification loss

ii. Train object detector

Train Faster R-CNN (including ResNet-101 layers)

Dataset: Visual Genome dataset / train 95k, eval 5k, test 5k

Training phases:
2. Train Faster R-CNN end to end (including ResNet-101 layers) using 5 loss functions related to object classes, bounding boxes and object attributes

iii. Train captioner

Train high-level encoder and language decoder

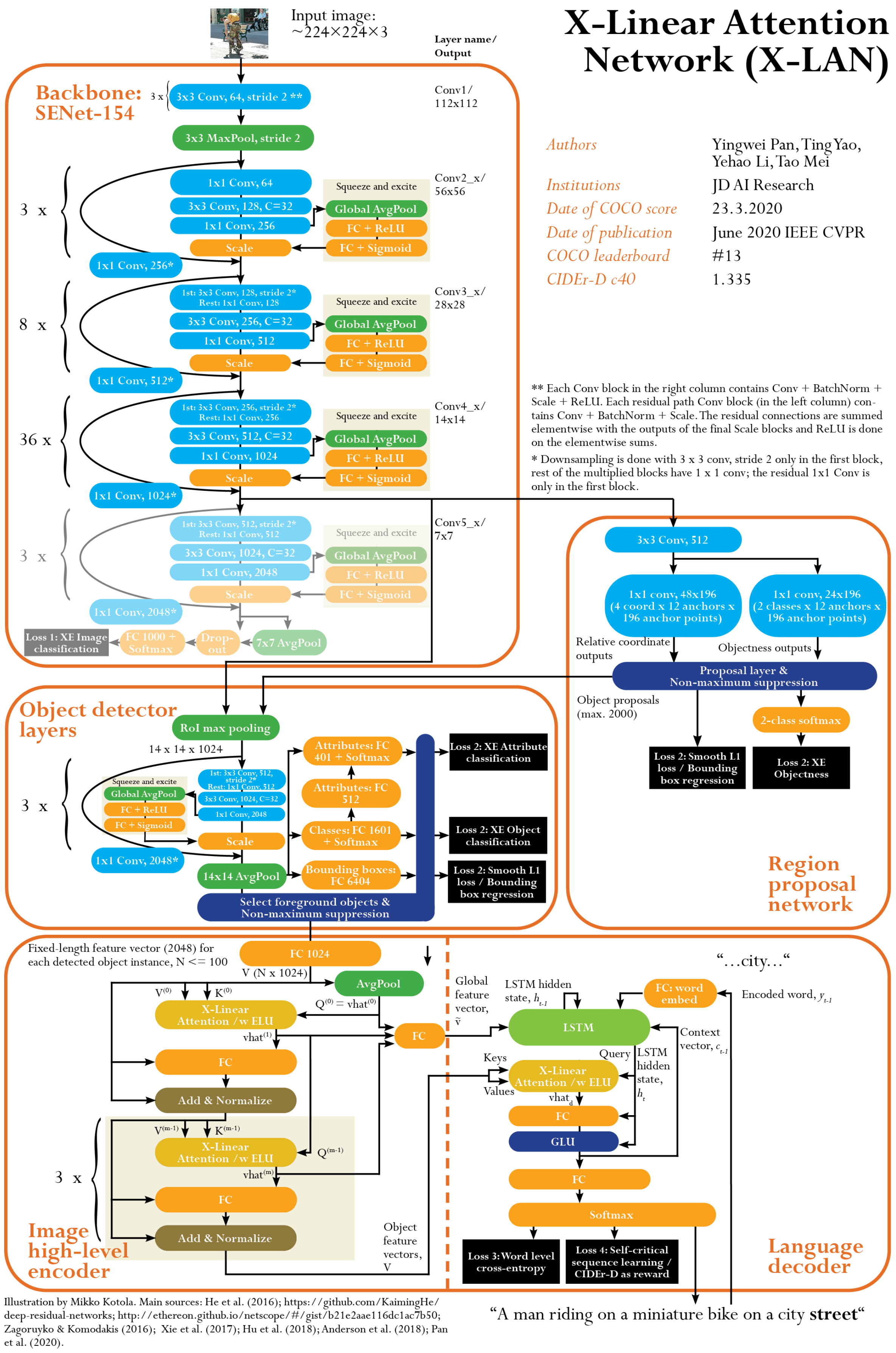
Dataset: COCO images & captions / train 118k, eval 5k, test 5k

Training phases:
3. Supervised training using word-level cross-entropy loss and ground truth captions
4. Reinforcement learning: self-critical sequence learning using beam search (width 20) and CIDEr-D c40 as objective

Main sources: He et al. (2016); Zagoruyko & Komodakis (2016); Cornia et al. (2020); Anderson et al. (2018)

Figure 4.9: Training of the whole Meshed-Memory Transformer system. The same three-stage training approach is used in all leading image captioners under investigation. The last stage has two phases, word-level cross-entropy training and sequence-level reinforcement training.

train than LSTM-based decoders due to the fact that all steps of the word-level cross-entropy training phase can be done in parallel. Also, the usage of beam search in the reinforcement learning phase and in the inference mode improves the fluency of the captions.



X-Linear Attention Network (X-LAN)

Authors	Yingwei Pan, Ting Yao, Yehao Li, Tao Mei
Institutions	JD AI Research
Date of COCO score	23.3.2020
Date of publication	June 2020 IEEE CVPR
COCO leaderboard	#13
CIDEr-D c40	1.335

** Each Conv block in the right column contains Conv + BatchNorm + Scale + ReLU. Each residual path Conv block (in the left column) contains Conv + BatchNorm + Scale. The residual connections are summed elementwise with the outputs of the final Scale blocks and ReLU is done on the elementwise sums.

* Downsampling is done with 3 x 3 conv, stride 2 only in the first block, rest of the multiplied blocks have 1 x 1 conv; the residual 1x1 Conv is only in the first block.

Illustration by Mikko Kotola. Main sources: He et al. (2016); https://github.com/KaimingHe/deep-residual-networks; http://ethereon.github.io/netscope/#/gist/b21e2aae116dc1ac7b50; Zagoruyko & Komodakis (2016); Xie et al. (2017); Hu et al. (2018); Anderson et al. (2018); Pan et al. (2020).

Figure 4.10: Architecture of the complete X-Linear Attention Network system. Note that the backbone is SENet-154.

4.3 X-Linear Attention Network

Pan et al. [35] present a captioning model called X-Linear Attention Network. The model is based on an new, enhanced attention-based block called an X-Linear Attention (X-LA) block. X-Linear Attention seeks to take advantage of higher-order interactions both within image object detections and between the image content and natural language context.

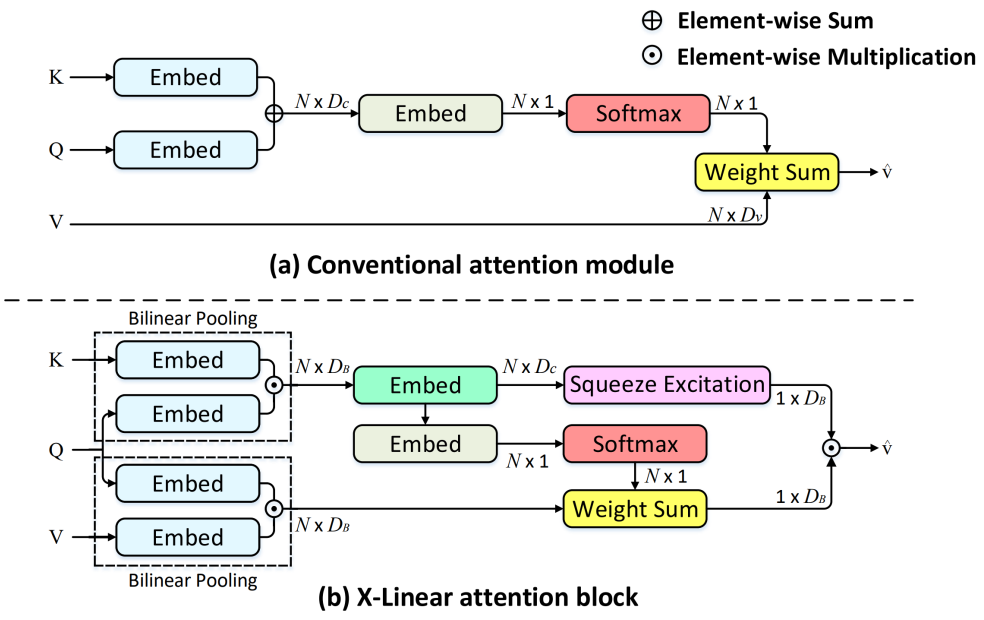


Figure 4.11: Architecture of the X-Linear Attention block and comparison to plain attention. The squeeze excitation route produces channel-wise attention and the softmax route produces spatial attention, meaning which refined object feature vectors to attend to. Image source: Pan et al. [35].

X-Linear Attention block. X-Linar Attention uses bilinear pooling to calculate the outer product of the key and the query to take into account all pairwise interactions between the query and the key. Within the X-LA block, after bilinear pooling, two embedding layers are used to predict attention weights for each spatial region (object detection). Then softmax is used to normalize the spatial attention vector. Additionally, the embedded outer product feature map is passed through a squeeze-excitation operation to aggregate the feature map across spacial regions and produce a channel-wise attention vector.

As visualized in figure 4.11, the X-Linear attention block extends the plain attention block. The main difference is using bilinear pooling to model in channel-level detail the interactions between the query $\boldsymbol{Q}$ and each key vector (transformed object detection feature vector) $\boldsymbol{k}_i$ on the one hand, and the query $\boldsymbol{Q}$ and each value vector (transformed object detection feature vector) $\boldsymbol{v}_i$ on the other.

The initial query-key bilinear pooling is calculated

$$\boldsymbol{B}_i^k = \text{ReLU}(\boldsymbol{W}_k \boldsymbol{k}_i) \odot \text{ReLU}(\boldsymbol{W}_q^k \boldsymbol{Q}) \tag{4.7}$$

where $\boldsymbol{W}_k \in \mathbb{R}^{D_B \times D_k}$ and $\boldsymbol{W}_q^k \in \mathbb{R}^{D_B \times D_q}$ are learnable embeddings, ReLU is the standard rectified linear unit activation $(\text{ReLU}(x) = \max(0, x))$ and $\odot$ is element-wise multiplication. The learned bilinear query-key representation $\boldsymbol{B}_i^k$ carries the second order feature interactions between query and the key matrix.

$\boldsymbol{B}_i^k$ is then used to calculate the attention between the bilinearly transformed keys by using two embedding layers and a softmax for normalization:

$$\boldsymbol{B}_i'^k = \sigma(\boldsymbol{W}_B^k \boldsymbol{B}_i^k) \tag{4.8}$$

$$\boldsymbol{b}_i^s = \boldsymbol{W}_b \boldsymbol{B}_i'^k \tag{4.9}$$

$$\boldsymbol{\beta}^s = \text{softmax}(\boldsymbol{b}^s) \tag{4.10}$$

where $\boldsymbol{B}_i'^k$ are transformed query-key vectors, $\boldsymbol{W}_B^k \in \mathbb{R}^{D_c \times D_B}$ and $\boldsymbol{W}_b \in \mathbb{R}^{1 \times D_C}$ are embeddings. Each element $\boldsymbol{\beta}_i^s$ in $\boldsymbol{\beta}^s$ is the normalized spacial attention weight for each key/value pair.

To calculate the channel-wise attention $\boldsymbol{\beta}^c$, the transformed query-key vectors $\boldsymbol{B}_i'^k$ are first average-pooled over all N object detections to get a global channel descriptor $\bar{\boldsymbol{B}}$:

$$\bar{\boldsymbol{B}} = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{B}_i'^k \tag{4.11}$$

Then an excitation operation, self-gating with a sigmoid, is used to calculate channel-wise attention distribution $\boldsymbol{\beta}^c$:

$$\boldsymbol{\beta}^c = \sigma(\boldsymbol{W}_e \bar{\boldsymbol{B}}) \tag{4.12}$$

where $\boldsymbol{W}_e \in \mathbb{R}^{D_B \times D_C}$ is an embedding matrix.

On the value branch of the process, the interaction of the values and queries are modeled using bilinear pooling:

$$\boldsymbol{B}_i^v = \text{ReLU}(\boldsymbol{W}_v \boldsymbol{v}_i) \odot \text{ReLU}(\boldsymbol{W}_q^v \boldsymbol{Q}) \tag{4.13}$$

where $\boldsymbol{W}_v \in \mathbb{R}^{D_B \times D_v}$ and $\boldsymbol{W}_q^v \in \mathbb{R}^{D_B \times D_q}$ are embedding matrices.

Finally the enhanced value vectors $\boldsymbol{B}_i^v$ are summed with spatial attention vectors $\boldsymbol{\beta}^s$ and then multiplied element-wise with the channel-wise attention distribution to get the final attended feature vectors:

$$\hat{\boldsymbol{v}} = F_{\text{X-Linear}}(\boldsymbol{K}, \boldsymbol{V}, \boldsymbol{Q}) = \boldsymbol{\beta}^c \odot \sum_{i=1}^{N} \boldsymbol{\beta}_i^s \boldsymbol{B}_i^v \tag{4.14}$$

In order to learn higher-order interactions, X-LA blocks are extended with Exponential Linear Unit (ELU) [4] nonlinearity within the bilinear pooling component [35]. This approach is argued to reduce the need to stack large numbers of X-LA block to model higher-order interaction between the input object instance feature vectors. The ELU-augmented version of the X-LA block is visualized in figure 4.12.

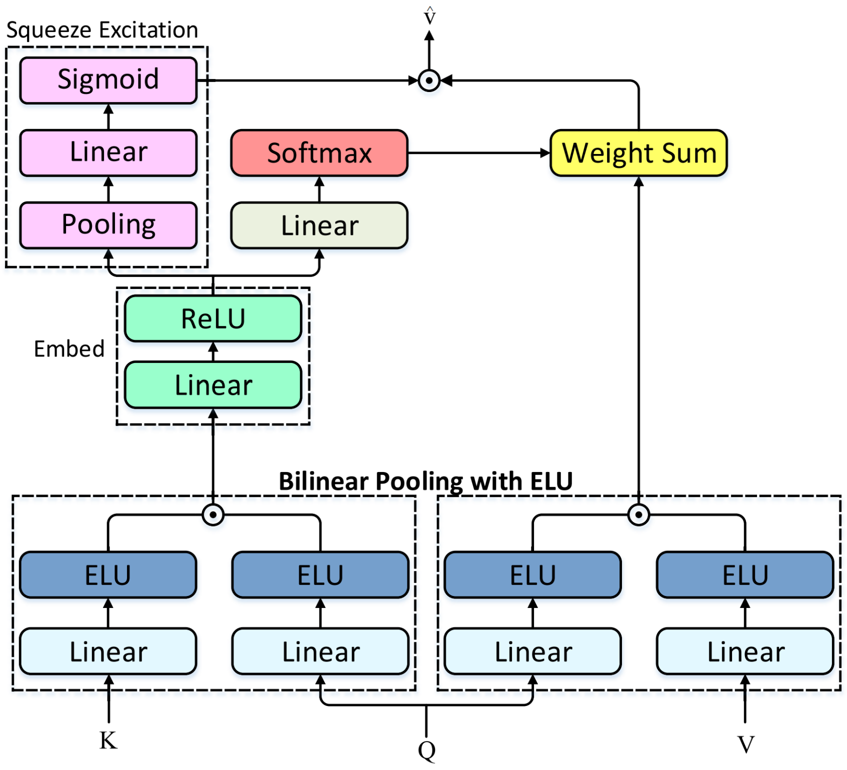


Figure 4.12: Architecture of the X-Linear attention block with Exponential Linear Unit (ELU) nonlinearity. The ELU-variant can approximately model infinity order feature interactions between inputs. Image source: Pan et al. [35].

X-Linear Attention Network. X-Linear Attention blocks are used in an LSTM-based image captioning network to build an X-Linear Attention Network (X-LAN) model [35]. The architecture of the whole X-LAN image captioning system is shown in figure 4.10. Again, the method relies on the Bottom-Up Top-Down object detections and features as input into the image captioner. The 2048-dimensional object feature vectors are transformed to 1024-dimensional vectors. This dimensionality is then used between the encoder and decoder blocks. Within an X-LA block, the bilinear query-key representation is 1024-dimensional and the transformed bilinear query-value is 512-dimensional. Four stacked X-Linear Attention blocks with ELU are used in the high-level image encoder to learn a set of enhanced region-level and image-level features (see figure 4.10). Each X-LA block takes the attended feature vector $\hat{\boldsymbol{v}}^{(m-1)}$ of the previous block as the input query $\boldsymbol{Q}^{(m-1)}$. In addition to the query, also the input keys $\boldsymbol{K}^{(m-1)} = \{\boldsymbol{k}_i^{(m-1)}\}_{i=1}^N$ and input values $\boldsymbol{V}^{(m-1)} = \{\boldsymbol{v}_i^{(m-1)}\}_{i=1}^N$ are taken from the output of the previous block. In each block, there is a fully connected embedding layer and an add and normalize layer to update the keys and values.

$$\hat{\boldsymbol{v}}^{(m)} = F_{\text{X-Linear}}(\boldsymbol{K}^{(m-1)}, \boldsymbol{V}^{(m-1)}, \hat{\boldsymbol{v}}^{(m-1)}) \tag{4.15}$$

$$\boldsymbol{k}_i^{(m)} = \text{LayerNorm}(\sigma(\boldsymbol{W}_m^k[\hat{\boldsymbol{v}}^{(m)}, \boldsymbol{k}^{(m-1)}]) + \boldsymbol{k}^{(m-1)}) \tag{4.16}$$

$$\boldsymbol{v}_i^{(m)} = \text{LayerNorm}(\sigma(\boldsymbol{W}_m^v[\hat{\boldsymbol{v}}^{(m)}, \boldsymbol{v}^{(m-1)}]) + \boldsymbol{v}^{(m-1)}) \tag{4.17}$$

where $\boldsymbol{W}_m^k$ and $\boldsymbol{W}_m^v$ are embedding matrices.

It is important to observe that the initial query $\hat{\boldsymbol{v}}^{(0)}$ is calculated by average pooling all the input object detection feature vectors. This vector is a global feature vector, carrying all the features present in the detected object areas in a condensed form. At each block, another higher-level global feature vector $\hat{\boldsymbol{v}}^{(m)}$ is calculated. These five global feature vectors (the initial $\hat{\boldsymbol{v}}^{(0)}$ and one from each encoder block) are concatenated and transformed using a fully connected layer into a final global image-level feature vector $\tilde{\boldsymbol{v}}$:

$$\tilde{\boldsymbol{v}} = \boldsymbol{W}_G[\hat{\boldsymbol{v}}^{(0)}, \hat{\boldsymbol{v}}^{(1)}, ..., \hat{\boldsymbol{v}}^{(m)}] \tag{4.18}$$

where $\boldsymbol{W}_G$ is an embedding matrix. The global feature vector is one of the two inputs into the decoder. The other input into the decoder is the set of refined object feature vectors $\boldsymbol{V}$ that is output from the final encoder block.

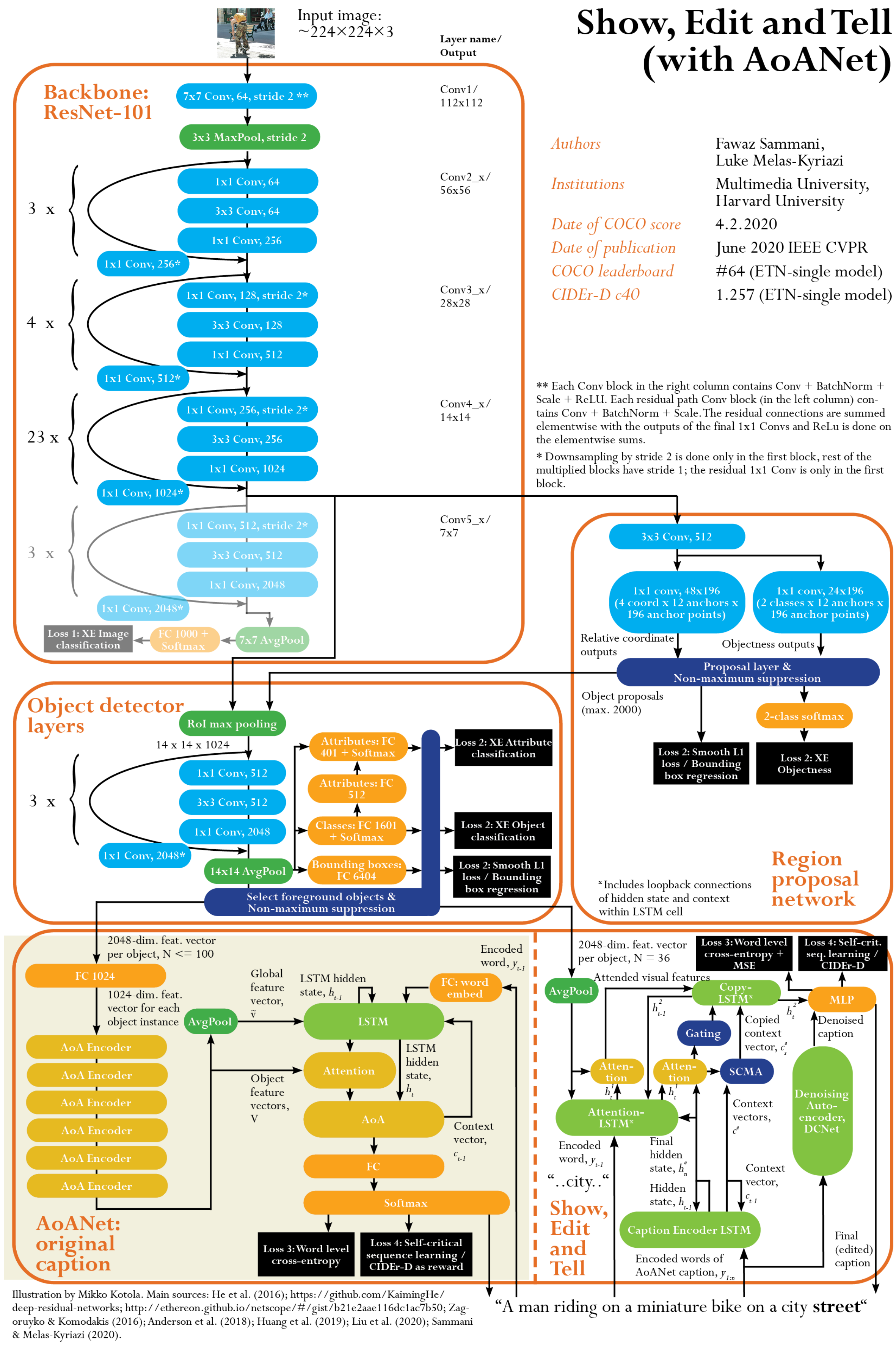
On the decoder side, an LSTM takes as input the global feature vector $\tilde{\boldsymbol{v}}$, the current input word $\boldsymbol{w}_t$, the previous LSTM hidden state $\boldsymbol{h}_{t-1}$ and the previous context vector $\boldsymbol{c}_{t-1}$. Then the output of the LSTM $\boldsymbol{h}_t$ is used as the query input into an X-LA with ELU block, and the enhanced object feature vectors $\boldsymbol{V}$ are used as keys and values. The output of the X-LA block $\hat{\boldsymbol{v}}_d$ is concatenated and embedded with the LSTM hidden state $\boldsymbol{h}_t$ using a fully connected layer, and the result is again concatenated with $\boldsymbol{h}_t$ and fed into a Gated Linear Unit (GLU). The output from the GLU is the context vector. The context vector $\boldsymbol{c}_t$ is then fed into a fully connected layer and softmax over the vocabulary is used to predict the next word to output. The context vector is also fed back to the LSTM as input to the next time step (to predict the next word in the caption).

The role of the X-Linear Attention block with ELU in the decoder is to explore the higher-order inter-modal interactions between visual content and natural language context to boost caption generation.

Training. The X-LAN model does not introduce new innovations in the training procedure, and follows the already previously described practice of two-phase training of the image captioner components. The main training phase is supervised learning with word-level cross-entropy loss, and the second phase is reinforcement learning with the CIDEr-D as the optimization objective using the self-critical sequence training approach.

Variations of the backbone and image captioner network. Pan et al. [35] conduct experiments with two different backbone CNNs, the traditional ResNet-101 and SENet-154. SENet-154 [24] is explained in detail later, in section 5.2. The results show that SENet-154 brings improvements to the CIDEr-D score: for the X-LAN the backbone change improves CIDEr-D from 1.303 to 1.328, and for the X-Transformer from 1.314 to 1.335. This in an important result and supports the hypothesis that utilizing innovations in the backbone and object detection networks can bring improvements to the image captioning systems as a whole. The original LSTM-based X-Linear Attention Network achieves high scores on the COCO online server, but X-LA blocks are also integrated into a transformer-based captioner to demonstrate the generalizability and further improve the CIDEr-D score. The usage of a transformer-based image captioning network over an LSTM-based on top of the SENet-154 backbone brings a modest, likely not statistically significant, improvement of 0.007 points, from 1.328 to 1.335. The complete architecture of an LSTM-based X-Linear Network with a SENet-154 backbone is shown in figure 4.10. Note that the architecture figure shows the LSTM-based version, while the listed X-LAN CIDEr-D score is for the transformer-based version.

Main contributions. The most important contribution of the X-Linear Attention Network is the enhanced X-Linear Attention block, which can model interdependencies both within the object feature vectors and between the object features and the language context. The other contribution is showing that changing backbone CNNs can improve system performance. Usage of the Exponential Linear Unit in the encoder to model higher-order interactions and the Gated Linear Unit in the decoder are also distinctive choices in relation to the other image captioners under investigation.



Show, Edit and Tell (with AoANet)

Authors	Fawaz Sammani, Luke Melas-Kyriazi
Institutions	Multimedia University, Harvard University
Date of COCO score	4.2.2020
Date of publication	June 2020 IEEE CVPR
COCO leaderboard	#64 (ETN-single model)
CIDEr-D c40	1.257 (ETN-single model)

** Each Conv block in the right column contains Conv + BatchNorm + Scale + ReLU. Each residual path Conv block (in the left column) contains Conv + BatchNorm + Scale. The residual connections are summed elementwise with the outputs of the final 1x1 Convs and ReLu is done on the elementwise sums.

* Downsampling by stride 2 is done only in the first block, rest of the multiplied blocks have stride 1; the residual 1x1 Conv is only in the first block.

ˣ Includes loopback connections of hidden state and context within LSTM cell

Illustration by Mikko Kotola. Main sources: He et al. (2016); https://github.com/KaimingHe/deep-residual-networks; http://ethereon.github.io/netscope/#/gist/b21e2aae116dc1ac7b50; Zagoruyko & Komodakis (2016); Anderson et al. (2018); Huang et al. (2019); Liu et al. (2020); Sammani & Melas-Kyriazi (2020).

Figure 4.13: Architecture of the complete Show, Edit and Tell system. This is the version reported in the COCO online test server leaderboard, that uses AoANet as primary captioner.

4.4 Show, Edit and Tell

Sammani and Melas-Kyriazi [44] approach the captioning task from a quite different direction than the other leading captioners. Instead of focusing on how to generate a final caption directly from the NLG decoder, they focus on iterative adaptive refinement of an existing caption. The iterative editing approach is used in other NLG tasks such as sequence-to-sequence and data-to-text generation [44, 28], but is not commonly utilized in leading image captioners. The approach aims to improve on the errors, where a captioner produces incorrect, inconsistent, or repetitive content. It does this by adding two new modules to any existing captioner to refine a caption through iterative editing. The new editing modules are directly compatible with any captioner. The added modules are EditNet, an LSTM- and attention-based network for adaptively copying data from the input caption, and DCNet, an LSTM-based denoising autoencoder. The version submitted to the COCO challenge uses AoANet as the primary captioner, and further edits the captions produced by AoANet. The architecture of the whole Show, Edit and Tell image captioning system is presented in figure 4.13. Notable is that the Show, Edit and Tell part of the system takes as input the image features output by the Bottom-Up object detector, and uses the precalculated features with a fixed number of object detections (36).

The CIDEr-D c40 score of Show, Edit and Tell (ETN-single_model) on the COCO online test server does not exceed those of the other networks under investigation. Part of the score is explained by the Show, Edit and Tell model being submitted as a single model, whereas the four other captioners report their highest score for an ensemble of four models. However, even in the reported offline evaluation CIDEr scores on the Karpathy test split, the Show, Edit and Tell only matches AoANet's scores, but does not exceed them.

One type of mistake the editing approach proposes to fix is repetition of same nouns in the caption several times. An example of this kind of mistake is given in figure 4.14. Here, AoANet has output a caption "A sandwich on a table with a table.", mistakenly repeating the noun "table". The sentence structure of the caption is fluent, but the latter noun is not correct. A caption-editing model should be able to recognize the noun repetition and modify the caption to be something like "A sandwich on a table with a glass of wine".

EditNet. EditNet [44] is a model that is taught to copy or edit each word in an input caption. The caption is first encoded using a unidirectional one-layer LSTM (dashed red box in figure 4.15, which shows the EditNet components in detail). The encoded input caption $\bar{\boldsymbol{h}}_s = [\boldsymbol{h}^e_1, \boldsymbol{h}^e_2, ..., \boldsymbol{h}^e_n]$ contains n tokens, where n equals the number of words in the input caption. The Attention-LSTM takes as input the

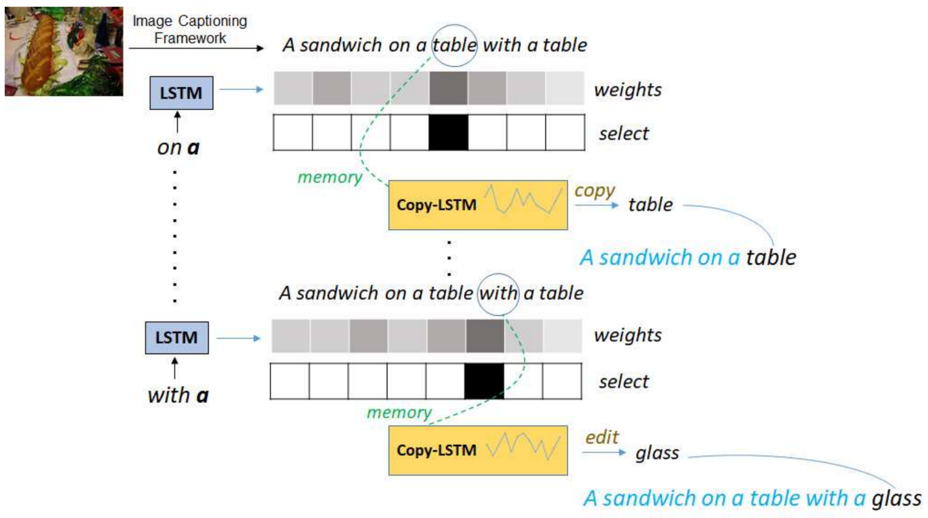


Image Captioning Framework
A sandwich on a table with a table
LSTM
weights
select
on a
memory
Copy-LSTM
copy
table
A sandwich on a table
A sandwich on a table with a table
LSTM
weights
select
with a
memory
Copy-LSTM
edit
glass
A sandwich on a table with a glass

Figure 4.14: The Show, Edit and Tell network takes as input a generated caption. At each decoding time step, the system generates attention weights (grey) indicating the importance of each word in the existing caption for the current word in the new caption. A selective copy memory attention (SCMA) operation is used to select the most likely word and directly copy its LSTM memory state to the Copy-LSTM. The new caption is then generated using the Copy-LSTM state. Image source: Sammani and Melas-Kyriazi [44].

concatenation of the word embedding vector $\boldsymbol{y}_{t-1}$ (marked x in figure 4.15), the last hidden state of the caption encoder $\boldsymbol{h}_n^e$, the mean-pooled image features $\bar{\boldsymbol{v}} = \frac{1}{k}\sum_i \boldsymbol{v}_i$, and the previous hidden state of the language LSTM $\boldsymbol{h}_{t-1}^2$:

$$\boldsymbol{x}_t^1 = [\boldsymbol{w}_t; \boldsymbol{h}_n^e; \bar{\boldsymbol{v}}; \boldsymbol{h}_{t-1}^2] \tag{4.19}$$

The output of the attention LSTM $\boldsymbol{h}_t^1$ is then used to calculate one attention vector over the visual features (red arrow in figure 4.15) and another attention vector over the textual features (blue arrow in figure 4.15). The two attention vectors are fused with a gating mechanism and input to the Copy-LSTM. The attention weights over textual features are also used as input $\boldsymbol{\alpha}_p$ to the Selective Copy Memory Attention (SCMA) module. The role of the SCMA module is to learn to select and copy states from the input caption LSTM. The SCMA does this by measuring the similarity between $\boldsymbol{h}_t^1$ and and each word in the previous caption $\bar{\boldsymbol{h}}_s$:

$$\boldsymbol{\alpha}_p = \text{softmax}(\boldsymbol{W}_a^T \tanh(\boldsymbol{W}_s \bar{\boldsymbol{h}}_s + \boldsymbol{W}_h \boldsymbol{h}_t^1) \tag{4.20}$$

where $\boldsymbol{W}_a^T$, $\boldsymbol{W}_s$ and $\boldsymbol{W}_h$ are embedding matrices and tanh is the hyperbolic tangent function. The SCMA then picks the word with the highest softmax output[1] from α_p (implying highest similarity to the hidden state of the Attention LSTM and therefore

[1]Details on the mathematical implementation of the SCMA operation are omitted here for brevity, refer to the original paper [44] for details.

the word being currently generated in the language model), and copies the corresponding memory state $\boldsymbol{c}_t^e$ from the input caption encoder.

The output of the SCMA $\boldsymbol{c}_s^e$ is used as one input into the Copy-LSTM. The Copy-LSTM is an LSTM cell with an added adaptive copy mechanism. It includes a copy gate that controls how much information is taken from the SCMA module ($\boldsymbol{c}_s^e$) relative to the other input sources, the attended visual features and the hidden state $\boldsymbol{h}_t^1$. The copy gate value c_{g_t} is calculated:

$$c_{g_t} = \sigma(\boldsymbol{W}_n \cdot [\boldsymbol{c}_t; \boldsymbol{c}_s^e]) \tag{4.21}$$

where $\boldsymbol{W}_n$ is an embedding matrix, $\boldsymbol{c}_t$ is the cell state vector of the LSTM, and $[;]$ is concatenation. The copy gate value is then used to calculate a modified LSTM memory state:

$$\boldsymbol{c}_{ap_t} = c_{g_t} * \boldsymbol{c}_s^e + (1 - c_{g_t}) * \boldsymbol{c}_t \tag{4.22}$$

The hidden state is then computed with a tanh activation function of the modified memory state multiplied by the output gate:

$$\boldsymbol{h}_t^2 = \boldsymbol{o}_t * \tanh(\boldsymbol{c}_{ap_t}) \tag{4.23}$$

If the gate c_{g_t} is 1, then the word from the input caption is copied as such, and if it is 0, then the word in the previous caption is ignored and the word is generated from scratch.

The Copy-LSTM outputs a hidden state $\boldsymbol{h}_t^2$, which is then passed to a fully connected layers (MLP in figure 4.15) to predict the probability distribution over the vocabulary using softmax. It is also passed back to the Attention-LSTM for use in the next time step. As a final step, the probability distribution over the vocabulary is fused with the output of the denoising autoencoder to produce the output word. The embedding and hidden size of both the LSTM encoder and decoder network is 1024 and the attention dimension is 512.

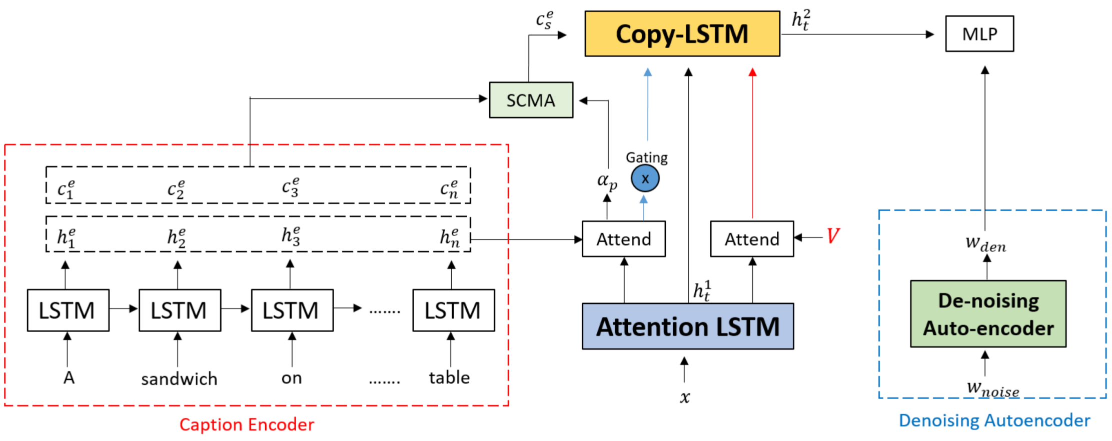


Figure 4.15: The detailed architecture of the EditNet. The DCNet is marked as a single component ("De-noising Auto-encoder"). Image source: Sammani and Melas-Kyriazi [44].

DCNet. Denoising autoencoders are commonly used to reconstruct noisy images. In the captioning context, the input caption is thought of as a noisy version of a true caption. DCNet[44] operates only on textual features, and is composed of a bi-directional LSTM (BiLSTM) encoder, which encodes the noisy caption into a latent representation, and an LSTM decoder, which decodes the latent representation. For the decoder, the LSTM-based Top-Down decoder (from Anderson et al.[2]) is used. The encoder BiLSTM has a hidden size of 512 for each direction, and a total dimension of 1024 including both directions. The LSTM decoder has a hidden size of 1024, an embedding dimension of 1024 and an attention dimension of 512.

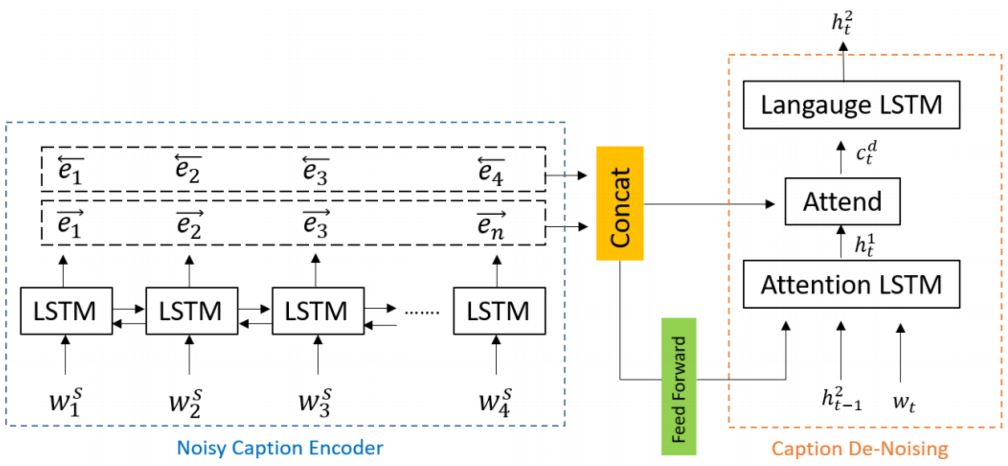


Figure 4.16: The architecture of the denoising autoencoder DCNet. The noisy caption encoder is a bidirectional LSTM. The decoder consist of an Attention LSTM and a Language LSTM (typo "Langauge" is in the original image). Image source: Sammani and Melas-Kyriazi [45]

Training. Show, Edit and Tell is initially trained with the same word-level cross-entropy loss as the other captioners. Additionally, the model is optimized using an altered loss, which contains the cross-entropy loss $\mathcal{L}_{\text{XE}}(\theta)$ and a mean squared error (MSE) loss:

$$\mathcal{L} = \mathcal{L}_{\text{XE}}(\theta) + \mathcal{L}_{\text{MSE}} \tag{4.24}$$

The mean squared error (MSE) is calculated between the last decoder hidden state of the language model and the last hidden state of the ground truth caption. The ground truth caption hidden state is calculated by running the ground truth caption through the encoder of the denoising autoencoder. The MSE loss is calculated using:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{h}_n^d - \boldsymbol{h}_n^g) \tag{4.25}$$

where the last hidden state of the language model $\boldsymbol{h}_n^2$ is linearly projected:

$$\boldsymbol{h}_n^d = \boldsymbol{W}_d\boldsymbol{h}_n^2 + \boldsymbol{b}_d \tag{4.26}$$

where $\boldsymbol{W}_d$ is an embedding matrix and $\boldsymbol{b}_d$ a bias vector.

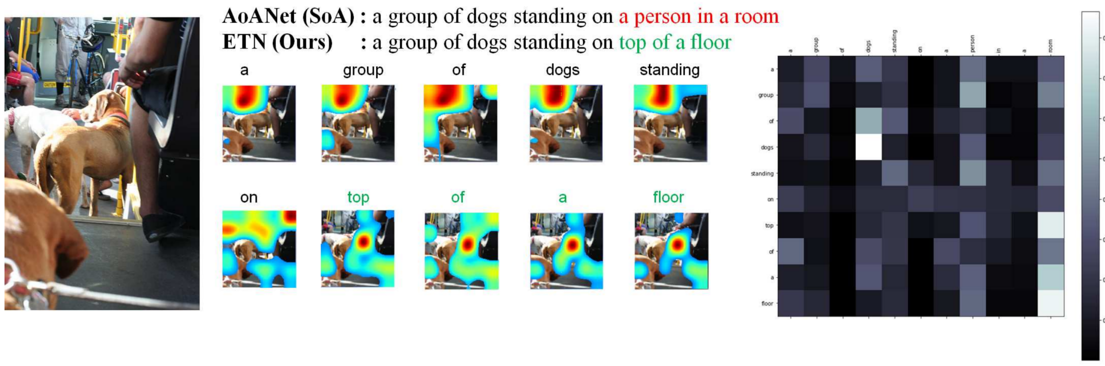
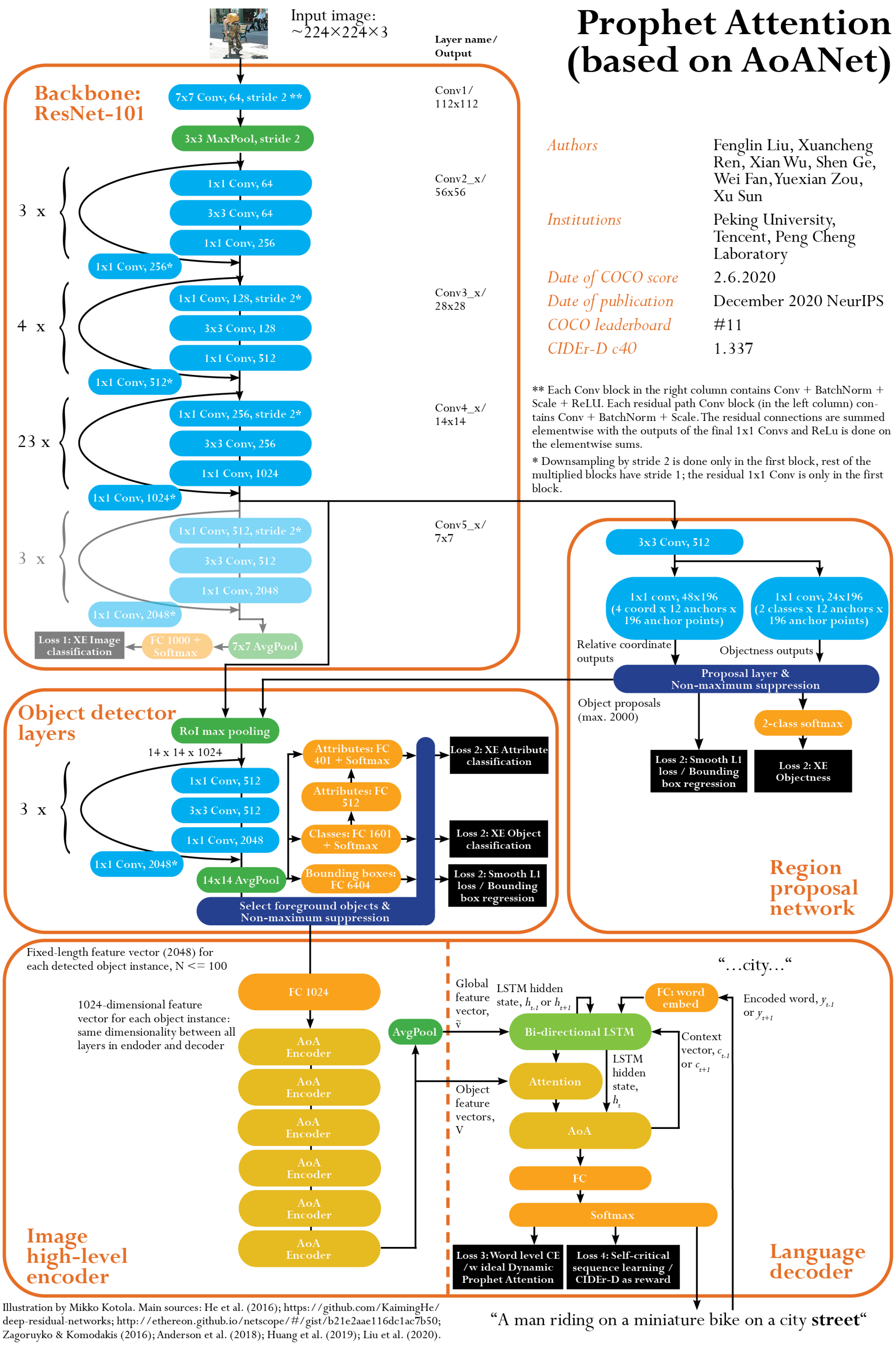


Figure 4.17: A comparison of the original caption by AoANet and the one edited by Show, Edit and Tell (ETN). The word-by-word visual attention maps of the ETN in the middle, and the textual alignment map on the right. ETN is aligning the newly generated words "top", "of", "a", "floor" to the last word in the original caption, and attending to the standing dog (and somewhat to the floor area below it) when generating these words. Image source: Sammani and Melas-Kyriazi [44].

Main contributions. One main contribution of Show, Edit and Tell is taking the approach of postprocessing by editing captions and supervising their fluency and grounding in the image contents. The editing is demonstrated in practice in figure 4.17, where Show, Edit and Tell fixes the incorrect end part of a caption generated by AoANet. The approach is unique, and some successful cases are demonstrated, but the overall performance of the model, as measured by the CIDEr score on the offline test set and the CIDEr-D score on the online test server, is only equal or even trailing to that of AoANet. The approach also adds complexity, as there are actually two captioners needed: one to generate the original caption and one to try to improve it by iterative editing. The presented editing model does not therefore deserve a named place in all future captioning systems: rather the idea of postprocessing or supervising a caption is something that can be further investigated.


Input image: ~224×224×3
Layer name/ Output
Backbone: ResNet-101
7x7 Conv, 64, stride 2 **
Conv1/ 112x112
3x3 MaxPool, stride 2
3 x
1x1 Conv, 64
3x3 Conv, 64
1x1 Conv, 256
1x1 Conv, 256*
Conv2_x/ 56x56
4 x
1x1 Conv, 128, stride 2*
3x3 Conv, 128
1x1 Conv, 512
1x1 Conv, 512*
Conv3_x/ 28x28
23 x
1x1 Conv, 256, stride 2*
3x3 Conv, 256
1x1 Conv, 1024
1x1 Conv, 1024*
Conv4_x/ 14x14
3 x
1x1 Conv, 512, stride 2*
3x3 Conv, 512
1x1 Conv, 2048
1x1 Conv, 2048*
Conv5_x/ 7x7
Loss 1: XE Image classification
FC 1000 + Softmax
7x7 AvgPool
Object detector layers
RoI max pooling
14 x 14 x 1024
3 x
1x1 Conv, 512
3x3 Conv, 512
1x1 Conv, 2048
1x1 Conv, 2048*
14x14 AvgPool
Attributes: FC 401 + Softmax
Attributes: FC 512
Classes: FC 1601 + Softmax
Bounding boxes: FC 6404
Loss 2: XE Attribute classification
Loss 2: XE Object classification
Loss 2: Smooth L1 loss / Bounding box regression
Select foreground objects & Non-maximum suppression
3x3 Conv, 512
1x1 conv, 48x196 (4 coord x 12 anchors x 196 anchor points)
1x1 conv, 24x196 (2 classes x 12 anchors x 196 anchor points)
Relative coordinate outputs
Objectness outputs
Proposal layer & Non-maximum suppression
Object proposals (max. 2000)
2-class softmax
Loss 2: Smooth L1 loss / Bounding box regression
Loss 2: XE Objectness
Region proposal network
Fixed-length feature vector (2048) for each detected object instance, N <= 100
FC 1024
1024-dimensional feature vector for each object instance: same dimensionality between all layers in endoder and decoder
AoA Encoder
AoA Encoder
AoA Encoder
AoA Encoder
AoA Encoder
AoA Encoder
Image high-level encoder
AvgPool
Global feature vector, $\bar{v}$
Object feature vectors, V
"…city…"
LSTM hidden state, h_{t-1} or h_{t+1}
FC: word embed
Encoded word, y_{t-1} or y_{t+1}
Bi-directional LSTM
Context vector, c_{t-1} or c_{t+1}
Attention
LSTM hidden state, h_t
AoA
FC
Softmax
Loss 3: Word level CE /w ideal Dynamic Prophet Attention
Loss 4: Self-critical sequence learning / CIDEr-D as reward
Language decoder
"A man riding on a miniature bike on a city street"

Prophet Attention (based on AoANet)

Authors	Fenglin Liu, Xuancheng Ren, Xian Wu, Shen Ge, Wei Fan, Yuexian Zou, Xu Sun
Institutions	Peking University, Tencent, Peng Cheng Laboratory
Date of COCO score	2.6.2020
Date of publication	December 2020 NeurIPS
COCO leaderboard	#11
CIDEr-D c40	1.337

** Each Conv block in the right column contains Conv + BatchNorm + Scale + ReLU. Each residual path Conv block (in the left column) contains Conv + BatchNorm + Scale. The residual connections are summed elementwise with the outputs of the final 1x1 Convs and ReLu is done on the elementwise sums.

* Downsampling by stride 2 is done only in the first block, rest of the multiplied blocks have stride 1; the residual 1x1 Conv is only in the first block.

Illustration by Mikko Kotola. Main sources: He et al. (2016); https://github.com/KaimingHe/deep-residual-networks; http://ethereon.github.io/netscope/#/gist/b21e2aae116dc1ac7b50; Zagoruyko & Komodakis (2016); Anderson et al. (2018); Huang et al. (2019); Liu et al. (2020).

Figure 4.18: Architecture of the complete Prophet Attention system. This is the version based on AoANet and reported in the COCO online test server leaderboard.

4.5 Prophet Attention

The key idea of attention-based encoder-decoder captioners is to have the decoder attend to relevant image regions at each step of the decoding process. Liu et al. [33] note that models usually use the hidden state of the current NLG input to attend to image regions, and end up attending to image regions that are related to the previous words in the caption, not the one being generated in the time-step in question. This is called the problem of *deviated focus*, and it impairs the performance of both grounding the word in the image regions and as a consequence also of the whole captioning task. The Prophet Attention model aims to fix the deviated focus problem by utilizing the future information (words that come after the one currently being generated) available in the training phase to calculate ideal attention weights for image regions in the decoder. The calculated ideal weights are then used to regularize the deviated attention, grounding words in the relevant image regions.

Deviated focus. The deviated focus problem is demonstrated in figure 4.19 using the Attention on Attention model. The model grounds each word more on regions related to the previous time steps than the current one. For example in time step 2, the model is expected to output the word "woman", but it only attends to the woman in time step 3. In time step 6, the model attends to the woman with the yellow raincoat, even if it is meant to output the word "umbrella". It attends to the umbrella only in step 7, at which point it ideally should already be attending again to the woman for outputting the word "wearing". The problem is caused by the decoder using the hidden state of the current input, already generated words, to attend to the image regions.

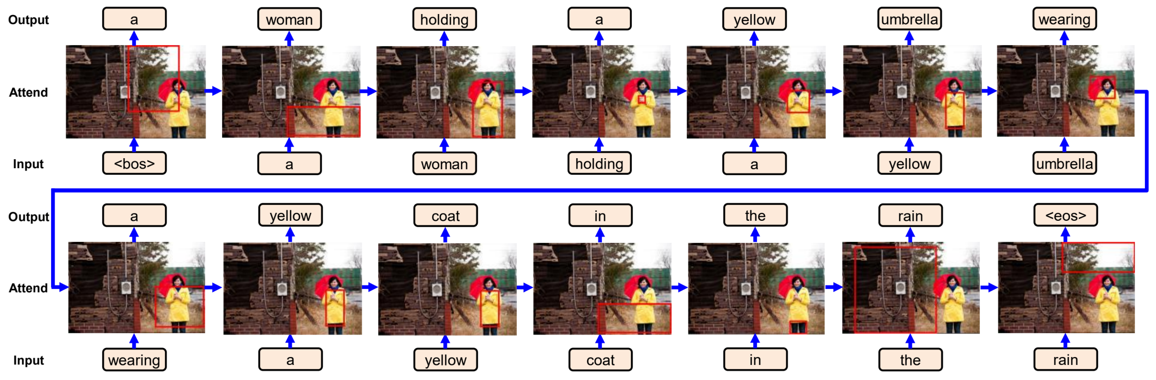


Figure 4.19: Deviated focus example. The image shows the Attention on Attention captioner's top 1 attended image region for each time step of the decoding process. In step 6, the model attends to the woman with the yellow raincoat, even if it is meant to output the word *umbrella*. It attends to the umbrella in step 7. Image source: Liu et al. [33].

Prophet Attention. Prophet Attention improves the grounding of words in more relevant image regions by calculating ideal attention weights by utilizing future information, which is readily available in the training phase [33]. The ideal attention weights are then used to guide the initial attention, which is based only on the already generated input words. The idea is to regularize the attention model based on future words.

In the regular attention approach, at each decoding time step t, the decoder LSTM takes as input the hidden state of the LSTM $\boldsymbol{h}_{t-1}$, and the generated word from the previous time step $\boldsymbol{y}_{t-1}$[1] embedded with learnable embeddings and concatenated with the global feature vector implemented as average pooling the visual features $\bar{\boldsymbol{v}} = \frac{1}{k}\sum_{i=1}^{k}\boldsymbol{v}_i$ [33]:

$$\boldsymbol{h}_t = \text{LSTM}(\boldsymbol{h}_{t-1}, [\boldsymbol{W}_e\boldsymbol{y}_{t-1}; \bar{\boldsymbol{v}}]) \tag{4.27}$$

where $\boldsymbol{h}_t$ is the resulting hidden state of the LSTM, $\boldsymbol{W}_e$ is a learnable word embedding matrix and $[;]$ is a concatenation operation. The LSTM output $\boldsymbol{h}_t$ is then used as a query to calculate attention scores of the relevant object feature vectors $\boldsymbol{V}$

$$\boldsymbol{\alpha}_t = f_{\text{Att}}(\boldsymbol{h}_t, \boldsymbol{V}) = \text{softmax}(\boldsymbol{w}_\alpha \tanh(\boldsymbol{W}_h\boldsymbol{h}_t \oplus \boldsymbol{W}_V\boldsymbol{V})) \tag{4.28}$$

where $\boldsymbol{w}_\alpha$ is an embedding vector, $\boldsymbol{W}_h$ and $\boldsymbol{W}_V$ are embedding matrices, tanh is the hyperbolic tangent function and $\oplus$ is matrix-vector addition, adding a vector to each column of a matrix. The attention scores $\boldsymbol{\alpha}_t$ are then multiplied with the object feature vectors $\boldsymbol{V}$ to generate the attended object features $\boldsymbol{c}_t$:

$$\boldsymbol{c}_t = \boldsymbol{V}\boldsymbol{\alpha}_t^T \tag{4.29}$$

Attended object features $\boldsymbol{c}_t$ and output of the LSTM $\boldsymbol{h}_t$ are used as input to a fully connected layer and softmax to calculate probabilities of the next word over the dictionary:

$$\boldsymbol{y}_t = \text{softmax}(\boldsymbol{W}_p[\boldsymbol{h}_t; \boldsymbol{c}_t] + \boldsymbol{b}_p) \tag{4.30}$$

where $\boldsymbol{W}_p$ is a learnable linear transformation and $\boldsymbol{b}_p$ a learnable bias vector.

The approach of Prophet Attention is to augment the network in the training phase with a Bidirectional LSTM (BiLSTM) to calculate ideal attention weights $\hat{\boldsymbol{\alpha}}_t$. The whole word sequence $\boldsymbol{y}_{i:j}$ is first encoded using the BiLSTM to get $\boldsymbol{h}'_{i:j}$, and then the ideal attention weights are calculated:

$$\hat{\boldsymbol{\alpha}}_t = f_{\text{Prophet}}(\boldsymbol{h}'_{i:j}, \boldsymbol{V}) = \frac{1}{j-i+1}\sum_{k=i}^{j} f_{\text{Att}}(\boldsymbol{h}'_k, \boldsymbol{V}) \tag{4.31}$$

where, i is the index of the first word in the sequence and j is the index of the last word, and f_{Att} is the same attention function as in the mainstream attention and shares the same parameters.

[1]The input word is in one-hot encoded format over the dictionary.

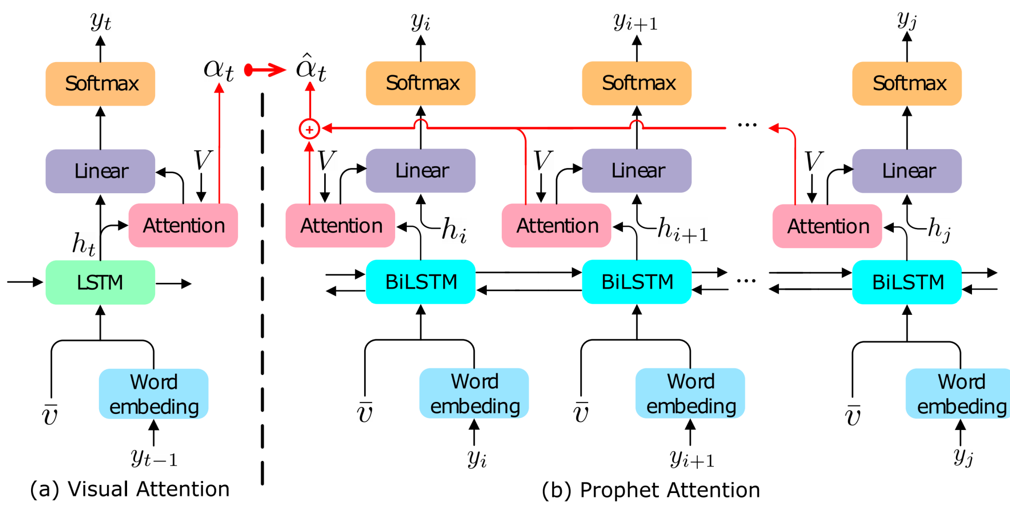


Figure 4.20: Prophet attention uses a bi-directional LSTM to calculate ideal attention weights $\hat{\boldsymbol{\alpha}}_t$ based on future words in the ground truth caption as a target for the main attention model weights $\boldsymbol{\alpha}_t$, which is based only on the previously generated words. Image source: Liu et al. [33].

Training. Prophet attention introduces new loss functions and changes the supervised training phase by introducing the regularization of the attention towards image features in the image decoder.

The original cross-entropy loss $\mathcal{L}_{\text{XE}}(\theta)$ is defined as:

$$\mathcal{L}_{\text{XE}}(\theta) = -\sum_{t=1}^{T} \log(p_\theta(y_t^* | y_{1:t-1}^*)) \tag{4.32}$$

where y_t^* is a word at time-step t, and $p_\theta(y_t^* | y_{1:t-1}^*)$ is the probability of a word conditional to the previously generated words in the sequence. To take the ideal losses into account during training, a regularizing loss function that uses the L1 norm to penalize deviation of predicted attention scores $\boldsymbol{\alpha}_t$ from the ideal attention scores $\hat{\boldsymbol{\alpha}}_t$ is introduced:

$$\mathcal{L}_{\text{Att}}(\theta) = -\sum_{t=1}^{T} \|\boldsymbol{\alpha}_t - \hat{\boldsymbol{\alpha}}_t\|_1 \tag{4.33}$$

The full training objective loss function function $\mathcal{L}_{\text{Full}}(\theta)$ is then defined as:

$$\mathcal{L}_{\text{Full}}(\theta) = \mathcal{L}_{\text{XE}}(\theta) + \hat{\mathcal{L}}_{\text{XE}}(\theta) + \lambda \mathcal{L}_{\text{Att}}(\theta) \tag{4.34}$$

where λ is a hyperparameter to control the amount of regularization. Based on experiments, λ value of 0.01 gives the best overall results for the AoANet-based Prophet network.

In the supervised training phase, the captioner is first trained without the Prophet part, only using $\mathcal{L}_{\text{XE}}(\theta)$, for a number of epochs (25 in the paper) to initialize the decoder parameters. Then supervised training is continued using the full loss function

$\mathcal{L}_{\text{Full}}(\theta)$. The self-critical reinforcement training phase with CIDEr-D as the optimization objective is used similarly as described in the other models under investigation. As the Prophet Attention is only used in the training phase to guide the attention, the inference stage, where the future words are naturally not available in any case, works precisely as before - the learned weights just guide the decoder to attend to more relevant image features.

Dynamic Prophet Attention. Dynamic Prophet Attention (DPA) [33] is a practical refinement of the Prophet Attention approach to account for there being different kinds of words in the captions. DPA takes into account noun phrases (NP)[1] (for example "a yellow umbrella"), non-visual words (NV) (for example "the" and "of") and groups all other words, such as verbs, together. For noun phrases, the prophet attention is utilized so that the ideal attention is based *only* on the words in the noun phrase (be they in the past, current or future time step). For non-visual words, the Prophet-related loss functions $\hat{\mathcal{L}}_{\text{XE}}(\theta)$ and $\mathcal{L}_{\text{Att}}(\theta)$ are disabled and only the original loss $\mathcal{L}_{\text{XE}}(\theta)$ is used, meaning that the attention is based mostly on the past words. For the other words, the ideal attention is based *only* on the current word.

Generalizability. The Prophet Attention approach is implemented on top of several LSTM-based captioners including the original Bottom-Up Top-Down captioner [2] and the AoANet. The AoANet-based Prophet network is the best version and is the one described in figure 4.18 and used for the COCO online test server scores. The effect of adding the Dynamic Prophet Attention to AoANet is increasing CIDEr-D c40 from 1.296 to 1.337 points, a notable improvement. An improvement of 0.55 CIDER c40 points is also demonstrated for the Bottom-Up Top-Down network in the offline evaluation.

Main contributions. The main contribution is an improved approach to grounding each generated word on the most relevant image regions instead of those regions most relevant to the already generated words. The Prophet Attention approach is novel and an isolated improvement, meaning that is can be easily incorporated into any attention-based encoder-decoder captioning model. The original paper demonstrates the implementation for LSTM-based decoders, utilizing a bidirectional LSTM for the training phase. Similar approach could be implemented for transformer-based decoders. This possibility is further discussed in chapter 7.

[1]The spaCy library is used for noun phrase tagging in the original Prophet implementation.

5. Improving the Backbone Image Processing Network

Object detection as a task is closely related to image classification, and both of these tasks use the same CNN-based backbone networks for the early stages of the processing. For this reason, advances in these backbone networks potentially contribute also to the performance of image captioning systems as a whole. Furthermore, most current state-of-the-art image captioners rely essentially on ResNet-101, which is already 5 years old, as used in Faster R-CNN [40] and Bottom-Up Top-Down [2]. Computer vision, object classification and object detection are very active areas of research. This chapter presents an overview to latest research in these areas and proposes ways to improve the backbone component. This chapter is followed up by reviewing possible improvements to the object detector components in chapter 6.

As described in chapter 4, X-Linear Attention Network experiments with changing the backbone network from ResNet-101 to SENet-154 and achieves improvements in the CIDEr score of the system. This demonstrates that changing backbones can improve captioning system performance. This is understandable, as the backbone is the largest part of the system, and the capacity of the backbone to learn image features that are useful in detecting objects and their attributes is central for the ability of the object detection layers, high-level image encoder and NLG decoder to detect objects, learn the interactions between the objects and the overall image context, and generate captions based on that information.

The backbones presented in this chapter were chosen based on their contribution to the accuracy of image classification or object detection. For example the Darknet [38, 39] networks used as backbones in the YOLO family of object detectors are not presented, because their contributions are focused on efficiency of training and inference, not primarily on advancing accuracy. The first three sections, 5.1 ResNeXt, 5.2 Squeeze-and-Excitation Networks and 5.3 EfficientNets, present central improvements to the ResNet architecture. Section 5.4 Vision Transformers presents an alternative, transformer-based approach to backbones. Section 5.5 discusses the role of large datasets in improving backbones, and finally section 5.6 draws together the most potential improvements to backbone networks.

5.1 ResNeXt

ResNeXt [55] evolves the ResNet architecture of repeating layers and residual connections by augmenting it with a split-transform-merge strategy. The split-transform-merge strategy comes from the Inception [49] modules, where the input is split into a few lower-dimensional embeddings by using 1×1 convolutions, transformed by a set of specialized convolutional filters (typically 3×3 or 5×5), and finally merged by concatenation [55]. The aim of an Inception module is to approximate the representational power of a single large layer with lower computational cost. The innovation of ResNeXt in relation to other Inception-based approaches is to keep the topology of each path constant, effectively introducing the size of the set of transformations, *cardinality*, as a new dimension alongside the width and depth of a network. Moreover, it is demonstrated, that increasing cardinality is a more effective way of gaining accuracy in image classification than increasing the depth or width of a CNN. The ResNeXt block can be implemented in three equivalent ways as shown in figure 5.1. Implementation *a* splits the input into 32 paths for all three convolutions and then concatenates the results together and adds them together with the residual connection. Implementation *b* replaces the last 1×1 convolutions by early concatenation and one larger 1×1 convolution. This is equivalent, because the 1×1 convolution learns features channel-wise and the inputs to it are also split channel-wise in both cases. Implementation *c* uses *grouped convolutions*, where input and output channels are divided into C groups, and convolutions are separately performed within each group. The grouped convolutional layer performs 32 groups of convolutions, where the input and output channels for each group are 4-dimensional. The grouped convolutional layer also concatenates the outputs into a 128-dimensional concatenated output vector, which is then fed into the last 1×1 convolution. This implementation looks quite similar to the original ResNet block, but the middle convolution layer is wider and has sparser connections.

5.2 Squeeze-and-Excitation Networks

Squeeze-and-Excitation Networks (SENets) [24] seek to improve the representational power of a CNN by explicitly modeling the interdependencies between the channels of its convolutional features. SENet achieves this by introducing a mechanism to allow the network to perform feature recalibration, to enable learning to use global information to selectively emphasize informative features and suppress less informative ones. SENet contributes by augmenting the stackable block structure, so the contribution is reusable

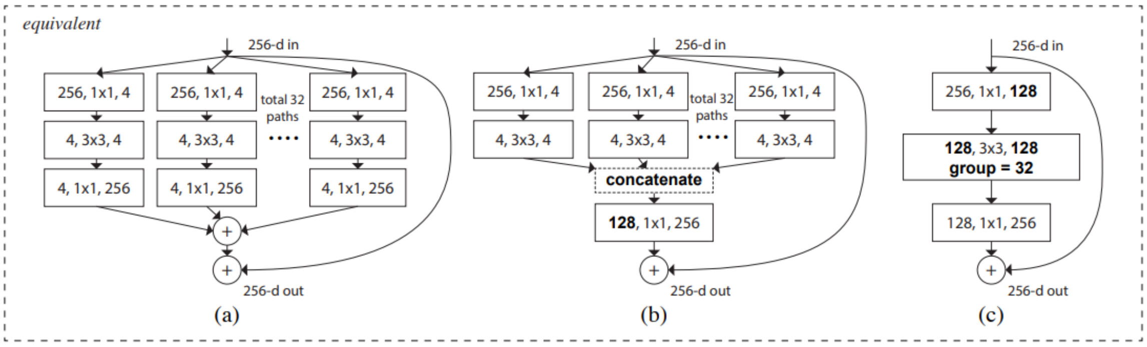


Figure 5.1: Three equivalent implementations of the ResNeXt block with 256-dimensional inputs and outputs. The implementation on the right is the one used as the base for SENets. Image source: Xie et al. [55]

in many CNNs based on stackable blocks, including ResNet and ResNeXt.

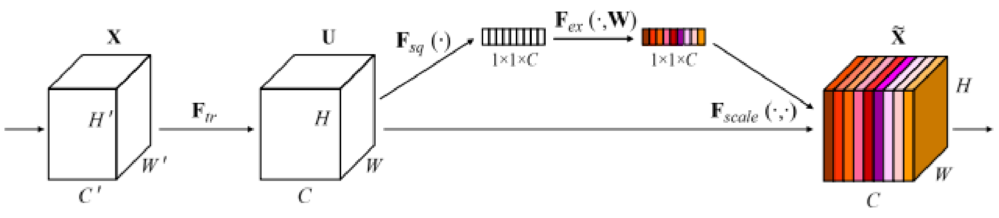


Figure 5.2: A Squeeze-and-Excitation block. The squeeze features (produced by $\boldsymbol{F}_{\text{sq}}$) learn useful channels and are then used to excite ($\boldsymbol{F}_{\text{ex}}$) the most useful channels in the output $\boldsymbol{U}$, resulting in channel-wise scaled features $\tilde{\boldsymbol{X}}$. Image source: Hu et al. [24].

A Squeeze-and-Excitation block is depicted in figure 5.2. The SE block is placed after some base transformation $\boldsymbol{F}_{\text{tr}}$ such as a ResNet block. The squeeze operation $\boldsymbol{F}_{\text{sq}}$ is implemented by globally average pooling each channel. The channel descriptor statistic vector $\boldsymbol{z} \in \mathbb{R}^C$ is calculated by shrinking the input $\boldsymbol{U} \in H \times W \times C$ through spatial dimensions height H and width W. The c:th element in $\boldsymbol{z}$ is

$$z_c = \boldsymbol{F}_{\text{sq}}(\boldsymbol{u}_c) = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} \boldsymbol{u}_c(i,j) \tag{5.1}$$

The excitation operation $\boldsymbol{F}_{\text{ex}}$ is implemented by a simple gating mechanism:

$$\boldsymbol{s} = \boldsymbol{F}_{\text{ex}}(\boldsymbol{z}, \boldsymbol{W}) = \sigma(\boldsymbol{W}_2 \text{ReLU}(\boldsymbol{W}_1 \boldsymbol{z})) \tag{5.2}$$

where $\boldsymbol{s}$ is the resulting channel-wise excitation weight vector, σ is the sigmoid function, $\boldsymbol{W}_1 \in \mathbb{R}^{\frac{C}{r} \times C}$, $\boldsymbol{W}_2 \in \mathbb{R}^{C \times \frac{C}{r}}$, and r is a reduction ratio for the bottleneck in the gating. The final output of the SE block is calculated by rescaling the input U with the activations $\boldsymbol{s}$:

$$\tilde{\boldsymbol{x}}_c = \boldsymbol{F}_{\text{scale}}(\boldsymbol{u}_c, s_c) = s_c \cdot \boldsymbol{u}_c \tag{5.3}$$

where $\tilde{\boldsymbol{x}} = [\tilde{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_2, ..., \tilde{\boldsymbol{x}}_C]$, and $s_c \cdot \boldsymbol{u}_c$ is channel-wise multiplication of the scalar scaling factor s_c and the channel feature map $\boldsymbol{u}_c \in \mathbb{R}^{H \times W}$.

The squeeze-and-excitation blocks can be applied upon a ResNet architecture [24]. A SE-ResNet Module is depicted in figure 5.3. The SE-block is placed after each convolutional ResNet block, before the identity branch is summed with the transformed branch. The SE-ResNet-101 is shown to outperform the plain ResNet-101 in the object detection task on the COCO dataset by improving average precision (AP) from 27.2 to 27.9. A complete SENet-154 Network architecture, based on a ResNeXt-152, is shown in the X-LAN architecture figure 4.10.

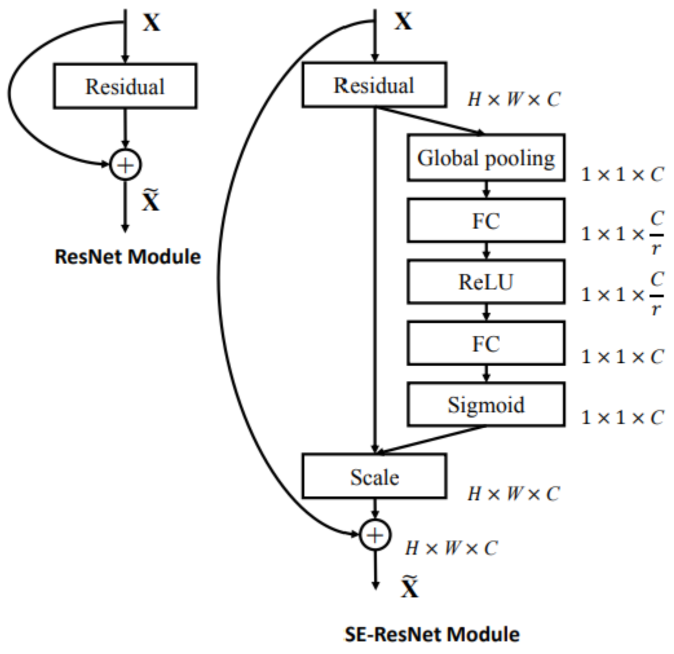


Figure 5.3: The structure of a plain ResNet Module and a squeeze-and-excite-augmented SE-ResNet Module. Image source: Hu et al. [24].

5.3 EfficientNets

Already in 2016, Zagoruyko and Komodakis [56] presented *Wide Residual Network* (WRN) architecture, which focused on increasing the width of a network, the number of channels in a convolutional layer, instead of depth, the number of layers. They showed that for an image classification task, a 16-layer wide network that uses residual blocks has the same accuracy as a 1000-layer thin deep network using similar blocks. Both networks have approximately the same amount of parameters, but the WRN is much faster to train. The computational efficiency of training WRNs seems to be related to GPUs being able to take advantage of efficient parallel matrix operations in wider convolution blocks, compared to a deeper network with thinner convolution blocks.

Tan and Le [50] continue the work on ways of scaling up CNNs. They observe that convolutional networks can be scaled up by increasing their depth, width or resolution. Depth is defined as the number of layers in the network, width as the number of channels in each convolutional layer and resolution as the number of pixels in the input image. They observe that scaling up just one of these dimensions does not produce optimal accuracy, but the accuracy gain diminishes or even saturates after some point of each dimension. Instead, they propose scaling up networks by uniformly scaling all dimensions of depth, width and resolution using a compound coefficient. The compound coefficient ϕ is used to determine the optimal constants α, β, γ, which control the scaling of the depth $d = \alpha^{\phi}$, width $w = \beta^{\phi}$ and resolution $r = \gamma^{\phi}$:

$$\alpha \cdot \beta^2 \cdot \gamma^2 \approx m \tag{5.4}$$

where $\alpha \leq 1, \beta \leq 1, \gamma \leq 1$ and the m on right side of the equation defines the multiplier of the resources that can be spent on scaling the network.

The systematic uniform scaling approach is demonstrated to work for efficient scaling of for example ResNets, but it is also used to create a new family of convolutional networks called EfficientNets [50]. The main building block of EfficientNets is the mobile inverted bottleneck module (MBConv) with added squeeze-and-excitation optimization. There are several innovations in the MBConv [46] module. Firstly, the residual connections are between the layers with fewer channels instead of the wider layers. The idea is that the bottlenecks contain all the necessary information, and the expansion layers act merely as implementation details that accompany a non-linear transformation of the tensor. The non-linearity is also removed from the bottleneck to prevent it from destroying too much information, making it a *linear bottleneck*. The convolutions used in the expansion layer are *depthwise separable convolutions* instead of plain convolutions. Depthwise separable convolutions replace a full convolutional operator with a factorized version that splits the convolution into two separate layers. The first layer is a depthwise convolution, which performs lightweight filtering by applying a single convolutional filter (most often 3×3 or 5×5 in the MBConv module) per input channel. The second layer is a pointwise 1×1 convolution, which is responsible for building new features by computing linear combinations of the input channels. Depthwise separable convolutions work almost as well as standard convolutions but the computational cost is 8-9 times smaller [46] in the backbone network settings. After the expansion layer, features are projected back to a low-dimensional representation using a linear (1×1) convolution. The comparison of the structure of an MBConv block to a regular residual block is shown in figure 5.4 . The MBConv module was originally invented to support efficient operations on low-resource mobile platforms, but they have since successfully been used also in large networks in high-resource environments.

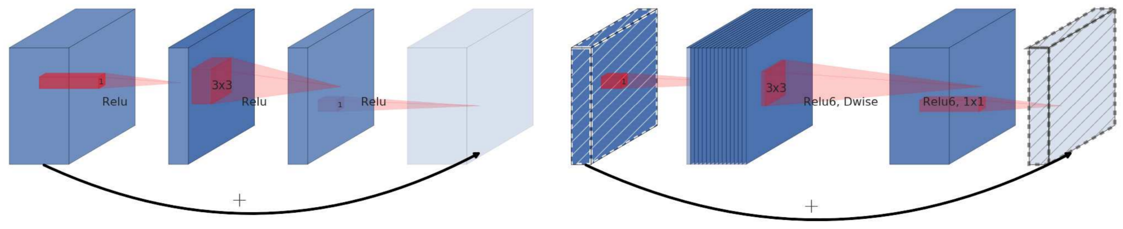


Figure 5.4: The traditional residual block on the left, the inverted residual MBConv block structure on the right. The main innovations in the MBConv block are connecting the low-dimensional representations by residual connections, using a linear bottleneck without non-linearity and using depthwise separable convolution in the expansion layer. The layers with diagonal markings do not use non-linearities. Image source: Sandler et al. [46].

The architecture of the smallest network of the EfficientNet family, EfficientNet-B0, is shown in figure 5.5. It is noteworthy, that the EfficientNet architecture has relatively more layers at higher resolutions than 14×14 than the ResNet-101.

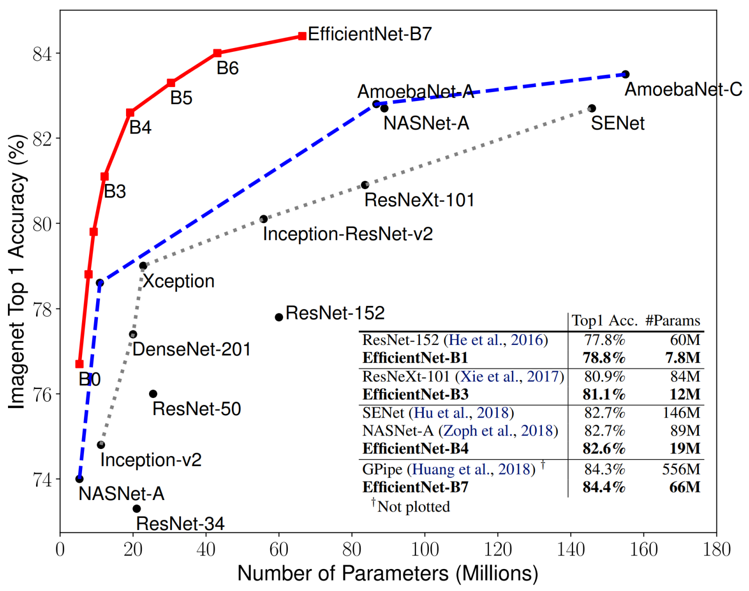
Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	28×28	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

Figure 5.5: The architecture of the baseline EfficientNet-B0. Resolution is the input resolution of the stage layers, and #Layers is the number of similar layers in a stage. Image source: Tan and Le [50].

Using multiplier 2 for a small baseline EfficientNet-B0, which has 18 convolutional layers, Tan and Le use grid search and discover that the optimal scaling constants are $\alpha = 1.2, \beta = 1.1, \gamma = 1.15$. They hypothesize that these scaling constants are quite near optimal for this network architecture. Using these constants, they define EfficientNets of different sizes, from EfficientNet-B0 with 5.3M parameters and a 76.3 % top-1 accuracy[1] on ImageNet, all the way to EfficientNet-B7 with 66M parameters and a 84.4 % top-1 accuracy.

[1]Top-1 accuracy means that the class predicted by a model to have the highest probability is the

A comparison of the ImageNet top-1 accuracy vs. number of parameters for several networks including EfficientNets, SENet-154, ResNeXt-101 and ResNet-152 is shown in figure 5.6.



	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**78.8%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.1%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.6%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.4%**	**66M**

†Not plotted

Figure 5.6: EfficientNets achieve higher ImageNet top-1 accuracy than other convolutional networks with a smaller number of parameters. Image source: Tan et al. [50].

5.4 Vision Transformers

Convolutional neural networks have for a long time been the leading approach in computer vision. Recently, transformer-based networks have been proposed as an alternative to CNNs also for multi-level feature learning, the essence of the backbone networks. Dosovitskiy et al. [15] propose to use Vision Transformers (ViT) instead of CNNs. Vision Transformers are an adaption of transformers to the domain of computer vision.

In Vision Transformers, the image is split into patches, and the sequence of linear embeddings of these patches is used as input to a transformer. Image patches are treated the same way as tokens (words) in the original natural language processing (NLP) related transformer model. The idea is to use the transformer model with minimal alterations in the computer vision context. The idea of the Vision Transformer is presented in figure 5.7.

Vision Transformers lack some of the core features of the CNNs, namely translational equivariance and locality. For this reason, they do not reach the same image

same as the ground truth class. Sometimes also top-5 accuracy is reported. In the top-5 case it is only required that the ground truth class in the top five most probable classes as predicted by the model.

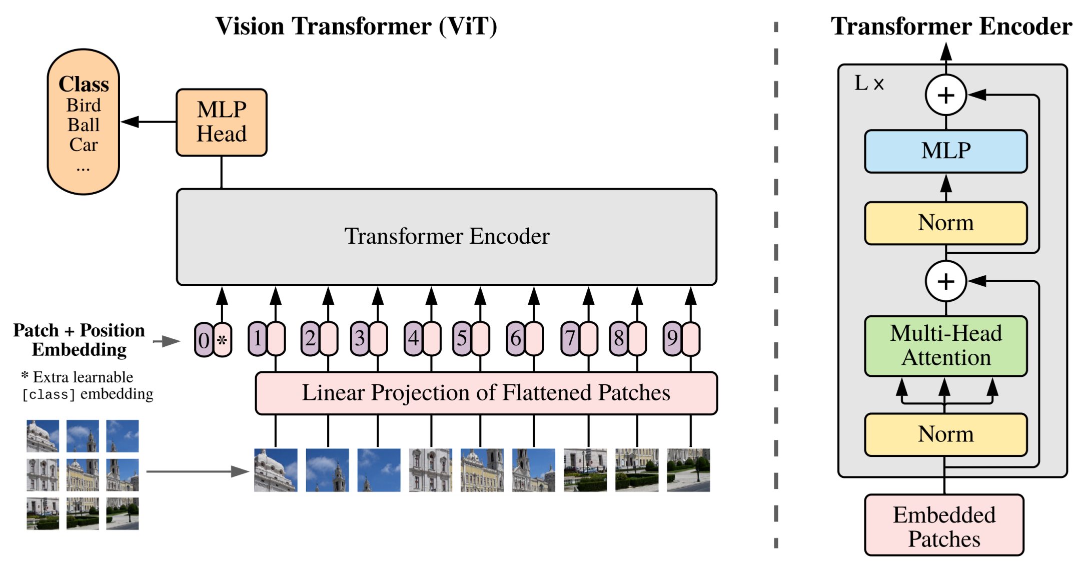


Figure 5.7: Vision Transformer model adapts the transformer into computer vision by splitting the image into patches, adding positional encodings and a class encoding, and using them as a sequence similarly to natural language processing domain transformer models. Image source: Dosovitskiy et al. [15]

classification performance as ResNet models when trained on only the widely used ImageNet-1k dataset (1,4 million images). But when trained with a still larger dataset such as the full ImageNet-21k (14,2 M images) or JFT-300M (303 M images), the ViT is able to overcome the initial handicap and reaches state-of-the-art performance (image classification top-1 accuracy of 88.55 % on ImageNet) [15]. Interestingly, the ViT models are demonstrated to learn the 2-dimensional relation of the image patches autonomously, and hybrid approaches partly using convolutional features do not improve performance when models are trained with large datasets. The ViT models are trained in a supervised fashion by adding a class embedding to the input and using the state of that embedding after the transformer encoding layers for classification.

Vision Transformers are a major breakthrough in the image processing backbone networks. Importantly, they reach higher classification accuracy than ResNet-based models and the accuracy increase that is associated with training on ever larger datasets does not saturate yet at the level of 300 million images [15], suggesting that even larger datasets can further improve the performance of this type of backbone network.

Caron et al. [10] build on Vision Transformers and present an approach based on self-supervised learning with knowledge distillation called DINO (knowledge **di**stillation with **no** labels). *Knowledge distillation* is a learning paradigm where a student network is trained to match the output of a teacher network. In DINO, the structure of the student and teacher networks is identical. Global views of an image, meaning crops covering over 50 % of an images, are passed through the teacher and more tightly cropped local views, covering a smaller area of the image, are passed through the student. Effectively, the student is taught to model the global features of an image from cropped, local views. Interestingly, the teacher network is not pre-trained, but is taught at the same time as the student network using a momentum encoder approach.

The Vision Transformer model trained with the DINO training approach and used together with a simple k-nearest neighbours (k-NN) classifier produces a good ImageNet top-1 accuracy of 78.3 % [10]. While this is clearly below the supervised ViT accuracy, the result is still good and demonstrates that a self-supervised approach can potentially reach even higher levels with more data. An additional note is that Vision Transformers do not use batch normalization, and DINO used with ViT is completely batch normalization free. I cannot go into more details of the DINO approach in this thesis – please refer to the DINO paper for more detailed information on the approach.

Touvron et al. [52] report a large Vision Transformer based model that achieves 86.5 % top-1 accuracy on ImageNet[1] when training with only ImageNet-21k data (not using for example JFT-300M). This result again strengthens the position of the vision transformer approach, since the result is achieved with only the ImageNet dataset. They name their architecture Class-Attention in Image Transformers (CaiT). The main contribution is splitting the processing into two stages. The first, self-attention stage is identical to the ViT transformer, but without a class embedding (CLS). The second stage is a class-attention stage, which compiles the set of image patch embeddings into a class embedding that is fed to a linear classifier. The two-phase architecture is shown in figure 5.8. The motivation for this improved architecture is to avoid the contradictory objective of guiding the attention process while at the same time processing the class embedding. In the class-attention stage, information is not copied from the class embedding to the patch embeddings during the forward pass: only the class embedding is updated. This architecture has similar elements as an encoder-decoder architecture.

5.5 Training on Large Datasets

The largest publicly available image dataset is the ImageNet-21k (14,2M images). As training models on larger datasets has been shown to improve the generalizability of computer vision models [2, 15], being able to train models on yet larger datasets can potentially further improve the quality of the learned features. Sun et al. [48] propose

[1]The exact model that achieves this result is CaiT-M48 $\uparrow$ 448Υ. This model has 48 self-attention blocks and 2 class attention blocks. The main training phases uses input resolution 224×224, but the model is fine-tuned with resolution 448×448 images (notated by $\uparrow$ 448) and uses a knowledge distillation training scheme (notated by Υ). See details in the CaiT paper [52].

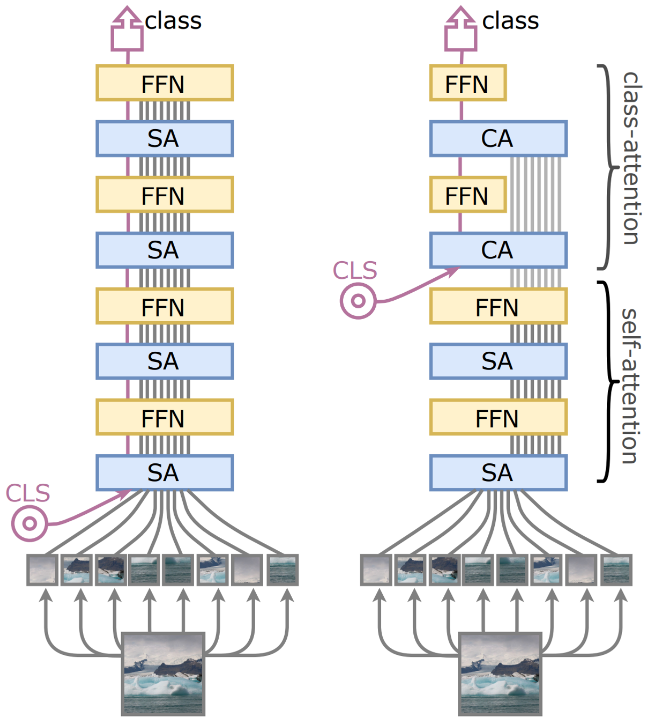


Figure 5.8: Class-Attention in Image Transformers improve the Vision Transformer architecture by separating the self-attention stage and the class-attention stage. Image source: Touvron et al. [52] (image modified by removing parts).

that the performance improvement of image-related models is logarithmic in relation to the size of the dataset, given that the model has enough capacity.

The Google-owned propriety image dataset JFT-300M [48] is approximately 21 times larger than ImageNet-21k. JFT-300M has automatically generated – and therefore noisy – labels. Even though the labels are noisy, the models can benefit from the sheer amount of source data more than the noisy labels hurt the performance. Having access to this dataset or another dataset of comparable size for pretraining the backbone network would likely improve the performance of the object detector and the image captioner built on top of it.

Most practitioners do not have access to JFT-300M. For that reason, a very important approach is better utilizing the public ImageNet-21k. Ridnik et al. [42] discuss the possibilities of pretraining on ImageNet-21k instead of ImageNet-1k and present an advanced pretraining scheme. ImageNet-21k can be transformed into a semantic multi-label dataset using the WordNet hierarchy. Ridnik et al. remove from the full ImageNet-21k dataset all classes with under 500 labels; the resulting cleaned-up dataset still contains 12 358 688 images from 11 221 classes. They then use the hierarchy of the WordNet to assign multiple labels to images. For example an image of a tunic, shown in figure 5.9, will get the labels (in hierarchical order from most detailed to most general) "tunic", "cloak", "overgarment", "garment", "clothing" and "artifact". Then, the hierarchical structure of ImageNet-21k tags can be exploited to train the network with several softmax layers, exactly one per each of the 11 hierarchy levels in WordNet, instead of the single layer. Only those softmax functions that relate to levels that the image has a label for are activated during training. This approach is called *semantic softmax training*. When still augmented with weighting the different level softmaxes and a dedicated semantic knowledge distillation scheme, an improved pretraining scheme is defined. With this advanced pretraining that takes advantage of the whole ImageNet-21K dataset and its rich semantic hierarchy, a wide range of models, both small and large, reach improved accuracy in image classification and object detection on several datasets and tasks, including multilabel image classification on COCO. For example, the multilabel mean average precision of a custom model (TResNet-M) is increased from 80.8 % to 82.2 % by changing the training procedure from single-label pretraining to pretraining with semantic softmax and knowledge distillation. The pretraining procedure systematically improves the performance of both small and large models, including the Vision Transformer.

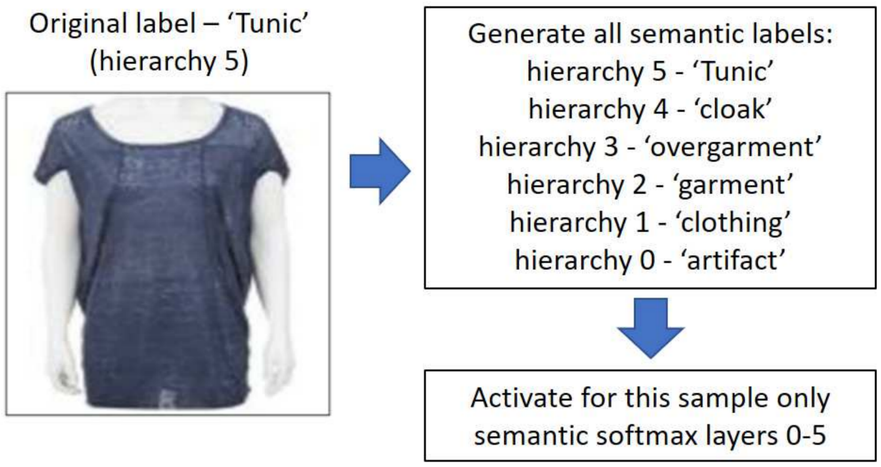


Figure 5.9: Semantic softmax training. The ImageNet-21k images are tagged with a hierarchy of labels, transforming the dataset from a single label to a semantic multilabel dataset. During training, the hierarchical labels are then used together with hierarchical softmaxes and loss functions. Image source: Ridnik et al. [42].

Another example of improving performance by training on very large datasets is Mahajan et al. [34] training a ResNeXt architecture on 3.5 billion social media images and demonstrating very competitive image classification and object detection results. Their object detector is successfully used by Shuster et al. [47] for training an image captioner with a conversational style.

5.6 Most Potential Backbone Improvements for Image Captioning

The backbones of leading image captioners can be improved by pretraining the backbones on larger datasets such as ImageNet-21k or JFT-300M, and by using the improved semantic softmax training scheme. Larger training data leads to richer feature vectors.

In addition to the amount of data and training approach, improved backbone network models can be used to improve performance. The improvement of changing the ResNet-101 to SENet-154 has already been demonstrated by X-LAN. Currently the best CNN-based backbone is the EfficientNet-B7. Using the improved backbone CNN should improve the performance of all leading image captioners, and backbones are quite easily interchangeable.

A more radical and at the same time most potential way to improve the backbone network is replacing the ResNet/CNN-based backbone with a Visual Transformer-based backbone. When trained with enough data, these new types of networks have been shown to outperform ResNets in image classification. Additional improvement may in the future be reached by changing the training approach of the backbone from a supervised to a self-supervision or knowledge distillation based approach. Since object detection in closely tied to image classification, the remaining problem would be to adjust the object detector to work with Vision Transformers or take into use a new object detector that works well on the ViT backbone. As both CNNs and Vision Transformers output features vectors, the compatibility should be easy to reach.

An important fact to consider is what the dimensionality of the feature vectors produced by Vision Transformers. In the DINO paper [10], ViT-based models are trained with 384- and 768-dimensional outputs. The feature vector dimension is smaller than the 2048 produced by the ResNet/SENet-based models. But all leading image captioners scale down the dimensionality to 1024 (AoANet; X-LAN; Show, edit and tell; Prophet) or 512 (Meshed-Memory Transformer) in the first layer of the high-level image encoder. This would suggest, that using the somewhat lower-dimensional feature vectors would not cause issues in the captioner parts of the systems.

6. Improving the Object Detector

As stated, current state-of-the-art image captioners rely essentially on Faster R-CNN [40], which is already 5 years old – an eternity in the current world of deep learning –, and improvements to its feature richness made in 2018 [2]. This chapter draws from latest research in object detection to suggest improvements to the object detection components of the image captioning systems. First, in section 6.1, the Feature Pyramid Network, which can be used to improve Faster R-CNN and many other detectors, is discussed. Then two alternative, CNN-based one-stage object detectors are presented: first the YOLO family of object detectors (section 6.2) and then EfficientDet (section 6.3). In section 6.4, a transformer-based, anchor-free detection approach called Detection Transformer is presented. Finally, panoptic segmentation and bounding polygon approaches as alternatives to bounding box based detection is discussed (section 6.5) and the most important ways to improve the object detector are drawn together (section 6.6).

6.1 Feature Pyramid Network

A CNN computes a feature hierarchy layer by layer, and due to the systematical down-sampling between stages, the feature hierarchy has an inherent multi-scale, pyramid shape. Faster R-CNN uses features from just one resolution, 14×14 in the ResNet-101 based version presented, to find regions of interest, and then detects objects within those regions. The Feature Pyramid Network (FPN) [30] explicitly takes advantage of features at several scales: it uses a top-down pathway to combine features from different scales into aggregated features to be used in the object detection layers. There is one pyramid level for each stage, a set of CNN layers with the same scale, and it is based on the last layer of the stage. For ResNets, last layers of the Conv2[2], Conv3, Conv4, and Conv5 stages are included, but Conv1 is omitted due to its large memory footprint. Using an FPN, detections can be done at several scales instead of using just one feature scale as in the original Faster R-CNN. The basic multi-scale idea of an

[2]See any of the image captioning system architecture figures, e.g. figure 4.7 for reference on the five convolutional stages of ResNet-101.

FPN is shown in figure 6.1.

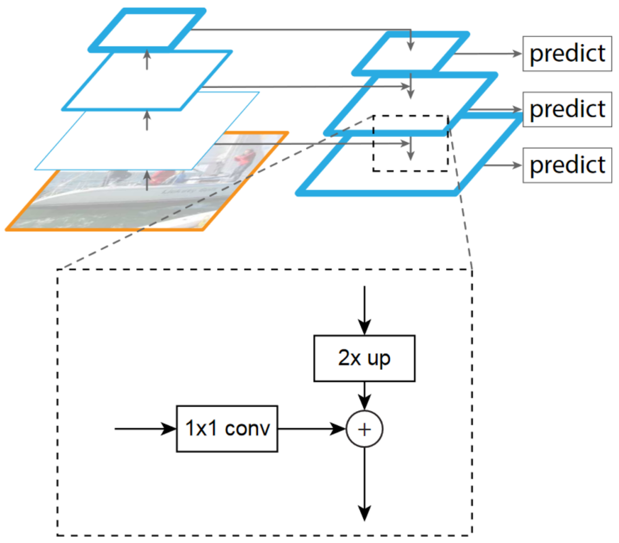


Figure 6.1: A Feature Pyramid Network uses top-down pathways to calculate aggregated features at several resolutions. Using an FPN, object detections can be done at several scales. The enriched multi-level feature maps are calculated by aggregation of upsampled higher-level feature maps and 1×1 convolved original feature maps at each level. Image source: Lin et al. [30].

The top-down pathway of the FPN upsamples the semantically stronger lower resolution features into a higher resolution. These features are enhanced using lateral connections with features from the original feature maps that were calculated in a bottom-up fashion. Each lateral connection merges feature maps of the same spatial size from the bottom-up and top-down pathways. The bottom-up feature maps have lower-level semantics, but they are more accurately localized. The aggregation is done by adding the upsampled higher-level feature map element-wise with the original feature map of that resolution, which has been run through a 1×1 convolution to reduce channels. After all enriched feature maps have been calculated, each enriched feature map is processed with a 3×3 convolution to reduce the aliasing effect of upsampling (the identicality of neighboring upsampled cells due to the upsampling process) to produce final enriched feature maps. The number of output channels is set to be similar in all enriched feature maps, 256 in the FPN paper, to support using the same object detectors on them. There are no non-linearities in the extra convolutional layers.

The FPN approach can be used with both one-stage and two-stage detectors like Faster R-CNN. Already in 2017, the FPN is used together with Faster R-CNN to improve the COCO object detection average precision (AP) score of Faster R-CNN by 2.3 points [30]. When used with Faster R-CNN, a region proposal network head (3×3 convolutional layer followed by sibling 1×1 convolutional layers for objectness and bounding box regression) is attached to each enriched FPN feature map. Anchor shapes 2:1, 1:1 and 1:2 are used at different resolution feature maps, and the 1:1 anchors have receptive field sizes of $32^2, 64^2, 128^2, 256^2, 512^2$ pixels at different scales from the most detailed to least detailed, bringing the total of different anchors to 15. The least detailed layer is calculated for RPN purposes by stride 2 subsampling the stage 5 layer – it is completely based on the features at the lowest resolution stage 5 of the feature pyramid. As to the object detection layers (Fast R-CNN in original Faster R-CNN), the RoI output by RPN is mapped to the corresponding enriched feature map. The design of the object detection head is also simplified: as the stage 5 convolutional layers are already used for creating the enriched feature pyramid, the object detection head is simplified to RoI pooling to extract 7×7 features, and then using two 1024-dimensional fully-connected layers, each followed by ReLU, before the classification and bounding box regression layer.

6.2 YOLO Family of Objection Detectors

The YOLO (You Only Look Once) family of object detectors is one of the well-known and leading object detection approaches. The YOLO detectors have had a strong focus on real-time object detection and inference speed, which is not the primary concern in image captioning.

YOLOv2 [38] uses a custom Darknet-19 CNN backbone and anchor boxes for object detection. Darknet-19 is a quite basic CNN, aiming at efficient training and inference. The anchor-box based bounding box prediction approach differs somewhat from Faster R-CNN by using k-means clustering to find good prior bounding box shapes, but is still a supervised anchor-box approach. YOLO9000 [38] expands the categories recognized by the object detector through training the detector with both image classification and object detection data and objectives. YOLO9000 uses this expanded cross-task training approach and WordTree, a rich hierarchical tree of visual concepts, to widen the categories recognized. The WordTree enables the detector to learn also non-exclusive concepts such as *terrier* and its parent concepts *hunting dog* and still further up the hierarchy *dog*. The WordTree concept hierarchy and its comparison to the mutually exclusive COCO categories and basic ImageNet categories is shown in figure 6.2.

YOLOv3 [39] refines the earlier versions. It uses a multilabel approach, and therefore uses independent logistic classifiers instead of softmax for predicting the most likely classes for each bounding box. This is a continuation of the hierarchical concept approach. YOLOv3 also uses multi-scale prediction similar to feature pyramid networks, predicting boxes at three different scales and taking advantage of convolutional feature maps at different scales. This approach makes it a one-stage detector (as opposed to

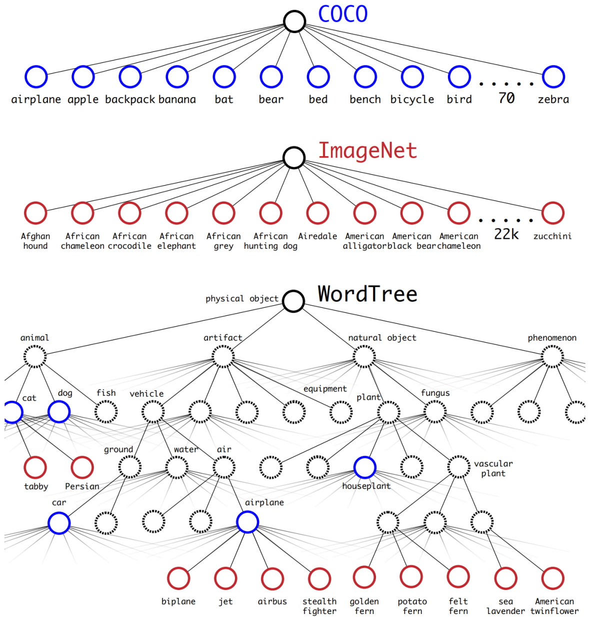


Figure 6.2: A hierarchical tree of visual concepts (WordTree). Image source: Redmon & Farhadi [38].

Faster R-CNN, which is a two-stage detector, which utilizes a region proposal network to select only some regions of interest for final object detection process). The backbone is an evolution of the earlier DarkNet-19, adding residual connections (similar to the ResNet architecture), ending up at a total of 53 layers and being named DarkNet-53. The accuracy of the backbone is on level with ResNet-101, but DarkNet-53 is able to run more frames per second at inference time. YOLOv4 [6] continues the mission of optimizing the operating speed of an object detector in production systems. It uses CSPDarknet53, another evolution of DarkNets, as the backbone, YOLOv3 as the object detection head, and some additional techniques (spatial pyramid pooling and a path aggregation network) to improve the data being fed to YOLOv3.

6.3 EfficientDet

Tan et al. [51] continue in the spirit of the systematically scalable architecture approach of EfficientNet, and propose a scalable object detection architecture, EfficientDet. EfficientDet uses EfficientNet backbones and a one-stage object detection approach consisting of a weighted bi-directional feature pyramid network (BiFPN) and simple class and bounding box prediction networks. The largest version EfficientDet-D7, which has 52M parameters, achieves new state-of-the-art 52.2 % COCO object detection average precision.

To achieve efficient multi-scale feature fusion, EfficientDet uses a weighted bi-directional feature pyramid network. This approach evolves the FPN approach by adding connections also in the bottom-up direction. In figure 6.3, a comparison of the architecture of the plain FPN and BiFPN is presented. The model also contains a weighted feature fusion approach, which cannot be covered in detail here; please refer to the EfficientDet paper [51] for details. The class and bounding box prediction networks follow the RetinaNet [31] structure and are a simple sequence of 3×3 convolutional layers followed by ReLU nonlinearities, with the box prediction network outputting 4 coordinates relative to the anchor location (similar to RPN), and the class prediction network outputting probabilities for each class. Note that the training of this simple object detection architecture relies heavily on the Focal loss approach [31], which is applied to all anchors, and which reduces the relative loss for well-classified examples and puts more focus on hard, misclassified examples.

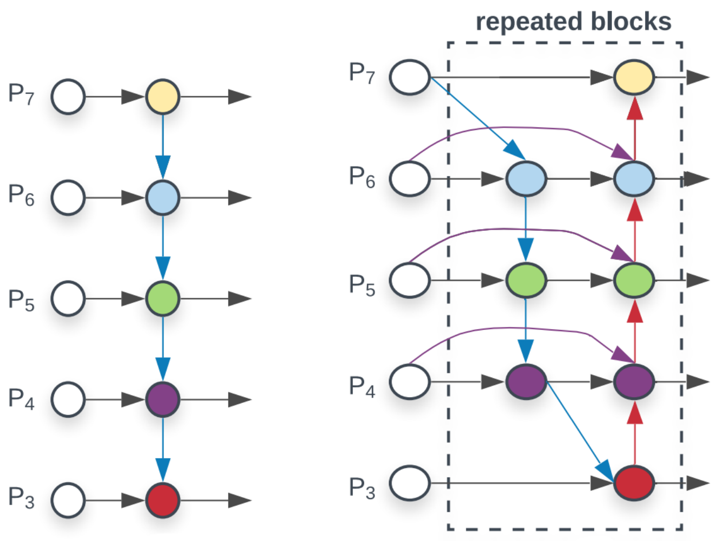


Figure 6.3: Architecture of a plain FPN with only lateral and top-down connections on the left and a bi-directional FPN (BiFPN) with both top-down and bottom-up connections on the right. BiFPN adds direct connections from the input of each layer to the corresponding output node. The stage numbers 3-7 correspond to the stages of the EfficientNet architecture. Image source: Tan et al. [51]. Image modified by omitting parts of the original.

The scaling method of EfficientDet is a logical continuation of the EfficientNet approach: it is most efficient to uniformly scale the resolution, depth, and width for all backbone, feature network, and box/class prediction networks at the same time. Backbone scaling parameters are as described in EfficientNet, except that the resolution is slightly modified to be compatible with the BiFPN. BiFPN scaling parameters are depth (how many consecutive BiFPN layers, starting at 1) and width, the number of channels in the BiFPN. The width of the box prediction and class prediction networks is fixed to be the same as the width of the BiFPN. The depth of the box prediction and class prediction networks is linearly increased by adding convolutional layers.

The architecture of the whole EfficientDet is shown in figure 6.4. The scaling parameters of the different size EfficientDet networks are summarized in figure 6.5.

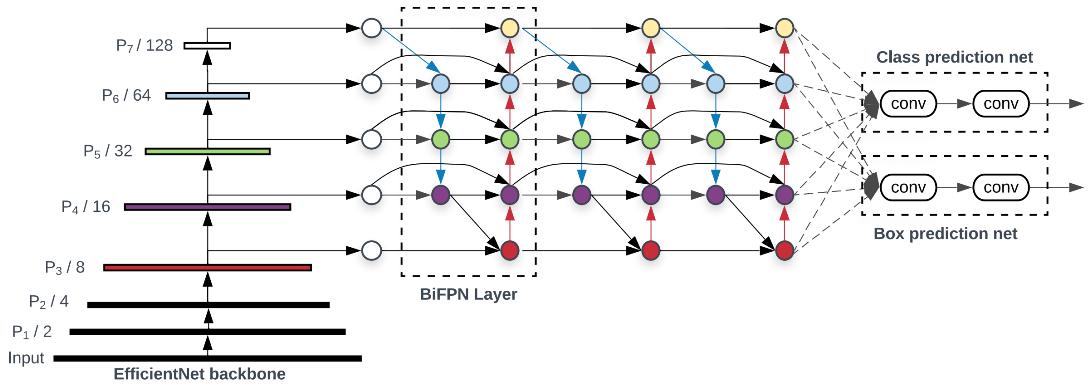


Figure 6.4: Architecture of the EfficientDet object detector. EfficientNet is used as the backbone, BiFPN as the feature network, and box and class prediction heads are simple convolution layer based networks. Image source: Tan et al. [51].

	Input size R_{input}	Backbone Network	BiFPN #channels W_{bifpn}	BiFPN #layers D_{bifpn}	Box/class #layers D_{class}
D0 ($\phi = 0$)	512	B0	64	3	3
D1 ($\phi = 1$)	640	B1	88	4	3
D2 ($\phi = 2$)	768	B2	112	5	3
D3 ($\phi = 3$)	896	B3	160	6	4
D4 ($\phi = 4$)	1024	B4	224	7	4
D5 ($\phi = 5$)	1280	B5	288	7	4
D6 ($\phi = 6$)	1280	B6	384	8	5
D6 ($\phi = 7$)	1536	B6	384	8	5

Figure 6.5: Parameters of EfficientDet object detection networks of different sizes. Image source: Tan et al. [51].

6.4 End-to-End Object Detection with Transformers

Carion et al. contribute to object detection with a transformer-based end-to-end trainable model named *Detection Transformer* (DETR) [9]. Detection Transformer belongs to the anchor-free family of object detectors [57], which directly find objects in an image without preset anchors. DETR architecture, depicted in figure 6.6, is simple: it consists of a CNN backbone, an encoder-decoder transformer and a feed-forward network to predict classes and bounding boxes. In DETR, object detection is approached as a direct set prediction problem. A key innovation in the model is usage of a bipartite matching loss function to calculate a correct amount of unique object instance predictions for an image.

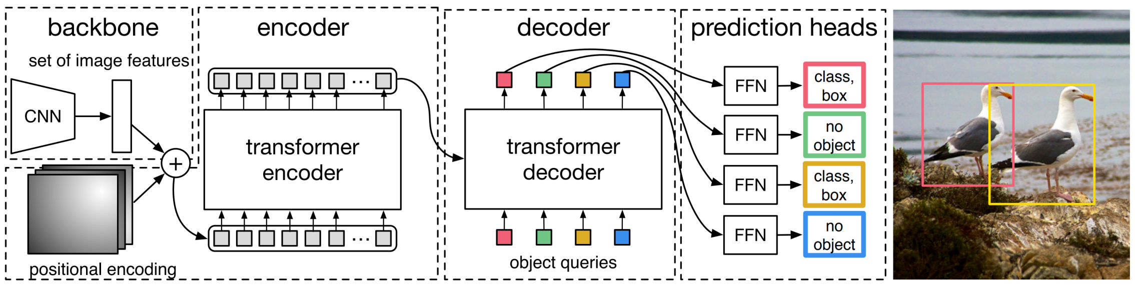


Figure 6.6: Detection Transformer (DETR) consists of a backbone CNN, transformer encoder-decoder and a number of feed-forward networks functioning as object class and bounding box prediction heads. Notice the blue and green "no object" detections, which are related to the fixed number of object queries per image. Image source: Carion et al. [9]

The transformer encoder first reduces the number of channels by using a 1×1 convolution and then collapses the spatial dimensions of the input feature maps into one dimension. The encoder layers follow a standard architecture: they first add fixed positional encodings to the input, then apply multi-head self-attention and finally use a feed-forward network. The transformer decoder takes as input a fixed small number of object queries, and attends to the encoder output. All object queries can be calculated in parallel. Each output embedding of the decoder is given as input to a shared 3-layer feed-forward network (FFN) with ReLU activation functions, and a final linear layer. The FFN outputs the bounding box center coordinates, height and width, and uses a softmax function to also predict the class label.

A central part of the model is the *bipartite matching loss function*, which uniquely assigns each prediction instance to a ground truth object instance. It is invariant to permutations of predicted objects, and therefore predictions of several object instances can computed in parallel [9]. DETR uses a fixed-size set on N predictions in a single pass through the decoder. N is a number significantly larger than the typical number of object instances in an image, for example 100. As N is larger than the number of object instances in the image, the set of predictions $\hat{y} = \{\hat{y}_i\}_{i=1}^{N}$ is padded with special *no object* class symbols ($\varnothing$). A bipartite matching between the ground truth set of objects y and the set of predictions $\hat{y}$ can be found by searching for a permutation of

N elements $\sigma \in \mathfrak{S}_N$ with the lowest cost:

$$\hat{\sigma} = \underset{\sigma \in \mathfrak{S}_N}{\arg\min} \sum_{i}^{N} \mathcal{L}_{\text{match}}(y_i, \hat{y}_{\sigma(i)}) \tag{6.1}$$

where $\mathcal{L}_{match}(y_i, \hat{y}_{\sigma(i)})$ is the pair-wise matching cost between ground truth y_i and the prediction with the index $\sigma(i)$ [9]. The optimal assignment of predictions to ground-truth labels can be done efficiently using the Hungarian algorithm. The matching cost considers both class prediction and similarity of boxes:

$$\mathcal{L}_{\text{match}}(y_i, \hat{y}_{\sigma(i)}) = -\mathbb{1}_{\{c_i \neq \varnothing\}} \hat{p}_{\sigma(i)}(c_i) + \mathbb{1}_{\{c_i \neq \varnothing\}} \mathcal{L}_{box}(b_i, \hat{b}_{\sigma(i)}) \tag{6.2}$$

where $\hat{p}_{\sigma(i)}(c_i)$ is the probability assigned to the predicted class, $\mathbb{1}$ is the indicator function, and $\mathcal{L}_{\text{box}}(b_i, \hat{b}_{\sigma(i)})$ is the loss between the ground truth and predicted boxes.

Once the bipartite matching is done, and all label-box predictions are aligned one-to-one with ground truth label-boxes, the total loss, which is also called the *Hungarian loss* is computed for all the pairs using the linear combination of the negative log-likelihood of the class prediction and the box loss:

$$\mathcal{L}_{\text{Hungarian}}(y, \hat{y}) = \sum_{i=1}^{N} \left[-\log \hat{p}_{\hat{\sigma}(i)}(c_i) + \mathbb{1}_{\{c_i \neq \varnothing\}} \mathcal{L}_{\text{box}}(b_i, \hat{b}_{\hat{\sigma}(i)}) \right] \tag{6.3}$$

The bounding box loss, which is used in both the matching loss and the Hungarian loss, is defined as a linear combination of the ℓ_1 loss and the generalized IoU loss:

$$\mathcal{L}_{\text{box}}(b_i, \hat{b}_{\hat{\sigma}(i)}) = \lambda_{\text{IoU}} \mathcal{L}_{\text{IoU}}(b_i, \hat{b}_{\sigma(i)}) + \lambda_{\text{L1}} \left\| b_i - \hat{b}_{\sigma(i)} \right\|_1 \tag{6.4}$$

where $\mathcal{L}_{\text{IoU}}(b_i, \hat{b}_{\sigma(i)})$ is an intersection-over-union loss function of the box, and $\lambda_{\text{IoU}}, \lambda_{\text{L1}} \in \mathbb{R}$ are hyperparameters.

DETR is reported to achieve better performance than Faster R-CNN on large objects, but performs worse on small objects [9]. This appears to be due to the fact that DETR uses more higher-level, lower-resolution features from the backbone CNN than Faster R-CNN. DETR is also slower to train than Faster R-CNN. Despite there deficiencies, the Detection Transformer does offer a very different approach to object detection than Faster R-CNN, and has potential for further development. It runs without anchor point generation, a region proposal network, and the non-maximum suppression procedure, greatly simplifying the object detection process.

6.5 Beyond Boxes: Masks, Polygons and Image Context

The leading image captioners rely on object feature vectors corresponding to rectangular areas chosen by the Faster R-CNN. There are several possible drawbacks to the bounding box based approach. Firstly, the foreground objects are rarely rectangular in the image, meaning there is possibly background noise encoded into the object detection. Secondly, the object detections do not contain data about the whole image. As stated when discussing image datasets and image-related machine learning tasks, the background stuff classes are a significant part of the image. These two problems seem like opposites: on the one hand the object detections are not focused tightly enough, on the other hand they are too tightly focused on only foreground objects.

The object detection approach could be modified in either of these directions. To make the object detections more tightly focused on the foreground objects, pixel-level masks or bounding polygons could be used. Detection Transformer [9] supports extending the model with a panoptic head to predict a binary mask for each of the predicted boxes, resulting in a panoptic segmentation of the image, including all pixels and both thing and stuff categories. A sample image with panoptic masks, and an illustration of the panoptic Detection Transformer architecture is shown in figure 6.7. The pixel-level mask variation based on Faster R-CNN is called Mask R-CNN [21]. It similarly predicts pixel-level masks objects for the image. The YOLOv3 detector is evolved to use bounding polygons instead of bounding boxes by the Poly-YOLO model [26].

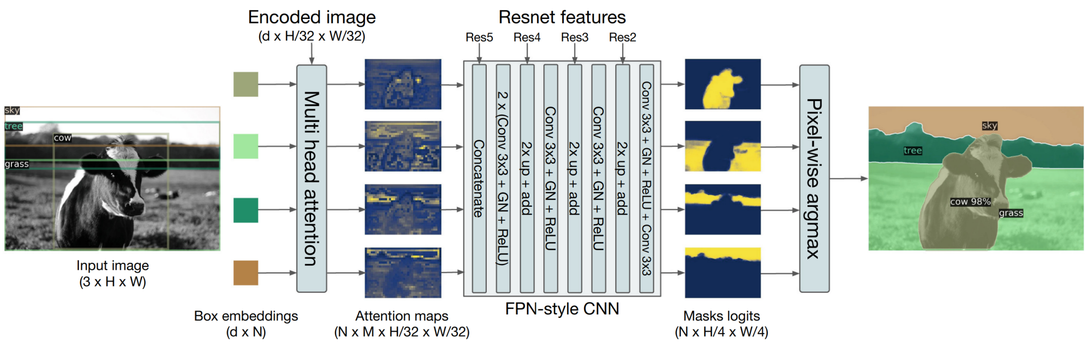


Figure 6.7: The Detection Transformer can be extended with a panoptic head to predict pixel-level masks for each thing and stuff object category in the image. Image source: Carion et al. [9].

For the image captioning task, the direction of including also background stuff objects and a full panoptic segmentation of the image seems like the most promising direction. Captions frequently refer to the environment of the picture, and having the stuff categories explicitly encoded in the object detection features would support attending to also these areas of the image during generation of the caption. The Bottom-Up feature vectors already include some information on the stuff categories thanks to the object detector training stage being done using the densely annotated Visual Genome dataset. As the Visual genome is still a relatively small dataset, the stuff richness of the feature vectors could still be improved.

6.6 How to Improve the Object Detector in Image Captioning

Faster R-CNN is still widely used as an object detector. In the field of CNN-based backbones and object detectors, current state-of-the art is the one-stage object detector EfficientDet, which uses the EfficientNet as backbone, a custom weighted bi-directional feature pyramid network as a feature network, and simple convolution-based class and bounding box prediction networks. Upgrading the object detector provides feature vectors that are better suited for object detection, and therefore provide a better quality grounding for attention in the image captioning components. The feature vectors output by EfficientDet are very similar to the ones output by Faster R-CNN, and the object detector and backbone components could be exchanged easily. The number of object detections per image would have to be defined - a good starting point is the range 30-100 used currently as input to the image captioning components.

The transformer-based object detection looks like a promising direction. The implementations so far have not surpassed convolution-based traditional object detection approaches, but the approach is still in early stages of development. There have not yet been transformer-based object detectors using Vision Transformers as backbone, but this kind of fully transformer-based object detector will certainly appear in the near future, and can possibly push the state-of-the art forward.

The most fundamental change in the object detector part of image captioning systems would be to move from bounding boxes to panoptic segmentation or bounding polygons. The basic version of the COCO dataset used for image captioning does not include stuff categories, but the more densely annotated Visual Genome (subset of COCO) does, and the stuff categories are used as objects in the object detector training phase. However, as the stuff categories are handled as normal objects with bounding boxes, the context provided by stuff categories and full panoptic segmentation is still largely an unexplored territory for image captioning. As the COCO-Stuff version of the dataset already includes metadata for stuff categories, using an object detector outputting feature vectors and a panoptic segmentation mask for each object category is already possible. The image captioner components would have to be adjusted to deal with the altered input, namely taking advantage of the pixel-level masks produced by the panoptic object detector, or alternatively encoding the information in the layers immediately preceding the segmentation mask prediction as part of the feature vectors fed into the image captioning components.

An important contribution to keep in mind is the feature vectors enriched with object attributes ("a *floral* dress") originally contributed by the Bottom-Up Top-Down [2] method. The attributes are important for the image captioning task, and a custom attribute loss function during training of an object detector is essential for this task, but not required by some other object detection based tasks. Utilizing rich region-based annotations of the Visual Genome dataset (region with caption "small houses on the hillside") could provide even more context relevant for image captioning, similar to what could be gained from the panoptic segmentation approach. The already semantically rich feature vectors are likely one of the reasons why the Faster R-CNN/Bottom up object detector is still effectively used in leading image captioning methods.

7. Improving the Image Captioning Layers and Top-level System Architecture

This chapter draws together the central innovations of the image captioning components from the leading image captioners in chapter 4 and discusses the overall architecture of the image captioning systems. Image captioning components are the high-level image encoder, the natural language decoder, and possible add-on caption editing components.

7.1 High-level Image Encoder

All high-level image encoders take the features precalculated by Anderson et al. in Bottom-Up Top-Down [2] as input. An exception is X-LAN that experiments with changing the backbone network from ResNet-101 to SENet-154 and recalculating the feature vectors. This variation still uses the same object detector and preserves the interface between the object detector and the high-level image encoder and the dimensionality of the object instance feature vectors. As discussed in the previous chapter, there are several options for using different backbones and object detectors, but they all output feature vectors of the same order of dimensionality. As the high-level encoders typically already use a fully connected layer to transform the input feature vectors to the internal dimensionality of the encoder, the exact dimensionality of the inputs is easy to adjust to. As discussed, the content of the feature vectors could include even richer feature vectors and encode the shape of the object in more detail than just bounding boxes. These types of improvements should not entail changes to the architecture of the image high-level encoder.

The main function of the high-level image encoder is to learn the interactions of the object features, both within the same object instance and between objects. All described encoders are based on attention refined in some way. Some are transformer-based, some implement the stackable encoder blocks architecture in alternate ways. Stacking of encoder blocks has been demonstrated by several methods to enable learning higher-level image features. Experimental results [12, 35] indicate that in practice performance starts to decrease already after 3-6 layers of stacked encoder blocks. Possible reasons for this are overfitting due to increased number of parameters with a fixed and limited training set size [35].

There is no single clearly best architecture for the image high-level encoder. The bilinear pooling attention approach demonstrated by X-LAN and the memory-augmented transformer-based approach of Meshed-Memory Transformers are the leading approaches, as they have demonstrated better results than the earlier Attention-on-Attention approach.

7.2 Natural Language Decoder

The goal of the natural language decoder component is to produce fluent natural language and attend to the relevant image parts and features based on the context of the decoder. Attending to relevant objects and features is an essential part of the architectures and design of the Meshed-Memory Transformer model, X-Linear Attention Network, Attention on Attention and Prophet Attention.

Prophet Attention shows that it is not adequate to look at the relevance of the image features related to the past time-steps: utilizing the future information available in the training phase to calculate ideal attention weights for image regions in the decoder, and guiding the main attention process using the ideal weights enables the model to attend to the most relevant regions for the current time step, and improves performance. The Prophet Attention paper demonstrates how to implement this for an LSTM-based decoder by utilizing a bidirectional LSTM for the training phase. The same approach should be quite easily implemented also in transformer-based decoders by implementing a separate step where the future words are not masked and ideal image feature attention for a time step is calculated, and those attention scores are then used to train the main attention path which does not see into the future. As the Prophet approach fixes the deviated focus problem, it should be used in both types of decoders.

The principle of learning to attend to relevant image feature based on the NLG decoder context was originally presented in the AoA paper. The principle was accepted and refined by $\mathcal{M}^2$ and X-LAN. Attending to both high-level and low-level image features is a major contribution of both the Meshed-Memory Transformer and the X-Linear Attention Network. X-LAN and LSTM-based models explicitly model the global image feature; $\mathcal{M}^2$ models it implicitly by learning gated attention to both low-

and high level features.

Good image caption generation results have been demonstrated using both LSTM-based and transformer-based decoders. Transformer-based decoders have the advantage of offering faster training due to being able to parallelize all time-steps in the word-level cross-entropy training phase. Based on the X-Linear Attention study [35], the performance difference is not large, but transformer-based decoders do achieve marginally better CIDEr-D score. Beam search of generated captions with a large enough width and reinforcement learning as the finetuning training approach can improve the fluency of the model, as demonstrated by the Meshed-Memory Transformer.

The described decoders have not utilized any transfer learning approaches. The fluency problems of the captions could be improved by using a caption-editing component (especially the denoising autoencoder) like Show, Edit and Tell. Another approach would be to utilize a *pretrained language model* such as GPT-3 [7] as part of the decoder. This could be implemented in the spirit of Chen et al. [11] by training a pointer generator that learns to alternate between attending to the image features and generating words based on just the pretrained language model. The pretrained model could also be used to supervise the fluency of the output and filter out captions from the beam search that do not fulfill a required fluency level. This kind of approach could lead towards a few-shot direction, but there is no transfer learning task or data available for the image high-level encoding at least currently.

7.3 System-Level Architecture

Image captioning systems are currently two stage-systems: the object detector detects objects in the image and encodes their features; and the image captioner takes the object features, interprets their interrelations and context, and generates a natural language caption. The connections between the components in the object detector – the backbone, feature networks or region proposal networks and class and location prediction networks – have been investigated, and there are several alternatives. Also the connections within the image captioner components – the image high-level encoder, natural language decoder and possible caption editing network – have been investigated. The only constant in the architecture has been the interface between the object detector and the image captioner. The interface is practical, as it enables training one part of the whole system while ignoring or keeping constant the other part. But logically there should be alternatives to also this interface. One alternative would be to output a *global context vector* from the object detector in addition to the foreground object detections. The role of this global context vector would be to encode information about the context of the whole image. A global context vector would be an alternative to adding the stuff categories and panoptic segmentation information to the object detections.

Within the current COCO captioning task and given the available components, the object detector should in all scenarios be pretrained using similar procedures as are currently being used. It would be possible to try letting the object detector layers be fine-tuned to the demands of the image captioning losses, instead of freezing it during the third stage of the training. This approach might turn out to not work in practice due to the massive size of the whole system and known problems with backpropagating gradients through very deep systems. The need for a system-level end-to-end training approach may also naturally arise from new datasets that offer more detailed grounding of captions to image areas. The Localized Narratives dataset already offers data in the format of mouse trace segments mapped to segments of the caption. Using Localized Narratives data, the caption words would have a supervised localized target in the image, which could be used to more firmly ground the caption words in the image. Using these kinds of current and future datasets, it may no longer be desirable or possible to freeze the object detector while training the image captioning components. But this kind of approach is a new machine learning task of its own, and may get its own competitions, perhaps succeeding the current COCO image captioning challenge as the most central captioning task. Datasets truly drive research.

8. Conclusions

This thesis has investigated the architectures of leading image captioning systems. The research question was: *What components and architectures are used in state-of-the-art image captioning systems and how could image captioning systems be further improved by utilizing improved components and architectures?*

Current leading image captioning systems consist of two subsystems: an object detector and an image captioner. A central finding was that the object detector used in leading systems, Faster R-CNN with Bottom-Up modifications, is somewhat outdated, and could quite easily be exchanged for a newer detector with proven superior performance. A low-risk way to improve the leading image captioning systems would be to replace Faster R-CNN with the state-of-the-art CNN-based one-stage object detector EfficientDet. As EfficientDet has much improved object detection accuracy demonstrated on the COCO object detection task, and the interface between the object detector and image captioner components is quite simple, this change should improve the image captioning performance. The attribute richness of the feature vectors would have to be ensured with a custom attribute training objective for the object detector, similar to that implemented by Anderson et al. [2], and training the object detector on a densely annotated dataset such as Visual Genome. Additionally, experimenting with a panoptic segmentation based encoding could improve the context data essential to generating high-quality captions.

Effective attention mechanisms are essential in the image captioning components. In the high-level image encoder, it is essential to learn the interactions between the object features, different object instances and the whole image context. In the natural language decoder, attending to the relevant image parts and features based on the context of the decoder is essential. Best ways to implement these attention mechanisms are currently the Meshed-Memory Transformer and the bilinear pooling attention of the X-Linear Attention Networks. Implementing the Prophet Attention approach of using the future words available in the supervised training phase to guide the main decoder attention path is an important addition, which fixes the deviated focus problem.

Vision Transformer based backbones have recently been shown to surpass the accuracy of the best CNN-based backbones. Transformer-based object detectors are getting close to the performance of CNN-based object detectors. The Meshed-Memory Transformer has shown a fully transformer-based image-captioner can reach top-level results. A likely future is that there will be very competitive completely transformer-based image-captioning systems, where the backbone, object detector, image high-level encoder and natural language decoder are all based on different kinds of transformer blocks.

A caption-editor has been shown to be an independent add-on component, that could be added onto any other image captioner as the final component. The caption editing approach has not, however, been demonstrated to improve the quality of the captions compared to state-of-the-art systems without an editing component. In the area of caption postprocessing, the use of pretrained language models, such as GPT-3, to supervise the generation of language is a potential approach.

The role of large datasets in pretraining the backbone are essential. A richer semantic training of backbones using a semantic softmax approach and the larger version of the ImageNet dataset, ImageNet-21k, can improve the quality of the learned features in backbones of all sizes. The semantically dense annotations of the Visual Genome dataset have been essential in training the object detector in currently leading image captioning systems. New ways to ground captions in images in a more detailed ways, such as the Localized Narratives dataset, offer possibilities for learning a more accurate connection between the image and the caption. If the coupling of the image and the caption grows tighter, end-to-end training approaches for the whole system will have to be investigated. It may no longer be desirable or possible to train the object detector as an independent subsystem.

Image captioning systems are one of the flagships on artificial intelligence research. They utilize and help push the boundaries in research in the fields of computer vision and natural language generation. The speed of development in these fields is very fast. It is important to understand where the research community currently stands with image captioning, but it is also certain, that the current state-of-the-art is legacy within a few years. But it is also very likely, that the new state-of-the-art grows from the current leading approaches and intellectual seeds planted today.

References

[1] P. Anderson, B. Fernando, M. Johnson, and S. Gould. Spice: Semantic propositional image caption evaluation. In B. Leibe, J. Matas, N. Sebe, and M. Welling, editors, *Computer Vision – ECCV 2016*, pages 382–398, Cham, 2016. Springer International Publishing.

[2] P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[3] P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang. Bottom-up and top-down attention for image captioning and visual question answering, supplementary materials. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[4] J. T. Barron. Continuously differentiable exponential linear units. *CoRR*, 1704.07483, 2017.

[5] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall. A dataset for semantic segmentation of point cloud sequences. *CoRR*, abs/1904.01416, 2019.

[6] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao. Yolov4: Optimal speed and accuracy of object detection. *CoRR*, 2004.10934, 2020.

[7] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, abs/2005.14165, 2020.

[8] H. Caesar, J. R. R. Uijlings, and V. Ferrari. Coco-stuff: Thing and stuff classes in context. *CoRR*, abs/1612.03716, 2016.

[9] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. End-to-end object detection with transformers. In A. Vedaldi, H. Bischof, T. Brox,

and J.-M. Frahm, editors, *Computer Vision – ECCV 2020*, pages 213–229, Cham, 2020. Springer International Publishing.

[10] M. Caron, H. Touvron, I. Misra, H. Jegou, J. Mairal, P. Bojanowski, and A. Joulin. Emerging properties in self-supervised vision transformers. *CoRR*, 2104.14294, 2021.

[11] Z. Chen, H. Eavani, W. Chen, Y. Liu, and W. Y. Wang. Few-shot nlg with pre-trained language model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 183–190, 2020.

[12] M. Cornia, M. Stefanini, L. Baraldi, and R. Cucchiara. Meshed-memory transformer for image captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[13] M. Cornia, M. Stefanini, L. Baraldi, and R. Cucchiara. Meshed-memory transformer for image captioning, supplementary material. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[14] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[15] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, 2010.11929, 2020.

[16] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[17] C. Gan, Z. Gan, X. He, J. Gao, and L. Deng. Stylenet: Generating attractive visual captions with styles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[18] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[19] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[20] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[21] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[22] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[23] M. Z. Hossain, F. Sohel, M. F. Shiratuddin, and H. Laga. A comprehensive survey of deep learning for image captioning. *ACM Comput. Surv.*, 51(6), Feb. 2019.

[24] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[25] L. Huang, W. Wang, J. Chen, and X.-Y. Wei. Attention on attention for image captioning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[26] P. Hurtik, V. Molek, J. Hula, M. Vajgl, P. Vlasanek, and T. Nejezchleba. Poly-yolo: higher speed, more precise detection and instance segmentation for yolov3. *CoRR*, abs/2005.13243, 2020.

[27] A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[28] Z. Kasner and O. Dusek. Data-to-text generation with iterative text editing. *CoRR*, 2011.01694, 2021.

[29] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International journal of computer vision*, 123(1):32–73, 2017.

[30] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[31] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[32] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár. Microsoft coco: Common objects in context. *CoRR*, abs/1405.0312, 2015.

[33] F. Liu, X. Ren, X. Wu, S. Ge, W. Fan, Y. Zou, and X. Sun. Prophet attention: Predicting attention with future attention. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1865–1876. Curran Associates, Inc., 2020.

[34] D. Mahajan, R. Girshick, V. Ramanathan, K. He, M. Paluri, Y. Li, A. Bharambe, and L. van der Maaten. Exploring the limits of weakly supervised pretraining. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[35] Y. Pan, T. Yao, Y. Li, and T. Mei. X-linear attention networks for image captioning. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10968–10977, 2020.

[36] B. A. Plummer, L. Wang, C. M. Cervantes, J. C. Caicedo, J. Hockenmaier, and S. Lazebnik. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. In *Proceedings of the IEEE international conference on computer vision*, pages 2641–2649, 2015.

[37] J. Pont-Tuset, J. Uijlings, S. Changpinyo, R. Soricut, and V. Ferrari. Connecting vision and language with localized narratives. In *ECCV*, 2020.

[38] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[39] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018.

[40] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017.

[41] S. J. Rennie, E. Marcheret, Y. Mroueh, J. Ross, and V. Goel. Self-critical sequence training for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[42] T. Ridnik, E. Ben-Baruch, A. Noy, and L. Zelnik-Manor. Imagenet-21k pretraining for the masses. *arXiv preprint arXiv:2104.10972*, abs/2104.10972, 2021.

[43] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[44] F. Sammani and L. Melas-Kyriazi. Show, edit and tell: A framework for editing image captions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[45] F. Sammani and L. Melas-Kyriazi. Supplementary material for show, edit and tell: A framework for editing image captions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[46] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[47] K. Shuster, S. Humeau, H. Hu, A. Bordes, and J. Weston. Engaging image captioning via personality. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[48] C. Sun, A. Shrivastava, S. Singh, and A. Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[49] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[50] M. Tan and Q. Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR, 09–15 Jun 2019.

[51] M. Tan, R. Pang, and Q. V. Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10781–10790, 2020.

[52] H. Touvron, M. Cord, A. Sablayrolles, G. Synnaeve, and H. Jégou. Going deeper with image transformers. *arXiv preprint arXiv:2103.17239*, abs/2103.17239, 2021.

[53] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors,

Advances in Neural Information Processing Systems 30, pages 5998–6008. Curran Associates, Inc., 2017.

[54] R. Vedantam, C. Lawrence Zitnick, and D. Parikh. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[55] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[56] S. Zagoruyko and N. Komodakis. Wide residual networks. *CoRR*, abs/1605.07146, 2016.

[57] S. Zhang, C. Chi, Y. Yao, Z. Lei, and S. Z. Li. Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. *arXiv*, abs/1912.02424, 2020.

[58] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

Appendix A. COCO Online Test Server Metrics

User	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROUGE-L	CIDEr-D
MSR-MS_Cog_Svcs	0.969 (1)	0.924 (1)	0.847 (1)	0.749 (1)	0.408 (1)	0.768 (1)	1.387 (1)
xiaoma666	0.965 (6)	0.916 (5)	0.837 (5)	0.738 (6)	0.394 (7)	0.750 (15)	1.360 (2)
yinanlee	0.966 (4)	0.919 (2)	0.841 (2)	0.744 (2)	0.396 (4)	0.753 (4)	1.360 (3)
IVA-CASIA	0.962 (12)	0.914 (8)	0.836 (6)	0.739 (5)	0.398 (2)	0.758 (2)	1.357 (4)
Young24	0.965 (5)	0.916 (6)	0.836 (7)	0.737 (7)	0.394 (6)	0.750 (11)	1.354 (5)
yun3300612	0.966 (3)	0.917 (4)	0.838 (4)	0.740 (4)	0.396 (3)	0.753 (6)	1.354 (6)
mrwu_xmu	0.967 (2)	0.919 (3)	0.840 (3)	0.742 (3)	0.393 (9)	0.752 (8)	1.345 (7)
mayiwei	0.963 (10)	0.912 (10)	0.832 (9)	0.733 (10)	0.391 (12)	0.747 (23)	1.341 (8)
zxyXMU	0.964 (8)	0.913 (9)	0.830 (11)	0.731 (13)	0.391 (14)	0.746 (24)	1.340 (9)
DOTrans	0.965 (7)	0.915 (7)	0.834 (8)	0.736 (8)	0.390 (15)	0.750 (14)	1.339 (10)
Prophet	0.963 (11)	0.912 (12)	0.832 (10)	0.733 (9)	0.393 (8)	0.751 (9)	1.337 (11)
Tsinghua-Samsung	0.960 (17)	0.908 (17)	0.827 (18)	0.727 (18)	0.395 (5)	0.756 (3)	1.336 (12)
Yingwei.Pan	0.957 (24)	0.905 (20)	0.825 (21)	0.724 (22)	0.392 (11)	0.750 (13)	1.335 (13)
LSDT	0.963 (9)	0.912 (11)	0.830 (12)	0.730 (14)	0.389 (19)	0.748 (21)	1.331 (14)
DiMBERT	0.961 (15)	0.910 (13)	0.830 (13)	0.731 (12)	0.392 (10)	0.750 (18)	1.331 (15)
gzx3227556	0.957 (25)	0.904 (26)	0.822 (25)	0.722 (25)	0.387 (29)	0.739 (46)	1.328 (16)
luo3300612	0.961 (16)	0.909 (16)	0.828 (14)	0.729 (15)	0.388 (25)	0.744 (31)	1.325 (17)
lifeGAN	0.962 (14)	0.910 (14)	0.828 (15)	0.728 (17)	0.388 (27)	0.744 (35)	1.325 (18)
kleinLee	0.959 (19)	0.904 (22)	0.822 (27)	0.721 (29)	0.388 (28)	0.739 (44)	1.324 (19)
zxy1004	0.962 (13)	0.909 (15)	0.827 (19)	0.725 (19)	0.387 (30)	0.742 (39)	1.324 (20)
Meshed-Memory-Tr.	0.960 (18)	0.908 (18)	0.827 (17)	0.728 (16)	0.390 (18)	0.748 (22)	1.321 (21)
KingSoft_AILAB	0.957 (23)	0.906 (19)	0.825 (20)	0.725 (20)	0.388 (26)	0.750 (16)	1.317 (22)
Bridging_the_Gap	0.957 (22)	0.904 (25)	0.822 (26)	0.722 (24)	0.390 (16)	0.746 (25)	1.316 (23)
songzl	0.952 (51)	0.904 (24)	0.827 (16)	0.732 (11)	0.390 (17)	0.753 (5)	1.315 (24)
IVA-HUAWEI	0.956 (30)	0.903 (27)	0.822 (24)	0.722 (26)	0.389 (24)	0.749 (20)	1.314 (25)
gzx3221466	0.958 (21)	0.902 (30)	0.816 (36)	0.713 (38)	0.380 (50)	0.735 (61)	1.311 (26)
Curya2	0.956 (29)	0.904 (23)	0.823 (23)	0.723 (23)	0.391 (13)	0.752 (7)	1.310 (27)
tcts-2	0.955 (32)	0.902 (29)	0.821 (28)	0.722 (27)	0.389 (20)	0.751 (10)	1.309 (28)
MIL-HDU	0.956 (27)	0.905 (21)	0.824 (22)	0.724 (21)	0.389 (23)	0.750 (12)	1.309 (29)
VisualPersistence_E4	0.955 (33)	0.902 (28)	0.821 (29)	0.721 (28)	0.389 (22)	0.749 (19)	1.306 (30)
dry	0.958 (20)	0.901 (31)	0.817 (33)	0.715 (34)	0.386 (31)	0.741 (41)	1.306 (31)
TCTS	0.954 (38)	0.901 (34)	0.819 (30)	0.720 (30)	0.389 (21)	0.750 (17)	1.304 (32)
GAT_Chi	0.951 (52)	0.897 (43)	0.815 (38)	0.714 (36)	0.384 (35)	0.744 (30)	1.298 (33)
AoANet	0.950 (56)	0.896 (48)	0.813 (42)	0.712 (41)	0.385 (33)	0.745 (29)	1.296 (34)
birdl	0.954 (39)	0.900 (36)	0.817 (34)	0.715 (33)	0.384 (36)	0.745 (26)	1.296 (35)
fy1994	0.953 (42)	0.899 (38)	0.815 (37)	0.714 (37)	0.384 (39)	0.744 (34)	1.296 (36)
VisualPersistence	0.954 (37)	0.901 (33)	0.818 (31)	0.717 (31)	0.384 (37)	0.745 (28)	1.295 (37)
CVDDL	0.954 (36)	0.900 (35)	0.817 (32)	0.716 (32)	0.384 (38)	0.745 (27)	1.295 (38)
erictyloo	0.954 (40)	0.897 (42)	0.812 (44)	0.711 (43)	0.386 (32)	0.737 (50)	1.294 (39)
JLTX	0.951 (53)	0.896 (47)	0.813 (41)	0.712 (40)	0.383 (43)	0.742 (38)	1.293 (40)
ooo	0.952 (47)	0.898 (41)	0.812 (43)	0.709 (45)	0.382 (47)	0.740 (42)	1.290 (41)
han-liu18	0.952 (49)	0.896 (45)	0.811 (46)	0.709 (44)	0.384 (42)	0.743 (37)	1.289 (42)
NGSAN	0.950 (59)	0.893 (50)	0.806 (50)	0.702 (48)	0.384 (34)	0.740 (43)	1.286 (43)
TJUCaption	0.956 (28)	0.899 (39)	0.811 (45)	0.708 (46)	0.384 (41)	0.739 (45)	1.286 (44)
longyuyang	0.954 (34)	0.896 (46)	0.806 (49)	0.702 (49)	0.382 (49)	0.736 (54)	1.281 (45)
cxq	0.949 (69)	0.891 (53)	0.804 (51)	0.700 (52)	0.380 (52)	0.737 (51)	1.278 (46)
TencentAI.v2	0.955 (31)	0.900 (37)	0.809 (48)	0.701 (50)	0.377 (66)	0.737 (53)	1.278 (47)
DY-APR	0.952 (50)	0.897 (44)	0.813 (39)	0.711 (42)	0.383 (45)	0.743 (36)	1.276 (48)
wyong	0.946 (93)	0.887 (77)	0.798 (73)	0.693 (69)	0.382 (48)	0.735 (60)	1.276 (49)
cryingface	0.947 (78)	0.889 (64)	0.803 (54)	0.700 (51)	0.384 (40)	0.742 (40)	1.276 (50)

User	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROUGE-L	CIDEr-D
TXH-mercury	0.950 (58)	0.894 (49)	0.809 (47)	0.708 (47)	0.383 (44)	0.744 (33)	1.273 (51)
huangyq3	0.949 (63)	0.891 (58)	0.800 (64)	0.694 (68)	0.376 (72)	0.733 (67)	1.270 (52)
AnonymousApproach4	0.946 (91)	0.888 (71)	0.800 (62)	0.696 (59)	0.379 (59)	0.736 (55)	1.268 (53)
SRCB_ML_Lab	0.954 (35)	0.898 (40)	0.813 (40)	0.713 (39)	0.373 (78)	0.731 (83)	1.267 (54)
GLIED	0.945 (103)	0.887 (76)	0.798 (72)	0.695 (63)	0.380 (55)	0.736 (56)	1.267 (55)
dlq	0.947 (86)	0.887 (73)	0.799 (71)	0.693 (72)	0.379 (58)	0.735 (62)	1.266 (56)
aacccclll	0.946 (94)	0.887 (75)	0.801 (59)	0.700 (53)	0.380 (53)	0.738 (49)	1.265 (57)
Pro-LSTM-single	0.948 (72)	0.890 (61)	0.802 (57)	0.697 (56)	0.380 (51)	0.737 (52)	1.265 (58)
Caption_Man	0.948 (74)	0.889 (65)	0.800 (65)	0.694 (64)	0.375 (75)	0.733 (71)	1.259 (59)
LosAn	0.948 (71)	0.891 (56)	0.801 (58)	0.694 (65)	0.374 (77)	0.735 (63)	1.258 (60)
rookieno.1	0.946 (88)	0.886 (78)	0.794 (85)	0.686 (87)	0.373 (81)	0.730 (84)	1.258 (61)
nanly	0.944 (107)	0.885 (89)	0.795 (81)	0.691 (80)	0.377 (65)	0.734 (65)	1.258 (62)
caption_recall	0.947 (84)	0.885 (84)	0.792 (89)	0.684 (90)	0.372 (90)	0.727 (93)	1.257 (63)
ETN-single_model	0.947 (82)	0.888 (70)	0.799 (67)	0.695 (60)	0.378 (63)	0.735 (58)	1.257 (64)
qqqqmy	0.944 (109)	0.883 (91)	0.795 (82)	0.692 (76)	0.379 (57)	0.735 (64)	1.256 (65)
jianwang123	0.944 (111)	0.886 (81)	0.799 (70)	0.695 (61)	0.379 (62)	0.735 (59)	1.256 (66)
LALALA2	0.948 (75)	0.889 (62)	0.799 (68)	0.692 (78)	0.373 (80)	0.733 (66)	1.256 (67)
yucheng	0.950 (55)	0.891 (57)	0.799 (69)	0.693 (74)	0.373 (83)	0.733 (72)	1.256 (68)
Pro-LSTM	0.943 (117)	0.882 (93)	0.791 (91)	0.686 (86)	0.376 (70)	0.731 (80)	1.255 (69)
MultiKK	0.946 (95)	0.885 (82)	0.796 (79)	0.692 (77)	0.379 (61)	0.736 (57)	1.254 (70)
junhaohahaha	0.949 (64)	0.887 (74)	0.798 (75)	0.693 (73)	0.376 (69)	0.729 (89)	1.252 (71)
jasonlxz1234	0.946 (96)	0.885 (88)	0.792 (90)	0.684 (91)	0.372 (89)	0.729 (88)	1.252 (72)
mikewallace250	0.949 (68)	0.888 (69)	0.795 (84)	0.687 (85)	0.371 (91)	0.730 (85)	1.252 (73)
RDN_Res	0.953 (44)	0.890 (59)	0.801 (61)	0.695 (62)	0.378 (64)	0.733 (69)	1.252 (74)
yss22691	0.943 (116)	0.882 (95)	0.791 (93)	0.684 (93)	0.372 (87)	0.727 (91)	1.251 (75)
AnonymousTeam	0.950 (62)	0.893 (51)	0.801 (60)	0.692 (75)	0.372 (86)	0.731 (79)	1.251 (76)
lkkkkkk	0.950 (60)	0.889 (67)	0.802 (55)	0.698 (55)	0.379 (60)	0.733 (68)	1.250 (77)
CapJK	0.953 (43)	0.891 (52)	0.803 (53)	0.698 (54)	0.377 (67)	0.733 (70)	1.247 (78)
ZhengYi_FDU	0.945 (105)	0.885 (86)	0.795 (83)	0.689 (82)	0.373 (79)	0.730 (86)	1.246 (79)
TingYao	0.949 (70)	0.889 (66)	0.798 (74)	0.691 (79)	0.373 (85)	0.729 (87)	1.246 (80)
AnonymousResearcher	0.956 (26)	0.901 (32)	0.817 (35)	0.715 (35)	0.382 (46)	0.744 (32)	1.243 (81)
clw	0.944 (106)	0.882 (98)	0.787 (98)	0.677 (99)	0.369 (98)	0.725 (96)	1.242 (82)
susijixx	0.950 (61)	0.891 (54)	0.803 (52)	0.697 (57)	0.380 (54)	0.738 (47)	1.242 (83)
ttry_speak	0.952 (46)	0.891 (55)	0.802 (56)	0.696 (58)	0.376 (73)	0.732 (76)	1.240 (84)
LiuDaqing	0.949 (65)	0.888 (68)	0.797 (77)	0.690 (81)	0.370 (94)	0.731 (82)	1.238 (85)
a406599452	0.945 (100)	0.882 (96)	0.788 (97)	0.678 (98)	0.368 (101)	0.724 (99)	1.237 (86)
stack_ccap	0.949 (67)	0.889 (63)	0.800 (63)	0.694 (66)	0.379 (56)	0.738 (48)	1.237 (87)
dccdeedde	0.945 (102)	0.882 (94)	0.788 (96)	0.678 (97)	0.367 (106)	0.723 (101)	1.233 (88)
weihy	0.947 (85)	0.888 (69)	0.799 (66)	0.694 (67)	0.370 (93)	0.733 (73)	1.232 (89)
chenlizhi	0.948 (73)	0.886 (80)	0.791 (92)	0.684 (92)	0.368 (103)	0.731 (81)	1.232 (90)
gnsa6	0.944 (108)	0.878 (105)	0.782 (105)	0.674 (106)	0.368 (102)	0.726 (94)	1.226 (91)
Cascaded-Agents	0.943 (118)	0.877 (109)	0.780 (108)	0.668 (111)	0.367 (111)	0.721 (107)	1.226 (92)
wzn0828	0.940 (128)	0.874 (121)	0.777 (118)	0.668 (112)	0.367 (109)	0.720 (112)	1.224 (93)
A_SSRP	0.953 (45)	0.890 (60)	0.797 (76)	0.686 (88)	0.372 (88)	0.731 (78)	1.224 (94)
BrianJ	0.949 (66)	0.885 (87)	0.793 (87)	0.688 (83)	0.367 (105)	0.720 (111)	1.223 (95)
fkxssaa	0.944 (112)	0.880 (99)	0.784 (101)	0.674 (105)	0.370 (95)	0.722 (103)	1.220 (96)
stack_vs_cap	0.944 (110)	0.882 (97)	0.787 (99)	0.677 (101)	0.369 (97)	0.725 (97)	1.220 (97)
NAIC-CMAL	0.943 (120)	0.872 (124)	0.772 (128)	0.661 (127)	0.364 (120)	0.720 (109)	1.212 (98)
hello_man	0.934 (138)	0.867 (132)	0.774 (125)	0.667 (115)	0.365 (118)	0.709 (143)	1.211 (99)
wsw	0.939 (130)	0.874 (119)	0.778 (115)	0.667 (114)	0.367 (108)	0.720 (110)	1.210 (100)

Table A.1: COCO online test server metrics (c40 versions), top 100 models by CIDEr-D c40 per 31.3.2021. Data source: Microsoft COCO Image Captioning Challenge, `https://competitions.codalab.org/competitions/3221`.