Multi-agent reinforcement learning and object detection asstructured prediction

Nicolas Carion

PSL
UNIVERSITÉ PARIS

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée à l'Université Paris-Dauphine

Multi-agent reinforcement learning and object detection as structured prediction

Soutenue par
Nicolas CARION
Le 4 Décembre 2020

École doctorale n°543
École doctorale SDOSE

Spécialité
CNU 27 - Informatique

Composition du jury :

Cordelia Schmid Google Research INRIA	*Présidente*
Florent Perronnin Naver Labs	*Rapporteur*
Karl Tuyls Google Deepmind	*Rapporteur*
Jessica Hamrick Google Deepmind	*Examinatrice*
Gabriel Synnaeve Facebook AI Research	*Examinateur*
Alessandro Lazaric Facebook AI Research	*Examinateur*
Nicolas Usunier Facebook AI Research	*Co-directeur de thèse*
Tristan Cazenave Université Paris-Dauphine	*Directeur de thèse*

Dauphine | PSL
UNIVERSITÉ PARIS

UNIVERSITÉ PARIS DAUPHINE
FACEBOOK AI RESEARCH

Abstract

Université Paris Dauphine
Facebook AI Research

Doctor of Philosophy

Multi-agent reinforcement learning and object detection as structured prediction

by Nicolas CARION

This thesis explores the use of structured losses in two different domains. In the first contribution, we focus on multi-agent reinforcement learning (MARL), in environments that can be separated into several loosely coupled tasks. We set out to find policies that can generalize well to more agents and tasks than seen during training, effectively scaling up the size of problems that can be tackled. Our solution assigns agents to tasks by approximately solving a centralized optimization problem whose objective function is parameterized by a neural network. We study how the expressivity of the optimization problem and that of the neural network influence the generalization capabilities of the model, and show that with the right choices, the policy can generalize to more than 5 times more agents than seen during training.

In the second contribution we formulate object detection as a set prediction problem, and design a model that can effectively tackle this formulation. Our solution leverages a deep convolutional network, as is customary in computer vision, and a transformer encoder-decoder network, an architecture that has enabled significant progress in natural language processing. Crucially, our solution incorporates minimal inductive bias, thereby alleviating the need for hand-designed detection-specific components such as anchors or non-maximal suppression. With a comparable parameter budget, our model matches the performance of well-established and highly-optimized baselines such as Retinanet and Faster R-CNN on the challenging COCO detection dataset. Finally, we show that the method can be naturally extended to perform panoptic segmentation, where it outperforms competing approaches, thus showing the versatility of the model.

Acknowledgements

First and foremost, I wish to thank my advisors, Nicolas Usunier, Gabriel Synnaeve and Alessandro Lazaric in Facebook, and Tristan Cazenave in Paris Dauphine University, for giving me the opportunity to embark on this three-year journey. Your mentorship, support, trust and friendship were invaluable in helping me navigate the tumult of a PhD. I owe you some key steps on the path to becoming a researcher. I also thank my jury, Jessica Hamrick, Cordelia Schmid, Karl Tuyls and Florent Perronnin. I am extremely grateful to have been able to conduct most of this PhD in the exceptional offices of Facebook AI Research in Paris. The environment was what every student can dream for, for the ability to connect to so many talented people across the world, both for intense research brainstorming as well as fun social interactions. Special acknowledgement to my PhD friends, Guillaume Lample, Alexandre Sablayrolles, Alexandre Defossez, Neil Zegidour, Rahma Chabouni, Angela Fan, Louis Martin, Pierre Stock, Léonard Blier, Pauline Luc, Alexis Conneau, Adrien Dufraux, as well as all my Foosball partners for the countless games that helped me keep my sanity. I also sincerely thank my closest collaborators, Sergey Zagoruyko and Francisco Massa, for these intense, thought-provoking, enriching adventures that got us much further than I could have hoped for or forseen. Thanks to my friends, Danielle, Robin, Ludovic, Élodie, Jessica, Luke, Amélie, Romain for their support, and for occasionally pulling me out of the research bubble. Special thanks to my family, without whom I would not be where I am today, if it was not for their unconditional support through every step, small or big. Finally, last but not least, I thank you, Aishwarya, mon amour, for bringing so much light to my life, and giving me the energy to finish this adventure even through the grim auspices of a global pandemic.

Contents

List of Figures

List of Tables

0.1 Foreword and organization of the manuscript

When I started my PhD at Facebook in September 2017, I joined a team which was aiming at making progress in the game of Starcraft, an involved real-time strategy game. The challenges in this domain were plenty, since little progress had been made in the field, and the gap with the best human players and the best bots was still significant, but we managed to make some progress (Synnaeve, Z. Lin, et al., 2018; Synnaeve, Gehring, et al., 2019) . Given the sheer complexity of the game, I decided to focus mainly on methods that can generalize well, so as to reduce the learning burden: no need for the agent to see all the possible states of the game if it can leverage effectively at test-time what was learned in the small training scenarios.

Towards the end of 2018, a competing team made a press release around a method they had developed, and gained a lot of media coverage. Their approach was very different, since they were mainly relying on imitation of human experts, thereby alleviating a lot of the issues that arise when trying to learn to play this kind of games from scratch. Nonetheless, the interest of Facebook for this research subsided, as it was hard to justify the sustained significant resource investments it entails. With the project winding down, it was hardly conceivable for me to continue working on Starcraft, due to the lack of engineering support.

From there on, I decided to focus on simpler environments, such as the Atari emulation platform (Bellemare et al., 2013), to study model-based reinforcement learning approaches. My idea was that if one can learn a compact model of the dynamics, then this model can potentially generalize to slightly more complex environments (eg. more enemies), and thus enabling the agent to generalize as well. My plan to achieve this was to learn a model at the object level. Instead of trying to predict the RGB pixels of the screen directly, which was the dominant approach at the time, I wanted to predict the behavior of each individual entity in the game. The goal was to build a compact relational model of the environment, where we learn how each pair of entities interact. This has a potential for generalization: most of the interactions are very local (eg a ball bouncing on a paddle), and hence do not depend on the actual number of entities in the scene.

Environments like Atari require dealing with a varying amount of objects in each episode, since objects appear and disappear all the time. This called for an approach centered around sets, and I slowly converged to a transformer-based method. However, there were significant short-comings. Firstly, the method required a ground-truth annotation of the position of all the individual entities in each frame of the game, which is not readily available in Atari. We resorted to building manual sprite detectors of the game objects to tackle that, but it was realistically limiting the scope to a handful of games. Moreover, demonstrating convincing results on these games was also challenging, since the transformer approach was not entirely ironed out yet and the competing baselines were generally very strong.

For all these reasons, I felt that this work was not ready for publication. Coincidentally, I realized that the same underlying principles could be applied in a different way to detect the objects instead of predicting their movements. I thus changed my focus and successfully applied a very similar model to a purely supervised object detection task, reaching state-of-the-art performance.

In fine, because of this missing link, my thesis is composed of two seemingly unrelated works. To avoid creating far-fetched, artificial links between the two topics, I chose to embrace this diversity, and separate my manuscript into two distinct halves.

Chapter 1

Introduction

Major breakthroughs in reinforcement learning have allowed us to reach super-human performance in a variety of tasks that were long thought to be out of reach for machine-learning based systems. Perhaps the most striking example is that of Go (Silver, Huang, et al., 2016; Silver, Hubert, et al., 2018), where the algorithm, provided with only the rules of the game, attained super-human performance in early 2016. At the core, the method relies on simulating millions and millions of games against itself, and then using this experience to improve the policy, in an iterative fashion. Because the game is deterministic and fully observable, it is possible to enumerate a large number of possible actions and anticipate the response of the opponent. Most real-life settings are not as favorable, hence methods were proposed to tackle stochastic and information-imperfect games such as Dota (Berner et al., 2019) and Starcraft (Vinyals, Babuschkin, et al., 2019). These approaches also rely on simulating countless hours of self-play to generate training experience for the networks. However, the computational costs required to train agents to play such games are increasing prohibitively, suggesting that new paradigms must be devised. Moreover, these approaches rely on having access to a fast, accurate simulator of the environment, which in this case is the game engine itself. Extending these methods to the physical world thus remains a major challenge, since perfect simulation is impractical and using physical robots to generate experience would take far too long.

Another characteristic of these methods is that they typically can deal only with the task they were trained for, a phenomenon sometimes called "narrow AI". If the game rules are modified, then the networks need to be retrained from scratch, with all the computational overhead it entails. To alleviate this issue, some works have been focusing on creating models that can generalize to situations that have not been seen at training time. In Chapter 2, we address similar issues in the context of collaborative multi-agent problems. Specifically, we seek out to find policies that generalize to situations with more agents to control, or more tasks to tackle. Finding policies that generalize to account for more agents is a crucial endeavor for multi-agent reinforcement learning, and many real-life scenarios involve collaborating with varying number of agents, possibly unknown or even human. In our approach, we leverage the generalization property of our method to train agents in small environments, where the learning complexity is manageable, and then scale the fine-grained collaboration patterns that were learnt to bigger environments than was previously possible. Our solution relies on leveraging the structure of these multi-agent scenarios, by combining an optimization procedure with a neural network that determines the weights of the objective function. We study how the design decisions for the neural network and the optimization procedure impact the generalization capabilities of such a formulation, and show that it outperforms pure neural-network based approaches. In Section 1.1, we present the collaborative multi-agent framework, and some of the approaches that have been proposed.

Apart from control, one of the other major domains of interest in machine learning is perception. Tremendous progress has been made in building models that can interpret visual and audio signals, sometimes even surpassing human experts in specialized domains such as medical imaging (Rajpurkar et al., 2017). Detecting and localizing objects in an image is one of the challenging tasks that the community has made significant progress on over the last decade. It is a corner stone of computer vision, that has many applications, from self-driving cars to medical imaging, and in Section 1.2, we present the main paradigms used to tackle it. Most of the recent approaches rely heavily on hand-designed components, such as candidate boxes (anchors) to help the prediction of the box localization. Such priors require careful tuning, demanding a particular expertise and computational resources, thus potentially hampering performance on specific data distributions, as may be found in medical images for example. In a true deep learning spirit, we set out to remove these hand-designed components and train a model completely end-to-end. In Chapter 3, we present a streamlined approach to detection, which, unlike mainstream methods, places objects at the center of its focus. We remove the need for any detection-specific components, and rely exclusively on architectures that have been widely adopted by the deep learning community at large, namely residual networks (He, X. Zhang, et al., 2016) and Transformers (Vaswani et al., 2017). The attention mechanisms that are at the core of the Transformers allow the model not only to reason globally about the image, which notably improves detection on large objects, but also to reason over the predicted objects as a set, thereby avoiding the prediction of duplicates, which typically plague mainstream detectors. Such formulation of the problem as a set prediction not only allows us to reach competitive performance, but also holds great promise in enabling the development of models that can reason about the objects, their relationships, attributes and interactions.

Both these contributions rely on making use of specific structures of the problem in ways that were not explored before. In both cases, the structures we rely on involve finding a solution to a matching problem. Since matchings form a central building block in both the following chapters, we review in Section 1.3 the underlying mathematics, as well as several ways to optimize the variants of the problem we are focusing on in this manuscript.

1.1 Collaborative Multi-agent RL

The drive to control agents or systems to perform a given task as efficiently as possible dates back to the very early days of the development of computer science. In the 1950s, pioneers like Richard Bellman started studying the problem of optimal control, and introduced the main theoretical foundation, the Markov Decision Process, or MDP (Bellman, 1957). The simplest setting in optimal control is concerned with one agent, alone in the environment. Example includes a robot trying to navigate out of a maze, or an airplane auto-pilot trying land a plane as smoothly as possible. Some challenges arise when the number of agents increases. Historically, two-player games have been extensively studied, for example by explicitly modeling possible moves of the opponent and finding the best response (Silver, Huang, et al., 2016; Edwards and Hart, 1961), or, when the game becomes too complex, simply treating the opponent as part of the environment (Vinyals, Babuschkin, et al., 2019). However, many real-world problems involve more than one or two agents. These problems can be categorized in various ways. Firstly, they may differ in the way the agents are interacting with each-other: the task can be either collaborative (the agents are maximizing a common,

shared reward), competitive (each agent has its own reward function), or mixed (a combination of the two previous modes). Secondly, the problems vary in the extent to which agents are able to communicate. This forms a continuum from totally isolated, independent agents, to agents that are able to communicate in some limited ways, all the way to fully shared information and decision-making between the agents. The problems we are interested in, in this manuscript, are collaborative and centralized, meaning that the agents can fully share information amongst themselves, and try to optimize a joint reward. In the section 1.1.1, we formalize these notions precisely, and in section 1.1.2 we review some of the main approaches to tackle them.

1.1.1 Formal definition

Fully Observable Centralized Model

In the fully observable case, it is relatively straightforward to define a centralized model. The first formal definition comes from Boutilier, 1996, and is a simple extension of the MDP model.

Definition 1.1.1 (MMDP)**.** A multi-agent Markov decision process (MMDP) is defined as a tuple
$\mathcal{M} = \langle \mathbb{D}, \mathbb{S}, \{\mathbb{A}_i\}_{i\in\mathbb{D}}, T, R\rangle$ where

- $\mathbb{D} = \{1, \cdots, n\}$ is the set of n agents.
- $\mathbb{S}$ is a (finite) set of states.
- $\mathbb{A}_i$ is the set of actions available to agent i. This set may be state dependent. We denote the *joint* action as $\mathbb{A} = \times_{i\in\mathbb{D}}\mathbb{A}_i$.
- T is the transition probability function: $T : \mathbb{S} \times \mathbb{A} \to \Delta(\mathbb{S})$ where $\Delta(\mathbb{S})$ is the probability simplex over the next state.
- R is the reward function: $R : \mathbb{S} \times \mathbb{A} \to \mathbb{R}$
- $b_0 \in \Delta(\mathbb{S})$, is the initial state distribution at time $t = 0$.

In this setting, the reward is joint, and observed by all agents. Note that all agents have access to the full state, but not necessarily to the actions of others. However, it is reasonable to assume that they do, since in most case they can compute them themselves.

We note that extending this definition to the partially-observable case is not straightforward, because we need to specify how the agents are allowed to share information, if at all.

Multi-agent decision problems

In the most general formulation of a Multi-agent decision problem, we define what is a multi-agent environment, and what it means for an agent to be part of it, i.e what it can observe from the environment, other agents, and how it can act upon them.

Following Oliehoek and Amato, 2016, we define a Markov Multi-agent Environment (MME):

Definition 1.1.2 (MME)**.** The Markov multiagent environment (MME) is defined as a tuple
$\mathcal{M} = \langle \mathbb{D}, \mathbb{S}, \{\mathbb{A}_i\}_{i\in\mathbb{D}}, T, \{\mathbb{O}_i\}_{i\in\mathbb{D}}, O, \{R_i\}_{i\in\mathbb{P}}, b_0\rangle$, where

- $\mathbb{D} = \{1, \cdots, n\}$ is the set of n agents.
- $\mathbb{S}$ is a (finite) set of states.
- $\mathbb{A}_i$ is the set of actions available to agent i. This set may be state dependent. We denote the *joint* action as $\mathbb{A} = \times_{i \in \mathbb{D}}$.
- T is the transition probability function: $T : \mathbb{S} \times \mathbb{A} \to \Delta(\mathbb{S})$ where $\Delta(\mathbb{S})$ is the probability simplex over the next state.
- $\mathbb{O}_i$ is the set of observations available to agent i.
- O_i is the observation probability function: $O_i : \mathbb{S} \times \mathbb{A} \to \Delta(\mathbb{O}_i)$
- R_i is the immediate reward function for agent i: $R_i : \mathbb{S} \times \mathbb{A}_i \to \mathbb{R}$
- $b_0 \in \Delta(\mathbb{S})$, is the initial state distribution at time $t = 0$.

This definition makes the dynamic of the environment explicit, but leaves the dynamics of the agents themselves unspecified.

To remedy this, Oliehoek and Amato, 2016 defines an agent component:

Definition 1.1.3 (Agent component)**.** A fully-specified agent component, can be formalized as a tuple
$m = \langle \mathbb{D}, \{\mathbb{I}_i\}_{i \in \mathbb{D}}, \{I_{i,0}\}, \{\mathbb{A}_i\}_{i \in \mathbb{D}}, \{\mathbb{O}_i\}_{i \in \mathbb{D}}, \{\mathbb{Z}_i\}_{i \in \mathbb{D}}, i, \{\pi\} \rangle$, where

- $\mathbb{D} = \{1, \cdots, n\}$ is the set of n agents.
- $\mathbb{I}_i$ is the internal state of agent i. Sometimes called *belief*.
- $I_{i,0}$ is the initial internal state of agent i.
- $\mathbb{A}_i$ is the set of actions that the agent i can execute. May be state dependent.
- $\mathbb{O}_i$ is the set of observations available to agent i.
- $\mathbb{Z}_i$ is the set of auxiliary observations (e.g. obtained by communication) available to agent i.
- π_i is the (stochastic) policy of agent i: $\pi_i : \mathbb{I}_i \to \Delta(S)$.
- l_i is the (stochastic) information state function of agent i: $l_i : \mathbb{I}_i \times \mathbb{A}_i \times \mathbb{O}_i \times \mathbb{Z}_i \to \mathbb{I}_i$

We now have all the required pieces to define some partially observable multi-agent models.

Partially Observable Centralized Model This is the most natural extension of MMDP to a partially observable case. Specifically, agents share their beliefs and observations of the state, but can't observe the full-state.

Definition 1.1.4 (MPOMDP)**.** A multi-agent POMDP (MPOMDP) is specified by a MME $\mathcal{M}$ and a partially specified agent model m where:

- $m = \langle \mathbb{D}, \{\mathbb{I}_i\}, \{I_{i,0}\}, \{\mathbb{A}_i\}, \{\mathbb{O}_i\}, \{\mathbb{Z}_i\}, \{l_i\}, \cdot \rangle$
- All the internal states sets are the same, and all agents start with the same belief, and the belief-update functions are the same: for all $i, j \in \mathbb{D}^2$, $I_{i,0} = I_{j,0}$, $l_i = l_j$ and $\mathbb{I}_i = \mathbb{I}_j$

- The auxiliary observations are the concatenation of the observations of other agents are obtained by instantaneous communication: $\mathbb{Z}_i = \bigotimes_{j \neq i} \mathrm{O}_j$

Note that this definition is independent of how decisions are actually going to be taken in a physical system: it could be that a central system decides the action of all the agents, which in turn merely execute them, or that each agent is responsible for taking its own action (this setting is sometimes referred to as "decentralized execution" in the literature). However, from a purely theoretical point of view, this difference does not make much of a difference: given that the agents have the same beliefs and observations, they can recompute the actions of their peers at will (if needed, the weights of the networks can also be shared across the perfect communication channel, or as part of the belief).

Decentralized models Even though we consider exclusively centralized settings in this manuscript, we briefly review the decentralized counterparts to highlight the similarities and differences.

The most extreme case of decentralization occurs when agents can't communicate at all, i.e. when for all agent i, $\mathbb{Z}_i = \varnothing$. This is known as the Dec-POMDP model (Oliehoek, Spaan, and Vlassis, 2008). By contrast, some recent works assume agents don't share their observations, but allow some form of communication between them, for example through graph-based models (Foerster, Assael, et al., 2016; Foerster, Farquhar, et al., 2018; Jiang, Amo, and Lu, 2018; Jiang and Lu, 2018; Das et al., 2019; Witt et al., 2019). However, the communication protocol is most of the time under-specified, and often times seems more catered to the technical solution than any actual restriction from the problem itself. In fully cooperative games, there is no incentive for the agents to withhold any of their information and observation, hence as long as communication is possible, it is unclear why they would refrain from doing so. Even considering real-life scenarios with fleet of robots communicating over WiFi or Bluetooth using appropriate lossless compression algorithms, it seems likely that they would be able to share enough information to form a common belief. Moreover, graph-based methods also often implicitly assume several rounds of communication. Under such assumption, it is possible to devise an execution scheme where the agents agree upon a leader that is responsible for computing the actions of all the others, then broadcast them such that they can simply execute them (such scheme is also possible with only one communication round if the leader is pre-agreed upon and can consume the messages of all other agents before broadcasting its response). All in all, these considerations blur the line between such communicating decentralized approaches in collaborative environments and the fully centralized model presented in the previous paragraph. However, the distinction becomes fully apparent in mixed collaboration-cooperation settings, where agents do not share the same reward function and thus must learn to communicate only the part of their knowledge that aligns with their self-interested goals.

1.1.2 Approaches to collaborative multi-agent problems

Multi-agent collaborative problems have been studied in many different ways by different research communities over time. For example, many specific problems have been studied through the lens of Operational Research. Amongst others, we can mention the famous multi-vehicle routing problem (Toth and Vigo, 2002), or various instances of Unmanned Aerial Vehicle fleet management problems (Shetty, Sudit, and Nagi, 2008; Richards et al., 2002). In this manuscript however, we are

interested in model-free reinforcement learning approaches. In this context, small scale problems can be tackled by simply considering the joint action-space and applying standard reinforcement learning algorithms. However, as the number of agents increases, the set of actions grows exponentially, very quickly becoming intractable. One approach to side-step this issue is to exploit some structure in the underlying dynamics (Meuleau et al., 1998; Tesauro, 2005; Proper and Tadepalli, 2009): for example, one may assume that the dynamics of the agents are mostly independent, or only dependent on a handful of other agents. Another approach gaining a lot of attention is to treat all the agents mostly independently, but allow them to communicate through some graph-based models (Sukhbaatar, Szlam, and Fergus, 2016; Foerster, Assael, et al., 2016; Zambaldi et al., 2019; Yang et al., 2018; Jiang, Amo, and Lu, 2018; Jiang and Lu, 2018; Lowe et al., 2017; A. Singh, Jain, and Sukhbaatar, 2019; Das et al., 2019; Witt et al., 2019). These methods essentially break down the complexity of the problem by considering only the behavior of a given agent at a time, then supplementing it with some learnt communication scheme with the other agents, which is typically designed to enforce a limited amount of shared information in order to keep the complexity manageable.

One of the main challenge in multi-agent problems lies in the adaptability and the scalability of the proposed approaches. Adaptability requires the proposed methods to be robust to changes in the number of agents, and is desirable for real-world applications (Barrett et al., 2017): in most scenarios, training is expensive, hence it is undesirable to have to retrain the whole policy if, for example, one of the robots breaks or some new ones are added to the fleet. Scalability is also a challenge, as it is often hard to scale up current multi-agent methods beyond a dozen units. We note that good adaptability can potentially help scalability: if a method is able to learn a policy in situations with few units, where the learning complexity is manageable, and then generalize it to bigger scenarios without retraining, then the method can naturally be scaled up. This is the approach we pursue in this manuscript. Amongst the notable approaches to large scale MAARL, we can note the Mean-Field Approach (Yang et al., 2018) which consists in approximating the effect of the other agents (possibly restricted to a neighborhood) by considering the mean of the actions taken by them. This is a relatively crude approximation which prevents fine-grained collaboration, but allows to handle scenarios with up to a thousand agents comfortably, and even varying number of agents (since the mean is always well defined). Less stringent approximations are possible, for example by explicitly constraining the set of neighbors an agent can communicate with (Jiang and Lu, 2018; Jiang, Amo, and Lu, 2018). This allows some small scale generalization in simple environments. However, even with additional inductive biases, the generalization results reported on environments as complex as StarCraft are mostly negative (Zambaldi et al., 2019; Usunier et al., 2016), suggesting that additional mechanisms are required to achieve collaboration.

Contributions

In this manuscript, we focus on problems that can be hierarchically decomposed in a set of loosely coupled tasks, similarly to previous approaches (Proper and Tadepalli, 2009). Example includes battle scenarios, where each enemy unit can be seen as a task, or collaborative navigation, where a task is a way-point to be visited. Such problems can be approached by first assigning a task to each agent, then letting each agent carry out its task, possibly interacting with other agents assigned to the same one. Contrary to some graph-based approaches (Jiang and Lu, 2018; Jiang, Amo, and Lu, 2018) that tend to limit interactions based on geographic proximity, we group agents

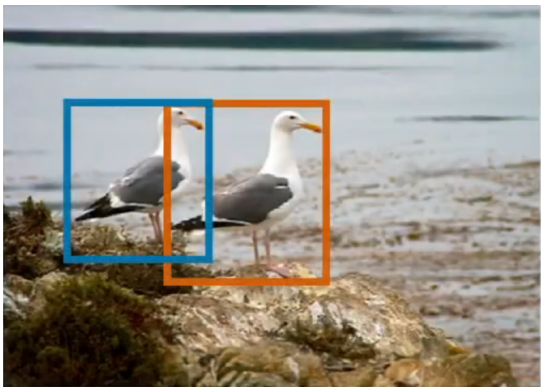


FIGURE 1.1: Object detection consists in finding tight bounding boxes around visible objects.

based on the task they are carrying out, which allows more sensible coordination. By contrast, the assignment step must coordinate all the units at once. To efficiently search amongst the exponential number of possible assignments, we assume some structure: for example, we may assume that each agent is able to contribute a certain amount (positive or negative) to each possible task, and the contributions of all agents assigned to a given task are additive. This results in a linear assignment problem (see Section 1.3.2) that can be efficiently solved. We use a neural network trained using reinforcement learning to predict the expected contributions of each agent to each task. We finally show that the design decisions for the collaboration optimization problem as well as the expressivity of the neural network both impact the generalization potential of the policy found, and with the appropriate choices, we demonstrate effective generalization to up to 5 times more units than seen during training. This contribution has been accepted as a spotlight presentation at Neurips 2019 (Carion, Usunier, et al., 2019).

1.2 Object detection

Object detection has historically been a core computer vision task over the last decades of research (Zou et al., 2019; L. Liu et al., 2020). Some of the earliest attempts can be traced back to the 70s (Fischler and Elschlager, 1973), where a dynamic programming algorithm was used to detect faces from a predefined template. The task is closely related to object recognition, which involves classifying the most salient object of the image. By contrast, in object detection, one must not only classify the object but also regress its localization and extent, for example through the prediction of a tight bounding box (Everingham et al., 2010) around the instance, as depicted in Fig.1.1. A given image may contain an arbitrary number of instances, that must all be detected individually. More formally, for an image $\mathcal{I}$ of dimension $H \times W$ we define

$$\mathbb{B}(\mathcal{I}) = \Big\{(x,w,y,h) \in [0,W]^2 \times [0,H]^2 \mid 0 \leq x - \frac{w}{2} < x + \frac{w}{2} \leq W \\ \text{and } 0 \leq y - \frac{h}{2} < y + \frac{h}{2} \leq H\Big\}$$

as the set of all possible bounding boxes contained in the image, characterized by their center (x,y), their height h and width w. We assume a given finite discrete set of category labels $\mathcal{L}$ that correspond to the classes of objects we are interested

in detecting. The task of object detection can then be defined as producing a set $\mathcal{B}(\mathcal{I}) = \{(b_1, l_1), \cdots, (b_n, l_n)\} \in \mathbb{B} \times \mathcal{L}$ of bounding boxes and their associated category label.

1.2.1 Object detection as a classification problem

The most common way to address the object detection task is to treat it as a classification problem. The set of category labels is first extended as $\overline{\mathcal{L}} = \mathcal{L} \cup \{\varnothing\}$, where $\varnothing$ is the label representing a box that does not canonically contain an object, sometimes referred to as the "background class". Assuming we have access to a classifier $\hat{c} : \mathbb{B}(\mathcal{I}) \to \mathcal{L}$, we can construct the predicted set of detection as:

$$\widehat{\mathcal{B}(\mathcal{I})} = \{(b, \hat{c}(b)) \mid b \in \mathbb{B}(\mathcal{I}), \hat{c}(b) \neq \varnothing\}$$

Since $\mathbb{B}$ is infinite, in practice it needs to be quantized to a finite subset that we denote as $\tilde{\mathbb{B}}$. This has two main consequences:

- It introduces a quantization error. To cope with this, some methods additionally train a regressor to predict the quantization error, to try to compensate for it.
- It makes the problem ill-posed. In general, the true boxes are unlikely to be contained in $\tilde{\mathbb{B}}$, which means that in theory the classifier should label almost all the boxes in the quantized set as $\varnothing$.

To make the problem solvable nonetheless, a common solution is to relax the classification rule, by defining a heuristic label assignment rule which will assign a foreground label to quantized boxes even when they are not exactly equal to a ground-truth box. One possible way of doing so is to assign to a quantized box the label of the ground-truth box that has the highest Intersection over Union (IoU), provided that it is above a predefined threshold, and then train the classifier on this proxy classification task. In general, since the matching is non-unique, there could be several quantized boxes that are highly overlapping with the same ground-truth box and thus receive the same label corresponding to the same instance. Training a classification model on these artificial labels will lead the model to produce redundant predictions. As a result, methods based on this principle often need an additional de-duplication set operation.

We now review some of the historical breakthroughs in object detection using a classification formulation.

First real-time face detector A notable milestone is the work of Paul Viola and Micheal Jones (Viola and Jones, 2001) who obtained the first real-time object detection algorithm. Though their method is in theory generic, it was particularly well suited to detect faces, notably because of the low variability between instances. Their approach relied on Adaboost (Freund and Schapire, 1995) to select features from a set of simple and generic filters (Haar filters), then used a sliding-window approach to apply them all over the image. They also relied on cascading the filters to speed-up the computation and ensure real-time detection.

Deformable Part Models The next notable milestone is the work of Navneet Dalal and Bill Triggs (Dalal and Triggs, 2005). Their method was following the sliding window technique, but their feature descriptors consisted in Histogram of Gradients (HOG), which consist in counting the local gradient orientation in a given patch of

the image. The classification part had also been improved, following the progress in pattern recognition, and they elected to use linear Support Vector Machines (Boser, Guyon, and Vapnik, 1992). This approach was successfully applied to pedestrian detection, even though the idea is in theory not class-specific. Some follow up works (Ross Brook Girshick, 2012; Felzenszwalb, McAllester, and Ramanan, 2008) improved on this approach by focusing on individual parts of the object instead of trying to detect it as a whole: these models were called Deformable Part-based models.

Two-stage detectors In 2012, a major breakthrough occurred in computer vision: using a deep convolutional network, Alex Krizhevsky et al. obtained significant performance improvement on a challenging image recognition dataset (Krizhevsky, Sutskever, and Hinton, 2012). From there on, these neural networks replaced traditional feature descriptors. One of the first attempt at incorporating convolutional networks in a detection pipeline is RCNN (Ross B. Girshick et al., 2014). This approach is, in spirit, quite similar to the previous methods, with some improvements to make it more efficient:

- Instead of naively classifying all possible regions, it relies on a heuristic to select a few thousands of candidate boxes
- Instead of using HOG-like descriptors, it uses a convolutional network followed by a linear SVM to classify each proposal, as well as a regressor to correct the exact localization of the box.

In subsequent work, the SVM was replaced by fully connected layers (Ross B. Girshick, 2015), then the heuristic used to generate candidate boxes was replaced by a dedicated network, called the Region Proposal Network (S. Ren et al., 2015a). This family of methods is called *two-stage* detectors: a first network proposes regions, and a second one classify and regress the precise bounding boxes.

It has to be noted that the Region Proposal Network (RPN) typically uses a set of manually designed anchor boxes. They serve as a seed to the regression, since the network's predictions are made relatively to the anchor, and they enforce a diversity in the scales and aspect ratios of the boxes considered. However, they introduce additional hyper-parameters, which may require careful tuning when applying the method to custom datasets with peculiar boxes distribution.

As noted previously, these methods suffer from duplicate detections, due to the relaxed classification formulation. The most prevalent solution to this problem is to apply a de-duplication step called the Non-Maximal Suppression (NMS). This procedure greedily considers pair of boxes which predicts the same object category and have a high overlap, and discards the one with the lowest confidence, then iterate this process until reaching a fixed point. This building block (or its variants such as Bodla et al., 2017) has been ubiquitous in two-stage detection pipelines to this day. Its sequential nature makes it difficult to implement efficiently on modern computation accelerators (GPUs, TPUs), although some more parallel versions have been proposed, such as MatrixNMS (Xinlong Wang et al., 2020). However, the presence of such a building block poses several issues:

- Since it is greedily based on a fixed overlap threshold, it forces a precision-recall trade-off. Getting rid of duplicated boxes might imply removing true-positive boxes corresponding to highly overlapping objects.
- It is susceptible to adversarial attacks (Derui Wang et al., 2019)

- It is non-differentiable, making it impossible to use detected objects in downstream tasks while training end-to-end.

To this day two-stage methods are still considered the gold-standard for detection, and winning entries in the object detection challenges, such as COCO (T.-Y. Lin, Maire, et al., 2014) are based are almost always built on such principles.

One-stage detectors The two detection stages (box proposal, then refinement) yield accurate detectors, but often at the expense of speed. A parallel line of works explore the possibility of detecting in only one stage, trading a bit of localization accuracy for speed, and thus yielding detectors that can comfortably operate in real-time.

One the very first one stage detection model was OverFeat (Sermanet et al., 2014), presented in 2013. It uses a freezed feature extractor trained on Imagenet Classification, as well as the dense classification head, and trains a class-specific box regression head that densely predicts box coordinates at each location and at several scales. The predictions are then aggregated across space and scales in a greedy manner.

In 2016, Redmon et al. proposed YOLO (Redmon, Divvala, et al., 2016). The approach consists in tiling the image in a 7×7 grid, then for each cell of this coarse grid predicts a classification label, and 2 boxes, characterized by their position relative to the cell, their size, and a probability that this box correspond to a real object (used for confidence score). Since the model considers only 98 possible boxes, it is able to run very efficiently. However, it struggles to detect small objects, and one limitation is that a single coarse cell is assumed to contain only boxes of a single class. Contrary to OverFeat, the whole network is trained jointly. This first version of YOLO is a bit unique amongst other detection methods in that its label propagation rule, by construct, will enforce that the label of each ground-truth box will be applied to exactly one proposed box. Indeed, the algorithm looks in which cell the center of the ground box is falling, then picks the predicted box from that cell that has the highest overlap with the ground-truth box. In that sense, it creates an implicit greedy matching between ground-truth and predicted boxes. This makes the classification problem harder for the network: not only it has to classify boxes with good overlap with the ground-truth, but also has to implicitly rank them, so that the best overlapping box gets the true label and the other are classified as background. This approach turned out to be impractical as the number of candidate boxes increases, because it increases the number of "good" candidates and thus makes it harder for the classifier to pick the best one. It was thus dropped in subsequent work (Redmon and Farhadi, 2017) as well as parallel works such as SSD (W. Liu, Anguelov, Erhan, Szegedy, S. Reed, et al., 2016), where the number of candidate boxes was drastically increased, using proper anchor boxes (box regression was made with respect to these anchors instead of the grid cell). In these methods, the ground-truth label is propagated to any candidate box whose IoU with the ground-truth is above some fixed threshold (typically 0.5).

Increasing the number of candidate boxes is necessary to improve recall, but it comes with a drawback: it amplifies the imbalance between foreground boxes (the ones that contain an object) and background boxes (those which are empty). To alleviate this issue, some method such as RetinaNet (T.-Y. Lin, Goyal, et al., 2017) adjust the cross-entropy loss to down-weight the contribution of well classified boxes, forcing the network to focus on hard, ambiguous examples.

Anchor-free methods Most of the state-of-the-art detectors presented in the previous paragraphs rely on some pre-defined anchors to make predictions. These anchors

encode prior knowledge about the task, and may require careful tuning if the distribution of box shapes is significantly different than the typical detection dataset (COCO or Pascal VOC for example), as it may be the case for specialized domain like medical imagery. Some methods have thus sought to avoid relying on such building blocks.

The first line of work consists in treating some points on the boundary of the object or its bounding box as keypoints and detecting them as such. CornerNet (Law and Deng, 2018) seeks to detect the top-left and bottom-right corners of each object. To do so, it predicts two high-resolution heatmaps: one for top-left corners and one for bottom-right corners. Then it extracts the peaks of both heatmaps (using a 3×3 max pooling), as well as the feature vector associated with its location. To match corners by pairs, it then proceed to compute the distance between each pair of embeddings, and match the most similar ones. Some improvements to this approach where then proposed, to improve the matching between corners (Dong et al., 2020) or detect more natural keypoints that lie on the object boundary (Zhou, Zhuo, and Krähenbühl, 2019).

Another line of work consists in detecting objects from their center (Duan et al., 2019; Z. Tian et al., 2019; Zhou, Dequan Wang, and Krähenbühl, 2019). In FCOS (Z. Tian et al., 2019), each cell of all the feature-maps (at different resolution) are treated as positive if they lie within the ground-truth of a given object. Then, depending on the size of the object, a feature map at a particular resolution is heuristically selected, and only cells within this feature map are eventually treated as positive. For each of them, the network predicts the class of the object, as well as the distance from the center of the cell to the four sides of the bounding box. The remaining negative cells are simply trained to predict the background class. This results in many boxes predicted for each instance, hence NMS is applied as usual. It has been shown (S. Zhang et al., 2020) that using comparable implementation details, the difference between FCOS and an anchor-based counter-part such as RetinaNet is negligible. CenterNet (Duan et al., 2019; Zhou, Dequan Wang, and Krähenbühl, 2019) takes a slightly different approach in that it considers as positive only the cell that contains the true center of the bounding box, and thus only needs to regress the height and width of the box. Similar to CornerNet, it thus produces a heatmap of all centers, and it extracts the peaks (cells whose response value is greater than their 8 connected neighbors), which in this case replace the non-maximal suppression step.

Overall, these anchor-free methods often require much bigger backbones to compete with the anchor-based counterparts, while still relying on non-differentiable set de-duplication such as (soft)NMS or peak extraction in the center heatmap.

1.2.2 Beyond classification based approaches

At a conceptual level, the issue behind classification-based approaches is that they don't optimize for the true objective, which is to predict a set. As a result, telling how many instances of a given class are present in the image can prove challenging: indeed the NMS acts as a way to control the precision recall tradeoff but can't guarantee that the true number of objects are going to be predicted. Recently some approaches try to learn NMS with a convolutional network (Hosang, Benenson, and Schiele, 2017), which is a step in the right direction but doesn't yield state of the art performance.

A parallel line of work in object detection (Stewart, Andriluka, and Ng, 2016) and instance segmentation (Romera-Paredes and Torr, 2015; Park and Berg, 2015; M. Ren and Zemel, 2017; Salvador et al., 2017) fully embraces the set nature of the problem and uses recurrent models to predict autoregressively all the objects that are present

in the image. The main limitation of these approaches is that the sequential prediction of the objects makes the inference time proportional to the number of instances which is not compatible with real time applications. Moreover they suffer from low recall leading to poor scores on challenging detection datasets such as COCO.

Contributions

In this manuscript, we show that a set based approach is possible using modern architectures such as transformers. This formulation side-steps many problems introduced by the classical classification approach, particularly the class imbalance for the background classes, and the duplicate detections. This allows us to construct a streamlined detection pipeline that is fully differentiable and don't require any detection-specific components. Crucially, we perform the detection in parallel which not only alleviates the speed limitations encountered by the previous recurrent approaches, but also ensures a competitive recall score. We show that using a comparable parameter budget and similar inference times as compared to well-established two stage methods, we also match the average precision of these baselines. Furthermore, we show that the architecture can be naturally extended to perform panoptic segmentation, obtaining state-of-the-art performances, which demonstrate the generality of the approach. This contribution has been accepted as an oral presentation at ECCV 2020 (Carion, Massa, et al., 2020).

1.3 Matching

Matchings are an important concept in computer science, and have been studied extensively through various lenses, including graph theory and operational research. We start by recalling the mathematical definition and fixing some vocabulary in Section 1.3.1. Matchings form an essential building block for both of the following chapters: in the multi-agent chapter (Chapter 2), they constitute the foundation of the policy executed by our agents, and thus we study the specific tools required in Section 1.3.3. In the object detection chapter (Chapter 3), they appear in the computation of our structured loss. The corresponding tools are exposed in Section 1.3.2.

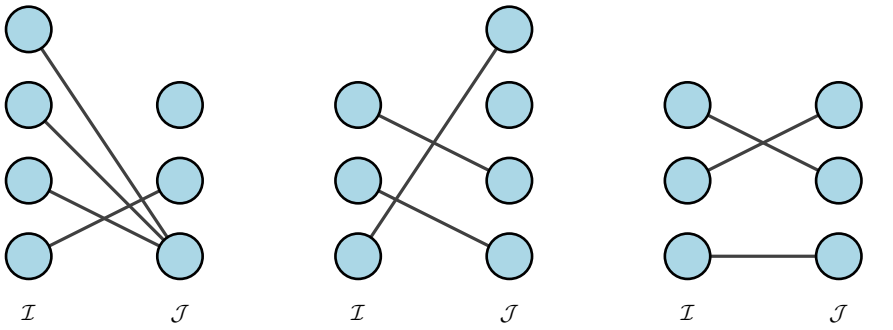
1.3.1 Mathematical definition

In this section, we will consider two discrete, finite sets $\mathcal{I}$ and $\mathcal{J}$, and w.l.o.g. we can assume they are subset of the set of positive integers $\mathbb{N}$. We denote $n = \|\mathcal{I}\|$ and $m = \|\mathcal{J}\|$, and unless stated explicitly we do not assume that $n = m$. In Chapter 2, $\mathcal{I}$ will correspond to the set of agents, and $\mathcal{J}$ to the set of tasks that are to be solved. In Chapter 3, both sets will correspond, respectively, to the set of predicted objects and the set of ground truth objects.

Definition 1.3.1. A *matching* between two discrete finite sets $\mathcal{I}$ and $\mathcal{J}$ is simply a function $\beta : \mathcal{I} \to \mathcal{J}$. We call it a *strict matching* if β is injective. If $n = m$, a strict matching is actually a bijection, and we call it a *bipartite* matching. See illustration in Fig. 1.2

With a slight abuse of notation, we may refer to a matching β by the corresponding assignment matrix $(\beta)_{i,j}$, defined as:

$$\forall i, j \in \mathcal{I} \times \mathcal{J}, \beta_{i,j} = \begin{cases} 1 & \text{if } \beta(i) = j \\ 0 & \text{otherwise} \end{cases}$$



(A) A general matching (B) A *strict* matching (C) A *bipartite* matching

FIGURE 1.2: Illustration of various matching types

Under this notation, $(\beta)_{i,j}$ is thus a binary right-stochastic matrix. There are m^n possible matchings, and $\frac{m!}{(m-n)!}$ strict matchings (assuming $m \geq n$, otherwise no strict matching exists). In practical applications, though, we are typically interested in a very limited subset of all possible matchings, the one that we deem "interesting", assuming we have a cost function that allows to compare various assignment. It then becomes an optimization problem, and the exact nature of the cost function will lead to different strategies for finding an exact or approximate solution.

In Section 1.3.2, we consider the problem of finding bipartite matching under a linear cost function. In Section 1.3.3, we consider quadratic cost functions, and review how it is possible to impose extra constraints to the matching.

1.3.2 Linear sum assignment

In this section, we consider bipartite matchings, under a linear cost function. Concretely, for each $i \in \mathcal{I}$ and $j \in \mathcal{J}$, assigning i to j is associated with a cost $c_{i,j}$, and we seek to find the matching that maximizes the total cost. In other word, we wish to solve the following integer linear program (ILP):

$$\begin{aligned} \min \ & \sum_{i,j} \beta_{i,j} c_{i,j} \\ \text{s.t. } & \forall j \in \mathcal{J}, \ \sum_i \beta_{i,j} = 1 \\ & \forall i \in \mathcal{I}, \ \sum_j \beta_{i,j} = 1 \\ & \forall i,j \in \mathcal{I} \times \mathcal{J}, \ \beta_{i,j} \in \{0,1\} \end{aligned}$$

In this ILP, the constraints enforce that the assignment matrix is binary bi-stochastic. For this program to be solvable, we require that $\|\mathcal{I}\| = \|\mathcal{J}\|$. In practice though, it is possible to deal with situations where both sets do not have equal cardinality, and in this case, we are looking for a strict matching instead of a bipartite one. The most straightforward way of doing so is to create some dummy padding elements as follows. W.l.o.g, we assume that $n = \|\mathcal{I}\| < \|\mathcal{J}\| = m$. We then define $\overline{\mathcal{I}} = \mathcal{I} \cup \{d_{n+1}, \cdots, d_m\}$ where d_k are some dummy elements that take arbitrary values (distinct from the values in $\mathcal{I}$ and $\mathcal{J}$). We then define the new cost function $\overline{c_{i,j}}$ as

follows:

$$\overline{c_{i,j}} = \begin{cases} c_{i,j} & \text{if } i \in \mathcal{I} \\ C & \text{otherwise} \end{cases}$$

where C is a very large real number ($C >> \max c_{i,j}$). This is equivalent to lifting one of the sum constraints in the ILP, and amounts to finding a match for the elements of the bigger set that can be best matched, and leaving the rest un-matched.

One of the earliest algorithm devised to find best assignment in this setting is the Kuhn-Munkres algorithm (Kuhn, 1955), also known as the "Hungarian" algorithm, which has a $O(n^4)$ complexity. The complexity was subsequently improved by slightly tweaking the algorithm, leading to the Jonker-Volgenant algorithm (Jonker and Volgenant, 1987), which brings the complexity down to $O(n^3)$, and is one of the most popular algorithm used to date.

1.3.3 Constrained and quadratic cost matchings

In this section, we lift the requirements of the previous section and focus on general assignments, which we characterize by the following ILP:

$$\begin{aligned} &\min C(\beta) \\ &\text{s.t. } \forall i \in \mathcal{I},\ \sum_j \beta_{i,j} = 1 \\ &\qquad \forall i,j \in \mathcal{I} \times \mathcal{J},\ \beta_{i,j} \in \{0,1\} \end{aligned}$$

where $C(\beta)$ is the cost function that we minimize.

We now review some special cases that appear in the rest of the manuscript, and describe approaches to solve them.

Linear cost, no constraints If the cost function C is linear, as in Section 1.3.2, that is $C(\beta) = \sum_{i,j} \beta_{i,j} c_{i,j}$, and in the absence of additional constraints, the solution to the ILP is trivial to obtain: we can simply greedily assign each element of $\mathcal{I}$ to the element of $\mathcal{J}$ that incurs the minimal cost.

Linear cost, capacity constraints With a linear cost function, a more interesting scenario occurs when some constraints are added to the assignment. As a practical example, we may consider a scenario where computational workloads are to be assigned to servers in a computer cluster. Each workload i may require some amount of resources r_i, while each server j has a limited capacity c_j. We impose that the sum of the resources required by the workloads assigned to each server doesn't exceed its capacity.

Formally, that amounts to adding the following constraints to the ILP:

$$\forall j \in \mathcal{J},\ \sum_i r_i \beta_{i,j} \leq c_j$$

Note that if $r_i = c_j = 1$ for all i, j, then we recover the bipartite matching case. However, with arbitrary values for r_i and c_j, the problem becomes more complicated, and the Hungarian algorithm and its variants do not apply anymore.

One way to approach this problem is to directly try to optimize the linear program. In the special bipartite case ($r_i = c_j = 1$), the constraint matrix is totally unimodular,

and in this case it is known that the solution of the ILP is the same as the solution of the relaxed linear program, so one can simply solve the relaxed program using efficient methods such as the simplex algorithm. In the general case, this nice property may not hold, and it may be the case that the fractional solution is not integral. In this case, the standard approach is to apply some rounding to retrieve an integral solution to the matching. Many methods, such as randomized rounding (Raghavan and Tompson, 1987) have been devised to tackle this, depending on the task at hand.

Quadratic cost If the cost function is quadratic, we obtain a quadratic assignment problem (QAP), which is known to be amongst the hardest discrete optimization problems:

$$\begin{aligned} \min \sum_{i,j} \beta_{i,j} c_{i,j} + \sum_{i,j,k,l} \beta_{i,j} \beta_{k,l} d_{i,j,k,l} \\ \text{s.t. } \forall i \in \mathcal{I},\ \sum_{j} \beta_{i,j} = 1 \\ \forall i,j \in \mathcal{I} \times \mathcal{J},\ \beta_{i,j} \in \{0,1\} \end{aligned}$$

It is also possible to add the constraints introduced in the previous paragraph.

Due to its importance in the operational research community, it has been widely studied, and many solutions or heuristics have been proposed. The methods include exact branch and bound algorithms (Hahn and Grant, 1998), reduction to ILPs (Adams and Johnson, 1994), and various blackbox optimization techniques like tabu search (Battiti and Tecchiolli, 1994) or simulated annealing (Connolly, 1990).

In the context of this thesis, we are interested in small scale QAPs (mostly n and m around 10, rarely 100), and speed is crucial since we are required to take a decision within a few milliseconds. With this computational budget, most blackbox optimization methods tends to be too high variance for our needs, since in this case the final result will most likely depend on the quality of the initialization. To avoid such variance, we seek a different class of methods. First, similarly to the linear case, we consider the relaxed version of the program. We now have to optimize a continuous function over a constrained, convex set. This cost function is not convex in general, but nonetheless we apply convex optimization methods, knowing that we may get stuck into a local minima. Given that our constraints are linear, an attractive method is the Frank-Wolfe algorithm (Frank and Wolfe, 1956). Let us denote by $\mathcal{D}$ the convex domain implied by the linear constraints. This algorithm works by repeating the following steps until convergence:

1. We first find the current optimization direction. For that, we use the first order Taylor expansion of our cost function C around the current point x, which is $\hat{C}(y) = C(x) + \nabla C(x)(y - x)$. Minimizing this approximation with respect to y is equivalent to finding $s = \arg\min_{y \in \mathcal{D}} \nabla C(x) y$. This sub-problem is exactly a linear program, which we can solve efficiently using, for example, the simplex method.

2. Then we need to determine a step-size γ. An easy way to do that is use a decaying step-size at each iteration of the algorithm.

3. Finally, we update the current point by moving in the chosen direction for the chosen step-size: $x \mathrel{+}= \gamma(s - x)$

Several variants have been proposed to improve the convergence rate of the algorithm. We use the Pairwise Frank-Wolf variant (Lacoste-Julien and Jaggi, 2015) which considers a slightly different optimization direction based on the previous steps that were taken.

Once this algorithm converges, we obtain a real-valued solution to the quadratic program. Similarly as for the linear case, the last step is thus to apply some rounding technique to retrieve a binary assignment.

With an appropriate convergence tolerance and an efficient simplex solver, this method is finding a solution quickly enough to fit our computational requirements, and on average it outperforms other blackbox approaches, given the same computational budget.

Chapter 2

A Structured Prediction Approach for Generalization in Cooperative Multi-Agent Reinforcement Learning

Abstract

Effective coordination is crucial to solve multi-agent collaborative (MAC) problems. While centralized reinforcement learning methods can optimally solve small MAC instances, they do not scale to large problems and they fail to generalize to scenarios different from those seen during training. In this chapter, we consider MAC problems with some intrinsic notion of locality (e.g., geographic proximity) such that interactions between agents and tasks are locally limited. By leveraging this property, we introduce a novel structured prediction approach to assign agents to tasks. At each step, the assignment is obtained by solving a centralized optimization problem (the inference procedure) whose objective function is parameterized by a learned scoring model. We propose different combinations of inference procedures and scoring models able to represent coordination patterns of increasing complexity. The resulting assignment policy can be efficiently learned on small problem instances and readily reused in problems with more agents and tasks (i.e., zero-shot generalization). We report experimental results on a toy search and rescue problem and on several target selection scenarios in StarCraft: Brood War[1], in which our model significantly outperforms strong rule-based baselines on instances with 5 times more agents and tasks than those seen during training.

2.1 Introduction

Multi-agent collaboration (MAC) problems often decompose into several intermediate tasks that need to be completed to achieve a global goal. A common measure of size, or difficulty, of MAC problems is the number of agents and tasks: more tasks usually require longer-term planning, the joint action space grows exponentially with the number of agents, and the joint state space is exponential in both the numbers of tasks and agents. While general-purpose reinforcement learning (RL) methods Sutton and Barto, 2018 are theoretically able to solve (centralized) MAC problems, their learning (e.g., estimating the optimal action-value function) and computational (e.g., deriving the greedy policy from an action-value function) complexity grows exponentially

[1] StarCraft and its expansion StarCraft: Brood War are trademarks of Blizzard Entertainment™.

with the dimension of the problem. A way to address this limitation is to learn in problems with few agents and a small planning horizon and then *generalize* the solution to more complex instances. Unfortunately, standard RL methods are not able to perform any meaningful generalization to scenarios different from those seen during training. In this chapter we study problems whose structure can be exploited to learn policies in small instances that can be efficiently generalized across scenarios of different size.

Well-known MAC problems that are solved by a suitable sequence of agent-task assignments include search and rescue, predator-prey problems, fleet coordination, or managing units in video games. In all these problems, the dynamics describing the interaction of the "objects" in the environment (i.e., agents and tasks) is regulated by *constraints* that may greatly simplify the problem. A typical example is *local proximity*, where objects' actions may only affect nearby objects (e.g., in the predator-prey, the prey's movements only depend on nearby agents). Similarly, constraints may be related to *assignment proximity*, as agents may only interact with agents assigned to the same task.

The structure of problems with *constrained interaction* has been exploited to simplify the learning of value functions (e.g., Proper and Tadepalli, 2009) or dynamics of the environment (e.g., Guestrin, Koller, et al., 2003). These approaches effectively generalize from easier to more difficult instances: we may train on small environments where the sample complexity is practical and generalize to large problems without ever training on them (*zero-shot generalization*). The main drawback is that when generalizing value functions or the dynamics, the optimal (or greedy) policy still needs to be recomputed at each new instance, which usually requires solving an optimization problem with complexity exponential in the number of objectives (e.g., maximizing the action-value function over the joint action space).

In this chapter, we build on the observation that in MAC problems with constrained interaction, optimal policies (or good approximations) can be effectively represented as a combination of coordination patterns that can be expressed as reactive rules, such as creating subgroups of agents to solve a single task, avoiding redundancy, or combinations of both. We decompose agents' policies into a high-level agent-task assignment policy and a low-level policy that prescribes the actual actions agents should take to solve the assigned task. As the most critical aspect of MAC problems is the coordination between agents, we assume low-level policies are provided in advance and we focus on learning effective high-level policies. To leverage the structure of the assignment policy, we propose a structured prediction approach, where agents are assigned to tasks as a result of an optimization problem. In particular, we distinguish between the *coordination inference procedure* (i.e., the optimization problem itself) and *scoring models* (the objective function) that provide a score to agent-task and task-task pairs. In its more complex instance, we define a quadratic inference procedure with linear constraints, where the objective function uses learned pairwise scores between agents and tasks for the linear part of the objective, and between different tasks for the quadratic part. With this structure we address the intrinsic exponential complexity of learning in large MAC problems through *zero-shot generalization*: **1)** the parameters of the scoring model can be learned in small instances, thus keeping the learning complexity low, **2)** the coordination inference procedure can be generalized to an arbitrary number of agents and tasks, as its computational complexity is polynomial in the number of agents and tasks. We study the effectiveness of this approach on a search and rescue problem and different battle scenarios in "StarCraft: Brood War". We show that the linear part of the optimization problem (i.e., using agent-task scores) represents simple coordination

patters such as assigning agents to their closest tasks, while the quadratic part (i.e., using task-task scores) may capture longer-term coordination such as spreading the different agents to tasks that are far away to each other or, on the contrary, create groups of agents that focus on a single task.

2.2 Related Work

Multi-agent reinforcement learning has been extensively studied, mostly in problems of decentralized control and limited communication (see Busoniu, Babuska, and De Schutter (2008) for a survey). By contrast, this chapter focuses on *centralized control under full state observation*.

Our work is closely related to generalization in relational Markov Decision Processes (Guestrin, Koller, et al., 2003) and decomposition approaches in loosely and weakly coupled MDPs (S. P. Singh and Cohn, 1998; Meuleau et al., 1998; Guestrin, Lagoudakis, and Parr, 2002; Tesauro, 2005; Proper and Tadepalli, 2009). The work on relational MDPs and the related object-oriented MDPs and first-order MDPs (Guestrin, Koller, et al., 2003; Diuk, Cohen, and Littman, 2008; Sanner and Boutilier, 2012) focus on learning and planning in environments where the state/action space is compactly described in terms of objects (e.g., agents) that interact with each other, without prior knowledge of the actual number of objects involved. Most of the work in this direction is devoted to either efficiently estimating the environment dynamics, or approximate the planning in new problem instances. Whereas the type of environments and problems we aim at are similar, we focus here on model-free learning of policies that generalize to new (and larger) problem instances *without* replanning.

Loosely or weakly coupled MDPs are another form of structured MDPs, which decompose into smaller MDPs with nearly independent dynamics. These works mostly follow a decomposition approach in which global action-value functions are broken down into independent parts that are either learned individually, or serve as guide for an effective parameterization for function approximation. The policy parameterization we develop follows the task decomposition approach of Proper and Tadepalli (2009), but the policy structures we propose are different. Proper and Tadepalli (2009) develop policies based on pairwise interaction terms between tasks and agents similar to our quadratic model, but the pairwise terms are based on interactions dictated by the dynamics of the environment (e.g., agent actions that directly impact the effect of other actions) aiming at a better estimation of the value function of low-level actions of the agents once an assignment is fixed, whereas our quadratic term aims at assessing the long-term value of an assignment.

Many deep reinforcement learning algorithms have been recently proposed to solve MAC problems with a variable number of agents, using different variations of communication and attention over graphs (Sukhbaatar, Szlam, and Fergus, 2016; Foerster, Assael, et al., 2016; Zambaldi et al., 2019; Yang et al., 2018; Jiang, Amo, and Lu, 2018; Jiang and Lu, 2018; Lowe et al., 2017; A. Singh, Jain, and Sukhbaatar, 2019). However, most of these algorithms focus on fixed-size action spaces, and little evidence has been given that these approaches generalize to larger problem instances (Usunier et al., 2016; Foerster, Farquhar, et al., 2018; Zambaldi et al., 2019). Rashid et al. (2018) and K. Lin et al. (2018) address the problem of learning (deep) decentralized policies with a centralized critic during learning in structured environments. While they do not address the problem of generalization, nor the problem of learning a centralized controller, we use their idea of a separate critic computed based on the full state information during training.

2.3 Multi-agent Task Assignment

We formalize a general MAC problem. To keep notation simple, we present a fixed-size description, but the end goal is to design policies that can be applied to environments of arbitrary size.

As customary in reinforcement learning, the objective of solving the tasks is encoded through a reward function that needs to be maximized over the long run by the coordinated actions of all agents. An environment with m tasks and n agents is modeled as an MDP $\langle \mathcal{S}^m, \mathcal{X}^n, \mathcal{A}^n, r, p \rangle$, where $\mathcal{S}$ is the set of possible states of each *task* (indexed by $j = 1, \dots, m$), $\mathcal{X}$ and $\mathcal{A}$ are the the set of states and actions of each agent (indexed by $i = 1, \dots, n$). We denote the joint states/actions by $\mathbf{s} \in \mathcal{S}^m$, $\mathbf{x} \in \mathcal{X}^n$, and $\mathbf{a} \in \mathcal{A}^n$. The reward function is defined as $r : \mathcal{S}^m \times \mathcal{X}^n \times \mathcal{A}^n \to \mathbb{R}$ and the stochastic dynamics is $p : \mathcal{S}^m \times \mathcal{X}^n \times \mathcal{A}^n \to \Delta(\mathcal{S}^m \times \mathcal{X}^m)$, where Δ is the probability simplex over the (next) joint state set. A joint deterministic policy is defined as a mapping $\pi : \mathcal{S}^m \times \mathcal{X}^n \to \mathcal{A}^n$. We consider the episodic discounted setting where the action-value function is defined as $Q^\pi(\mathbf{s}, \mathbf{x}, \mathbf{a}) = \mathbb{E}_\pi \left[r(\mathbf{s}, \mathbf{x}, \mathbf{a}) + \sum_{t=1}^{T} \gamma^t r(\mathbf{s}_t, \mathbf{x}_t, \mathbf{a}_t) \right]$, where $\gamma \in [0, 1)$, $\mathbf{a}_t = \pi(\mathbf{s}_t, \mathbf{x}_t)$ for all $t \geq 1$, $\mathbf{s}_t$ and $\mathbf{x}_t$ are sampled from p, and T is the time by when all tasks have been solved. The goal is to learn a policy π close to the optimal $\pi^* = \arg\max_\pi Q^\pi$ that we can easily generalize to larger environments.

Task decomposition. Following a similar task decomposition approach as (Tesauro, 2005) and (Proper and Tadepalli, 2009), we consider hierarchical policies that first assign each agent to a task, and where actions are given by a lower-level policy that only depends on the state of individual agents and the task they are assigned to. Denoting by $\mathcal{B} = \{\beta \in \{0, 1\}^{n \times m} : \sum_{j=1}^{m} \beta_{ij} = 1\}$ the set of assignment matrices of agents to tasks, an assignment policy first chooses $\hat{\beta}(\mathbf{s}, \mathbf{x}) \in \mathcal{B}$. In the second step, the action for each agent is chosen according to a lower-level policy $\tilde{\pi}$. Using $\pi_i(\mathbf{s}, \mathbf{x})$ to denote the action of agent i and $\hat{\beta}_i(\mathbf{s}, \mathbf{x}) \in \{1, ..., m\}$ for the task assigned to agent i, we have $\pi_i(\mathbf{s}, \mathbf{x}) = \tilde{\pi}(s_{\hat{\beta}_i(\mathbf{s}, \mathbf{x})}, x_i)$, where s_j and x_i are respectively the internal states of task j and agent i in the full state $(\mathbf{s}, \mathbf{x})$. In the following, we focus on learning high-level assignment policies responsible for the collaborative behavior, while we assume that the lower-level policy $\tilde{\pi}$ is *known and fixed*.

2.4 A Structured Prediction Approach

In this section we introduce a novel method for centralized coordination. We propose a structured prediction approach in which the agent-task assignment is chosen by solving an optimization problem. Our method is composed of two components: a **coordination inference procedure**, which defines the shape of the optimization problem and thus the type of coordination between agents and tasks, and a **scoring model**, which receives as input the state of agents and tasks and returns the parameters of the objective function of the optimization. The combination of these two components defines an agent-task assignment policy $\hat{\beta}$ that is then passed to the low-level policy $\tilde{\pi}$ (that we assume fixed) which returns the actual actions executed by the agents. Finally, we use a **learning algorithm** to learn the parameters of the scoring model itself in order to maximize the performance of $\hat{\beta}$. The overall scheme of this method is illustrated in Fig. 2.1.

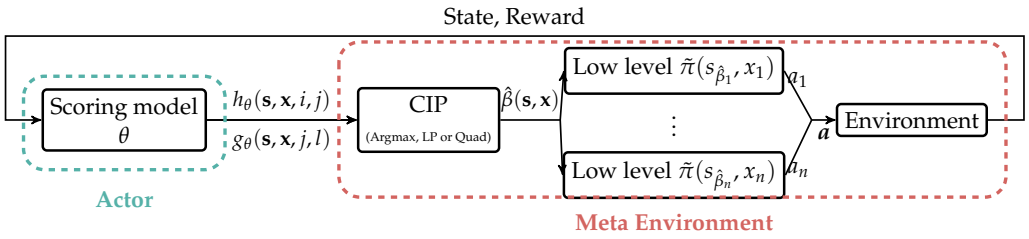


FIGURE 2.1: Illustration of the approach, where the agent-task assignment is computed by a coordination inference procedure (CIP) which receives as input agent-task (h) and task-task (g) scores computed by a scoring model parametrized by θ. The assignment $\hat{\beta}$ is then passed to fixed low level policies that return the actions played by each agent. The learning algorithm tunes θ and performs "meta-actions" h_θ and g_θ on to the "meta-environment" composed by the inference procedure, the low-level policies, and the actual environment.

2.4.1 Coordination Inference Procedures

The collaborative behaviors that we can represent are tied to the specific form of the objective function and its constraints. The formulations we propose are motivated by collaboration patterns important for long-term performance, such as creating subgroups of agents, or spreading agents across tasks.

Greedy assignment. The simplest form of assignment is to give a score to each agent-task pair and then assign each agent to the task with the highest score, ignoring other agents at inference time. In this approach, that we refer to as AMAX strategy, a model $h_\theta(\mathbf{x}, \mathbf{s}, i, j) \in \mathbb{R}$ parameterized by θ receives as input the full state and returns the score of agent i for task j. The associated policy is then

$$\hat{\beta}^{\text{AMAX}}(\mathbf{s}, \mathbf{x}, \theta) = \arg\max_{\beta \in \mathcal{B}} \sum_{i,j} \beta_{i,j} h_\theta(\mathbf{s}, \mathbf{x}, i, j), \tag{2.1}$$

which corresponds to assigning each agent i to the task j with largest score $h_\theta(\mathbf{x}, \mathbf{s}, i, j)$. As a result, the complexity of coordination is reduced from $O(m^n)$ (i.e., considering all possible agent-to-task assignments) down to a linear complexity $O(nm)$ (once the function h_θ has been evaluated on the full state). We also notice that AMAX bears strong resemblance to the strategy used in (Proper and Tadepalli, 2009), where the scores are replaced by approximate value functions computed for any agent-task pair.[2]

Linear Program assignment. Since AMAX ignores interactions between agents, it tends to perform poorly in scenarios where a task has a high score for all agents (i.e., $h(s, x, i, j)$ is large for a given j and for all i). In this situation, all agents are assigned to the same task, implicitly assuming that the "value" of solving a task is additive in the number of agents assigned to it (i.e., if n agents are assigned to the same task then we could collect a reward n times larger). While this may be the case when the number of agents assigned to the same task is small, in many practical scenarios this effect tends to saturate as more agents are assigned to a single task. A simple way to overcome this undesirable behavior is to impose a restriction on the number of agents assigned to a task. We can formalize this intuition by introducing $\mu_{(i,j)}(\mathbf{s}, \mathbf{x})$ as the *contribution* of an agent i to a given task j, and $u_j(\mathbf{s}, \mathbf{x})$ as the *capacity* of the task j. In the simplest case, we may know the maximum number of agents n_j that is necessary

[2] An alternative approach is to sample assignments proportionally to $h_\theta(\mathbf{s}, \mathbf{x}, i, j)$. Preliminary empirical tests of this procedure performed worse than AMAX and thus we do not report its results.

to solve each task j, and we can set the capacity of each task to be n_j, and all the contributions $\mu_{i,j}$ to be 1. Depending on the problem, the capacities and contributions are either prior knowledge or learned as a function of the state. Formally, denoting by $\overline{\mathcal{B}}(\mathbf{s}, \mathbf{x})$ the constrained assignment space

$$\overline{\mathcal{B}}(\mathbf{s}, \mathbf{x}) = \left\{ \beta \in \{0,1\}^{n \times m} \middle| \ \forall i, \sum_{j=1}^{m} \beta_{i,j} \leq 1; \forall j, \ \sum_{i=1}^{n} \mu_{i,j}(\mathbf{s}, \mathbf{x}) \beta_{i,j} \leq u_j(\mathbf{s}, \mathbf{x}) \right\}, \tag{2.2}$$

the resulting policy infers the assignment by solving an integer linear program

$$\hat{\beta}^{\text{LP}}(\mathbf{s}, \mathbf{x}, \theta) = \underset{\beta \in \overline{\mathcal{B}}(\mathbf{s}, \mathbf{x})}{\arg\max} \sum_{i,j} \beta_{i,j} h_\theta(\mathbf{s}, \mathbf{x}, i, j), \tag{2.3}$$

Notice that even with the additional constraints in (2.2), some agents may not be assigned to any task, hence inequality $\sum_{j=1}^{m} \beta_{i,j} \leq 1$ instead of strict equality.

In order to optimize (2.3) efficiently, we trade off accuracy for speed by solving its linear relaxation using an efficient LP library (*The Glop Linear Solver* n.d.), and retrieving a valid assignment using greedy rounding: Let us denote as $\beta^*_{i,j}$ the solution of the relaxed ILP; we iterate over agents i in descending order of $\max_j \beta^*_{i,j}$, and assign each agent to the task of maximum score that is not already saturated.

Quadratic Program assignment. The linear program above avoids straightforward drawbacks of a greedy assignment policy, but is unable to represent grouping patterns that are important on the long-run in coordination and collaboration problems. For instance, it may be convenient to "spread" agents among unrelated tasks, or, on the contrary, group agents together on a single task (up to the constraints) and then move to other tasks in a sequential fashion. Such grouping patterns can be well represented with a quadratic objective function of the form

$$\hat{\beta}^{\text{QUAD}}(\mathbf{s}, \mathbf{x}, \theta) = \underset{\beta \in \overline{\mathcal{B}}(\mathbf{s}, \mathbf{x})}{\arg\max} \left[\sum_{i,j} \beta_{i,j} h_\theta(\mathbf{s}, \mathbf{x}, i, j) + \sum_{i,j,k,l} \beta_{i,j} \beta_{k,l} g_\theta(\mathbf{s}, \mathbf{x}, j, l) \right], \tag{2.4}$$

where $g_\theta(\mathbf{x}, \mathbf{s}, j, l)$ plays the role of a (signed) distance between two tasks and $\overline{\mathcal{B}}$ is the same set of constraints as in (2.2). In the extreme case where $g_\theta(\mathbf{x}, \mathbf{s}, ., .)$ is a diagonal matrix, the quadratic part of the objective favors agents to carry on the same task (if the diagonal terms are positive) or on the contrary carry on different tasks (if the terms are negative). In general, negative $g_\theta(\mathbf{x}, \mathbf{s}, j, l)$ disfavors agents to be assigned to j and l at the same time step depending on $|g_\theta(\mathbf{x}, \mathbf{s}, j, l)|$. For instance, in the search and rescue problem, this captures the idea that agents should spread to explore the map.

As for the LP , we optimize a continuous relaxation of (2.4) using the same rounding procedure. The objective function may not be concave, because there is no reason for $g_\theta(\mathbf{x}, \mathbf{s}, ., .)$ to be negative semi-definite. In practice, we use the Frank-Wolfe algorithm (Frank and Wolfe, 1956) to deal with the linear constraints; the algorithm is guaranteed to converge to a local maximum and was efficient in our experiments.

2.4.2 Scoring Models

In order to allow generalizing the coordination policy $\widehat{\beta}$ to instances of different size, the h_θ and g_θ functions should be able to compute scores for pairs agents/tasks and tasks/tasks, independently of the actual amount of those. In order to make the presentation concrete, in the following we illustrate different scoring models in the case where the agents and tasks are objects located in a fixed-size 2D grid, and are

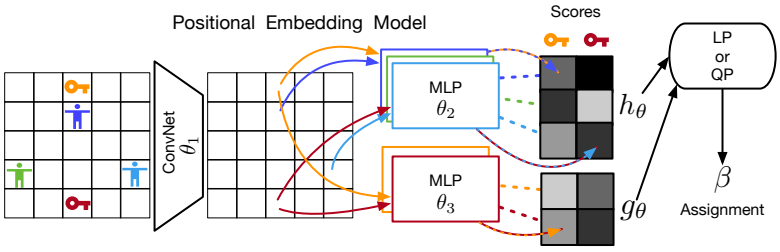


FIGURE 2.2: The PEM model for 3 agents and 2 tasks, arrows of interactions are only drawn for 2 agents each with one task, and one interaction of the two tasks. Agents are in cold (blue/green) colors, tasks are in warm (red/orange) colors.

characterized by an internal state. The position on the grid is part of this internal state.[3]

Direct Model (DM)

The first option is to use a fully decomposable approach (*direct model*), where the score for the pair (i, j) only depends on the internal states of agent i and task j: $h_\theta(\mathbf{s}, \mathbf{x}, i, j) = \tilde{h}_\theta(s_j, x_i)$ for some function $\tilde{h} : \mathcal{S} \times \mathcal{X} \to \mathbb{R}$. This model only uses the features of the pair of objects to compute the score. Precisely, $\tilde{h}_\theta(s_j, x_i)$ is obtained by concatenating the feature vectors of agent i and task j, and by feeding them to a fully-connected network of moderate depth. In the quadratic program strategy, the function g_θ follows the same structure as h (but uses different weights).

While this approach is computationally efficient, if used in the simple AMAX procedure (2.1), it leads to a policy that ignores interactions between agents altogether and is thus unable to represent effective collaboration patterns. As a result, the direct model should be paired with more sophisticated inference procedures to achieve more complex coordination patterns. On the other hand, as it computes scores by ignoring surrounding agents and tasks, once learned on small instances, it can be directly applied (i.e., zero-shot generalization) to larger instances independently of the number of agents and tasks.

General Model

An alternative approach is to take h_θ as a highly expressive function of the full state. The main challenge is this case is to define an architecture that can output scores for a variable number of agents and tasks. In the case where agents/tasks are in a 2D grid, we can define a **positional embedding model (PEM)** (see illustration in Fig.2.2) that computes scores following ideas similar to non-local networks (Xiaolong Wang et al., 2018). We use a deep convolutional neural network that outputs k features planes at the same resolution as the input. This implies that each cell is associated with k values that we treat as an embedding of the position. We divide this embedding in two sub-embeddings of size $k/2$, to account for the two kinds of entities: the first $k/2$ values represent an embedding of an agent, and the remaining ones represent an embedding of a task. To compute the score between two entities, we concatenate the embeddings of both entities and the input features of both of them, and run that

[3]Notice that the direct model illustrated below does not leverage this specific scenario, which, on the other hand, is needed to define the general model.

through a fully connected model, using the same topology as described for the direct model.

By leveraging the *full* state, this model can capture non-local interactions between agents and tasks (unlike the direct model) depending on the receptive field of the convolutional network. Larger receptive fields allow the model to learn more sophisticated scoring functions and thus better policies. Furthermore, it can be applied to variable number of agents and tasks as a position contains at most one agent and one task. Nonetheless, as it depends on the full state, the application to larger instances means that the model may be tested on data points outside the support of the training distribution. As a result, the scores computed on larger instances may be not accurate, thus leading to policies that can have trouble generalizing to more complex instances.

2.4.3 Learning Algorithm

As illustrated in Fig. 2.1, the learning algorithm optimizes a policy parametrized by θ that returns as actions the scores h_θ and g_θ, while the combination of the assignment $\widehat{\beta}$ returned by the optimization, the low-level policy $\widetilde{\pi}$ and the environment, plays the role of a "meta-environment". While any policy gradient algorithm could be used to optimize θ, in the experiments we use a synchronous Advantage-Actor-Critic algorithm (Mnih et al., 2016), which requires computing a state-value function. As advocated by Rashid et al. (2018) in the context of learning decentralized policies, we use a global value function that takes the whole state as input. We use a CNN similar to the one of PEM, followed by a spatial pooling and a linear layer to output the value. This value function is used only during training, hence its parametrization does not impact the potential generalization of the policy. The pseudo-code of the learning algorithm is presented in Algorithm 1 for the worker threads and Algorithm 2 for the learning threads.

Correlated exploration. Reinforcement learning requires some form of exploration scheme. Many algorithms using decompositional approaches for MAC problems (Guestrin, Lagoudakis, and Parr, 2002; Tesauro, 2005; Proper and Tadepalli, 2009; Rashid et al., 2018) rely on variants of Q-learning or SARSA and directly randomize the low-level actions taken by the agents. However, this approach is not applicable to our framework. In our case, the randomization is applied to the scores (denoted as $H_\theta(\mathbf{s}, \mathbf{x}, i, j)$ and $G_\theta(\mathbf{s}, \mathbf{x}, j, l)$) before passing them to the inference procedure. We can't use a simple gaussian noise, since at the beginning of the training, when the scoring model is random, it would cause the agents to be assigned to different tasks at each step, thus preventing them from solving any task and getting any positive reward. To alleviate this problem, we correlate temporally the consecutive realizations of H_θ and G_θ using auto-correlated noise as studied in (e.g., Wawrzynski, 2015), so that the actual sequence of assignments executed by the agent is also correlated. To correlate the parameters over p steps, at time t, we sample $H_{t,\theta}(i, j)$ according to (dropping dependence on $(\mathbf{s}_t, \mathbf{x}_t)$ for clarity): $\mathcal{N}\left(h_{t,\theta}(i, j) + \sum_{t'=t-p}^{t-1}(H_{t',\theta}(i, j) - h_{t',\theta}(i, j)), \frac{\sigma}{p}\right)$. This is equivalent to correlating the sampling noise over a sliding window of size p. During the update of the model, we ignore the correlation, and assume that the actions were sampled according to $\mathcal{N}(h_{t,\theta}(i, j), \sigma)$.

2.5 Experiments

We report results in two different problems: search and rescue and target selection in StarCraft. Both experiments are designed to test the generalization performance

Algorithm 1 Structured Prediction RL algorithm - Worker thread

// p is the number of correlated steps, σ the exploration standard deviation
// We denote by $Queue_p$ a queue of fixed size p. We assume we can sum all the elements in it.
$T \leftarrow 0$
repeat
  Start new a episode, $t \leftarrow 0$
  for all agent i and task j **do**
    Init noise history for the linear part to 0: $noise\_hist\_lin(i,j) = \text{Queue}_p(0)$
  end for
  for all pair of task j, k **do**
    Init noise history for the quadratic part to 0: $noise\_hist\_quad(j,k) = \text{Queue}_p(0)$
  end for
  repeat
    Observe state s_t
    for all agent i and task j **do**
      Compute $h_\theta(i,j,s_t)$ using the network
      Sample the actual action $H_{i,j}(s_t) \sim \mathcal{N}\left(h_\theta(i,j,s_t) + \sum noise\_hist\_lin(i,j), \frac{\sigma}{p}\right)$
      Store the current noise $noise\_hist\_lin(i,j)$.append$(H_{i,j}(s_t) - h_\theta(i,j,s_t))$
      Compute $\mu_{i,j}(s_t)$ the contribution of the agent to the task
      Compute $u_j(s_t)$ the capacity of the task
    end for
    for all pair of task j, k **do**
      Compute $g_\theta(j,k,s_t)$ using the network
      Sample the actual action $G_{j,k}(s_t) \sim \mathcal{N}\left(g_\theta(j,k,s_t) + \sum noise\_hist\_quad(j,k), \frac{\sigma}{p}\right)$
      Store the current noise $noise\_hist\_quad(j,k)$.append$(G_{j,k}(s_t) - g_\theta(j,k,s_t))$
    end for
    Compute the assignment using the constrained optimizer $\beta \leftarrow \text{solve}(H, G, \mu, u)$
    Execute the assignment in the environment, observe reward r_t
    The policy is $\pi(s_t) = [h_\theta(s_t), g_\theta(s_t)]$, and the action $a(s_t) = [H, G]$
    Send to the learner thread $(\pi(s_t), a(s_t), s_t, r_t)$
    $t \leftarrow t + 1$
    $T \leftarrow T + 1$
  until Episode ends
until $T > T_{max}$

of our method: we learn the scoring models on small instances and the learned policy is tested on larger instances with no additional re-training. We test different combinations of coordination inference procedures and scoring models. Among the inference procedures, AMAX should be considered as a basic baseline, while we expect LP to express some interesting coordination patterns. The QUAD is expected to achieve the better performance in the training instance, although its more complex coordination patterns may not effectively generalize well to larger instances. Among the scoring models, PEM should be able to capture dependencies between agents and tasks in a single instance but may fail to generalize when tested on instances with a number of agents and tasks not seen at training time. On the other hand, the simpler

Algorithm 2 Structured Prediction RL algorithm - Learner thread

// p is the number of correlated steps, N is the return length we use, σ the exploration standard deviation
γ is the discount factor, λ is the policy weight
repeat
  Reset gradients $d\theta \leftarrow 0$
  Receive N consecutive samples from a worker thread $[(\pi_1, a_1, s_1, r_1), \ldots, (\pi_N, a_N, s_N, r_N)]$
  $R = \begin{cases} 0 & \text{if } s_N \text{ is terminal} \\ V(s_N, \theta) & \text{otherwise} \end{cases}$
  for $t = N - 1$ **to** 1 **do**
    $R = \begin{cases} r_t & \text{if } s_t \text{ is terminal} \\ r_t + \gamma R & \text{otherwise} \end{cases}$
    Accumulate value loss $d\theta \leftarrow d\theta + \nabla_\theta \left(|R - V(s_t, \theta)|\right)$
    Compute advantage $A_t \leftarrow R - V(s_t)$
    Compute new policy $\pi_{\text{new},t}(s_t, \theta)$ using the network
    Compute likelihood of action according to old policy (assuming $a_t \sim \mathcal{N}(\pi_t, \sigma)$)
    $l_{\text{old}} \leftarrow \text{NormalPDF}(a_t, \pi_t, \sigma)$
    Compute likelihood according to new policy (assuming $a_t \sim \mathcal{N}(\pi_{\text{new},t}, \sigma)$)
    $l_{\text{new}} \leftarrow \text{NormalPDF}(a_t, \pi_{\text{new},t}, \sigma)$
    Compute importance ratio $ir \leftarrow \frac{l_{\text{new}}}{l_{\text{old}}}$
    Accumulate policy loss $d\theta \leftarrow d\theta + \lambda \nabla_\theta (ir * A * \log(l_{\text{new}}))$
  end for
  Take an optimization step according to $d\theta$.
until training ends

DM should generalize better if paired with a good coordination inference procedure.

The PEM + AMAX combination roughly corresponds to independent A2C learning and can be seen as the standard approach baseline, and we also provide strong hand-crafted baselines. Most previous approaches didn't aim achieving effective generalization, and often relied on fixed-size action spaces, rendering direct comparison impractical.

2.5.1 Search and Rescue

Setting. We consider a search and rescue problem on a grid environment of 16 by 16 cells. Each instance is characterized by a set of n ambulances (i.e., agents) and m victims (i.e., tasks). The goal is that all the victims are picked up by one of the ambulances as quickly as possible. This problem can be seen as a Multi-vehicle Routing Problem, which makes it NP-hard. Each ambulance can move in any of the 8 adjacent cells at each step, and is assumed to be able to pick-up an infinite number of victims. At the beginning of each episode, m victims and n ambulances are spawned uniformly at random on the grid. An ambulance picks up a victim as soon as it reaches its cell, regardless of whether this ambulance was effectively assigned to that victim or if it visited the cell contingently. When it happens, the state of the victim changes to reflect the fact that that task is solved, yet nothing prevents the model from continuing to assign ambulances to it.

The reward is -0.01 per time-step until the end of an episode (when all the victims have been picked up). The learning task is challenging because the reward is

TABLE 2.1: *Search and Rescue*. Average number of steps to solve the validation episodes, depending on the train scenario. Δ denotes the improvement over baseline. Best results are in bold, with an asterisk when they are statistically ($p < 0.0001$) better than the second best. Results like "*10.3*(1.1%)" mean that the evaluation failed in 1.1% of the test scenarios, and had an average score of 10.3 on the remaining 98.9%. In case of evaluation failures, the reported improvement over baseline are indicative (reported in italics between parenthesis).

	Train ($n \times m$)	Test	Baseline	Topline	AMAX-PEM	LP-PEM	QUAD-PEM	AMAX-DM	LP-DM	QUAD-DM
lower	2×4	2×4	14.34	10.28	11.98	12.09	**11.44**	13.78	11.98	11.55
is		5×10	13.61	7.19	13.36	10.69	*9.67*	12.49	10.24	**9.32***
better		8×15	11.8	n.a	*15.8*(0.7%)	9.86	*10.3*(1.1%)	11.06	9.71	**7.85***
lower	5×10	2×4	14.34	10.28	12.05	12.94	*13.23*(1%)	13.84	12.22	**11.78***
is		5×10	13.61	7.19	9.84	10.24	10.43	12.26	10.12	**9.36***
better		8×15	11.8	n.a	8.60	9.37	9.51	10.57	8.63	**7.95***
higher	In domain Δ				22%	20%	21%	7%	21%	**25%**
is	Out of domain Δ				(*22%*)	17%	(*18%*)	7%	21%	**29%**
better	Total Δ		0%	38%	(*18%*)	18%	(*19%*)	7%	21%	**28%**

uninformative and coupled; it is difficult for an agent to assign credit to the resolution of an individual task (i.e., picking up a victim). The assignment policy $\widehat{\beta}$ matches ambulances to victims, while the low-level policy $\widetilde{\pi}$ takes an action to reduce the distance between the ambulance and its assigned victim. The victims are static, their initial position is chosen randomly. In this environment, only one ambulance is needed to pick-up a particular victim, hence the saturation $u_j(\mathbf{s}, \mathbf{x})$ is set to 1.

Baseline and topline.

To evaluate the performance of the agents, we design a baseline and a topline, to get a ballpark estimate of the performances that are to be expected in the scenarios.

Baseline The baseline is a simple deterministic greedy policy that assigns each ambulance to the closest victim, regardless of what the others are doing. Given the movement patterns of the ambulances, the distance function to be considered between two points $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$ is $d(p_1, p_2) = \max(|x_1 - x_2|, |y_1 - y_2|)$. This corresponds to the number of steps needed for an ambulance starting in p_1 to reach p_2.

Topline For the small instances of the problem, it is possible to use an exact algorithm (topline) that solves the Multi-Vehicle Routing Problem exactly. To do it efficiently, we first pre-compute for each subset of victims (there are 2^m possible subsets), the shortest path to visit them all, starting from each of the victims of the subset (that is, if a subset contains 5 victims, we compute the 5 shortest paths that go through all of them starting from each of the victims). This pre-computation phase is done using a dynamic programming algorithm that has a complexity of $O(2^m m^3)$. This allows us to compute, for a given ambulance, what is the optimal way of visiting a given subset of the victims: we simply need to choose which victim to visit first and then execute the optimal path that we have cached between the remaining ones. What remains to be done is to partition the victims in n sets, and assign those sets to the n ambulances, so that the maximal time required by the ambulances to visit their subset optimally is minimized. To solve this partition/assignment problem

efficiently, we model it as an Integer Linear Program, and solve it exactly using an efficient branch-and-cut solver (SCIP (Gleixner et al., 2017))

Model

We now delve into the details of the models used. We recall that the method requires two networks: one for computing the value function, and one for the policy (i.e. the pairwise scores), which can be either a direct model or a PEM (see 2.4.2)

Features For all the models, the input of consists in 4 feature planes: the first one contains a 1 if the corresponding cell contains a victim (0 otherwise), the second contains a 1 if the corresponding cell contains an ambulance, and the last two planes contain respectively the x/y coordinates of the entity (victim or ambulance) contained in that cell, if there is any.

Network architectures The value network is a residual network (He, X. Zhang, et al., 2016) made of 3 residual blocks of 2 convolutional layers, with kernel size 3, stride 1, and padding such that the output of each layer has the same spatial dimension as the input. Each convolutional layer outputs 32 feature planes, and we use ReLUs as non-linearities, as well as BatchNorm layers for regularization. The output of the last block is fed to a fully connected layer, which outputs one single value, the value function. For the direct model, we use a simple fully connected network, with 3 linear layers separated by ReLUs, with 32 hidden units each. The architecture of the PEM network is the same as the value network.

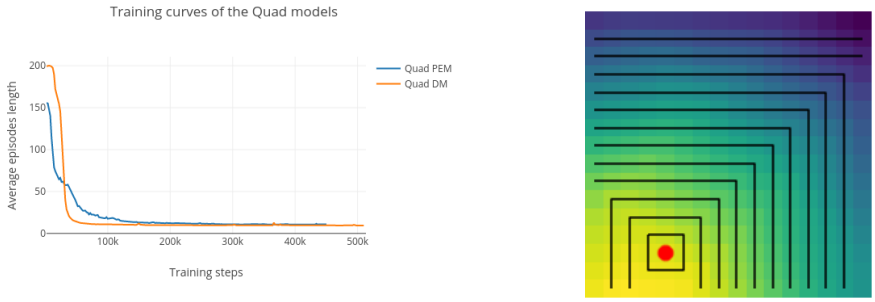
Training

We train the model using a synchronous Advantage-Actor-Critic algorithm (A2C) (see Mnih et al., 2016 for the asynchronous version), using the ELF platform (Y. Tian et al., 2017). We batch 128 observations together, and run 256 agents in parallel.

We optimize the hyper-parameters of the learning procedure using a random search. We sample 128 sets of hyper-parameters for each combination of model/scenario, and we train the models during 8 hours on a Nvidia Volta. We report the performance of the best performing model after training according to the average reward on training problems. The parameters that were tuned are the following: learning rate of the value network (we sample $c \in [0, 5]$, uniformly at random, and use 10^{-c}), the learning rate of the policy network (same sampling), the variance of the random policy (uniformly in $[0.1, 2]$), the number of correlated steps in the exploration (uniformly in $[1, 10]$), the number of steps of the n-step return (uniformly in $[2, 5]$), and the optimization algorithm (SGD or Adam Kingma and Ba, 2015).

Experiments

We trained our models on two instances ($n = 2, m = 4$ and $n = 5, m = 10$) and we test them on the training scenarios, as well as in out of domain instances with a larger number of victims and ambulances. At test time, we evaluate the policies on a fixed set of 1000 random episodes (with different starting positions). The agents use the same variance and number of correlated steps as they had during training. The results are summarized in Tab. 2.1, where we report the average number of steps required to complete the episodes. Because of its computational cost, the topline for



(A) (B)

FIGURE 2.3: (left) Learning curves of the Quad models; (right) The distance from the red dot computed by the model (depicted as the density), is skewed towards the corner, compared to the ground-truth distance (depicted as the contours)

the biggest instance (8×15) is not available. Numbers in bold denote the best model, numbers with an asterisk indicate that the model statistically outperforms the second best one, according to the Wilcoxon test (Wilcoxon, 1945), with a p-value of less than 0.0001. Numbers in italics indicate models for which the evaluation got stuck in an endless loop in some of the scenarios. The failure rate is indicated between brackets, and we report, for information, the performance without taking the failures into account. In the last rows of the table, we aggregate the average improvements over the baseline ($100 * \frac{\text{baseline}-\text{method}}{\text{baseline}}$). The in-domain scores correspond to the scores obtained when the test instance matches the train instance. Conversely, the out of domain scores correspond to the performances on unseen instances. Note that no model was trained on 8×15.

Results. We first notice that the PEM scoring model tends to overfit to the train scenario, leading to poor generalization (i.e., in some configuration it fails to complete the problem). On the other hand, for the DM, the generalization is very stable. Regarding the inference procedures [4], AMAX tends to perform at least as well as the greedy baseline, by learning how to compute the relevant distance function between an ambulance and a victim. The LP strategy can rely on the same distance function and perform better, since it enforces the coordination between agents and avoids sending more than one ambulance to the same victim. Finally, the QUAD strategy is able to learn long-term strategies, and in particular how to spread efficiently the ambulances across the map (e.g., if two victims are very close, it is wasteful to assign two distinct ambulances to them, since one can efficiently pick-up both victims sequentially, while the other ambulance can be assigned to further victims).

A notable fact is that the AMAX DM agent performs on average 7% better than the baseline. This is unexpected since the only features it has access to are the positions of the ambulance and the victim, thus it would be expected to learn to send each ambulance to the closest victim. We credit this development to the ability of the model to learn to break ties between equally close victims, which it does by choosing the victims that are on the furthest peripheral positions of the map. Since these positions tend to be the outliers, they are on average more difficult to access than the ones in the middle, so it is beneficial to favor them. The baseline breaks ties randomly and thus does not account for that. This can be observed in Fig 2.3b in which we plotted the similarity function learned by the model, by computing the score the model would

[4] We study their performance when paired with DM.

give when the agent is located at the red dot and the task is located anywhere else. While the function it generates resembles the ground-truth distance, the distribution is slightly skewed towards the bottom-left corner.

Overall, the constrained strategies (LP and QUAD) perform better than the AMAX strategy, and the the QUAD strategy is better than the LP strategy, irrespective of the feature model used. This is an evidence that the structure introduced into the inference procedure is indeed able to improve the performance of the agent. These strategies also leverage the structure of the problem through the constraints on the assignment, which means that there is less room for learning something that cannot be computed directly from the pairwise positions using the additional information provided by the access to the full view. Since the extra information does not directly help, it slows down the learning procedure, and hinders the final performance. This can be seen in Fig 2.3a, where the PEM agent is slower to converge, and converges to a worse performance than the DM agent, when both use the QUAD strategy.

2.5.2 Target Selection in StarCraft

StarCraft is a real-game strategy game, in which the player must build an army and control individual units to destroy her opponent's. In this paper, we do not consider full games of StarCraft, but rather focus on a specific sub-problem in StarCraft: battles between two groups of units. This setting, often referred to as *micromanagement*, has already been studied in the literature, using a mixture of scripted behaviours and search (Churchill, Saffidine, and Buro, 2012; Churchill and Buro, 2013; Ontanón et al., 2013; Churchill, Z. Lin, and Synnaeve, 2017), or using RL techniques (Usunier et al., 2016; Foerster, Farquhar, et al., 2018). In these battles, a crucial aspect of the policy is to assign a target enemy unit (*the task*) to each of our units (*the agents*), in a coordinated way. Since we focus on the agent-task assignment (the high-level policy $\widehat{\beta}$), we use a simple low-level policy for the agents (neither learnt nor scripted) relying on the built-in "attack" command of the game, which moves each unit towards its target, possibly avoiding obstacles or other units on the way, and shoots as soon as the target is in range. This contrasts with previous works, which usually allow more complex movement patterns (e.g., retreating while the weapon is reloading). While such low-level policies could be integrated in our framework, we preferred to use the simplest "attack" policy to better assess the impact of the high-level coordination.

In this problem, the capacity $u_j(\mathbf{s}, \mathbf{x})$ of a task j is defined as the remaining health of the enemy unit, and the contribution $\mu_{i,j}(\mathbf{s}, \mathbf{x})$ of an agent i to this task is defined as the amount of damage dealt by unit i to the enemy j. These constraints are meant to avoid dealing more damage to an enemy than necessary to kill it, a phenomenon known as *over-killing*.

Experimental setup

All the experiments are played on a fully empty map, where the units are centered in the middle, outside their respective fire range. We disable the Fog-of-War, meaning that all units have full vision of the whole map. Because of the kind of movement policy we use, the units never go near the edge of the map, hence there is no collision involved but the collisions between units themselves. The bot takes its actions every 3 frames, but we reevaluate the target assignment only every 6 frames. If the target of a unit dies in the meantime, then it does nothing until the next re-assignment. The

reward at time t is defined as

$$r_t = (\text{ourHealth}_t - \text{ourHealth}_{t-1} + \text{theirHealth}_{t-1} - \text{theirHealth}_t)/\text{ourHealth}_0.$$

At the beginning of every battle, the units are spawned according to a seeded normal distribution, which is skewed towards the Y dimension, in order to form more coherent initial spread of the units. Even though the starting positions are deterministic with respect to the seed, the outcome of a battle is not, even if the policy of the players are totally deterministic. This is due to the internal randomness of the game engine, which affects things like the random miss probability of each attack (any attack has a $\frac{1}{256}$ probability of failing) and the initial orientation of the units.

The opponent army is controlled by the built-in AI. In practice, it means that we give an "attack-move" order to the opponent, with the target position being the centroid of our units. This order is repeated every 60 frames with an updated centroid. The attack-move order causes the built-in AI to take over the unit, globally moving it towards the target position and attacking any visible unit along the way.

We wish to emphasize the fact that all the details of the experimental protocol have a significant impact on the outcome of the battles. In particular, the frequency at which the opponent's attack-move command is re-issued matters: if the frequency is too high, then the units tend to attack less, because spamming orders have some counter-intuitive effects on the game engine. Conversely, if the frequency is too low, then the attack point might be significantly off the centroid of our army, leading to sub-optimal attacking behaviour. Similarly, the initial positions of the units plays an important role. As a result, the exact win-rate are not directly comparable with previous works, where neither precise details of the setup nor source code are available. The differences can be observed even in the win-rate of the baseline heuristics. For this work, we refer to the source code in the supplementary material for the exact details used.

Scenarios We consider three different kinds of scenarios:

Wraith These are ranged flying units, which means that they don't collide with any other unit. We denote these scenarios as wNvM where N is the size of our army, and M is the size of the enemy army. Following the previous works, we train on the imbalanced scenario w15v17, where the two additional units given to the opponent are required to make the scenario challenging.

Marines These are ranged ground units, which do have collisions. We denote these scenarios as mNvM where N is the size of our army, and M is the size of the enemy army. Since the built-in AI has a better control policy for ground units, we train on the balanced scenario m10v10, which is challenging enough.

Zergling-Hydralisk Zerglings are fast melee units (they can only attack if they are in contact of their target), while Hydralisks are slower ranged units. In this scenario, we investigate the opportunity to learn distinct behaviours depending on the type of the agent or its task. To further amplify their relative capabilities, we give the Zerglings a speed boost ("Metabolic Boost" upgrade), and the Hydralisks a range boost ("Grooved Spines" upgrade). We denote these scenarios as "zh$N$vM" which correspond to N Zerglings and N Hydralisks versus M Zerglings and M Hydralisks. We train on zh10v10.

Baseline Heuristics To properly assess the strength of our model, we use a set of baseline heuristics designed to get a good baseline performance in our scenarios.

Some of them are standard heuristics from the literature (Usunier et al., 2016), and we also propose some improved versions to provide an attempt at the best scripted policy that could be used in a bot. Note that for fair comparison with the model, we focus on the targeting exclusively, and none of the baselines incorporate special movement behavior. We recall that generally, in these battles, two things matter predominantly:

1. Focus firing. The intuition is that it is crucial to kill enemy units as quickly as possible to create a numeric advantage that snowballs into victory.

2. Avoid over-killing. The goal is here is to avoid wasting any shot. If we have 3 units dealing 10 damage each, and an enemy with 22 hit-points remaining, then assigning all 3 units to this enemy will deal an expected 30 damage, hence wasting $30 - 22 = 8$ damage. It would probably be wiser to assign one of our units to another target instead.

Both of these design goals are in conflict, hence the difficulty lies in balancing them properly, particularly when some enemy units are close to dying. With these design goals in mind, here are the heuristics we consider as a comparison. Note that these heuristics are re-evaluated every 6 frames.

Closest (c) Each unit independently picks the units that is the closest, as measured by the distance function used by the game engine. Ties are broken using the unit internal ID, which is randomly assigned at the beginning of the game but consistent for the duration of the episode.

Weakest Closest (wc) All units are collectively assigned to the weakest enemy unit. The distance is used as a tie breaker.

Weakest closest No-Overkill (wcnok) We select the weakest-closest enemy unit, then assign greedily in an arbitrary order as many units as possible as long as the total sum of damage to this unit is lesser than its health. When this enemy is saturated, if some of our units don't have a target yet, then we select the second weakest-closest, and so on until all enemy units have been exhausted or all our units have a target.

Weakest Closest No-Overkill No Change(wcnoknc) Same as wcnok, but once a unit starts attacking a target, it keeps doing so until the target dies. When the target dies, a new target is computed as in wcnok. Keeping the same target can reduce some instability in the assignment found by wcnok, but can lead to over-killing.

Weakest Closest No-Overkill Smart (wcnoks) Same as wcnoknc, except that the target is kept only as long as it doesn't risk causing over-killing. When it does, a new target is computed as in wcnok.

Random No change (rand-nc) Each unit pick a target at random at the beginning of the episode, and keep attacking it until it is dead. When that happens, a new target is picked randomly.

Training

Given the poor results of PEM in the previous experiment, we only train DM with all the possible inference procedures. Each unit is represented by its features: whether it

TABLE 2.2: Best hyper-parameters found through random search on the different models

Experiment	Model	Learn. rate	Policy-loss weight λ	Explor. std-dev σ	Returns length	# correlated steps
Marine	QUAD	4.272e-05	1.585e+01	2.910e+00	5	10
	LP	2.280e-05	2.157e-03	7.017e-01	6	5
	AMAX	4.330e-05	5.396e-01	2.418e+00	2	2
Wraith	QUAD	2.826e-05	2.514e+02	1.899e+00	6	7
	LP	5.600e-05	5.534e-01	4.244e-01	4	6
	AMAX	2.525e-05	6.847e+01	1.994e+00	8	10
Zergling-Hydra	QUAD	5.325e-05	6.277e-01	7.185e-01	3	3
	LP	6.534e-05	4.455e-02	2.902e+00	8	1
	AMAX	1.885e-05	2.119e-02	1.881e+00	3	10

is an enemy, its position, velocity, current health, range, cool-down (number of frames before the next possible attack), and one hot encoding of its type. This amounts to 8 to 10 features per units, depending on the scenario. For training, we sample 100 sets of hyper-parameters for each combination of model/scenario, and train them for 8 hours on a Nvidia Volta with a batch-size of 32. For the learning rate we sample c u.a.r in $[-6, -3]$, and use 10^c, for the policy-loss weight λ, we sample d u.a.r in $[-3, 3]$ and use 10^d, the exploration standard deviation σ is sampled u.a.r in $[0.1, 3]$, the return-length used in A2C is an integer sampled u.a.r in $[2, 10]$, and the number of correlated exploration steps is an integer sampled u.a.r. in $[1, 10]$. The best parameters found are reported in Table 2.2.

The scoring models h_θ and g_θ are fully-connected networks consisting of 3 linear layers with ReLU. To compute $h_\theta(i, j)$, we give as input to the network the concatenation of the features of ally unit i and the features of enemy unit j, along with 2 additional features: a boolean flag that indicates whether i was attacking j in the previous step (this is meant to facilitate temporal consistency of the actions), and the distance between both units, as computed by the internal game engine. The input of $g_\theta(j, k)$ only contains the features of enemy units j and k, with no additional features.

In this experiment, we found that the training algorithm is relatively sensitive to the random seed. To better assess the performances, we re-trained the best set of hyper-parameters for each model/scenario on 10 random seeds, for 18 hours. The performances we report are the median of the performances of all the seeds, to alleviate the effects of the outliers. The results are aggregated in Tab. 2.3. Although the number of units is a good indicator of the difficulty of the environment, whether the numbers of units are balanced in both teams dramatically change the "dynamics" of the game. For instance, zh10v12 is unbalanced and thus much more difficult than zh11v11, which is balanced. The performance of the baseline can be seen as a relatively accurate estimate of the difficulty of the scenario.

Results

Results are provided in Table 2.3. As StarCraft is a real-time game, one first concern regards the runtime of our algorithm. In the biggest experiment, involving 80 units vs 82, our algorithm returns actions in slightly more than 500ms (5ms for the forward in the model, 500ms to solve the inference of QUAD). Given the frequency at which we take actions (every 6 frames), such timings allow real-time play in StarCraft.

Amongst the scenarios, the Wraith setting (wNvM) are the ones where the assumption of independence between the tasks holds the best, since in this case there are no collisions between units. These scenarios also require good coordination, since it is important to focus fire on the same unit. Indeed, for these scenarios, the best

TABLE 2.3: Results on StarCraft: average win-rate of the different methods and all the heuristics. Bests results are in bold, the best heuristic on each scenario is in italics. We report 95% confidence intervals (using the Normal approximation interval).

		Heuristics						**RL**		
Train	Test	c	wc	wcnok	wcnoknc	wcnoks	rand-nc	LP DM	QUAD DM	AMAX
m10v10	m5v5	0.77 ± 0.03	0.88 ± 0.02	0.56 ± 0.03	0.86 ± 0.02	0.83 ± 0.02	0.15 ± 0.02	$\mathbf{0.90 \pm 0.02}$	0.83 ± 0.02	0.84 ± 0.02
	m10v10	0.77 ± 0.03	0.44 ± 0.03	0.00 ± 0.00	0.45 ± 0.03	0.56 ± 0.03	0.01 ± 0.01	$\mathbf{0.94 \pm 0.02}$	0.83 ± 0.02	0.82 ± 0.02
	m10v11	0.25 ± 0.03	0.07 ± 0.02	0.00 ± 0.00	0.05 ± 0.01	0.11 ± 0.02	0.00 ± 0.00	$\mathbf{0.52 \pm 0.03}$	0.28 ± 0.03	0.29 ± 0.03
	m15v15	0.75 ± 0.03	0.03 ± 0.01	0.00 ± 0.00	0.14 ± 0.02	0.18 ± 0.02	0.00 ± 0.00	$\mathbf{0.92 \pm 0.02}$	0.69 ± 0.03	0.77 ± 0.03
	m15v16	0.40 ± 0.03	0.00 ± 0.00	0.00 ± 0.00	0.03 ± 0.01	0.02 ± 0.01	0.00 ± 0.00	$\mathbf{0.68 \pm 0.03}$	0.32 ± 0.03	0.43 ± 0.03
	m30v30	0.69 ± 0.03	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	$\mathbf{0.74 \pm 0.03}$	0.06 ± 0.02	0.36 ± 0.03
w15v17	w15v17	0.07 ± 0.02	0.00 ± 0.00	0.58 ± 0.03	0.33 ± 0.03	0.81 ± 0.02	0.01 ± 0.01	0.53 ± 0.03	$\mathbf{0.89 \pm 0.02}$	0.30 ± 0.03
	w30v34	0.01 ± 0.01	0.00 ± 0.00	0.36 ± 0.03	0.31 ± 0.03	0.90 ± 0.02	0.01 ± 0.01	0.76 ± 0.03	$\mathbf{0.99 \pm 0.01}$	0.37 ± 0.03
	w30v35	0.00 ± 0.00	0.00 ± 0.00	0.10 ± 0.02	0.08 ± 0.02	0.60 ± 0.03	0.01 ± 0.00	0.56 ± 0.03	$\mathbf{0.94 \pm 0.02}$	0.24 ± 0.03
	w60v67	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.07 ± 0.02	0.00 ± 0.00	0.33 ± 0.03	$\mathbf{0.72 \pm 0.03}$	0.13 ± 0.02
	w60v68	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.01 ± 0.01	0.00 ± 0.00	0.21 ± 0.03	$\mathbf{0.52 \pm 0.03}$	0.07 ± 0.02
	w80v82	0.00 ± 0.00	0.00 ± 0.00	0.09 ± 0.02	0.08 ± 0.02	0.32 ± 0.03	0.00 ± 0.00	0.11 ± 0.02	$\mathbf{0.36 \pm 0.03}$	0.03 ± 0.01
zh10v10	zh10v10	0.86 ± 0.02	0.26 ± 0.03	0.00 ± 0.00	0.54 ± 0.03	0.64 ± 0.03	0.00 ± 0.00	$\mathbf{0.90 \pm 0.02}$	0.83 ± 0.02	0.84 ± 0.02
	zh10v11	0.30 ± 0.03	0.01 ± 0.01	0.00 ± 0.00	0.04 ± 0.01	0.09 ± 0.02	0.00 ± 0.00	$\mathbf{0.46 \pm 0.03}$	0.24 ± 0.03	0.40 ± 0.03
	zh10v12	0.03 ± 0.01	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	$\mathbf{0.06 \pm 0.02}$	0.01 ± 0.01	$\mathbf{0.06 \pm 0.01}$
	zh11v11	$\mathbf{0.87 \pm 0.02}$	0.15 ± 0.02	0.00 ± 0.00	0.38 ± 0.03	0.56 ± 0.03	0.00 ± 0.00	$\mathbf{0.87 \pm 0.02}$	0.75 ± 0.03	0.80 ± 0.02
	zh12v12	$\mathbf{0.85 \pm 0.02}$	0.05 ± 0.01	0.00 ± 0.00	0.23 ± 0.03	0.42 ± 0.03	0.00 ± 0.00	0.82 ± 0.02	0.64 ± 0.03	0.75 ± 0.03

performing heuristics are the more complicated ones that focus fire on the weakest while preventing over-killing. This is the type of policies that only the QUAD model is able to represent, explaining its dominance in this scenario. During these battles, both armies tend to overlap totally, hence it becomes almost impossible to use surrogate coordination principles based on spatial disposition only, such as targeting the closest unit. In this case, the quadratic part of the score function is crucial to learn focus-firing and results show that without the ability to represent such a long-term coordination pattern, both LP and AMAX fail to reach the same level of performance. Notably, the coordination pattern learned by QUAD generalizes well, outperforming the best heuristics in instances as much as 5 times the size of the training instance.

The other settings, Marine (mNvM) and Zergling-Hydralisk (zh$N$vM) break the independence assumption because the units now have collisions. It is even worse for the Zerglings, since they are melee units. This can be seen because the best performing heuristic tends to be *closest*, showing that it might be better for each unit to focus on close-by enemies, and avoid picking faraway targets that would result in maneuvering and thus likely losing of the army cohesion. In the Marine scenario, all three models usually learn to wait (by picking no target) in the beginning of the battle, and start picking a target only when the enemies are about to get in range. This strategy allows them to keep a rather good formation, which they would otherwise lose, had they picked a wrong target. This gives them a little edge in the battle, and then they proceed by following a strategy that looks like an optimized version attacking the closest unit. In the Zergling-Hydralisk scenarios, the models learn to wait in the beginning as well. This is an exploit on the rushing behaviour of the opponent: the faster zerglings of the enemy will engage first, while its hydralisks are lagging behind. This allows the models to first clear up the zergling wave with the combined forces of their zerglings and hydras, before turning to the enemy hydralisks. The coordination patterns are then harder to learn for the QUAD model, and they generalize poorly. However, these scenarios with collisions also tend to require less long-term coordination, and the immediate coordination patterns learned by the LP model are enough to significantly outperform the heuristics, even when transferring to unseen instances.

2.6 Conclusion

In this chapter we proposed a structured approach to multi-agent coordination. Unlike previous work, it uses an optimization procedure to compute the assignment of agents to tasks and define suitable coordination patterns. The parameterization of this optimization procedure is seen as the continuous output of an RL trained model. We showed on two challenging problems the effectiveness of this method, in particular in generalizing from small to large instance, and discussed the impact of the choice of the expressivity of the score-functions on the generalization abilities of the model.

Chapter 3

End-to-End Object Detection with Transformers

Abstract

We present a new method that views object detection as a direct set prediction problem. Our approach streamlines the detection pipeline, effectively removing the need for many hand-designed components like a non-maximum suppression procedure or anchor generation that explicitly encode our prior knowledge about the task. The main ingredients of the new framework, called DEtection TRansformer or DETR, are a set-based global loss that forces unique predictions via bipartite matching, and a transformer encoder-decoder architecture. Given a fixed small set of learned object queries, DETR reasons about the relations of the objects and the global image context to directly output the final set of predictions in parallel. The new model is conceptually simple and does not require a specialized library, unlike many other modern detectors. DETR demonstrates accuracy and run-time performance on par with the well-established and highly-optimized Faster R-CNN baseline on the challenging COCO object detection dataset. Moreover, DETR can be easily generalized to produce panoptic segmentation in a unified manner. We show that it significantly outperforms competitive baselines. Training code and pretrained models are available at `https://github.com/facebookresearch/detr`.

3.1 Introduction

The goal of object detection is to predict a set of bounding boxes and category labels for each object of interest. Modern detectors address this set prediction task in an indirect way, by defining surrogate regression and classification problems on a large set of proposals (S. Ren et al., 2015b; Cai and Vasconcelos, 2019), anchors (T.-Y. Lin, Goyal, et al., 2017), or window centers (Zhou, Dequan Wang, and Krähenbühl, 2019; Z. Tian et al., 2019). Their performances are significantly influenced by postprocessing steps to collapse near-duplicate predictions, by the design of the anchor sets and by the heuristics that assign target boxes to anchors (S. Zhang et al., 2020). To simplify these pipelines, we propose a direct set prediction approach to bypass the surrogate tasks. This end-to-end philosophy has led to significant advances in complex structured prediction tasks such as machine translation or speech recognition, but not yet in object detection: previous attempts (Stewart, Andriluka, and Ng, 2016; Hosang, Benenson, and Schiele, 2017; Bodla et al., 2017; S Hamid Rezatofighi et al., 2018) either add other forms of prior knowledge, or have not proven to be competitive with strong baselines on challenging benchmarks. In this chapter, we aim to bridge this gap.

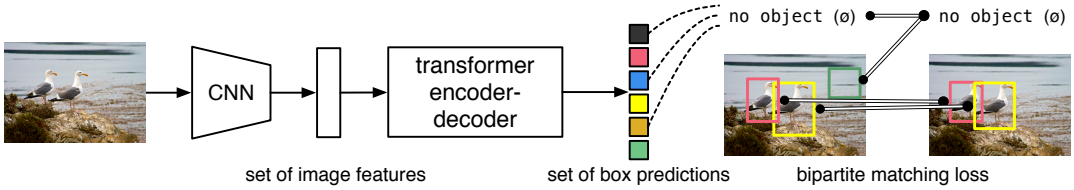


FIGURE 3.1: DETR directly predicts (in parallel) the final set of detections by combining a common CNN with a transformer architecture. During training, bipartite matching uniquely assigns predictions with ground truth boxes. Prediction with no match should yield a "`no object`" (∅) class prediction.

We streamline the training pipeline by viewing object detection as a direct set prediction problem. We adopt an encoder-decoder architecture based on transformers (Vaswani et al., 2017), a popular architecture for sequence prediction. The self-attention mechanisms of transformers, which explicitly model all pairwise interactions between elements in a sequence, make these architectures particularly suitable for specific constraints of set prediction such as removing duplicate predictions.

Our DEtection TRansformer (DETR, see Figure 3.1) predicts all objects at once, and is trained end-to-end with a set loss function which performs bipartite matching between predicted and ground-truth objects. DETR simplifies the detection pipeline by dropping multiple hand-designed components that encode prior knowledge, like spatial anchors or non-maximal suppression. Unlike most existing detection methods, DETR doesn't require any customized layers, and thus can be reproduced easily in any framework that contains standard ResNet (He, X. Zhang, et al., 2016) and Transformer (Vaswani et al., 2017) classes.

Compared to most previous work on direct set prediction, the main features of DETR are the conjunction of the bipartite matching loss and transformers with (non-autoregressive) parallel decoding (Oord et al., 2018; Gu et al., 2018; Ghazvininejad et al., 2019; Devlin et al., 2019). In contrast, previous work focused on autoregressive decoding with RNNs (Stewart, Andriluka, and Ng, 2016; Romera-Paredes and Torr, 2015; Park and Berg, 2015; M. Ren and Zemel, 2017; Salvador et al., 2017). Our matching loss function uniquely assigns a prediction to a ground truth object, and is invariant to a permutation of predicted objects, so we can emit them in parallel.

We evaluate DETR on one of the most popular object detection datasets, COCO (T.-Y. Lin, Maire, et al., 2014), against a very competitive Faster R-CNN baseline (S. Ren et al., 2015b). Faster R-CNN has undergone many design iterations and its performance was greatly improved since the original publication. Our experiments show that our new model achieves comparable performances. More precisely, DETR demonstrates significantly better performance on large objects, a result likely enabled by the non-local computations of the transformer. It obtains, however, lower performances on small objects. We expect that future work will improve this aspect in the same way the development of FPN (T.-Y. Lin, Dollár, et al., 2017) did for Faster R-CNN.

Training settings for DETR differ from standard object detectors in multiple ways. The new model requires extra-long training schedule and benefits from auxiliary decoding losses in the transformer. We thoroughly explore what components are crucial for the demonstrated performance.

The design ethos of DETR easily extend to more complex tasks. In our experiments, we show that a simple segmentation head trained on top of a pre-trained

DETR outperfoms competitive baselines on Panoptic Segmentation (Kirillov, He, et al., 2019), a challenging pixel-level recognition task that has recently gained popularity.

3.2 Related work

Our work build on prior work in several domains: bipartite matching losses for set prediction, encoder-decoder architectures based on the transformer, parallel decoding, and object detection methods.

3.2.1 Set Prediction

There is no canonical deep learning model to directly predict sets. The basic set prediction task is multilabel classification (see e.g., Seyed Hamid Rezatofighi et al., 2017; Pineda et al., 2019 for references in the context of computer vision) for which the baseline approach, one-vs-rest, does not apply to problems such as detection where there is an underlying structure between elements (i.e., near-identical boxes). The first difficulty in these tasks is to avoid near-duplicates. Most current detectors use postprocessings such as non-maximal suppression to address this issue, but direct set prediction are postprocessing-free. They need global inference schemes that model interactions between all predicted elements to avoid redundancy. For constant-size set prediction, dense fully connected networks (Erhan et al., 2014) are sufficient but costly. A general approach is to use auto-regressive sequence models such as recurrent neural networks (Vinyals, S. Bengio, and Kudlur, 2016). In all cases, the loss function should be invariant by a permutation of the predictions. The usual solution is to design a loss based on the Hungarian algorithm (Kuhn, 1955), to find a bipartite matching between ground-truth and prediction. This enforces permutation-invariance, and guarantees that each target element has a unique match. We follow the bipartite matching loss approach. In contrast to most prior work however, we step away from autoregressive models and use transformers with parallel decoding, which we describe below.

3.2.2 Transformers and Parallel Decoding

Transformers were introduced by Vaswani et al., 2017 as a new attention-based building block for machine translation. Attention mechanisms (Bahdanau, Cho, and Y. Bengio, 2015) are neural network layers that aggregate information from the entire input sequence. Transformers introduced self-attention layers, which, similarly to Non-Local Neural Networks (Xiaolong Wang et al., 2018), scan through each element of a sequence and update it by aggregating information from the whole sequence. One of the main advantages of attention-based models is their global computations and perfect memory, which makes them more suitable than RNNs on long sequences. Transformers are now replacing RNNs in many problems in natural language processing, speech processing and computer vision (Devlin et al., 2019; Lüscher et al., 2019; Synnaeve, Xu, et al., 2019; Radford et al., 2019; Parmar et al., 2018).

Transformers were first used in auto-regressive models, following early sequence-to-sequence models (Sutskever, Vinyals, and Le, 2014), generating output tokens one by one. However, the prohibitive inference cost (proportional to output length, and hard to batch) lead to the development of parallel sequence generation, in the domains of audio (Oord et al., 2018), machine translation (Gu et al., 2018; Ghazvininejad et al., 2019), word representation learning (Devlin et al., 2019), and more recently speech

recognition (Chan et al., 2020). We also combine transformers and parallel decoding for their suitable trade-off between computational cost and the ability to perform the global computations required for set prediction.

3.2.3 Object detection

Most modern object detection methods make predictions relative to some initial guesses. Two-stage detectors (S. Ren et al., 2015b; Cai and Vasconcelos, 2019) predict boxes w.r.t. proposals, whereas single-stage methods make predictions w.r.t. anchors (T.-Y. Lin, Goyal, et al., 2017) or a grid of possible object centers (Zhou, Dequan Wang, and Krähenbühl, 2019; Z. Tian et al., 2019). Recent work (S. Zhang et al., 2020) demonstrate that the final performance of these systems heavily depends on the exact way these initial guesses are set. In our model we are able to remove this hand-crafted process and streamline the detection process by directly predicting the set of detections with absolute box prediction w.r.t. the input image rather than an anchor.

Set-based loss. Several object detectors (Erhan et al., 2014; W. Liu, Anguelov, Erhan, Szegedy, S. E. Reed, et al., 2016; Redmon, Divvala, et al., 2016) used the bipartite matching loss. However, in these early deep learning models, the relation between different prediction was modeled with convolutional or fully-connected layers only and a hand-designed NMS post-processing can improve their performance. More recent detectors (S. Ren et al., 2015b; T.-Y. Lin, Goyal, et al., 2017; Zhou, Dequan Wang, and Krähenbühl, 2019) use non-unique assignment rules between ground truth and predictions together with an NMS.

Learnable NMS methods (Hosang, Benenson, and Schiele, 2017; Bodla et al., 2017) and relation networks (Hu et al., 2018) explicitly model relations between different predictions with attention. Using direct set losses, they do not require any post-processing steps. However, these methods employ additional hand-crafted context features like proposal box coordinates to model relations between detections efficiently, while we look for solutions that reduce the prior knowledge encoded in the model.

Recurrent detectors. Closest to our approach are end-to-end set predictions for object detection (Stewart, Andriluka, and Ng, 2016) and instance segmentation (Romera-Paredes and Torr, 2015; Park and Berg, 2015; M. Ren and Zemel, 2017; Salvador et al., 2017). Similarly to us, they use bipartite-matching losses with encoder-decoder architectures based on CNN activations to directly produce a set of bounding boxes. These approaches, however, were only evaluated on small datasets and not against modern baselines. In particular, they are based on autoregressive models (more precisely RNNs), so they do not leverage the recent transformers with parallel decoding.

3.3 Background: transformer architecture

Since our model is based on the Transformer architecture, we present here the general form of attention mechanisms we use for completeness. The attention mechanism follows Vaswani et al., 2017, except for the details of positional encodings (see Equation 3.6) that follows Cordonnier, Loukas, and Jaggi, 2020.

3.3.1 Multi-head

The general form of *multi-head attention* with M heads of dimension d is a function with the following signature (using $d' = \frac{d}{M}$, and giving matrix/tensors sizes in underbrace)

$$\text{mh-attn} : \underbrace{X_q}_{d\times N_q}, \underbrace{X_{kv}}_{d\times N_{kv}}, \underbrace{T}_{M\times 3\times d'\times d}, \underbrace{L}_{d\times d} \mapsto \underbrace{\tilde{X}_q}_{d\times N_q} \tag{3.1}$$

where X_q is the *query sequence* of length N_q, X_{kv} is the *key-value sequence* of length N_{kv} (with the same number of channels d for simplicity of exposition), T is the weight tensor to compute the so-called query, key and value embeddings, and L is a projection matrix. The output is the same size as the query sequence. To fix the vocabulary before giving details, multi-head *self*-attention (mh-s-attn) is the special case $X_q = X_{kv}$, i.e.

$$\text{mh-s-attn}(X, T, L) = \text{mh-attn}(X, X, T, L)\,. \tag{3.2}$$

The multi-head attention is simply the concatenation of M single attention heads followed by a projection with L. The common practice (Vaswani et al., 2017) is to use residual connections, dropout and layer normalization. In other words, denoting $\tilde{X}_q = \text{mh-attn}(X_q, X_{kv}, T, L)$ and $\bar{\bar{X}}^{(q)}$ the concatenation of attention heads, we have

$$X'_q = [\text{attn}(X_q, X_{kv}, T_1); ...; \text{attn}(X_q, X_{kv}, T_M)] \tag{3.3}$$

$$\tilde{X}_q = \text{layernorm}\big(X_q + \text{dropout}(LX'_q)\big)\,, \tag{3.4}$$

where $[;]$ denotes concatenation on the channel axis.

3.3.2 Single head

An attention head with weight tensor $T' \in \mathbb{R}^{3\times d'\times d}$, denoted by $\text{attn}(X_q, X_{kv}, T')$, depends on additional positional encoding $P_q \in \mathbb{R}^{d\times N_q}$ and $P_{kv} \in \mathbb{R}^{d\times N_{kv}}$. It starts by computing so-called query, key and value embeddings after adding the query and key positional encodings (Cordonnier, Loukas, and Jaggi, 2020):

$$[Q; K; V] = [T'_1(X_q + P_q); T'_2(X_{kv} + P_{kv}); T'_3 X_{kv}] \tag{3.5}$$

where T' is the concatenation of T'_1, T'_2, T'_3. The *attention weights* α are then computed based on the softmax of dot products between queries and keys, so that each element of the query sequence attends to all elements of the key-value sequence (i is a query index and j a key-value index):

$$\alpha_{i,j} = \frac{e^{\frac{1}{\sqrt{d'}} Q_i^T K_j}}{Z_i} \quad \text{where } Z_i = \sum_{j=1}^{N_{kv}} e^{\frac{1}{\sqrt{d'}} Q_i^T K_j}\,. \tag{3.6}$$

In our case, the positional encodings may be learnt or fixed, but are shared across all attention layers for a given query/key-value sequence, so we do not explicitly write them as parameters of the attention. We give more details on their exact value when describing the encoder and the decoder. The final output is the aggregation of values weighted by attention weights: The i-th row is given by

$$\text{attn}_i(X_q, X_{kv}, T') = \sum_{j=1}^{N_{kv}} \alpha_{i,j} V_j \tag{3.7}$$

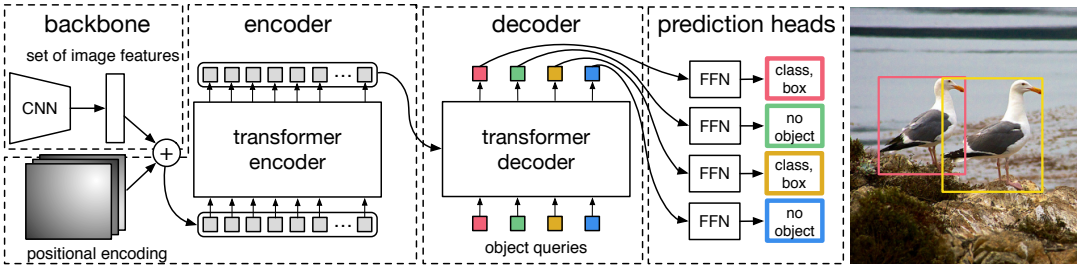


FIGURE 3.2: DETR uses a conventional CNN backbone to learn a 2D representation of an input image. The model flattens it and supplements it with a positional encoding before passing it into a transformer encoder. A transformer decoder then takes as input a small fixed number of learned positional embeddings, which we call *object queries*, and additionally attends to the encoder output. We pass each output embedding of the decoder to a shared feed forward network (FFN) that predicts either a detection (class and bounding box) or a "`no object`" class.

3.3.3 Feed-forward network (FFN) layers

The original transformer alternates multi-head attention and so-called FFN layers (Vaswani et al., 2017), which are effectively multilayer 1x1 convolutions, which have Md input and output channels in our case. The FFN we consider is composed of two-layers of 1x1 convolutions with ReLU activations. There is also a residual connection/dropout/layernorm after the two layers, similarly to (3.4).

3.4 The DETR model

Two ingredients are essential for direct set predictions in detection: (1) a set prediction loss that forces unique matching between predicted and ground truth boxes; (2) an architecture that predicts (in a single pass) a set of objects and models their relation. We describe our architecture in detail in Figure 3.2.

In our work we use a Hungarian permutation-invariant loss (Kuhn, 1955), a standard option adopted in many other set prediction problems (Vinyals, S. Bengio, and Kudlur, 2016), including detection (Hu et al., 2018; Hosang, Benenson, and Schiele, 2017). For this loss, we first find an optimal bipartite matching between predictions and the ground truth objects, according to some *matching cost*. Next, a pair-wise task loss is calculated for each of the found prediction-target matching. We describe the matching cost and the *pair-wise loss* we use in Section 3.4.1.

3.4.1 Object detection set prediction loss

DETR infers a fixed-size set of N predictions, in a single pass through the decoder, where N is set to be significantly larger than the typical number of objects in an image. One of the main difficulties of training is to score predicted objects (class, position, size) with respect to the ground truth. Our loss produces an optimal bipartite matching between predicted and ground truth objects, and then optimize object-specific (bounding box) losses.

Let us denote by y the ground truth set of objects, and $\hat{y} = \{\hat{y}_i\}_{i=1}^{N}$ the set of N predictions. Assuming N is larger than the number of objects in the image, we consider y also as a set of size N padded with $\varnothing$ (no object). To find a bipartite

matching between these two sets we search for a permutation of N elements $\sigma \in \mathfrak{S}_N$ with the lowest cost:

$$\hat{\sigma} = \arg\min_{\sigma \in \mathfrak{S}_N} \sum_{i}^{N} \mathcal{L}_{\text{match}}(y_i, \hat{y}_{\sigma(i)}), \tag{3.8}$$

where $\mathcal{L}_{\text{match}}(y_i, \hat{y}_{\sigma(i)})$ is a pair-wise *matching cost* between ground truth y_i and a prediction with index $\sigma(i)$. This optimal assignment is computed efficiently with the Hungarian algorithm, following prior work (*e.g.* (Stewart, Andriluka, and Ng, 2016)).

The matching cost takes into account both the class prediction and the similarity of predicted and ground truth boxes. Each element i of the ground truth set can be seen as a $y_i = (c_i, b_i)$ where c_i is the target class label (which may be $\varnothing$) and $b_i \in [0, 1]^4$ is a vector that defines ground truth box center coordinates and its height and width relative to the image size. For the prediction with index $\sigma(i)$ we define probability of class c_i as $\hat{p}_{\sigma(i)}(c_i)$ and the predicted box as $\hat{b}_{\sigma(i)}$. With these notations we define $\mathcal{L}_{\text{match}}(y_i, \hat{y}_{\sigma(i)})$ as $-\mathbb{1}_{\{c_i \neq \varnothing\}} \hat{p}_{\sigma(i)}(c_i) + \mathbb{1}_{\{c_i \neq \varnothing\}} \mathcal{L}_{\text{box}}(b_i, \hat{b}_{\sigma(i)})$.

This procedure of finding the matching plays the same role as the heuristic assignment rules used to match proposal (S. Ren et al., 2015b) or anchors (T.-Y. Lin, Dollár, et al., 2017) to ground truth objects in modern detectors. The main difference is that we need to find one-to-one matching for direct set prediction without duplicates.

The second step is to compute the loss function, the *Hungarian loss* for all pairs matched in the previous step. We define the loss similarly to the losses of common object detectors, *i.e.* a linear combination of a negative log-likelihood for class prediction and a box loss $\mathcal{L}_{\text{box}}(\cdot, \cdot)$ defined later:

$$\mathcal{L}_{\text{Hungarian}}(y, \hat{y}) = \sum_{i=1}^{N} \left[-\log \hat{p}_{\hat{\sigma}(i)}(c_i) + \mathbb{1}_{\{c_i \neq \varnothing\}} \mathcal{L}_{\text{box}}(b_i, \hat{b}_{\hat{\sigma}}(i)) \right], \tag{3.9}$$

where $\hat{\sigma}$ is the optimal assignment computed in the first step (3.8). In practice, we down-weight the log-probability term when $c_i = \varnothing$ by a factor 10 to account for class imbalance. This is analogous to how Faster R-CNN training procedure balances positive/negative proposals by subsampling (S. Ren et al., 2015b). Notice that the matching cost between an object and $\varnothing$ doesn't depend on the prediction, which means that in that case the cost is a constant. In the matching cost we use probabilities $\hat{p}_{\hat{\sigma}(i)}(c_i)$ instead of log-probabilities. This makes the class prediction term commensurable to $\mathcal{L}_{\text{box}}(\cdot, \cdot)$, and we observed better empirical performances.

Bounding box loss

The second part of the matching cost and the Hungarian loss is $\mathcal{L}_{\text{box}}(\cdot)$ that scores the bounding boxes. Unlike many detectors that do box predictions as a Δ w.r.t. some initial guesses, we make box predictions directly. While such approach simplify the implementation it poses an issue with relative scaling of the loss. The most commonly-used ℓ_1 loss will have different scales for small and large boxes even if their relative errors are similar. To mitigate this issue we use a linear combination of the ℓ_1 loss and the generalized IoU loss (H. Rezatofighi et al., 2019) $\mathcal{L}_{\text{iou}}(\cdot, \cdot)$ that is scale-invariant:

$$\mathcal{L}_{\text{iou}}(b_{\sigma(i)}, \hat{b}_i) = 1 - \left(\frac{|b_{\sigma(i)} \cap \hat{b}_i|}{|b_{\sigma(i)} \cup \hat{b}_i|} - \frac{|B(b_{\sigma(i)}, \hat{b}_i) \setminus b_{\sigma(i)} \cup \hat{b}_i|}{|B(b_{\sigma(i)}, \hat{b}_i)|} \right). \tag{3.10}$$

$|.|$ means "area", and the union and intersection of box coordinates are used as shorthands for the boxes themselves. The areas of unions or intersections are computed by

min / max of the linear functions of $b_{\sigma(i)}$ and $\hat{b}_i$, which makes the loss sufficiently well-behaved for stochastic gradients. $B(b_{\sigma(i)}, \hat{b}_i)$ means the largest box containing $b_{\sigma(i)}, \hat{b}_i$ (the areas involving B are also computed based on min / max of linear functions of the box coordinates).

Overall, similarly to Romera-Paredes and Torr, 2015; M. Ren and Zemel, 2017, we linearly combine these two losses:

$$\mathcal{L}_{\text{box}}(b_{\sigma(i)}, \hat{b}_i) = \lambda_{\text{iou}}\mathcal{L}_{\text{iou}}(b_{\sigma(i)}, \hat{b}_i) + \lambda_{\text{L1}}||b_{\sigma(i)} - \hat{b}_i||_1 \,, \tag{3.11}$$

where $\lambda_{\text{iou}}, \lambda_{\text{L1}} \in \mathbb{R}$ are hyperparameters. These two losses are normalized by the number of objects inside the batch. Extra care must be taken for distributed training: since each GPU receives a sub-batch, with a varying number of objects, it is not sufficient to normalize by the number of objects in the local batch, since in general the sub-batches are not balanced across GPUs. Instead, it is important to normalize by the total number of objects in all sub-batches. This avoids biasing the loss towards images with few objects, that tend to be large, salient objects, that are anyway well detected by the model.

Auxiliary decoding losses

We found helpful to use auxiliary losses (Al-Rfou et al., 2019) in the decoder during training, especially to help the model output the correct number of objects of each class. The output of each decoder layer is normalized with a shared layer-norm then fed to the shared prediction heads (classification and box prediction). We then apply the Hungarian loss as usual for supervision.

3.4.2 DETR architecture

The overall DETR architecture is surprisingly simple and depicted in Figure 3.2. It contains three main components, which we describe below: a CNN backbone to extract a compact feature representation, an encoder-decoder transformer, and a simple feed forward network (FFN) that makes the final detection prediction.

Unlike many modern detectors, DETR can be implemented in any deep learning framework that provides a common CNN backbone and a transformer architecture implementation with just a few hundred lines. Inference code for DETR can be implemented in less than 50 lines in PyTorch (Paszke et al., 2019). We hope that the simplicity of our method will attract new researchers to the detection community.

Backbone

Starting from the initial image $x_{\text{img}} \in \mathbb{R}^{3\times H_0\times W_0}$ (with 3 color channels[1]), a conventional CNN backbone generates a lower-resolution activation map $f \in \mathbb{R}^{C\times H\times W}$. Typical values we use are $C = 2048$ and $H, W = \frac{H_0}{32}, \frac{W_0}{32}$.

Transformer encoder

Our transformer architecture is illustrated in Fig.3.3. First, a 1x1 convolution reduces the channel dimension of the high-level activation map f from C to a smaller dimension d. creating a new feature map $z_0 \in \mathbb{R}^{d\times H\times W}$. The encoder expects a sequence as

[1]The input images are batched together, applying 0-padding adequately to ensure they all have the same dimensions (H_0, W_0) as the largest image of the batch.

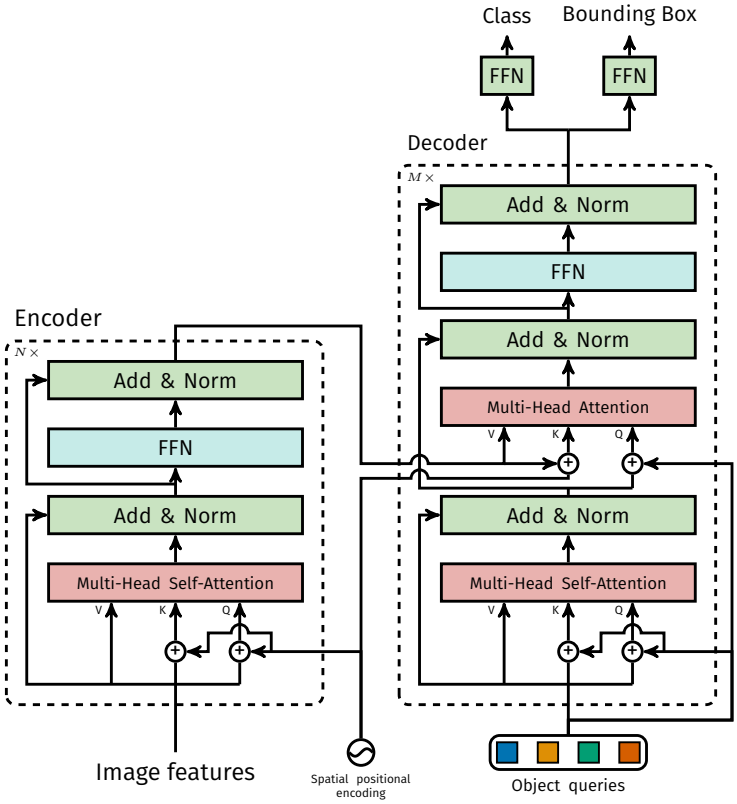


FIGURE 3.3: Architecture of DETR's transformer.

input, hence we collapse the spatial dimensions of z_0 into one dimension, resulting in a $d \times HW$ feature map. Each encoder layer has a standard architecture and consists of a multi-head self-attention module and a feed forward network (FFN). Since the transformer architecture is permutation-invariant, we supplement it with fixed positional encodings (Parmar et al., 2018; Bello et al., 2019) that are added to the input of each attention layer. We defer to the supplementary material the detailed definition of the architecture, which follows the one described in (Vaswani et al., 2017).

Transformer decoder

The decoder follows the standard architecture of the transformer, transforming N embeddings of size d using multi-headed self- and encoder-decoder attention mechanisms. The difference with the original transformer is that our model decodes the N objects in parallel at each decoder layer, while Vaswani et al. (Vaswani et al., 2017) use an autoregressive model that predicts the output sequence one element at a time. We refer the reader unfamiliar with the concepts to the supplementary material. Since the decoder is also permutation-invariant, the N input embeddings must be different to produce different results. These input embeddings are learnt positional encodings that we refer to as *object queries*, and similarly to the encoder, we add them to the input of each attention layer. The N object queries are transformed into an output embedding by the decoder. They are then *independently* decoded into box coordinates and class labels by a feed forward network (described in the next subsection), resulting N final predictions. Using self- and encoder-decoder attention over these embeddings, the model globally reasons about all objects together using pair-wise relations between them, while being able to use the whole image as context.

Prediction feed-forward networks (FFNs)

The final prediction is computed by a 3-layer perceptron with ReLU activation function and hidden dimension d, and a linear projection layer. The FFN predicts the normalized center coordinates, height and width of the box w.r.t. the input image, and the linear layer predicts the class label using a softmax function. Since we predict a fixed-size set of N bounding boxes, where N is usually much larger than the actual number of objects of interest in an image, an additional special class label $\varnothing$ is used to represent that no object is detected within a slot. This class plays a similar role to the "background" class in standard object detection approaches.

Spatial positional encoding

Encoder activations are associated with corresponding spatial positions of image features. In our model we use a fixed absolute encoding to represent these spatial positions. We adopt a generalization of the original Transformer (Vaswani et al., 2017) encoding to the 2D case (Parmar et al., 2018). Specifically, for both spatial coordinates of each embedding we independently use $\frac{d}{2}$ sine and cosine functions with different frequencies. We then concatenate them to get the final d channel positional encoding.

Computational complexity

Every self-attention in the encoder has complexity $\mathcal{O}(d^2HW + d(HW)^2)$: $\mathcal{O}(d'd)$ is the cost of computing a single query/key/value embeddings (and $Md' = d$), while $\mathcal{O}(d'(HW)^2)$ is the cost of computing the attention weights for one head. Other computations are negligible. In the decoder, each self-attention is in $\mathcal{O}(d^2N + dN^2)$, and cross-attention between encoder and decoder is in $\mathcal{O}(d^2(N + HW) + dNHW)$, which is much lower than the encoder since $N \ll HW$ in practice.

3.5 Experiments

We show that DETR achieves competitive results compared to Faster R-CNN (S. Ren et al., 2015b) and RetinaNet (T.-Y. Lin, Goyal, et al., 2017) in quantitative evaluation on COCO. Then, we provide a detailed ablation study of the architecture and loss, with insights and qualitative results. Finally, to show that DETR is a versatile model, we present results on panoptic segmentation, training only a small extension on a fixed DETR model.

Dataset

We perform experiments on COCO 2017 detection and panoptic segmentation datasets (T.-Y. Lin, Maire, et al., 2014; Kirillov, Ross B. Girshick, et al., 2019), containing 118k training images and 5k validation images. Each image is annotated with bounding boxes and panoptic segmentation. There are 7 instances per image on average, up to 63 instances in a single image in training set, ranging from small to large on the same images. If not specified, we report AP as bbox AP, the integral metric over multiple thresholds. For comparison with other models we report validation AP at the last training epoch, and in ablations we report the median over the last 10 epochs.

TABLE 3.1: Comparison with RetinaNet and Faster R-CNN with a ResNet-50 and ResNet-101 backbones on the COCO validation set. The top section shows results for models in Detectron2 Wu et al., 2019, the middle section shows results for models with GIoU H. Rezatofighi et al., 2019, random crops train-time augmentation, and the long 9x training schedule. DETR models achieve comparable results to heavily tuned Faster R-CNN baselines, having lower AP_S but greatly improved AP_L. We use torchscript models to measure FLOPS and FPS. Results without R101 in the name correspond to ResNet-50.

Model	GFLOPS/FPS	#params	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
RetinaNet	205/18	38M	38.7	58.0	41.5	23.3	42.3	50.3
Faster RCNN-DC5	320/16	166M	39.0	60.5	42.3	21.4	43.5	52.5
Faster RCNN-FPN	180/26	42M	40.2	61.0	43.8	24.2	43.5	52.0
Faster RCNN-R101-FPN	246/20	60M	42.0	62.5	45.9	25.2	45.6	54.6
RetinaNet+	205/18	38M	41.1	60.4	43.7	25.6	44.8	53.6
Faster RCNN-DC5+	320/16	166M	41.1	61.4	44.3	22.9	45.9	55.0
Faster RCNN-FPN+	180/26	42M	42.0	62.1	45.5	26.6	45.4	53.4
Faster RCNN-R101-FPN+	246/20	60M	44.0	63.9	**47.8**	**27.2**	48.1	56.0
DETR	86/28	41M	42.0	62.4	44.2	20.5	45.8	61.1
DETR-DC5	187/12	41M	43.3	63.1	45.9	22.5	47.3	61.1
DETR-R101	152/20	60M	43.5	63.8	46.4	21.9	48.0	61.8
DETR-DC5-R101	253/10	60M	**44.9**	**64.7**	47.7	23.7	**49.5**	**62.3**

Technical details

We train DETR with AdamW (Loshchilov and Hutter, 2019) setting the initial transformer's learning rate to 10^{-4}, the backbone's to 10^{-5}, and weight decay to 10^{-4}. We also apply gradient clipping, with a maximal gradient norm of 0.1. All transformer weights are initialized with Xavier init (Glorot and Y. Bengio, 2010), and the backbone is with ImageNet-pretrained ResNet model (He, X. Zhang, et al., 2016) from TORCHVISION with frozen batchnorm layers. We observe that having the backbone learning rate roughly an order of magnitude smaller than the rest of the network is important to stabilize training, especially in the first few epochs. In the transformer, we also use additive dropout of 0.1 after every multi-head attention and FFN before layer normalization. For training, we use a linear combination of ℓ_1 and GIoU losses for bounding box regression with $\lambda_{L1} = 5$ and $\lambda_{iou} = 2$ weights respectively. All models were trained with $N = 100$ decoder query slots.

We report results with two different backbones: a ResNet-50 and a ResNet-101. The corresponding models are called respectively DETR and DETR-R101. Following (Li et al., 2017), we also increase the feature resolution by adding a dilation to the last stage of the backbone and removing a stride from the first convolution of this stage. The corresponding models are called respectively DETR-DC5 and DETR-DC5-R101 (dilated C5 stage). This modification increases the resolution by a factor of two, thus improving performance for small objects, at the cost of a 16x higher cost in the self-attentions of the encoder, leading to an overall 2x increase in computational cost. A full comparison of FLOPs of these models, Faster R-CNN and RetinaNet is given in Table 3.1.

We use scale augmentation, resizing the input images such that the shortest side is at least 480 and at most 800 pixels while the longest at most 1333 (Wu et al., 2019). To help learning global relationships through the self-attention of the encoder, we also apply random crop augmentations during training, improving the performance by approximately 1 AP. Specifically, a train image is cropped with probability 0.5 to a

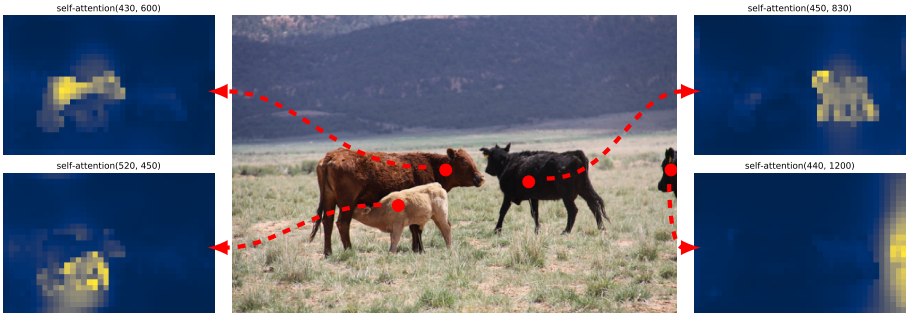


FIGURE 3.4: Encoder self-attention for a set of reference points. The encoder is able to separate individual instances. Prediction made with baseline DETR on a validation image.

random rectangular patch which is then resized again to 800-1333. The transformer is trained with default dropout of 0.1. At inference time, some slots predict empty class. To optimize for AP, we override the prediction of these slots with the second highest scoring class, using the corresponding confidence. This improves AP by 2 points compared to filtering out empty slots. For our ablation experiments we use training schedule of 300 epochs with a learning rate drop by a factor of 10 after 200 epochs, where a single epoch is a pass over all training images once. Training the baseline model for 300 epochs on 16 V100 GPUs takes 3 days, with 4 images per GPU (hence a total batch size of 64). For the longer schedule used to compare with Faster R-CNN we train for 500 epochs with learning rate drop after 400 epochs, which improves AP by 1.5 points.

3.5.1 Comparison with Faster R-CNN and RetinaNet

Transformers are typically trained with Adam or Adagrad optimizers with very long training schedules and dropout, and this is true for DETR as well. Faster R-CNN, however, is trained with SGD with minimal data augmentation and we are not aware of successful applications of Adam or dropout. Despite these differences we attempt to make our baselines stronger. To align it with DETR, we add generalized IoU (H. Rezatofighi et al., 2019) to the box loss, the same random crop augmentation and long training known to improve results (He, Ross B. Girshick, and Dollár, 2019). We performed a grid search to find the best weights for the losses and the final models use only GIoU loss with weights 20 and 1 for box and proposal regression tasks respectively. Results are presented in Table 3.1. In the top section we show results from Detectron2 Model Zoo (Wu et al., 2019) for models trained with the 3x schedule. In the middle section we show results (with a "+") for the same models but trained with the 9x schedule (109 epochs) and the described enhancements, which in total adds 1-2 AP. In the last section of Table 3.1 we show the results for multiple DETR models. To be comparable in the number of parameters we choose a model with 6 transformer and 6 decoder layers of width 256 with 8 attention heads. Like Faster R-CNN with FPN this model has 41.3M parameters, out of which 23.5M are in ResNet-50, and 17.8M are in the transformer. Even though both Faster R-CNN and DETR are still likely to further improve with longer training, we can conclude that DETR can be competitive with Faster R-CNN with the same number of parameters, achieving 42 AP on the COCO val subset. The way DETR achieves this is by improving AP_L (+7.8), however note that the model is still lagging behind in AP_S (-5.5). DETR-DC5 with the same number of parameters and similar FLOP count has higher AP, but is

TABLE 3.2: Effect of encoder size. Each row corresponds to a model with varied number of encoder layers and fixed number of decoder layers. Performance gradually improves with more encoder layers.

#layers	GFLOPS/FPS	#params	AP	AP_{50}	AP_S	AP_M	AP_L
0	76/28	33.4M	36.7	57.4	16.8	39.6	54.2
3	81/25	37.4M	40.1	60.6	18.5	43.8	58.6
6	86/23	41.3M	40.6	61.6	19.9	44.3	60.2
12	95/20	49.2M	41.6	62.1	19.8	44.9	61.9

still significantly behind in AP_S too. Results on ResNet-101 backbone are comparable as well.

FLOPS computation

Given that the FLOPS for Faster R-CNN depends on the number of proposals in the image, we report the average number of FLOPS for the first 100 images in the COCO 2017 validation set. We compute the FLOPS with the tool `flop_count_operators` from Detectron2 Wu et al., 2019. We use it without modifications for Detectron2 models, and extend it to take batch matrix multiply (`bmm`) into account for DETR models.

3.5.2 Ablations

Attention mechanisms in the transformer decoder are the key components which model relations between feature representations of different detections. In our ablation analysis, we explore how other components of our architecture and loss influence the final performance. For the study we choose ResNet-50-based DETR model with 6 encoder, 6 decoder layers and width 256. The model has 41.3M parameters, achieves 40.6 and 42.0 AP on short and long schedules respectively, and runs at 28 FPS, similarly to Faster R-CNN-FPN with the same backbone.

Number of encoder layers

We evaluate the importance of global image-level self-attention by changing the number of encoder layers, and summarize findings in Table 3.2). Without encoder layers, overall AP drops by 3.9 points, with a more significant drop of 6.0 AP on large objects. We hypothesize that, by using global scene reasoning, the encoder is important for disentangling objects. In Figure 3.4, we visualize the attention maps of the last encoder layer of a trained model, focusing on a few points in the image. The encoder seems to separate instances already, which likely simplifies object extraction and localization for the decoder.

Number of decoder layers

We apply auxiliary losses after each decoding layer (see Section 3.4.1), hence, the prediction FFNs are trained by design to predict objects out of the outputs of every decoder layer. We analyze the importance of each decoder layer by evaluating the objects that would be predicted at each stage of the decoding (Fig. 3.5). Both AP and AP_{50} improve after every layer, totalling into a very significant +8.2/9.5 AP improvement between the first and the last layer. With its set-based loss, DETR does

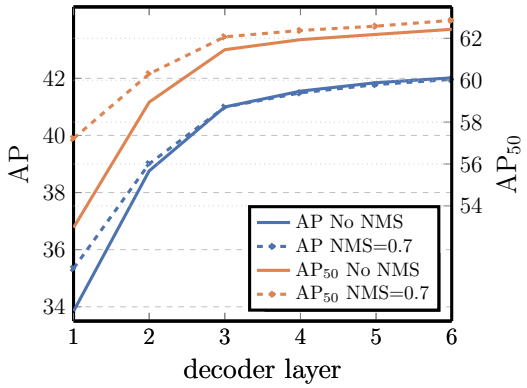


FIGURE 3.5: AP and AP_{50} performance after each decoder layer in a long schedule baseline model. DETR does not need NMS by design, which is validated by this figure. NMS lowers AP in the final layers, removing TP predictions, but improves it in the first layers, where DETR does not have the capability to remove double predictions.

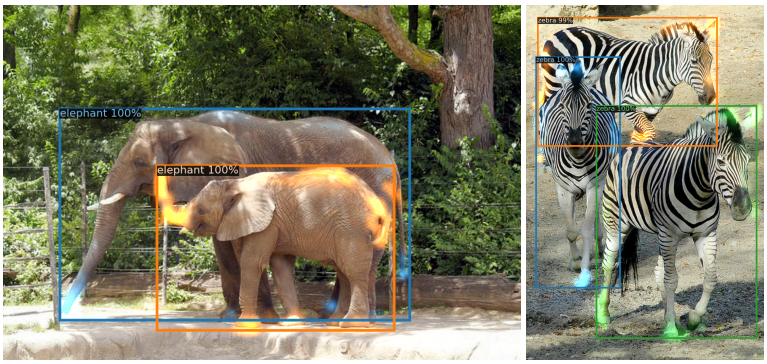


FIGURE 3.6: Visualizing decoder attention for every predicted object (images from COCO `val` set). Predictions are made with DETR-DC5 model. Decoder typically attends to object extremities, such as legs and heads.

not need NMS by design. To verify this we run a standard NMS procedure with default parameters (Wu et al., 2019) for the outputs after each decoder. NMS improves performance for the predictions from the first decoder. This can be explained by the fact that a single decoding layer of the transformer is not able to compute any cross-correlations between the output elements, and thus it is prone to making multiple predictions for the same object. In the second and subsequent layers, the self-attention mechanism over the activations allows the model to inhibit duplicate predictions. We observe that the improvement brought by NMS diminishes as depth increases. It hurts AP in the last layers, as it incorrectly removes true positive predictions.

Similarly to the visualization of the attention in the encoder, we visualize decoder attentions in Fig. 3.6, coloring attention maps for each predicted object in different colors. We observe that the attention in the decoder fairly local, meaning that it mostly attends to object extremities such as heads or legs. We hypothesise that after the encoder has separated instances via global attention, the decoder only needs to attend to the extremities to extract the class and object boundaries.

TABLE 3.3: Effect of loss components on AP. We train two models turning off ℓ_1 loss, and GIoU loss, and observe that ℓ_1 gives poor results on its own, but when combined with GIoU improves AP_M and AP_L. Our baseline (last row) combines both losses.

class	ℓ_1	GIoU	AP	Δ	AP_{50}	Δ	AP_S	AP_M	AP_L
✓	✓		35.8	-4.8	57.3	-4.4	13.7	39.8	57.9
✓		✓	39.9	-0.7	**61.6**	0	**19.9**	43.2	57.9
✓	✓	✓	**40.6**	-	**61.6**	-	**19.9**	**44.3**	**60.2**

TABLE 3.4: Results for different positional encodings compared to the baseline (last row), which has fixed sine pos. encodings passed at every attention layer in both the encoder and the decoder. Learned embeddings are shared between all layers. Not using spatial positional encodings leads to a significant drop in AP. Interestingly, passing them in decoder only leads to a minor AP drop. All these models use learned output positional encodings.

spatial pos. enc.		output pos. enc.				
encoder	decoder	decoder	AP	Δ	AP_{50}	Δ
none	none	learned at input	32.8	-7.8	55.2	-6.5
sine at input	sine at input	learned at input	39.2	-1.4	60.0	-1.6
learned at attn.	learned at attn.	learned at attn.	39.6	-1.0	60.7	-0.9
none	sine at attn.	learned at attn.	39.3	-1.3	60.3	-1.4
sine at attn.	sine at attn.	learned at attn.	**40.6**	-	**61.6**	-

Loss ablations

To evaluate the importance of different components of the matching cost and the loss, we train several models turning them on and off. There are three components to the loss: classification loss, ℓ_1 bounding box distance loss, and GIoU H. Rezatofighi et al., 2019 loss. The classification loss is essential for training and cannot be turned off, so we train a model without bounding box distance loss, and a model without the GIoU loss, and compare with baseline, trained with all three losses. Results are presented in Table 3.3. GIoU loss on its own accounts for most of the model performance, losing only 0.7 AP to the baseline with combined losses. Using ℓ_1 without GIoU shows poor results. We only studied simple ablations of different losses (using the same weighting every time), but other means of combining them may achieve different results.

Importance of FFN

FFN inside tranformers can be seen as 1×1 convolutional layers, making encoder similar to attention augmented convolutional networks (Bello et al., 2019). We attempt to remove it completely leaving only attention in the transformer layers. By reducing the number of network parameters from 41.3M to 28.7M, leaving only 10.8M in the transformer, performance drops by 2.3 AP, we thus conclude that FFN are important for achieving good results.

Importance of positional encodings

There are two kinds of positional encodings in our model: spatial positional encodings and output positional encodings (object queries). We experiment with various

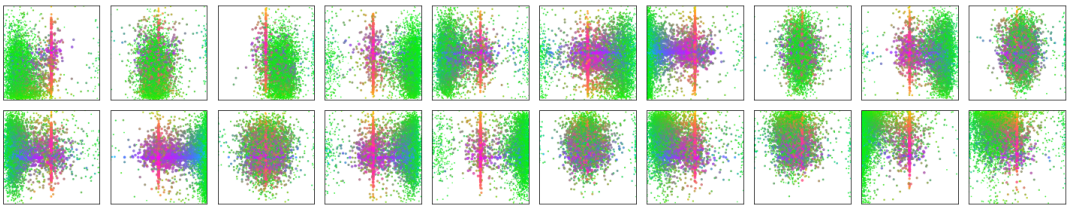


FIGURE 3.7: Visualization of all box predictions on all images from COCO 2017 val set for 20 out of total $N = 100$ prediction slots in DETR decoder. Each box prediction is represented as a point with the coordinates of its center in the 1-by-1 square normalized by each image size. The points are color-coded so that green color corresponds to small boxes, red to large horizontal boxes and blue to large vertical boxes. We observe that each slot learns to specialize on certain areas and box sizes with several operating modes. We note that almost all slots have a mode of predicting large image-wide boxes that are common in COCO dataset.

combinations of fixed and learned encodings, see results in appendix. Output positional encodings are required and cannot be removed, so we experiment with either passing them once at decoder input or adding to queries at every decoder attention layer. In the first experiment we completely remove spatial positional encodings and pass output positional encodings at input and, interestingly, the model still achieves more than 32 AP, losing 7.8 AP to the baseline. Then, we pass fixed sine spatial positional encodings and the output encodings at input once, as in the original transformer (Vaswani et al., 2017), and find that this leads to 1.4 AP drop compared to passing the positional encodings directly in attention. Learned spatial encodings passed to the attentions give similar results. Surprisingly, we find that not passing any spatial encodings in the encoder only leads to a minor AP drop of 1.3 AP. When we pass the encodings to the attentions, they are shared across all layers, and the output encodings (object queries) are always learned.

Given these ablations, we conclude that transformer components: the global self-attention in encoder, FFN, multiple decoder layers, and positional encodings, all significantly contribute to the final object detection performance.

3.5.3 Analysis

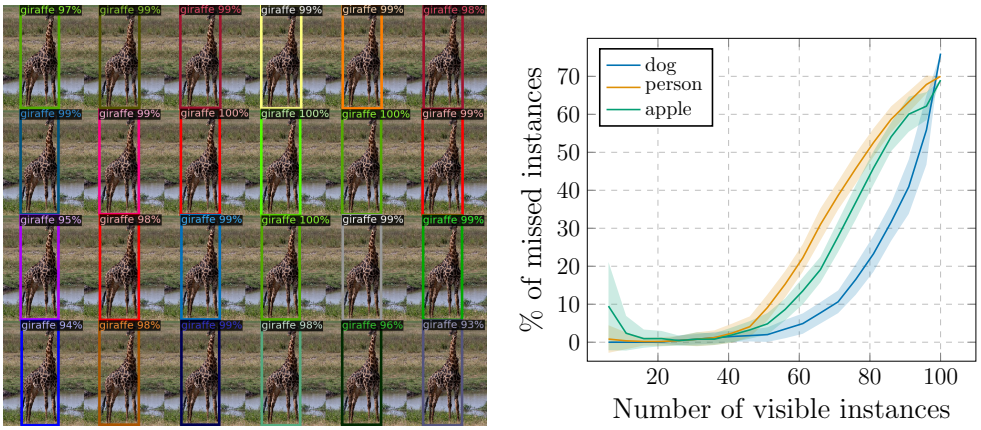
In this section, we probe the model more in depth, to gain an understanding of its inner workings and its limits.

Decoder output slot analysis

In Fig. 3.7 we visualize the boxes predicted by different slots for all images in COCO 2017 val set. DETR learns different specialization for each query slot. We observe that each slot has several modes of operation focusing on different areas and box sizes. In particular, all slots have the mode for predicting image-wide boxes (visible as the red dots aligned in the middle of the plot). We hypothesize that this is related to the distribution of objects in COCO.

Generalization to unseen numbers of instances

Some classes in COCO are not well represented with many instances of the same class in the same image. For example, there is no image with more than 13 giraffes



(A) Out of distribution generalization for rare classes (B) Fraction of missed objects as a function of the number of instances.

FIGURE 3.8: In (a), we study the generalization to higher number of instances. Even though no image in the training set has more than 13 giraffes, DETR has no difficulty generalizing to 24 and more instances.In (b), we analyze the number of instances of various classes missed by DETR depending on how many are present in the image We report the mean and the standard deviation. As the number of instances gets close to 100, DETR starts saturating and misses more and more objects.

in the training set. We create a synthetic image[2] to verify the generalization ability of DETR (see Figure 3.8a). Our model is able to find all 24 giraffes on the image which is clearly out of distribution. This experiment confirms that there is no strong class-specialization in each object query.

By design, DETR cannot predict more objects than it has query slots, i.e. 100 in our experiments. We further analyze the behavior of DETR when approaching this limit. We select a canonical square image of a given class, repeat it on a 10×10 grid, and compute the percentage of instances that are missed by the model. To test the model with less than 100 instances, we randomly mask some of the cells. This ensures that the absolute size of the objects is the same no matter how many are visible. To account for the randomness in the masking, we repeat the experiment 100 times with different masks. The results are shown in Fig.3.8b. The behavior is similar across classes, and while the model detects all instances when up to 50 are visible, it then starts saturating and misses more and more instances. Notably, when the image contains all 100 instances, the model only detects 30 on average, which is less than if the image contains only 50 instances that are all detected. The counter-intuitive behavior of the model is likely because the images and the detections are far from the training distribution. Indeed, there are very few example images with a lot of instances of a single class. However, we note that it is difficult to disentangle, from this experiment, two types of out-of-domain generalization: the aspect of the image itself (a synthetic grid, with masked-out, black cells), and the number of instances visible. But since COCO doesn't contain images with many instances of a single class, this type of experiment represents our best effort to understand whether query objects overfit the label and position distribution of the dataset. Overall, the experiments suggests that the model does not overfit on these distributions since it yields near-perfect detections up to 50 objects.

[2]Base picture credit: https://www.piqsels.com/en/public-domain-photo-jzlwu

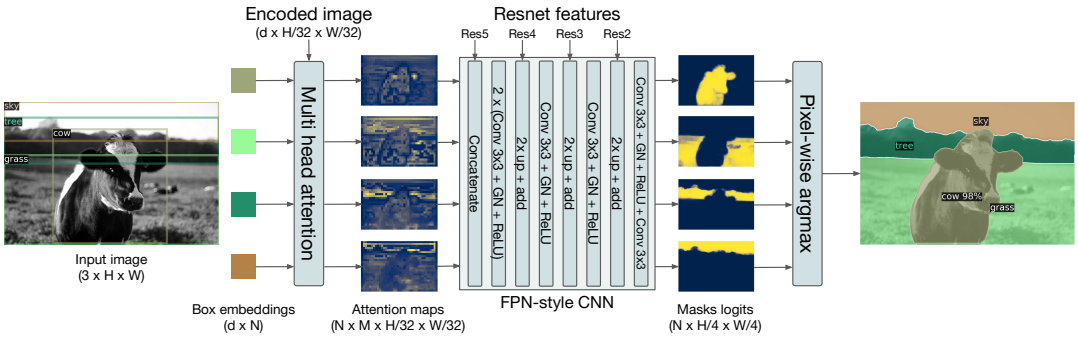


FIGURE 3.9: Illustration of the panoptic head. A binary mask is generated in parallel for each detected object, then the masks are merged using pixel-wise argmax.

3.5.4 DETR for panoptic segmentation

Panoptic segmentation (Kirillov, He, et al., 2019) has recently attracted a lot of attention from the computer vision community. Similarly to the extension of Faster R-CNN (S. Ren et al., 2015b) to Mask R-CNN (He, Gkioxari, et al., 2017), DETR can be naturally extended by adding a mask head on top of the decoder outputs. In this section we demonstrate that such a head can be used to produce panoptic segmentation (Kirillov, He, et al., 2019) by treating stuff and thing classes in a unified way. We perform our experiments on the panoptic annotations of the COCO dataset that has 53 stuff categories in addition to 80 things categories.

We train DETR to predict boxes around both *stuff* and *things* classes on COCO, using the same recipe. Predicting boxes is required for the training to be possible, since the Hungarian matching is computed using distances between boxes. We also add a mask head which predicts a binary mask for each of the predicted boxes, see Figure 3.9. It takes as input the output of transformer decoder for each object and computes multi-head (with M heads) attention scores of this embedding over the output of the encoder, generating M attention heatmaps per object in a small resolution. To make the final prediction and increase the resolution, an FPN-like architecture is used. We refer to the supplement for more details. The final resolution of the masks has stride 4 and each mask is supervised independently using the DICE/F-1 loss (Milletari, Navab, and Ahmadi, 2016) and Focal loss (T.-Y. Lin, Goyal, et al., 2017).

The mask head can be trained either jointly, or in a two steps process, where we train DETR for boxes only, then freeze all the weights and train only the mask head for 25 epochs. Experimentally, these two approaches give similar results, we report results using the latter method since it is less computationally intensive.

To predict the final panoptic segmentation we simply use an argmax over the mask scores at each pixel, and assign the corresponding categories to the resulting masks. This procedure guarantees that the final masks have no overlaps and thus DETR does not require a heuristic (Kirillov, He, et al., 2019) to align different masks.

Training details

We train DETR, DETR-DC5 and DETR-R101 models following the recipe for bounding box detection to predict boxes around stuff and things classes in COCO dataset. The new mask head is trained for 25 epochs (see supplementary for details). During inference we first filter out the detection with a confidence below 85%, then compute

TABLE 3.5: Comparison with the state-of-the-art methods UPSNet (Xiong et al., 2019) and Panoptic FPN (Kirillov, Ross B. Girshick, et al., 2019) on the COCO `val` dataset We retrained PanopticFPN with the same data-augmentation as DETR, on a 18x schedule for fair comparison. UPSNet uses the `1x` schedule, UPSNet-M is the version with multiscale test-time augmentations.

Model	Backbone	PQ	SQ	RQ	PQ^{th}	SQ^{th}	RQ^{th}	PQ^{st}	SQ^{st}	RQ^{st}	AP
PanopticFPN+	R50	42.4	79.3	51.6	49.2	82.4	58.8	32.3	74.8	40.6	37.7
UPSnet	R50	42.5	78.0	52.5	48.6	79.4	59.6	33.4	75.9	41.7	34.3
UPSnet-M	R50	43.0	79.1	52.8	48.9	79.7	59.7	34.1	78.2	42.3	34.3
PanopticFPN+	R101	44.1	79.5	53.3	**51.0**	**83.2**	60.6	33.6	74.0	42.1	**39.7**
DETR	R50	43.4	79.3	53.8	48.2	79.8	59.5	36.3	78.5	45.3	31.1
DETR-DC5	R50	44.6	79.8	55.0	49.4	80.5	60.6	37.3	**78.7**	46.5	31.9
DETR	R101	45.1	79.9	55.5	50.5	80.9	61.7	37.0	78.5	46.0	33.0
DETR-DC5	R101	**45.6**	**80.0**	**56.1**	50.9	80.9	**62.2**	**37.5**	78.6	**46.8**	33.1

the per-pixel argmax to determine in which mask each pixel belongs. We then collapse different mask predictions of the same stuff category in one, and filter the empty ones (less than 4 pixels).

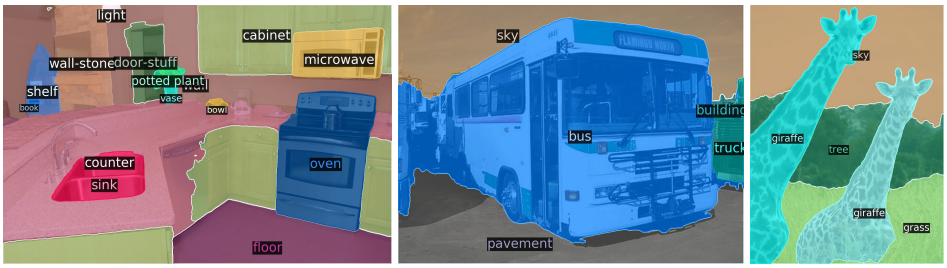
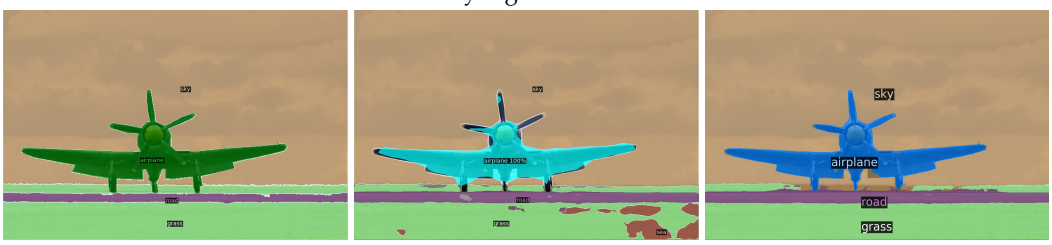


FIGURE 3.10: Qualitative results for panoptic segmentation generated by DETR-R101. DETR produces aligned mask predictions in a unified manner for things and stuff. Images are taken from COCO `val` set (ids 969, 2427, 3499).

Main results

Qualitative results are shown in Figure 3.10 and Figure 3.11. In table 3.5 we compare our unified panoptic segmenation approach with several established methods that treat things and stuff differently. We report the Panoptic Quality (PQ) and the breakdown on things (PQ^{th}) and stuff (PQ^{st}). We also report the mask AP (computed on the things classes), before any panoptic post-treatment (in our case, before taking the pixel-wise argmax). We show that DETR outperforms published results on COCO-val 2017, as well as our strong PanopticFPN baseline (trained with same data-augmentation as DETR, for fair comparison). The result break-down shows that DETR is especially dominant on stuff classes, and we hypothesize that the global reasoning allowed by the encoder attention is the key element to this result. For things class, despite a severe deficit of up to 8 mAP compared to the baselines on the mask AP computation, DETR obtains competitive PQ^{th}. We also evaluated our method on the test set of the COCO dataset, and obtained 46 PQ. We hope that our approach will inspire the exploration of fully unified models for panoptic segmentation in future work.

(A) Failure case with overlapping objects. PanopticFPN misses one plane entirely, while DETR fails to accurately segment 3 of them.



(B) `Things` masks are predicted at full resolution, which allows sharper boundaries than PanopticFPN

FIGURE 3.11: Comparison of panoptic predictions. From left to right: Ground truth, PanopticFPN with ResNet 101, DETR with ResNet 101

3.6 Conclusion

We presented DETR, a new design for object detection systems based on transformers and bipartite matching loss for direct set prediction. The approach achieves comparable results to an optimized Faster R-CNN baseline on the challenging COCO dataset. DETR is straightforward to implement and has a flexible architecture that is easily extensible to panoptic segmentation, with competitive results. In addition, it achieves significantly better performance on large objects, likely due to the processing of global information performed by the self-attention.

This new design for detectors also comes with new challenges, in particular regarding training, optimization and performances on small objects. Current detectors required several years of improvements to cope with similar issues, and we expect future work to successfully address them for DETR.

Chapter 4

Conclusion

In this thesis, we have made two main contributions. In the first one, we explored multi-agent collaborative problems, and we showed that borrowing some tools from the operational research community, in the form of centralized optimization problems, and marrying them with the adequate neural networks allow to get the best of both worlds and obtain a class of solution that can structurally generalize better than their fully neural counter-parts. This paves the way for exciting real world applications, from teams of robots playing soccer to fleets of quadcopters coordinating their flight patterns. In the future, an exciting avenue of research is enable this seamless collaboration even when the other agents are unknown, and especially if they are human.

The second contributions explored the opportunity to develop object-centric models for object detection, thus breaking from mainstream approaches. It showed that with little hand-crafted knowledge, we can indeed achieve competitive detection performance, using readily available building blocks such as Resnets and Transformers. Beyond detection, an object-centric model such as DETR could enable exciting new developments in future works, by providing an end-to-end pipeline to all the downstream tasks that require reasoning about objects, such as object tracking, keypoint estimation, forward prediction. Its similarity with popular speech detection models as well as neural language processing models could allow bridging the gap between these modalities, bringing us closer to a model that can jointly reason about all of them at once.

Appendix A

DETR minimal inference code in Pytorch

To demonstrate the simplicity of the approach, we include inference code with PyTorch and Torchvision libraries in Listing 1. The code runs with Python 3.6+, PyTorch 1.4 and Torchvision 0.5. Note that it does not support batching, hence it is suitable only for inference or training with DistributedDataParallel with one image per GPU. Also note that for clarity, this code uses learnt positional encodings in the encoder instead of fixed, and positional encodings are added to the input only instead of at each transformer layer. Making these changes requires going beyond PyTorch implementation of transformers, which hampers readability. Refer to the official source code for the full implementation.

```python
import torch
from torch import nn
from torchvision.models import resnet50

class DETR(nn.Module):

    def __init__(self, num_classes, hidden_dim, nheads,
                 num_encoder_layers, num_decoder_layers):
        super().__init__()
        # We take only convolutional layers from ResNet-50 model
        self.backbone = nn.Sequential(*list(resnet50(pretrained=True).children())[:-2])
        self.conv = nn.Conv2d(2048, hidden_dim, 1)
        self.transformer = nn.Transformer(hidden_dim, nheads,
                                          num_encoder_layers, num_decoder_layers)
        self.linear_class = nn.Linear(hidden_dim, num_classes + 1)
        self.linear_bbox = nn.Linear(hidden_dim, 4)
        self.query_pos = nn.Parameter(torch.rand(100, hidden_dim))
        self.row_embed = nn.Parameter(torch.rand(50, hidden_dim // 2))
        self.col_embed = nn.Parameter(torch.rand(50, hidden_dim // 2))

    def forward(self, inputs):
        x = self.backbone(inputs)
        h = self.conv(x)
        H, W = h.shape[-2:]
        pos = torch.cat([
            self.col_embed[:W].unsqueeze(0).repeat(H, 1, 1),
            self.row_embed[:H].unsqueeze(1).repeat(1, W, 1),
        ], dim=-1).flatten(0, 1).unsqueeze(1)
        h = self.transformer(pos + h.flatten(2).permute(2, 0, 1),
                             self.query_pos.unsqueeze(1))
        return self.linear_class(h), self.linear_bbox(h).sigmoid()

detr = DETR(num_classes=91, hidden_dim=256, nheads=8, num_encoder_layers=6, num_decoder_layers=6)
detr.eval()
inputs = torch.randn(1, 3, 800, 1200)
logits, bboxes = detr(inputs)
```

LISTING 1: DETR PyTorch inference code. For clarity it uses learnt positional encodings in the encoder instead of fixed, and positional encodings are added to the input only instead of at each transformer layer. Making these changes requires going beyond PyTorch implementation of transformers, which hampers readability. The entire code to reproduce the experiments will be made available before the conference.

Bibliography

Adams, Warren P. and Terri A. Johnson (1994). "Improved Linear Programming-Based Lower Bounds for the Quadratic Assignment Problem". In: *DIMACS series in discrete mathematics and theoretical computer science* 16, pp. 43–77.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2015). "Neural Machine Translation by Jointly Learning to Align and Translate". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun.

Barrett, Samuel et al. (Jan. 1, 2017). "Making Friends on the Fly: Cooperating with New Teammates". In: *Artificial Intelligence* 242, pp. 132–171. ISSN: 0004-3702. DOI: 10.1016/j.artint.2016.10.005.

Battiti, Roberto and Giampietro Tecchiolli (1994). "The Reactive Tabu Search". In: *ORSA journal on computing* 6.2, pp. 126–140.

Bellemare, Marc G. et al. (2013). "The Arcade Learning Environment: An Evaluation Platform for General Agents". In: *Journal of Artificial Intelligence Research* 47, pp. 253–279.

Bellman, Richard (1957). "A Markovian Decision Process". In: *Journal of mathematics and mechanics*, pp. 679–684.

Bello, Irwan et al. (2019). "Attention Augmented Convolutional Networks". In: *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. IEEE, pp. 3285–3294. DOI: 10.1109/ICCV.2019.00338.

Berner, Christopher et al. (2019). "Dota 2 with Large Scale Deep Reinforcement Learning". In: *arXiv preprint arXiv:1912.06680*.

Bodla, Navaneeth et al. (2017). "Soft-NMS - Improving Object Detection with One Line of Code". In: *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*. IEEE Computer Society, pp. 5562–5570. DOI: 10.1109/ICCV.2017.593.

Boser, Bernhard E., Isabelle M. Guyon, and Vladimir N. Vapnik (1992). "A Training Algorithm for Optimal Margin Classifiers". In: *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pp. 144–152.

Boutilier, Craig (1996). "Planning, Learning and Coordination in Multiagent Decision Processes". In: *Proceedings of the 6th Conference on Theoretical Aspects of Rationality and Knowledge*, pp. 195–210.

Busoniu, Lucian, Robert Babuska, and Bart De Schutter (2008). "A comprehensive survey of multiagent reinforcement learning". In: *IEEE Transactions on Systems, Man, And Cybernetics-Part C: Applications and Reviews, 38 (2), 2008*.

Cai, Zhaowei and Nuno Vasconcelos (2019). "Cascade R-CNN: High Quality Object Detection and Instance Segmentation". In: *PAMI*.

Carion, Nicolas, Francisco Massa, et al. (May 28, 2020). "End-to-End Object Detection with Transformers". In: arXiv: 2005.12872 [cs].

Carion, Nicolas, Nicolas Usunier, et al. (2019). "A Structured Prediction Approach for Generalization in Cooperative Multi-Agent Reinforcement Learning". In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information*

Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada. Ed. by Hanna M. Wallach et al., pp. 8128–8138.

Chan, William et al. (2020). "Imputer: Sequence Modelling via Imputation and Dynamic Programming". In: *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*. Vol. 119. Proceedings of Machine Learning Research. PMLR, pp. 1403–1413.

Churchill, David and Michael Buro (2013). "Portfolio greedy search and simulation for large-scale combat in StarCraft". In: *Computational Intelligence in Games (CIG), 2013 IEEE Conference on*. IEEE, pp. 1–8.

Churchill, David, Zeming Lin, and Gabriel Synnaeve (2017). "An Analysis of Model-Based Heuristic Search Techniques for StarCraft Combat Scenarios". In: AAAI Publications, Thirteenth Artificial Intelligence and Interactive ....

Churchill, David, Abdallah Saffidine, and Michael Buro (2012). "Fast Heuristic Search for RTS Game Combat Scenarios." In: *AIIDE*, pp. 112–117.

Connolly, David T. (1990). "An Improved Annealing Scheme for the QAP". In: *European Journal of Operational Research* 46.1, pp. 93–100.

Cordonnier, Jean-Baptiste, Andreas Loukas, and Martin Jaggi (2020). "On the Relationship between Self-Attention and Convolutional Layers". In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Dalal, Navneet and Bill Triggs (2005). "Histograms of oriented gradients for human detection". In: *CVPR*.

Das, Abhishek et al. (2019). "TarMAC: Targeted Multi-Agent Communication". In: *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, pp. 1538–1546.

Devlin, Jacob et al. (June 2019). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, pp. 4171–4186. DOI: 10.18653/v1/N19-1423.

Diuk, Carlos, Andre Cohen, and Michael L. Littman (2008). "An object-oriented representation for efficient reinforcement learning". In: *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*. Ed. by William W. Cohen, Andrew McCallum, and Sam T. Roweis. Vol. 307. ACM International Conference Proceeding Series. ACM, pp. 240–247. DOI: 10.1145/1390156.1390187.

Dong, Zhiwei et al. (2020). "CentripetalNet: Pursuing High-Quality Keypoint Pairs for Object Detection". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*. IEEE, pp. 10516–10525. DOI: 10.1109/CVPR42600.2020.01053.

Duan, Kaiwen et al. (2019). "CenterNet: Keypoint Triplets for Object Detection". In: *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. IEEE, pp. 6568–6577. DOI: 10.1109/ICCV.2019.00667.

Edwards, Daniel James and T. P. Hart (1961). "The Alpha-Beta Heuristic". In.

Erhan, Dumitru et al. (2014). "Scalable Object Detection Using Deep Neural Networks". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*. IEEE Computer Society, pp. 2155–2162. DOI: 10.1109/CVPR.2014.276.

Everingham, Mark et al. (2010). "The Pascal Visual Object Classes (Voc) Challenge". In: *International journal of computer vision* 88.2, pp. 303–338.

Felzenszwalb, Pedro F., David A. McAllester, and Deva Ramanan (2008). "A discriminatively trained, multiscale, deformable part model". In: *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008), 24-26 June 2008, Anchorage, Alaska, USA*. IEEE Computer Society. DOI: 10.1109/CVPR.2008.4587597.

Fischler, Martin A. and Robert A. Elschlager (1973). "The Representation and Matching of Pictorial Structures". In: *IEEE Transactions on computers* 100.1, pp. 67–92.

Foerster, Jakob N., Yannis M. Assael, et al. (2016). "Learning to Communicate with Deep Multi-Agent Reinforcement Learning". In: *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*. Ed. by Daniel D. Lee et al., pp. 2137–2145.

Foerster, Jakob N., Gregory Farquhar, et al. (2018). "Counterfactual Multi-Agent Policy Gradients". In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*. Ed. by Sheila A. McIlraith and Kilian Q. Weinberger. AAAI Press, pp. 2974–2982.

Frank, Marguerite and Philip Wolfe (1956). "An algorithm for quadratic programming". In: *Naval Research Logistics (NRL)* 3.1-2, pp. 95–110.

Freund, Yoav and Robert E. Schapire (1995). "A Desicion-Theoretic Generalization of on-Line Learning and an Application to Boosting". In: *European Conference on Computational Learning Theory*. Springer, pp. 23–37.

Ghazvininejad, Marjan et al. (Nov. 2019). "Mask-Predict: Parallel Decoding of Conditional Masked Language Models". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, pp. 6112–6121. DOI: 10.18653/v1/D19-1633.

Girshick, Ross B. (2015). "Fast R-CNN". In: *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*. IEEE Computer Society, pp. 1440–1448. DOI: 10.1109/ICCV.2015.169.

Girshick, Ross B. et al. (2014). "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*. IEEE Computer Society, pp. 580–587. DOI: 10.1109/CVPR.2014.81.

Girshick, Ross Brook (2012). *From Rigid Templates to Grammars: Object Detection with Structured Models*. Citeseer.

Gleixner, Ambros et al. (2017). *The SCIP Optimization Suite 5.0*. eng. Tech. rep. 17-61. Takustr. 7, 14195 Berlin: ZIB.

Glorot, Xavier and Yoshua Bengio (2010). "Understanding the difficulty of training deep feedforward neural networks". In: *AISTATS*.

Gu, Jiatao et al. (2018). "Non-Autoregressive Neural Machine Translation". In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Guestrin, Carlos, Daphne Koller, et al. (2003). "Generalizing Plans to New Environments in Relational MDPs". In: *Proceedings of the 18th International Joint Conference on Artificial Intelligence*. IJCAI'03. Acapulco, Mexico: Morgan Kaufmann Publishers Inc., pp. 1003–1010.

Guestrin, Carlos, Michail G. Lagoudakis, and Ronald Parr (2002). "Coordinated Reinforcement Learning". In: *Machine Learning, Proceedings of the Nineteenth International Conference (ICML 2002), University of New South Wales, Sydney, Australia, July 8-12, 2002*. Ed. by Claude Sammut and Achim G. Hoffmann. Morgan Kaufmann, pp. 227–234.

Hahn, Peter and Thomas Grant (1998). "Lower Bounds for the Quadratic Assignment Problem Based upon a Dual Formulation". In: *Operations Research* 46.6, pp. 912–922.

He, Kaiming, Ross B. Girshick, and Piotr Dollár (2019). "Rethinking ImageNet Pre-Training". In: *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. IEEE, pp. 4917–4926. DOI: 10.1109/ICCV.2019.00502.

He, Kaiming, Georgia Gkioxari, et al. (2017). "Mask R-CNN". In: *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*. IEEE Computer Society, pp. 2980–2988. DOI: 10.1109/ICCV.2017.322.

He, Kaiming, Xiangyu Zhang, et al. (2016). "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

Hosang, Jan Hendrik, Rodrigo Benenson, and Bernt Schiele (2017). "Learning Non-maximum Suppression". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, pp. 6469–6477. DOI: 10.1109/CVPR.2017.685.

Hu, Han et al. (2018). "Relation Networks for Object Detection". In: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. IEEE Computer Society, pp. 3588–3597. DOI: 10.1109/CVPR.2018.00378.

Jiang, Jiechuan, Iñigo Fernández del Amo, and Zongqing Lu (2018). "Graph Convolutional Reinforcement Learning for Multi-Agent Cooperation". In: *CoRR* abs/1810.09202.

Jiang, Jiechuan and Zongqing Lu (2018). "Learning Attentional Communication for Multi-Agent Cooperation". In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*. Ed. by Samy Bengio et al., pp. 7265–7275.

Jonker, Roy and Anton Volgenant (1987). "A Shortest Augmenting Path Algorithm for Dense and Sparse Linear Assignment Problems". In: *Computing* 38.4, pp. 325–340.

Kingma, Diederik P. and Jimmy Ba (2015). "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun.

Kirillov, Alexander, Ross B. Girshick, et al. (2019). "Panoptic Feature Pyramid Networks". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. Computer Vision Foundation / IEEE, pp. 6399–6408. DOI: 10.1109/CVPR.2019.00656.

Kirillov, Alexander, Kaiming He, et al. (2019). "Panoptic Segmentation". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. Computer Vision Foundation / IEEE, pp. 9404–9413. DOI: 10.1109/CVPR.2019.00963.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton (2012). "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information*

Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States. Ed. by Peter L. Bartlett et al., pp. 1106–1114.

Kuhn, Harold W. (1955). "The Hungarian Method for the Assignment Problem". In: *Naval research logistics quarterly* 2.1-2, pp. 83–97.

Lacoste-Julien, Simon and Martin Jaggi (2015). "On the Global Linear Convergence of Frank-Wolfe Optimization Variants". In: *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*. Ed. by Corinna Cortes et al., pp. 496–504.

Law, Hei and Jia Deng (2018). "Cornernet: Detecting Objects as Paired Keypoints". In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 734–750.

Li, Yi et al. (2017). "Fully Convolutional Instance-Aware Semantic Segmentation". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, pp. 4438–4446. DOI: 10.1109/CVPR.2017.472.

Lin, Kaixiang et al. (2018). "Efficient Large-Scale Fleet Management via Multi-Agent Deep Reinforcement Learning". In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*. Ed. by Yike Guo and Faisal Farooq. ACM, pp. 1774–1783. DOI: 10.1145/3219819.3219993.

Lin, Tsung-Yi, Piotr Dollár, et al. (2017). "Feature Pyramid Networks for Object Detection". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, pp. 936–944. DOI: 10.1109/CVPR.2017.106.

Lin, Tsung-Yi, Priya Goyal, et al. (2017). "Focal Loss for Dense Object Detection". In: *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*. IEEE Computer Society, pp. 2999–3007. DOI: 10.1109/ICCV.2017.324.

Lin, Tsung-Yi, Michael Maire, et al. (2014). "Microsoft COCO: Common objects in context". In: *ECCV*.

Liu, Li et al. (Feb. 1, 2020). "Deep Learning for Generic Object Detection: A Survey". In: *International Journal of Computer Vision* 128.2, pp. 261–318. ISSN: 1573-1405. DOI: 10.1007/s11263-019-01247-4.

Liu, Wei, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, et al. (2016). "Ssd: Single Shot Multibox Detector". In: *European Conference on Computer Vision*. Springer, pp. 21–37.

Liu, Wei, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, et al. (2016). "SSD: Single Shot MultiBox Detector". In: *ECCV*.

Loshchilov, Ilya and Frank Hutter (2019). "Decoupled Weight Decay Regularization". In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Lowe, Ryan et al. (2017). "Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon et al., pp. 6379–6390.

Lüscher, Christoph et al. (2019). "RWTH ASR Systems for LibriSpeech: Hybrid vs Attention - w/o Data Augmentation". In: *arXiv:1905.03072*.

Meuleau, Nicolas et al. (1998). "Solving very large weakly coupled Markov decision processes". In: pp. 165–172.

Milletari, Fausto, Nassir Navab, and Seyed-Ahmad Ahmadi (2016). "V-net: Fully convolutional neural networks for volumetric medical image segmentation". In: *3DV*.

Mnih, Volodymyr et al. (2016). "Asynchronous Methods for Deep Reinforcement Learning". In: *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*. Ed. by Maria-Florina Balcan and Kilian Q. Weinberger. Vol. 48. JMLR Workshop and Conference Proceedings. JMLR.org, pp. 1928–1937.

Oliehoek, Frans A. and Christopher Amato (2016). *A Concise Introduction to Decentralized POMDPs*. Vol. 1. Springer.

Oliehoek, Frans A., Matthijs TJ Spaan, and Nikos Vlassis (2008). "Optimal and Approximate Q-Value Functions for Decentralized POMDPs". In: *Journal of Artificial Intelligence Research* 32, pp. 289–353.

Ontanón, Santiago et al. (2013). "A survey of real-time strategy game AI research and competition in StarCraft". In: *IEEE Transactions on Computational Intelligence and AI in games* 5.4, pp. 293–311.

Oord, Aäron van den et al. (2018). "Parallel WaveNet: Fast High-Fidelity Speech Synthesis". In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. Ed. by Jennifer G. Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 3915–3923.

Park, Eunbyung and Alexander C. Berg (2015). "Learning to decompose for object detection and instance segmentation". In: *arXiv:1511.06449*.

Parmar, Niki et al. (2018). "Image Transformer". In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. Ed. by Jennifer G. Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 4052–4061.

Paszke, Adam et al. (2019). "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach et al., pp. 8024–8035.

Pineda, Luis et al. (2019). "Elucidating image-to-set prediction: An analysis of models, losses and datasets". In: *arXiv:1904.05709*.

Proper, Scott and Prasad Tadepalli (2009). "Solving Multiagent Assignment Markov Decision Processes". In: *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*. AAMAS '09. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, pp. 681–688. ISBN: 978-0-9817381-6-1.

Radford, Alec et al. (2019). "Language Models are Unsupervised Multitask Learners". In.

Raghavan, Prabhakar and Clark D. Tompson (Dec. 1, 1987). "Randomized Rounding: A Technique for Provably Good Algorithms and Algorithmic Proofs". In: *Combinatorica* 7.4, pp. 365–374. ISSN: 1439-6912. DOI: 10.1007/BF02579324.

Rajpurkar, Pranav et al. (Dec. 25, 2017). "CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning". In: arXiv: 1711.05225 [cs, stat].

Rashid, Tabish et al. (2018). "QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning". In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. Ed. by Jennifer G. Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 4292–4301.

Redmon, Joseph, Santosh Kumar Divvala, et al. (2016). "You Only Look Once: Unified, Real-Time Object Detection". In: *2016 IEEE Conference on Computer Vision*

and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016. IEEE Computer Society, pp. 779–788. DOI: 10.1109/CVPR.2016.91.

Redmon, Joseph and Ali Farhadi (2017). "YOLO9000: Better, Faster, Stronger". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, pp. 6517–6525. DOI: 10.1109/CVPR.2017.690.

Ren, Mengye and Richard S. Zemel (2017). "End-to-End Instance Segmentation with Recurrent Attention". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, pp. 293–301. DOI: 10.1109/CVPR.2017.39.

Ren, Shaoqing et al. (2015a). "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*. Ed. by Corinna Cortes et al., pp. 91–99.

— (2015b). "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*. Ed. by Corinna Cortes et al., pp. 91–99.

Rezatofighi, Hamid et al. (2019). "Generalized Intersection over Union". In: *CVPR*.

Rezatofighi, S Hamid et al. (2018). "Deep perm-set net: Learn to predict sets with unknown permutation and cardinality using deep neural networks". In: *arXiv:1805.00613*.

Rezatofighi, Seyed Hamid et al. (2017). "DeepSetNet: Predicting Sets with Deep Neural Networks". In: *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*. IEEE Computer Society, pp. 5257–5266. DOI: 10.1109/ICCV.2017.561.

Al-Rfou, Rami et al. (2019). "Character-Level Language Modeling with Deeper Self-Attention". In: *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*. AAAI Press, pp. 3159–3166. DOI: 10.1609/aaai.v33i01.33013159.

Richards, Arthur et al. (2002). "Coordination and Control of Multiple UAVs". In: *AIAA Guidance, Navigation, and Control Conference and Exhibit*, p. 4588.

Romera-Paredes, Bernardino and Philip Hilaire Sean Torr (2015). "Recurrent Instance Segmentation". In: *ECCV*.

Salvador, Amaia et al. (2017). "Recurrent Neural Networks for Semantic Instance Segmentation". In: *arXiv:1712.00617*.

Sanner, Scott and Craig Boutilier (2012). "Practical linear value-approximation techniques for first-order MDPs". Preprint on arXiv:1206.6879.

Sermanet, Pierre et al. (2014). "OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks". In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun.

Shetty, Vijay K., Moises Sudit, and Rakesh Nagi (2008). "Priority-Based Assignment and Routing of a Fleet of Unmanned Combat Aerial Vehicles". In: *Computers & Operations Research* 35.6, pp. 1813–1828.

Silver, David, Aja Huang, et al. (2016). "Mastering the Game of Go with Deep Neural Networks and Tree Search". In: *nature* 529.7587, pp. 484–489.

Silver, David, Thomas Hubert, et al. (2018). "A General Reinforcement Learning Algorithm That Masters Chess, Shogi, and Go through Self-Play". In: *Science* 362.6419, pp. 1140–1144.

Singh, Amanpreet, Tushar Jain, and Sainbayar Sukhbaatar (2019). "Individualized Controlled Continuous Communication Model for Multiagent Cooperative and Competitive Tasks". In: *International Conference on Learning Representations*.

Singh, Satinder P. and David Cohn (1998). "How to dynamically merge Markov decision processes". In: *Advances in neural information processing systems*, pp. 1057–1063.

Stewart, Russell, Mykhaylo Andriluka, and Andrew Y. Ng (2016). "End-to-End People Detection in Crowded Scenes". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, pp. 2325–2333. DOI: 10.1109/CVPR.2016.255.

Sukhbaatar, Sainbayar, Arthur Szlam, and Rob Fergus (2016). "Learning Multiagent Communication with Backpropagation". In: *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*. Ed. by Daniel D. Lee et al., pp. 2244–2252.

Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le (2014). "Sequence to Sequence Learning with Neural Networks". In: *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*. Ed. by Zoubin Ghahramani et al., pp. 3104–3112.

Sutton, Richard S and Andrew G Barto (2018). *Reinforcement learning: An introduction*. MIT press.

Synnaeve, Gabriel, Jonas Gehring, et al. (2019). "Growing Up Together: Structured Exploration for Large Action Spaces". In.

Synnaeve, Gabriel, Zeming Lin, et al. (2018). "Forward Modeling for Partial Observation Strategy Games - A StarCraft Defogger". In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*. Ed. by Samy Bengio et al., pp. 10761–10771.

Synnaeve, Gabriel, Qiantong Xu, et al. (2019). "End-to-end ASR: from Supervised to Semi-Supervised Learning with Modern Architectures". In: *arXiv:1911.08460*.

Tesauro, Gerald (2005). "Online resource allocation using decompositional reinforcement learning". In: *AAAI*. Vol. 5, pp. 886–891.

The Glop Linear Solver (n.d.). https://developers.google.com/optimization/lp/glop.

Tian, Yuandong et al. (2017). "ELF: An Extensive, Lightweight and Flexible Research Platform for Real-time Strategy Games". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon et al., pp. 2659–2669.

Tian, Zhi et al. (2019). "FCOS: Fully Convolutional One-Stage Object Detection". In: *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. IEEE, pp. 9626–9635. DOI: 10.1109/ICCV.2019.00972.

Toth, Paolo and Daniele Vigo (2002). *The Vehicle Routing Problem*. SIAM.

Usunier, Nicolas et al. (2016). "Episodic exploration for deep deterministic policies: An application to starcraft micromanagement tasks". Preprint on arXiv:1609.02993.

Vaswani, Ashish et al. (2017). "Attention is All you Need". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing*

Systems 2017, December 4-9, 2017, Long Beach, CA, USA. Ed. by Isabelle Guyon et al., pp. 5998–6008.

Vinyals, Oriol, Igor Babuschkin, et al. (2019). "Grandmaster Level in StarCraft II Using Multi-Agent Reinforcement Learning". In: *Nature* 575.7782, pp. 350–354.

Vinyals, Oriol, Samy Bengio, and Manjunath Kudlur (2016). "Order Matters: Sequence to sequence for sets". In: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun.

Viola, Paul and Michael Jones (2001). "Robust Real-Time Object Detection". In: *International journal of computer vision* 4.34-47, p. 4.

Wang, Derui et al. (2019). "Daedalus: Breaking Non-Maximum Suppression in Object Detection via Adversarial Examples". In: *arXiv*, arXiv–1902.

Wang, Xiaolong et al. (2018). "Non-Local Neural Networks". In: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. IEEE Computer Society, pp. 7794–7803. DOI: 10.1109/CVPR.2018.00813.

Wang, Xinlong et al. (2020). "SOLOv2: Dynamic, Faster and Stronger". In: *arXiv preprint arXiv:2003.10152*.

Wawrzynski, Pawel (2015). "Control policy with autocorrelated noise in reinforcement learning for robotics". In: *International Journal of Machine Learning and Computing* 5.2, p. 91.

Wilcoxon, Frank (1945). "Individual comparisons by ranking methods". In: *Biometrics bulletin* 1.6, pp. 80–83.

Witt, Christian Schröder de et al. (2019). "Multi-Agent Common Knowledge Reinforcement Learning". In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach et al., pp. 9924–9935.

Wu, Yuxin et al. (2019). *Detectron2*. https://github.com/facebookresearch/detectron2.

Xiong, Yuwen et al. (2019). "UPSNet: A Unified Panoptic Segmentation Network". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. Computer Vision Foundation / IEEE, pp. 8818–8826. DOI: 10.1109/CVPR.2019.00902.

Yang, Yaodong et al. (2018). "Mean Field Multi-Agent Reinforcement Learning". In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. Ed. by Jennifer G. Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 5567–5576.

Zambaldi, Vinícius Flores et al. (2019). "Deep reinforcement learning with relational inductive biases". In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Zhang, Shifeng et al. (2020). "Bridging the Gap Between Anchor-Based and Anchor-Free Detection via Adaptive Training Sample Selection". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*. IEEE, pp. 9756–9765. DOI: 10.1109/CVPR42600.2020.00978.

Zhou, Xingyi, Dequan Wang, and Philipp Krähenbühl (2019). "Objects as points". In: *arXiv:1904.07850*.

Zhou, Xingyi, Jiacheng Zhuo, and Philipp Krähenbühl (2019). "Bottom-Up Object Detection by Grouping Extreme and Center Points". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20,*

2019. Computer Vision Foundation / IEEE, pp. 850–859. DOI: 10.1109/CVPR.2019.00094.

Zou, Zhengxia et al. (May 15, 2019). "Object Detection in 20 Years: A Survey". In: arXiv: 1905.05055 [cs].

RÉSUMÉ

Cette thèse explore l'utilisation de fonctions de perte structurées dans deux domaines distincts. Dans la première contribution, nous nous intéressons à l'apprentissage par renforcement multi-agent, dans le contexte d'environnements qui peuvent être séparés en plusieurs tâches faiblement dépendantes. On s'attache à trouver des politiques qui se généralisent à plus d'agents et de tâches que les scénarios d'entraînement, permettant ainsi d'augmenter la taille des problèmes qui peuvent être approchés. Notre solution affecte les agents aux tâches en résolvant un problème d'optimisation centralisé dont la fonction objectif est paramétrée par un réseau de neurones. On montre que l'expressivité du problème d'optimisation et celle du réseau de neurones influencent la capacité du modèle à généraliser, et qu'avec les bons choix, la politique peut généraliser à plus de 5 fois plus d'agents que pendant l'entraînement.

Dans la seconde contribution, nous formulons la détection d'objets comme un problème de prédiction d'ensemble, et nous concevons un modèle dans cette optique. Notre solution utilise un réseau convolutionel profond, comme souvent en vision par ordinateur, et un encodeur-décodeur de Transformer, une architecture qui a récemment permis d'importants progrès en traitement du langage. Remarquablement, notre solution n'incorpore que peu de biais inductif, et ne nécessite donc pas de composants spécifiques à la détection d'objets, tels que les ancres de détection. Avec un nombre de paramètres comparable, notre modèle égale la performance de modèles de référence, tels que Retinanet et Faster R-CNN sur le dataset de détection COCO. Pour finir, nous montrons que la méthode peut naturellement être étendue à la segmentation panoptique, où elle surpasse les approches concurrentes, démontrant ainsi sa généralité.

MOTS CLÉS

Apprentissage profond, système multi-agent, Apprentissage par renforcement, vision par ordinateur, détection d'objets

ABSTRACT

This thesis explores the use of structured losses in two different domains. In the first contribution, we focus on multi-agent reinforcement learning (MARL), in environments that can be separated into several loosely coupled tasks. We set out to find policies that can generalize well to more agents and tasks than seen during training, effectively scaling up the size of problems that can be tackled. Our solution assigns agents to tasks by approximately solving a centralized optimization problem whose objective function is parameterized by a neural network. We study how the expressivity of the optimization problem and that of the neural network influence the generalization capabilities of the model, and show that with the right choices, the policy can generalize to more than 5 times more agents than seen during training.

In the second contribution we formulate object detection as a set prediction problem, and design a model that can effectively tackle this formulation. Our solution leverages a deep convolutional network, as is customary in computer vision, and a transformer encoder-decoder network, an architecture that has enabled significant progress in natural language processing. Crucially, our solution incorporates minimal inductive bias, thereby alleviating the need for hand-designed detection-specific components such as anchors or non-maximal suppression. With a comparable parameter budget, our model matches the performance of well-established and highly-optimized baselines such as Retinanet and Faster R-CNN on the challenging COCO detection dataset. Finally, we show that the method can be naturally extended to perform panoptic segmentation, where it outperforms competing approaches, thus showing the versatility of the model.

KEYWORDS

deep learning, multi-agent, reinforcement learning, computer vision, object detection