THÈSE DE DOCTORAT DE L'UNIVERSITÉ DE LYON
opérée au sein de
INSA LYON

École Doctorale 512
Informatique et Mathématique de Lyon
(INFOMATHS)

Spécialité
Informatique

Présentée par

Théo Jaunet

Pour obtenir le grade de
DOCTEUR de L'UNIVERSITÉ DE LYON

Sujet de la thèse :

Deep Learning Interpretability with Visual Analytics: Exploring Reasoning and Bias Exploitation

Interprétabilité de l'Apprentissage Profond via Analyse Visuelle : Exploration de Raisonnements et de l'Exploitation de Biais

ABSTRACT

In the last couple of years, Artificial Intelligence (AI) and Machine Learning have evolved, from research domains addressed in laboratories far from the public eye, to technology deployed on industrial scale widely impacting our daily lives. This trend has started to raise legitimate concerns, as it is also used to address critical problems like finance and autonomous driving, in which decisions can have a life-threatening impact. Since a large part of the underlying complexity of the decision process is learned from massive amounts of data, it remains unknown to both the builders of those models and to the people impacted by them how models take decisions. This led to the new field of eXplainable AI (XAI) and the problem of analyzing the behavior of trained models to shed their reasoning modes and the underlying biases they are subject to. This thesis we contributed to this emerging field with the design of visual analytics systems tailored to the study and improvement of interpretability of Deep Neural Networks. Our goal was to empower experts with tools helping them to better interpret the decisions of their models. We also contributed with explorable applications designed to introduce Deep Learning methods to non-expert audiences. Our focus was on the under-explored challenge of interpreting and improving models for different applications such as robotics, where important decisions must be taken from high-dimensional and low-level inputs such as images. All tools designed during this thesis were published as open-source projects, and when possible, our visualizations have been made available online as prototypes, as they are highly interactive and thus can help to foster further research by the research community.

In the first part of this thesis, we addressed how a robot may answer natural questions (Visual Question Answering (VQA)) with state-of-the-art transformer models based on self-attention. Our focus was on the interpretation of their attention maps which we analyzed with a visual analytics system we designed. We showed how our system can be used by model builders to detect when their model may exploit biases, i.e. undesired shortcuts in learning by exploiting spurious correlations in data, instead of resorting to the requested reasoning process.

In a second part, we addressed the interpretability of robot navigation, *i.e.* how a robot may explore its environment (*e.g.* an office). In our case, such a step was addressed by neural models which include a recurrent memory, which is at the core of their decisions. We focused on how this memory (and its impact on decisions) can be interpreted. We designed a new visual analytics system to account for the challenge of analyzing this type of neural model.

The third part of this thesis focused on the interpretability of robotics systems trained in simulation and deployed to real physical environments, which has now quickly become a standard mode of operations. We focused on the tasks of robot ego-localization and the interpretations of its decisions. We provided an interpretation of potential sources of gaps with the help of a new visual analytics interface, with which we identified biases and gaps that affect models' decisions.

Finally, we concluded this thesis with a reflection across these parts to highlight the challenges of building an end-to-end model to robotic task of answering natural language questions. This conclusion also introduce novel and still open research directions to provide better interpretability of such models' decisions and mitigate their bias.

RÉSUMÉ

Au cours des dernières années, l'IA et l'apprentissage automatique ont évolué, passant de domaines de recherche traités dans des laboratoires éloignés du public à des technologies déployées à l'échelle industrielle ayant un impact considérable sur notre vie quotidienne. Étant donné que ces technologies sont également utilisées pour résoudre des problèmes critiques tels que la finance et la conduite autonome, dans lesquels les décisions peuvent mettre des personnes en danger, cette tendance a commencé à susciter des inquiétudes légitimes. Puisqu'une grande partie de la complexité sous-jacente du processus de décision est apprise à partir de quantités massives de données, la manière dont les modèles prennent des décisions reste inconnue, tant pour les constructeurs de ces modèles que pour les personnes concernées. Cela a conduit au nouveau domaine de l'eXplainable AI (XAI) et au problème de l'analyse du comportement des modèles entraînés pour mettre en lumière leurs modes de raisonnement et les biais auxquels ils sont soumis. Dans cette thèse, nous avons contribué à ce domaine émergent avec la conception de systèmes d'analyse visuelle destinés à l'étude et à l'amélioration de l'interprétabilité des réseaux neuronaux profonds. Notre objectif était de permettre aux experts de disposer d'outils les aidant à mieux interpréter les décisions de leurs modèles. Nous avons également proposé des applications explorables conçues pour présenter les méthodes d'apprentissage profond à un public non-expert. Nous nous sommes concentrés sur le défi sous-exploré de l'interprétation et de l'amélioration des modèles pour différentes applications telles que la robotique, où des décisions importantes doivent être prises à partir d'entrées de haut niveau et de haute dimension telles que des images. Tous les outils conçus au cours de cette thèse ont été publiés en tant que projets open-source, et lorsque cela était possible, nos visualisations ont été mises à disposition en ligne en tant que prototypes, puisque ces outils sont hautement interactifs et peuvent donc contribuer à encourager la poursuite des recherches par la communauté des chercheurs.

Dans la première partie de cette thèse, nous avons abordé la manière dont un robot peut répondre à des questions naturelles (VQA) avec des modèles état de l'art *transformer* basés sur attention. Nous nous sommes concentrés sur l'interprétation de leurs cartes d'attention que nous avons analysées avec un système d'analyse visuelle que nous avons conçu. Nous avons montré comment notre système peut être utilisé par les constructeurs de modèles pour détecter lorsque ceux-ci peuvent exploiter des biais, c'est-à-dire des raccourcis indésirables dans l'apprentissage en exploitant des corrélations trompeuses dans les données, au lieu de recourir au processus de raisonnement demandé.

Dans une deuxième partie, nous avons abordé l'interprétabilité de la navigation des robots, c'est-à-dire la manière dont un robot peut explorer son environnement (par exemple un bureau). Dans notre cas, une telle étape a été traitée par des modèles neuronaux qui incluent une mémoire récurrente, qui est au cœur de leurs décisions. Nous nous sommes concentrés sur la façon dont cette mémoire (et son impact sur les décisions) peut être interprétée. Nous avons conçu un nouveau système d'analyse visuelle pour tenir compte du défi que représente l'analyse de ce type de modèle neuronal.

La troisième partie de cette thèse s'est centrée sur l'interprétabilité des systèmes robotiques entraînés en simulation et déployés dans des environnements physiques réels, ce qui est rapidement devenu un mode d'opération standard. Nous nous sommes concentrés sur les tâches d'ego-localisation du robot et les interprétations de ses décisions. Nous avons fourni une interprétation des sources potentielles de lacunes à l'aide d'une nouvelle interface d'analyse visuelle, avec laquelle nous avons identifié les biais et les lacunes qui affectent les décisions des modèles.

Enfin, nous avons conclu cette thèse par une réflexion sur l'ensemble de ses parties pour mettre en évidence les défis de la construction d'un modèle de bout en bout pour la tâche robotique de réponse aux questions en langage naturel. Cette conclue présente également les nouvelles directions de recherche encore ouvertes pour fournir une meilleure interprétabilité des décisions de ces modèles et atténuer leur biais.

REMERCIEMENTS

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
NN	Neural Network
DNN	Deep Neural Networks
DRL	Deep Reinforcement Learning
RL	Reinforcement Learning
MLP	Multi-Layer Perceptron
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
POS	Part-Of-Speech
NLP	Natural Language Processing
VQA	Visual Question Answering
HCI	Human-Computer Interaction
LV	Language-to-Vision
VL	Vision-Language
A2C	Advantage Actor-Critic
GRU	Gated Recurrent Unit
A3C	Asynchronous Advantage Actor-Critic
FC	Fully Connected
FoV	Field of View
DQN	Deep Q-Networks
CV	Computer Vision
GT	Ground-truth
DARPA	Defense Advanced Research Projects Agency
MSE	Mean Squared Error
IoU	Intersection over Union
EQA	Embodied Question Answering
MDP	Markov Decision Process

CHAPTER 1

INTRODUCTION

Contents

Artificial Intelligence (AI), is a vast domain with concrete applications now in our daily lives. It has however a long history of both definitions and applications [1]. In its early days, AI's popularity emerged in science-fiction from movies and literature, often in the shape of autonomous and sometimes sentient robots, such as in "Metropolis" (1926), the "Wizard of Oz" (1939), or "Tobor the great" (1954). Such popularity anchored high expectations for AI, to the point that in 1968 A. Clarke, and S. Kubrick estimated, in their movie entitled "2001: a space odyssey", that by 2001 machines would have at least matched Humans' intelligence. Such popularity, lead to the first wave of AI in academia, with government agencies such as Defense Advanced Research Projects Agency (DARPA) funding research. This was motivated by the advocacy of researchers who argued, among others, that the ability to interpret and translate languages was within reach in the next couple of years. While climbing the hill of the first advents of AI research, came in 1974 the disillusion that many mountains remained to be escalated to meet such high expectations. One of the major setbacks, at this time, was the lack of computational power to match the theoretical groundwork. This led to a downfall of funding, which is now referred to as "winters".

This winter has been "re-heated" thanks to endeavors, and discoveries that lead, among others, to the birth of expert systems in the '80s. This was also encouraged by the ever-increasing performance of computers—in terms of memory, storage, and computation—which improved their affordability. Expert systems consist of a set of rules established by an expert of a domain, *e.g.* a doctor, that the AI could rely on when yielding a diagnostic, for example. Despite their requirement for an exhaustive set of rules, those systems surged in industry and were applied to many domains such as health care [137]. However, in the '90s those systems began to be disregarded because the more complex they became, the harder they

1. Throughout the manuscript we refer to AI the set of methods that provides to computers the ability to tackle high-end reasoning tasks humans can achieve successfully.

were to maintain and took time and effort to set up, without, once again, meeting the same high expectations inherent to AI. For instance, the myth of "sentient" computer programs able to sustain a conversation with a human.

Surprisingly, during this new winter, AI reached new milestones. One of the most memorable at this time was in 1997 by beating for the first time in history the then world chess champion Gary Kasparov. This event reignited the spark of the so-called era of "heat" of AI, which kept increasing to its highest ever nowadays. Once again the non-stopping technological advances took a major role in this, by leading computers to be powerful enough to train artificial Neural Network (NN). Those NNs are composed of inter-connected artificial neurons, which design began in the '40s with the first mathematical model of a biological neuron [134]. At the core of the success of NNs is their ability to learn to identify patterns from data they are fed to. Since the capacity of computers grows, we are able to build bigger models, with neurons inter-connected in multiple layers known as Deep Neural Networks (DNN). This helped models to formulate more elaborated rules, and address more challenging tasks. Hence, as a repercussion, the size of models kept increasing. In 1998, LeNet-5 [116] one of the first DNN that gained popularity, had around 60 thousand of trainable parameters. Fourteen years later, AlexNet [112] had over 60 million trainable parameters.

Nowadays, with models reaching hundreds of billions of trainable parameters [28], researchers across many domains reached new horizons and breakthroughs in problems previously seen as unfathomable. Among others, there is notable examples in biology by unfolding proteins [93], in healthcare by identifying cancerous cells [46, 110], beating humans at long-term strategies board [186] and video games [145, 220], autonomously driving cars from cameras [80], and generating creative art designs such as music [62], and images [15, 69]. With such endeavors, the high expectations of sentient robots from the past century have never seemed so close! These successes soared thanks to the DNN's ability to learn to associate inputs and outputs through a mapping driven by rules they discover buried in the data they consume. Prior to that, developers had to manually write those rules themselves in their code. For instance, with rules, one may establish that an animal with two pointy ears and whiskers should be classified as a cat. Nonetheless, such a classification may not exclude the possibility that this animal may be a tiger, or that a cat may be missing an ear. Thus, due to mistakes that may occur, and the unrealistic amount of time it would require to account for every possibility, developers could not address problems with an exhaustive set of rules as a machine would. However, the side effect of letting any algorithm learn its own system, is that the underlying process that leads to an outcome remains unknown to both developers and end-users.

Performances that may exceed Humans raise nonetheless trust issues as they do not ensure the fairness of decisions that might be reached thanks to rules grasped from a biased design or data. This concern is emphasized in critical

situations where even small errors of judgment can lead to dire consequences. For instance, when a driverless car may decide to turn on the road with no apparent reason for the humans inside [209]. Understanding models' decisions, even in non-life-threatening situations, is primordial as nowadays they have a broad impact on our everyday lives. For instance, whenever we query information using keywords on search engines, when we receive recommendations of related topics or videos, when paths and directions are presented according to traffic predictions in navigation mobile applications, and sometimes even when candidates are selected for job interviews. Hence the rise of ethical and fairness questions of who decides how a model should behave, and on what ground a model should reach a decision. Such interests are also accentuated by legislation at country or continental level. The GDPR [160] is a notable example to protect Europeans with is a set of regulations applicable since May 2018, stating that: *"any AI system that is integrated into people's lives must be capable of contest, account, and redress to citizens and representatives of the public interest"* [179].

Numerous examples of failure cases of AI that impacted Humans' lives have been revealed and investigated. As an example, and illustrated in Figure 1.1 ①, when using Pulse [138], an up-sampling model with the image the most left, the model is trained to yield images with better quality. However, We can observe in the reconstructed image (① bottom) that the model fails to convey the individual's features replaced them with Caucasian ones. In addition, models designed to treat language by predicting the next words of sentences, have been shown to leverage discriminating samples towards individuals and ethnicities from their training dataset [28]. As a result, when following words of certain religions or minorities, the model predicted only negative words and racial slurs. Sadly, such models have also been shown to convey gender bias [166, 21]. For instance, by translating occupations where men are over-represented as male *e.g.*"he is a doctor", while other occupations are translated as female *e.g.*"she is a nurse". In order to learn to predict words, those models have been fed with a very large amount of data directly sampled from the internet, hence it may not have been properly curated. This led models to leak personal information such as home addresses [33] simply by prompting the name of a person and letting a model auto-complete the rest of the sentence. As those models are the underlying program that runs widely in platforms we use daily, a very large audience is facing those biases. Many problems have been raised, but only a few solutions emerged to tackle them. For instance, Google Photos' model for image classification identified some humans as "gorillas", and in three years span, the only solution found was to remove image categories related to primates such as "gorilla", "chimpanzee", and "monkey" from their image labeling models. This solution cannot be reliable as it requires taking into account every bias from datasets and mistakes made by models as many of them are hidden, it would require an unfathomable amount of time to only obtain an incomplete model. In addition, models tend to behave

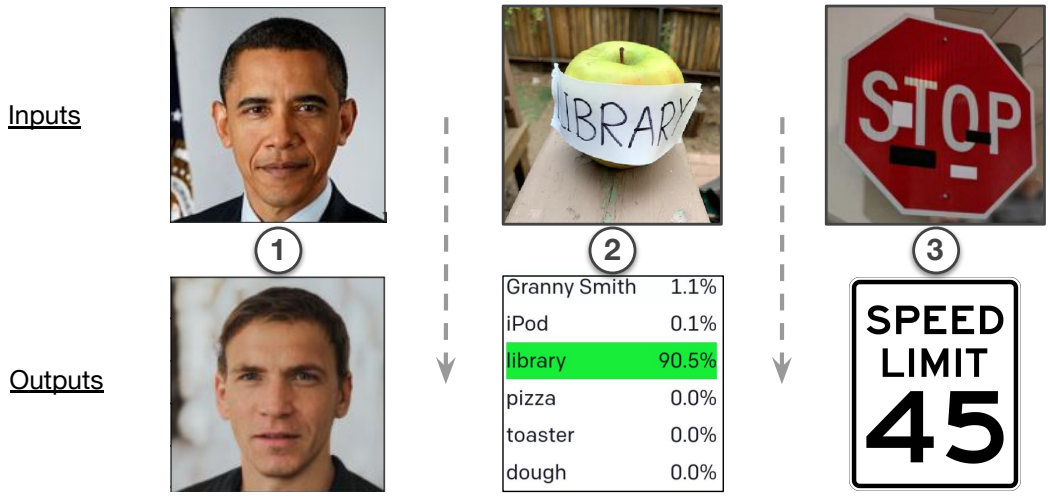


Figure 1.1 – Examples of failures of DL models: ① Pulse [138], with an input image of the former US president Barack Obama, the model tasked to reproduce it, outputs the image of a Caucasian man. ② the CLIP model [168], can be induced to output wrong classification by simply taping tags to objects, *e.g.* as illustrated here, the apple is predicted as "library" [72]. ③, traffic signs detection models can interpret stop signs as speed limit ones only with only a bit of duct tape on it [64].

unexpectedly [117] for example discovering and exploiting unknown glitches in video games [60]. This is because Deep Learning (DL) models tend to be "lazy", in the sense that they often learn to rely on shortcuts in their process to yield outputs. As an example, a model designed to predict what object is represented in an image can be easily be fooled by taping a tag with the name of another object on it. As illustrated in Figure 1.1 ②, CLIP [168] one of the most popular DL models of 2021 matching image and text, can be fooled and classify the image of an apple with a tag labeled as "library" as a library [72].

Another issue with models deployed in production is their ability to be tricked by malicious users. Small disturbances on images that may even be imperceptible to humans [74] can alter a model's decision to its opposite. As depicted in Figure 1.1 ③, stop traffic signs can be attacked by small patches of tape to fool models' and make them recognize a speed limit sign [64]. Such an attack can easily be overlooked by humans as the patches of tape may not alter their ability to identify stop signs. Those problems may only be the tip of the iceberg, and yet they hinder any deployment of the achievements of DL algorithms to real-life applications that may have an influence on one's life, and be life-threatening in critical applications (*e.g.* driverless cars).

Among the many ways to prevent biases in DL algorithms, the field of *interpretabilty* aims at empowering developers with a set of tools that helps them to identify and understand the hidden rules that those models may exploit to yield a decision. *Interpretabilty* also aims at benefiting end-users with increased fairness,

and ethics of their algorithms which are currently difficult to formulate in a way that those algorithms could leverage while learning [56]. In addition, being able to inquire why a model reaches a certain output will foster a broader social acceptance [36] which can lead to a more genuine trust of their decisions [172, 58]. But what does it means to interpret a model? How can one dig in the billions of hidden connections within models to extract hidden rules that lead to decisions? How can one leverage insights gained to further improve models and prevent the exploitation of bias? This manuscript is dedicated to the design of visual analytics systems conceived to address these challenges. Our aim is to develop interpretability methods for robotic tasks, to help robot reach their history of high expectations while mitigating their biases.

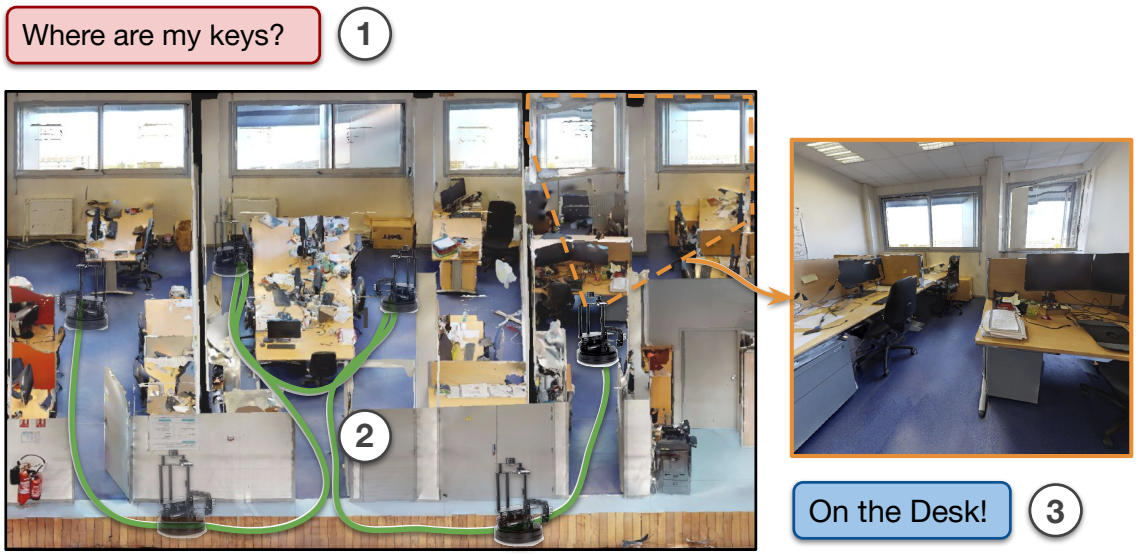


Figure 1.2 – To be able to answer mundane questions such as *"where are my keys"*, robots are required to master three reasoning skills: first ①, the ability to understand natural language questions to grasp what is asked, and analyze its vision to answer. Then, robots need the ability to navigate in an environment to look for those keys as fast as possible, and then, the ability to self-localize to both avoid going to already searched rooms, and, when found, to communicate the position of the keys.

1.1 Robotics Task: "Where are my keys"

Robotics is among the most promising application of artificial intelligence, and probably the most represented one in movies and literature. It has applications, from robots doing our chores such as groceries and house-works, to social robots assisting humans with handicaps, and even autonomous cars. But despite the

advances in academia and the soaring of DL, we haven't encountered them yet in a fully automated and reliable way in our daily life. This is due to the fact that deploying robots into our world requires addressing a combination of challenges yet to be successfully tackled enough to ensure their safety. As illustrated in Figure 1.2, to be able to answer an everyday question such as *"Where are my keys?"* a robot needs to tackle multiple sub-problems. First, *natural language* knowledge to grasp what needs to be found from the question, and a *Visual recognition* ability to identify the keys among hundreds of other items. This is known as ***Visual Question Answering (VQA)***. Second, the robot needs ***navigation*** to efficiently search the keys without being a danger to itself and others (*e.g.* bumping into humans, or falling off stairs). Finally, ***ego-pose localization*** enables the robot to communicate the position of the keys, and avoid searching the same area twice. The slightest error in those sub-problems can lead to a radical drop in performance and thus reliability on them. If the robot takes too much time to move, or often confuse items to identify, it would be pointless to rely on it. In addition, a robot also needs to cope with the ever-changing difficulties generated by real-world conditions such as opened/closed doors, moving objects (*e.g.* humans), or luminosity throughout the day that may prevent it to accomplish its task.

1.2 Visual Analytics for Model Interpretability

Visualizations, and especially Visual Analytics, are well suited to ease access to relevant information leading to a model's decision while conveying intuitive interpretability—*i.e.* the "I know it when I see it" statement [153, 56]. This is due to the fact that there is not a stereotypical explanation for a model's decision [189], and thus in many cases, experts need to rely on multiple blocks to build their hypotheses on how a model reached a decision in order to interpret it. Visual Analytics enables humans to explore large data spaces, such as models parameters or outputs. This is particularly useful during early exploration stages where problems are unknown and thus the research space is vast. Visual Analytics is potentially efficient for model builders to decide on what data it will train on, what functions the model will need to optimize, and thus ultimately what constitutes a good behavior for the model. Therefore, model builders are in the front line of the efforts for bias reduction and model improvements. However, to understand their models and their decisions, experts need to go through millions (and sometimes billions) of parameters per decision which is not feasible in a reasonable amount of time. The exploration of any hypotheses they draw on a model's behavior can then be carried through Human-computer interactions to associate parts of the model as rules leading to a decision. Finally, any insight gained needs to be

extended to a complete dataset by gathering statistical evidence, in order to grasp more global tendencies in models' behaviors beyond single decisions.

One aspect of visual analytics systems is that they are usually designed for specific tasks and users. For instance, experts require in-depth analysis of models' decisions which can lead them to compare different hypotheses through models' parameters. This can be too overwhelming for end-users who are more drawn towards one main explanation, often obtained through guided visualizations design to confirm a model's decision rather than explore its source in its inner workings. While it is capital that end-users understand and are able to contest decisions of models, those models require nonetheless experts' insights to be improved and then deployed to real-life situations. Therefore as a first mission to reach interpretability, this thesis focuses on designing interactive visual analytics systems for experts analyzing their models on specific tasks and datasets. Chapter 6 discuss how those systems may be generalized to other models and tasks. While most of the contributions in this thesis are dedicated to experts, we also provide complementary works dedicated to the introduction to Deep Learning methods to non-expert audiences through interactive web pages—referred to as explainables or explorables [5, 63].

1.3 Thesis Overview

The rest of the manuscript is organized as follows: first, Chapter 2 provides a background of the visual analytics and interpretability works related to this thesis. Then, as depicted in Figure 1.2, the following chapters are ordered around the robotic challenges of designing models able to answer open-ended human-readable questions such as: *"where are my keys?"*. Chapter 3 covers step ①, *i.e.* the ability for models to answer natural language questions on given images, using computer vision. This chapter includes the design of VisQA, a visual analytics system in which users can browse the inner structure of transformer models to identify shortcut-biases, and leverage such insights to improve the reasoning of their models. Then, Chapter 4 addresses step ②, the challenge of using Deep Reinforcement learning approaches to tackle automatic navigation, along with examples on how visualization systems we designed help experts interpret a model's decisions. The *ego-pose localization* step (step ③) is covered in Chapter 5, in which we also explore how those models may face performance gaps when deployed to real-world's ever-changing conditions. Finally, Chapter 6 presents the current challenges and perspectives for future works of this thesis.

This manuscript relies on the following published material:

Chapter 3

- Théo Jaunet, Corentin Kervadec, Romain Vuillemot, Grigory Antipov, Moez Baccouche, and Christian Wolf. "VisQA: X-raying Vision and Language Reasoning in Transformers". In: *IEEE Transactions on Visualization and Computer Graphics (TVCG), 2021.*
- Corentin Kervadec, Théo Jaunet, Grigory Antipov, Moez Baccouche, Romain Vuillemot, and Christian Wolf. "How Transferable are Reasoning Patterns in VQA?". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2021.*

Chapter 4

- Théo Jaunet, Romain Vuillemot, and Christian Wolf. "DRLViz: Understanding Decisions and Memory in Deep Reinforcement Learning". *Computer Graphics Forum (Proceedings of EuroVis 2020), 2020.*
- Théo Jaunet, Romain Vuillemot, and Christian Wolf. "What if we Reduce the Memory of an Artificial Doom Player?". *IEEE Workshop on Visualization for AI Explainability at IEEE VIS (VISxAI), 2019.* This work was awarded "best-paper".

Chapter 5

- Théo Jaunet, Romain Vuillemot, and Christian Wolf. "Théo Guesser". *IEEE Workshop on Visualization for AI Explainability at IEEE VIS (VISxAI), 2020.*
- Théo Jaunet, Guillaume Bono, Romain Vuillemot, and Christian Wolf. "SIM2REALVIZ: Visualizing the Sim2Real Gap in Robot Ego-Pose Estimation". *NeurIPS XAI Workshop on eXplainable AI approaches for debugging and diagnosis, 2021.* This work will be extended in a follow-up journal.

CHAPTER 2

RELATED WORK

Contents

This thesis is related to designing interactive visual analytics systems to assist experts in gaining knowledge on their trained deep neural networks by interpreting their decisions. This can help them get a better understanding of their neural networks, and ultimately leverage this knowledge to improve them. This chapter first provides in Section 2.1 a broad overview and definition of interpretability, along with other main concepts. Then, Section 2.2 depicts the building blocks of visually interpretable trained systems, and finally, Section 2.3 introduces how these blocks are combined to form elaborated visual analytics tools. Additional related work will be provided in the next chapters, to cover specific Deep Neural Networks (DNN)s and application domains.

2.1 Definitions

The following sections present the definitions of the key terms **visual analytics**, **interpretability**, and **explainability**, often used interchangeably or assimilated as

interpretability, with the lens of different communities and backgrounds within the scope of Machine Learning. Along with those terms, follows a broader approach of such a discipline as perceived by other domains (*e.g.* biology) known as **explanation**, and how it may benefit the Machine Learning (ML) community.

2.1.1 Visual Analytics

Visual analytics studies decision-making processes and analytical reasoning, and how they can be assisted by the design of interactive visualization systems [45]. Initially, visual analytics emerged from information visualization and diverged to focus on the complete analysis of decision processes by combining visualization, human factors, and data analysis. Visual analytics heavily relies on humans for tasks that cannot be automated, *e.g.* to decide what piece of information may be relevant [101], and formulate hypotheses on its influence on a decision. The popularity of visual analytics systems, lead in 2007 to the creation of IEEE Visual Analytics Science and Technology (VAST), a Symposium of the IEEE 2006, which was held jointly with the IEEE Vis conference, one of the main conferences of visualization. Since 2021, VAST, along with infoVis and sciVis merged to IEEE VIS. As stated by Hohman et al. [87], thanks to early endeavors of visualization and ML communities, the popularity of visual analytics soared when it was leveraged for the interpretation of deep neural networks. While Visual Analytics systems can provide insightful information on how inspected models behave this domain is still in its early stage, with challenges to overcome [101, 87].

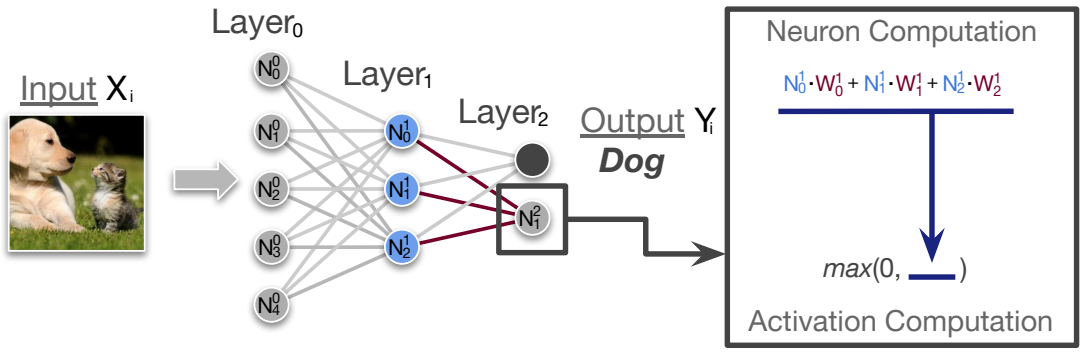


Figure 2.1 – Illustration of the structure of a Deep Neuron Network. With the input image X_i, the neurons of the model, arranged in inter-connected layers, yield the output Y_i "dog". This output is the outcome of a multitude of neuron computations. As depicted on the right of this figure, each neuron (*e.g.* here N_1^2) relies on the result of neurons from the previous layer. A more formal representation of this computation is depicted in Equation 2.1.

2.1.2 Deep Neural Networks

DNNs are often considered as black-boxes with the unique purpose to convert inputs X into outputs Y. Formally this can be represented as the following function $Y_i = f(\Theta, X_i)$ in which Θ corresponds to a set of parameters, and i is an element of the sets X and Y. In practice, a DNN is composed of a multitude of simple linear functions—referred to as neurons—generally connected with non-linear activations. As illustrated in Figure 2.1, those neurons represented as circles, are arranged in interconnected layers. Each connection between two neurons of successive layers is weighted by a parameter W_i^l where l indicates the layer of the incoming neuron, and i is the index of this neuron within its layer. Formally, in i's simplest form as a fully connected layer, a neuron (*e.g.* N_0^{l+1}) and its incoming weighted connections can be represented as it follows:

$$N_0^{l+1} = \sum_{i=0}^{i} W_i^l N_i^l + b_i^l \tag{2.1}$$

Where N corresponds to the ensemble of neurons in the model, W the weights, and b a bias added to provide to models the ability to formulate more complex functions (*i.e.* avoid passing by $(0, 0)$ by adding a term). However, despite a multitude of interconnected neurons, a model can only formulate complex linear functions. Thus, to provide non-linearity, in DNNs, for each neuron, after the computation of Equation 2.1, the yielded value is then used in a non-linear function (*e.g.* sigmoid, tanh, or ReLU [2]). The ReLu function, widely used in the Deep Learning community, can be represented as follows: $ReLU(\lambda) = max(0, \lambda)$. Those functions, referred to as *activation functions*, are key for models to be able to approximate elaborated functions, such as, for instance, image classification. And thus, the result of this operation is called activation, *i.e.* an intermediate result of the model computation towards an output which can be used to try to grasp the decision process of the model. Since the neurons of a model are interconnected by layers, all activations of a selected layer represent how the model considers the given input X_i. Such a representation is called an embedding, and, as illustrated in Section 2.2.3, embeddings can be used to capture how the model may construct an inner presentation of a given dataset.

Now that we have interconnected neurons capable of formulating non-linear functions, the next step is to be able to tune this Neural Network for a specific task, *i.e.* what we call training. Aside from adjusting a model's structure, *i.e.* adding more or fewer layers, or neurons, and swapping activation functions, the only element that can be changed in a model is Θ: its weights (W). In biology, Θ can be seen as the neuro-transmitters we have in our brain, dampening or increasing a signal from another connected neuron. Hence, in computer science, those weights (or parameters) are values that need to be adjusted in order to fit

the desired task. To do so, models are first initialized with random weights, and then start working—*i.e.* convert inputs X into outputs Y. Obviously, with random parameters, the model is doomed to yield errors which can be evaluated with a loss function, for example, the Mean Squared Error (MSE) $\mathcal{L} = ||y - \widehat{y}||^2$, the most widely used in Deep Learning. Such a loss indicates how wrong the model is for a set of outputs y compared with the ground-truth optimal results $\widehat{y}$. With such information, the next step is to determine how one should adjust the model's weights to reduce $\mathcal{L}$. This is done using gradients computed from the partial derivative of the loss with respect to the parameters as follows:

$$\nabla_{\Theta i} = \left[\frac{\partial \mathcal{L}}{\partial \Theta_i}\right] \tag{2.2}$$

Here, i indicates the batch, *i.e.* a computational step after which the model yielded enough outputs to have a loss computed, and its weights updated. The resulting gradient ∇_Θ describes towards which direction the values of weights should evolve to reach a better result. However, if such changes were followed blindly, after many iterations the weights of the model could oscillate around its optimal configuration without ever reaching it. Thus, to limit how much the weights of a model may change, and help it find its optimal solution, the gradient is multiplied by the hyper-parameter σ (*i.e.* a parameter manually set by a human), a single value, called the learning rate. Typical values for a learning rate range over orders of magnitude from 1×10^{-2} to 1×10^{-6}. The smaller the value, the longer the model will need to be trained, but the more precise it will be. Hence, usually, there is a trade-off to be found between performances and training time [1].

In the previous paragraphs, DNNs were introduced with the lens of Multi-Layer Perceptron (MLP)s, the most straightforward structure of models. However, there exist multiple types usually preferred for different types of inputs. Usually, for image processing, CNNs are preferred over MLPs as they are able to preserve local spatial coherence, and hence have fewer weights than an MLP would have for the same input. Because convolutions are shift equi-variant operations and share weights spatially, their usage lead to a significant reduction of the number of parameters needed. In this case, as illustrated in Figure 2.2, we refer to the weight of a model as filters which are convolved with the inputs for the first layers or with the previous activation. Similar to MLP the result of a neuron, despite being a matrix, is processed by the same activation functions. In many cases within the Deep Learning (DL) community, and in this thesis, CNN models often include some Fully Connected (FC) layers—*i.e.* MLP usually considered as the core of those models' decisions. This thesis also uses both Recurrent Neural Network (RNN)s and transformer models, whose design and structures particularities will be

1. Initially large values also provide a regularization effect that may help to avoid some local minima.

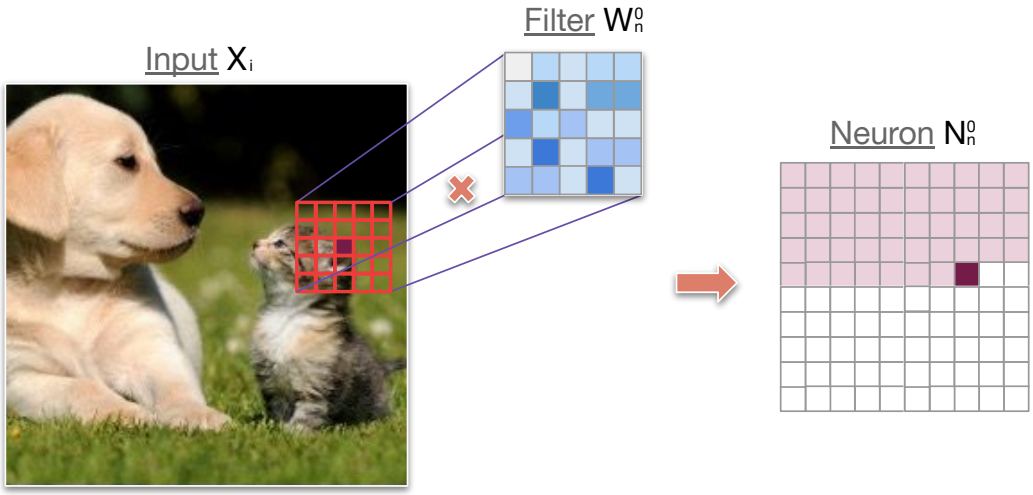


Figure 2.2 – Illustration of the convolution operation within the first layer of a Convolutional Neural Network (CNN) model. Given an image x_i, the model applies a filter W_n^0 of learnable weights over the complete input, which yields the values that the neuron N_n^0 will convey to an activation function, and the next layer.

further introduced within concerned chapters, namely Chapter 4 for RNNs, and Chapter 3 for transformers.

2.1.3 Interpretability

Historically, the concern of understanding models emerged in the '80s by studying expert systems [137], Bayesian networks [113], and more lately in the '90s, by studying neural networks [7]. However, the domain only gained popularity during the past decade along with the soaring performances and complexity of Deep Networks. Despite such a history, and a large amount of literature of different approaches and definitions to what may be seen as understanding a model, a broadly accepted definition is yet to be discovered. While a vast majority of those definitions centers around humans (*e.g.* "the ability to explain or to present in understandable terms to a human" by Doshi-Velez and Kim [56]), they are torn on what it should focus on. They either focus on:

- **The decision** (*i.e.* output) of a trained model, analyzed after-the-fact (referred to as post-hoc.
- **The inner workings** and designs of models, describing, for instance, how many layers should be used—sometimes referred to as transparency.
- **The mapping of data**, which addresses how models can tie together a dataset of inputs to outputs—also called representation [126].

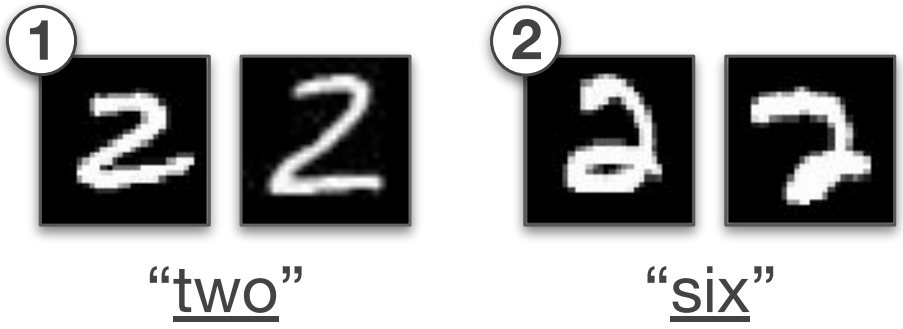


Figure 2.3 – On the left ①, two examples of an image containing a digit which have been correctly evaluated as "two" by a DNN. On the right ②, two other images which according to the MNIST dataset also contain hand-written "two". However, the same DNN fails and labels those images as "six". Such mistakes may be more acceptable to Humans, as those ambiguous images may even induce us to estimate that they contain sixes.

This difference may affect the design and purpose of approaches to interpretability. For instance, to focus on a single decision of a trained model, one needs to design a system capable of analyzing this decision along with its input, otherwise it may be irrelevant. To illustrate this, let's consider the analysis of a decision of a Deep Network trained to detect hand-written digits in a given image. With the images shown in Figure 2.3 ①, the model predicts that they correspond to "two"s. By looking at the image, one may indeed understand that those digits correspond to twos, and hence that the model is well trained. In contrast, with images of ambiguous digits, where even for humans it is hard to recognize the corresponding digit, (Figure 2.3 ②), the model yielding "six" instead of "two" is rather understandable. Hence such a mistake may be acceptable to users who may gain more trust in it, as the model made a mistake that a human also would [172].

In contrast, to focus on the inner workings, one may either target the structure of models [231], or the learned parameters [216, 222]. Both of those targets do not rely on the analysis of inputs or outputs. Hence, when analyzing the same model that outputs the hand-written digit in images, with the lens of inner workings, one only relies on the model itself. As an example, this can be done by collecting the parameters of the model, and browsing through them to try to grasp the shapes and colors patterns it may be sensitive to (*e.g.* white curves which may be seen in twos). This, however, is inherently much more difficult for Deep Networks (due to the number of parameters, and abstract data they manipulate) than, for example, for rule-based expert systems.

Finally, interpretability, within the scope of mapping of data, arises from the assessment that models are or should be considered as black-boxes where

only their inputs and outputs may be understandable to humans. And hence that interpretability endeavors should not focus on shedding light on the inner workings of a model, which cannot be understood, but rather on how it processes data. As an example with the same hand-written digits recognition model as earlier, such a model would be fed with a plethora of images and the conducted analysis may consist in observing what the model may consider as similar images, or images with which the model is undecided.

2.1.4 Interpretability Vs. Explainability

Despite being used interchangeably by the community, and carrying a similar objective (*i.e.* to grasp an understanding of Artificial Intelligence (AI)s' behaviors), the terms interpretability and explainability may convey different meanings. For instance, according to Rudin [175], interpretability comes through the design of machine learning models (*i.e.* is intrinsic) either by using Deep Networks trained to communicate their decision process (*e.g.* in natural language), or with simpler models such as rule-based ones in which a path resulting in a classification can be followed by Humans. In contrast, to him, explainability comes once a model is trained, either with the usage of a simpler model, *e.g.* rule-based, trained to "mimic" the behavior of a more complex Deep Network. To him, explainability can also be reached by using methods after-the-fact (*i.e.* post-hoc) to analyze a decision of a neural network. Roscher et al. [174], indicate that interpretability refers to elements that inform humans about a models' decision process, and that the explainability of models, is the human-centric approach of using those interpretable features to grasp the causes that lead to a decision.

In this manuscript, we suggest the burden of analysis as a distinctive feature between those two concepts. In interpretability, the burden of exploration and analysis of a model's behavior is tackled by the user, while in explainability, information that may help understand a model's behavior is presented to the user. While the distinction between those terms is thin, we argue that this reinforces the distinction between targeted audiences, and thus the design of solutions. As an example, experimented users may be more inclined to analyze models themselves, while end-users may seek more accessible information. Such a distinction between those terms will be carried through the manuscript, which focuses on a human-centered approach to assist domain experts, often builders of models, to interpret the decisions of their models by exploring their inner workings visually, and through human-computer interaction.

2.1.5 Explanations

The quest for interpretability is a path also explored outside of Machine Learning and Computer Science, in domains such as law, biology, social or cognitive sciences. However, according to Mittelstadt et al. [143], in ML, the most recent discipline, interpretability endeavors do not rest on the shoulders of giants to leverage already existing methods and definitions from those domains, hence the emergence of a gap between them. For Miller [140], closing such a gap would benefit the ML community, along with focusing on *everyday explanations*. To him, *everyday explanations* target local "why-questions", *e.g.* "why did the model provide this output?", as opposed to *scientific explanations* which may address more global approaches *e.g.* "What did the model learn?". The benefit of everyday explanation is that it may encourage users to build trust as they may be able to more easily identify causes for a decision. This is due to the fact that *everyday explanations* are expected to convey key causes leading to a decision, as a human would, rather than providing the complete chain. Such explanations are also encouraged by the fact that human cognitive functions tend to be overwhelmed when there are too many elements to focus on, and hence, humans tend to be more reserved towards complete explanations with many parameters to account for.

While there is no consensus on definitions in machine learning interpretability, the term "explanation", is often described as addressing a model's outcome individually by using both the input and the models' perception of outputs (*e.g.* class discriminative features) [84, 147]. This more broadly falls in post-hoc interpretability area, with the exception that explanations focus on one instance at a time. Following these works, throughout the manuscript, we will refer to an explanation as an answer to "why" questions. For instance, " Why did the autonomous car decide to hit the breaks?". An answer could be because a traffic light turned red, forcing by law any car to stop there. Another explanation could be that a vehicle ahead was stopping, thus breaking was mandatory to prevent any collision. We refer to post-hoc interpretability as the collection of methods that may shed the light on this ambiguity, for example, by addressing questions designed to verify potential causes for a decision such as "has the car seen the traffic lights?", thus potentially leading to the explanation to "Why did the autonomous car decide to hit the breaks?". An explanation may rely on multiple interpretations which might co-exist in order to grasp the behavior of a model. In addition, explanations can be erroneous, biased, or victim of human over-interpretation, hence the need for multiple explanations and tools to analyze them.

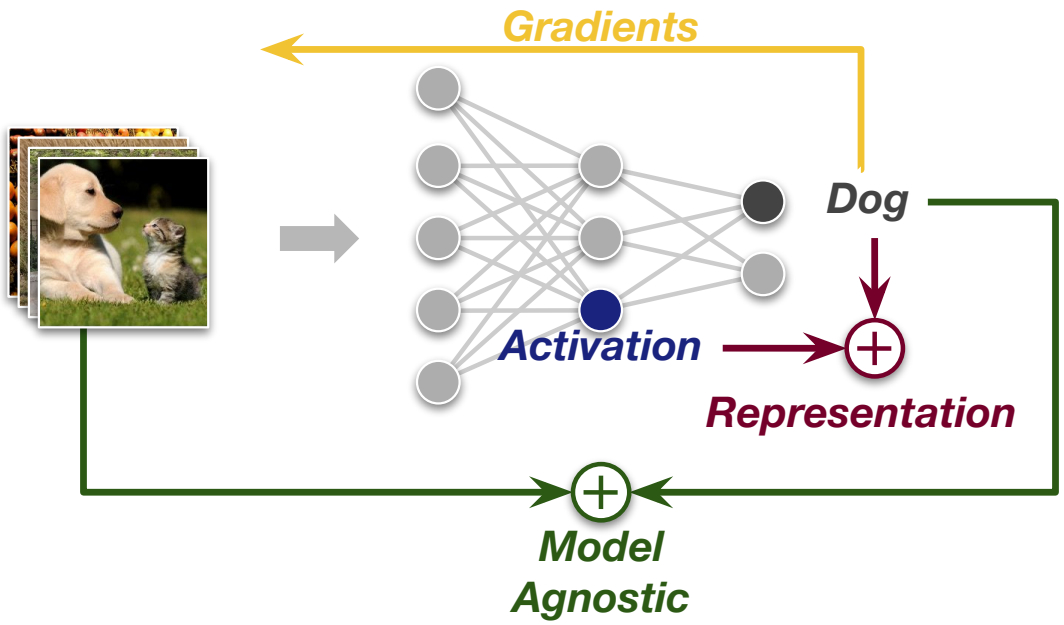


Figure 2.4 – Building blocks for Deep Learning interpretability are designed to provide an understanding of a model's decision or behavior. Here illustrated using an MLP, they can be divided in four categories. For a given input, a model produces intermediate results (**Activations**), to reach an output which can be used either by derivation to produce **Gradients**, or with output to produce **Representation**. Finally inputs and outputs can be combined to yield **Model-agnostic** interpretations.

2.2 Building Blocks of DNN Interpretability

Interpreting Deep Neural Networks (DNN) is often tackled with respect to their application domain (*e.g.* image, or text), and their type such as, for example, MLP, CNN, or RNN. The following presents some of the building blocks of visualization for Deep Learning interpretability, *i.e.* individual approaches designed to provide an understanding of a model's decision or behavior. As illustrated in Figure 2.4, those building blocks are here presented in four categories, namely: **Activations** which focus on the intermediate results extracted from the inner operations a model does to reach an output, **Gradients** produced when deriving an output with respect to either the input, or activations, **Representation** which addresses tendencies and global behavior of models through their embeddings, and finally **Model-agnostic** methods leveraging inputs and outputs of models to yield insights on their behaviors. Those building blocks of visualization which mostly emerged from ML communities, and are presented in ML, are detailed in the following sections.

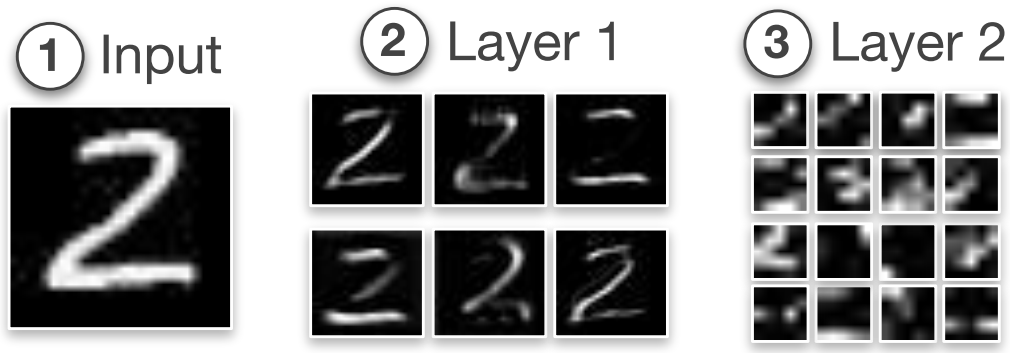


Figure 2.5 – Comparison of activations from a simple CNN model (Lenet5 [116]) trained to identify the hand-written digit in its input image. We observe that for the input image of a "2" ①, the activations (the whiter the higher) of the first layer are sensitive to the overall image, and shape of the digit ②, whereas activations from the last layer are more abstract shapes harder to analyze ③. In this figure, images were resized for the sake of readability. Initially, both the image input and first layer activations were 28×28, while layer second layer activations were 5×5 pixels.

2.2.1 Activations of Neurons

As introduced in Section 2.1.2, and illustrated in Figure 2.1, during their execution Deep Neural Networks (DNNs) provide intermediate results referred to as activations. Those activations can be visualized for local explanations of what regions of the input a model may be sensitive to. Hence the reason why to analyze them. In computer vision with CNNs and images as input, activations are often visualized as images describing how a neuron might be sensitive to the shapes or colors of the input image. Usually, the most targeted neuron activations to visualize are those within the first layers. This is due to fact that these activations are directly tied to the image input, and thus the activations can be interpreted in pixel space, *i.e.* be compared with the input image. Since layers in CNNs are sequential, *i.e.* the layer $n-1$ provides inputs for the layer n, the deeper in the model activations are sampled from, the more abstract and smaller they become, making chances to interpret them grow thinner. As illustrated in Figure 2.5 ②, with a CNN model designed to classify digits within an image, the activations (the whiter the more intense) from the first layer can be assimilated close to the input image ①, whereas activations for the last CNN layer are harder to analyze as they are. Those activations, however, remain a vast majority within DNNs, hence the raising desire to understand what information they may convey throughout the model.

Figure 2.6 – Left ①, the Top-k image with the highest activations per neuron. Each row is a neuron, and each column from left to right corresponds the top-10 images. We can, for example, observe that the neuron corresponding to the first row might be sensitive to circular patterns such as dog eyes and snouts. Credits for this sub-figure go to Springenberg et al. [190]. Right ②, examples of overlap between a manually annotated segmentation dataset, and a model's activation from 4 neurons. We can observe that the neurons corresponding to the two rows on the left are sensitive to houses. Hence indicating that some neurons may function as "object detectors". Credits for this sub-figure are due to Bau et al. [16].

The visualization of features from deeper layers can be addressed with other approaches, however, many of them revolve around the challenge of contextualizing them with respect to their input, *e.g.* display them over input image. For example, as depicted in Figure 2.6, to grasp to what shape or color a neuron may be sensitive to, one can do a forward pass on each dataset sample and can collect the activations of this neuron. Then, the next step is to sort those activations by their intensity, *e.g.* in the case of CNNs with a matrix as activation noted a, with n rows and m columns the following sum:

$$\sum_{i=0}^{n}\sum_{j=0}^{m} a_{ij} \tag{2.3}$$

The goal is to only preserve the top-k activations and their corresponding input image. Then, by visualizing those images one may grasp patterns and trends in the top-k images such as objects (*e.g.* cars) or shapes (*e.g.* circles) which may convey what the neuron may be sensitive to. Thanks to this, as illustrated in Figure 2.6, one may observe that some neurons may be sensitive to circular patterns, while others to text, or lines. Using this approach, Zhou et al. [243] have studied some neurons of their analyzed CNN model were used as "object detector", *i.e.*, where only highly activated when a certain object (*e.g.* a lamp) was within the input image. This can also be tackled in the opposite way, *i.e.* by selecting an activation map from a neuron, and evaluating its upsampled Intersection over Union (IoU),

with an annotated segmentation dataset of objects [16]. That way, if the activation map completely overlaps the regions of the image in the dataset corresponding to an object, one may conclude that the corresponding neuron might be sensitive to it. As depicted in Figure 2.6 we can observe a strong correlation between activations of neurons, and the presence of "house" or "dog" within the inputs.

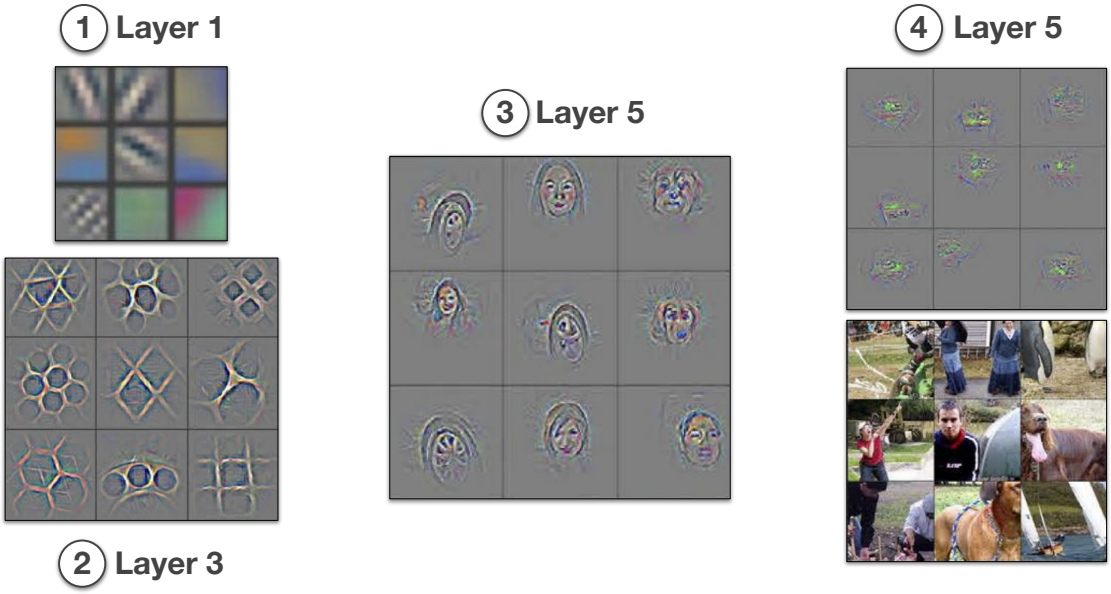


Figure 2.7 – Visualization of activations of an image classification CNN using DeConv [239]. This figure represents a manually selected sub-set of neurons which are displayed in a 3×3 grid of their most active images. We can observe that the first layer ① contains patterns such lines and color gradients, the second layer ② seems to seek for textures or patterns, while the layer 5 ③ seems to responsive to more complex image feature such as faces. In ④, we combined with the display of the top-k images one make sens of abstract DeConv activations, *e.g.* the grass in the background. Activations in this figure were sampled from [239].

While those methods yielded promising results, they remain subject to humans' interpretations to either grasp patterns within the top-k images which may not be always straightforward or limit the range of activations to only visualize those that match previously annotated objects. An alternative, introduced by Zeiler et al. [239], is to convert those activations from deep layers back to the input space, *i.e.* with the size of the image, and its highlighted regions corresponding to what a selected activation might be sensitive to. To do so, such an approach, named Deconv [239], implements another inverted model, in parallel to a traditional CNN model, with the purpose to convert any activation back to the input. This inverted model samples activations from a selected layer, and nullifies (*i.e.* set their values to 0) all of them except the one analyzed. As illustrated in Figure 2.7,

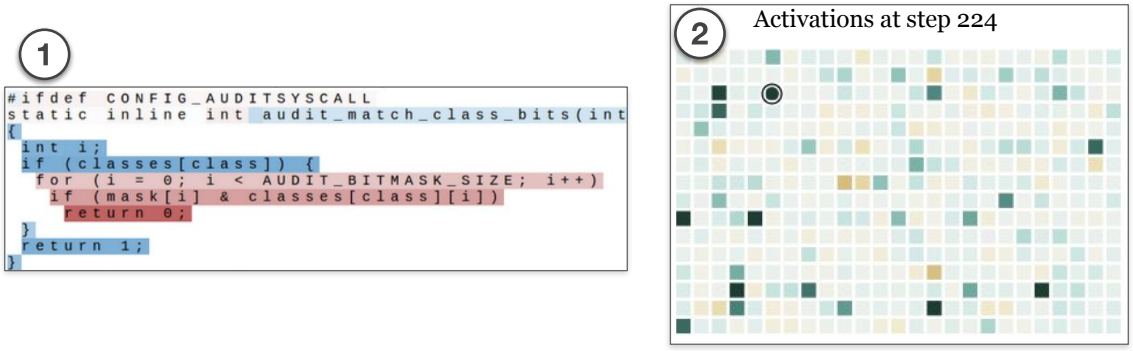


Figure 2.8 – Left ① visualization of a manually selected activation of a hidden state of a recurrent model. Such activation is displayed over its input text ranging from blue (negatives values) to red (positive values). It can be observed that this activation seems correlated with the level of text indentation (credits to [100]). Right ②, visualization of a complete hidden state as a grid. It can be observed that only a handful of activation are high at the same time, and thus that they may be used as "functions" representing different elements of the inputs. Sampled from [35].

this results in an image, the same size as the input with shapes highlighting what triggered the selected activation. It can be observed, as mentioned earlier, that the activations of the first layer of the model tend to be responsive to lines and color hue ①. However, as the activation are sample deeper within the model, they become complex, with shapes such as dog faces, or certain textures in layer#5 (Figure 2.7 ③). As illustrated by Zeiler et al. [239], those visualizations reach their full potential when combined with the top-k images maximizing an activation method. This can help humans to make sense of otherwise hard to grasp patterns resulting from Deconv, by helping humans grasp the common factor within top-k images those activations may correspond to. For example, in Figure 2.7 ④ using the combination of those methods, it has been concluded that this neuron from layer#5 might be sensitive the presence of grass in the background.

With Natural Language Processing (NLP) tasks addressing textual input (*e.g.* translation), one of the main challenges for models is being able to retain information seen from previous words of a sentence which may still have an influence on the current input. To do so, we rely on RNNs, another kind of model which maintains a recurrent inner state (called hidden state) that updates after an input. Formally, this can be represented as it follows: $h_t = \Phi(h_{t-1}, \Theta_\Phi, a^n)$ where h_t is the current inner state, h_{t-1} the one from the previous input, and a^n represents all activations from the layer n, to which the recurrent layer is connected. Hence the hidden state is a time-varying vector which is often represented as a heatmap that can be displayed over inputs (*e.g.* highlighting words and characters in sentences) [100]. As illustrated in Figure 2.8 ①, thanks to such a visualization, it has

been discovered that the hidden states of those models are able to model hierarchy in texts such as indentation, may be able to count characters in sentences without being asked to, enabling them to convey information from inputs outside of is expected from their design. Heatmap visualization of hidden states can also be applied to trajectories in which some activations activations are sensitive, among other, to trajectories' direction [35] (Figure 2.8 ②). Thanks to this it has been discovered that their model was, for example, sensitive to trajectories going up or down. More recently, NLP tasks are addressed with attention models [218], whose attention, *i.e.* activations, can be visualized as bipartite graphs [155], and heatmaps [176, 12] to grasp how a model may associate concepts and words together while providing an output. Chapter 4 and Chapter 3 are respectively to the visualization and analysis of activation in RNNs, and transformers. Furthermore, as depicted in the following section, methods to visualize activations can be further improved when used jointly with back-propagated gradients.

2.2.2 Visualization with Gradients

As introduced in Section 2.1.2, gradients are at the center of neural algorithms' learning, they are used to update model weights with respect to their output, thus it is only natural to visualize them. However, a multitude of gradients can be computed from DNN models, and the gradient computed with Equation 2.2, may not be the most suitable in the scope of interpretability. This is due to the fact that it only indicates how the model's weights may need change to better bring closer its outputs y and ground-truth labels $\hat{y}$, and for large models there may be billions of them! Thus a human may have a hard time understanding the meaning of those changes. More suitable information, would be to understand what pixels of an input image x_i are the most influential for a model to yield its output y_i. This can be done with the following gradient:

$$\nabla_{xi} = \left[\frac{\partial y_i}{\partial x_i}\right] \tag{2.4}$$

In other words, how a change of pixels values of the input, may affect the output. A key feature of this gradient is its ability to transform this information into a medium that a human can make sense of, *i.e.* the input image. This is referred to as saliency maps [187], in which the gradient ∇_{xi} can be converted into an image and displayed over the input to display relevant parts of the image for a selected output. The downside of such an approach is that in many cases a majority of the image is highlighted, making it difficult to assess what particular points drew the model's attention. To tackle this issue, guided-backpropagation [190] (illustrated in Figure 2.9 ②) proposes to clip to zero parts of the gradient's contributions to other classes, *i.e.* those that may be negative near the output, but positive throughout

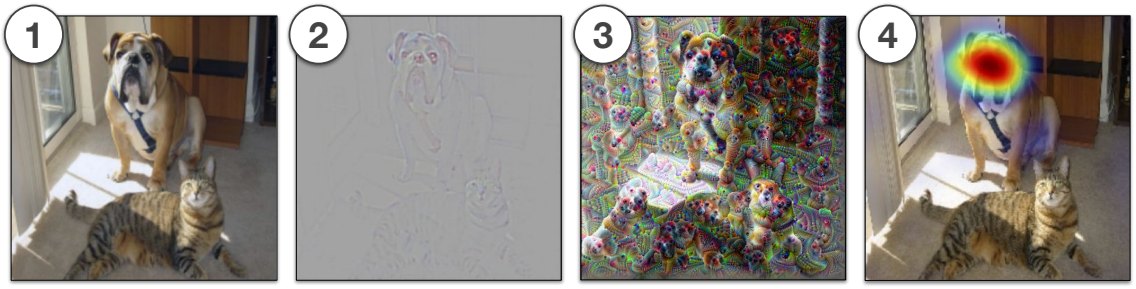


Figure 2.9 – Examples of visualization with gradients-based building blocks. From left to right, ① with an input image, ② guided back-propagation, class [190], ③ optimization of input [136] targeting the class "dog", and ④ Grad-Cam [182] highlight pixel assimilated to the class "dog".

the model. This results in the highlight of only key pixels to classify images, such as, for example, the snout of a dog. By doing so, in contrast to activation-based methods, one can assess the pixels of an image that may be associated with a class not even targetted by the model. Integrated gradients [199], propose to tackle the saturation of gradients by using as input the integral interpolation of an image and a baseline (usually a black image). This approach yields visualizations in which features of images such, for instance, a dog's snout are more reflected. This is due to the fact that in many cases, models can successfully classify an image indistinguishable by humans (*e.g.* with pixels having less than 10% of their original value) [195], hence the saturation of gradients.

The gradient ∇_{xi} extracted from Equation 2.4, can also be used to update the input image with respect to the model's output. The intuition behind this is that, by modifying the pixels responsible for the model's output, one may ultimately grasp what patterns the model may seek in an image to yield such an output. By doing this iteratively to alter an image (usually filled random noise) one can maximize an output to provide 100% confidence on a class [149]. As depicted in Figure 2.9 ③, this results in the creation of abstract artifacts of key class's features all over the input image such as, for instance, snouts for dogs [156]. This indicates that for the model to output the class "dog" for an image, the model needs, among others, to distinguish a snout. When used in combination with neuron activations, input optimization can provide insights on the role of each unit of a model, *i.e.* what each neuron is looking for in an image. To do so, instead of optimizing the input image to maximize the output, one can maximize the activation of a selected unit [20] as it follows:

$$\nabla_{xi} = \left[\frac{\partial a_n^l}{\partial x_i} \right] \tag{2.5}$$

Where a_n^l is the n^{th} activation of the layer l. Such an approach has been improved through regularization using an image prior [136], and generative models [151] to provide more human-readable images. Such an approach can also be applied to NLP, in which words of sentences can be optimized in order to grasp how a unit may associate words together [17].

To date, the most popular method to visualize gradients of CNN models with images as input is grad-CAM [182], which relies on the activations of the last convolutional layer to produce a heatmap over the image input highlighting the regions related to a particular class. Such a heatmap indicates the attention of the model, *i.e.* the regions of the input the model looks at in order to provide the output. Thus, humans can assess if the model focuses on the correct objects within images corresponding to the outputted class. This approach weights the activations of the last convolutional layer (a^l) using global average pooling of gradients issued from the derivation of the output y_i with respect to each activation of the last layer l indexed by n as it follows:

$$\alpha_{ln}^{y} = \frac{1}{Z} \sum_{j=0}^{w} \sum_{k=0}^{h} \left[\frac{\partial y_i}{\partial a_{jk}^{ln}} \right] \tag{2.6}$$

Where w, h are the width and height of the activation map. Here, α_{ln}^{y} is the weights of the neuron n of layer l extracted from the output y. Such a weight is then combined with activations, and passed to a ReLU function to only preserve positive values, *i.e.* those contributing to the output as it follows:

$$m^y = ReLU\left(\sum_n \alpha_{ln}^{y} a_n^l \right) \tag{2.7}$$

The resulting m^y corresponds to a heatmap, the size of activations of the last CNN layer, describing which regions of the input image are responsible for the model's output. As illustrated in Figure 2.9 ④, we can observe with the heatmap—the redder, the more intense—that according to the model the dog's face is a key feature for the model to output the class "dog". However, in order to be relevant, grad-cam needs for the activations to have a sufficient size, otherwise the resulting heatmap, which needs to be upsampled to match the input's size, may cover too much of the input, hindering any analysis. The size of activations is dictated by the model's architecture, and thus needs to be considered before using grad-cam heatmaps. Nonetheless, thanks to such an approach experts were able to grasp bias such as, for instance, how a model exploits humans' faces to predict their job, instead of their surroundings and tools they manipulate [182].

2.2.3 Inner Representation of Data

As introduced in Section 2.1.2, like a funnel, as the dataflow reaches deeper layers of the model, the amount of parameters grows thinner. Because of this, the model needs to build an *inner representation* of inputs in its own abstract space. Thus, the complete set of activations with one layer, noted A^l, describes how the model perceives the given image, *i.e.* what information it deemed important to convey. This is referred to as an embedding and its core to the analysis of models with the lens of **representation**. In such an analysis, one may seek to understand how the models behave globally, *i.e.* how the model may perceive the data that it is fed to, and how the model may associate data-points together. To do so, such methods often focus on activations in the last layer. This is due to the fact that it is argued that such a layer is the most suited to be analyzed since it should convey high-level semantics [182] which are then the premise of decisions made by fully-connected layers. Hence, the first step of any study of a model's inner representation of data is to pass a complete dataset through it and collect, for each data-point its corresponding embedding.

However, due to the nature and size of models, those embeddings are high-dimensional elements which makes comparison or interpretation by humans as they are, nearly impossible. Thus, to do so, the dimensionality of the embeddings is frequently reduced by techniques such as PCA, UMAP [135], or t-SNE [217]. The following describes the execution of t-SNE, the most popular method. First, it computes the probability for each pair of points to be neighbors, using the euclidean distance between them. This is formally represented as it follows:

$$p_{j|i} = \frac{exp(-||x_i - x_j||^2/2\sigma_i^2)}{\sum_{k \neq i} exp(-||x_i - x_k||^2/2\sigma_i^2)} \tag{2.8}$$

Where $p_{j|i}$ is the probability that the embedding x_i picks x_j as neighbor, when embeddings follow the probability density under a Gaussian centered at xi of variance σ_i^2. Then this needs to be converted into a low dimensional space (the visualization space, usually 2D or 3D) using the following approach:

$$q_{j|i} = \frac{exp(-||y_i - y_j||^2)}{\sum_{k \neq i} exp(-||y_i - y_k||^2)} \tag{2.9}$$

in which y_i and y_j represents the data points x_i and x_j in this low dimensional space, and $q_{j|i}$ the probability that they are neighbors. Hence, the objective of t-SNE is to minimize the difference between $p_{j|i}$, and $q_{j|i}$ to 0 which is a synonym of a perfect mapping between those two spaces with respect to the distance between x_i and x_j. Such minimization is computed using the sum of Kullback-Leibler divergences on embeddings as follows:

$$C = \sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{j|i} log \frac{p_{j|i}}{q_{j|i}} \tag{2.10}$$

This yields C, the cost function of all probabilities $P|Q$, which is then used to update data-points in low-dimensional space following a gradient-descent approach $\frac{\partial C}{\partial y_i}$. Intuitively, this can be seen as data-points attracting and repelling each other in order to minimize their overall distance when their $p_{j|i}$, and $q_{j|i}$ are too different. t-SNE is a stochastic and non-parametric approach, which means that it must be re-executed in order to add new data-points to the representation, and doing so may yield different results.

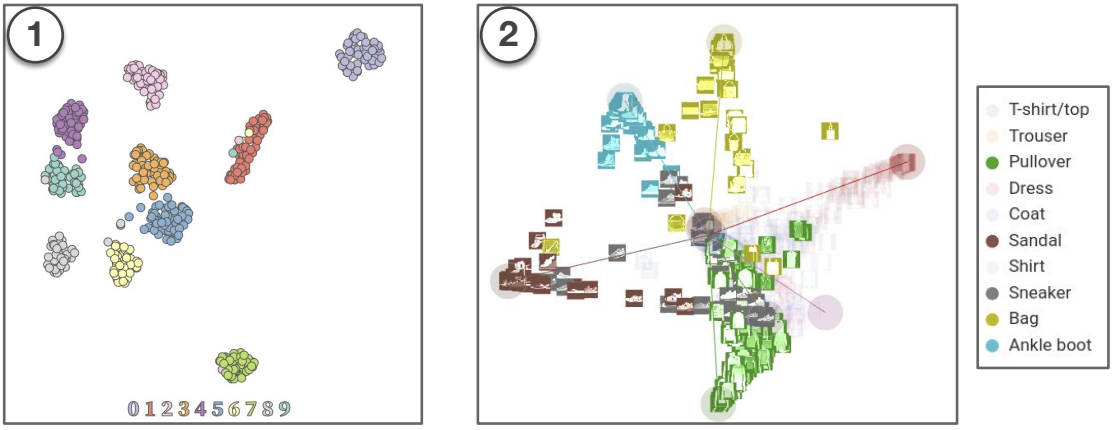


Figure 2.10 – Examples of visualization with the representation building blocks. From left to right, ① t-SNE representation of embedings on the MNSIT dataset, each dot is an input, and its color represents it class. ② Grand-tour of the MNIST-Fashion [122] with as many dimensions as the number of classes displayed on the most right of the figure.

As illustrated in Figure 2.10 ①, with such a dimensionality reduction method, DNN embeddings can be projected into two [23] dimensions. In such a visualization, each dot is an embedding, at its color encoding, for instance, its ground-truth class. In any case, the goal is to regroup similar embeddings in clusters, and eventually compare a selected instance to the rest of the dataset—referred to as explanation by example by Lipton [126]. The more a model is able to successfully divide a dataset into clusters of the same class, the more it has a good representation, hence indicating successful learning of classification. In contrast, in those clusters, we can also observe sources of confusion for the model by identifying inputs that are between two clusters, or within a wrong cluster. This can be complemented by sampling embeddings from inputs and outputs of an intermediate layer, instead of the last layer. By doing so, we can compare the scatter plot before

and after going through the layer, and thus observe the influence as the model produces outputs. Such an influence can also be observed using grand-tour [11], a projection of embeddings in n-dimensions. For instance, as seen in Figure 2.10 ② n equals to the number of classes, the model as to identify in its training. However, visualizing n-dimensions on a 2D screen is a challenge by itself. Grand-tour tackle this by relying on animation to rotate such space and emphasize a few dimensions at the time, and on interactions to enable users to manually arrange dimensions as they want, to pick which dimensions to visualize more easily at the expense of others. Using this, it has been observed how the separation of different concepts such as classes corresponding to shoes, and those corresponding to clothes may emerge from early layers [122]. The impact of training on models' inner representation can also be monitored using this approach by animating representation scatter plots after each training step to, hopefully, highlight the convergence of the model. By monitoring the coordinates of centroids from each cluster, after each training epoch, one can use such an approach to grasp an overview of a model's training [170]. While the study of inner representations through the help of dimensionality reduction techniques may yield interpretable results, due to the stochasticity of algorithms such as t-SNE and fine-tuning of hyper-parameters that may be required to optimize the final layout, we argue such visualization should serve as an entry-point to draw hypotheses from rather than a way to interpret a model's decisions.

2.2.4 Model-Agnostic Methods

As illustrated in Figure 2.4, another approach to reach interpretability of Deep Neural Networks is to consider them as black-boxes from which only inputs and outputs may be understood by humans. The advantage of such a building block is that it is model-agnostic, hence it can be applied regardless of a model's architecture, and in some cases even of the data they manipulate. A common method is to distill a trained model's knowledge into a smaller, and thus easier to interpret, model [203]. This relies on the argument that simpler models such as decision trees can reach sufficient performances to be deployed in real-world situations such as in healthcare [192] while providing to health professionals the ability to inspect their decision process. To do so, we can train a DNN, as we normally would, *e.g.* a large model with a lot of training data. Such a model may then reach high performances on its task, and learn to model patterns in its training data, key in distillation. Finally, the next step is to use a smaller model, *e.g.* a small DNN, or a rule-based model, and train it to mimic the bigger model's outputs on the same inputs. For instance, this can be done by altering the training loss $\mathcal{L}$ as introduced in Section 2.1.2, to $||y - y||^2$, where y' is the output of the smaller model, and y the output of the larger one. As a result, the smaller model not only learns to be accurate, as it would have using ground-

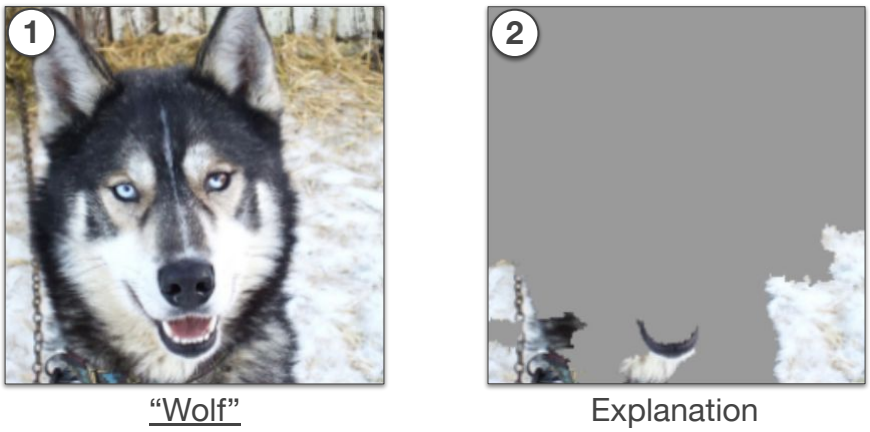


Figure 2.11 – Example of insight gained using LIME [172], when given the image of a husky ①, the analyzed model fails and predicts "wolf". By looking at LIME's explanation, with the gray areas corresponding to the "super-pixels" removed, we can see that the analyzed model relies on the presence of snow in the background rather than the dog to yield an output, hence the mistake. This example and images were sampled from [172].

truth labels but also how to behave, *i.e.* how to handle unexpected data, as the bigger would. In particular, the smaller model learns how the bigger model distributes errors over non-ground-truth classes, which contains a large amount of knowledge on its decision process [85]. That way, the decision process of a complex DNN can be summarized as rules to follow to grasp why, for example, a patient may have been classified as potentially carrying a disease (*e.g.* under 30 years old, and fever). Since we rely on smaller models, such a process may yield performance drops, their accuracy, hence distillation thrives when applied to situations where decisions' accuracy is of the utmost importance (*e.g.* healthcare). Thus, in many cases, any explanation provided by a model is required to be analyzed by a domain expert (*e.g.* a doctor) who evaluates, and sometimes fixes models' decisions.

However, despite promising results, and distillation being an active research domain for interpretability, there is an equilibrium to be found between a distilled model's accuracy, and its complexity that may make it inherently difficult to interpret. To illustrate, while each rule in a rule-based model might interpretable, the process that leads the creation of those rules, and paths to decisions remain unclear. This is emphasized as the number of rules increases, and hence when it becomes too laborious to follow the rules applied to a particular dataset [126]. Nonetheless, when applied locally *i.e.* for a single data-point, distillation can provide useful insights on how a model behaves.

Another way to analyze DNNs with a Model-Agnostic lens (*i.e.* using only inputs and outputs), is to apply disruptions over an input, and observe how they

may affect the output. This is the case for LIME [172] one of the most popular model-agnostic approaches for interpretability in the industry. In LIME, an input is altered multiple times. For instance, when applied to an image, it consists in dividing it into clusters of pixels using a segmentation algorithm, *e.g.* growing regions. Then, LIME computes the complete combination of this image in which those clusters may be replaced by their average value, gray or dark. Each variation is forwarded to the analyzed model, in order to collect its impact on the output (*e.g.* less confidence on the prediction of a class). This is then used to train a simple linear model to predict the analyzed model's output with respect to its altered input. By doing so, the linear model learns which super-pixels are key in order for the model to yield its output. Thus, it can be used to grasp the minimal number of super-pixels required for the model to reach an output similar to what it did with its unaltered input. As illustrated in Figure 2.11, using LIME, Ribeiro et al. understood how a neural network exploits the snow in the background of images to distinguish between huskies and wolves. Similarly, occlusion patches can be applied to images in order to visualize which regions of the image may increase or decrease the model's confidence for an output [239]. While being less precise than LIME, the advantage of such an approach is that it does not require a regression by a linear model, and hence can be faster as there are fewer combinations of inputs occlusion to evaluate.

2.3 Leveraging Building Blocks in Visual Analytics

While the aforementioned building blocks of model interpretability are crucial for the analysis of single elements (*e.g.* a neuron, or a class), it can be quite time-consuming to go through a large number of individual blocks to produce explanations [115]. However, in order for experts to obtain a broad understanding of the decision process of models, and how it may convey biases, they need to analyze different decisions which themselves can have multiple explanations [189]. Thus, raising the need for interactive interfaces to quickly probe building blocks. To this end, the community relies on Visual Analytics systems which often combine those blocks in multi-coordinated views to study decisions through the lens of different blocks at once [87]. Those systems which mainly revolve around activations, often include a representation block, such as a dimensionality reduction of embeddings. When targeting an expert audience, the purpose of those tools is often to draw a hypothesis of a model's behavior which then may be confirmed or denied through an in-depth analysis and statistical evidence external to the tool.

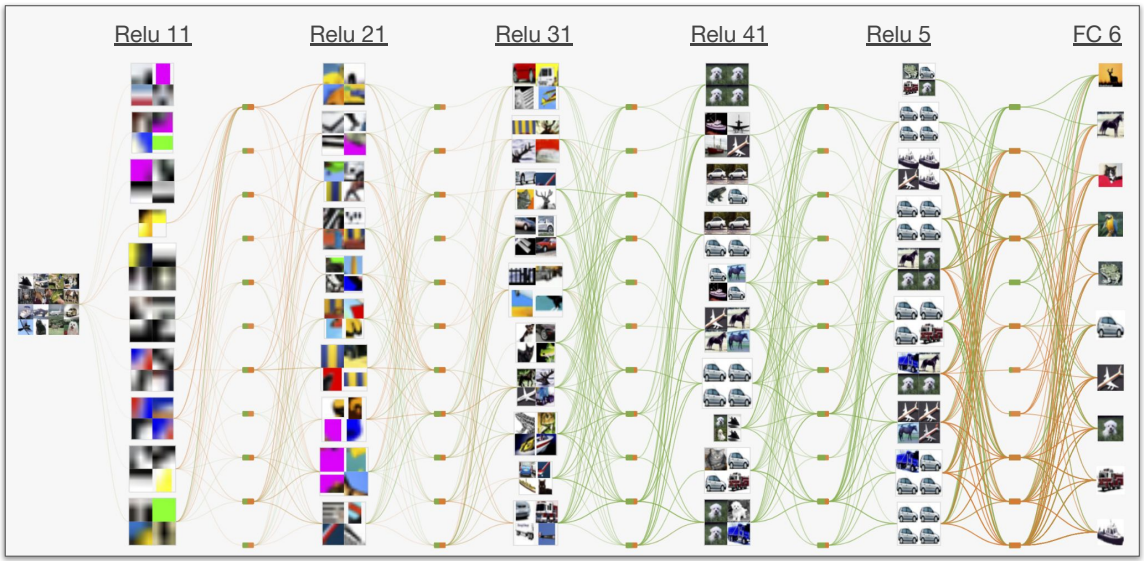


Figure 2.12 – Overview of CNNVis [128], a visual analytics system leveraging activation building blocks to display to what neurons may be sensitive to. This system follows, the model's architecture to display with the help of clustering methods, among others, the top-k images of neurons. This view was sampled from an online prototype [43].

2.3.1 Interactive Activations

By itself, the visualization of a single activation only provides cues on what triggers the inspected neuron. However, a model often relies on a large combination of neurons to produce an output, hence the need to understand how neurons interact with each other. To this end, early works provided an overview, in which users could browse the layers of the model, each depicted as a matrix with cells corresponding to the activation of a neuron for a given input (a picture sampled from a webcam in [236]). Similarly, in CNNVis [128] an activation overview display activations in a directed node-link diagram. Such a diagram follows the design of a parallel plot, in which, from left (inputs), to the right (outputs), each layer is a vertical axis, composed of activations of neurons within it, and interconnected to the previous layer and the next one. As illustrated in Figure 2.12, neurons are represented by the top-k images yielding the highest activation for the complete dataset. Such an overview helped experts grasp insights on how, for example, the model's gradients vanished during training. The interactions as designed in those systems, *i.e.* the ability to browse layers, and select neurons for further information such as alternative visualization, is a key feature for experts to actively debug their model. This is due to the fact that causes for a model to fail can be multiple, and occur at different levels, *e.g.* gradients throughout the model, or a neuron confusing similar features of classes. CNNVis targets datasets with a

limited amount of classes to output, thus raising scaling issues when deployed to larger datasets with thousand of classes such as ImageNet [52].

However, displaying each neuron of a model can be overwhelming, especially with the ever-increasing size of state-of-the-art models. Thus, more recent endeavors reduce the number of activations for experts to assess, and scale up those systems to larger models and datasets. In ACTIVIS [97], a layer can be selected by clicking on it in a model's structure view, similar to tensorflow's [231], and then, each neuron is displayed as a column in a table with instances as rows. That way, each cell encodes through a single hue-colored rectangle the intensity of an activation. Thanks to this, ACTIVIS provides traditional table interactions such as filtering and re-ordering to quickly probe interesting activations. In Activation Atlas [34], feature attributions, *i.e.* activation + gradients, of each neuron are displayed in a dimensionality reduction layout (representation block) to bring together similar activations of the same layer. This helped users grasp insights on how the model may tend to focus on the water in the background to separate "fireboat" and "streetcar" classes from ImageNet. One limitation of such a system, apart from its computational cost, is that each neuron is analyzed individually, while it has been argued that DNNs may learn a hierarchical representation of the data, and hence that they rely on a combination of neuron across layers to identify the class of an image [239]. To get a hierarchical sense of data flowing through convolutional layers and units, openAI's Microscope [158], arranged model activations in a node-link diagram following the model's structure. Such a tool can also provide details on-demand switching from layers from which only a summary of each layer is visible, to a unit view with every unit from a layer displayed, to finally a detailed view of a selected unit, with among others, the top-k images. Despite those endeavors, this system suffers from scalability issues as the size of models to analyze increases, going through all layers and units can be an overwhelming task. In order to tackle such an issue, SUMMIT [86] proposes to build a semantic graph that highlights the influential connections between neurons for a given image, along with an interactive representation block aside to provide an overview on how the model associates classes. This tool helped experts grasp, for instance, how a model distinguishes between the similar brown and black bears classes using neurons sensitive to the respected colors of furs and faces of bears.

2.3.2 Sequential Activations and Gradients

In domains the manipulated data is sequential. For instance, in NLP, with sentiment analysis in which words of a sentence need to be evaluated jointly. This raises new challenges as models may need to handle time-dependent concepts to yield accurate outputs, *e.g.* the position of words in a sentence to generate text. Such a sequential constraint on the data they manipulate impacts their design to

the point where those models incorporate time-varying activations which can, for example, convey information from previous inputs. Such a behavior hinders a straightforward comprehension of their decisions. Hence a clear understanding of which previous inputs weigh in when a model predicts a decision being at the stem of interpreting them.

In a vast majority of visual analytics systems addressing the challenge of interpreting models handling time-varying data, the focus is on those activations and how their intensity patterns may be correlated with inputs. For example, in LSTMVis [193], which design focuses on the study of the activations of recurrent models applied to the character and word prediction, activations are represented as lines with inputs as the horizontal axis, and intensity as a vertical axis. In this system, users formulate hypotheses as queries filtering activations matching a selected interval of inputs. Such queries can be guided by complementary information such as a Part-Of-Speech (POS) tagging of words, top-k predictions and, coordinated complementary heatmaps of data such as word count. Unlike most instance-based visual analytic systems, LSTMVis offers a comparison with similar examples (*i.e.* sentences) from a dataset, key to grasping patterns in sequential activations. In RNNVis [141], those activations are groups of clusters with similar intensity and represented as heatmaps linked to POS-colored word cloud visualization of inputs over the complete dataset. A key feature of this system is its glyph encoding (*i.e.* a combination of graphical objects to represent multiple attributes of a data-point) which summarizes the evolution of activations with respect to words in a sentence. This enables users to quickly compare the increase or decrease of aggregated activations intensity through bar-charts as inputs go through the model. In opposition, RNNBow [37] focuses on the evolution of gradients of a character prediction RNN model during training represented as bar-charts (one bar per training batch). Thanks to this, users can overview how the model may learn, and upon selection of a batch, understand how the model may be sensitive to previous outputs. RNNBow has been shown to be particularly useful to detect vanishing gradients *i.e.* the limit of how far in the past a model may be able to seek information.

More recently, models with attention [12] increasingly gained popularity due to their improvement of state-of-the-art performance, and their attention mechanism is considered more interpretable than recurrent models. Such attention is a form of activation which associates inputs together. Commonly, attention is represented in instance-based visualization as graphs with bipartite connections from which each side is an input sequence (*e.g.* a sentence), and thickness of connections between elements of this sequence (*e.g.* words) encodes the intensity of the association. In seq2seq-vis [194], such a view is complemented with a display of the top-k prediction for each input, which can at anytime replace an input with an interaction, and compared to the initial sequence of inputs. This system also provides complementary information such as a representation view

of neighbor inputs, along with a list of similar sequences containing a selected input.

Nowadays, attention models such as transformers [218, 54] exploit attention in modules used in each layer, as opposed to recurrent models with a single activation layer addressing the sequential aspect of its data. Thus, along with those models arose the challenge to visualize each of those attention maps—which can spread over hundreds of modules. To do so, BertViz [219] relies on interaction to enable users to quickly switch between modules, and within one module, to display on hover the connections of one input (words) to the others, while filtering those from other inputs. In order to guide users' interaction, such a system also provides an overview that displays each attention graph in a matrix in which rows are layers, and columns attention modules within a layer. Thanks to this, users can grasp patterns that may require further inspection. In EXBERT [88], such an overview is represented as a heatmap, in which each module is a column, and each input is a row. Similar to seq2seq-vis, this system provides, upon selection of an input, a complementary representation view of neighbors with respect to corpus meta-data such as POS. Language interpretability tool [206] sought to provide a modular, and easy-to-use system for domain experts. In such a tool, the attention graph must be selected through a drop-down list of each module available. Indeed, this tool provides a vast collection of interpretability building blocks such as, among others, a model-agnostic one with LIME, a representation with a projection of embeddings, and metrics such as confusion matrices. Those blocks may help users understand which attention module might be relevant, and thus limit the amount of exploration required by users. Finally, Attention Flows [53] provides an overview of attention modules as heatmaps regrouped per input in a radial layout. Such a tool addresses the influence of BERT pre-training on model predictions by comparing two transformers, and two input sequences at once.

While those visual analytics systems designed to address attention in models yielded great results, it is worth noting that according to [91], they might be taken carefully following the well-known precept *"correlation is not causation"*. This is due to the fact that according to them, in order for attention to be explanations, other approaches such as gradient-based saliency should yield overlapping results. Additionally, if we were to force a particular attention map on the model (*e.g.* to focus on an input), the model's output should evolve accordingly. Since then, those claims have been argued and challenged by Wiegreffe et al [230]. According to them, their experiment to evaluate the overlap of attention and gradient-based saliency is insufficient to prove that they are indeed unaligned. In addition, concerning the second point, this might be due to a disagreement on the definition of explanation, in a sense that, following Wiegreffe et al, attention modules are not the complete explanation to a model's decision but rather an explanation among others (as detailed in Section 2.1.5). Nonetheless, according to

[29, 88] in order to prevent the pitfall of miss-leading interpretations, attention should be analyzed with the lens of complementary views such as for instance, embeddings and corpus-wide meta information.

For further information on the design of attention models, and visual-analytics systems addressing the interpretability of models with attention such as transformers [218] applied to Visual Question Answering (VQA), please refer to Chapter 3.

2.3.3 Interacting with Models for Non-experts

A common use of building blocks is to provide a visual introduction to a Deep Neural Network or method to non-expert users. In order to be successful, such an approach heavily relies on trial-error using exploratory interactions. For example, drawing an input to probe a model's behavior on ambiguous handwritten digits [208, 167], hence the need for tools to be resilient to potential mistakes. TensorFlow playground [188], and GAN Lab [110], took the decision to use toy datasets, with low dimensionality, and small models. Along with encouraging users to get an understanding of the intuitions behind the inner workings of deep learning, this also enables such a tool to train those models within the interface, while monitoring the evolution of its inner representation of data. Such a representation is at the core of those two systems in which users are invited to explore hyper-parameters such as the number of layers, or units within a layer to fit an expected output distribution. In Adversarial-Playground [152], the available hyper-parameters are adversarial methods that can be used to alter a model's inputs and observe the new predictions. Similar to model-agnostic blocks, such a system focuses on the inputs and outputs of a model rather than its inner workings. In CNN Explainer [229], the light is shed on the inner operations of CNNs, such as those leading to the creation of activations. Similarl to TensorFlow playground, those activations are combined in a node-link diagram directed from the left for the input, to the right for the output. On users selection, any intermediate result can be decomposed down to the mathematical operations connecting a selected result to the corresponding one from the previous layer, such as, for instance, the pixel per pixel *max* function of the ReLU activation method.

Outside of traditional academic media, visual interpretability for education, flourished online as interactive blogs, referred to as *explorables*. A key feature of explorables, in opposition to other visual analytics tools, is how information is presented to users through scrollable storytelling. This is done to progressively provide to users the intuitions needed to understand the core method presented. In most explorables, available interaction often offers alternative scenarios, on a few selected datapoints, rather than a broad exploration of a dataset. Pearce et al [163, 164] provided a collection of explorables addressing, among others, how biases may emerge from models asked to classify data. Such a collection

has the particularity to enable users, through interactions, to propose their own approach to classification, with the conclusion that a perfect solution, even outside of computers may be nearly impossible to reach. To do so, those explorables rely on model-agnostics blocks of metrics and input-output of simplistic classifier models. In order to introduce users to potential biases and productiveness of model decisions, users can be invited, through interaction, to explore relevant features of image faces [191]. During such an experiment, users should grasp that contrary to popular belief, those models may not follow human logic, and rely upon unexpected face attributes to output a gender, *e.g.* one's left eyebrow.

2.3.4 Interpretable Visual Analytics Throughout this Manuscript

As it can be observed in a vast majority of visual analytics systems introduced in the previous sections, activations of models are a core element to analyze. This might be due to fact that each of them may be visualized individually and eventually interpreted, hence one of the main hindrances to interpreting a complete decision process of a DNN, is their number. Nonetheless, those systems share common features such as the presence of an interactive representation view—usually on the left side of the system and are often instance-based, *i.e.* focusing on a single input at the time, while providing interactions to manually switch inputs. Each of those systems is intertwined with the model they are dedicated to, and its task. Hence, in many cases, they cannot be extended to another model, task, or dataset without requiring a large refactoring, and reshaping them. However, a vast majority of those systems are either applied to CNN models for image classification, or RNNs and attention models applied to text processing (*e.g.* translation, or completion). This raises gaps of interpretability between those popular models and their tasks, and the ones less addressed such as robotics. Those systems are, however, particularly useful for DL experts to design models, ultimately increasing the general audience's trust in their models [58].

The next chapters of this thesis are dedicated to the design of visual analytics systems relying on DNN activations as their core element. However, in contrast to the previously introduced systems, in this manuscript, we tackle the under-explored challenge of analyzing, and improving models for robotics. As introduced in Section 1.1, our focus is on three steps of robotics, namely: VQA, navigation, and sim2real ego-pose localization. Each of them is addressed with a different kind of model, along with different challenges. We tackle VQA (Chapter 3) tasks with transformer models, which by their size, and by the fact that they manipulate both texts, and images, and some of their activations working as attention, require particular attention in their design to prevent users to feel overwhelmed by the quantity of data to analyze. Navigation (Chapter 4), is addressed by Deep Reinforcement Learning (DRL) algorithms which, in our case combines both CNNs layers, and RNNs ones which raise the challenge of how to

combine visualization of those two kinds of layers. As such a model uses images as input, the visualization of recurrent activation with respect to inputs needs includes multiple images at once without occluding other visualizations on one's screen. In addition, by nature, DRL algorithms are trained by themselves, *i.e.* without ground-truth, thus when analyzed, they require for humans to evaluate themselves whether, for example, if a trajectory follows a coherent path. Finally, Sim2Real ego-pose (Chapter 5) is addressed using a regression CNN model, in such application, the challenge lies in how one may assess the gaps between simulation and real-world as perceived by the model. Hence, one may need a visual analytics system designed to visualize and compare activations from related sim/real inputs to grasp what may be difficult to transfer for the model, and then contextualize that information within its environment. We expect those contributions to pave the way to broaden the design horizons of visual analytics systems for DNN interpretability outside of the popular image classification and text-processing domains. We also expect that they contribute to raising concerns about the biases of models' decisions (*e.g.* the learning of shortcuts), along with helping DNN builders to further study and evaluate the reasoning capabilities of their models.

CHAPTER 3

VISUAL QUESTION ANSWERING

Contents

As stated in Chapter 1, and in illustrated in Fig. Figure 3.1, this chapter addresses the first step ① of our robotic goal, *i.e.* the ability to understand questions asked in natural language and answer them by analyzing images. Such a domain is known as Visual Question Answering (VQA). This is a testbed for learning high-level reasoning with primary use in Human-Computer Interaction (HCI), for instance, by providing audio description for the visually impaired. However, recent research has shown that state-of-the-art models tend to produce answers exploiting biases and shortcuts in the training data, and sometimes overlook

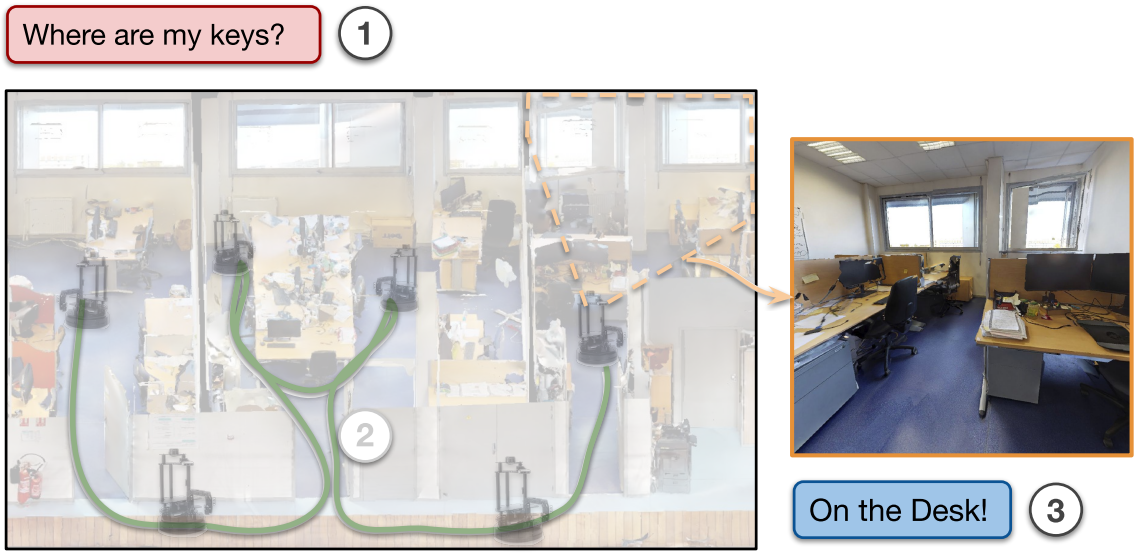


Figure 3.1 – This is chapter is dedicated to providing to robots the ability to answer natural-language questions about images. For example here, when asked the mundane question "Where are my keys?" ①, the robot needs to understand what we are looking for, and then search for it within a given image to provide an answer *e.g.* here "On the desk!".

input images, instead of performing the required reasoning steps. This chapter addresses this by introducing VisQA, a visual analytics tool that explores the question of reasoning vs. bias exploitation. It exposes the key element of state-of-the-art neural models — attention maps in transformers [202]. Our working hypothesis is that reasoning steps leading to model predictions are observable from attention distributions, which are particularly useful for visualization.

The work presented here is a result of a collaboration of three fields, machine learning, vision and language reasoning, and data analytics, which lead to two contributions: one in a Computer Vision (CV) venue, CVPR 2021, and another in Visualization, IEEE VIS 2021 as follows:

- Théo Jaunet, Corentin Kervadec, Romain Vuillemot, Grigory Antipov, Moez Baccouche, and Christian Wolf. "VisQA: X-raying Vision and Language Reasoning in Transformers". In: *IEEE Transactions on Visualization and Computer Graphics (TVCG), 2021.*

- Corentin Kervadec, Théo Jaunet, Grigory Antipov, Moez Baccouche, Romain Vuillemot, and Christian Wolf. "How Transferable are Reasoning Patterns in VQA?". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2021.*

The design process of VisQA was motivated by well-known bias examples from the fields of deep learning and Vision-Language (VL) reasoning and evaluated in two ways. We believe that this work led to a better understanding of bias exploitation of neural models for VQA, which eventually resulted in an impact on its design and training through the proposition of a method for the transfer of reasoning patterns from an oracle model. Secondly, the design of VisQA, and a goal-oriented evaluation of VisQA targeting the analysis of a model decision process from multiple experts, provides evidence that it makes the inner workings of models accessible to users, and that those models may be improved.

VisQA is available online as an interactive prototype https://visqa.liris.cnrs.fr, and code source and data are available as an open-source project: https://github.com/Theo-Jaunet/VisQA .

A special thanks goes to Corentin Kervadec, Grigory Antipov, and Moez Baccouche without whom the works presented in this chapter would have not been possible.

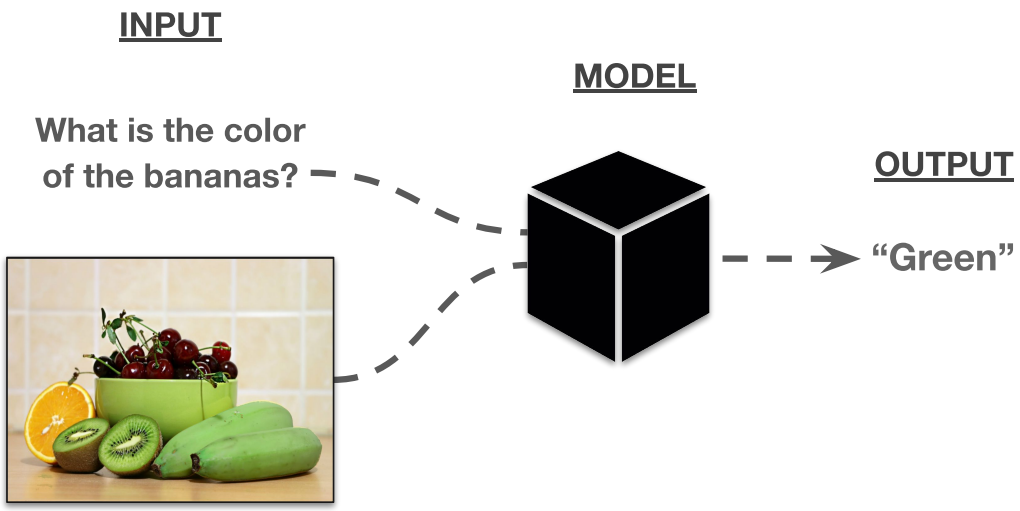


Figure 3.2 – In Visual Question Answering tasks, we provide to a model a question in a textual form along with an image. In our case, we expect the model to analyze the image to answer the given question with a single word. As an example here, when asked "What is the color of the bananas?", the model should output "Green".

3.1 Introduction

Visual Question Answering (VQA) systems [8] attempt to answer questions provided as input in a textual form together with a corresponding image. As an example, asking the question *"What is the color of the bananas?"* to a model, with the input image shown in Figure 3.2 should yield the answer *"Green"*.

Direct applications of such systems are support for the visually impaired, semi-autonomous robot navigation through language instructions, and, more generally, Artificial Intelligence (AI) tools covering a broad spectrum of tasks guided through language input. In particular, VQA serves as a testbed for learning high-level reasoning from data, as the performance of targeted models and methods relies on advances in CV, Natural Language Processing (NLP), and Machine Learning (ML). The task deals with large varieties, and solving an instance can involve visual recognition, logic, arithmetic, spatial reasoning, intuitive physics, causality, and multi-hop reasoning. It also requires combining two modalities of different nature: images and language.

Recent VQA models are based on a powerful type of deep neural network called transformers [218]. Originally developed for NLP tasks, these models have been extensively applied to VQA [129, 202, 237] and recently even on pixel-level in pure image-based problems [57, 59, 70, 227, 241]. Transformers are conceptually simple models, which, however, can learn very complex relationships between the items of un-ordered sets, each of which is represented as a (learned) embedding in a vector space. Making sense of a learned neural model and verifying its inner workings is a difficult problem, which we address in this work.

In this chapter, we focus on a typical and important problem arising with trained neural models, and in particular models for vision and language reasoning: as they are trained with supervision to provide correct answers, they often tend to find shortcuts in learning and learn to exploit spurious biases in training data instead of the desired reasoning a human would apply in a similar situation [3, 76, 107, 133]. To provide an example, if a model is asked *"What is the color of the bananas"*? with he input image shown in Figure 3.2, it might learn to answer *"yellow"* despite the real color being *"green"*. This is due to fact that *"yellow"* is the correct answer for a large majority of input images including bananas. Hence, learning to output *"yellow"* when bananas are involved is easier than solving the correct reasoning problem for only a small minority of cases. An exact definition of the term "correct reasoning" is difficult, we refer to [24, 107] and define it as *algebraically manipulating words and visual objects to answer a new question*. In particular, we interpret reasoning as the opposite of exploiting spurious biases in training data.

Existing work on bias reduction and the evaluation of bias origins tends to focus on statistical techniques, whose power lies in quantitative evaluation and visualizations on dataset-level, showing full or marginal distributions of inputs, features, and outputs, and resort to dimensionality reduction. While these techniques are very useful, their power is limited when we search for insights into detailed inner workings of neural models, for which an investigation per sample is more helpful. Only for a single instance, it is possible to observe the origins for lack of reasoning, which can include, aside from errors in the trained reasoning

module itself, also problems in the input pipeline (the object detection module) and wrong annotations of ground truth data.

We introduce VISQA, an instance-based visual analytics tool designed to help domain experts, referred to as model builders [87]. Using VISQA, we investigate how information flows in a neural model and how the model relates different items of interest to each other in vision and language reasoning. Attention maps are at the heart of transformer-based deep networks, and as such are the primary object studied by VISQA. It allows an expert to browse through image and question pairs sorted by an automatic estimate of the amount of reasoning that went into answering each sample. Once a pair is selected, users can explore the different attention maps represented as heatmaps. The exploration is guided by their position in the model, but also by color codes that convey the intensity of each head, i.e. whether they focus attention narrowly on specific items, or broadly over the full input set. Complementary dataset-wide statistics are provided for each selected attention head, either globally, or with respect to specific reasoning modes of language functions, e.g. *"What is"*, *"Where is"*, *"What color"* etc. While the tool is post-hoc, it is also interactive and allows certain modifications to the internal structure of the model. At any time, attention maps can be pruned to observe their impact on the output answer.

VISQA is the result of a collaboration between experts in visual analytics, and experts in Visual Question Answering systems and Machine Learning. As will be detailed in Section 3.4, this collaboration, and data gathered using VISQA, led to improvements of the reasoning capabilities of transformer-based models by introducing new methodological contributions in machine learning and computer vision, which we also recently reported in a different associated publication [106]. The usability of VISQA has been evaluated by different experts in deep learning, who were not involved in the project nor its design. We report experiments with qualitative interviews and results in Section 3.8.

In this work, we contribute to a better understanding of bias in VQA models as follows:

- **VisQA an interactive visual analytics tool** which helps experts to explore the inner workings of transformers models for VQA by displaying models' attention heads in an instance-based fashion.

- **A set of visualizations to address bias in VQA systems** designed to explore models' performances in real-time with altered attention, and/or by asking free-text questions.

- **Insights on the emergence bias in transformers for VQA** gained by experts through an in-depth analysis using VISQA, along with an evaluation of its usability to estimate models' predictions and eventually bias exploitation.

3.2 Background

We first introduce some background on understanding neural networks in vision and language reasoning, the context of this work, and we provide a short and concise introduction into transformers, the type of neural networks which currently dominates academic and industrial research in language reasoning, and in vision-based language problems.

3.2.1 Transformers and Attention

Following the introduction and success of transformers applied to natural language processing tasks [218, 54], transformer-based models were also proposed for VQA [68, 237]. Their key strength is the ability to contextualize input representations, *i.e.* to take input items like words and objects, each one represented in a vectorial form called "*embedding*", and to enrich them, adding information on relationships. This is achieved by a series of transformations of the input vectors, which effectively encodes the reasoning process, and the underlying key mechanism is attention (self-attention). We start with a brief overview of how a typical language transformer works by applying it to encode the question *"What is the name of the clothing item that is white?"*.

Step 1: Preparation: Sentence Tokenization — We split the question into elementary language items (called "tokens") with the WordPiece tokenizer [233]. Two special tokens are added at the beginning and at the end of the sentence (respectively 'CLS' and 'SEP'). While the latter encodes the end of the sentence, the former is of the uttermost importance; it is transformed by the model as are the other tokens, with the difference that the 'CLS' token is transformed to encode the task-specific information, and the answer. In the given example, at the end of the transformation, it is expected to contain the information required to predict the name of the white clothing. Each token (including special tokens) is then projected into a high-dimensional vector space through a learned dictionary, resulting in a sequence of N token embeddings: $\boldsymbol{L} = [\boldsymbol{l}_{CLS}, \boldsymbol{l}_1, \ldots, \boldsymbol{l}_i, \ldots, \boldsymbol{l}_{N-2}, \boldsymbol{l}_{SEP}], \boldsymbol{l}_i \in \mathbb{R}^n$, n being the (chosen) embedding dimension.

Step 2: Attention Maps — Transformers progressively contextualize each of the N input embeddings by a sequence of self-attention operations (layers), with the objective of making each token embedding "aware" of the neighboring embeddings. In our example, it might be helpful to combine the information from the embeddings of tokens *'item'*, *'clothing'* and *'white'* into one "enriched" embedding describing the referred object, as the three words are semantically related. More generally, the so-called "self-attention operations" (layers) are the key elements

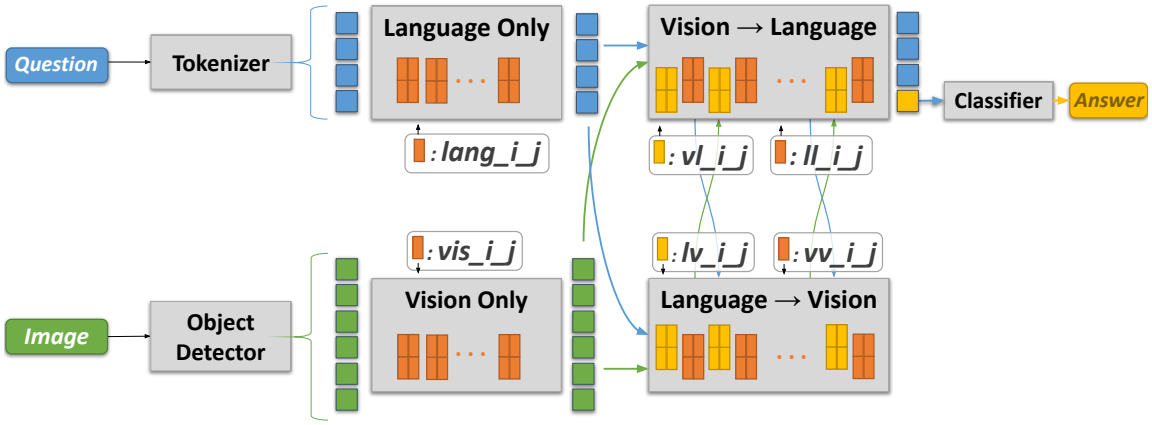


Figure 3.3 – An Illustration of the VL-Transformer architecture we rely on. Question and image are first tokenized and then encoded using vision (in green) and language (in blue) only transformers [218], followed by (bi-directional) inter-modality transformers [202]. The answer is predicted from the *"CLS"* token. Yellow and orange rectangles represent, respectively, inter- and intra-modality attention heads. i and j are the layers and head indices used for naming attention heads through the manuscript.

of this type of model. They are implemented via the calculation of *attention maps* $\boldsymbol{A} = \{\boldsymbol{\alpha}_{ij}\}$, which reflect the N^2 pairwise interactions between the tokens, each $\boldsymbol{\alpha}_{ij}$ being the similarity between token embeddings i and j. The similarity function, here, the scaled dot-product, is calculated between a trainable projection of the embedding i, called *query*, and a trainable projection of the embedding j, called *key*. The per-token attention energy is normalized into a probability distribution using a row-wise softmax. In our example, the row vector $A_7 = \{\boldsymbol{\alpha}_{7j}\}_{j\in\{0,\dots,N-1\}}$ encodes the N interactions between *'clothing'* and the other words.

Step 3: Token Updates — Each token embedding is updated as a linear combination of a trained function of the input embeddings, called *values*, weighted by the attention map $\boldsymbol{A}$. Hence, the model transforms each token by learning a strategy for looking at specific other words.

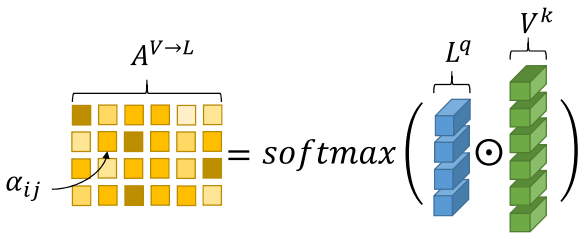
Multi-headed Attention — Like most classical neural networks, transformers are organized into a sequence of layers. Each of these layers bundles together *multiple attention heads* working in parallel, which allows the model to learn different cooperating strategies. Different heads might learn different syntactic or semantic language functions, as shown in [221] for language models, and as we will show in the experimental section for vision and language reasoning. The outputs of the attention heads are combined with standard neural network blocks.

3.2.2 Vision-Language (VL)-Transformers

Transformers have been extended to reasoning on multiple modalities, in particular vision and language, through different types of layers: Language-only and vision-only layers, referred to as *intra*-modal layers, and language-vision layers, referred to as the *inter*-modal ones. Figure 3.3 depicts the transformer architecture designed for VQA which we call "*VL-Transformer*". Each layer is named as *X_i_j*, where *X* denotes the layer type (*e.g.* vision-only intra-modal layer, vision-language inter-modal layer, etc.) and *i* and *j* index, respectively, layer and head.

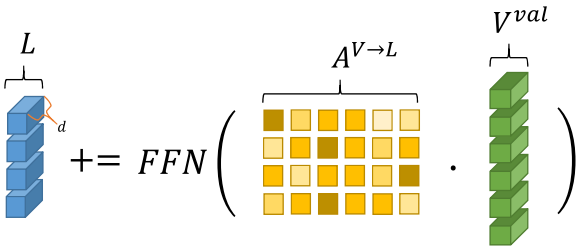
Intra-modality — Both modalities are first processed in two independent streams (cf. Figure 3.3): heads *lang_i_j* encode question words, and *vis_i_j* heads encode visual objects detected in the image by an off-the-shelf object detector, a mainstream approach in VQA [6, 202]. Visual embeddings are the concatenation of 2048-dimensional object appearance embeddings and 4-dimensional bounding box coordinates.

Inter-modality — Subsequent layers combine information between both modalities, c.f. Figure 3.3, in a bidirectional way: from question words to visual objects in *lv_i_j*, and vice-versa in *vl_i_j* (*lv* means 'language to vision' while the opposite *vl* means 'vision to language'). This requires a minor, but essential, modification of the attention mechanism. Intuitively, and a bit simplified, in vision-to-language heads, the language (word) embeddings are transformed by taking each word and checking its similarity to the full set of visual input objects, and vice-versa for language-to-vision heads. More precisely, *query* vectors are taken from the modality to be contextualized, and the *key* and *value* vectors from the other one. Thereby, in the case of the vision to language heads, *vl_i_j*, attention maps $A^{V->L}$ are computed as the outer product between the *query* projections L^q of the language embeddings and the *key* projections V^k of the visual ones:



(3.1)

The softmax is applied row-wisely, such that each attention map's row sum to 1. Then, the language embeddings L are updated with the *value* projections V^{val} of visual tokens:



(3.2)

Where $+=$ represents a residual connection and FFN is a trainable feed-forward layer. For the sake of clarity, we omit the multi-head mechanism in Equation 3.1 and Equation 3.2. Nevertheless, it is important to notice that the inter-modality transformers are multi-headed, similarly to the intra-modal ones. As shown in Figure 3.3 of the manuscript, each *lv* or *vl* attention head is immediately followed by an intra-modal attention head called, respectively, *vv* or *ll*.

Predicting the answer — The answer is produced by decoding the final representation of the "*CLS*" token using a 2-layered neural network. It predicts a probability vector over the most frequent answers found in the training set, the answer with the highest score is then chosen.

Training details — For the experiments in Section 3.8, we set the embedding size to d=128 and the number of heads per-layers to h=4. Following [202], our model is composed of 9 language only and 5 vision only intra-modality transformers layers, and 5 language $\rightarrow$ vision and vision $\rightarrow$ language layers. In addition to the VQA objective, we train the model parameters also on MS-COCO [125] and Visual-Genome [111] images following the semi-supervised BERT [54]-like strategy introduced in [202]. In particular, we trained the model to perform simple tasks such as recognizing masked words and visual objects, or predicting if a given sentence matches the question. After pre-training on these auxiliary tasks, the model is fine-tuned on the GQA [90] dataset with the VQA objective. Our VL-Transformer is a variant of the LXMERT model [202], in line with the many works adapting BERT [54]-like pre-training to vision and language tasks [196, 41, 123, 67, 129].

Discussion — In this work, we focus on the interpretation of the attention maps, as they contain crucial cues on the internal reasoning in transformers. These maps highlight to what extent a given token has been contextualized by which neighbors, high attention α_{ij} indicating strong interaction between tokens i and j. We argue, that attention maps provide strong insights on how our the model handles interactions between the question the image.

3.3 Related Work

Our work is related to building visual analytics tools for interpretability of Deep Learning. In contrast to Chapter 2, this section focuses on how visualizations can be applied to the attention maps from transformer models specialized for VQA. Our design targets in particular the study of attention maps from transformers models to grasp insights on their potential exploitation of bias. This section reviews previous work on the visual analysis of deep learning models, and a review of previous work from machine learning communities to tackle bias in VQA systems.

3.3.1 Interpretability of VQA

The work the most related to VisQA is Attention Flows [53] which addresses the influence of BERT pre-training on model predictions by comparing two transformers models. Like VisQA, such a tool displays an overview of each attention head with a color encoding their activity. While Attention Flows is designed to address the comparison of attention maps of models trained for NLP tasks, in this work, we tackle the challenges provided by the bi-modality of vision and language reasoning, and expand the interpretability of VQA systems which can rely on visual cues or dataset biases. Current practices of VQA visualization include attention heatmaps of selected VL heads based on their activation [121] to highlight word/key-object associations, global overview heatmaps of attention heatmaps towards a specific token [32], and guided backpropagation [77] to highlight the most relevant words in questions. Following those works, VisQA provides a visualization of every head's attention heatmaps and word/object associations, along with an overview of their activations.

Here, our focus is on post-hoc interpretability [126], *i.e.* the analysis of a trained model's decision policy after-the-fact. Our approach relies on instance-based analysis, which displays inner model parameters with respect to the current input. Such analysis is often combined with direct manipulation mechanisms designed to let users experiment with desired input conditions, like drawing the input [35]. Our working hypothesis is that a transformer-based model's mode of operation, i.e. whether it is reasoning or exploiting dataset biases, is observable from its trained parameters, and in particular from attention maps, an intermediate representation dependent on parameters.

3.3.2 Bias Reduction in VQA

Bias reduction has been addressed on the data side through cleaning and balancing. In particular, GQA [90] focuses on semantics with the help of human-

annotated scene graphs and automatically generated questions. On the contrary, VQAv2 [76] dataset is crowdsourced, which leads to more natural questions, but includes cognitive and/or social biases [61] as well as annotation mistakes. In addition, evaluation benchmarks have been specifically designed to identify the presence of bias dependencies. VQA-CP [4] proposes to evaluate models bias dependency by introducing distribution shifts between training and testing sets. However, this approach has limitations to evaluate the decisions and biases exploitation of models as it emphasizes the diversity in answers rather than the reasoning itself. As a result, random predictions can also contribute to improving a model's score on this benchmark [205, 185]. On the other side, GQA-OOD [107] undertakes a similar strategy while keeping the training set distribution untouched. Instead, the authors propose to evaluate the VQA performances on question-answer pairs regarding their frequency in the dataset. This makes it possible to evaluate both in- and out-of-distribution performances and, at the same time, estimate the reasoning ability of the system. Indeed, if a model's prediction is correct while the question-answer pair is rare, there are chances that the model has not used statistical biases. These datasets and evaluation benchmarks have led to the conception of diverse bias-reduction methods. While an exhaustive survey of these methods is out of our scope, one can mention families of approaches such as training regularization [185], or counterfactual learning [1, 73].

In this work, we define bias as the way how a model learns regularities or shortcuts from its training dataset, which may push it to provide answers which it frequently encountered as correct during training, without even considering information from the input image. Technically, we evaluate bias as introduced in [107]. Hence, every question from the dataset is grouped using their topic (*e.g.* "furniture") and function (*i.e.* task extracted from the semantic of the question). Both the semantic and topic metadata are provided by the GQA dataset. Then for each kind of question, the frequency of their answers is computed. We estimate that the model exploits bias when it incorrectly predicts an answer which is among the 20% most frequent answers for the given question, while its ground-truth answer is among the 20% least frequent. While our proposed tool allows us to partially open the black-box of transformer-based neural vision and language models in a very general sense, making it possible to inspect how they handle relationships between data items, we nevertheless specifically focus on the important problem of bias reduction.

3.4 Motivating Case Study

This work was primarily motivated by growing concerns in the field over bias exploitation of models trained in large-scale settings, in particular when trained on very broad problems like vision and language reasoning [3, 76, 133]. Contributions

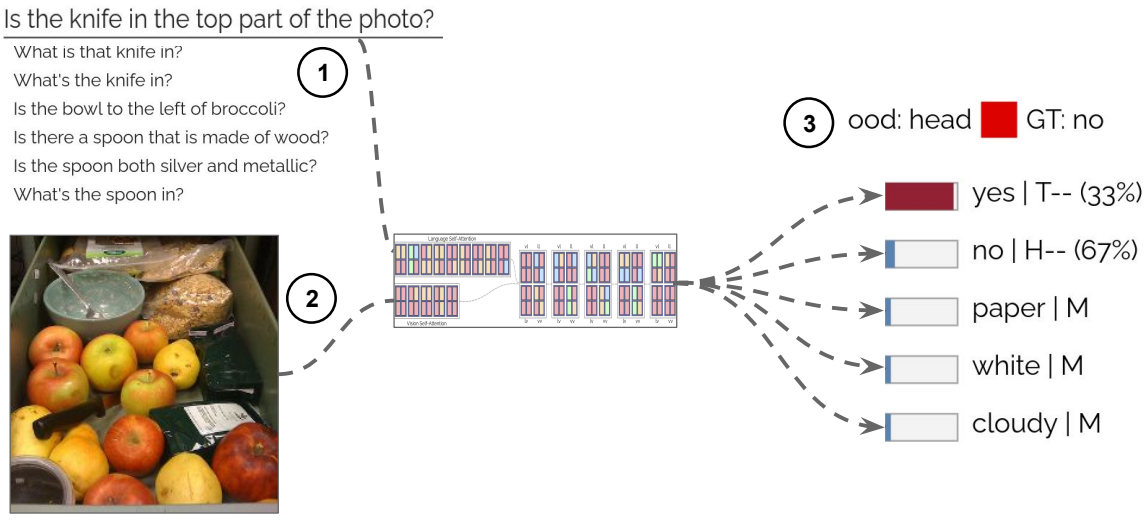


Figure 3.4 – When asked *"Is the knife in the top part of the photo"* ① the tiny-LXMERT model, with the image of a knife at the bottom ②, incorrectly outputs *"yes"* ③ with more than 95% confidence. While an exploitation of bias can be considered, we can observe that the answer *"yes"* represents only 33% of answers of similar questions over the complete dataset. Thus in-depth analysis of the attention of the model may be required to grasp what led to such a mistake.

of our group in this area include new benchmarks [107] and additional supervision and regularization for neural models [108]. Here we extend these previous efforts by providing a tool for instance-level visualizations and performing visual analytics on a single sample. This choice was ultimately taken when comparisons of different trained models through statistics failed to provide concrete answers on the sources of confusion and errors. Statistical models enabled us to drill down examinations to a certain minimal level of aggregation, for instance, linguistic function groups, but this kind of analysis was not fine-grained enough.

We illustrate the advantages and the power of instance-level visualizations with our contribution, VisQA on the following case study. It is based on the exploration of a tiny version of the state-of-the-art neural model LXMERT [202] as described in Section 3.2.2. We provide it with the following input instance, *i.e.* the image given in Figure 3.4 ②, and associated question *"Is the knife in the top part of this photo?"* ①. The correct ground truth answer is of course *"No"*, but the baseline tiny-LXMERT model incorrectly answers *"Yes"* ③. We see the frequency of the different possible answers provided in the interface, and observe that the wrong answer *"Yes"* is not the most frequent one for this kind of question as *"No"* is the correct answer 67% of the time, which does not provide evidence for bias exploitation. The objective is to use VisQA to dive deeper into the inner workings of the model.

A first step is to analyze whether the model is provided with all necessary information. While the input image itself does contain all clear pictures of the

answer, the neural transformer model reasons over a list of objects detected by a first object detection and recognition module (Faster R-CNN [171]) which may output errors. In the rest of this manuscript, we will refer to this tiny-LXMERT as the *noisy model*.

Is the knife detected by the vision module? VISQA provides access to the bounding boxes of the objects detected by the input pipeline. Each bounding box can be displayed superimposed over the input image along with the corresponding object label predicted by the object recognition module. We can observe that the key object *"knife"* lacks a suitable bounding box or class label, it has not been detected. Since this object is required to answer the question for this image, the model cannot predict a coherent answer. However, the question remains why the wrong answer is *"yes"*, corresponding to the presence of a knife.

Can attention maps provide cues for reasoning modes? VISQA focuses on attention maps, which are a key feature of transformer-based neural models, as they fully determine relationships between input items. Users can select different heads and explore the corresponding attention maps. For the example case, we are interested in checking the correspondence between the question word *"knife"* and the set of bounding boxes. This should provide us with evidence of whether the model was capable of associating the concept with the visual object in the scene, which is, of course, not sufficient for a correct answer, but a necessary step. This verification is non-trivial, however, since the model is free to perform this operation in any of the inter-modality layers and heads. VISQA allows to select the different heads, and we could observe that none of the heads provides a correct association. As an example, we can see the behavior of a head in Figure 3.5 ①, which associates the word *"knife"* to various objects, mostly fruits. No other head is found indicating a more promising relationship.

Is computer vision the bottleneck? From the example above, as well as similar observations in other instances, we conjecture that the computer vision input pipeline (notably, the imperfect object detector) is one of the main bottlenecks in preventing correct reasoning. To validate this hypothesis, we explored training an *Oracle* model with perfect sight, which thus takes as input the ground truth objects provided by human annotation instead of the noisy object detections by a trained neural model. This improves the performance of the model considerably, reaching ∼80% accuracy on the difficult questions with rare ground-truth answers, compared to ∼20% for the standard model reasoning on noisy input. This particularly high difference in performance for questions with rare answers suggests a higher performance in correct reasoning of the oracle model. By loading this model into VISQA, we observe in Figure 3.5 ②, that there exists an attention map which associates the word *"knife"* to a visual object *"knife"*, which, we recall, is an

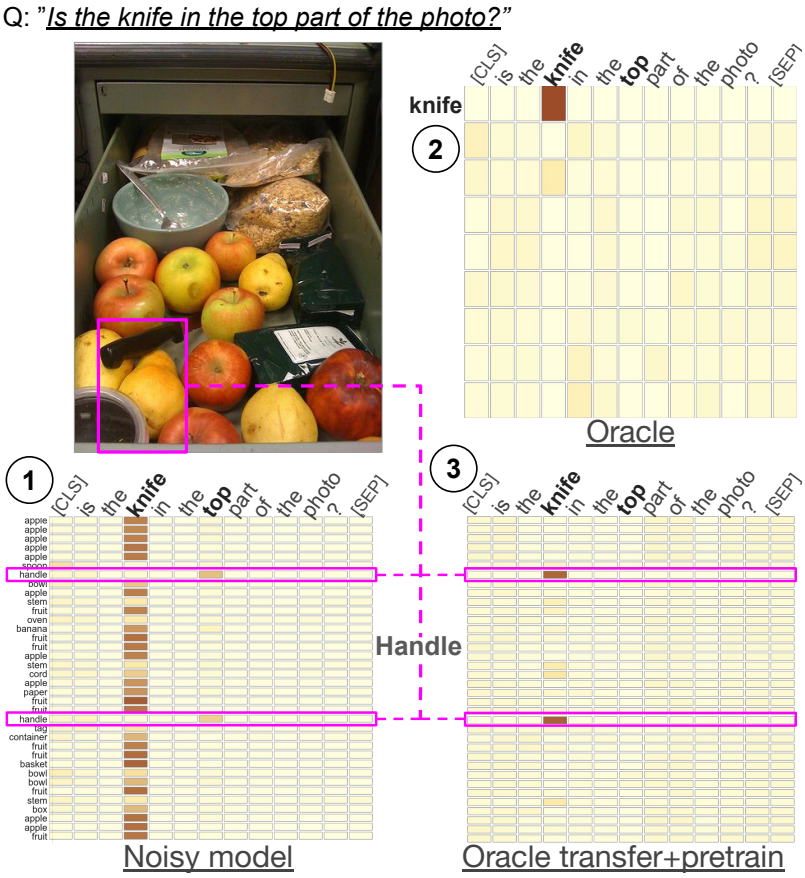


Figure 3.5 – Visualization of a selected vision-to-language head and attention map for two different models. ① the *noisy model* associates the *"knife"* word with a large number of different objects, including fruit. ② the oracle model learns a perfect association between the word *"knife"* and the *"knife"* object; ③ the oracle transfer model associates the word *"knife"* with two different bounding boxes of type *knife handle*, whose embeddings are sufficiently close for correct reasoning. Head selections are not comparable between models, we therefore checked for permutations.

object indicated through human annotation. This correct association is reassuring, but by itself does not yet guarantee correct reasoning — further exploration is possible, but we will now concentrate on this problem of finding correspondences between words and visual objects and explore this question further.

Transferring reasoning modes — Given these observations, we conjecture that a transformer model with perfect sight can learn modes of reasoning which are less biased than models trained on real but noisy input data. This is an insight we gathered from using VisQA as a tool for visual analytics. Oracle models, on the downside, are not deployable to real-life problems, as they work on human annotations only, per definition. We explore a solution to transfer reasoning modes

from oracle models to noisy input data, which can be done using knowledge transfer by parameter initialization [235]. In more detail, we pre-train the Oracle model on ground-truth data, and once it reaches convergence, we use its weights as initialization for the model using noisy inputs.

We load this model, which we call *Oracle Transfer*, into VISQA, with the objective of exploring whether reasoning modes have been transferred into a deployable model capable of providing answers given real input. Figure 3.5 shows the vision-to-language attention maps from the same layer as the ones explored above [1]. We can observe that the model's attention is drawn towards *"handle"* visual objects, which are parts of the only knife in the picture. While the object classes *"knife"* and *"handle"* are logically different, we can conjecture that their vectorial feature embeddings are different but sufficiently similar to allow reasoning. More importantly, we can deduce that the model adapted to the absence of the knife object by relying on what is available, i.e. knife handles. This leads the model to correctly answer *"No"*, as did the Oracle model on ground truth data, and in contrast to the baseline *noisy model* without transfer — see also the illustration in Figure 3.6. We further describe this Oracle knowledge transfer, and improvements it provides to transformer-based VQA in a different associated publication [106]. This model will be used in evaluations performed by deep learning experts in Section 3.8. All models, noisy, oracle, and oracle transfer, can be tested and explored online in our prototype.

3.5 Design Goals

Prior to the design of VISQA, when working on an associated publication [106], we conducted discussions with experts (co-authors of the papers associated with this chapter), about their workflow to analyze bias in VQA systems. From those discussions, and literature review introduced in Section 3.3, we distill their needs in the following four main themes of design goals for instance-based analysis.

G1 Examine the performances of each instance for a given model. To investigate bias in VQA systems as introduced in Section 3.3.2, it is first important to examine the model predictions, along with its confidence score with respect to its inputs. In order to be useful, those predictions need to be combined with the ground-truth, to estimate if the model is wrong, and how frequent is the ground-truth answer and predictions. Inputs need to be inspected as well as they may convey ambiguities that may be at the beginning of an explanation for a mistake. Finally, due to a large amount of data available for inspection, experts

1. Each layer contains several attention heads, and these heads are not ordered. Reasoning associated with a given head in one model can correspond to the reasoning mode of a different head in another model. We cope with these possible permutations in our experiments by searching over all possible heads of a layer

may prioritize inputs the more likely to be biased, *i.e.* those with infrequent ground-truth answers and frequent predictions.

G2 Browse the attention of the model for an instance. Analyzing the attention maps of LXMERT models is crucial for understanding what factors influenced its decision, and eventually whether or not the model attends to both language and vision. While visualizing individually each attention map is feasible, we aim at improving such an exploration, by contextualizing each attention head with their neighborhood (*i.e.* other attention heads directly connected to it), and position within the model. This is relevant as attention heads get closer to the output, they both encapsulate previous attention, and may be more influential on models' decisions. In addition, experts may need to prioritize heads conveying salient attention, thus those heads need to be summarized and/or emphasized. Finally, for in-depth analysis, experts need to visualize the complete attention map and link each of their elements to human-understandable information, *i.e.* words of the question and bounding boxes within the input image.

G3 Link attention to language tasks. Once a relevant attention head is observed by an expert, the user should be able to contextualize it with the rest of the dataset. In particular, experts are interested in evaluating whether or not this head is responsive to certain tasks provided by the semantics of questions (*e.g.* find a color), or rather if the head is responsive to certain topics (*e.g.* clothing). Such information can be provided in the VQA training dataset, considered as categories.

G4 Explore alternative scenarios. Once cues on how the model uses its attention to output a decision are gathered, the next step, is to test this knowledge by querying the model on altered input or parameters. Ultimately the experts desire to answer questions such as: "would the model have a similar attention if the question were on another object of the image?", or "is this head or group of heads relevant for the final decision?". This can be regrouped into two categories first, the possibility to ask free-from questions, and second the possibility to modify the model's attention. In order to be usable, and due to the number of queries an expert may need to execute, both those manipulations require to interact with the model in a reasonable amount of time, *e.g.* less than a couple of seconds.

3.6 Design of VisQA

We designed VisQA, an visual analytics tool designed to facilitate in-depth analysis of the internal structure of transformers models applied to visual question answering. VisQA implements attention heads and interactive heatmaps visualizations, and other interactions such as free-form question and pruning

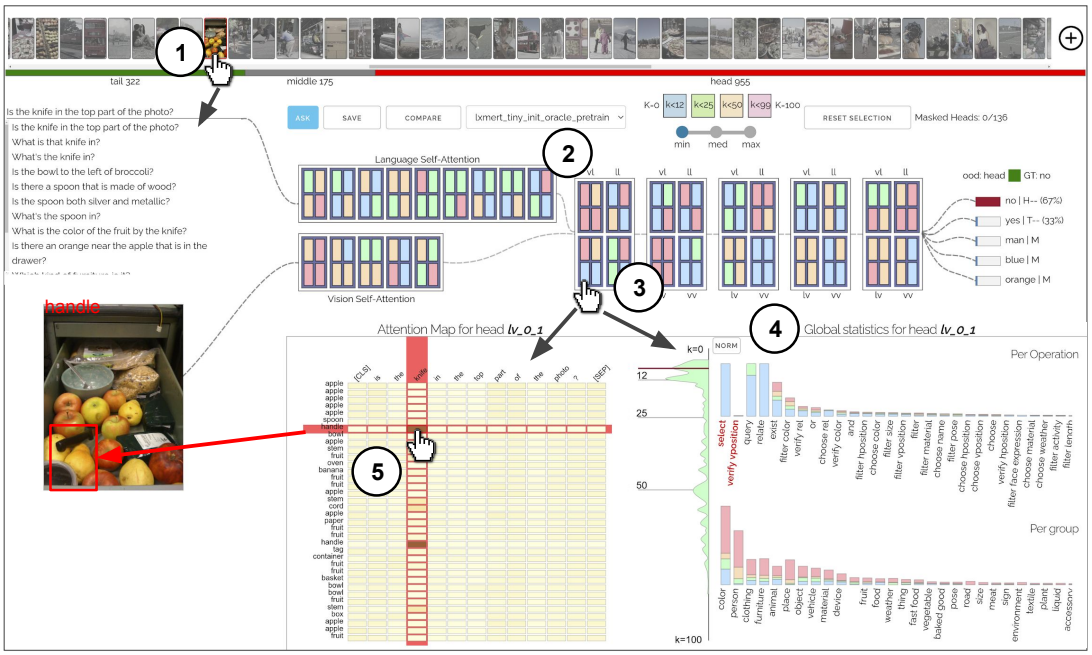


Figure 3.6 – Opening the black box of neural models for vision and language reasoning: given an open-ended question and an image ①, VISQA enables to investigate whether a trained model resorts to reasoning or to bias exploitation to provide its answer. This can be achieved by exploring the behavior of a set of attention heads ②, each producing an attention map ⑤, which manages how different items of the problem relate to each other. Heads can be selected ③, for instance, based on color-coded activity statistics. Their semantics can be linked to language functions derived from dataset-level statistics ④, filtered and compared between different models. This tool is available online at: https://visqa.liris.cnrs.fr.

mechanisms, to assess whether a model resorts to reasoning or bias exploitation when answering questions.

3.6.1 Workflow

Through iterative design resulting from frequent meetings with VQA experts co-authors of this work, we extracted the following workflow of use of VISQA, based on their experience analyzing VQA systems, and the mantra *overview first, zoom and filter, then details on demand* [184]:

1. the user picks and loads into the model an instance, informed by the likeliness of the question to be answered with bias;
2. the result of the model is presented both as attention heads (internal structure of the model), as well as the top-5 predicted answers;

3. the user then may interact with the model internal structure (*e.g.* heads intensity, attention maps, etc.) which triggers updates to the statistical views;
4. bounding boxes are displayed on the input image to reflect based on attention heat-map selections, showing how the model associates words and visual objects.

VisQA can also be used beyond this typical workflow for in real-time query of the model by asking user-defined questions, or pruning attention heads, as further detailed in Section 3.6.4.

3.6.2 Visualization of Instances

At its core, VisQA is organized following an end-to-end approach, from model input to output. This is done to contextualize attention heads with their neighborhood and position with the model (**G4**). We added input selection to guide users in their exploration, and a visual summary of the attention of the model to keep it visually compact.

Image ranking-by-feature (Figure 3.6 ①) — In order to ease user exploration over the complete dataset, VisQA displays images in the top bar from left to right based on the likelihood of their questions to be answered using statistical biases. To do so, we classify questions using ground truth answers, as proposed in [107]: top 20% of the most frequent answer will be classified as *Head*, as opposed to *Tail* which describes questions with the least frequent answers. We attribute a score to each image based on their *Head*-questions/*Tail*-questions ratio. The more an image has *Tail*-questions over *Head*-questions, the higher its score is. The underlying hypothesis is that frequent answers will be chosen more likely when a model tends to exploit biases (e.g. *"Yellow bananas"*). Also, frequent questions are harder to analyze since if any bias is exploited by the model, it will answer correctly. Answers and their frequencies are displayed in (Figure 3.6 ② right) following design goal (**G4**).

Instance view (Figure 3.6 ②) — This view, inspired by VL-Transformer representations illustrated in Figure 3.3, is the root of any analysis of attention maps using VisQA (**G4**). It matches the internal data flow through the internal structure of the model, from left to right: the input image/question pair, layers and heads with intra-modality layers first, and finally the answer output distribution (encoded as horizontal bars). A particular design decision was to display all attention maps at once, using a single-colored rectangle encoding the attention intensity as k-number [169] (see next paragraph for details). In the *Instance view*, attention heads can be selected with a mouse-over interaction as illustrated in Figure 3.6 ③

in order to provide details on demand, by displaying the corresponding attention heat-map Figure 3.6 ④ and head statistics.

Visual summary — Our model uses 136 attention maps with dimensions varying with respect to the number of words in the question and bounding boxes provided. As displaying all of those matrices would prevent experts to analyze them in a reasonable amount of time, we rely on summarizing each of them to a single scalar. Such a scalar, referred to as *k-number* [169], represents the normalized amount of tokens per row summed up to reach a threshold of 90% of energy. A *k-number* close to 0 indicates that the corresponding row has peaky attention focusing on only one column (as seen in Figure 3.5 ①), and a high *k-number* encodes a uniform attention (as in Figure 3.5 ②). Then we combine each of those *k-number* together using either *min*, *median*, or *max* functions. Such functions can be selected in VISQA by users, depending on the attention maps intensity they want to investigate. VISQA provides this interaction because for a head to have a low *k-number*, the majority of its rows needs to be highly activated. This can shadow attention maps with less than half of their rows with peaky attention. In VISQA, the *k-number* is discretized and color encoded in 4 categories as it follows: k<12 k<25 k<50 k<99. The decision to use a logarithmic discretization and color encoding, as introduced in [169], was done to emphasize peaky attention maps ($k < 12$) that need to be prioritized for analysis. In addition, this discretization, used throughout VISQA, is particularly useful to regroup instances in head-stat view Figure 3.6 ④ and select them by clicking on the corresponding square (legend on the right of ② in Figure 3.6) for *Head pruning* further detailed in Section 3.6.4. Finally, this reduces the learning curve of VISQA, as experts are familiar with this discretization of *k-numbers*.

3.6.3 Visualization of Selected Heads

VISQA provides details for a selected head in the Instance view using the attention map and head statistics.

Attention Maps (Figure 3.6 ⑤) — are represented as heat-maps, with cell colors encoding the attention intensity over a sequential, single hue scale from *no attention* (*i.e.* 0) in beige, to *full attention* (*i.e.* 1) in brown. This matrix-based approach contrasts with bipartite connections representations found in Seq2Seq-Vis [194] and BertViz [219]. We use matrices as they provide a clutter-free representation of attention and they can display multiple heads at once by aggregating connections. However, it may suffer from visual clutter issues as the number of words grows. Seq2Seq-Vis tackles this issue by using interactions to hide graph lowest connections (*i.e.* lowest attention). In our case, we argue that visualizing the attention of

one head at a time can lead to a better understanding of its function in the model. Such an exploration of head functions is further detailed in Section 3.8.

Head Statistics (Figure 3.6 ④) — are represented using three charts. A vertical area chart (leftmost chart) represents the distribution of *k-numbers* of the selected head over the complete validation dataset (around 1500 image/question pairs). The vertical axis encodes the values of *k-numbers*, while the horizontal axis encodes the density of the corresponding *k-number*. The current *k-number*, for the selected head, which corresponds to the image/question pair loaded in VisQA, is represented as a horizontal red bar positioned on the vertical axis. This area-chart provides insights such as the detection of useless heads with constant high *k-number* which can be reduced to calculation on average overall items instead of selecting specific items. In contrast, heads with constant low *k-number* can be interpreted as conveying key information. More specialized heads, with bi-modal *k-number* distributions, can also be observed. Two stacked bar-charts represent the *k-numbers* of the selected head grouped by question operations (**G3**). Question operations are ground truth information provided by the GQA dataset [90] describing the semantic reasoning operation of asked questions (*e.g.* select, query..). Each bar is associated with an operation, and its length encodes how many questions in the dataset are using the corresponding operation.

3.6.4 Interacting with Models

VisQA also includes features, complementary to the usual workflow introduced in Section 3.6.4, designed to investigate hypotheses on reasoning.

Free-form questions — By default, VisQA loads the GQA dataset [90] to provide images and questions. But at any time, users can type and ask free-form open-ended questions (**G4**). Such an interaction allows investigating the model's bias exploitation. For instance, when asked the following question from the GQA dataset *"Is this a mirror or a sofa"*, the model correctly outputs "*mirror*". However, when asked the following user-inputted question *"Is there a mirror in this image?"*, the model fails and outputs *"no"*. This suggests that the model might have exploited biases when it answered the first question, which is supported by the fact that in the GQA dataset, *"mirror"* is the correct answer to the question *"Is this a mirror or a sofa"* in 85% of all cases.

Head filtering (Figure 3.7) — As shown in Figure 3.7 ①, attention heat-maps feature two interactions. First, hovering with the cursor a row, a column, a cell, which corresponds to an object in the image, automatically displays its corresponding bounding box along with its label over the input (Figure 3.7 ②). The user can also, by clicking on a row, column, or cell, filter attention heads

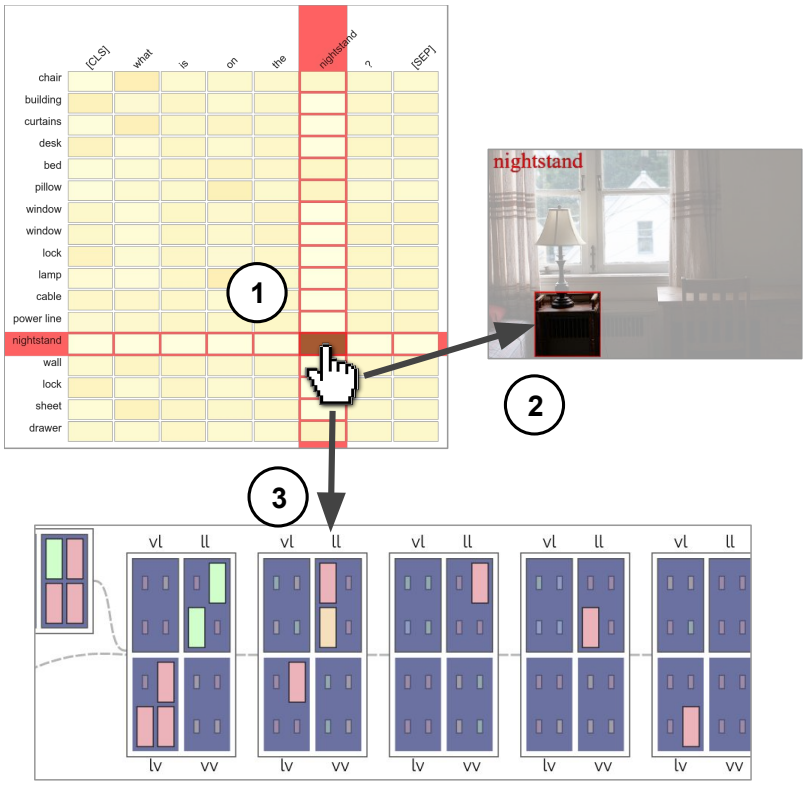


Figure 3.7 – Hovering the mouse over a cell of the attention maps ① filters the corresponding object bounding box in the input image ②. While clicking on this cell filters attention heads in *instance-view* to display those within which the selected cell is highly activated.

to only keep the ones in which the corresponding clicked element has attention above a threshold. For rows and columns which contain multiple attention values, such a filtering process will merge those values by using one of three functions *min*, *median*, and *max* depending on a user selection. The result of such a filtering, as displayed in Figure 3.7 ③, occurs in the *instance view* in which the size and opacity heads that do not match with the user's query are reduced while the others are preserved. Such interactions facilitate seeking for heads in which a specific association is expected *e.g.* a word in the question with an object of the image required to answer.

Head pruning — Users can select attention heads by clicking on them in the *instance view*, or by their *k-number* category. Such a selection can then be used to prune the corresponding head for the next forward of the model. Pruning here means that the attention head does not perform any focused attention, but uniformly distributes attention over the full set of items (objects or words). Each row of a pruned attention map is thus the equivalent of an average calculation. At any time, users can request a new forward pass of the model by clicking on the top left button "ask" (G4), which allows to see the effect of the configured

pruning on the model's predictions. This can be used in order to test hypotheses on attention head interpretations as explored in Section 3.8.2.

3.7 Implementation

The GQA [90] standard dataset provides question/image pairs along with their answers, the ground truth of bounding boxes, and semantic descriptions of questions. By default, VisQA provides around 1500 question/image pairs, but as images are loaded progressively when users request it, such a quantity can be increased without affecting performances. The models have been trained on a significantly larger amount of training data (about 9M image/sentence pairs), which is different from the validation data on which the performance is evaluated. The user interface of VisQA is implemented using D3 [22], and directly interacts with transformer models implemented in Pytorch [162], using JSON files through a python Flask server. As the possibility to ask free-form questions offers an infinity of possible combinations, each question/image pair is forwarded through the model in a plug-in fashion, *i.e.* without altering the model and its performances.

3.8 Evaluation with Domain Experts

We conducted a user study with 6 experts with experience in building deep neural networks, who were not involved in the project or the design process of VisQA. We report on their feedback using VisQA to evaluate the decision process of the *Oracle transfer model*, with 57.8% accuracy on GQA, introduced in Section 3.4 on several provided problem instances, as well as insights they received from this experience. Hypotheses drawn from single instances cannot be confirmed or denied, but as illustrated in the following sections, such a fine-grained analysis aims to provide cues (often unexpected) that can later be explored through statistical evidence outside of VisQA.

3.8.1 Evaluation Protocol

For each expert, we conducted an interview session lasting on average two hours. Sessions were organized remotely and began with a training on VisQA, showing step-by-step how to analyze attention maps. During this presentation, experts were able to ask questions. The study then began with questions on 6 problem instances, *i.e.* image/question pairs loaded into VisQA in a browser window on participants' workstations. Those instances were balanced between the prediction failures and successes, head or tail distributions of question rarity as described in Section 3.3.2, as well as our estimation on whether the model resorts to bias

for this instance grasped using VisQA. VisQA, configured as conditioned during evaluations is accessible online at: (https://theo-jaunet.github.io/visqEval/). The model outputs were hidden and the experts were asked to use VisQA to provide an estimate for two different questions: (**Q1**) will the model predict a correct answer, (**Q2**) what will it be?, and (**Q3**) does it exploit biases for its prediction, or does it reason correctly? During this part of the interview, experts were asked to explain out loud what lead them to each decision. Once those questions were completed, post-study questions were asked on the usability of VisQA, such as *"Which part of* VisQA *is the least useful?"*, and *"What was the hardest part to understand?"*.

Table 3.1 – Experts performances averaged by instances while evaluating VisQA. We can observe that Expert#3 reached the best accuracy on every questions, while Expert#5 reached the lowest results. Overall, the average performance for Q1 and Q3 is 75% of accuracy, and 61% for Q2.

Experts	Q1	Q2	Q3
Expert#1	0.6	0.5	0.67
Expert#2	0.83	0.67	0.83
Expert#3	**1**	**0.83**	**1**
Expert#4	0.83	0.67	0.5
Expert#5	0.5	0.33	0.67
Expert#6	0.67	0.67	0.83
Total	0.75	0.61	0.75

Results — The ability of users to predict failures and specific answers of VQA systems has already been addressed through evaluation [39] under different conditions. The experiment closest to ours is question+image attention [130] with instant feedback — similarly to ours, users were asked to estimate whether a model will predict a correct answer when provided with attention visualizations of the model, and reaching a similar score of ∼75% accuracy. The difference is that in [130] attention is overlaid over the visual input, whereas our attention maps allow to inspect reasoning in a more detailed and fine-grained manner, and not necessarily tied to the visual aspects. The similarity in results changes when users are asked to provide the specific answer predicted by the model: this accuracy drops to 61% in our case, and to 51% in [39]. While our results are promising, they cannot be directly compared to their results due to the different pool and amount of users. Future work will address studies on a larger number of human experts.

Figure 3.8 – When asked *"Is the person wearing shorts?"*, the *oracle transfer* model successfully answers *"yes"*. It can be observed in its first Language-to-Vision attention maps, that the word *"shorts"* (column) is strongly associated with the object *"shorts"* (row). The same phenomenon is also observed for the word *"person'*, strongly associated with objects labeled as *"woman"* among others.

More importantly, our work focuses on qualitative results of bias estimation in which experts obtained a precision of 75% on whether the model exploited any bias. We extracted the ground truth estimate by comparing the rarity of the question, following the Head/Tail attribution from GQA-OOD [107]. These results are encouraging, as they provide a first indication that the reasoning behavior of VL models can be examined and estimated by human users with VisQA. While 75% of performance reasoning vs. bias is not a perfect score, it is also far away from the random performance of 50%, which is important given the large capacity of these models, which contain millions of trainable parameters. Details on individual performance with respect to questions can be observed in Table 3.1.

3.8.2 Object Detection and Attention

To provide an answer, a model must first grasp which objects from the image are requested and thus are essential to focus on. Such an association needs to occur early in the model as those objects are needed for further reasoning. The experts widely observed high intensity in the first Language-to-Vision (LV) layer. As illustrated in Figure 3.8, when asked *"Is the person wearing shorts?"*, the attention map *LV_0_1* has peaky activations in the columns *"person"* and *"shorts"*. This can be interpreted as the model correctly identifying with its self-attention for language that those two words are essential to answer the given question. In

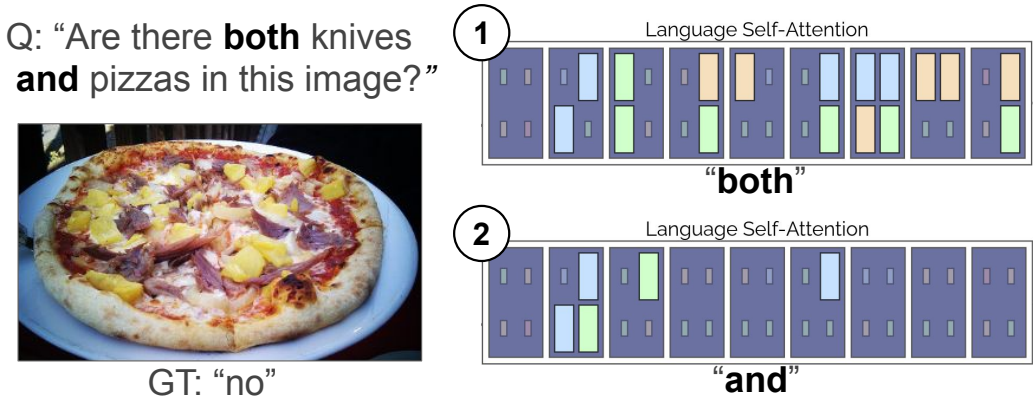


Figure 3.9 – When asked *"Are there both knives and pizzas in this image?"*, the *oracle transfer* model fails and answers *"yes"*. By filtering heads associated with a selected word, we can observe that language self-attention heads are more responsive to the word *"both"* ①, as opposed to the word *"and"* ②.

Figure 3.8, the word *"person"* is associated with the bounding boxes labeled as *"woman"*,*"shirt"*,*"shoe"*,*"leg"*, while the word *"shorts"* is associated with the *"shorts"* bounding box. Based on this observation, all experts concluded that the model correctly sees the required objects, and more broadly over the evaluation instances, that the first LV layer might be responsible for the recognition of objects with respect to the question. One of the experts mentioned that therefore, *"if we don't see a good word/bounding-box association here, the model can hardly cope with such a mistake and might exploit dataset biases"*. In order to verify such a statement, we pruned the four heads in this LV layer, to observe how the model would behave with no association in them. From such pruning, we observe that the following vision-to-language layers have lower attention distributions, close uniform in some cases. In addition, after pruning, the model's prediction wrongly switched from *"Yes"*, a rare answer (in Tail), to *"No"*, the most frequent one.

3.8.3 Questions with Logical Operators

During the evaluation, experts were shown two instances with questions containing the word *"and"*. Such instances are interesting because, as one of the experts mentioned, *"this word has a lot of importance is this question"*. To answer correctly, the model needs to grasp that it must analyze the image over two different aspects. With the image, illustrated in Figure 3.9, and asked *"Are there both knives and pizzas in this image?"*, the model fails and answer *"yes"*, the most frequent answer despite having no knife in the picture nor provided bounding-boxes. However, when asked *"Are there knives in this image?"* the model correctly answers *"no"*. This suggests that the model failed to grasp the meaning of the keyword *"and"*, and thus that the self-attention language heads might associate

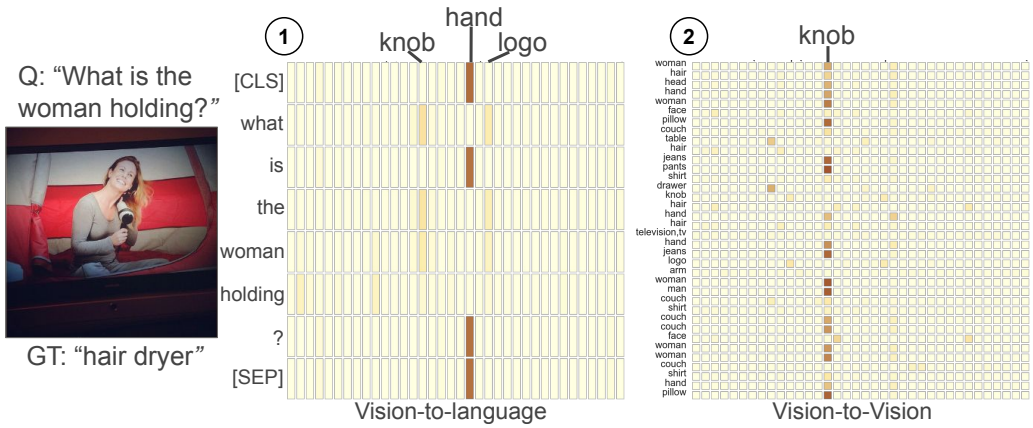


Figure 3.10 – Without any *"hair dryer"* provided by the object detector, the *oracle transfer* associates in its vision-to-language ① the object *"hand"* with the words *{"[CLS]","is","?","[SEP]"}*. While vision-to-vision focuses on a *"knob"* object ②.

wrong words. Also, swapping the terms *"knives"* and *"pizzas"* in the question, yields the correct answer, *i.e. "no"*. This indicates that the model ignores the first term when questions contain the operator *"and"*. Using the head-filtering interaction, we can observe that in self-attention heads, the word *"and"* has little to no attention. Instead, the word *"both"* has peaky attention scattered across most of self-language layers, and some language-to-language heads. Pruning those 19 heads makes the model correctly yield *"no"*, regardless of the order the words *"knives"* and *"pizzas"* are in the question. Such a behavior can be observed over our evaluation dataset, in which 34 questions have the keyword *"and"*. On those questions the model, without pruning, can provide a correct answer 62% of cases, up to 64% with the two words around *"and"* are swapped. In opposition, while having the 19 attention-heads with peaky attention for the word *"both"* pruned, the model reached an accuracy of 76%, down to 74% with words around *"and"* swapped. Thus, in the worst case, this pruning of the 19 attention heads illustrated in Figure 3.9 is responsible for an improvement of 10% on question containing the operator *"and"*.

3.8.4 Vision to Vision Contextualization

When asked *"What is the woman holding?"*, with the image in Figure 3.10, the *Oracle transfer* model fails and outputs *"remote-control"*, a frequent answer, instead of *"hair dryer"*. This can be interpreted as the model exploiting a statistical bias from its training dataset. However, in such a dataset, *"remote-control"* is not among the 10 most common answers to this question. This raises the question of what leads the model to output such an answer. During evaluation on this instance, experts noticed that the object detector failed to provide a *"hair dryer"* object. Similar to the use case given in Section 3.4, such a mistake forces the *Oracle*

transfer to draw its attention towards other bounding boxes related to the missing object. In this case, as observed by experts, a majority of the vision-to-language reached their highest association between the word *"holding"* and bounding boxes labeled as *"hands"*. Such an association is expected as held objects are directly related to hands, and no *"hair dryer"* bounding box is provided. Among those bounding boxes, we can observe the presence of one labeled as *"television"*, and another as *"knob"* which are associated to *"holding"* and *"woman"* in both vision-to-vision_2_2 and early vision-to-language layers. This suggests that those heads might have influenced the model's predictions towards *"remote-control"* instead of the most common dataset bias. This can be confirmed by pruning those heads which yields a more frequent answer: *"cell phone"*. In addition, one of the experts also highlighted that those attention heads had a high association with the tokens *"[CLS]"*, *"is"*, *"?"*, and *"[SEP]"*. Which the expert interpreted as *"the model correctly transferred the context of the question"*.

3.9 Discussions, Limitations and Future Work

Usability of VisQA — Overall, VisQA was positively received by experts, during discussions at the end of interviews, they expressed that VisQA is *"well designed"*, *"complete"*, and *"particularly useful as VQA transformers are hard to interpret"*. Two out of the six experts confessed that they felt *"overwhelmed at first"*, but gradually *"grasped where to look"*, and getting *"used to interactions"*. Four out of six experts stated that the *head-statistics* Figure 3.6 ④ is the least useful feature implemented in VisQA, as it felt *"hard to understand"*. One expert mentioned that such a view is *"useful in theory, but less usable in practice"*. However, the rest of the experts mentioned that the *head-statistics* view helped them while analyzing attention to *"see if heads acted out of their distribution"*. Experts were unanimous, the attention heat-map, and in particular, its interactions, is the most useful feature of VisQA, as it *"provides a lot of information'*, and *head filtering "speeds up the analysis"*. One of the experts used this interaction on the first language to vision layer as an entry-point to grasp if the model had every information required to correctly answer the given question.

Expert Suggestions — As expressed by experts during evaluation, the main limitation of VisQA is how instances can be selected by users. Currently, such selections are handled by a bar at the top of the tool (Figure 3.6 ①), displaying images ordered from left to right based on the likelihood of their questions to be answered using statistical biases. However, during interviews, experts mentioned their need to quickly switch between similar instances in order to grasp if a behavior can be seen across different cases. To do so, experts suggested that such similarity could be measured at three levels: switch to an instance with

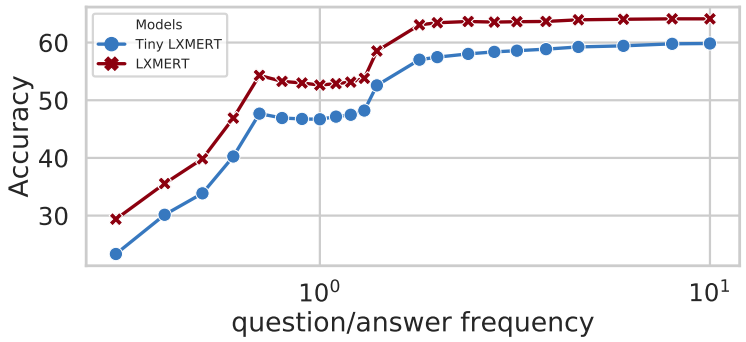


Figure 3.11 – Despite different performances, we can observe that both **tiny-LXMERT** and **large-LXMERT** have the same behavior as the frequency of question/answers increases. Hence, we can estimate that they may be exploiting similar shortcuts defined in GQA-OOD [107].

the exact same question, switch to a similar image, and switch to an instance in which a selected head has similar attention distribution. In addition, currently in VisQA, it is difficult to evaluate how influential a head is over the model's output. To tackle such an issue, experts proposed to encode this information in the representation size of the *instance-view* attention heads. The impact of each head over the model's input could be retrieved through back-propagation, in particular by calculating the gradient of the model outputs with respect to a statistic of the attention head, for instance, its k-number. In addition to addressing those limits and experts' suggestions, we plan as future works to adapt the usage of VisQA on other problems involving transformers, such as machine translation, and to further investigate the role of each attention heads as done in [221].

Scalability — To date, the largest model loaded in VisQA is LXMERT with 12 attention heads per block and 768 hidden dimensions. As illustrated in Figure 3.11, such a model (in red), and tiny-LXMERT (in blue) experiments the same performance drops as question/answer frequencies decreases, for average accuracy of 57.8% on GQA, compared to 59.6% for the large-LXMERT. It is also worth noting that tiny-LXMERT can outperform the standard LXMERT with auxiliary losses [109]. In addition, LXMERT is less accurate than the tiny-oracle model we used for evaluation, on questions with rare answers–*i.e.* those that may be the most susceptible to convey biases. While inspecting this model with VisQA, we noted that the more the number of heads increases, the more summary and filtering of heads became relevant for faster analysis. However, our visual encoding of heads may become tedious to use as the number of heads increases, and the number of pixels allocated per head decreases. A workaround can be to increase the designated space of the model in instance-view, but ultimately, VisQA will be limited

by screen real-estate. Thus, larger models (*e.g.* UNITER-large [41]), may need their heads to be filtered (*e.g.* by k-numbers) to avoid being overwhelming, before being displayed in VisQA. Similarly, heatmaps of attention may suffer from the same limitation. However, we decided to use them for attention as the maximum visual object (36 in most VQA systems using a Faster R-CNN), and the average of words per question is known beforehand. For larger sets of inputs, alternative designs such as bipartite graphs combined with aggregation methods to hide elements under a threshold of attention may be more space-efficient. However, we argue that this is a trade-off as displaying every bounding boxes and their attention may be relevant to evaluate a prediction (as depicted in Section 3.4).

Generalization — This work focuses on detector-based VL transformers such as LXMERT, however other VQA systems may include single stream models such as [41, 196]. While VisQA is applicable to both single and 2-stream models, we decided to focus on the latest during evaluation. Those models are considered by experts as more interpretable because self-attention layers for language and vision can be observed separately, and there are no significant differences in performance between both architecture [30]. VQA systems' state-of-the-art often alternate between detector-based approaches (*e.g.* VinVL [240]) and pixel models [89]. However, at its current state, VisQA is not applicable to the latter type, as it would only require adapting to the higher number of visual tokens (depending on the granularity of the method) which raises the challenge of scalability of heatmaps discussed above. Future works can address this through aggregation methods to reduce the number of tokens displayed (*e.g.* by dividing input images into regions). Finally, our work mainly focuses on insights on reasoning skills grasped from models on the GQA dataset. This decision was taken because it has been argued [90] that it involves a larger variety of reasoning skills (spatial, logical, relational, and comparative) than in a dataset with where questions were annotated by humans directly (*e.g.* VQA2 [76]) which contains questions targeting difficult external knowledge (*e.g.* a name of a baseball team). Despite not being presented in this work, VisQA can directly be used on the VQA2 dataset.

Comparison of instances (Figure 3.12) — Comparison between models, and instances can also yield interesting insights on how a model behaves [53]. Currently, to this end, VisQA memorizes inputs and all intermediate and final results including *k-numbers* and attention maps. This state can be saved, and then used at any time and compared to a new current instance through the *compare* button at the top of the *instance-view*. The comparison itself is obtained by computing the difference of *k-numbers*, and complete attention maps. As a result, head representations in *instance-view* now encode, with a single hue color scale, the difference between the two attention maps. Besides, as shown in Figure 3.12, attention heat-maps also encodes such a difference using a diverging color scale

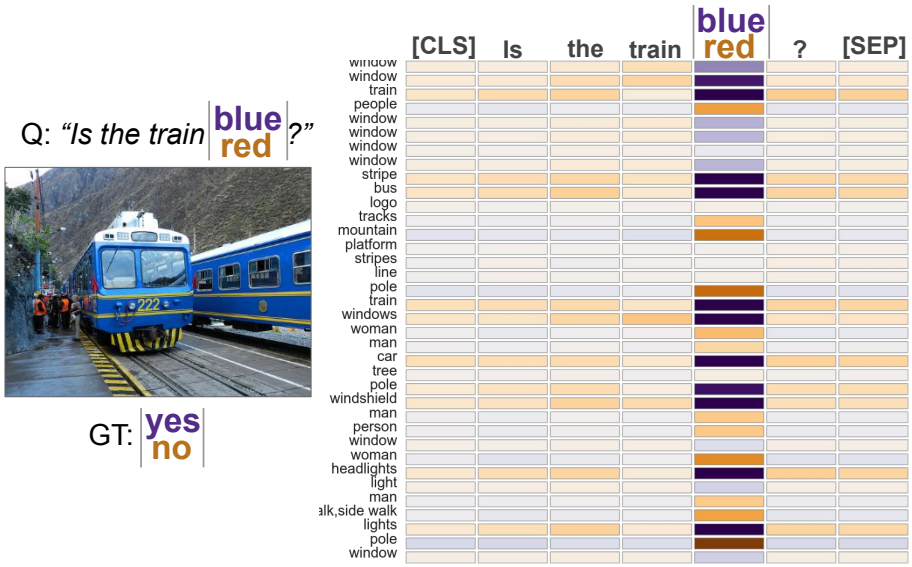


Figure 3.12 – Difference between the attentions of Head LV_1_0, when asked "is the the train blue?", and "is the the train red?". We can observe that in this head, the attention focuses on different objects (row) depending on the color asked (column).

from dark blue for values close to -1, *i.e.* a cell with high attention in the previous instance but not in the current one, to brown for values close to 1, the opposite. Comparisons are particularly useful when combined with the possibility to ask free-form questions. As it can be observed in Figure 3.12, the manually asked questions *"Is the train blue?"* and *"Is the train red?"* are responsible for different attention modes between the word representing the color and bounding boxes of the image. By browsing those bounding boxes, it can be observed that the selected head *LV_1_00* associates them, in this case, to the word color only if their color matches. As an example, in Figure 3.12 (right), the third row, which corresponds to the bounding box of a blue train, is associated through attention with the word *"blue"* but not the word *"red"*. In order for such a comparison to be relevant, shifts between two instances must be on a few words of the question, as otherwise, attention maps rows and columns would not align between the instances, and thus any difference would occur for wrong reasons. We plan for future work to enhance this functionality of VisQA through better visual encoding, more intuitive interactions, and an evaluation.

3.10 Conclusion

We introduced VisQA, an interactive visual analytics tool designed to perform an instance-based in-depth analysis of the reasoning behavior transformer neural networks for vision and language reasoning, in particular visual question answering. VisQA allows users to select display VQA instances based on the likelihood of

bias exploitation; to display attention head intensities; to inspect attention distributions; to prune attention heads; and to directly interact with the model by asking free-form questions. Our quantitative evaluations are encouraging, providing first evidence that human users can obtain indications on the reasoning behavior of a neural network using VisQA, i.e. estimates on whether it correctly predicts an answer, and whether it exploits biases. VisQA received positive feedback from these experts, who also provided additional qualitative feedback on the nature of the information they extracted on the behavior of different neural models.

CHAPTER 4

NAVIGATION

Contents

Thanks to VQA as introduced in Chapter 3, we have robots able to understand what we are looking for, and seek within an image for it. The next step to *find our keys*, is for robots to successfully explore an environment (*e.g.* our apartment) as depicted in Figure 4.1. The following chapter addresses **step** ②, referred to as navigation, and how visual analytics systems can be designed to interpret models trained with Deep Reinforcement Learning (DRL) and ultimately improve them.

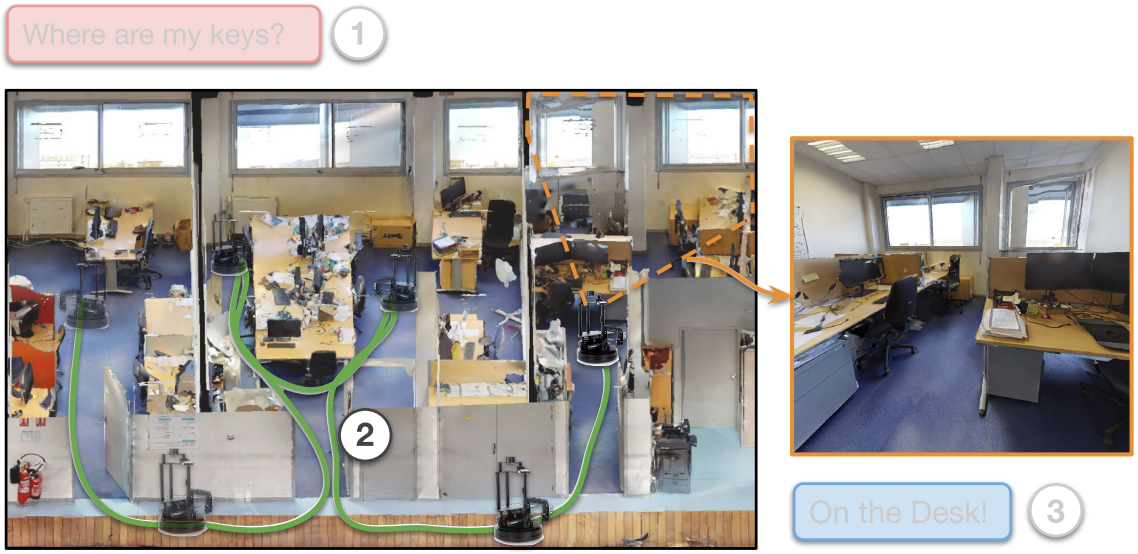


Figure 4.1 – This chapter is dedicated to **step** ② of *finding my keys*, *i.e.* the ability for robots to explore unknown environments. To do so, such a robot relies on images sampled from an onboard camera (most right) to decide what should be its next direction.

In this chapter, we present DRLViz, a visual analytics interface to interpret the internal memory of an agent (e.g. a robot) trained using DRL. This memory is composed of large temporal vectors updated when the agent moves in an environment and is not trivial to understand due to the number of dimensions, dependencies to past vectors, spatial/temporal correlations, and co-correlation between dimensions. It is often referred to as a black box as only inputs (images) and outputs (actions) are intelligible for humans. Using DRLViz, experts are assisted to interpret to investigate the role of parts of the memory when errors have been made (e.g. a wrong direction). In this chapter, DRLViz is applied in the context of video games simulators (ViZDoom) for a navigation scenario with item gathering tasks in order to speed up the training process. The challenge of transferring the knowledge acquired in simulation to robots in the real-world will be addressed in Chapter 5. DRLViz has been evaluated by experts.

Finally, we conducted the experimentation of pruning the memory of DRL agents down to core identified element, and evaluated the impact on their performances.

This was inspired by Macdonald et al. [132]'s definition of interpreting Deep Neural Networks (DNN)s, to only consider a set of features relevant if a model's output distribution is preserved despite randomizing the remaining features. We also designed *MemRed*, an online explorable of such an experiment, tailored for non-expert users, which is detailed in Section 4.7.1 of this chapter.

This chapter is related to the following published materials:

- Théo Jaunet, Romain Vuillemot, and Christian Wolf. "DRLViz: Understanding Decisions and Memory in Deep Reinforcement Learning". *Computer Graphics Forum (Proceedings of EuroVis 2020), 2020.*

- Théo Jaunet, Romain Vuillemot, and Christian Wolf. "What if we Reduce the Memory of an Artificial Doom Player?". *IEEE Workshop on Visualization for AI Explainability at IEEE VIS (VISxAI), 2019.*

Both code (`https://github.com/sical/drlviz`), and an limited interactive prototype (`https://sical.github.io/drlviz`) of DRLViz are available online. Similarly, the *MemRed* (`https://theo-jaunet.github.io/MemoryReduction/`) and its code (`https://github.com/Theo-Jaunet/MemoryReduction`), are also accessible.

4.1 Introduction

Automatic navigation is among the most challenging problems in Computer Science with a wide range of applications, from finding the shortest paths between pairs of points, to efficiently exploring and covering unknown environments, up to complex semantic visual problems ("*Where are my keys?*"). Addressing such problems is important for modern applications such as autonomous vehicles to improve urban mobility, social robots, and assisting elderly people. Historically, navigation was often solved with discrete optimization algorithms such as Dijkstra [55], A-Star [82], Front-propagation [234] etc., applied in settings where spatial maps are constructed simultaneously with solving the navigation problem. These algorithms are well understood, but are restricted to simple waypoint navigation. Recently, techniques from Machine/Deep Learning have shown spectacular progress on more complex tasks involving visual recognition, especially in settings where the agent is required to discover the problem statement itself from data. In particular, Reinforcement Learning (RL) and the underlying Markov Decision Process (MDP) provide a mathematically founded framework for a class of problems focusing on interactions between an agent and its environment [200]. In combination with deep networks as function approximators, this kind of model was very successful in problems like game playing [145, 186], navigation in simulated environments [48, 75, 159], and work in Human-Computer Interaction (HCI) emerging[51].

The goal of Deep Reinforcement Learning (DRL) is to train agents which interact with an environment. The agent sequentially takes decisions a_t, where t is a time instant, and receives a scalar reward R_t, as well as a new observation o_t. The reward encodes the success of the agent's behavior, but a reward R_t at time t does not necessarily reflect the quality of the agent's action at time t. As an example, if an agent is to steer an autonomous vehicle, receiving a (very) negative reward at some instant because the car is crashed into a wall, this reflects a sequence of actions taken earlier than the last action right before the crash, which is known as the *credit assignment problem*. The reinforcement learning process aims at learning an optimal policy of actions which optimizes the expected accumulated future reward $V_t = \sum_{t'=t}^{t+\tau} R_t$ over a horizon τ.

If agents trained with DRL were deployed to real-world scenarios, failures and unexpected behaviors [117] could lead to severe consequences. This raises new concerns in understanding on what ground models' decisions (*e.g.* brake) are based [172]. To assess the decision of a trained model, developers [87] must explore its context (*e.g.* a pedestrian on the road, speed, previous decisions) and associate it with millions of deep networks parameters which is not feasible manually. Analyzing a decision after-the-fact, referred to as post-hoc interpretability [126], has been a common approach in visualization. It consists in collecting any relevant information such as inputs and inner-representations produced while the model outputs decision. With such an approach, DRL experts explore their models without having to modify them and face the trade-off between interpretability and performances. Visual analytics for post-hoc interpretability [87] yields promising results on tasks such as image classification [157], or text prediction [193]; however, it remains an under-explored challenge for DRL *with* memory.

We built DRLViz, a novel Visual Analytics interface aimed at making Deep Reinforcement Learning models with memory more interpretable for experts. DRLViz exposes a trained agent's memory using a set of interactive visualizations the user can overview and filter, to identify sub-sets of the memory involved in the agent's decision. DRLViz targets expert users in DRL, who are used to work with such models (referred to as *developers* in [87]). Typically, those experts have already trained agents and want to investigate their decision-making process. We validated DRLViz usability with three experts and report on their findings that informed us on future improvements such as applicability to other scenarios, and novel interactions to reduce the memory of an agent and better find patterns within it.

4.2 Context and Background

The context of our work is related to building deep neural network models to train robots to achieve human assistance tasks in real-world environments.

As the sample efficiency of current RL algorithms is limited, training requires a massive amount of interactions of the agent with the environment — typically in the order of a billion. Simulators can provide this amount of interactions in a reasonable time frame, and enable to work with a constantly controlled world, that will generate less noise (*e.g.* a shade) in the agent's latent representation. We will discuss in the perspectives section, and in Chapter 5 the extension of our work beyond simulators and the knowledge transfer from simulation to real-world scenarios, where variability (*e.g.* lighting, clouds, shades, etc.) and non-deterministic behaviors (*e.g.* robots may turn more or less depending on its battery charge) occur.

4.2.1 Navigation Problem Definitions

Our focus is on navigation problems, where an *agent* (*e.g.* robot, human) moves within a 2D space we call *environment* (Figure 4.2). An environment contains obstacles (*e.g.* walls), items the agent may want to gather or avoid, and is usually bounded (*e.g.* a room). The goal of the agent can vary according to the problem variation, but typically is to reach a particular location (*e.g.* gather items, find a particular spot). Importantly, the goal itself needs to be discovered by the agent through feedback in the form of a scalar reward signal the environment provides: for instance, hitting a wall may provide a negative reward, finding a certain item may result in a positive reward. To discover and achieve the goal, the agent must explore its environment using actions. In our case, those actions are discrete and elements of the following alphabet: $a \in A$, with $A = \{forward, forward+right, right, left, forward+left\}$. The navigation task ends when the agent reaches its goal, or when it fails (*e.g.* dies, timeout).

As the agent explores its environment, it produces a trajectory. A trajectory is a series of positions p (x,y) in a space S bounded by the environment. Those positions are ordered by time-step $t \in T$, where $t_0 < t_1 < t_n$, and the interval between t_n and t_{n+1} is the time the agent takes to act. In addition to positions, trajectories contain complementary attributes b, which may vary depending on the agent goal (*e.g.* number of gathered items, velocity, etc.). We call $step_t$ the combination of both the agent position p and its attributes b, at a given time-step t. Thus $step_t$ can be represented as follows $<p_t, b_t>$. The transition between steps occurs as the agent makes a decision. An *episode* groups all iterations from the first step at t_0, until the agent wins or looses at t_n.

4.2.2 Navigation using the ViZDoom Simulation

The simulation environment we use to train agents to navigate is ViZDoom [103] which provides instances of the navigation problem based on Doom, a very

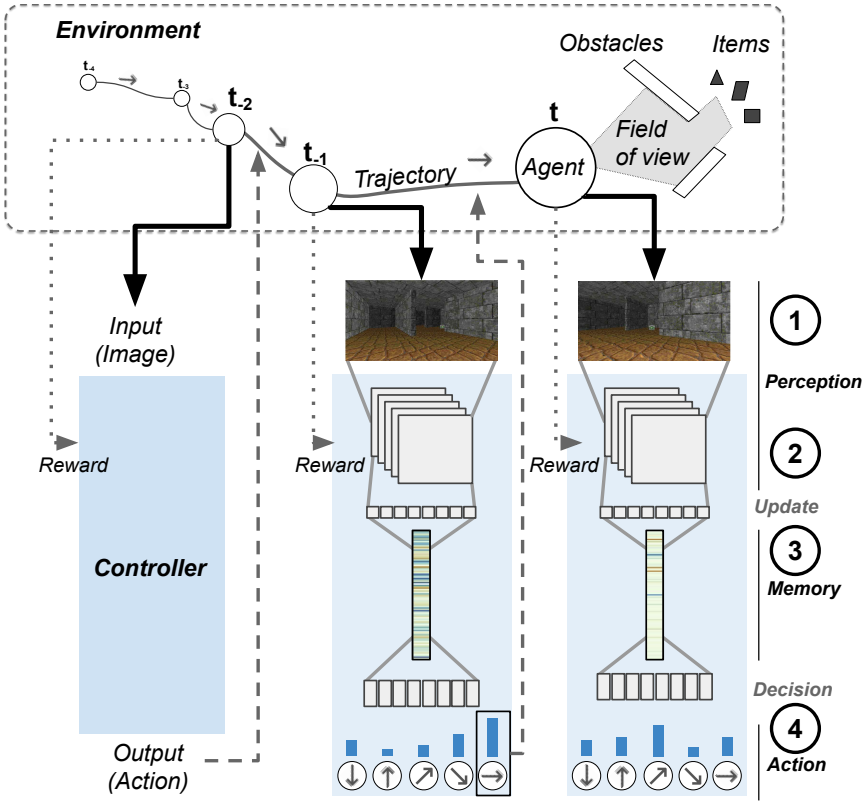


Figure 4.2 – Our navigation problem consists in solving a visual task (*e.g.* fetch, interact, or recognize items) while avoiding obstacles in an environment. Deep Reinforcement Learning can be used to solve this problem by using an image as input ① at time t. Features are then extracted from this image ②, and combined with the previous memory vector $t-1$ ③. Using this memory vector, the agent decides to move forward or turn left, for instance ④.

popular video game in the '90s. ViZDoom is a 3D world and as such is a proxy problem to mobile service robotics. It supplies different scenarios focusing on various goals (*e.g.* survive, reach a location, gather items, avoid enemies, etc.). For expert evaluation, and Figure 4.2 we used the *k-items* scenario from [18] with $k = 4$. In this scenario, the agent, walls, and items are randomly placed in an environment at the beginning of each episode. Then the agent needs to explore the environment until it succeeds, fails or reaches a timeout of 525 steps. To succeed the agent must first gather a green armor, then a red armor, followed by a Health Pack, and finally a soul-sphere (blue circle). Gathering the items in another order instantly kills the agent and ends the episode (fail). Gathering an item in the right order grants a $+0.5$ reward r, while failing grants a reward of -0.25. Additionally, the agent receives a reward of -0.0001 at each step. This scenario has been designed to provide sufficient learning signals to the agent, pushing it towards learning the positions of objects in maps. This is not the case

for simpler tasks requiring to search an object like "ObjectNav" [14], as in this case collecting an object only requires to reach it immediately upon its detection.

Despite ViZDoom being a 3D world, the agent positions p are within a bounded continuous 2D plane corresponding to the bird's eye view of the environment. We summarize a time-step t as follows: $<p_t, (r_t)>$.

This task is challenging as the agent is required to take a decision on the next action based on partial information of the environment, *i.e.* the task is partially observable. The observed image represents the agent's Field of View (FoV) (*i.e.* what is in front of it), in a 90-degree range and unlimited depth. The agent is required to recall previously seen observations in some way as it doesn't have access to a global view of the map. These views are stored in its latent memory, the representation studied in this work. The agent should also use its memory to encode information on the items it gathered, and the positions of items or walls encountered in order to quickly complete this task.

4.2.3 Deep Reinforcement Learning and Memory

As expressed in the taxonomy [10], DRL reached state-of-the-art performance in tasks such as robot control [118] and board games [186, 94] where it even surpasses humans in some cases. Recent Deep Reinforcement learning models, such as Deep Q-Networks (DQN)s [144, 145], and Asynchronous Advantage Actor-Critic (A3C) [146], learned to play video games with human-level control using only images as input. As a result, they achieved human-level performances on Atari 2600 games [19] such as breakout. Those models rely on the hypothesis that the optimal action can be decided based on a single frame.

However, these approaches operate on environments that can be totally observed (like chess or GO board game), and not partially observable, with a field of view that is smaller than the environment. To address this, an internal latent memory can be introduced [83] to provide a space the model can use to store an approximation of the history of previous inputs and solve navigation problems [142, 245, 154], allowing learning in simulated environments such as Matterport3D [40], ViZDoom [103, 18].

4.2.4 Visual Analytics and Deep Reinforcement Learning

Thanks to analyses conducted with the help of visual analytics systems addressing sequential models (introduced in Chapter 2, Section 2.3.2), it has been observed that a decision at a time-step t can be affected by an input seen at $t - n$. In our case, such inputs are images and experts must first assess what the model did grasp from them before exploring what is stored in the memory. In addition, the rewards from navigation tasks are often sparse which results in a lack of

supervision over actions, known as the credit assignment problem inherent to RL problems (the reward provided at a given time step can correspond to decisions taken at arbitrary time steps in the past). The model interacts with an environment it only sees partially, therefore, its performances can be altered by factors outside its inputs. This forces experts to visualize multiple time-steps in order to analyze a single decision which makes them more difficult to analyze with existing tools.

To our knowledge, DRL visualizations are under-represented in the literature compared to other methods of visualizing deep learning. LSTM activations from an A3C agent [142] have been displayed using a t-SNE [217] projection. Despite being effective for an overview of the agent's memory, it offers limited information on the role of the memory. T-SNE projections have also been applied to memory-less DRL agents on 2D Atari 2600 games, in the seminal DQN paper [145], and in [238]. DQNViz [226] displays the training of memory-less models under 4 perspectives. First an overview of the complete training, action distribution of one epoch, a trajectory replay combined with metrics such as rewards, and whether an action was random. DQNViz also includes a detailed view to explore Convolutional Neural Network (CNN) parameters. Such a tool demonstrates the effectiveness of visual analytics solutions applied to DRL. However, DQNViz focuses on the training of the model, and how random decisions through training can affect it. In addition, the model of DQNViz is limited to fully observable 2D environments in which the only movements available are left or right and thus cannot be applied to navigation tasks. Finally, DQNViz is not designed to display or analyze any memory.

In this chapter, we address the under-explored challenge of visualizing a trained DRL model's memory in a 3D partially observed environment. We contextualize this memory with output decisions, inputs, and derived metrics. We also provide interaction to overview, filter, and select parts of such memory based on this context to provide clues on agents' decision reasoning and potentially identify how the model uses its memory elements.

4.3 Model and Design Goals

This section presents the model we used to design and implement DRLViz. We describe the inner workings of those models and data characteristics. One key aspect is how the memory of DRL is created and updated by the agent, over space and time. Note that those data will be generated and then visualized with DRLViz after the training phase.

4.3.1 DRL Model

The DRL model we relied on only receives pixels from an RGB image as input, from which it decides the action the agent should perform with the Advantage Actor-Critic (A2C) [146] algorithm. The model is composed of 3 convolutional layers followed by a layer of Gated Recurrent Unit (GRU) [42], and Fully Connected (FC) layers to match the actions set A. This model is inspired by *LSTM A3C* as presented in [142] with A3C instead of A2C, and a LSTM [78] instead of GRU. Those changes reduce the agent's training time while preserving its performance. The underlying structure that allows our model to associate raw pixels to an action is illustrated on Figure 4.2 and described as follows:

Stage ①: Environment → Image. First, the agent's field of view is captured as image x_t, i.e. a matrix with dimensions of 112×64 with 3 RGB color channels.

Stage ②: Image → Feature vector. The image x_t is then analyzed by 3 convolutional layers designed to extract features f_t, resulting in a tensor of 32 features shaped as a 10×4 matrices. These features are then flattened and further processed with a FC layer. Formally, the full stack of convolutional and FC layers is denoted as function $f_t = \Phi(x_t, \theta_\Phi)$ with trainable parameters θ_Φ taking x_t as input and given features f_t as output.

Stage ③: (Features + previous memory) → New memory. The model maintains and updates a latent representation of its inputs using a GRU [42] layer, a gated variant of Recurrent Neural Network (RNN). This representation, called hidden state h_t, is a time varying vector of 128 dimensions, which is updated at each time-step t with a trainable function Ψ taking as input the current observation, encoded in features f_t, and the previous hidden state h_{t-1}, as follows: $h_t = \Psi(h_{t-1}, f_t, \theta_\Psi)$.

Stage ④: Memory vector → Action. The model maps the current hidden state h_t to a probability distribution over actions A using a fully connected layer followed by a softmax activation function, denoted as the following trainable function: $a_t = \xi(h_t, \theta_\xi)$ with trainable parameters θ_ξ. The highest probability corresponds to the action a_t which the agent estimated as optimal for the current step t.

The full set of parameters $\theta = \{\theta_\Phi, \theta_\Psi, \theta_\xi\}$ is trained end-to-end. We used 16 parallel agents and updated the model every 128 steps in the environments. The gamma factor was 0.99, and we used the RMSProp [211] optimizer with a learning rate of $7e-4$. We trained the agent over 5 million frames, with a frame-skip of 4. During training, the agent does not necessarily pick the action with the highest probability, as it needs to explore its environment, and eventually find better

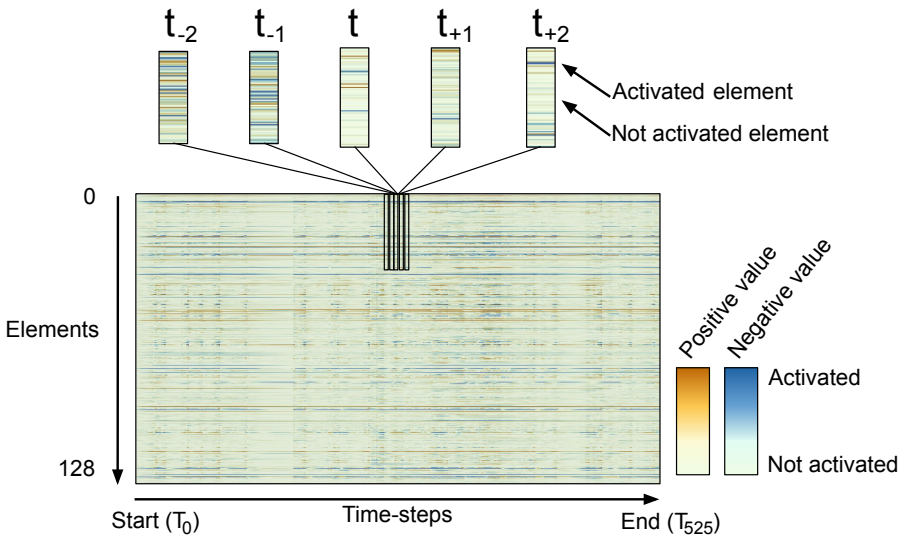


Figure 4.3 – Memory construction process: at a current time-step t, the agent updates its memory by producing a new memory vector. Each dimension of this vector (represented as a column) is appended to the previous ones chronologically (from left to right). As a result, each row of the appended vectors represent the actions of a single memory element.

solutions. However, once the agent is trained, it always chooses the action with the highest probability.

4.3.2 Constructing the Memory of DRL

In the partially observed navigation problem we focus on, the agent only sees the current observation, *i.e.* what is in its field of view at the time-step t. However, past observations are also relevant for decision making (*e.g.* to gather previously seen items). Therefore the agent needs to build a representation of relevant information extracted from the history of observations. This information is encoded in h_t, a high dimensional (128 in our case) time-varying vector.

Figure 4.3 represents the construction process of the hidden states matrix, which consists of the hidden states h_t over the time of an episode — the central visualization in DRLViz (Figure 4.4). Each hidden state is vertically aligned per time-step t at which they are produced. Therefore, the accumulation of hidden states forms a large 2D matrix, where the horizontal axis is time ($h_{t-1} < h_t < h_{t+1}$) and the rows are elements. A row of this 2D matrix represents the evolution and activity of a hidden state element through time and space as the agent moves. The activity of a hidden state element is characterized by its value. In our case, each element of the hidden states is a quantity within the range $[-1, 1]$. A value close to 0 represents low activity, whereas a value close to any extremity represents high activity. As it can be seen in Figure 4.3, hidden states can drastically change their values between two time-steps. Such value changes can be widely observed

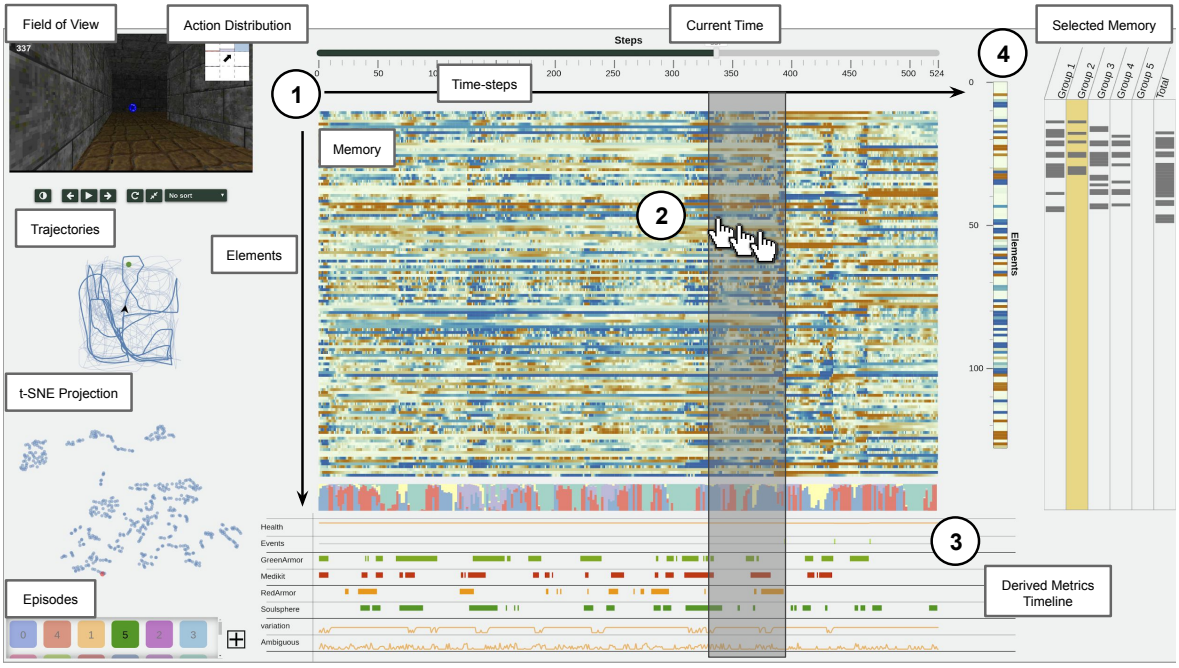


Figure 4.4 – DRLViz displays a trained agent memory, which is a large temporal vector, as a horizontal heat-map ①. Analysts can *browse* this memory following its temporal construction; *filter* according to movements of the agent and derived metrics we calculated ② (*e.g.* when an item is in the field of view ③); and *select the memory* to filter elements and compare them ④.

across hidden states elements during episodes. However, it remains unclear which elements, correspond to which representations, and thus, are responsible for decisions.

Norms of latent activations are an informative way of visualizing influences [31, 242]. With modern training methods such as weight decay, dropout, and batch normalization, it is highly improbable that a high activation can occur for unused features. An alternative to hidden state activations would be to analyze gradients of action probabilities with respect to hidden states [183, 37]. Such an approach can provide information on how a chosen action is directly tied to the current state of the memory, and which dimension influences the more this decision. However, in DRLViz we focus on actions through the episode as a sequence rather than small sub-sequences. When visualizing activations of an LSTM on text, Karpathy et al. [99] discovered a pattern occurring outside back-propagation limitations of the gradient signal. A solution would be to display both activations and gradients, however preserving the usability and interpretability of a tool conveying such information is a challenge yet to be tackled.

4.4 Design of DRLViz

We built DRLViz as a visual analytics interface to understand the connections between the latent memory representation (as depicted in Figure 4.3) and decisions of an agent trained using Deep Reinforcement Learning. DRLViz primarily exposes the internal memory (Figure 4.4) which is interactive and provides *overviewing*, *filtering* and *reduction* both for exploration and knowledge generation [101]. DRLViz is designed towards experts in DRL to identify elements responsible for both low-level decisions (*e.g.* move towards a spotted Health Pack) and eventually higher-level strategies (*e.g.* optimizing a path).

4.4.1 Design Motivation and Goals

We iteratively built DRLViz with frequent meetings from colleagues experts in Deep Learning and DRL (a total of 12 meetings with three experts over 7 months). We first identified their current process to analyze trained agents, *e.g.* recording videos to playback agents episodes (from its point of view) and decisions (actions probability) to get a sense of the strategy. We also observed experts do a system print of the models' inner values, sometimes add conditions to those prints (*e.g.* when an item is in the field of view of the agent), and manually look for unusual values. Our approach was to re-build a similar interface in DRLViz with input/output views and facilitate playback, but 1) in an interactive way, and 2) by adding advanced, coordinated views to support advanced models visualization aimed at models developers [87] (*e.g.* view on the agent's memory).

Based on a review of the current practices of researchers from our focus group, and related work, we identified the following design goals (**G**) to be addressed to understand the behavior of a trained agent using a learning model for navigation problems:

- **G1** **Show an agent's decisions over (a) space and (b) time**, especially input and outputs of the model.
- **G2** **Expose the memory's internal structure**, *i.e.* the temporal vector built over time (Figure 4.3).
- **G3** **Link memory over (a) time and (b) decisions** with multiple endpoints, *e.g.* starting from any time, location, memory or trajectory point.
- **G4** **Filter a sub-set of the memory (a sub-space)** tied to a specific agent behavior or strategy.

4.4.2 Overview and Workflow of DRLViz

Figure 4.4 shows an overview of DRLViz where the most prominent visualization is the memory timeline of a trained agent (**G2**). The primary interaction is

browsing the timeline and playback the input video feed and action probabilities (**G1**). Beyond re-playing scenarios, DRLViz implements multiple interactions to:

1. *Overview* the memory and check what the agent sees and its decisions; visual cues for selection are dark, consecutive patterns (Figure 4.3).
2. *Filter* the timeline when something is of interest, *e.g.* related to the activation, but also with additional timelines (actions, etc.).
3. *Select* elements whose activation behavior is linked to decisions. Those elements are only a subset of the whole memory and are visible on Figure 4.4 ④.

Those interactions are carried out using a *vertical thumb* similar to a slider to explore time-steps t and select intervals. Such a selection is propagated to allow the views on the interface, whose main ones are *image (perception)* and *probabilities (of actions)* which provide context on the agent's decisions (**G1** (b)). The input image can be animated as a video feed with the playback controls, and a saliency map overlay can be activated [190, 79] representing the segmentation of the image by the agent. The *trajectories* view (Figure 4.4) displays the sequence of agent positions $p_{t-1} > p_t > p_{t+1}$ on a 2D map (**G1** (a)). This view also displays the items in the agent's field of view as colored circles matching the ones on the timeline. The position p_t, and orientation of the agent are represented as an animated triangle. The user can brush the 2D map to select time-steps, which filters the memory view with corresponding time-steps for further analysis (**G3** (a)). DRLViz, also includes a *t-SNE* [217] view of time-steps t using a two-dimensional projection (Figure 4.4 bottom left). T-SNE is a dimensionality reduction technique, which shows similar items nearby, and in this view, each dot represents a hidden state h occurring in a time-step t. The dot corresponding to the current time-step t is filled in red, while the others are blue. The user can select using a lasso interaction clusters of hidden states to filter the memory with the corresponding time steps. Dimensions among the selected hidden states can then be re-ordered with any criteria listed in Table 4.1, and brushed vertically (Figure 4.4 ④).

The result of such an exploratory process is the subset of elements of the memory (rows) that are linked to an agent's decision (Figure 4.4 ④). This addresses the design goal **G4**. Such subset can be seen as a memory reduction that can be used as a substitute to the whole memory (we will discuss it in the perspective sections). This subset can also be used in other episodes listed as clickable squares at the bottom left corner of DRLViz.

4.4.3 Memory Timeline View

The *memory timeline* exposes the memory's internal structure (**G2**), which is vector (vertical column) of 128 dimensions over 525 time-steps as a heat-map (Figure 4.4 ②) from which an interval can be brushed for further analysis. Each cell (square) encodes a quantitative value, whose construction is illustrated in

Table 4.1 – List of re-ordering criteria as they appear in DRLViz. t is the current time-step, n the number of steps (525 at most), and i the element.

Criteria	Formula	Description
ACTIVATION	$\sum_{t=1}^{n} \lvert h_{ti} \rvert$	Elements most involved in decisions.
CHANGE	$\sum_{t=1}^{n-1} \lvert h_{ti} - h_{t+1i} \rvert$	Maximal change.
STABLE	CHANGE^{-1}	Minimal change.
SIMILAR	$\lvert \frac{1}{n} \sum_{t=1}^{n-1} h_{ti} - \frac{1}{k} \sum_{t=1}^{k-1} h_{tj} \rvert$	Elements in average different during an interval of k time-steps than outside it.

Figure 4.3, using a bi-variate color scale from [124] with blue for negative values and orange for positive values. Preserving the values as they were initially produced by the model is pertinent as some memory elements (rows) can have both positive and negative values, which may not have the same signification for the model and thus cause different decisions. This will be further explored in Section 4.6.4.

By default DRLViz displays the vector as it is produced by the model, hence the order of elements has no particular semantic. The memory can be re-ordered using a drop-down menu according to comparison criteria listed in Table 4.1. With the ACTIVATION criteria, a user can observe elements that may be most involved in decisions, while with CHANGE, elements that may be the most used by the model are emphasized, with SIMILAR, a user can see elements with constant activations during selected intervals. In addition to those criteria, we provided the possibility to re-order the memory as presented in [35] *i.e.* a one-dimensional t-SNE projection of the absolute values. The re-ordering can either be applied to the whole memory or a selected interval. An order is preserved across memory filters and episodes until the user changes it.

4.4.4 Derived Metrics View

The derived metrics timeline addresses the design goals **G3** and **G4**. It represents metrics calculated from ground truth information provided by the simulator. Those metrics aim at supporting the user in finding interesting agent behaviors such as *What does a trained agent do when it has no health pack in its field of view?*. The view encodes measures of both the inputs (*e.g.* health pack is spotted) simulator (*e.g.* reward) and outputs (*e.g.* actions). Finally, DRLViz features a stacked area chart of actions probabilities encoding probabilities per action represented by

Table 4.2 – List of *derived metrics* (from top to bottom on Figure 4.4 ③)

Metric	Data Type	Values
Health of the agent	Quantitative	death **[0,100]** full
Event (item gathered)	Flag	**(1)** gathered
Item in FoV	Binary	no item **(0, 1)** item
Orientation to items	Degree	left **[-45,45]** right
Variation of orientation	Quantitative	stable **[0,30]** change
Decision ambiguity	Ratio	sure **[0,1]** unsure

colors corresponding to the ones on the action distribution graph in 4.4. With this visualization, users can observe similar sequences of decisions.

The derived metrics and stacked area chart are below the memory timeline and share the vertical thumb from the memory slider (**G3** (a)) to facilitate comparisons between the memory and the behavior of the agent (**G3** (b)) as depicted in Figure 4.5. The derived metrics can be dragged vertically by the user as an overlay of the memory to compare metrics with activation values, and identify memory elements related to them (**G4**). A constant activation of an element during the same intervals of a metric such as *Health Packs in FoV*, while being different otherwise; may hint that they are related. We provide a full list of metrics in Table 4.2. Two metrics are particularly complex and described as follows:

variation describes how the the agent's orientation (*i.e.* its FoV) changes over three consecutive time-steps. High variations indicate hesitation in directions and intervals during which the agent turns around, whereas low variations indicate an agent aligned with where it wants to go. However, in some cases (*e.g.* the agent stuck into a wall), actions may have no effect on the agent's orientation which leads the variation to remain low.

ambiguity of a decision is computed using the variance V of action probabilities. The variance describes how uniform actions probabilities are with respect to the mean. A variance $V = 0$ indicates that there is no difference between actions probabilities, and hence that the agent is uncertain of its decision. On the other way, a high variance represents strong differences in the actions probabilities and the agent's high confidence in its decision. Since the sum of all actions probabilities equals to 1, the variance is bounded within the range $[0, 1]$. To ease the readability, the variance is inverted as follows: $ambiguity = 1 - V$. An ambiguity close to 1 represents incertitude in the agent's decision.

4.5 Implementation

To explore the memory of a trained agent, one needs to create *instances* of exploration scenarios. For experts evaluations (Section 4.6) we used a trained

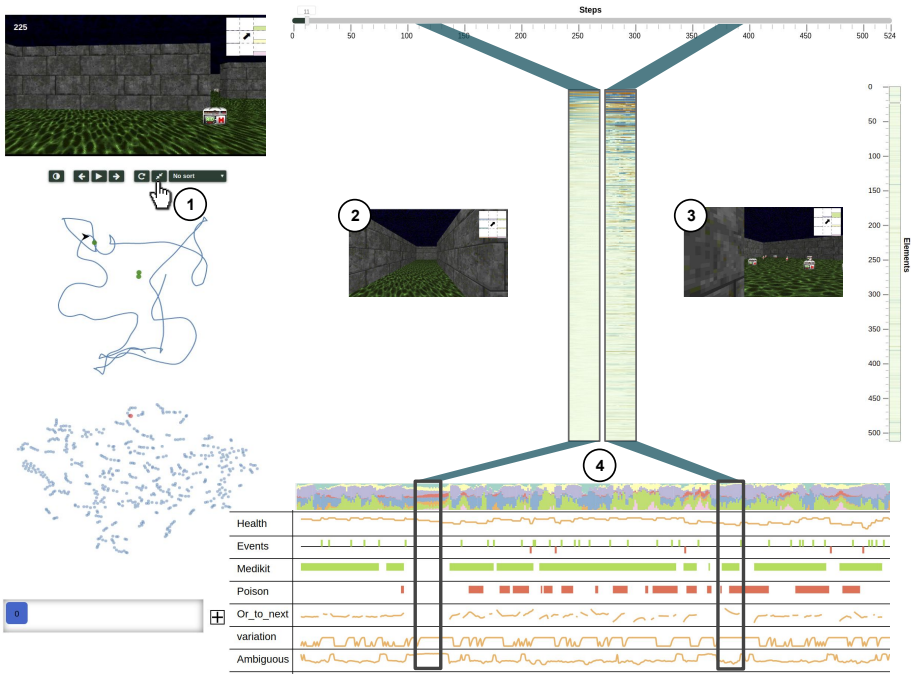


Figure 4.5 – DRLViz allows to compare selected time intervals ①. For instance to compare when agents face dead-ends ② and when they face health-packs ③. One can observe that more elements are active while the agent is facing Health Packs than while facing a dead-end. Perhaps those elements are encoding information concerning Health Packs. When facing a dead-end, both the orientation variation and decision ambiguity are high which can be interpreted as the agent hesitating on which action to choose.

agent to explore the environment 20 times with different configurations (*i.e.* positions of items, start position of the agent, its orientation). During those episodes, we collected information from the agent at each time-step such as its FoV image, action probabilities, memory vector, and information from the environment such as the items in the agent's FoV, the position of the agent, the agent's orientation and its health. The collected data is formatted as a JSON file which groups data elements per episode and then per step with an average of 30Mo per episode. Those data are generated using DRL models implemented in Pytorch [161], and formatted in Python 3. The user interface of DRLViz loads data using JavaScript and D3 [22]. The interactions between the model and the front-end are handled by a Flask Python server. The data, separated per episode is generated in a plug-in fashion *i.e.* without altering the model or the simulator.

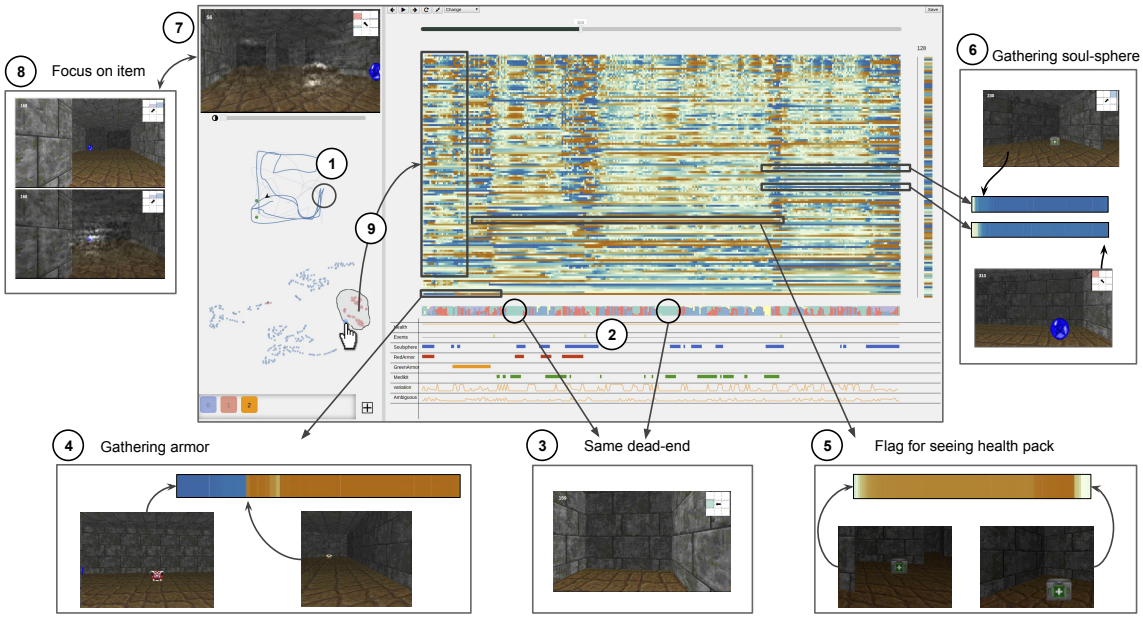


Figure 4.6 – Summary of the insights gained by the experts. Expert #1 noticed two intervals during which the agent only turned right, by using both trajectory ① and stacked area chart of actions ② views. Once he replayed those sequences, he stated that the agent came twice in the same dead-end ③. Expert #3 observed a hidden state dimension which is blue when the agent sees the red armor before the green armor, and then remained orange until when he saw the green armor ④. Expert #2 probed a dimension that is active as the agent first saw the Health Pack, and remained active until it gathered it. Expert #1 also identified two hidden state elements that change as the agent gathered the health pack and then kept their values until the end of the episode ⑥. Using saliency maps ⑦, Expert #2 observed that the agent ignored the soul-sphere until it gathered the first three items ⑧. Finally, Expert #3 identified clusters in the t-SNE projection which corresponds to the agent's objectives *e.g.* gathering the green armor ⑨.

4.6 Evaluation by Experts

We conducted a user study with three DRL experts who are experienced researchers building DRL models and match the target profile for DRLViz [87]. We report on their use of DRLViz, as well as insights they gathered. Those results provide hints that can be leveraged to formulate hypotheses that can then be studied outside DRLViz *e.g.* through statistical evidence.

4.6.1 Protocol and Navigation Problem

We recruited three DRL experts (Expert #1, Expert #2, Expert #3) from two different academic laboratories to evaluate DRLViz. They were shown a 10

minutes demonstration of DRLViz on a simple ViZDoom scenario: *health gathering supreme*. The evaluation started with DRLViz loaded with data extracted from a model developed by Expert #1, and ended after 35 minutes, during which experts could explore the displayed data. While using DRLViz experts were told to explain their thoughts and what they wanted to see. Then, experts were asked to fill a post-study questionnaire to collect their first impressions with open questions such as *"Which part of DRLViz was the least useful?"*. The evaluation ended with a discussion guided by the post-study questionnaire on their experience using DRLViz and how it can be improved. The complete evaluation lasted on average 1 hour depending on the length of the discussion. The model used was an A2C [146] with 3 convolutional layers and GRU layer with 128 dimensions as memory.

4.6.2 Feedback from Expert #1

Expert #1 is the most experienced expert for this evaluation as he designed both the model and the navigation task [18] and created animations of agents' behaviors. Expert #1 was our primary collaborator to design and build DRLViz.

Figure 4.6 shows DRLViz loaded with the *k-item* scenario. Expert #1 first selected an interval corresponding to the agent searching and gathering the last item. This interval started one step after the agent gathered the Health Pack (third item), and ended as the agent gathered the soul-sphere (last item). Expert #1, then used the CHANGE criteria to re-order the interval. While replaying it, he noticed two elements with similar activations (Figure 4.6 ⑥). Those elements remained blue during the interval, however, they were inactivated (gray) during the rest of the episode. With further investigation, Expert #1 noticed that those elements were active 4 steps before the agent gathered the Health Pack. Expert #1 described those elements as *flags i.e.* elements that encodes binary information. Expert #1's intuition was that the agent learned to complete the navigation problem by focusing on one item at a time. And only used its memory to encode information on items it already gathered, and hence which item it should currently gather. **Expert #1 concluded that the two elements may be the agent's representation that it gathered the Health Pack, and hence that it should now focus on gathering the soul-sphere.**

Then using the action probabilities temporal stacked area chart (Figure 4.6 ②), Expert #1 noticed a specific time interval during which the agent repeated the same action for almost 15 steps. Intrigued by such behavior, Expert #1 replayed this interval and noticed that the agent was within a dead-end (Figure 4.6 ③) and repeated the action *right* until it changed its orientation up to 180 degrees. Expert #1 commented that observing such interval is interesting because as the agent converges towards an optimal policy, it may have fewer chances to encounter dead-ends, and thus forget how to escape them. Expert #1 also observed a similar interval with only *right* actions in which the agent escaped the same dead-end.

Expert #1 concluded that the dead-end was not encoded in the agent's memory, and hence the agent returned to it while searching for items.

4.6.3 Feedback from Expert #2

Our second expert, Expert #2, started by re-ordering the memory using the STABLE criteria. He noticed a hidden state element, and zoomed (vertical brush) on it. This element had continuous activations starting as the agent first saw the Health Pack and remained active until the agent gathered both the red armor and the Health Pack. **Because such element is active regardless of the items the agent gathered yet, Expert #2 interpreted this element as a flag encoding if the agent has seen the health pack or not.**

Then Expert #2 focused on the saliency maps combined with episode playback. He noticed that in one episode, the agent encountered the soul-sphere (last item) before it gathered the red armor (second item). During those time-steps, the saliency maps are not activated towards the soul-sphere despite being the agent's FoV (Figure 4.6 ⑦), and the memory had no visible changes. Expert #2 intuition was that the agent did not perceive the item. In the final steps of the episode, once the agent gathered the firsts 3 items and then re-encountered the soul-sphere, the saliency maps were activated towards it (Figure 4.6 ⑧) and the memory activations changed. Expert #2 expressed that *"It is interesting because as soon as it sees it [the soul-sphere] its behavior changes"*. **Expert #2 concluded that the agent intentionally ignored the soul-sphere before it gathered previous items, and as Expert #1 mentioned, the agent learned to solve this navigation problem by focusing on one item at a time.**

4.6.4 Feedback from Expert #3

Expert #3 began his exploration with the t-SNE 2D projection as an entry point to identify clusters of hidden states. Expert #3 selected groups of states using the lasso selection (Figure 4.6 ⑨) to filter the memory timeline. The selected cluster represented consecutive steps, forming a continuous time interval. After replaying this interval, Expert #3 observed that it started at the beginning of the episode and ended when the green armor (first item) entered the agent's FoV. **Expert #3 interpreted this cluster as corresponding to an agent objective, in this case gathering the first item.**

Following up on the previously identified cluster, Expert #3 re-ordered it with the STABLE criteria. Expert #3 noticed one particular hidden state dimension that was activated in blue until the green armor entered the agent's FoV, and then was activated in orange for the rest of the episode. Expert #3 interpreted such element activation as a flag encoding if the agent has seen the green armor. However, after

observing this element activations across episodes, Expert #3 noted that it was inactivated (grayish) at the start of an episode. After re-playing this episode he observed that the agent had no armor in its FoV, as opposed to the first episode analyzed where the agent started with the red armor in its FoV. In another episode, where the agent has the green armor in its FoV since the start, the element was constantly activated in orange. **Expert #3 concluded that this element encoded if the agent saw an armor rather than just the green armor.** However, once the agent gathered the green armor, the element remained orange despite still having the red armor in the agent's FoV. **Expert #3 added that this element also encodes if the agent gathered the green armor.**

4.7 Memory Reduction

The following depicts how memory elements may be non-essential in order for DRL agents to preserve their performances and a coherent behavior.

4.7.1 Evaluation of Reductions with DRLViz

As experts noticed during interviews, the agents' memory can often be sparse (*e.g.* Figure 4.4) or hold redundancy (*e.g.* Figure 4.6 ⑥). Thus we hypothesize that some elements may either never be activated, or there might be redundant activations at the same time. In order to explore such a hypothesis, we conducted an experiment to assess that some sub-set of the memory is sufficient to solve a navigation problem, and the rest may be discarded. We used the *health gathering supreme* scenario in which the agent must collect Health Packs to survive, hence it is easier to solve than k-item. During the experiment, with a memory of 512 dimensions, we "removed" hidden state elements by multiplying them by 0 before going any further in the model's decision process. And thus, while still within the model, those nullified elements are conveying no information, and hence should have little to no impact on any decision outside of their absence of activation. We then re-run such a model with its reduced memory and generate new episodes to be evaluated. For the experiment's sake, we decided to only preserve the half of most activated elements of the memory, sampled from DRLViz, with the ACTIVATION re-order metric on the *health gathering supreme* scenario and a large memory of 512 dimensions.

The results of this experiment are presented in Table 4.3 which shows similar performances between agents with full and half memory over the same 100 episodes lasting on average around 500 steps. The most notable difference is on the agents' ability to avoid poison vials, with the reduced model taking on average slightly more of them. This might be explained by replaying episodes generated of this model, in which we can observe that the agents' actions seem a bit more

Table 4.3 – Performances of agents with different memory reduction strategies (each averaged over 100 episodes). Best result of each column is bold.

Type of reduction	Steps survived	Health Pack gathered	Poison gathered	Health
Full-memory	**503.98**	37.56	**4.28**	81.47
Half-memory	493.92	**37.88**	4.66	**81.61**

erratic, and thus, miss-steps more often. One hypothesis to draw is that the agent has at least 256 non-essential elements. Efficient selection of those elements remains a challenge, as it must account for complex temporal dependencies. This might be due to the fact that the size of the memory (*i.e.* the number of elements), is a hyper-parameter manually set by experts. And, when deciding on a size, experts often tend to opt for a large memory to ensure that the memory is not a bottleneck preventing the agent from learning complex behaviors.

With such memory reduction, *i.e.* masking undesired elements, they remain within the model and take part in the computation. However, having smaller models would be useful as they may be more interpretable, with fewer memory elements to analyze, but also require less computing power and have a lower energy consumption footprint. A solution would be to train a model with large memory, through our reduction method grasp how many memory elements would be optimal, and then re-train a new model with such a memory size.

4.7.2 MemRed, an Online Explorable

To address the challenge of optimizing the memory of DRL agents, and introduce it to non-experts, we designed MemRed, an online explorable visualization (illustrated in Figure 4.7). This explorable leverage the views of DRLViz to deliver to non-expert more accessible insights on a simplified agent's memory and how it can be reduced. In this explorable, users can only interact with a small memory for easier interpretation. Users can experiment with different strategies to reduce the memory of a simplified model with only 32 dimensions in the k-item scenario, and asses how different their corresponding trajectories are. Here, the focus is on a single episode, and hence our goal for models with reduced memory to complete it with the same number of steps as with full memory. In addition, in order to avoid being too overwhelming, the activation of memory elements is encoded in a single color hue from 0 white to 1 brownish. Traditionally those elements' activations range from -1 to 1, hence we only display their absolute value, representing how "used" is an element rather than what information it conveys.

First, as an introduction, we provide random reductions, which yields agents failing at different degrees from turning on the opposite direction of an item to

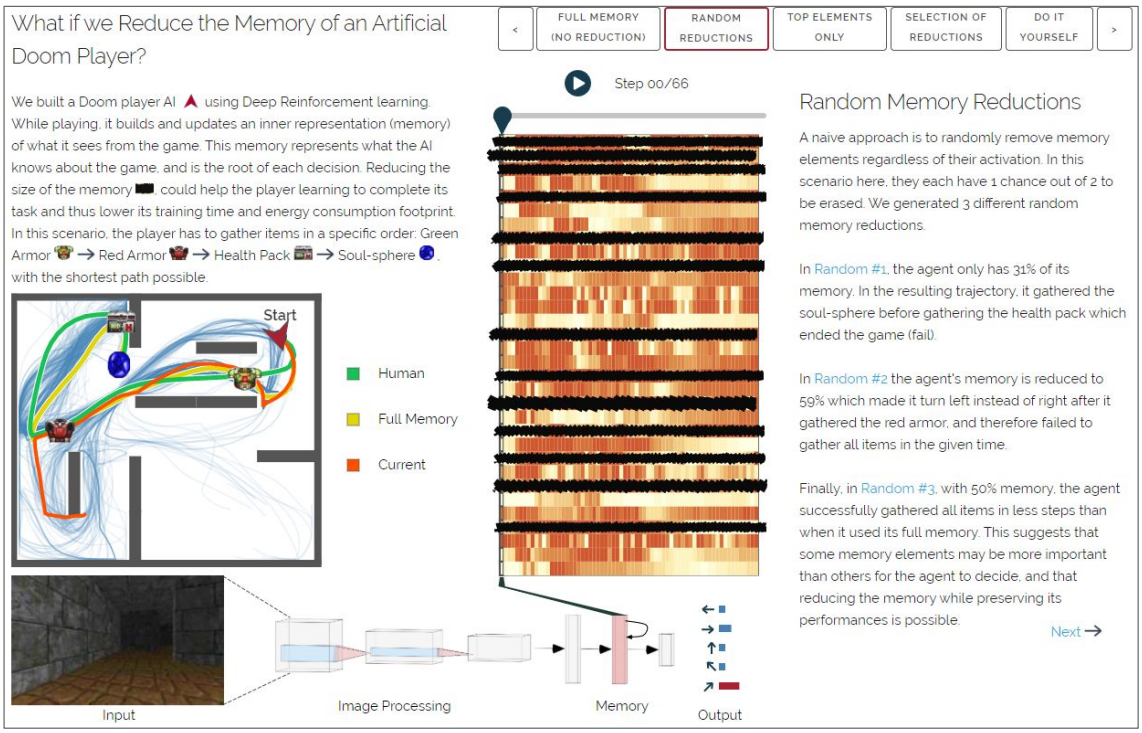


Figure 4.7 – Overview of MemRed, an online explorable in which users are invited to try different strategies to reduce the memory of a DRL agent and observe how it affects its behavior. This explorable earned the distinction of "best paper" at the VISxAI workshop 2019.

collect, to turning in circles, skipping items, or succeeding. It is observed that such degrees of failure are not correlated with the number of elements removed, but rather which one. The next strategy is, similarly to the previous section, to re-order the memory using metrics, and only preserve the top half or top-quarter. With this, the best result, with the least number of elements is achieved with to top 8 most changing elements, with the agent almost reaching the last items, but taking too many steps to be there, ending the episode. Finally, the overall best results are obtained by using those top 8 elements and manually adding 5 of them that were active at the end when there was no reduction. This yields an agent with only 40% of its memory able to successfully complete this selected episode. While this paves the way for direct filtering by elements of the memory heat-map in a future version of DRLViz, those results need to be analyzed through the lens of multiple episodes and trajectories required, in order to ensure that the agent is not lacking any knowledge to complete its task in harder situations.

4.8 Discussion

In this section, we discuss the collected feedback from experts, as well as the limits of the current version of DRLViz.

4.8.1 Summary of Experts Feedback

Experts filled a post-study questionnaire relative to DRLViz usefulness and usability. Overall DRLViz was positively received by all of them: both Expert #1 and Expert #2 stated that DRLViz is "*interesting to explain the behavior of the agent*" and "*easy to use*". However, Expert #3 stated that he felt "*overwhelmed at first, but soon got used to navigation*". All 3 experts evaluated the 2D t-SNE projection as the most useful view because it can provide insights on the agent's memory and strategies. They used this view as an entry point on at least one episode. They commented that the re-ordering was effective to observe desired hidden states dimensions. Both Expert #2 and Expert #3 used the STABLE criteria because it highlights elements that are different from the rest and should correspond to the selected interval. On the other hand, Expert #1 preferred the CHANGE re-ordering criteria because those elements have information concerning the interval. Expert #3 also noted that "*it ´s handy being able to drag it up [derived metrics timeline] and overlay it on the hidden states*" (**G3**). The experts concluded that the agent learned to solve this task sequentially, *i.e.* by focusing on gathering one item at a time. And thus that the agent only stored information corresponding to which items it has gathered rather than the positions of every seen item at any time-steps.

All three experts evaluated the memory reduction interaction that filters the memory view (zoom) as not intuitive and hard to use without losing visual contact with the hidden state dimensions they wanted to focus on. This partially validates our memory reduction goal (**G4**). On this matter, Expert #1 commented that since this agent's memory has 128 dimensions the zoom is not as useful as it could on larger memories. Expert #2 also commented on the use of the different re-ordering criteria, and that their specific functioning was hard to understand, especially the projection. Expert #3 also mentioned that he "*doesn't fully understand how the projections re-ordering methods are helpful*". To tackle those issues, Expert #3 suggested using the derived timeline to re-order the memory, *i.e.* observe hidden states activations when a feature enters the FoV. Expert #3 also commented that a horizontal zoom could be useful to focus on one particular time interval, and reduce the number of steps to observe. Expert #1 mentioned that brushing the memory while keeping activation areas as *squares*, *i.e.* both horizontally and vertically could be a better way to implement a more consistent zooming interaction.

4.8.2 Limits

Generalization and *scalability* are the main limits of the current version of DRLViz. Regarding generalization, specific calculations need to be made such as for the derived metrics timeline that is generated from the simulator *i.e.* the items in the agent's FoV. So the current metrics are tied to ViZDoom but a minor adaptation of the tool to specific environments will be needed, but requiring technical knowledge. In the next section, we will explain how the interaction techniques in DRLViz can be used beyond the tool for better timeline comparisons. Scalability is always a concern with visualization techniques. DRLViz supports long episodes and variable memory size. However, if those are larger than the screen real estate (*e.g.* beyond on average 1200 steps and more than 1000 memory dimensions) each memory cell would be smaller than one pixel, and thus difficult to investigate. To tackle such an issue, LSTMVis [193] introduced a parallel coordinate plot with each line encoding a memory element. However, with DRLViz we sought to support trend detection and thus encode the memory overview using colored stripes [66] which complies with our data density challenge and requirement to align the memory with the derived metrics below. We then rely on zoom interactions for details for both time and elements.

We plan in the future to support aggregation strategies [232] to provide a more compact representation of the timelines. Alignment by an event of interest [50] (*e.g.* gathering an item) may also provide more compact representations of metrics, and also better support comparison of behavior before and after this event. A concern raised by experts was the communication of insights gained during the exploration process. We plan to explore summarizing techniques for those insights, such as state charts [178] in which each state corresponds to a local strategy *e.g.* reach an item.

4.9 Perspectives

We present and discuss three works in progress that may be potential improvements of DRLViz, based on experts feedback, that primarily expand its exploration power and generalization.

4.9.1 Guiding Exploration with Extended Timelines

During our interviews, experts suggested to better support the memory analysis process, as the current version of DRLViz relies on visual exploration by the user, with no specific guidance. We identified two areas of improvement for a future version of DRLViz: adding more metrics, and advanced filtering. Regarding the metrics, Table 4.2 introduced derived indicators from the agent decision. Figure 4.8

Figure 4.8 – Extended version of DRLViz loaded on with death-match data. From a slit square selection ① outputs a timeline that summarizes the agent's point of view ②. And the additional metrics and operators ③.

illustrates that more metrics can be added using variations of their parameters (*e.g.* changing variability thresholds or the distance to consider an enemy is in the FoV or not) which support more questions a user may want to investigate. Such metrics are represented in a compact way, and easy to activate by scrolling down, while remaining focused on the memory. Regarding the comparison, the current version only implements *juxtaposition* and *overlay*; while *explicit encoding* [71] is a third way to compare timelines and memory. We applied this third way by adding a boolean queries builder [119] using `AND` or `OR` to filter timelines. Those boolean operators are also applicable to all views of DRLViz, such as 2D-map, t-sne, or a brush on the memory. This helps users to combine multiple views and answer questions such as *Where are the areas of memory with the agent has high health, in this part of the environment, with an enemy and an explosive barrel in FoV?* This results in intervals in which the agent is susceptible to shoot on barrels to kill enemies.

In order to summarize the input images and ease their comparison with derived metrics, we developed *slit square* interaction based on slit tears [204]. With a slit square, a user can brush a square on the inputs. Those squares are then compacted to the width shared by all time-aligned elements in DRLViz.

4.9.2 Generalization to other Scenarios and Simulations

Finally, we started investigating using DRLViz as a general-purpose tool for any trained agents with a memory and spatio-temporal information. Figure 4.8 illustrates DRLViz loaded with a different scenario where the agent shoots towards enemies on the death-match [114] with the Arnold model. In general, DRLViz can be used beyond ViZDoom (*e.g.* referred in [18]), such as Atari games [226] without

any major change. Using pixel-based representations [102] and zooming [105] would assure scalability of the timeline representations with scenarios requiring more time steps. We plan to conduct further research to identify other metrics and extend DRLViz to other simulators mentioned by our experts, such as Matterport3D [40] and Habitat-AI [180] for real-world scenarios, and competitions such as Animal-AI [47].

4.10 Conclusion

In this work, we introduced DRLViz, a visual analytics interface leveraging activation and representation interpretability building blocks. This interface empowers users to *overview*, *filter* and *select* the memory of Deep Reinforcement Learning. Analysts using DRLViz were able to explain parts of the memory of agents trained to solve navigation problems of the ViZDoom game simulator, in particular local decisions and higher-level strategies. DRLViz received positive feedback from experts familiar with DRL models, who managed to browse an agent memory and form hypotheses on it. DRLViz paves the way for tools to better support memory reductions of such models that tend to be large and mostly inactive.

CHAPTER 5

FROM SIMULATION TO REALITY

Contents

The final step, to *find our keys*, now that we are able to understand what we are looking for (Chapter 3), and able to explore unknown environments (Chapter 4), is to communicate their location (coordinates and orientation angle) from a given image (as illustrated in Figure 5.1). In addition, in the previous chapters, our robots were trained in simulation which bolsters thousandfold the number of experiments possible in the same amount of time. However, despite the increasing accuracy of realistic 3D simulators for large-scale training of robots, those robots, once deployed to the real-world tend to fail. This is known as the Sim2Real gap which emerges from gaps between simulation and real physical environment (*e.g.* approximated physics engine), as well as dynamic changes in the real world such as luminosity and lights through days, and objects displacements.

This chapter is dedicated to the analysis of the Sim2Real gap applied to the task of ego-pose estimation, *i.e.* the estimation of a robot's position using trained models. To do so, we introduce Sim2RealViz, a visual analytics tool to assist experts in understanding and reducing this gap for robot ego-pose estimation

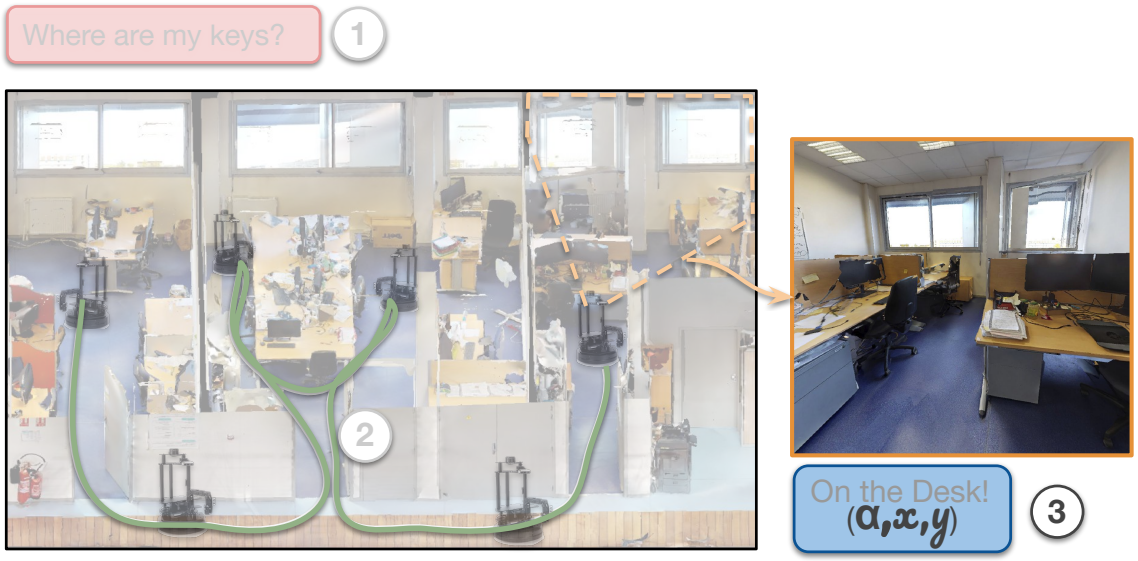


Figure 5.1 – When answering questions such as *"Where are my keys?"*, robots need to be able to communicate their position. To do so, we trained models to yield coordinates and orientation angle from any given image ③. Such a task is also used as a proxy to evaluate gaps between simulation and reality.

tasks. SIM2REALVIZ displays details of a given model and the performance of its instances in both simulation and real-world. Experts can identify environment differences that impact model predictions at a given location and explore through direct interactions with the model hypothesis to fix it. In this chapter, we detail the design of the tool, and case studies related to the exploit of the regression to the mean bias and how it can be addressed, and how models are perturbed by the vanish of landmarks such as bikes. Along with a new visual analytics system, we designed *Théo Guesser*, an online explorable in which non-expert users are introduced to the ego-pose localization task by playing against a model on it. In addition, users are progressively introduced to the challenge that arises with using real-world data and a model trained in a simulator, along with basic domain randomization and domain adaptation methods to tackle it. Both this challenge and solution are demonstrated using interpretability building blocks we designed. This explorable is available online: `https://theo-jaunet.github.io/theo-guesser`, and so is its code source: `https://github.com/Theo-Jaunet/theo-guesser`.

The works presented here relies on the following published materials:

- Théo Jaunet, Romain Vuillemot, and Christian Wolf. "Théo Guesser". *IEEE Workshop on Visualization for AI Explainability at IEEE VIS (VISxAI), 2020.*

- Théo Jaunet, Guillaume Bono, Romain Vuillemot, and Christian Wolf. "SIM2REALVIZ: Visualizing the Sim2Real Gap in Robot Ego-Pose Estimation". *NeurIPS XAI Workshop on eXplainable AI approaches for debugging and diagnosis, 2021.*

A special thanks goes to Guillaume Bono who handled the robot to collect real-world data, and helped fine-tune the simulator-end of Sim2RealViz.

5.1 Introduction

Visual navigation is at the core of most autonomous robotic applications such as self-driving cars or service robotics. One of the main challenges for the robot is to efficiently explore the environment, to robustly identify navigational space, and eventually be able to find the shortest paths in complex environments with obstacles. The Robotics and Deep Learning communities have introduced models trained with Reinforcement Learning (RL), Inverse RL, or Imitation Learning, targeting complex scenarios requiring visual reasoning beyond waypoint navigation and novel ways to interact with robots, *e.g.* combining vision, robotics, and natural language processing through queries like "*Where are my keys?*". Current learning algorithms are not sampled efficiently enough, this kind of capability requires an extremely large amount of data. In the case of RL, this is in the hundreds of millions or in the billions of interactions — this simply cannot be addressed in a reasonable amount of time using a physical robot in a real environment, which also may damage itself in the process.

To tackle this issue, the field heavily relies on simulation, where training can proceed significantly faster than in physical (wall clock) time on fast modern hardware, easily distributing multiple simulated environments over a large number of cores and machines. However, neural networks trained in simulated environments often perform poorly when deployed on real-world robots and environments, mainly due to the"*Sim2Real gap*", — *i.e.* the lack of accuracy and fidelity in simulating real-world environment conditions such as, among others, image acquisition conditions, sensors noise, but also furniture changes and other moving objects. The exact nature of the gap is often difficult to pinpoint. It is well known that adversarial examples, were only a few pixel shifts occur, considered as small artifacts by humans, or which might even be undetectable by humans, can directly alter the decisions of trained models [74, 148, 127].

The SIM2REAL GAP is currently addressed by various methods, including domain randomization, where the physical reality is considered to be a single parametrization of a large variety of simulations [212, 136], and Domain Adaptation, *i.e.* explicitly adapting a model trained in simulation to the real-world [215, 38]. However, identifying the sources of the sim2real gap would help experts in designing and optimizing transfer methods by directly targeting simulators and design choices of the agents themselves. To this end, we propose SIM2REALVIZ, a

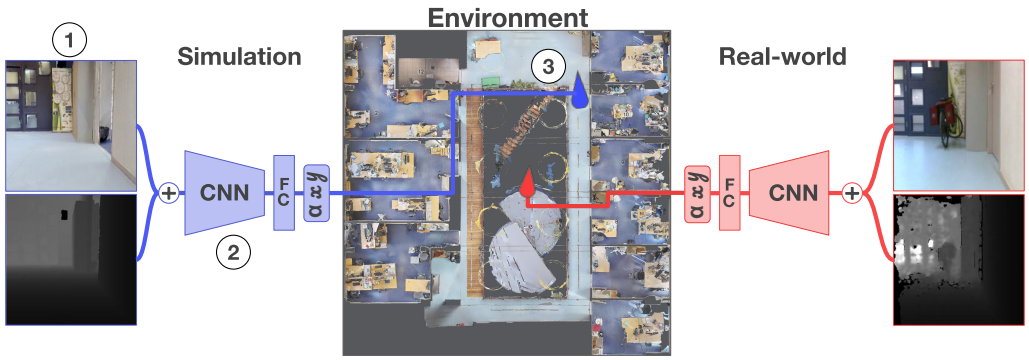


Figure 5.2 – In the studied robot ego-localization task, an RGB-D image ①, is given to a trained model ②, which uses it to regress the location (x, y), and orientation angle (α) in the environment from which this image was taken from ③. As illustrated above, images taken from the same coordinates in simulation and real-world ① may lead to different predictions due to differences, such as here, among others, the additional presence of a bike in the scene. We are interested in reducing the gap between the SIM and REAL predictions.

visual analytics interface aiming to understand the gap between a simulator and a real-world environment. We claim that this tool is helpful to gather insights on the studied agents' behavior by comparing decisions made in simulation and in the real-world physical environment. Sim2RealViz exposes different trajectories, and their divergences, in which a user can dive deeply for further analysis. In addition to behavior analysis, it provides features designed to explore and study the models' inner representations, and thus grasp differences between the simulated environment and the real-world as perceived by agents. Experts can rely on multiple-coordinated views, which can be used to compare model performances estimated with different metrics such as a distance, orientation, or a UMAP [135] projection of latent representations. In addition, experts dispose of three different approaches to highlight the estimated sim2real gap overlaid over either 3D projective inputs or over a bird's eye view ("Geo-map") of the environment.

Sim2RealViz targets domain experts, referred to as *model builders and trainers* [193, 87]. The goal is assistance during real-world deployment, pinpointing the root causes of decisions. Once a model is trained in simulation, those experts are often required to adapt it to real-world conditions through transfer learning and similar procedures. In Section 5.6, we report on insights gained through experiments using Sim2RealViz, on how a selection of pre-trained neural models exploits specific sensor data, hints on their internal reasoning, and sensibility to sim2real gaps.

5.2 Context and problem definition

We study trained models for Ego-Localization of mobile robots in navigation scenarios, which regress the coordinates (x, y) and camera angle α from observed RGB and depth images. Physical robots take these inputs from a depth-cam, whereas in the simulation they are rendered using computer graphics software from a 3D scan of the environment.

Figure 5.2 provides a concrete example, where two images are taken at the same spatial coordinates ①, one from simulation and the other from a physical robot. As our goal is to estimate the sim2real gap, we do not focus on generalization to unseen environments. Instead, our simulated environment corresponds to a 3D scanned version of the same physical environment in which the robot navigates, which allows precise estimation of the difference in localization performance, as the gap leads to differences in predicted positions. The full extent of the gap, and how it may affect models is hard to understand by humans, which makes it difficult to take design choices and optimize decisions for sim2real transfer.

Simulation — we use the Habitat [180] simulator and load a high fidelity 3D scan of a modern office building [1] created with the Matterport3D software [40] from individual 360-degree camera images taken at multiple viewpoints. The environment is of size 22×22 meters and can potentially contain differences to the physical place due to estimation errors of geometry, texture, lighting, alignment of the individual views, as well as changes done after the acquisition, such as moved furniture, or opened/closed doors. The Habitat simulator handles environment rendering and agent physics, starting with its shape and size (*e.g.* a cylindrical with diameter 0.2m and height 1.5m), its action space (*e.g.* turn left, right or move forward), and sensors — a simulated *e.g.* RGB-D camera. Depending on the hardware, the simulator can produce up to 10.000 frames per second, allowing to train agents on billions of interactions in a matter of days.

Real-world — As illustrated in Figure 5.3, our physical robot is a "*Locobot*" [150] featuring a RGB-D camera and an additional LIDAR sensor which we installed. We use the LIDAR and the *ROS NavStack* to collect ground truth information on the robot's position (x^*, y^*) and angle α^*, used as a reference to evaluate ego-pose localization performances on the real-world. To increase precision, we do not build the map online with SLAM, but instead, export a global map from the 3D scan described above and align this map with the LIDAR scan using the *ROS NavStack*.

Ego-pose estimation: the trained agent — Traditionally, ego-pose estimation

1. The second floor of the CITI laboratory at INSA-Lyon.

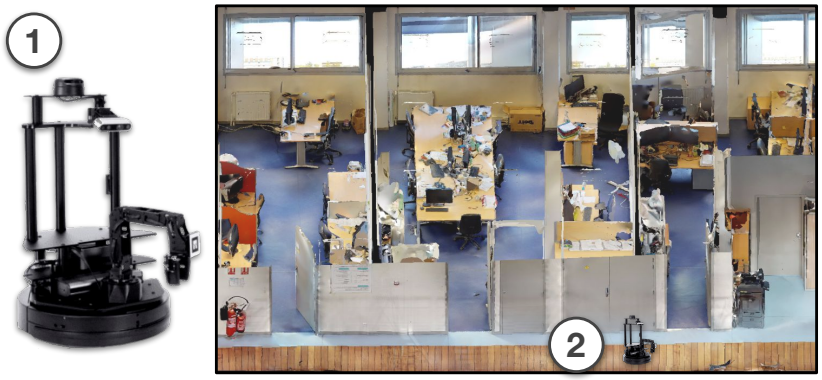


Figure 5.3 – The capture of real-world images are carried by an embedded RGB-D camera on a "*Locobot*" [150] ①. On the right ② such a robot is displayed with respect to proportions in its environment.

of robots is performed from various inputs such as LIDAR, odometry, or visual input. Localization from RGB data is classically performed from keypoint-based approaches and matching/alignment [27]. More recently, this task has been addressed using end-to-end training of deep networks.

We opted for the latter, and, inspired by poseNet [104], trained a deep convolutional network to take a stacked RGB-D image X_i of shape $(256 \times 256 \times 4)$ and directly output a vector $Y_i = (x_i, y_i, \alpha_i)$ of three values: the coordinates x_i, y_i and the orientation angle α_i. The model is trained on 60.000 images sampled from the simulator with varying positions and orientations while assuring a minimal distance of 0.3 meters between data points. We optimize the following loss function between predictions $Y_i = (x_i, y_i, \alpha_i)$ and Ground-truth (GT) $Y_i^* = (x_i^*, y_i^*, \alpha_i^*)$ over training samples i:

$$\mathcal{L} = (1-\gamma) \sum_i \left|\left| \begin{bmatrix} x_i \\ y_i \end{bmatrix} - \begin{bmatrix} x_i^* \\ y_i^* \end{bmatrix} \right|\right|_2^2 + \gamma \sum_i |\alpha_i - \alpha_i^*| \text{mod } 2\pi \tag{5.1}$$

where $||.||_2$ is the L_2 norm and γ is a weighting hyper-parameter set to 0.3 in our experiments.

5.3 Related work

Closing the sim2real gap and transfer learning — Addressing the SIM2REAL GAP relies on methods for knowledge transfer, which usually combine a large number of samples from simulation and/or interactions obtained with a simulator

simulation with a significantly smaller number of samples collected from the real-world. Although machine learning is a primary way of addressing the transfer, it remains important to assess and analyze the main sources of discrepancies between simulation and real environments. A common strategy is to introduce noise to the agent state based on statistics collected in the real-world [180]. Additionally, tweaking the collision detection algorithm to prevent wall sliding has been shown to improve the performance in the real-world of navigation policies trained in simulation [95], which tend to exploit inaccurate physics simulation. Another approach is to uniformly alter simulation parameters through domain randomization, *e.g.* modifying lighting and object textures, to encourage models to learn invariant features during training [212, 213]. This line of work highly benefits from domain expert knowledge on the targeted environment, which can provide randomizations closer to reality [165, 92].

A different family of methods addresses the SIM2REAL GAP through Domain Adaption, which focuses on modifying trained models' and their features learned from simulation to match those needed for high performance in real environments. This has been explored by different statistical methods from the machine learning toolbox, including discriminative adversarial losses [215, 38]. Feature-wise adaptation has also been addressed by extensive use of loss [65, 214], and through fine-tuning [177]. Instead of creating invariant features, other approaches perform Domain Adaption at pixel level [26, 25]. Despite great results, Domain Adaptation suffers from the need for real-world data, which is often hard to come by. We argue that there is a need for the assistance of domain experts and model builders to understand the main sources of sim2real gaps, which can then be leveraged for targeted adapted domain transfer, e.g. through specific types of representations or custom losses.

Interpretable robotics — This classical line of work remains an under-explored challenge when applied to regression tasks such as robot ego-localization, our targeted application, in which attributions may be harder to interpret. To our knowledge, visualization of transfer learning, and especially targeting sim2real is an under-explored area, in particular for navigation tasks, where experiments with real physical robots are harder to perform compared to, say, grasping problems. In [201], the evolution of features is explored before and after transfer through pair-wise alignment. Systems such as [131] address transfer gaps through multi-coordinated views and in-depth analysis for models weights and features w.r.t. domains. Finally, common visualizations consist of heatmaps designed to illustrate results from papers in Machine Learning (ML) communities such as [215, 244]. Despite providing insights on how models adapt to different domains, and in contrast to our work, those methods have not been designed to directly target what parts of the environment, or which sensor settings may produce sim2real gaps, which we consider as relevant information for domain experts.

5.4 Design Motivation

Prior to the design of Sim2RealViz, we conducted interviews with 3 experts in Robotics and discussed their workflow, with the objective being to address and identify sim2real gaps. Two of those experts, co-authors of this work, then took part in the design of Sim2RealViz. The workflow of interrogated experts consisted in identifying failure cases through statistics or video replaying a robot's trajectory, and then manually finding equivalent images in simulation to compare to.

5.4.1 Tasks analysis

From those discussions with experts, and literature review introduced in Section 5.3, we distill the process of analyzing Sim2Real gap transfer in the following three families of tasks.

T1. Fine-grained assessment of model performance gap between SIM and REAL — What is the best performing sim2real transfer method (*e.g.* fine-tuning, domain randomization etc.)? What are the optimal hyper-parameters? Answering those questions requires experts to study a large number of predictions in SIM and REAL from a large number of observed images and evaluate performance distribution over different environment conditions and factors of variation.

T2. Identification of the source of the performance gap — what are the factors of variation in the environment, agent, or trained model, which are responsible for the performance gap? This is inherently difficult, as the sources may be due to the global environment (differences in e.g., lightening, 3D scanning performance), the agent (*e.g.* differences in camera focal length or height) or changes due to the time span between scanning and physical deployment (*e.g.* furniture changes). In addition, some gaps may also be beyond human comprehension such as adversarial noise. For a human designer, it may not immediately be clear, which differences will have the largest impact on prediction performance.

T3. Closing the sim2real gaps — successful knowledge transfer requires the simulator to be as close as possible to the real-world scenario with respect to the factors of variation identified in **T2.** The goal is to close the loop and increase prediction performance using the insights gained from using Sim2RealViz.

5.4.2 Design goals

Based on the identified kind of tasks (**T.**) introduced in Section 5.4.1, literature review, and interviews with domain experts, we distill the following key design goals (**G.**) for SIM2REALVIZ.

G1. Summary of models' performances w.r.t. their domain. Following challenge (**T1.**), in most cases the entry-point for any analysis of sim2real transfer starts with experts trying the assess the performances of their models. Therefore, to give users an overview of their model performances, we aim to create a visual summary of each model, along with their performance distribution, while ensuring a layout that eases comparisons between sim and real overall performances, but also inter-domain comparisons of models to answer questions such as:*"Which is the most successful transfer knowledge method on real-world images?"*

G2. Gather instances based on their performances. To ease the analysis of sim2real performance gaps, users need to quickly identify instances sharing the same performance gaps (**T2.**). For example, every image that contains a bike in simulation, which is absent in the real-world, hence causing trouble in predictions. That way, the analysis of sim2real gaps in any image of the set of instances, should be useful for the rest of the set. In order to bolster its efficiency, This selection of instances should rely on different metrics such as, among others, the sim or real performance, the position of ground-truth, the position of sim/real prediction, and instances invariant towards transfer knowledge approaches tried.

G3. Per instance Sim and Real comparison. To identify the source of performance gap (**T2.**) on a designated instance, users need to inspect human-understandable data —*i.e.* both RGB and Depth images from simulation and real-world used as input. Thus, those images need to be as close as possible in order to quickly overview the most obvious changes such as furniture. Inputs also need to be inspected as perceived by the model, as some sim2real gaps that may be salient to humans, can be irrelevant for it. Finally, the performance of models is usually computed by the distance of their prediction to the corresponding ground-truth. However, if the distance between sim and real is close to 0, and the distance to ground-truth high, perhaps the corresponding instance is not influenced by any sim2real gap.

G4. Adjust Sim and Real images. In order to ease the sim2real knowledge transfer, users need for the simulation and real-world to be as close as possible (**T3.**). However, aside from the camera settings such as height and resolution, other shifts need to be tackled by manually adjusting images parameters such as brightness or dynamic range from depth images. The task of finding the correct configuration which makes real-world predictions more accurate can be

fastidious. Thus we aim at helping users with ways to manipulate images settings and simulation settings of sensors to help experts quickly address some sim2real gaps issued from inaccurate configurations.

5.5 Sim2RealViz: A visual analytics tool to explore the sim2real gap

We introduce Sim2RealViz, an interactive visual analytics tool designed to assist domain experts in conducting in-depth analyses of the performance gaps between simulation and real environments of models whose primary task is ego-localization. The tool is implemented in JavaScript and the D3 [22] library to run in modern browsers and directly interacts with models implemented in Pytorch [162]. The source code is available as an open-source project at: https://github.com/Theo-Jaunet/sim2realViz .

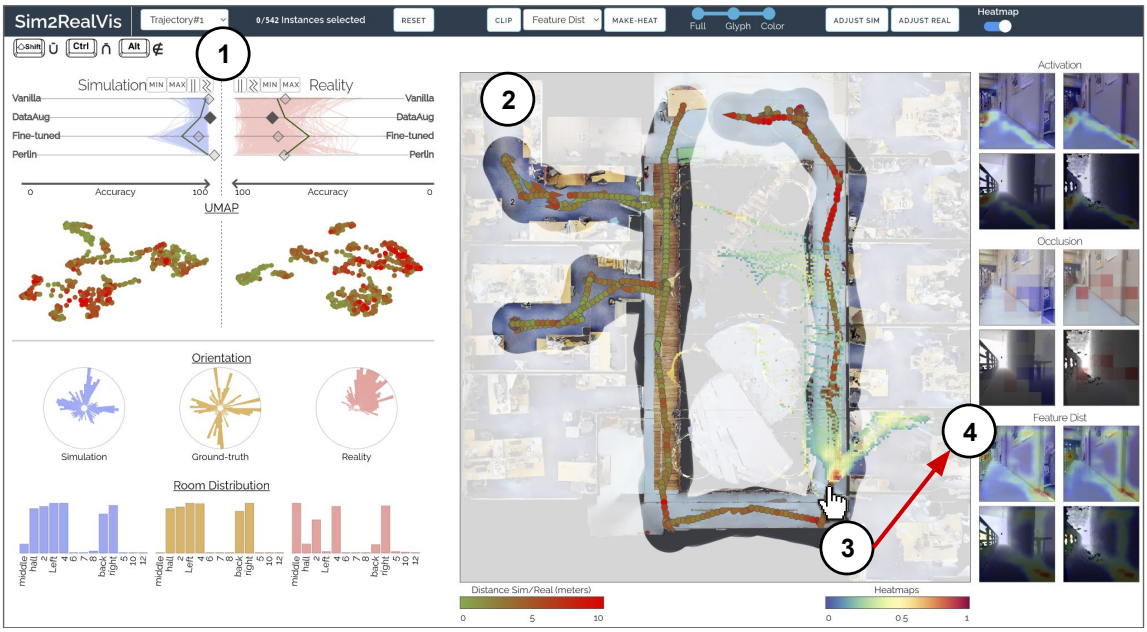


Figure 5.4 – Using Sim2RealViz, the Sim2Real gap of a Data Augmentation model can be compared against other models (*e.g.* Vanilla or Fine-tuned) and displayed on the real-world environment map along with its performance metrics. In particular, Sim2RealViz shows ① this model is particularly effective in simulation but we identified errors in the environment, such as the model failing to regress its position because of a closed-door that was opened in training. Such an error can then be selected by instance on the map ② to identify key features extracted by the model either as superimposed on the heat-map ③ or as a first-person view ④.

5.5.1 Design rationale

Our design is centered around the comparison of simulation instances and real-world ones. As we deal with complex objects, and because sim2real gaps can emerge from various sources, we implemented several views with different comparison strategies [71]. As illustrated in Figure 5.4, SIM2REALVIZ follows the *overview+detail* interface scheme [44] with a range from the most global views (left), to the most specific ones (right). To ease the comparison, simulation and real-world data are displayed next to each other within each view, with, if possible, simulation on the left side and real-world on the right side. The objective of the *Statistics view* (Figure 5.4 ①) is to help in quickly identifying the performance of a model and to grasp global behavior with simple visualizations. The *Geo-map* (Figure 5.4 ②), is key in providing context on the instance predictions, and for users to grasp what factors of variation may cause sim2real gaps. Finally, the *Instance view* (Figure 5.4 ③), displays how models may perceive sim2real gaps under different scopes. To encode the main information related to the gap we used three colors corresponding to either SIM, REAL, or GT, across the visualizations. We also used color to encode the distance between two sets of coordinates or the intensity of the models' attention towards parts of input images using a continuous turbo [139] color scale, commonly used by experts, to emphasize the most critical instances, *i.e.* those with high values.

5.5.2 Main-stream workflow

We now provide a typical workflow of use of SIM2REALVIZ:

1. Models are pre-loaded and their overall performances on both SIM and REAL are displayed on the top left of SIM2REALVIZ (Figure 5.4 ①).
2. After model selection, and following **G1.**, users can start a fine-grained performance analysis of SIM and REAL models by observing global statistics views such as a UMAP [135] projection of embeddings in which each dot is an instance, and its color encodes how far it is to its counterpart (sim or real). Followed by a radial bar chart of predicted or ground-truth orientation, and finally, a distribution of positions in which each room is a bar, and their height corresponds to how many predictions there are. In any of those views, users can select a set of instances to be inspected, for example, a cluster in UMAP **G2.**).
3. Any of those selection updates a geo-map (Figure 5.4 ②), i.e. a "geometric" bird's eye view, in which users can inspect the predictions in a finer scale. Users can adapt the geo-map to either *color*-mode which only displays ground-truth positions with their colors indicating how far sim and real predictions are, or *full*-mode which displays sim predictions, ground-

truth positions, and real predictions. Instances can be selected for further inspection by mouse-hovering them.

4. An instance selection updates the instance view (Figure 5.4 ③) and displays heatmaps, which highlights the portions of images on which the model most focuses on, or which it perceives as different (**G3.**). Such a heatmap is also back-projected over the geo-map to highlight portions of the environment, which most likely carry sim2real gaps (Figure 5.4 ④).

The views in SIM2REALVIZ are multi-coordinated, i.e. any of them, including the geo-map, can be used as an entry point to formulate complex queries such as "*what are the instances which perform poorly in simulation, but good in real-world while being in a selected region of the environment?*". Concretely, those combinations of selection can be done using sets operations (*{union, intersection, and complementary}*), which can be selected through interactions with the corresponding views. This is further emphasized by the fact that the performance differences between simulation and real-world are also color-encoded on the geo-map.

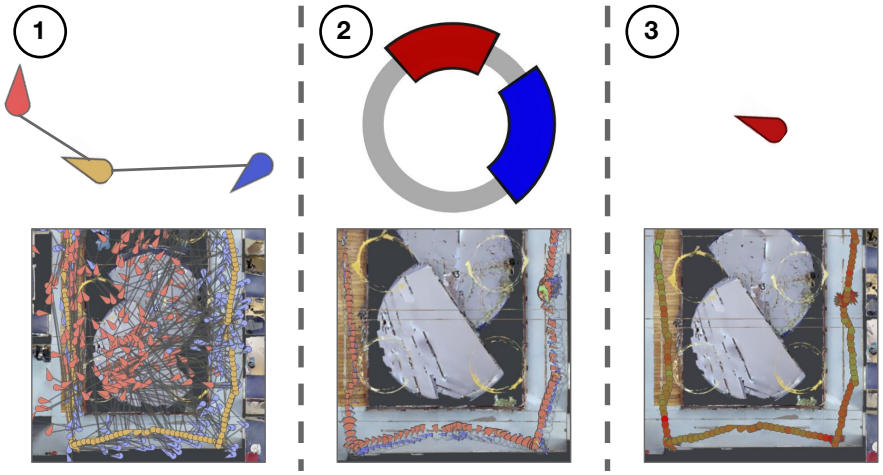


Figure 5.5 – To tackle over-plotting in *geo-map* while preserving the insights users can grasp with this view, SIM2REALVIZ provides three way to encode SIM / REAL predictions along with their ground-truth. From ① which displays a pin per domain and link them to their ground truth, to ② a glyph in which portion size encodes the distance to their color encoded domain, while their position encodes their direction. Or ③, only display ground-truth pins with their color encoding the distance between simulation and real-world.

5.5.3 Geo-Map and Encoding of Positions

Predictions need to be contextualized within their environment in order to have a grasp on which portions of the environment may be susceptible to convey

sim2real gaps. This is emphasized by the fact that not every miss-prediction, despite a similar distance, has the same significance. For example, we should distinguish between predictions within the same room, and predictions in different rooms or unreachable locations. As depicted in Figure 5.5 ①, the ground-truth coordinates (x, y) and view-point orientation angle (α), along with their sim and real predictions are represented as pins on a bird's eye top-down view of the environment, with their color encoding the domain they belong to. However, when several instances are displayed over the geo-map, pins of a single instance need to be connected to avoid any confusion. With such an amount of marks displayed per instance, this view can easily suffer from occlusion and over-plotting (Figure 5.5 ①).

To address the challenge of the triplet of coordinates visualizations, SIM2REALVIZ provides three levels of aggregation of triplets that users can select at any time. To limit occlusion, while preserving the information displayed, instances can be encoded as glyphs (Figure 5.5 ②) positioned at ground-truth locations, and with two wedges to represent SIM and REAL information. The radius of each wedge encodes the distance of its color-encoded prediction to ground-truth, while the centroid of the wedge points towards the position of this prediction. Users can then easily compare sim and real predictions for selected instances. If those wedges overlap, their intersection is color encoded in green, indicating that both simulation and real-world predictions may be similar. Finally, as illustrated in Figure 5.5 ③, by default instances are displayed on the geo-map using only ground-truth pins, with their color encoding the distance between simulation and real-world per instance. To ease the readability, the background map of the environment can be clipped to only highlight the surroundings of selected instances.

5.5.4 Heatmaps

To facilitate the inspection of the sim2real gap through image comparisons, SIM2REALVIZ provides heatmaps superimposed over images, from a selected instance, to draw user attention towards key portions of inputs extracted by the trained model (Figure 5.4 ③). Feature-wise visualizations are essential, as visual differences between simulated and real-world images perceived by humans may not correspond to differences in features with a high impact on model decisions. Figure 5.4 ③ illustrates the result of three approaches to generate those heatmaps, as follows (from top to bottom):

Regression activation mapping — Inspired by grad-CAM [182] and RAM [228], we design heatmaps to highlight regions in the input, which have a high impact on model prediction. For each forward-pass of a model, we collect feature maps from the last Convolutional Neural Network (CNN) layer and multiply them by

the weights of the last Fully Connected (FC) layer, obtaining an overlay of the size of the feature map, which is then re-scaled to fit the input image and normalized to fit a turbo color scale (Section 5.5.1). The similarity of activation maps between two similar SIM and REAL images suggests a similarity of the two input images from the model's reasoning perspective.

Sim/Real occlusion — Occlusion sensitivity [239] is a common method to visualize how neural networks in computer vision rely on some portions of their input images. It consists in applying gray patches over an image, forwarding it to a model, and observing its impact on the model's prediction. By sampling a set of patches, we can then overlay the input with this information, blue color indicating that the occluded prediction is closer to the original ground truth, and red otherwise.

In our case, the intuition and solution are slightly different from the standard case. We are interested in areas of the input image, where the model performance is improved when the real-world observation is replaced by simulated information, indicating a strong sim2real gap. We, therefore, occlude input REAL images with RGB or Depth patches from the corresponding simulated image. Thus, a further advantage of this approach is the possibility to discriminate between gaps in RGB or Depth input. The size of the patches is governed by a Heisenberg-like uncertainty trade-off between localization performance and measurement power. After experimenting with patch sizes ranging from 2×2 pixels to 128×128, we concluded that patches of 40×40 pixels, *i.e.* a total of 6×6 patches per image, are the more suitable to analyze images on our computer as we estimated that response time for such an interaction should be less than one second. This is due to the fact that this is displayed on mouse-over, hence multiple instances can quickly be probed by a user, and a longer interaction time dampens the usability and user experience of SIM2REALVIZ.

Feature map distance — Another approach implemented in SIM2REALVIZ is to gather the feature map of the last CNN layer during a forward pass on both the simulation and its corresponding real-world image, and then compute a distance between them. The result is a matrix with the size of the feature map which is then overlaid like the activation mapping. After some iterations, we opted for the product of the cosine distance which favors small changes, and L1 which is more inclined to produce small spots. Such a product offers a trade-off between highlighting every change and facing over-plotting while focusing only on one specific spot with the risk of losing relevant changes.

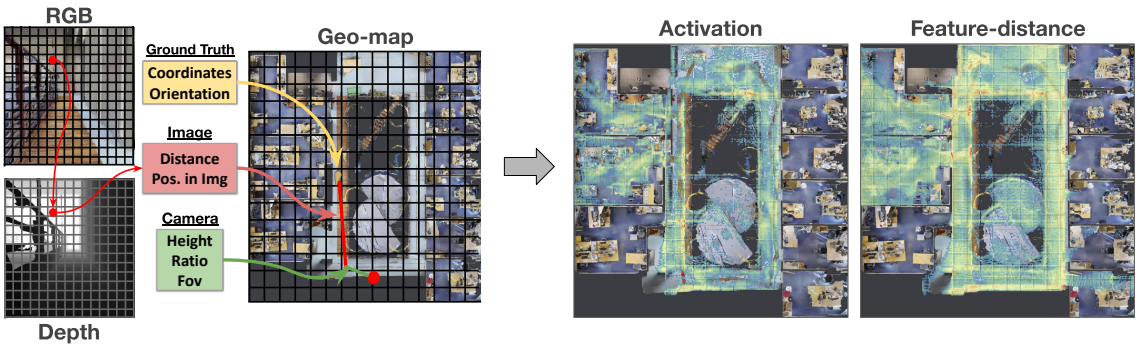


Figure 5.6 – Conversion from pixels on a first-person point of view image to coordinates on a bird's eye geo-map (left) using inverse projection given a calibrated camera. Such a process, used in SIM2REALVIZ to display global heatmaps (right) on the geo-map, relies on ground-truth, image, and camera information. To optimize their computation, geo-maps are discretized into squares larger than a pixel, as a trade-off between the accuracy of projections, and the user to the feedback.

5.5.5 Contextualization on the global geo-map

As illustrated in Figure 5.4 ④ and in Figure 5.6, information from the individual first-person 3D projective input images, including heatmaps, can be projected into the global bird's eye view, and thus overlaid over the geo-map. This is possible thanks to ground truth information, *i.e.* coordinates, and orientation of the instance, combined with information of the calibrated onboard cameras (simulated and real) themselves such as its field-of-view, position on the robot, resolution, and the range of the depth sensor. To do so, the environment is discretized in 264×264 blocks initially filled with zeroes, and images are downsampled to 128×128. Each cell is converted into (x, y) coordinates, and its average value from a heatmap is summed with the closest environment block to (x, y) coordinates. Finally, the values of environment blocks are normalized to fit the turbo color scale and then displayed as an overlay on the *geo-map*. This process can also be applied to the complete dataset available at once to provide an overview of sim2real gaps of the environment as perceived by a model. Figure 5.6 shows the conversion of heatmaps from the complete real-world dataset to a geo-map overlay using different aggregation strategies. This overlay can be displayed using the button *make-heat* from the geo-map view (Figure 5.4 ②).

5.5.6 Exploration of input configurations

Following the design goal **G4.**, to check the impact of sim2real gaps due to global imaging parameters, SIM2REALVIZ provides ways to adjust real-world images through filters such as brightness, contrast, temperature, and dynamic range

of depth. As illustrated in Figure 5.7, those filters can be generated with sliders on the right of instance view (Figure 5.4 ④). Any adjustment on a selected instance updates the corresponding prediction in real-time. Once a set of adjustments is validated by the user, it can be saved, applied to the whole real-world dataset, and treated as a new model in the *model gaps overview* ① for further analysis.

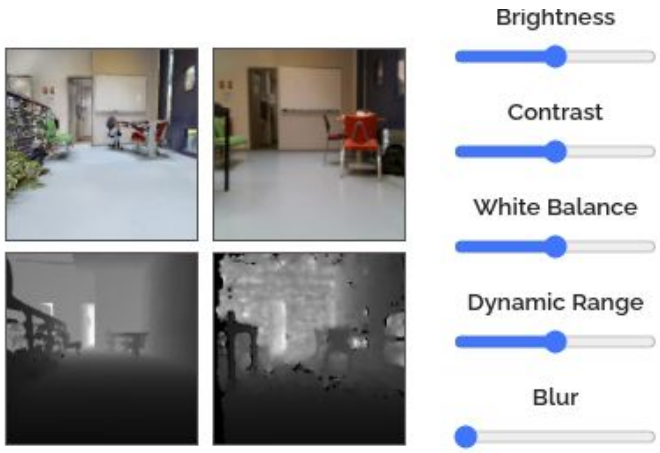


Figure 5.7 – By clicking on the *adjust* button (on the top-right of Figure 5.4), users can display sliders on the right of instance view Figure 5.4 ④) that can be used to fine-tuning real-world images with filters and observe how it affect models' prediction.

The configuration of inputs can also be used on images from simulation, to analyze the performance of the model under specific Domain Randomization configurations, or simulation settings such as, for example, the height of the camera, or its Field of View (FoV). Of course, for the simulation to have an impact on how models perceive real-world images, they need to be retrained outside of SIM2REALVIZ. To do so, one must first generate a new dataset with the modified settings, in our case 60*k* images, which can take around half an hour as we also need to enforce diversity of sample images (coordinates and orientation). Then, we train from scratch our model takes around 3 to 4 hours on our single NVIDIA Quadro P4000 GPU. Despite such a delay, adjusting simulation images in SIM2REALVIZ can be useful to help manually extract parameters of the real-world camera, and hence assist in the configuration of the simulator. Producing images, by configuring the simulator with direct feedback, should reduce the workload usually required to configure simulators and real-world robots.

5.6 Case studies

We report on illustrative case studies we conducted to demonstrate how SIM2REALVIZ can be used to provide insights on how different neural mod-

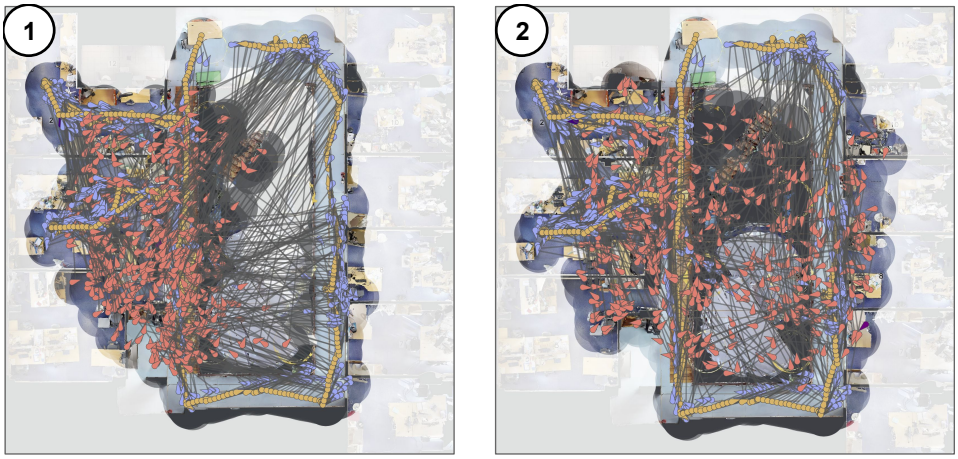


Figure 5.8 – By using the *full* encoding, we can observe that most real-world predictions are located in the half left of the environment ①. Hence, instances sampled from the half-right of the environment provide the worst predictions. However, when we slightly increase the brightness of each real-world image, we can observe that instances are more evenly distributed over the environment ②.

els may be influenced by sim2real gaps. During these experiments, SIM2REALVIZ is loaded with the following methods for sim2real transfer: *vanilla* (*i.e.* no transfer, deployment as-is), *dataAug* (*i.e.* with Domain Randomization over brightness, contrast, dynamic range, hue), *fine-tuning* on real-world images, and *perlin*, a hand-crafted noise on depth images designed to be similar to real-world noise. We use visual data extracted from two different trajectories of the physical Locobot agent in the real environment performed with several months between them and at different times of the day, which provides a diverse range of sim2real gaps and optimizes generalization. Those models and data are available in our GitHub repository at: `https://github.com/Theo-Jaunet/sim2realViz` .

Insights on sim2real gaps grasped using SIM2REALVIZ can be leveraged from two different perspectives echoing current sim2real transfer approaches. First, similar to Domain Adaptation, we can provide global modifications of the REAL images (*e.g.* brightness), which can be placed as filters and used in SIM2REALVIZ. Second, related to Domain Randomization, by modifying the simulator settings (*e.g.* adding or removing objects in the environment), and then by training a new model on it. In what follows, we describe several types of sim2real gaps, which have been identified and partially addressed in our experiments.

Unveiling biases in predictions — Once loaded, users can observe how models perform on simulated and real-world data provided by different models trained and transferred with different methods, as shown in Figure 5.4 ①. We report that best real-world performances are reached using *dataAug*, with an average of 84% accuracy, rather than *Fine-tuning*, with an average accuracy of 80%. This

performance is evaluated on traj#1, whereas traj#2 had been used for fine-tuning on real-world data, ensuring generalization over experimental conditions in the environment. In what follows we will focus on the *dataAug* model, which a user can further analyze by clicking on its corresponding diamond (Figure 5.4 ①). This updates every other view to display data extracted from this model. To assess what the worst real-world prediction is, users can use the *min* filter on the top of Figure 5.4 ①. This removes from each view of Sim2RealViz instances whose real-world performances are not among the bottom 15%. In our case, the remaining data displayed corresponds to instances sampled from the right corridor regardless of the model used. We conjecture, that corridors are among the most difficult instances as they are quite large and lack discriminative landmarks. However, in opposition, by using the *max* filter, we can also observe that the left-side corridor is among the most successful predictions. By hovering those corridor instances with a successful transfer, we can inspect activation heatmaps and observe that model attention is driven towards the limit between a wooden path (unique to the left corridor) and a wall. Thus, the model seems to have learned to discriminate between corridors, which suggests that the confusion between them may be due to other reasons. By switching the encoding on the geo-map to *full* using the slider on the middle top of Sim2RealViz, the geo-map updates to display SIM, REAL, and GT positions (Figure 5.8 ①). With this, we can observe that the *vanilla* model, incorrectly predicts real-world positions from the half-right of the environment in the half-left. Since those instances are correctly predicted in simulation, this indicates a very strong bias from most of the half-right real-world instances. A similar phenomenon is also observed for the *dataAug* model with instances on the right corridor creating predictions pointing to the middle of the environment, which is also an unreachable area.

Closing the loop with filters —We verify the hypothesis of regression to the mean, which is often an "easy" short-cut solution for a model in absence of regularities in data, or when regularities are not learned. The following focuses on the *vanilla* model, as it is the one with the most real-world predictions on the half left of the environment. We perform global adjustments of the imaging parameters of the real-world images as described in Section 5.5.6, in particular setting both RGB and depth input to zero (*i.e.* uniform black images), leading to the same constant predictions in the middle of the environment, corroborating the hypothesis.

While adjusting the brightness filter, we noticed that making images from the right corridor darker, yielded real-world predictions to be even more to the half left of the environment. In opposition, by making those images 15% brighter, yielded real-world predictions, more accurately, in the half right of the environment leading to a slight improvement of the overall performance of 1.5% (Figure 5.8 ②).

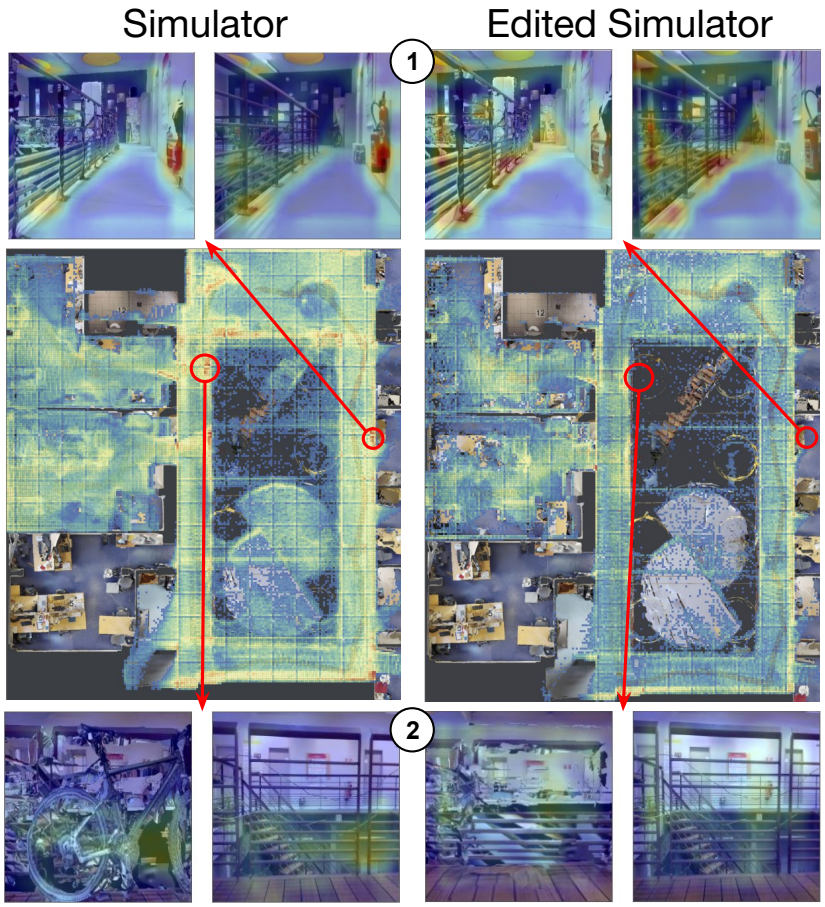


Figure 5.9 – With global heatmaps of *feature-distance*, we can observe (in red) areas of the environment that may be affected by a sim2real gap. Those areas correspond to changes in objects present in the simulation, for instance as illustrated here, a fire-extinguisher. By removing such objects in simulation and retraining a model on them, we can observe that they disappeared from most highlighted areas.

Sim2real gaps due to layout changes — Trained models deployed to real environments need to be resilient to dynamic layout changes such as opened/closed doors, the presence of humans or moved furniture and other objects. In SIM2REALVIZ, this can be investigated using the global heatmap with *feat-dist*, displayable with the *make heat* button on the middle top of SIM2REALVIZ as seen in Figure 5.4. In such geo-map overlay, some areas of the environment noticeably stand out (red in Figure 5.9). By hovering over instances on the geo-map nearby those areas, and browsing their corresponding images (as in Figure 5.4 ④), we can observe that those areas are triggered by different factors. For instance, in Figure 5.9 ②, the highlighted area corresponds to the presence of a bike in the simulated data, which was not present when the real-world data had been captured. Other areas correspond to changed furniture, and imperfections of the simulation when ren-

dering, for instance, a fire-extinguisher (Figure 5.9 ①). Such behavior, which can be observed across models, may benefit from specific attention while addressing sim2real transfer.

Editing the simulator — In order to test such a hypothesis, we manually edited the simulator and removed objects corresponding to two red areas using *blender*, a 3D modeling software. This new simulation is then used to generate 60*k* new images. Using these data, we trained a new *dataAug* model and loaded its predictions in SIM2REALVIZ to evaluate the influence of those changes on real-world performance. Figure 5.9 shows the global *feat-dist* heatmap on trajectory#1 created with the new model, taking into account the changes. We can see that the areas with the most significant differences are more uniformly distributed over the environment. Since global heatmaps are normalized over the complete dataset, this indicates that, to the model, those areas are now closer in feature space.

The experiments described above have led to a better understanding of the sim2real gap of our baseline agent, and we reported more robust localization performance once these insights were leveraged to modify the simulator or by learning filters for the existing model. We hope that SIM2REALVIZ will be adopted and facilitate the design and training of trained robotic agents.

5.7 Limitations and Perspectives

Evaluation — While the case studies detailed in Section 5.6 showcase how SIM2REALVIZ can be useful to identify potential sources of sim2real gaps, it would be beneficial to conduct a thorough expert evaluation—similar to the other visual analytics systems introduced in thesis. Such perspective should address both the usability of SIM2REALVIZ for Deep Learning (DL) experts, and their ability to identify sim2real gaps with our visual analytics system. To this end, we plan to evaluate SIM2REALVIZ in two ways. First, experts can be asked to identify the most adequate model and instances the most susceptible to convey sim2real gaps as quickly as possible. Time would be a relevant performance measure for this task, along with comparing the instances picked by experts. This should provide a better understanding of how reliable SIM2REALVIZ can be when used by experts. The second phase of the evaluation will be follow-up interviews with the same experts to collect qualitative insight from this experience. Thirdly, in order to evaluate SIM2REALVIZ's ability to identify the source of sim2real gaps, we propose to manually select instances, and balance their distribution on their potential presence or not of gaps, and rooms from which they are sampled. Then, these instances will be loaded in SIM2REALVIZ while hiding their real-world predictions, and ask experts to answer the following questions: 1) *"Where do you think the*

real-world prediction is?", 2) *"Do think such an instance may be sensitive to a sim2real gap?"*, and 3) *"If yes, what can it be?"*. Answers will be collected and analyzed quantitatively in order to assess SIM2REALVIZ's ability to empower experts to conduct an in-depth analysis of models' decisions.

Comparison of models and robots — Currently, in SIM2REALVIZ models can only be compared through the parallel plot visualization as illustrated in Figure 5.4 ①. However, such a comparison could also be beneficial if it was propagated to other views (*e.g.* the geo-map or the instance view). That way, we expect experts could to be more confident to evaluate if a particular model has overcome a sim2real gap that the others did not. On the geo-map, such a comparison could also enable experts to analyze if certain models are more adapted to certain areas of the environment, or rather, as illustrated in Section 5.6, follow the same *regression to the mean* bias. Furthermore, aside from comparing models, it could be useful to adapt SIM2REALVIZ to account for different robots. Indeed, many sim2real gaps may directly emerge from the robot itself. For example, a different focal lens of the camera, vibrations that blur images, or more basically a different colorimetry. In the future, being able to distinguish from which robot an instance is sampled, may empower experts to more quickly identify what are the differences and hence the potential gaps that may occur.

In addition, the only way to leverage any insight on how a model behaves is to train a new model on, for instance, a new simulation. However, such a process is time-consuming as it requires for a model to be completely trained for hours, and manually added into SIM2REALVIZ. This is doomed to be repeated if the analysis of the new model is not conclusive. Hence hindering the deployment of our system into any expert's workflow. This may be tackled by including the analysis of models in earlier stages of their training into SIM2REALVIZ. That way, experts could compare models without having to wait for their training, and hence more quickly assess if any of their updates is effective. While such a solution is limited as experts still need to manually adapt their models (*e.g.* add or remove layers, add a new loss function) outside of SIM2REALVIZ, we argue that we cannot anticipate every changes an expert may potentially execute on their model, and hence that at some point they will be forced to go back to their own code.

Future visualizations — Finally, another perspective is to extend the views of SIM2REALVIZ to include a deeper level of available information on the environment. For instance, the bird's eye geo-map could be combined with a 3D view of the simulation. Such a view can exploit the same first-person to coordinates projection to produce a heatmap on textures (*e.g.* a red tint on a wall could indicate a presence of a gap). This would result in a more immersive and precise analysis of sim2real gaps, while the bird's view provides a more global view of the environment's heatmap. Additionally, the encoding of positions on the geo-map may suffer

from overplotting issues as the number of instances to be displayed increases. Our glyph encoding only partially tackles this problem, occlusion issues can still be encountered as the density of positions increases. Future works could address this issue by relying on aggregation methods to limit the number of instances to be displayed. A representation for such an aggregation could be a new glyph that emphasizes the number of instances it encodes along with the overall discrepancies (*e.g.* average distance between them) between simulation, ground-truth, and real-world predictions.

Finally, Sim2RealViz heavily relies on real-world images with ground-truth labels, and hence their simulation equivalent. Here, such an alignment for pairs of images is done manually with the help of heuristics and a lidar. Hence, such a process is laborious and may have approximations. As a result, the amount of real-world data available for analysis is limited. Future works could loosen this requirement by empowering users with the ability to manually adjust ground-truth labels to fix any potential approximations. Furthermore, the requirement for the lidar to generate ground-truth labels of real-world images could be avoided. This can be done by giving real-world images (without any label) to the model and collecting their predictions. That way, we could rely on those potentially false predictions to generate to which simulation image they correspond to. By comparing both images, we may be able to assess if the model yielded correct coordinates and orientation angle for the real-world image. Such a comparison can either be done manually or with the help of metrics—(*e.g.* with the Mean Squared Error (MSE)). While this process does not provide the same sim/real alignment, it has the benefit to be able to process a large amount of data, and sort instances by how likely they are to be influenced by sim2real gaps. As a result, this approach could potentially help experts browse through more discrepancies, and eventually n adjust false predictions for more in-depth analysis which requires an alignment.

5.8 Conclusion

We introduced Sim2RealViz, an interactive visual analytics tool designed to perform an in-depth analysis of the emergence of sim2real gaps from neural networks applied to robot ego-localization. Sim2RealViz supports both overview and comparison of the performances of different neural models, which instances can be browsed based on metrics such as performance or distribution. Those metrics can be combined using set operations to formulate more elaborated queries. We also reported scenarios of use of Sim2RealViz to investigate how models are inclined to exploit biases, such as regression to the mean, and are easily disturbed by layout changes, such as moved objects.

CHAPTER 6

CONCLUSION AND FUTURE DIRECTIONS

Contents

6.1 Summary of Contributions

This thesis provides contributions to multiple fields, including interactive data visualization, machine learning, and how to build bridges between those two worlds. Those contributions can be summarized as follows:

Assessing reasoning bias in Visual Question Answering (VQA) — Preventing models, especially large ones such as transformers, to learn to blindly yield outputs known by heart instead of learning a proper reasoning process, is key for a future of Artificial Intelligence (AI) that impacts our everyday life. In Chapter 3, we identified, thanks to interviews and participatory design with model builders, their workflow to analyze bias in the attention of their models, and design goals for a visual analytics system to assist them. We designed VisQA, an interactive visual analytics system, with a new set of visualizations to analyze the attention mechanism of bi-modal transformer models. This visualization is the first to display attention maps in a layout following the studied model's structure enabling a fast identification, and selection of these maps while mitigating the learning curve required by such a tool. VisQA is available as an online prototype (`https://visqa.liris.cnrs.fr`), and is open-source (`https://github.com/Theo-Jaunet/VisQA`). This system is the first to address the bi-modality attention of VQA transformer models, and is the first to provide interactions to alter models' attention in real-time. As a result, experts can explore models' predictions and draw hypotheses on the role of a selected set of attention maps. Through an evaluation protocol we

designed, we showed first evidence that human users can obtain indications on the reasoning behavior of a neural network using our system. During this evaluation, domain experts, builders of Deep Neural Networks (DNN) models, were able to estimate whether the model correctly predicted an answer and whether it exploited biases. These insights lead to improvements of the state-of-the-art LXMERT model.

Understanding the Memory of Deep Reinforcement Learning (DRL) — The memory of robots trained with DRL is at the root of their decisions, thus analyzing it, and being able to grasp how it is used by a model may provide a glimpse into the black-box that is their decision process. In Chapter 4, we addressed the under-explored challenge of visualizing such a memory and went further by contextualizing it within the environment, previous decisions, and other metrics. After discussions with domain experts about their workflow of training DRL agents on navigation tasks, we designed a new interactive visual analytics system to help them study the memory of their trained models. This prototype is available online (https://sical.github.io/drlviz), and is open-source (https://github.com/sical/drlviz). In this system, the contextualization of the memory is carried by a surrounding set of multiple-coordinated views which can be used to filter and query any subset of memory. In particular, we provide a timeline of events dissecting a robot's action which can be used as an overlay of the memory through an interaction we designed. This tool has been evaluated by domain experts during qualitative interviews during which they noticed, among other things, how some memory elements may function as "flags" encoding whether or not the model has collected an item. It can be observed that in some cases the agent may miss an item and lose vision of it, inducing those memory elements to behave as they would if the agent had successfully gathered such an item. This led to the design of *memory reduction*, a new interaction directly tied to the model. Which such an interaction, a user can select a subset of the memory, and observe how the model would behave without it. This was combined with the intuition that the size of memories is manually picked by the model's builder, hence resulting in an often too large memory. Using this, we discovered that in some cases the robot can preserve a similar performance with only a key subset of less than half of the memory elements. We extended such interaction and addressed it to a non-expert audience in an online explorable (https://theo-jaunet.github.io/MemoryReduction), in which they were invited to explore different strategies to reduce a robot's memory, along with an explanation of how it works.

Identifying the Root of Gaps Between Simulation and Real-world — Despite great progress in DRL, robots are yet to be seen in our everyday life. This can be explained by the fact that in many cases robots trained in simulation tend to have unexpected behavior when deployed into real-world conditions. However, we

need to rely on simulation to train them as they require millions, and sometimes billions, of interactions with their environment to learn, which is infeasible in real-time. In Chapter 5, we introduced non-expert users to this challenge, by designing *Théo Guesser*, and online explorable (https://theo-jaunet.github.io/theo-guesser/) in which users are invited to explore how a model trained in simulation might be sensible to noises on input images, and one may tackle such an issue. In addition, we addressed the challenge of transferring the knowledge of a DNN model trained for ego-pose localization on simulation to the real-world. To do so, we designed and trained a model to yield its position within its environment from an image input. Along with such a model, we designed an interactive visual analytics system to visually compare how the model may differently perceive a simulation and a real-world environment. This open-source system (https://github.com/Theo-Jaunet/sim2realViz) provides an overview in multiple-coordinated views on how the model performs in simulation and real-world, which can be used to provide a bird's eye view of the environment, and only display data-points that may be altered by real-world conditions (*e.g.* a moving object). In such a system, we provide visualization methods, such as the ability to convert gradient heatmaps such as grad-cam (as introduced in Section 2.2.2) from the robot's first-person point of view, into a 2D bird's eye view of the environment. By doing so, one can quickly analyze which are the most troublesome areas of the environment and hence act in consequence (*e.g.* edit the simulator). Additionally, in this chapter we managed for the simulation and real-world images to be aligned—*i.e.* have a simulation and real-world image sampled from the same coordinates and camera orientation. Inspired by occlusion [239], we designed a new technique named domain occlusion which applies patches over an input image to investigate if the output is further or closer than the ground-truth. However, instead of applying gray patches, we switch from one domain to another, *e.g.* replace a region of an image from simulation with the region from real-world. This allows to assess how the model may exploit biases such as a regression to the mean, and whether it is sensitive to moving objects such as bikes, or wrong texture mapping in simulation. Thanks to the system's ability to directly manipulate the simulator settings, and apply filters (*e.g.* increase brightness) to real-world images, we explored how those biases and layout-related issues may be tackled.

6.2 Perspectives for Future Works

The contributions of this thesis, along with the recent advances from literature, spurred new opportunities, and a wide range of perspectives for future work. In this section, we distilled some possible directions for the challenges we tackled

in this thesis, along with the rise of more global research directions for the community.

6.2.1 Invade and Conquer Model Builders' Workflow

Despite the plethora of visual analytics systems, and building blocks for DNNs' interpretability designed by the community, interpretability is yet to be a common practice in model builders' workflow. Nonetheless, during evaluations, it has been shown that with the help of those systems, model builders were able to identify issues among their models, and eventually leverage such information to improve them. We can therefore raise the question: *Why are such tools not included in the design process of models yet?* A light can be shed onto this question using the following lenses:

The trenches of training and optimization— A vast majority of visual analytics systems for interpretability, including those introduced in this manuscript, are designed and used to analyze trained DNNs, which is often done after the hyper-parameter optimization phase. As a result, most of this process, which is the most time-consuming step for model builders, is excluded from these visual analytics systems, hence forcing them to rely on alternative tools and methodology to study their models in their early stages. For example, a common approach is to either directly monitor a model's loss evolution on a terminal console, or rather to use TensorBoard [231, 207]. TensorBoard is one of the most popular toolkits, which, among other features, enables users to plot metrics as line charts. We argue that easing the burden of optimizing a model under construction and including this step of model-building into the design of our visual analytics systems could encourage model builders to share such a time-span, usually allocated to optimization, with interpretability. Thus reinforcing the message that a model's accuracy on a dataset does not describe its ability to exploit, or convey biases [56]. Currently, such a step may be incorporated manually in visual analytics systems by using snapshots of a model. However, in addition to being too laborious to be incorporated into builders' workflow, such a solution, and our current visual analytics systems also lack the capability to compare models. Such a comparison is, however, core to the optimization phase, and perhaps could also be crucial for its interpretability. For instance, comparing activations of two different models using the same input may reveal which one is relying on the most interpretable information to convey an output.

Vanquish the learning curve— By itself, understanding the decision process arising from a single neuron is straightforward. It is their overpowering number within a DNN, and the time it would require for a Human to analyze them all, that makes a model a "black-box". Similarly, our visual analytics systems, through their

large number of multi-coordinated visualizations, interactions available, and data displayed tend to be overwhelming for users. This results in a steep learning curve, which in many cases discourages users and may even prevent them to understand the core functionalities of a visual analytics system without any assistance. This is emphasized by the fact that many visual analytics systems are instance-based, meaning that they are dedicated to the analyzing of a single data-point and output at a time. While this provides a fine-grained analysis, this also raises the challenging task of selecting which data-point may be susceptible to portrait a model's faults, and hence worth analyzing. As a result, those visual analytics systems tend to convey a feeling of *"Where to begin?"* even to experimented users. Currently, in the community, such an issue is tackled through the use of an interactive representation block (*e.g.* Chapter 4), in which one may click on any dot, updating other views with the corresponding input. Another approach is to rely on a metric describing the model performance to re-order inputs, and draw users' attention to the worst ones (*e.g.* Chapter 3). However, this approach depends on the metric's ability to describe bias exploitation from a model, and the users' comprehension of such a metric. Finally, some works in the shape of prototypes, choose to only offer the possibility to switch between a handful of inputs, manually selected for their ability to portrait the studied model's biases, hence avoiding for their users the task of finding relevant inputs to analyze. With the ever-growing size of state-of-the-art models and the amount of data they digest, future studies should focus on understanding how experts use visual analytics systems, and provide different levels of information, in order to ease their experience and screen-space real-estate taken by visualizations, and hence limit the amount of time required to efficiently analyze a single data-point. Along with such a study, understanding how to more efficiently identify inputs that should interest the user, *e.g.* with data-mining methods, and more global visualization of models' behavior is a core research direction for better interpretability of DNNs.

Blitz deployment of systems— Designing a visual analytics system to interpret a model's decision requires time, and lasting collaboration between model builders, and visual analytics designers. This is emphasized by the fact that such systems often require a lot of programming and engineering, shaped as the collection of data from models (*e.g.* activations), the construction of systems, and the infrastructure to bridge models often on python, and in-browser visualizations on JavaScript, along with knowledge from both worlds. As a result, in most cases, the emergence of tools to interpret a new model comes with a delay. Thus many published models are not evaluated with respect to their interpretability, and visual analytic systems fail to cope with the ever-increasing fast-paced creation of new models. This is because those systems are designed for one specific implementation of a model within a class of models (*e.g.* the LXMERT among transformers in Chapter 3), a specific dataset (*e.g.* GQA), a unique task (*e.g.* VQA).

Therefore, the deployment of a visual analytics system to another configuration requires for such an engineering time needs to be repeated, hence hindering model builders from easily interpreting variations of a model. However, as experimented in this manuscript despite different models, datasets, and tasks addressed, our visual analytics systems and those from the community share common features. Each of those systems needs to obtain information from models, and for example, share common visualizations such as the visualization of attention and memory as heatmaps, the presence of interactive representation view issued from a dimensionality reduction algorithm, or even the softmax distribution of top classes for an output. Hence, in the future, we should inquire how can we rest on the shoulders of giants, and ease the burden of future engineering requirements. This can be done by adapting a system to another related model, by studying and designing more modular and generic systems. Along with that comes the need for future works to tackle the unexplored challenge of adapting those visual interfaces to the ever-increasing size of models and dataset, dampening their ability to have a reasonable response time for their interactions, and increasing overall performances required to even load those systems in a web browser.

6.2.2 Mitigating Human Biases

When analyzing and interpreting models' decision processes one should also be on alert for one's own biases. For instance, as humans, we tend to expect models to have explanations for each of their decisions, because we mistakenly believe that each of our decisions has rational explanations, instead of anchored priors [96]. Such biases may blind the shortcomings of our interpretations and should be taken into account when analyzing a model's decision. Future works could address this with the following research directions:

Evaluate interpretations — When interpreting a decision, we, as humans, are looking for patterns and clues that help us seek a reasoning process similar to our own, thanks to DNN's data (*e.g.* activations). This is known as confirmation bias, and because of this, we may interpret an explanation as in favor of our intuition concerning a decision, and consider it as an artifact any explanation that points toward something different. As an example, a heatmap around a cat's whiskers from an input image may be interpreted as the model correctly assimilating that indeed cats have whiskers, when in reality, it may be the model mistaking the background in-between whiskers as a discriminatory feature for cats. Furthermore, deciding when one has enough information to interpret a model's decision, and not missing any piece of information that could shatter one's conclusion, is a challenge even for model builders. This may result in a trade-off between jumping to conclusions in the interest of time and eventually missing crucial information, and being too meticulous, hence taking too much

time on a single input and eventually skipping inputs that could yield valuable insight.

The study of human biases, and their effect on decision-making through visualization [224], and visual analytics [223], is an on-growing area of research that provided metrics describing biases to a user [223], and a design space to mitigate cognitive biases [225] which may be hidden when analyzing a visualization. In addition to those contributions, we should also consider studying those biases with respect to DNN interpretability. Hence a rising need for stronger evaluation protocols of how users may interpret decisions thanks to visual analytics systems. Currently, a vast majority of research reports introducing new visual analytics systems for DNN interpretability provide illustrative examples and case studies. While this may showcase how a new system can be used and what kind of explanation can be found with it, we argue that a more thorough evaluation of how users may adapt to this new system, and evaluation of the interpretations they can grasp may be beneficial. This trend may be explained by the fact that interpreting a model's decision is a challenging task, and thus such an operation is often addressed in lengthy qualitative interviews, with model builders experimented with the studied model. Such conditions limit the pool of available experts able to participate in an experiment, resulting in only a handful of experts. In a few cases, these experts contributed to the design of the visual analytics system being evaluated, which hinders the evaluation of a new user's adaptation to the system. Despite the emergence of strategies and protocols to evaluate an interpretation [56, 181], the quantification of what is a "good" interpretation, and when a model is "good enough" remains an open challenge. Thus, their applications to visual analytics remain elusive. As an example, for VisQA (Chapter 3), such an application relies on a more practical approach consisting in hiding the model's output and asking users to estimate, based on their analysis, what it should be. That way, we quantitatively compare the results between users, and between an answer and the model's prediction.

Include end-users into the loop — When building a model, we tend to follow our intuition and biases based on our experience. For example, with statements such as *"Traditionally this amount of parameters are enough"*, or *"Usually such a loss yields great results"*. However, it has been shown that such fine-grained changes may have a direct impact on how a model behaves [173]. A solution would be to formulate an exhaustive exploration of every combination of hyper-parameters per model, which is not feasible in a timely manner and would sky-rocket the energy consumption of model training. This illustrates that breaking away from Human biases is nearly impossible, especially if a model builder analyzing a model has no knowledge of what may constitute an inequality of opportunity [81]. Thus, a key research direction is to empower end-users with the ability to experiment not only how a model may treat one's data, but also how other cases are handled.

However, this leads to the seemingly insurmountable challenge of adapting visual analytics systems to accommodate non-expert users to prevent them from being overwhelmed. Or rather build on existing non-expert works for interpretability, which are often relying on simplified models, and toy datasets. Those works then need an adaptation to fit the desired model and dataset. Such an adaptation may, however, bring back the same feeling of being overwhelmed, despite having a limited exploration space and hence a chance to miss potential explanations. Moreover, even with the consideration that end-users are able to analyze the decision of models, how they may leverage any insight to improve them, remains an open problem.

6.2.3 Finally Finding those Keys!

Throughout this manuscript, we designed and analyzed DNN models, each addressing a step required for a robot to answer a question such as: "Where are my keys?". The next logical step is to combine these models into a single stack to finally have a robot able to understand what we are looking for from a natural language question, and through its environment (*e.g.* an apartment) finding and communicating its location. To do so, the VQA model, as introduced in Chapter 3, should both communicate to the model dedicated to the robot's navigation what it should look for, and when it should stop (*e.g.* when the keys are visible). Technically, this would require for the navigation model to be re-trained to support a slightly different task, in a more realistic simulator. Instead of reaching several items in an order, as instructed by a reward function, the model must focus on exploring efficiently its environment (*e.g.* avoid coming back into previously scanned rooms), and thus yield the robot's next actions until the VQA model detects the presence of "keys". This approach could be improved as it would result in a robot aimlessly roaming its environment, instead of being guided by intuitive biases. For instance, if the navigation model knew that it is looking for keys, it could exploit prior knowledge such as that the keys are often left on a desk. Hence a better solution would be to adapt the VQA model to share with the navigation model what item it must detect. Such information may take the form of an embedding of the question or rather be issued from a modification of the VQA model with an auxiliary branch trained to output the name of the object from a question (*e.g.* "keys") which can then be used as input to the navigation model (using a representation of this class (*e.g.* one-hot vector, embedding, or a word2vec representation). That way, such a model, may learn the intuition that the keys are often on the same desk, directly head for it without exploring the complete environment. And once the VQA model detects those keys, it can provide the ego-pose localization model with the image from the robot's perspective to yield the position of the keys. In that sense, one could consider for the navigation model, and the ego-pose localization one the share the

same Convolutional layers and weights, since both of their goals require an inner representation of the environment's layout. This would have the benefit of easing endeavors to interpret the decisions of those models as they would have fewer parameters to analyze. This has also the benefit of preserving task-specialized visualizations on their respected fully-connected layers.

Similarly, the task of using a robot to answer VQA questions, known as Embodied Question Answering (EQA) [48], with a hierarchical approach gained popularity [75, 49]. This approach combines low-level controllers executing actions (similar to our navigation model), along with higher-level modules either designed to process the natural language questions, or designed as a planner that yields a sequence of action to execute. Alternatively, EQA have been tackled by a single model. For instance, in target-driven navigation [245], such a model is asked to reach a destination by using as input the image corresponding to the point of view of the robot, along is a target image corresponding to the view from the robot's perspective of its destination. More recently, Suglia et al. [197] relied on a single transformer model using jointly the input image of the robot's point of view, and is asked to yield the next action the robot should do. While there is no consensus in the community on whether a single model, or a multiple-stacked one, may have the best performances, we argue that relying on multiple models may be more interpretable. One argument is that each model may be interpreted and evaluated with respect to its own goal—*e.g.* the ability to identify the presence of keys in an image. Therefore, model builders may be able to act upon the observation of bias exploitation on a specific task (*e.g.* re-train a model with new loss), without dampening the performances of other models.

Nonetheless, in contrast to the works introduced in this manuscript (*i.e.* the study individual models), in large industries, the deployment of multiple stacked models is frequent *e.g.* Google home [120], or Amazon's Alexa [210], and by self-driving cars by Waymo [13, 198] or Tesla [98, 9]. Those solutions, and the future works for EQA stacked models, carry the yet-to-tackle challenge of designing visual analytics systems able to help domain experts explore the overwhelming amount of data those collection models may represent, along with how they depend on each other. For example, raising questions such as: "Is a mistake occurring because the last model needs to be re-trained, or rather because it received faulty inputs?". This is emphasized by the fact that in many cases models within a stack are trained individually, and hence the model builder may not have access to the upstream, *i.e.* to inner workings of models it depends on, nor to the downstream ones affected by it. Thus, a challenge for designing more interpretable models is not only to mitigate their bias exploitation, but also to understand how their decisions may induce errors in downstream models of a stack. As an example, the LXMERT model introduced in Chapter 3, first receives inputs from a RCNN model, whose noisy predictions have a direct impact on the downstream transformer model's ability to learn a proper reasoning. This

influence may not be straightforward to grasp when analyzing the RCNN by itself. Such findings pave the road for new research opportunities, especially for the interpretability and understanding of biases that may be buried within decisions and the design of models in a stack.

BIBLIOGRAPHY

[1] Ehsan Abbasnejad, Damien Teney, Amin Parvaneh, Javen Shi, and Anton van den Hengel. "Counterfactual vision and language learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 10044–10054 (cit. on p. 47).

[2] Abien Fred Agarap. "Deep Learning using Rectified Linear Units (ReLU)". In: *arXiv preprint arXiv:1803.08375* (Mar. 2018). URL: https://arxiv.org/abs/1803.08375v2 (cit. on p. 11).

[3] Aishwarya Agrawal, Dhruv Batra, and Devi Parikh. "Analyzing the behavior of visual question answering models". In: *arXiv preprint arXiv:1606.07356* (2016) (cit. on pp. 40, 47).

[4] Aishwarya Agrawal, Dhruv Batra, Devi Parikh, and Aniruddha Kembhavi. "Don't just assume; look and answer: Overcoming priors for visual question answering". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 4971–4980 (cit. on p. 47).

[5] *ALL OF IT | Explorable Explanations*. URL: https://explorabl.es/all/ (cit. on p. 7).

[6] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. "Bottom-up and top-down attention for image captioning and visual question answering". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 6077–6086 (cit. on p. 44).

[7] Robert Andrews, Joachim Diederich, and Alan B. Tickle. "Survey and critique of techniques for extracting rules from trained artificial neural networks". In: *Knowledge-Based Systems* 8.6 (Dec. 1995), pp. 373–389 (cit. on p. 13).

[8] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. "Vqa: Visual question answering". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 2425–2433 (cit. on p. 39).

[9] *Artificial Intelligence & Autopilot | Tesla*. URL: https://www.tesla.com/AI (cit. on p. 125).

[10] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. "Deep Reinforcement Learning: A Brief Survey". In: *IEEE Signal Processing Magazine* 34.6 (Nov. 2017), pp. 26–38 (cit. on p. 75).

[11] Daniel Asimov. "The Grand Tour: A Tool for Viewing Multidimensional Data". In: *Scientific and Statistical Computing* 6.1 (July 2006), pp. 128–143 (cit. on p. 27).

[12] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate". In: *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. International Conference on Learning Representations, ICLR, Sept. 2015. URL: https://arxiv.org/abs/1409.0473v7 (cit. on pp. 22, 32).

[13] Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. "ChauffeurNet: Learning to Drive by Imitating the Best and Synthesizing the Worst". In: *Robotics: Science and Systems*. 2019. URL: https://sites.google.com/view/learn-to-drive/ (cit. on p. 125).

[14] Dhruv Batra, Aaron Gokaslan, Aniruddha Kembhavi, Oleksandr Maksymets, Roozbeh Mottaghi, Manolis Savva, Alexander Toshev, and Erik Wijmans. "ObjectNav Revisited: On Evaluation of Embodied Agents Navigating to Objects". In: *CoRR* (Jan. 2020) (cit. on p. 75).

[15] David Bau, Hendrik Strobelt, William Peebles, Jonas Wulff, Bolei Zhou, Jun-Yan Zhu Torralba, and Antonio. "Semantic Photo Manipulation with a Generative Image Prior". In: *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)* 38.4 (2019) (cit. on p. 2).

[16] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. "Network Dissection: Quantifying Interpretability of Deep Visual Representations". In: *Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 6541–6549. URL: http://netdissect.csail.mit.edu (cit. on pp. 19, 20).

[17] Alex Bäuerle and James Wexler. "What does BERT dream of?" In: *Proceedings of the Workshop on Visualization for AI explainability (VISxAI)*. Ed. by Mennatallah El-Assady, Duen Horng (Polo) Chau, Fred Hohman, Adam Perer, Hendrik Strobelt, and Fernanda Viégas. 2020 (cit. on p. 24).

[18] Edward Beeching, Christian Wolf, Jilles Dibangoye, and Olivier Simonin. "Deep Reinforcement Learning on a Budget: 3D Control and Reasoning Without a Supercomputer". In: *arXiv preprint arXiv:1904.01806* (2019) (cit. on pp. 74, 75, 86, 93).

[19] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. "The Arcade Learning Environment: An Evaluation Platform for General Agents". In: *Journal of Artificial Intelligence Research* (2013) (cit. on p. 75).

[20] Y Bengio, Aaron Courville, Dumitru Erhan, Yoshua Bengio, and Pascal Vincent. *Visualizing Higher-Layer Features of a Deep Network Visualizing Higher-Layer Features of a Deep Network Département d'Informatique et Recherche Opérationnelle*. Tech. rep. University of Montreal, 2009. URL: https://www.researchgate.net/publication/265022827 (cit. on p. 23).

[21] Tolga Bolukbasi, Kai-Wei Chang, James Y. Zou, Venkatesh Saligrama, and Adam T. Kalai. "Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings". In: *Advances in Neural Information Processing Systems* 29 (2016) (cit. on p. 3).

[22] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. "D3: Data-Driven Documents". In: *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)* (2011) (cit. on pp. 58, 84, 104).

[23] Bostock Mike. *Let's Try t-SNE!* URL: https://observablehq.com/@mbostock/lets-try-t-sne (cit. on p. 26).

[24] Léon Bottou. "From machine learning to machine reasoning". In: *Machine learning* 94.2 (2014), pp. 133–149 (cit. on p. 40).

[25] Konstantinos Bousmalis, Alex Irpan, Paul Wohlhart, Yunfei Bai, Matthew Kelcey, Mrinal Kalakrishnan, Laura Downs, Julian Ibarz, Peter Pastor, Kurt Konolige, and others. "Using simulation and domain adaptation to improve efficiency of deep robotic grasping". In: *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 4243–4250 (cit. on p. 101).

[26] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. "Unsupervised pixel-level domain adaptation with generative adversarial networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 3722–3731 (cit. on p. 101).

[27] Eric Brachmann, Martin Humenberger, Carsten Rother, and Torsten Sattler. "On the Limits of Pseudo Ground Truth in Visual Camera Re-localisation". In: *arXiv preprint arXiv:2109.00524* (Sept. 2021). URL: http://arxiv.org/abs/2109.00524 (cit. on p. 100).

[28] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems* 2020-December (May 2020). URL: https://arxiv.org/abs/2005.14165v4 (cit. on pp. 2, 3).

[29] G Brunner, Y Liu, DP Ortiz, O Richter, and M Ciaramita. "On Identifiability in Transformers". In: *International Conference on Learning Representations (ICLR)*. 2020. URL: https://research.google/pubs/pub48972/ (cit. on p. 34).

[30] Emanuele Bugliarello, Ryan Cotterell, Naoaki Okazaki, and Desmond Elliott. "Multimodal Pretraining Unmasked: A Meta-Analysis and a Unified Framework of Vision-and-Language BERTs". In: *Transactions of the Association for Computational Linguistics* (2021). URL: https://arxiv.org/abs/2011.15124 (cit. on p. 65).

[31] Remi Cadene, Hedi Ben-Younes, Matthieu Cord, and Nicolas Thome. "Murel: Multimodal relational reasoning for visual question answering". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 1989–1998 (cit. on p. 79).

[32] Jize Cao, Zhe Gan, Yu Cheng, Licheng Yu, Yen-Chun Chen, and Jingjing Liu. "Behind the scene: Revealing the secrets of pre-trained vision-and-language models". In: *European Conference on Computer Vision*. Springer, 2020, pp. 565–580 (cit. on p. 46).

[33] Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, Alina Oprea, and Colin Raffel. "Extracting Training Data from Large Language Models". In: (Dec. 2020). URL: https://arxiv.org/abs/2012.07805v2 (cit. on p. 3).

[34] Shan Carter, Zan Armstrong, Ludwig Schubert, Ian Johnson, and Chris Olah. "Activation Atlas". In: *Distill* (2019) (cit. on p. 31).

[35] Shan Carter, David Ha, Ian Johnson, and Chris Olah. "Experiments in Handwriting with a Neural Network". In: *Distill* (2016) (cit. on pp. 21, 22, 46, 82).

[36] DV Carvalho, EM Pereira, and JS Cardoso. "Machine learning interpretability: A survey on methods and metrics". In: *Electronics* 8.8 (2019), p. 832. URL: https://www.mdpi.com/503892 (cit. on p. 5).

[37] Dylan Cashman, Genevieve Patterson, Abigail Mosca, and Remco Chang. "RNNbow: Visualizing Learning via Backpropagation Gradients in Recurrent Neural Networks". en. In: (), p. 9 (cit. on pp. 32, 79).

[38] Aaron Chadha and Yiannis Andreopoulos. "Improved techniques for adversarial discriminative domain adaptation". In: *IEEE Transactions on Image Processing* 29 (2019), pp. 2622–2637 (cit. on pp. 97, 101).

[39] A Chandrasekaran, V Prabhu, D Yadav, P Chattopadhyay, and D Parikh. "Do explanations make VQA models more predictable to a human?" In: *EMNLP*. 2018 (cit. on p. 59).

[40] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. "Matterport3D: Learning from RGB-D Data in Indoor Environments". In: *International Conference on 3D Vision (3DV)*. 2017, pp. 667–676. URL: https://matterport.com/ (cit. on pp. 75, 94, 99).

[41] Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. "Uniter: Universal image-text representation learning". In: *European Conference on Computer Vision*. Springer, 2020, pp. 104–120 (cit. on pp. 45, 65).

[42] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. "Empirical evaluation of gated recurrent neural networks on sequence modeling". In: *arXiv preprint arXiv:1412.3555* (2014) (cit. on p. 77).

[43] *CNNVIs online prototype*. URL: http://shixialiu.com/publications/cnnvis/demo/ (cit. on p. 30).

[44] Andy Cockburn, Amy Karlson, and Benjamin B Bederson. "A review of overview+ detail, zooming, and focus+ context interfaces". In: *ACM Computing Surveys (CSUR)* 41.1 (2009), pp. 1–31 (cit. on p. 105).

[45] Kristin A; Cook and James J Thomas. *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. Los Alamitos: IEEE Computer Society Press, 2005. URL: https://www.osti.gov/biblio/912515 (cit. on p. 10).

[46] Nicolas Coudray, Paolo Santiago Ocampo, Theodore Sakellaropoulos, Navneet Narula, Matija Snuderl, David Fenyö, Andre L. Moreira, Narges Razavian, and Aristotelis Tsirigos. "Classification and mutation prediction from non–small cell lung cancer histopathology images using deep learning". In: *Nature Medicine 2018 24:10* 24.10 (Sept. 2018), pp. 1559–1567. URL: https://www.nature.com/articles/s41591-018-0177-5 (cit. on p. 2).

[47] Matthew Crosby, Benjamin Beyret, and Marta Halina. "The Animal-AI Olympics". In: *Nature Machine Intelligence* 1.5 (2019), p. 257 (cit. on p. 94).

[48] A Das, S Datta, G Gkioxari, S Lee, D Parikh, and D Batra. "Embodied Question Answering". In: *CVPR*. 2018 (cit. on pp. 71, 125).

[49] Abhishek Das, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. "Neural Modular Control for Embodied Question Answering". In: *Proceedings of the Conference on Robot Learning (CoRL)*. 2018 (cit. on p. 125).

[50] Taowei David Wang, Catherine Plaisant, Alexander J Quinn, Roman Stanchak, Ben Shneiderman, and Shawn Murphy. *Aligning Temporal Data by Sentinel Events: Discovering Patterns in Electronic Health Records*. 2008 (cit. on p. 92).

[51] Q Debard, J Dibangoye, S Canu, and C Wolf. "Learning 3D Navigation Protocols on Touch Interfaces with Cooperative Multi-Agent Reinforcement Learning". In: *To appear in European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*. 2019 (cit. on p. 71).

[52] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. 2009, pp. 248–255 (cit. on p. 31).

[53] Joseph F DeRose, Jiayao Wang, and Matthew Berger. "Attention Flows: Analyzing and Comparing Attention Mechanisms in Language Models". In: *IEEE Transactions on Visualization and Computer Graphics* (2020) (cit. on pp. 33, 46, 65).

[54] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "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)*. 2019, pp. 4171–4186 (cit. on pp. 33, 42, 45).

[55] E W Dijkstra. "A note on two problems in connexion with graphs". In: *Numerische mathematik* 1.1 (1959), pp. 269–271 (cit. on p. 71).

[56] Finale Doshi-Velez and Been Kim. *Towards A Rigorous Science of Interpretable Machine Learning*. Tech. rep. URL: https://arxiv.org/pdf/1702.08608.pdf (cit. on pp. 5, 6, 13, 120, 123).

[57] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=YicbFdNTTy (cit. on p. 40).

[58] Anca D Dragan, Kenton C T Lee, and Siddhartha S Srinivasa. "Legibility and predictability of robot motion". In: *2013 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 2013, pp. 301–308 (cit. on pp. 5, 35).

[59] B Duke, A Ahmed, C Wolf, P Aarabi, and G W Taylor. "SSTVOS: Sparse Spatiotemporal Transformers for Video Object Segmentation". In: *CVPR*. 2021 (cit. on p. 40).

[60] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. "Go-Explore: a New Approach for Hard-Exploration Problems". In: (Jan. 2019). URL: https://arxiv.org/abs/1901.10995v4 (cit. on p. 4).

[61] Carsten Eickhoff. "Cognitive biases in crowdsourcing". In: *Proceedings of the eleventh ACM international conference on web search and data mining*. 2018, pp. 162–170 (cit. on p. 47).

[62] Jesse H Engel, Kumar Krishna Agrawal, Shuo Chen, Ishaan Gulrajani, Chris Donahue, and Adam Roberts. "GANSynth: Adversarial Neural Audio Synthesis". In: *7th International Conference on Learning Representations, (ICLR)*. OpenReview.net, 2019. URL: https://openreview.net/forum?id=H1xQVn09FX (cit. on p. 2).

[63] *Explorable explanation - Wikipedia*. URL: https://en.wikipedia.org/wiki/Explorable_explanation (cit. on p. 7).

[64] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. "Robust Physical-World Attacks on Deep Learning Visual Classification". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 1625–1634. URL: https://iotsecurity.eecs.umich.edu/#roadsigns (cit. on p. 4).

[65] Kuan Fang, Yunfei Bai, Stefan Hinterstoisser, Silvio Savarese, and Mrinal Kalakrishnan. "Multi-task domain adaptation for deep learning of instance grasping from simulation". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 3516–3523 (cit. on p. 101).

[66] Johannes Fuchs, Fabian Fischer, Florian Mansmann, Enrico Bertini, and Petra Isenberg. "Evaluation of alternative glyph designs for time series data in a small multiple setting". In: *Proceedings of the SIGCHI conference on human factors in computing systems*. 2013, pp. 3237–3246 (cit. on p. 92).

[67] Zhe Gan, Yen-Chun Chen, Linjie Li, Chen Zhu, Yu Cheng, and Jingjing Liu. "Large-Scale Adversarial Training for Vision-and-Language Representation Learning". In: *Advances in Neural Information Processing Systems*. Ed. by H Larochelle, M Ranzato, R Hadsell, M F Balcan, and H Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 6616–6628. URL: https://proceedings.neurips.cc/paper/2020/file/49562478de4c54fafd4ec46fdb297de5-Paper.pdf (cit. on p. 45).

[68] Peng Gao, Zhengkai Jiang, Haoxuan You, Pan Lu, Steven C H Hoi, Xiaogang Wang, and Hongsheng Li. "Dynamic fusion with intra-and inter-modality attention flow for visual question answering". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 6639–6648 (cit. on p. 42).

[69] Leon Gatys, Alexander Ecker, and Matthias Bethge. "A Neural Algorithm of Artistic Style". In: *IEEE conference on computer vision and pattern recognition*. Vol. 16. 12. Association for Research in Vision and Ophthalmology (ARVO), Sept. 2016, p. 326 (cit. on p. 2).

[70] R Girdhar, J Carreira, C Doersch, and A Zisserman. "Video Action Transformer Network". In: *CVPR*. 2020 (cit. on p. 40).

[71] Michael Gleicher, Danielle Albers, Rick Walker, Ilir Jusufi, Charles D Hansen, and Jonathan C Roberts. "Visual comparison for information visualization". In: *Information Visualization* 10.4 (2011), pp. 289–309. URL: http://ivi.sagepub.com/content/10/4/289.short (cit. on pp. 93, 105).

[72] Gabriel Goh, Chelsea Voss, Daniela Amodei, Shan Carter, Michael PetrovJustin, Jay Wang, Nick Cammarata, and Chris Olah. *Multimodal Neurons in Artificial Neural Networks*. 2021. URL: https://openai.com/blog/multimodal-neurons/#rf22 (cit. on p. 4).

[73] Tejas Gokhale, Pratyay Banerjee, Chitta Baral, and Yezhou Yang. "MUTANT: A Training Paradigm for Out-of-Distribution Generalization in Visual Question Answering". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2020, pp. 878–892 (cit. on p. 47).

[74] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples". In: *arXiv preprint arXiv:1412.6572* (2014) (cit. on pp. 4, 97).

[75] Daniel Gordon, Aniruddha Kembhavi, Mohammad Rastegari, Joseph Redmon, Dieter Fox, and Ali Farhadi. "IQA: Visual Question Answering in Interactive Environments". In: *CVPR*. IEEE, 2018 (cit. on pp. 71, 125).

[76] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. "Making the V in VQA matter: Elevating the role of image understanding in Visual Question Answering". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 6904–6913 (cit. on pp. 40, 47, 65).

[77] Yash Goyal, Akrit Mohapatra, Devi Parikh, and Dhruv Batra. "Towards transparent ai systems: Interpreting visual question answering models". In: *arXiv preprint arXiv:1608.08974* (2016) (cit. on p. 46).

[78] Klaus Greff, Rupesh Kumar Srivastava, Jan Koutn\textbackslash'\{{\textbackslash}textback
Bas R Steunebrink, Jürgen Schmidhuber, Jan Koutník, Bas R Steunebrink,
Jürgen Schmidhuber, Jan Koutn\textbackslash'\{{\textbackslash}textbackslashi}k,
Bas R Steunebrink, Jürgen Schmidhuber, Jan Koutník, Bas R Steunebrink,
and Jürgen Schmidhuber. "LSTM: A Search Space Odyssey". In: 10 (), pp. 2222–2232 (cit. on p. 77).

[79] Sam Greydanus, Anurag Koul, Jonathan Dodge, and Alan Fern. "Visualizing and understanding atari agents". In: *arXiv preprint arXiv:1711.00138* (2017) (cit. on p. 81).

[80] Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. "A survey of deep learning techniques for autonomous driving". In: *Journal of Field Robotics* 37.3 (Apr. 2020), pp. 362–386. URL: https://onlinelibrary.wiley.com/doi/full/10.1002/rob.21918https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21918https://onlinelibrary.wiley.com/doi/10.1002/rob.21918 (cit. on p. 2).

[81] Moritz Hardt, Eric Price, Eric Price, and Nati Srebro. "Equality of Opportunity in Supervised Learning". In: *Advances in Neural Information Processing Systems* 29 (2016) (cit. on p. 123).

[82] P E Hart, N J Nilsson, and B Raphael. "A formal basis for the heuristic determination of minimum cost paths". In: *IEEE transactions on Systems Science and Cybernetics* 4.2 (1968), pp. 100–107 (cit. on p. 71).

[83] Matthew Hausknecht and Peter Stone. "Deep Recurrent q-learning for Partially Observable mdps". In: *2015 AAAI Fall Symposium Series*. 2015 (cit. on p. 75).

[84] Lisa Anne Hendricks, Zeynep Akata, Marcus Rohrbach, Jeff Donahue, Bernt Schiele, and Trevor Darrell. "Generating Visual Explanations". In: *European conference on computer vision*. Vol. 9908 LNCS. Springer, 2016, pp. 3–19. URL: https://link.springer.com/chapter/10.1007/978-3-319-46493-0_1 (cit. on p. 16).

[85] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. "Distilling the Knowledge in a Neural Network". In: *NIPS Deep Learning and Representation Learning Workshop*. 2015. URL: https://research.google/pubs/pub44873/ (cit. on p. 28).

[86] Fred Hohman, Haekyu Park, Caleb Robinson, and Duen Horng Chau. "Summit: Scaling Deep Learning Interpretability by Visualizing Activation and Attribution Summarizations". In: *IEEE Transactions on Visualization and Computer Graphics (TVCG)* (2020) (cit. on p. 31).

[87] Fred Matthew Hohman, Minsuk Kahng, Robert Pienta, and Duen Horng Chau. "Visual Analytics in Deep Learning: An Interrogative Survey for the Next Frontiers". In: *IEEE Transactions on Visualization and Computer Graphics* (2019) (cit. on pp. 10, 29, 41, 72, 80, 85, 98).

[88] Benjamin Hoover, Hendrik Strobelt, and Sebastian Gehrmann. "exBERT: A Visual Analysis Tool to Explore Learned Representations in Transformer Models". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Online: Association for Computational Linguistics, July 2020, pp. 187–196. URL: https://aclanthology.org/2020.acl-demos.22 (cit. on pp. 33, 34).

[89] Zhicheng Huang, Zhaoyang Zeng, Bei Liu, Dongmei Fu, and Jianlong Fu. "Pixel-bert: Aligning image pixels with text by deep multi-modal transformers". In: *arXiv preprint arXiv:2004.00849* (2020) (cit. on p. 65).

[90] Drew A Hudson and Christopher D Manning. "Gqa: A new dataset for real-world visual reasoning and compositional question answering". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 6700–6709 (cit. on pp. 45, 46, 56, 58, 65).

[91] Sarthak Jain and Byron C Wallace. "Attention is not Explanation". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies NAACL-HLT*. Ed. by Jill Burstein, Christy Doran, and Thamar Solorio. Association for Computational Linguistics, 2019, pp. 3543–3556. URL: https://doi.org/10.18653/v1/n19-1357 (cit. on p. 33).

[92] Stephen James, Paul Wohlhart, Mrinal Kalakrishnan, Dmitry Kalashnikov, Alex Irpan, Julian Ibarz, Sergey Levine, Raia Hadsell, and Konstantinos Bousmalis. "Sim-to-Real via Sim-to-Sim: Data-efficient Robotic Grasping via Randomized-to-Canonical Adaptation Networks". In: *Computer Vision and Pattern Recognition (CVPR)*. 2019 (cit. on p. 101).

[93] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. "Highly accurate protein structure prediction with AlphaFold". In: *Nature 2021 596:7873* 596.7873 (July 2021), pp. 583–589. URL: https://www.nature.com/articles/s41586-021-03819-2 (cit. on p. 2).

[94] Niels Justesen, Philip Bontrager, Julian Togelius, and Sebastian Risi. "Deep learning for video game playing". In: *IEEE Transactions on Games* (2019) (cit. on p. 75).

[95] Abhishek Kadian, Joanne Truong, Aaron Gokaslan, Alexander Clegg, Erik Wijmans, Stefan Lee, Manolis Savva, Sonia Chernova, and Dhruv Batra. "Sim2Real predictivity: Does evaluation in simulation predict real-world performance?" In: *IEEE Robotics and Automation Letters* 5.4 (2020), pp. 6670–6677 (cit. on p. 101).

[96] Daniel Kahneman. *Thinking, Fast and Slow.* 2013 (cit. on p. 122).

[97] M Kahng, P Y Andrews, A Kalro, and Polo Chau DH. "ACTIVIS: Visual Exploration of Industry-Scale Deep Neural Network Models." In: *IEEE Transactions on Visualization and Computer Graphics* 24.1 (2017), pp. 88–97 (cit. on p. 31).

[98] Andrej Karpathy. "[CVPR'21 WAD] Keynote". In: *Workshop on Autonomous Driving, IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021. URL: https://www.youtube.com/watch?v=g6bOwQdCJrc (cit. on p. 125).

[99] Andrej Karpathy, Justin Johnson, and Li Fei-Fei. "Visualizing and understanding recurrent networks". In: *arXiv preprint arXiv:1506.02078* (2015) (cit. on p. 79).

[100] Andrej Karpathy, Justin Johnson, and Li Fei-Fei. "Visualizing and Understanding Recurrent Networks". In: *Workshop track-ICLR*. 2016 (cit. on p. 21).

[101] Daniel Keim, Gennady Andrienko, Jean-Daniel Fekete, Carsten Görg, Jörn Kohlhammer, and Guy Melançon. "Visual Analytics: Definition, Process, and Challenges". In: *Information Visualization*. Springer Berlin Heidelberg, 2008, pp. 154–175 (cit. on pp. 10, 80).

[102] Daniel A Keim. "Designing Pixel-Oriented Visualization Techniques: Theory and Applications". In: *IEEE Transactions on Visualization and Computer Graphics* 6.1 (Jan. 2000), pp. 59–78. URL: http://dx.doi.org/10.1109/2945.841121 (cit. on p. 94).

[103] Michał Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaśkowski. "ViZDoom: A Doom-based AI Research Platform for Visual Reinforcement Learning". In: *IEEE Conference on Computational Intelligence and Games*. IEEE, Sept. 2016, pp. 341–348 (cit. on pp. 73, 75).

[104] Alex Kendall, Matthew Grimes, and Roberto Cipolla. "Posenet: A convolutional network for real-time 6-dof camera relocalization". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 2938–2946 (cit. on p. 100).

[105] Peter Kerpedjiev, Nezar Abdennur, Fritz Lekschas, Chuck McCallum, Kasper Dinkla, Hendrik Strobelt, Jacob M Luber, Scott B Ouellette, Alaleh Azhir, Nikhil Kumar, Jeewon Hwang, Soohyun Lee, Burak H Alver, Hanspeter Pfister, Leonid A Mirny, Peter J Park, and Nils Gehlenborg. "HiGlass: web-based visual exploration and analysis of genome interaction maps". In: *Genome Biology* 19.1 (Aug. 2018), p. 125. URL: https://doi.org/10.1186/s13059-018-1486-1 (cit. on p. 94).

[106] C Kervadec, T Jaunet, G Antipov, M Baccouche, R Vuillemot, and C Wolf. "How Transferable are Reasoning Patterns in VQA?" In: *CVPR*. 2021 (cit. on pp. 41, 51).

[107] Corentin Kervadec, Grigory Antipov, Moez Baccouche, and Christian Wolf. "Roses Are Red, Violets Are Blue... but Should Vqa Expect Them To?" In: *IEEE Conference on Computer Vision and Pattern Recognition* (2021) (cit. on pp. 40, 47, 48, 54, 60, 64).

[108] Corentin Kervadec, Grigory Antipov, Moez Baccouche, and Christian Wolf. "Weak Supervision helps Emergence of Word-Object Alignment and improves Vision-Language Tasks". In: *European Conference on Artificial Intelligence*. 2019 (cit. on p. 48).

[109] Corentin Kervadec, Christian Wolf, Grigory Antipov, Moez Baccouche, and Madiha Nadri. "Supervising the Transfer of Reasoning Patterns in VQA". In: *arXiv preprint arXiv:2106.05597* (2021) (cit. on p. 64).

[110] Sana Ullah Khan, Naveed Islam, Zahoor Jan, Ikram Ud Din, and Joel J.P.C. Rodrigues. "A novel deep learning based framework for the detection and classification of breast cancer using transfer learning". In: *Pattern Recognition Letters* 125 (July 2019), pp. 1–6 (cit. on pp. 2, 34).

[111] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, and others. "Visual genome: Connecting language and vision using crowdsourced dense image annotations". In: *International journal of computer vision* 123.1 (2017), pp. 32–73 (cit. on p. 45).

[112] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in neural information processing systems (NIPS)*. 2012. URL: http://code.google.com/p/cuda-convnet/ (cit. on p. 2).

[113] Carmen Lacave and Francisco J Díez. "A review of explanation methods for Bayesian networks". In: *The Knowledge Engineering Review* 17.2 (2002), pp. 107–127 (cit. on p. 13).

[114] Guillaume Lample, Devendra Singh Chaplot, Devendra Singh Chaplot, Devendra Singh Chaplot, and Devendra Singh Chaplot. "Playing FPS games with deep reinforcement learning". In: *Thirty-First AAAI Conference on Artificial Intelligence*. 2017 (cit. on p. 93).

[115] Sebastian Lapuschkin, Stephan Wäldchen, Alexander Binder, Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. "Unmasking Clever Hans predictors and assessing what machines really learn". In: *Nature Communications 2019 10:1* 10.1 (Mar. 2019), pp. 1–8. URL: https://www.nature.com/articles/s41467-019-08987-4 (cit. on p. 29).

[116] Y LeCun, B Boser, JS Denker, D Henderson Neural ..., and undefined 1989. "Backpropagation applied to handwritten zip code recognition". In: *Neural computation* (1989). URL: https://ieeexplore.ieee.org/abstract/document/6795724/ (cit. on pp. 2, 18).

[117] Joel Lehman, Jeff Clune, Dusan Misevic, Christoph Adami, Lee Altenberg, Julie Beaulieu, Peter J Bentley, Samuel Bernard, Guillaume Beslon, David M Bryson, and others. "The surprising creativity of digital evolution: A collection of anecdotes from the evolutionary computation and artificial life research communities". In: *Artificial life* 26.2 (2020), pp. 274–306 (cit. on pp. 4, 72).

[118] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. "End-to-End Training of Deep Visuomotor Policies". In: *The Journal of Machine Learning Research* 17.1 (2016) (cit. on p. 75).

[119] Alexander Lex, Nils Gehlenborg, Hendrik Strobelt, Romain Vuillemot, and Hanspeter Pfister. "UpSet: visualization of intersecting sets". In: *IEEE transactions on visualization and computer graphics* 20.12 (2014), pp. 1983–1992 (cit. on p. 93).

[120] Bo Li, Tara N Sainath, Arun Narayanan, Joe Caroselli, Michiel Bacchiani, Ananya Misra, Izhak Shafran, Hasim Sak, Golan Punduk, Kean Chin, Khe Chai Sim, Ron J Weiss, Kevin W Wilson, Ehsan Variani, Chanwoo Kim, Olivier Siohan, Mitchel Weintraub, Erik Mcdermott, Rick Rose, and Matt Shannon. "Acoustic Modeling for Google Home". In: *Interspeech*. 2017 (cit. on p. 125).

[121] Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. "What Does BERT with Vision Look At?" In: *ACL (short)*. 2020 (cit. on p. 46).

[122] Mingwei Li, Zhenge Zhao, and Carlos Scheidegger. "Visualizing Neural Networks with the Grand Tour". In: *Distill* 5.3 (Mar. 2020), e25. URL: https://distill.pub/2020/grand-tour (cit. on pp. 26, 27).

[123] Xiujun Li, Xi Yin, Chunyuan Li, Pengchuan Zhang, Xiaowei Hu, Lei Zhang, Lijuan Wang, Houdong Hu, Li Dong, Furu Wei, and others. "Oscar: Object-semantics aligned pre-training for vision-language tasks". In: *European Conference on Computer Vision*. Springer, 2020, pp. 121–137 (cit. on p. 45).

[124] Adam Light and Patrick J Bartlein. "The end of the rainbow? Color schemes for improved data graphics". In: *Eos, Transactions American Geophysical Union* 85.40 (2004), pp. 385–391. URL: www.ColorBrewer.org (cit. on p. 82).

[125] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. "Microsoft coco: Common objects in context". In: *European conference on computer vision*. Springer, 2014, pp. 740–755 (cit. on p. 45).

[126] Zachary C Lipton. "The Mythos of Model Interpretability: In machine learning, the concept of interpretability is both important and slippery." In: *Queue* 16.3 (2018), pp. 31–57 (cit. on pp. 13, 26, 28, 46, 72).

[127] Mengchen Liu, Shixia Liu, Hang Su, Kelei Cao, and Jun Zhu. "Analyzing the noise robustness of deep neural networks". In: *2018 IEEE Conference on Visual Analytics Science and Technology (VAST)*. IEEE, 2018, pp. 60–71 (cit. on p. 97).

[128] Mengchen Liu, Jiaxin Shi, Zhen Li, Chongxuan Li, Jun Zhu, and Shixia Liu. "Towards better analysis of deep convolutional neural networks". In: *IEEE transactions on visualization and computer graphics* 23.1 (2016), pp. 91–100 (cit. on p. 30).

[129] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. "Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks". In: *Advances in Neural Information Processing Systems*. 2019, pp. 13–23 (cit. on pp. 40, 45).

[130] Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. "Hierarchical question-image co-attention for visual question answering". In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. 2016, pp. 289–297 (cit. on p. 59).

[131] Y Ma, A Fan, J He, A R Nelakurthi, and R Maciejewski. "A Visual Analytics Framework for Explaining and Diagnosing Transfer Learning Processes." In: *IEEE Transactions on Visualization and Computer Graphics* (2020) (cit. on p. 101).

[132] J Macdonald, S Wäldchen, S Hauch arXiv preprint arXiv ..., and undefined 2019. "A rate-distortion framework for explaining neural network decisions". In: *arxiv.org* (). URL: https://arxiv.org/abs/1905.11092 (cit. on p. 71).

[133] Varun Manjunatha, Nirat Saini, and Larry S Davis. "Explicit bias discovery in visual question answering models". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 9562–9571 (cit. on pp. 40, 47).

[134] Warren S. McCulloch and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics 1943 5:4* 5.4 (Dec. 1943), pp. 115–133. URL: https://link.springer.com/article/10.1007/BF02478259 (cit. on p. 2).

[135] Leland McInnes, John Healy, and James Melville. "Umap: Uniform manifold approximation and projection for dimension reduction". In: *arXiv preprint arXiv:1802.03426* (2018) (cit. on pp. 25, 98, 105).

[136] Bhairav Mehta, Manfred Diaz, Florian Golemo, Christopher J Pal, and Liam Paull. "Active domain randomization". In: *Conference on Robot Learning*. PMLR, 2020, pp. 1162–1176 (cit. on pp. 23, 24, 97).

[137] William van Melle, Edward H Shortliffe, and Bruce G Buchanan. "EMYCIN: A knowledge engineer's tool for constructing rule-based expert systems". In: *Rule-based expert systems: The MYCIN experiments of the Stanford Heuristic Programming Project* (1984), pp. 302–313 (cit. on pp. 1, 13).

[138] Sachit Menon, Alexandru Damian, Shijia Hu, Nikhil Ravi, and Cynthia Rudin. "Pulse: Self-supervised photo upsampling via latent space exploration of generative models". In: *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*. 2020, pp. 2437–2445 (cit. on pp. 3, 4).

[139] Anton Mikhailov. *Turbo, An Improved Rainbow Colormap for Visualization*. 2019. URL: https://ai.googleblog.com/2019/08/turbo-improved-rainbow-colormap-for.html (cit. on p. 105).

[140] Tim Miller. "Explanation in artificial intelligence: Insights from the social sciences". In: *Artificial Intelligence* 267 (Feb. 2019), pp. 1–38 (cit. on p. 16).

[141] Yao Ming, Shaozu Cao, Ruixiang Zhang, Zhen Li, Yuanzhe Chen, Yangqiu Song, and Huamin Qu. "Understanding Hidden Memories of Recurrent Neural Networks". In: *IEEE Conference on Visual Analytics Science and Technology (VAST)* (Oct. 2017), pp. 13–24 (cit. on p. 32).

[142] Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andrew J Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, others, Dharshan Kumaran, Raia Hadsell, others, Dharshan Kumaran, and Raia Hadsell. "Learning to Navigate in Complex Environments". In: *5th International Conference on Learning Representations, {ICLR} 2017*. 2017 (cit. on pp. 75–77).

[143] Brent Mittelstadt, Chris Russell, and Sandra Wachter. "Explaining explanations in AI". In: *FAT* 2019 - Proceedings of the 2019 Conference on Fairness, Accountability, and Transparency*. Association for Computing Machinery, Inc, Jan. 2019, pp. 279–288 (cit. on p. 16).

[144] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. "Playing Atari with Deep Reinforcement Learning". In: *arXiv:1312.5602 [cs]* (Dec. 2013) (cit. on p. 75).

[145] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. "Human-level control through deep reinforcement learning". In: *Nature* 518.7540 (2015) (cit. on pp. 2, 71, 75, 76).

[146] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Tim Harley, Timothy P Lillicrap, David Silver, and Koray Kavukcuoglu. *Asynchronous Methods for Deep Reinforcement Learning*. Tech. rep. 2016 (cit. on pp. 75, 77, 86).

[147] Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. "Methods for Interpreting and Understanding Deep Neural Networks". In: *Digital Signal Processing* (2017). URL: https://arxiv.org/pdf/1706.07979.pdf (cit. on p. 16).

[148] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. "Deepfool: a simple and accurate method to fool deep neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2574–2582 (cit. on p. 97).

[149] A Mordvintsev, C Olah, and M Tyka. "Inceptionism: Going deeper into neural networks". In: (2015). URL: https://research.google/pubs/pub45507/ (cit. on p. 23).

[150] Adithyavairavan Murali, Tao Chen, Kalyan Vasudev Alwala, Dhiraj Gandhi, and Lerrel Pinto Saurabh Gupta Abhinav Gupta. "PyRobot: An Open-source Robotics Framework for Research and Benchmarking". In: *arXiv preprint arXiv:1906.08236* (). URL: https://www.pyrobot.org (cit. on pp. 99, 100).

[151] Anh Nguyen, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, and Jeff Clune. *Synthesizing the preferred inputs for neurons in neural networks via deep generator networks*. 2016 (cit. on p. 24).

[152] Andrew P. Norton and Yanjun Qi. "Adversarial-Playground: A visualization suite showing how adversarial examples fool deep learning". In: *2017 IEEE Symposium on Visualization for Cyber Security, VizSec 2017* 2017-October (Oct. 2017), pp. 1–4 (cit. on p. 34).

[153] Fabian Offert. ""I know it when I see it". Visualization and Intuitive Interpretability". In: *Proceedings of NIPS Symposium on Interpretable Machine Learning*. Nov. 2017. URL: https://arxiv.org/abs/1711.08042v2 (cit. on p. 6).

[154] Junhyuk Oh, Valliappa Chockalingam, Satinder Singh, and Honglak Lee. "Control of Memory, Active Perception, and Action in Minecraft". In: (May 2016) (cit. on p. 75).

[155] Chris Olah and Shan Carter. "Attention and Augmented Recurrent Neural Networks". In: *Distill* (2016). URL: http://distill.pub/2016/augmented-rnns (cit. on p. 22).

[156] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. "Feature Visualization". In: *Distill* 2.11 (Nov. 2017), e7. URL: https://distill.pub/2017/feature-visualization (cit. on p. 23).

[157] Chris Olah, Arvind Satyanarayan, Ian Johnson, Shan Carter, Ludwig Schubert, Katherine Ye, and Alexander Mordvintsev. "The Building Blocks of Interpretability". In: *Distill* (2018) (cit. on p. 72).

[158] *OpenAI Microscope*. URL: https://microscope.openai.com/models (cit. on p. 31).

[159] E Parisotto and R Salakhutdinov. "Neural Map: Structured Memory for Deep Reinforcement Learning". In: *ICLR* (2018) (cit. on p. 71).

[160] Parliament and C. of the European Union. *General data protection regulation*. 2016 (cit. on p. 3).

[161] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. "Automatic differentiation in PyTorch". In: (2017) (cit. on p. 84).

[162] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035 (cit. on pp. 58, 104).

[163] Adam Pearce. *Hidden Bias*. 2020 (cit. on p. 34).

[164] Adam Pearce. *Measuring Fairness*. 2020 (cit. on p. 34).

[165] Aayush Prakash, Shaad Boochoon, Mark Brophy, David Acuna, Eric Cameracci, Gavriel State, Omer Shapira, and Stan Birchfield. "Structured domain randomization: Bridging the reality gap by context-aware synthetic data". In: *Proceedings - IEEE International Conference on Robotics and Automation*. Vol. 2019-May. Institute of Electrical and Electronics Engineers Inc., May 2019, pp. 7249–7255 (cit. on p. 101).

[166] Marcelo O.R. Prates, Pedro H. Avelar, and Luís C. Lamb. "Assessing gender bias in machine translation: a case study with Google Translate". In: *Neural Computing and Applications* 32.10 (May 2020), pp. 6363–6381 (cit. on p. 3).

[167] Thomas Preusse. *Interactive Convolutional Neural Network*. URL: https://observablehq.com/@tpreusse/interactive-convolutional-neural-network (cit. on p. 34).

[168] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. "Learning Transferable Visual Models From Natural Language Supervision". In: *ArXiv 2103.00020* (2021). URL: https://github.com/OpenAI/CLIP. (cit. on p. 4).

[169] Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Lukas Gruber, Markus Holzleitner, Milena Pavlović, Geir Kjetil Sandve, Victor Greiff, and others. "Hopfield networks is all you need". In: *arXiv preprint arXiv:2008.02217* (2020) (cit. on pp. 54, 55).

[170] Paulo E Rauber, Samuel G Fadel, Alexandre X Falcao, and Alexandru C Telea. "Visualizing the hidden activity of artificial neural networks". In: *IEEE transactions on visualization and computer graphics* 23.1 (2016), pp. 101–110 (cit. on p. 27).

[171] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *Advances in Neural Information Processing Systems*. Ed. by C Cortes, N D Lawrence, D D Lee, M Sugiyama, and R Garnett. 2015, pp. 91–99 (cit. on p. 49).

[172] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should i trust you?: Explaining the predictions of any classifier". In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 2016, pp. 1135–1144 (cit. on pp. 5, 14, 28, 29, 72).

[173] Samuel Ritter, David G T Barrett, Adam Santoro, and Matt M Botvinick. "Cognitive Psychology for Deep Neural Networks: A Shape Bias Case Study". In: *ICML'17: Proceedings of the 34th International Conference on Machine Learning*. 2017 (cit. on p. 123).

[174] R Roscher, B Bohn, MF Duarte, J Garcke Ieee Access, and undefined 2020. "Explainable machine learning for scientific insights and discoveries". In: *ieee Access* (2020). URL: https://ieeexplore.ieee.org/abstract/document/9007737/ (cit. on p. 15).

[175] Cynthia Rudin. "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead". In: *Nature Machine Intelligence 2019 1:5* 1.5 (May 2019), pp. 206–215. URL: https://www.nature.com/articles/s42256-019-0048-x (cit. on p. 15).

[176] Alexander M Rush, Sumit Chopra, and Jason Weston. "A Neural Attention Model for Sentence Summarization". In: *Conference on Empirical Methods in Natural Language*. Association for Computational Linguistics, 2015 (cit. on p. 22).

[177] Andrei A Rusu, Matej Večerík, Thomas Rothörl, Nicolas Heess, Razvan Pascanu, and Raia Hadsell. "Sim-to-real robot learning from pixels with progressive nets". In: *Conference on Robot Learning*. PMLR, 2017, pp. 262–270 (cit. on p. 101).

[178] Sergio Salomón, Cristina Tîrnăucă, Sergio Salomón, and Cristina Tîrnăucă. "Human Activity Recognition through Weighted Finite Automata". In: *Proceedings* 2.19 (Oct. 2018), p. 1263 (cit. on p. 92).

[179] Giovanni Sartor. *The impact of the General Data Protection Regulation (GDPR) on artificial intelligence | Think Tank | European Parliament*. URL: https://www.europarl.europa.eu/thinktank/en/document/EPRS_STU(2020)641530 (cit. on p. 3).

[180] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. *Habitat: A Platform for Embodied AI Research*. Apr. 2019 (cit. on pp. 94, 99, 101).

[181] Udo Schlegel, Hiba Arnout, Mennatallah El-Assady, Daniela Oelke, and Daniel A. Keim. "Towards a rigorous evaluation of XAI methods on time series". In: *Proceedings - 2019 International Conference on Computer Vision Workshop, ICCVW 2019* (Oct. 2019), pp. 4197–4201 (cit. on p. 123).

[182] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. "Grad-cam: Visual explanations from deep networks via gradient-based localization". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 618–626 (cit. on pp. 23–25, 107).

[183] Ramprasaath R Selvaraju, Stefan Lee, Yilin Shen, Hongxia Jin, Shalini Ghosh, Larry Heck, Dhruv Batra, and Devi Parikh. "Taking a hint: Leveraging explanations to make vision and language models more grounded". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 2591–2600 (cit. on p. 79).

[184] Ben Shneiderman. *The eyes have it: A task by data type taxonomy for information visualizations* (cit. on p. 53).

[185] Robik Shrestha, Kushal Kafle, and Christopher Kanan. "A negative case analysis of visual grounding methods for VQA". In: *arXiv preprint arXiv:2004.05704* (2020) (cit. on p. 47).

[186] D Silver, J Schrittwieser, K Simonyan, I Antonoglou, A Huang, A Guez, T Hubert, L Baker, M Lai, A Bolton, Y Chen, T Lillicrap, F Hui, L Sifre, G van den Driessche, T Graepel, and D Hassabis. "Mastering the game of Go without human knowledge". In: *Nature* (2017) (cit. on pp. 2, 71, 75).

[187] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. "Deep inside convolutional networks: Visualising image classification models and saliency maps". In: *2nd International Conference on Learning Representations, ICLR - Workshop Track Proceedings*. International Conference on Learning Representations, ICLR, 2014 (cit. on p. 22).

[188] Daniel Smilkov, Shan Carter, D. Sculley, Fernanda B. Viégas, and Martin Wattenberg. "Direct-Manipulation Visualization of Deep Networks". In: Aug. 2017. URL: https://arxiv.org/abs/1708.03788v1 (cit. on p. 34).

[189] Kacper Sokol and Peter Flach. "One Explanation Does Not Fit All". In: *KI - Künstliche Intelligenz 2020 34:2* 34.2 (Feb. 2020), pp. 235–250. URL: https://link-springer-com.docelec.insa-lyon.fr/article/10.1007/s13218-020-00637-y (cit. on pp. 6, 29).

[190] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. "Striving for Simplicity: The All Convolutional Net". In: (Dec. 2014) (cit. on pp. 19, 22, 23, 81).

[191] Stefan Wojcik Emma Remy and Chris Baronavski. "How Does a Computer "See" Gender?" In: *Proceedings of the Workshop on Visualization for AI explainability (VISxAI)*. 2020. URL: https://www.pewresearch.org/interactives/how-does-a-computer-see-gender/ (cit. on p. 35).

[192] Gregor Stiglic, Primoz Kocbek, Nino Fijacko, Marinka Zitnik, Katrien Verbert, and Leona Cilar. "Interpretability of machine learning-based prediction models in healthcare". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 10.5 (2020), e1379 (cit. on p. 27).

[193] H. Strobelt, S. Gehrmann, H. Pfister, and A. M. Rush. "LSTMVis: A Tool for Visual Analysis of Hidden State Dynamics in Recurrent Neural Networks". In: *IEEE Transactions on Visualization and Computer Graphics* 24.1 (Jan. 2017), pp. 667–676 (cit. on pp. 32, 72, 92, 98).

[194] Hendrik Strobelt, Sebastian Gehrmann, Michael Behrisch, Adam Perer, Hanspeter Pfister, and Alexander M Rush. "Seq2seq-vis: A visual debugging tool for sequence-to-sequence models". In: *IEEE transactions on visualization and computer graphics* 25.1 (2018), pp. 353–363 (cit. on pp. 32, 55).

[195] Pascal Sturmfels, Scott Lundberg, and Su-In Lee. "Visualizing the Impact of Feature Attribution Baselines". In: *Distill* 5.1 (Jan. 2020), e22. URL: https://distill.pub/2020/attribution-baselines (cit. on p. 23).

[196] Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. "Vl-bert: Pre-training of generic visual-linguistic representations". In: *arXiv preprint arXiv:1908.08530* (2019) (cit. on pp. 45, 65).

[197] Alessandro Suglia, Qiaozi Gao, Jesse Thomason, Govind Thattai, and Gaurav Sukhatme. "Embodied bert: A transformer model for embodied, language-guided visual task completion". In: *arXiv preprint arXiv:2108.04927* (2021) (cit. on p. 125).

[198] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurélien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. "Scalability in Perception for Autonomous Driving: Waymo Open Dataset". In: *In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020. URL: http://www.waymo.com/open. (cit. on p. 125).

[199] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. "Axiomatic Attribution for Deep Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. PMLR, July 2017, pp. 3319–3328. URL: https://proceedings.mlr.press/v70/sundararajan17a.html (cit. on p. 23).

[200] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. 2018 (cit. on p. 71).

[201] Róbert Szabó, Dániel Katona, Márton Csillag, Adrián Csiszárik, and Dániel Varga. "Visualizing Transfer Learning". In: *arXiv preprint arXiv:2007.07628* (2020) (cit. on p. 101).

[202] Hao Hao Tan and Mohit Bansal. "LXMERT: Learning Cross-Modality Encoder Representations from Transformers". In: *Empirical Methods in Natural Language Processing*. 2019 (cit. on pp. 38, 40, 43–45, 48).

[203] Sarah Tan. "Interpretable Approaches to Detect Bias in Black-Box Models". In: *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*. AIES '18. New York, NY, USA: Association for Computing Machinery, 2018, pp. 382–383. URL: https://doi.org/10.1145/3278721.3278802 (cit. on p. 27).

[204] Anthony Tang, Saul Greenberg, and Sidney Fels. "Exploring video streams using slit-tear visualizations". In: *CHI'09 Extended Abstracts on Human Factors in Computing Systems*. ACM, 2009, pp. 3509–3510 (cit. on p. 93).

[205] Damien Teney, Kushal Kafle, Robik Shrestha, Ehsan Abbasnejad, Christopher Kanan, and Anton van den Hengel. "On the Value of Out-of-Distribution Testing: An Example of Goodhart's Law". In: *arXiv preprint arXiv:2005.09241* (2020) (cit. on p. 47).

[206] Ian Tenney, James Wexler, Jasmijn Bastings, Tolga Bolukbasi, Andy Coenen, Sebastian Gehrmann, Ellen Jiang, Mahima Pushkarna, Carey Radebaugh, Emily Reif, and Ann Yuan. *The Language Interpretability Tool: Extensible, Interactive Visualizations and Analysis for NLP Models*. 2020. URL: https://www.aclweb.org/anthology/2020.emnlp-demos.15 (cit. on p. 33).

[207] *TensorBoard | TensorFlow*. URL: https://www.tensorflow.org/tensorboard?hl=fr (cit. on p. 120).

[208] *TensorSpace.js*. URL: https://tensorspace.org/ (cit. on p. 34).

[209] *Tesla Autopilot Faces U.S. Inquiry After Series of Crashes - The New York Times*. URL: https://www.nytimes.com/2021/08/16/business/tesla-autopilot-nhtsa.html (cit. on p. 3).

[210] *The 16 Alexa-related papers at this year's Interspeech - Amazon Science*. URL: https://www.amazon.science/blog/the-16-alexa-related-papers-at-this-years-interspeech (cit. on p. 125).

[211] Tijmen Tieleman and Geoffrey Hinton. "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude". In: *COURSERA: Neural networks for machine learning* 4.2 (2012), pp. 26–31 (cit. on p. 77).

[212] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. "Domain randomization for transferring deep neural networks from simulation to the real world". In: *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2017, pp. 23–30 (cit. on pp. 97, 101).

[213] Jonathan Tremblay, Aayush Prakash, David Acuna, Mark Brophy, Varun Jampani, Cem Anil, Thang To, Eric Cameracci, Shaad Boochoon, and Stan Birchfield. "Training Deep Networks With Synthetic Data: Bridging the Reality Gap by Domain Randomization". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. June 2018 (cit. on p. 101).

[214] Eric Tzeng, Coline Devin, Judy Hoffman, Chelsea Finn, Pieter Abbeel, Sergey Levine, Kate Saenko, and Trevor Darrell. "Adapting deep visuomotor representations with weak pairwise constraints". In: *Algorithmic Foundations of Robotics XII*. Springer, 2020, pp. 688–703 (cit. on p. 101).

[215] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. "Adversarial discriminative domain adaptation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 7167–7176 (cit. on pp. 97, 101).

[216] F-Y Tzeng and K-L Ma. "Opening the black box-data driven visualization of neural networks". In: *VIS 05. IEEE Visualization, 2005.* IEEE, 2005, pp. 383–390 (cit. on p. 14).

[217] Laurens Van Der Maaten and Geoffrey Hinton. "Visualizing Data using t-SNE". In: *Journal of Machine Learning Research* 9 (2008), pp. 2579–2605 (cit. on pp. 25, 76, 81).

[218] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Kaiser, and Illia Polosukhin. "Attention is all you need". In: *Advances in neural information processing systems*. 2017, pp. 5998–6008 (cit. on pp. 22, 33, 34, 40, 42, 43).

[219] Jesse Vig. "A multiscale visualization of attention in the transformer model". In: *arXiv preprint arXiv:1906.05714* (2019) (cit. on pp. 33, 55).

[220] Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander S. Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L. Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. "Grandmaster level in StarCraft II using multi-agent reinforcement learning". In: *Nature 2019 575:7782* 575.7782 (Oct. 2019), pp. 350–354. URL: https://www.nature.com/articles/s41586-019-1724-z (cit. on p. 2).

[221] Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. "Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019, pp. 5797–5808 (cit. on pp. 43, 64).

[222] Chelsea Voss, Nick Cammarata, Gabriel Goh, Michael Petrov, Ludwig Schubert, Ben Egan, Swee Kiat Lim, and Chris Olah. "Visualizing Weights". In: *Distill* 6.2 (Feb. 2021), e00024.007. URL: https://distill.pub/2020/circuits/visualizing-weights (cit. on p. 14).

[223] Emily Wall, Leslie M. Blaha, Lyndsey Franklin, and Alex Endert. "Warning, Bias May Occur: A Proposed Approach to Detecting Cognitive Bias in Interactive Visual Analytics". In: *2017 IEEE Conference on Visual Analytics Science and Technology, VAST 2017 - Proceedings* (Dec. 2018), pp. 104–115 (cit. on p. 123).

[224] Emily Wall, Arpit Narechania, Adam Coscia, Jamal Paden, and Alex Endert. "Left, Right, and Gender: Exploring Interaction Traces to Mitigate Human Biases". In: *IEEE Transactions on Visualization and Computer Graphics* 28.1 (Jan. 2022), pp. 966–975 (cit. on p. 123).

[225] Emily Wall, John Stasko, and Alex Endert. "Toward a Design Space for Mitigating Cognitive Bias in Vis". In: *2019 IEEE Visualization Conference, VIS 2019* (Oct. 2019), pp. 111–115 (cit. on p. 123).

[226] Junpeng Wang, Liang Gou, Han-Wei Shen, and Hao Yang. "Dqnviz: A visual analytics approach to understand deep q-networks". In: *IEEE transactions on visualization and computer graphics* 25.1 (2018), pp. 288–298 (cit. on pp. 76, 93).

[227] X Wang, R Girshick, A Gupta, and K He. "Non-local Neural Networks". In: *CVPR*. 2018 (cit. on p. 40).

[228] Zhiguang Wang and Jianbo Yang. "Diabetic retinopathy detection via deep convolutional networks for discriminative localization and visual explanation". In: *arXiv preprint arXiv:1703.10757* (2017) (cit. on p. 107).

[229] Zijie J Wang, Robert Turko, Omar Shaikh, Haekyu Park, Nilaksh Das, Fred Hohman, Minsuk Kahng, and Duen Horng Chau. "CNN Explainer: Learning Convolutional Neural Networks with Interactive Visualization". In: *IEEE Conference on Visual Analytics Science and Technology (VAST)* (2020) (cit. on p. 34).

[230] Sarah Wiegreffe and Yuval Pinter. "Attention is not not explanation". In: *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference* (2020), pp. 11–20. URL: http://bit. (cit. on p. 33).

[231] Kanit Wongsuphasawat, Daniel Smilkov, James Wexler, Jimbo Wilson, Man´ Mané, Doug Fritz, Dilip Krishnan, Fernanda B Ví, and Martin Wattenberg. "Visualizing Dataflow Graphs of Deep Learning Models in TensorFlow". In: *IEEE transactions on visualization and computer graphics* (2017) (cit. on pp. 14, 31, 120).

[232] Krist Wongsuphasawat, John Alexis Guerra Gómez, Catherine Plaisant, Taowei David Wang, Ben Shneiderman, and Meirav Taieb-Maimon. *LifeFlow: Visualizing an Overview of Event Sequences*. 2011 (cit. on p. 92).

[233] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, and others. "Google's neural machine translation system: Bridging the gap between human and machine translation". In: *arXiv preprint arXiv:1609.08144* (2016) (cit. on p. 42).

[234] B Yamauchi. "A frontier-based approach for autonomous exploration". In: *Symposium on Computational Intelligence in Robotics and Automation*. 1997 (cit. on p. 71).

[235] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. "How transferable are features in deep neural networks?" In: *NIPS*. 2014 (cit. on p. 51).

[236] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. "Understanding Neural Networks Through Deep Visualization". In: *arXiv:1506.06579 [cs]* (June 2015) (cit. on p. 30).

[237] Zhou Yu, Jun Yu, Yuhao Cui, Dacheng Tao, and Qi Tian. "Deep Modular Co-Attention Networks for Visual Question Answering". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2019 (cit. on pp. 40, 42).

[238] Tom Zahavy, Nir Ben Zrihem, Shie Mannor, Nir Ben-Zrihem, Shie Mannor, Nir Ben Zrihem, Shie Mannor, Nir Ben-Zrihem, Shie Mannor, Nir Ben Zrihem, and Shie Mannor. "Graying the black box: Understanding dqns". In: *International Conference on Machine Learning*. Feb. 2016, pp. 1899–1908 (cit. on p. 76).

[239] Matthew D Zeiler and Rob Fergus. "Visualizing and understanding convolutional networks". In: *European conference on computer vision*. Springer, 2014, pp. 818–833 (cit. on pp. 20, 21, 29, 31, 108, 119).

[240] Pengchuan Zhang, Xiujun Li, Xiaowei Hu, Jianwei Yang, Lei Zhang, Lijuan Wang, Yejin Choi, and Jianfeng Gao. "Vinvl: Revisiting visual representations in vision-language models". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 5579–5588 (cit. on p. 65).

[241] H Zhao, J Jia, and V Koltun. "Exploring Self-attention for Image Recognition". In: *CVPR*. 2020 (cit. on p. 40).

[242] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. "Learning deep features for discriminative localization". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2921–2929 (cit. on p. 79).

[243] Bolei Zhou, Aditya Khosla, Àgata Lapedriza, Aude Oliva, and Antonio Torralba. "Object Detectors Emerge in Deep Scene CNNs." In: *International Conference on Learning Representations (ICLR)*. Jan. 2015 (cit. on p. 19).

[244] Fengda Zhu, Linchao Zhu, and Yi Yang. "Sim-real joint reinforcement transfer for 3d indoor navigation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 11388–11397 (cit. on p. 101).

[245] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. "Target-driven Visual Navigation in Indoor Scenes using Deep Reinforcement Learning". In: *2017 IEEE international conference on robotics and automation (ICRA)*. 2017, pp. 3357–3364 (cit. on pp. 75, 125).