UNIVERSITY OF PADUA

MASTER THESIS IN CYBERSECURITY

Social honeypots on Instagram: A study on technologies and methodologies to automate them

Author:
Sara Bardi
Student ID 2015137

Supervisor:
Prof. Mauro Conti
Co-supervisors:
Luca Pajola
Pier Paolo Tricomi

SPRITZ - Security and Privacy Research Group
Department of Mathematics "Tullio Levi-Civita"

September 12, 2022

"In the middle of every difficulty lies opportunity."

Albert Einstein

Abstract

Online Social Networks (OSNs) have gained increasing popularity in recent years leading to very fast growth in terms of registered users. While OSNs are widely used for legitimate content sharing, their rapid growth has also led to the emergence of illegal activities (e.g. spamming, profile cloning, profile hijacking) that take advantage of their popularity. One tool used to detect these malicious activities is the social honeypot. In principle, social honeypots consist in honeypot profiles, for instance Facebook pages or Twitter accounts, which are able to attract users for further analysis. However, we are convinced that social honeypots can be seen not only as a cybersecurity countermeasure, but also as a flexible system that can be adopted for many different purposes. For instance, for customers profiling and products advertising, or for understanding social trends among people.

This thesis aims to make a first attempt toward better understanding of the methodologies and technologies to build automated social honeypots on Instagram. This approach has never been exploited before, in fact there is no previous work that proposed social honeypots on this social network and, furthermore, all the social honeypots presented in the literature are not automated. Hence, our experiment consists in 21 social honeypots, deployed on Instagram, whose management is completely automatic. To this end, we have implemented two post generation strategies: one involves simpler methods such as using stock images, the second is based on more complex processes by using the latest Machine Learning technologies. Each honeypot is equipped with an engagement plan that identify how it generates engagement with other users.

Our results show that automatic social honeypots on Instagram are possible and that they can be customized according to our needs. We have demonstrated that the post generation strategy based on Machine Learning is not the best choice yet and that a simple interaction with other users, by just liking or commenting their posts, is the option to be preferred. Thanks to these results, we are convinced that the work presented in this thesis can pave the way to further researches and solutions.

Acknowledgements

I would like to express my deepest gratitude to my professor and supervisor who gave me the opportunity to work on this project. His feedback and his advice have been extremely helpful in developing this thesis. Special thanks to my co-supervisors who really supported me from the beginning, I think we did a great job together.

I would like to dedicate this thesis to my mom who has always been by my side, believing in me and in my abilities. Thank you mom, for being the main point of reference in my life. I would like also to extend my sincere thanks to Riccardo, for teaching me to be strong in this world, and to Licia, for always telling me "you can do it" when I needed it most. I also thank Gabriele, with the hope of being able to have more exchanges of views in the near future.

Last but not least, I would like to mention the person who has always been my moral support, Cri. Thank you for the patience shown over these long years, for understanding me without needing to say a single word, for being the best part of me. Without you, this arduous journey would have been even more difficult to travel.

Contents

List of Figures

List of Tables

Dedicated to "Mimmi"
and my beloved D&D gamers.

Chapter 1

Introduction

Over the past few years, Online Social Networks (OSNs) have experienced an exponential growth, both in terms of size and popularity (Alexa, 2022). In fact, major OSNs, such as Facebook and Instagram, now have billions users and the rapid growth does not seem to want to stop (Karl, 2022, Richter, 2022). The purpose of social networking is to allow individuals to meet people with similar interests, keep in touch with them and even to reconnect with lost people (Nisrine et al., 2016). Moreover, users are able to directly share and publish pictures and video in an extremely fast and efficient way. However, there are also several threats and risks that we need to take into consideration when using social networks. Indeed, due to the significant amount of people that use OSN, more and more cyber criminals are developing a particular interest in using social networks as a basis for carrying out other malicious activities (Sheikhi, 2020).

When we think about OSN, the first problem that can be identified is the spreading of personal information without the awareness of users. In fact, personal data can be targeted by cyber criminals to perform illegal actions such as personal data theft, identity impersonation, illegal distribution of content, canvassing, damage of image and reputation (Nisrine et al., 2016). Among the types of attacks that are based on OSNs, we find (Joshi and Kuo, 2011): plain impersonation, profile cloning, profile hijacking, profile porting, secondary data collection, crawling and harvesting, etc.

One of the most studied malicious activity performed on OSNs is spamming (Hu, Tang, and Liu, 2014, Zhu et al., 2012, Murugan and Devi, 2018). Spamming can be defined as all those messages, which we may receive, that try to advertise some product or sharing pornographic or undesired content or even for phishing purposes. There is a widely used tool to detect spamming activity that is **Social Honeypot** (De Cristofaro et al., 2014, Yang, Zhang, and Gu, 2014,Zhang, Zhang, and Yuan, 2019). Social honeypots consist in honeypot profiles, for instance Facebook pages or Twitter accounts, that try to lure users and "trap" them for further analysis or actions. All the proposed solutions in literature use social honeypot to gather information about spammers and their activities for developing countermeasures capable of reducing the presence of these fake profiles.

However, we believe that social honeypots are not strictly linked to malicious activity, in particular with spammers, but that they could also be used for other purposes such as marketing strategies or social studies. For instance, they could be useful when a company or a brand needs to profile their audience to understand which needs their possible customers have. From a social perspective, they could help in understanding how society is evolving and which are the new trends among people. Hence, we aim to demonstrate that they should not only be considered as a cybersecurity countermeasure, but also as a versatile strategy that can be adopted in different sectors.

Furthermore, the main idea on which this thesis is based is that, not only social honeypots should be considered as a flexible tool, and thus not limiting their use to be spamming detectors, but they could be even more worthwhile if their functioning is totally automatic. Besides, making a social honeypots automatic may involve also Machine Learning techniques. Nowadays, Machine Learning and Artificial Intelligence in general, are widely adopted in a large number of sectors due to the improvements achieved during the last few years. They are able to automate most of the tasks that were usually performed by humans, making their use a natural choice in modern technologies.

In the case of social honeypot, until now Machine Learning has been used to develop a kind of observation system that observes the users' activity with the honeypot and tries to classify this activity as legitimate or malicious (Zhang, Zhang, and Yuan, 2019,Yang, Zhang, and Gu, 2014). However, the latest Machine Learning studies carried out in fields such as Computer Vision, Natural Language Processing and image generation, have shown that the results that can be obtained, not only in standard classification tasks, are truly encouraging (Ramesh et al., 2022, Brown et al., 2020). Hence, this work aims to propose a new approach for building social honeypots that automates all those activities that usually require human intervention, thanks also to the support of the latest Machine Learning technologies.

For the experiments of this work we have decided to use Instagram, rather than Tweeter or Facebook. Instagram is a social network particularly used for sharing photos and videos. However, this platform is becoming more and more exploited for other purposes, not only to simply sharing content. Currently, it is used also as a platform for marketing in which people who reach a considerable number of followers have used their accounts as a place for advertising (Sheikhi, 2020). Despite its popularity, we were not able to find researches that deployed social honeypots on Instagram. For this reason, another purpose of this work is to discover how much this unexplored platform is suitable for further investigations.

1.1 Contributions

The main contributions of this thesis are:

- We define a novel concept of social honeypot. In our opinion, it is a flexible tool that can be used in many different scenarios and sectors. One of the advantages of our social honeypots is that they can be adapted to any need without having to change their internal implementation.
- We propose two strategies to generate automatically Instagram posts and three engagement plans which identify how the social honeypot interact with other users. The post generation strategies involve both simplest methods, such as using stock images, and more complex methods based on Machine Learning technologies.
- We have created 21 social honeypots on Instagram, equipped with different post generation strategies and engagement plans, to understand the best approach for their management and provide guidelines on how to build effective social honeypots.

1.2 Outline

What a social honeypot is and the related works that have used this tool as a cybersecurity mechanism, is explained deeply in chapter 2. Moreover, in this chapter, it can be found a description of the latest Machine Learning technologies with particularly emphasis on object detection, text generation and image generation tasks which are essentially in this project. This background is needed to understand the methodology, explained in chapter 3, which characterizes this thesis. Besides, since social honeypots may deals with private personal information and deception mechanisms, some ethical considerations are made in chapter 3. The practical implementation is shown instead in chapter 4. The purpose of this chapter is to give an insight of how our social honeypots have been developed and allow other research teams to conduct further experiments in order to improve our results described in chapter 5. The last chapter of this thesis, chapter 6, makes some considerations on our new approach and highlights some future works that can bring considerable improvements to our solution.

Chapter 2

Background

Before presenting the methodology and the subsequent implementation of this work, it is useful to highlight some basic notions to better understand our choices. What a *honeypot* is, in the field of cybersecurity, and what is meant by *social honeypot*, are concepts that will be explained in this chapter, in section 2.1 and section 2.2 respectively. Furthermore, as a last consideration, we would like to highlight some features of the machine learning technologies used in this project, with the aim of giving an overview of the entire process.

2.1 Honeypot

A significant security problem for networked systems is trespass of unauthorized user or software, also known as intruders. User trespass can take the form of unauthorized logon to a machine or acquisition of privileges or even performance of actions beyond those that have been authorized (Stallings et al., 2012). A security service that monitors and analyzes system events, in real-time or near real-time, with the purpose of protecting against unauthorized access attempts is called *Intrusion Detection*. Intrusion detection can be implemented with different tools such as Intrusion Detection System (IDS) or Intrusion Prevention System (IPS). Both of them are based on the assumption that the behavior of the intruder is different from that of a legitimate user and thus it can be quantified.

A further tool useful to detect unauthorized access is the *honeypot*. Honeypots are decoy systems that are designed to lure potential attackers away from critical systems (Stallings et al., 2012). The idea is to divert attackers from accessing the sensitive part of the network and encourage them to stay in the honeypot long enough so that we have time to gather information about their activities and to respond appropriately to the attack. In fact, the attacker is made to think that the attack to the honeypot is successful so that administrators have time to take action and log events without exposing critical systems. The reason why honeypots work is that there is no valid motivation for outsiders to interact with them and therefore any attempt to communicate with the honeypot is most likely an attack.

Honeypots are typically classified as follows:

- **Low interaction honeypot** consists of a software that emulates specific services or systems well enough to seem realistic but do not execute a full version of those services or systems.
- **High interaction honeypot** is a real system that implements a full operating system with its applications and services that are deployed when an attack is taking place, allowing the attacker to access them. It requires more resources with respect to the low interaction honeypot, but it is more realistic.

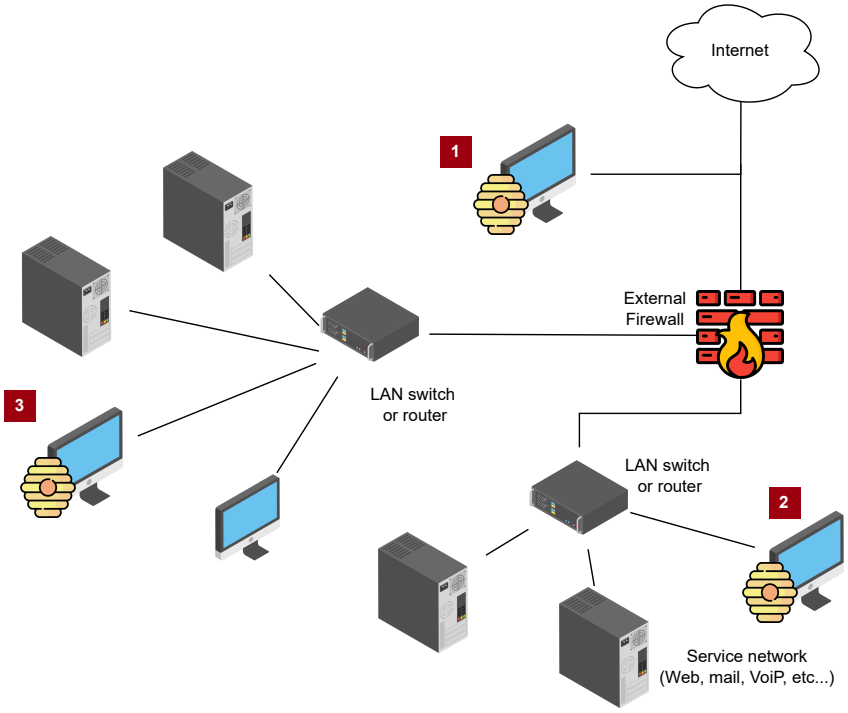


FIGURE 2.1: Honeypot Deployment.

Honeypots can be deployed in different locations and Figure 2.1 illustrates some possibilities. The location may depend on specific factors, such as the type of information we are interested in or the level of risk that can be tolerate. A honeypot outside the external firewall 1 is useful to track attempts to connect to IP addresses within the network boundaries. It does not increase the risk for internal network and it reduces the amount of traffic to the firewall thanks to its position. One disadvantage of this type of honeypot is that it has little or no capability to trap internal attackers. Another possibility is to place the honeypot in demilitarized zones (DMZ) 2. DMZs are separated subnets useful to connect hosts that provide services such as mail, Voice over Internet Protocol (VoIP) and web servers, while keeping the internal network isolated and safe from the external network. However, a honeypot at this location may not be able to catch relevant attacks because the DMZ zone is typically not fully accessible. That is, in addition to well-defined public services, no one else service should be available in this part of the network. This means that if an attacker tries to access the honeypot through a service that is not one of those well-defined, the firewall is going to block the traffic. We might think about letting the firewall allow traffic to the honeypot, but that would mean we are opening the firewall to a significant security risk. Finally, as last possible location, there is fully internal honeypot 3. In this case the honeypot can trap internal attackers and even misconfigured firewall. On the other hand, it must be carefully designed to completely trap the attacker because, if compromised, the honeypot could attack the other internal systems. Also, to continue trapping the attacker, we need to allow his attack traffic from the Internet to the honeypot and, as before, the firewall would be exposed to a high risk.

While there are advantages and disadvantages, the strategic use of honeypots may enhance the security of systems leaving them less vulnerable to cyber threats and exploits.

2.2 Social Honeypot

Today honeypots are not only used as security countermeasures to protect against network attacks, but also in various scenarios such as in Online Social Networks (OSNs). In this case, we can define them as *social honeypots*. Similarly to what a traditional honeypot does, social honeypots target online activities on social networks, typically through the deployment of honeypot profile (e.g., a Facebook page or Twitter account) to lure possible users and "trap" them. There might be several reasons why we want to lure users on social honeypots: for example to automatically collect evidence of illegal activity or, in a broader perspective, for marketing purposes. However, in most of the research that has been conducted so far, they are used to automatically detect spamming activity. Spam can be defined as all those unsolicited messages sent over a social network with specific purposes such that advertising, phishing, sharing undesired or pornographic content etc.

The first social honeypot on MySpace (Webb, Caverlee, and Pu, 2008) The first research work was proposed in 2008, where 51 honeypots were deployed on MySpace to provide the first characterization of social spammers and their behavior. All of these honeypot profiles were identical except for their geographic information: each of them had the same name, gender and birthday, they shared the same relationship status (single), body type and ethnicity. Bots constantly checked these profiles and, after receiving a new friend request, they collected the spam profile and stored a copy of it before rejecting the friend request. After four months, the results were analyzed and reported: the behavior of social spammers shows several recognizable temporal and geographic patterns that have specific characteristics. Thanks to these results, the authors were able to defines five categories of spam profiles. One of the most interesting features of these spam profiles is that the majority of them had an external link in the "About me" section. Following these links and after several redirection techniques, it was found out that all the URLs were pointing to a limited number of web pages, usually pornographic pages.

Social Honeypot on MySpace and Twitter (Lee, Caverlee, and Webb, 2010) After few years, the same authors proposed a new research study in 2010. Even in this case social honeypots were used to trap spammer, but they were deployed not only on MySpace but also on Twitter. Out of the five spam categories already defined on MySpace, five other categories of Twitter spammers have been identified. Moreover, in addition to the previous work, several other analysis were conducted to understand whether there were discernible spam signals in the harvest spam profiles that can be used to automatically differentiate spam profiles from legitimate profiles. As first step, machine learning classifiers were trained upon a dataset which contained both the collected spam profiles and randomly sampled legitimate profiles. The performance results of all the classifiers were successful, that is each classifier reached an accuracy greater than 98.4% for MySpace and greater than 82.7% for Twitter. Since these optimistic results, the authors tried other tests to find out if the classifiers can be effectively deployed over large collections of unknown profiles. While the accuracy decreased with an external MySpace dataset (Caverlee, 2008), the results obtained by using an already existing Twitter dataset were quite close to the previous reported performance.

Longer experiments and collaboration with Twitter (Stringhini, Kruegel, and Vigna, 2010) These two works have shown that social honeypots can be an effective strategy for detecting spammers on social networks, paving the way for new research activities. A useful research for the discussion of this thesis is the one that was presented a few months later, in December 2010. In this work, honeypots were distributed not only on Twitter and MySpace, but also on Facebook. Moreover, the research went on for a longer period, namely 12 months for honeypots on Facebook and 11 months for the ones on Twitter and MySpace. In this period, the authors collaborated with Twitter and correctly detected and deleted 15,857 spam profiles. The honeypot design was quite the same as the previous works: fake accounts were created and ran while scripts periodically connected to those accounts and checked their activities. Even in this case, honeypots were kind of passive in the sense that they do not send any friend requests or have any other activities but accepting received friend requests. To be precise, they received not only requests from spam profiles but also from legitimate profiles and, to discriminate between them, authors started to manually check all of them. During this analysis, it was found out that, as the previous works, spam bots share some common traits. An interesting result to be highlighted is that there are two kinds of spam bots: stealthy and greedy bots. Greedy ones include spam content in each message they send, while the stealthy bots send messages that look legitimate, and only once in a while inject a malicious message. This implies that greedy bots are easier to be detected than the stealthy ones. In addition to this analysis, new tests were performed to classify unseen users into spammers or legitimate users. The study was conducted by using machine learning techniques with datasets based on both the already trapped spammed profiles and randomly selected legitimate profiles. The tests produced an estimated false positive ratio of 2% and a false negative ratio of 1% on Facebook while an estimated false positive ratio of 2.5% and a false negative ratio of 3% on Twitter.

Understanding Facebook Like Fraud using social honeypots (De Cristofaro et al., 2014) A research which used social honeypots only on Facebook rather than Twitter is the one presented in 2014 . In this case, a comparative measurement study of the likes on Facebook pages was conducted to understand which are the different characteristics between likes obtained through popular like farms and the ones obtained through Facebook advertising campaigns. 13 honeypot pages were promoted using both methods and intentionally kept empty, thus they were passive honeypots. To be precisely, 5 of them were promoted by using Facebook ad campaigns and the remaining 8 were promoted by using popular like farms. After a month, garnered likes were analyzed based on likers' demographic, temporal and social characteristics to find out which are the differences between the two methods. As we can see, the aim of this research is slightly different from previous works: the authors are not interested in spamming in a general perspective but in identifying differences and similarities between two methods of obtaining likes, one legal and one illegal. Moreover, authors figured out that like farms have mainly two different behaviors: some farms seem to operate by bots and do not really try to hide the nature of their operations while other farms follow a much stealthier approach, aiming to mimic regular users' behavior. This finding resembles what has already been reported in previous works, further validating those results.

Long-Term study of spammers on Twitter with active social honeypots (Lee, Eoff, and Caverlee, 2011) As it is possible to notice, all the researches presented up to

now are based on "passive" social honeypots. They are static, fake accounts deployed mainly on MySpace and Twitter that do not do anything but receiving friend requests. Some of them even rejected these requests after saving the spam profile's information. However, if they have been able to get promising results with these passive honeypots, there is a chance that more active honeypots will perform better. In fact, one of the most cited work is the one proposed in 2011 in which active honeypots were used for tempting, profiling and filtering content polluters in social media. 60 honeypots were deployed on Twitter that resulted in the harvesting of 36,000 candidate content polluters. Honeypots differed from each other based on how often they post, the content and the type of their postings and their social network structure. Nevertheless, the social honeypots were intentionally designed to avoid interfering with activities of legitimate users. They only sent replay messages to each other and only followed other social honeypots accounts. Once a Twitter user contacted one of the social honeypot, information is passed to an Observation system that collects all information about the user's account and all the user's past tweets. A further analysis brought to identify four categories of content polluters, quite similar to the ones already identifies, along with other common characteristics among all the spam profiles. As the previous works, a classification framework was trained to discriminate between content polluters and legitimate profiles. The results was quite surprising: with Random Forest classifier they achieved 98.37% of accuracy.

Social honeypots and reversing engineering techniques (Yang, Zhang, and Gu, 2014) In 2014, active social honeypots on Twitter and reversing engineering techniques were adopted to provide guidelines for building more effective social honeypots. The idea was to first perform a measurement study by deploying "benchmark" social honeypots with different social behaviors to trap spammers. After five months of data collection, an in-depth analysis was conducted on how spammers find their targets. Based on this analysis, new guidelines were provided for making "advanced" social honeypots able to trap spammers around 26 times faster than "traditional" honeypots. Some of these new guidelines were: **(1)** post tweets related with specific topics; **(2)** post tweets containing special keywords such as Trending topics; **(3)** follow famous account related with specific areas. To evaluate the effectiveness of these guideline, the new "advanced" honeypots were deployed on Twitter and it was found out that indeed they were able to trap more spammers.

Pseudo-Honeypots (Zhang, Zhang, and Yuan, 2019) One may argue that all the existing solutions presented up to now are time-consuming and low efficient in the sense that they filter spams from a large set of blindly collected accounts (Zhang, Zhang, and Yuan, 2019). For this reason, in 2019, an innovative approach named pseudo-honeypot, were proposed for efficient spammers gathering. The general idea is to take advantage of the diversity of Twitter users and select accounts with attributes that are highly attractive to spammers. By constructing pseudo-honeypots on the top of these users, it is possible to collect tweets that have higher probability of including spammers' activities. The key challenge in building pseudo-honeypots is to select the effective attributes that have the high probability to attract spammers. At an early stage, the attributes selected to build the first pseudo-honeypots were those extensively studied in previous works. After a period of time, the list of attributes used was refined by making a choice from those that have been shown to

have the highest probability of trapping spammers. In this second stage, pseudo-honeypots were also equipped with Random Forest classifier to perform classification in the all 500-hour collected tweets. In the last part of the research, a new refinement of the attributes was carried out using the latest results obtained from the "advanced" pseudo-honey. Therefore, new pseudo-honeypot were built again and the system ran for about 50 hours. As final result, the efficiency of spammers gathering was 0.92 spammers per honeypot per hour.

Discussion It is clear that all of these solutions share common traits such as the fact that social honeypots have been used specifically to detect spammers or bot activity in general. Furthermore, they were mainly implemented on Twitter, and only a few research teams have used other social networks such as Facebook. We can identify several reasons for this trend. First of all, spamming is one of the most widespread malicious activities on social networks because it can be used as a basis for carrying out other more dangerous illegal activities. Additionally, Twitter has APIs and policies that facilitate data collection. Not to mention, there are widely adopted Twitter datasets that can be used to train machine learning classifiers. As far as we know, there are no solutions that use social honeypots on Instagram. This may be due to the fact that it is not easy to distribute, maintain and record honeypots' activities on this social network.

2.3 Machine Learning

Machine Learning is a field of Artificial Intelligence (AI) that studies how making computers perform tasks that usually could be performed by humans (Zhang and Lu, 2021). The term "Artificial Intelligence" was first proposed by John McCarthy at the Dartmouth Conference in the 1956 and since then AI has achieved milestone results that seem do no want to stop (Moor, 2006). AI is a wide discipline that interests a large number of sectors, starting from Computer Vision (CV), Natural Language Processing (NLP) and Image Generation. Each of them has its own peculiarities and specific characteristics, as well as its own purposes and algorithms. We will try to depict the state-of-the-art of some of them, in a general manner, with the aim of giving tools to understand our reasoning behind this project. In fact, this work makes massive use of AI, focusing not only on a single branch of it, but bringing together the most innovative solutions implemented in recent years, with the aim of proposing a new advanced approach for the implementation of social honeypots.

2.3.1 Computer Vision

The goal of computer vision is to enable computers to recognize and to understand the world through vision, as humans do (Zhang and Lu, 2021). CV embraces all those tasks typical of biological vision systems such as "seeing" or capturing visual stimulus to understand the surrounding environment and extract complex information. Classical applications can be facial recognition or object detection but even more complex tasks in Autonomous Vehicles or in the robotics sector.

CNN The building block of computer vision is Convolutional Neural Network (CNN). CNNs are specialized kind of deep neural network for processing data, usually images, with a grid-like topology (Goodfellow, Bengio, and Courville, 2016). As the name suggests, the network employs the convolution mathematical operation:

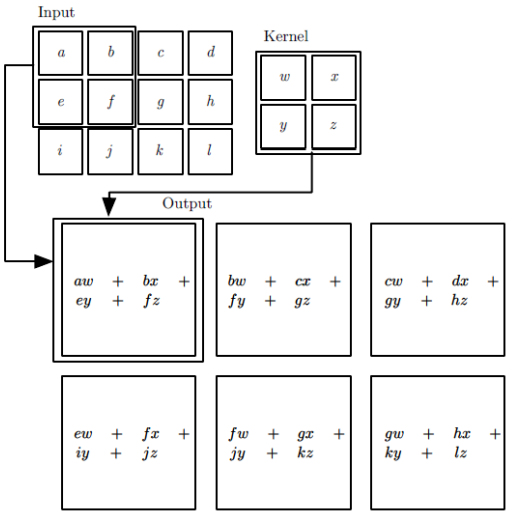


FIGURE 2.2: An example of convolution (Goodfellow, Bengio, and Courville, 2016)

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a)\,da, \tag{2.1}$$

where x is the input and w is the **kernel**, in the convolution network terminology. The output is sometimes referred to as the **feature map**. The input x is organized in three dimensions: height, width and depth, where the depth represents the channel number. Kernels, or filters, have also three dimensions as the input image, but height and width are smaller than the input's.

During the forward pass, the kernel slides across the image computing the image representation of that receptive region. The feature map is a two-dimensional representation of the image that represents the output of the kernel at each spatial position of the image. The sliding size of the kernel is called stride. The three main ideas on which CNNs are based are sparse interaction, parameter sharing and equivariant representation.

- **sparse interaction**: in traditional neural network layers each input unit is connected with each output unit. This means every output unit interacts with every input unit. On the contrary, CNN typically have sparse interactions since the kernel smaller than the input.
- **parameter sharing**: refers to using the same parameter for more than one function in a model.
- **equivalent representation**: the particular form of parameter sharing causes that if the input changes, the output changes in the same way:

$$f(g(x)) = g(f(x)). \tag{2.2}$$

Usually, CNNs can be represented as a stack of convolutional layers interspersed with a pooling layer and a fully connected layer. The pooling layer helps reducing the spatial size of the representation, which decreases the required of weights required and thus the computational resources needed. The fully connected layer is required to map the representation between the input and the output.

Team	Year	Place	mAP
UvA-Euvision	2013	1st	22.6%
Deep Insight	2014	3rd	40.5%
CUHK DeepID-Net	2014	2nd	40.7%
GoogLeNet	2014	1st	43.9%

TABLE 2.1: GoogLeNet Detection perfomance in ILSVRC14 Detection Challenge.

Inception Network

CNNs are a powerful tool that has a big problem: performance. As said before, CNN models most of the time just stack convolution layers one after another, going deeper and deeper, hoping to obtain better results. However, larger sizes typically mean a greater number of parameters which increase the use of computation resources. Besides, the chance of the model suffering from overfitting increased as well. For these reasons, (Szegedy et al., 2015) proposed a new architecture, called Inception, to improve the utilization of the computing resources inside the network. The general idea is to move from fully connected to sparsely connected architectures, even inside the convolution layers. Thus, instead of having deep layers, we have parallel layers which makes the model wider rather than deeper.

The first simple version of the Inception module which has been proposed is depicted in Figure 2.3. This module performs convolution on an input with 3 different filters of sizes 1×1, 3×3, 5×5. Additionally, max pooling is also added in parallel. The filter's outputs are concatenated and used as inputs to the next Inception module. Hence, we have a stack of these Inception modules which are nothing more than a series of convulation operations, with filters of different sizes, processed in parallel. One drawback with this simple architecture is that even the 5×5 convolutional block is expensive with a large number of filters. For this reason, another version was proposed (Figure 2.4) to take under control the computational requirements. In this case, 1×1 blocks are used to compute reductions before the more expensive ones. Recall that 1×1 convolutions are much more faster and cheaper than the 5×5 blocks. This new architecture is known as GoogLeNet (Inception V1): it is 22 layers deep (or 27 layers if we also count pooling) with 9 inception modules stacked linearly.

Still, as with any very deep network, this network is subject to the vanishing gradient problem[1]. One solution proposed by the authors was to introduce two auxiliary classifiers, in the middle part of the network, to encourage discrimination in the lower stages and increase the gradient signal back-propagated. These classifiers are just smaller convolutional networks put on top of the output of Inception modules. During training, their loss is added to the total loss of the network with a discount weight of 0.3. Of course, at inference time, these auxiliary networks are discarded.

This model was responsible for setting the new state-of-the-art for classification and detection in the ImageNet Large-Scale Visual Recognition Challenge 2014 (ILSVRC14). The results, for detection challenge, are shown in Table 2.1. Notice that the reported values are based the mean average precision (mAP) metric.

[1]It is a typical problem with Neural Networks whose layers use certain activation functions, like the sigmoid function. In these cases, the gradients of the loss function approach zero, making the network difficult to train.

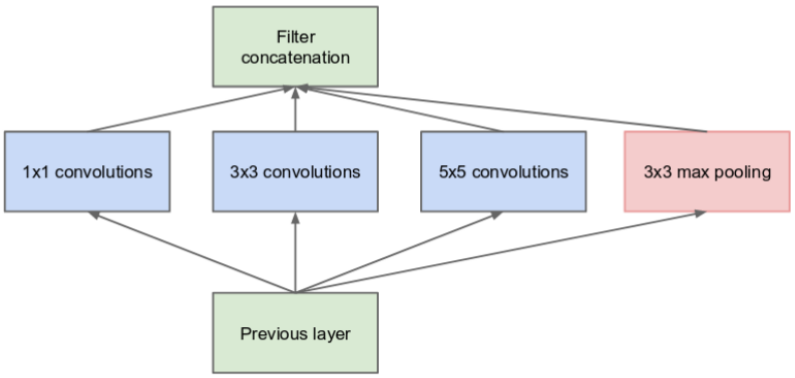


FIGURE 2.3: Inception module, naïve version (Szegedy et al., 2015).

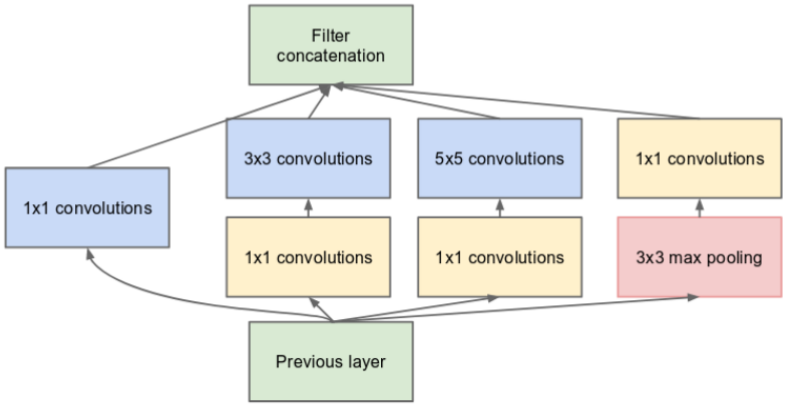


FIGURE 2.4: Inception module with dimension reduction (Szegedy et al., 2015).

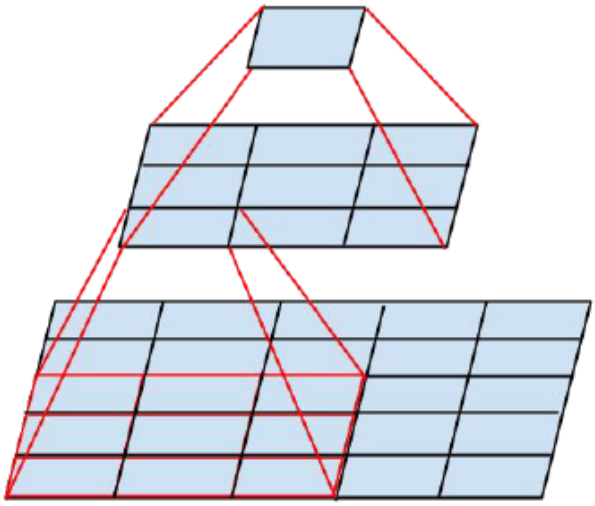


FIGURE 2.5: Mini network replacing the 5×5 convolution block (Szegedy et al., 2016).

InceptionV3 Actually, the complexity of the InceptionV1 architecture makes it more difficult to make changes to the network as, if the network is scaled up carelessly, much of the computational gains can be immediately lost (Szegedy et al., 2016). For these reasons, few years later, more advanced versions of the Inception module, InceptionV2 and InceptionV3, were presented by (Szegedy et al., 2016). Several design principles were suggested to increase the performance of InceptionV1:

- Avoid representational bottlenecks, especially early in the network. The idea is that neural networks perform better when the dimensions of the input are reduced gradually and not in a drastic way.
- Higher dimensional representations are easier to process locally within the network.
- Balance the width and depth of the network helps. The optimal improvement can be reached if both are increased in parallel.
- More suitable factorization methods can make the convolutions more efficient. From now on, we will focus mostly on this last principle.

Convolutions with large spatial filters such as 5×5 tend to be computationally expensive but they can be factorized in a clever way (Szegedy et al., 2016). Looking at the Figure 2.5, we can notice that each output looks like a fully-connected network sliding over the 5×5 block over its input. By using translation invariance, we can replace the fully-connected component with a two layer convolutional architecture: the first layer is a 3×3 block, the second is a fully connected layer on top of the 3×3 block. By sliding this small network over the input we can replace the 5×5 with two 3×3. In a more general fashion, any $n \times n$ convolution can be replaced by one $1 \times n$ block followed by another $n \times 1$ block. This insight leads to the possibility of substituting the 3×3 with two blocks, one of 1×3 and one 3×1. Figure 2.6 shows what the authors called InceptionV2 (version 2 of the original inception module) in which the 5×5 has been replaced with its factorization as well as the 3×3. These changes made the network extremely fast and it obtains better results in terms of convergence.

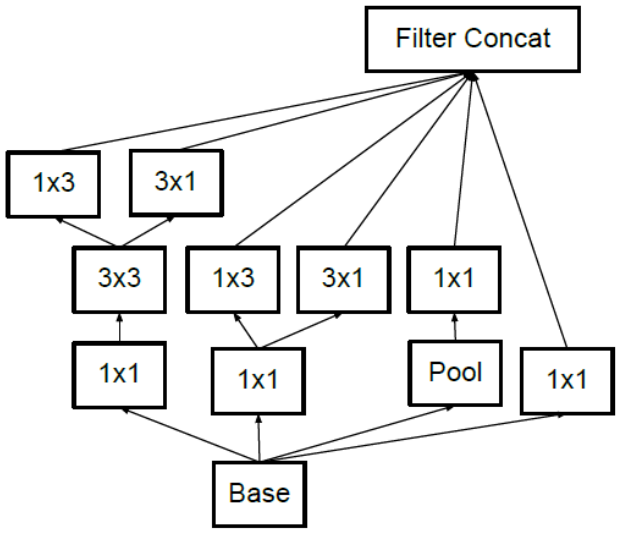


FIGURE 2.6: InceptionV2: making the inception module wider.

Network	Models Evaluated	Crops Evaluated	Top-1 Error	Top-5 Error
VGGNet (Simonyan and Zisserman, 2014)	2	-	23.7%	6.8%
GoogLeNet (Szegedy et al., 2015)	7	144	-	6.67%
PReLU (He et al., 2015)	-	-	-	4.94%
BN-Inception (Ioffe and Szegedy, 2015)	6	144	20.1%	4.9%
InceptionV3	4	144	17.2%	3.58%

TABLE 2.2: Ensemble evaluation results comparing multi-model, multi-crop reported results (Szegedy et al., 2016).

Not completely satisfied, they proposed also the InceptionV3 (version 3 of the original inception module) which uses the same type of inception module of InceptionV2 but with some additions. It adopts RMSProp Optimizer, BatchNorm in the auxiliary classifiers and Label Smoothing technique that is a mechanism to regularize the classifier layer by estimating the marginalized effect of the label-dropout during training (Szegedy et al., 2016). The InceptionV3 is the model that obtained the best results compared with the best published ensemble inference results on the ILSVRC 2012 classification benchmark (see Table 2.2).

2.3.2 Natural Language Processing

Natural Language Processing (NLP) refers to the ability of computers to recognize and understand human text language, which is an interdisciplinary subject between computer science and human linguistics (Zhang and Lu, 2021). NLP can be divided into many directions: grammatical and semantic analysis, information extraction, natural language generation, information retrieval, machine translation, sentiment analysis, question answering system and dialog system (Zhang, Xu, and Chen, 2020). NLP is perhaps the most talked about discipline in AI because automatic text analysis, as for humans, requires a much deeper understanding of natural language by machines, which is still far from reality. (Chowdhary, 2020). Before 2017, the Recurrent Neural Networks (RNN) (Rumelhart, Hinton, and Williams, 1986), Long Short-term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) and gated recurrent neural networks (Chung et al., 2015) have been widely used in sequence modelling and transduction problems.

RNN RNNs are a family of neural networks for processing sequential data. In general, an RNN takes as input a sentence, processes it one word at time and, by aligning the positions to steps in computation time, it generates a sequence of hidden states (see Figure 2.7):

$$h^{(t)} = f(h^{(t-1)}, x^{(t)}; \theta), \tag{2.3}$$

where $x^{(t)}$ is the input at time t, $h^{(t-1)}$ the previous hidden state and θ some additional parameters.

RNNs work well with short sentences or those that do not require to remember too much past context. However, in real use cases, we need to deal with long-term dependencies. The drawback of RNNs is that they can not remember long term dependencies due to vanishing gradient. LSTMs and gated recurrent neural networks are explicitly designed to overcome this problem.

LSTM has feedback connections able to process the entire sequence of word that helps in remembering the context. A common LSTM unit has three gates: an input gate, an output gate and a forget gate, that have the ability to carefully remove or add information to the cell state. The first step, in LSTM units, is to decide what information are going to be threw away from the state. This decision is made by the forget gate: it looks at $h^{(t1)}$ and $x^{(t)}$, and outputs a number between 0 and 1, where 0 represents "forget this" while a 1 represents "keep this". The input gate deals with the new information carried by the input. It decides what new information are going to be stored in the cell state. The output of the LSTM cell $h^{(t)}$ is decided by the output gate. First, a sigmoid layer decides which parts of the cell state will be the output and then this value is multiplied by the tanh of the current cell state.

Transformers

Even with the help of LSTM mechanisms, it is undeniable that RNNs have another problem to face: since they process words sequentially, they are hard to parallelize, thus they require a lot of computational power. This makes the training procedure on large dataset a truly challenge. It was in 2017 that a new proposal changed drastically the state-of-the-art not only for NLP but for AI in general. Transformers, developed by a Google and University of Toronto research team, is a model that makes use of an Attention mechanism to deal with global dependencies between input and output, giving up completely to recurrence (Vaswani et al., 2017). Without so much efforts, Transformers outperform any type of model used until that moment. As we can see from Figure 2.8, Transformers are composed of two main parts: the left part for encoding and the right part for decoding. At the end of the decoder, there is an output layer that generates the final result. In the original paper, the encoder is a stack of 6 identical layers, each of which with two sublayers: the first is a Multi-Head Attention mechanism, which will be described in section 2.3.2, and the second is a simple feed-forward network. After of each of these two blocks there is a normalization layer. The output of the encoder will be a fixed length vector representation for each word of the initial input that will consider the full context of the sequence. The decoder part is quite similar but it contains an additional block which performs Masked Multi-Head Attention that is an Attention block with some differences to make the decoder performing better.

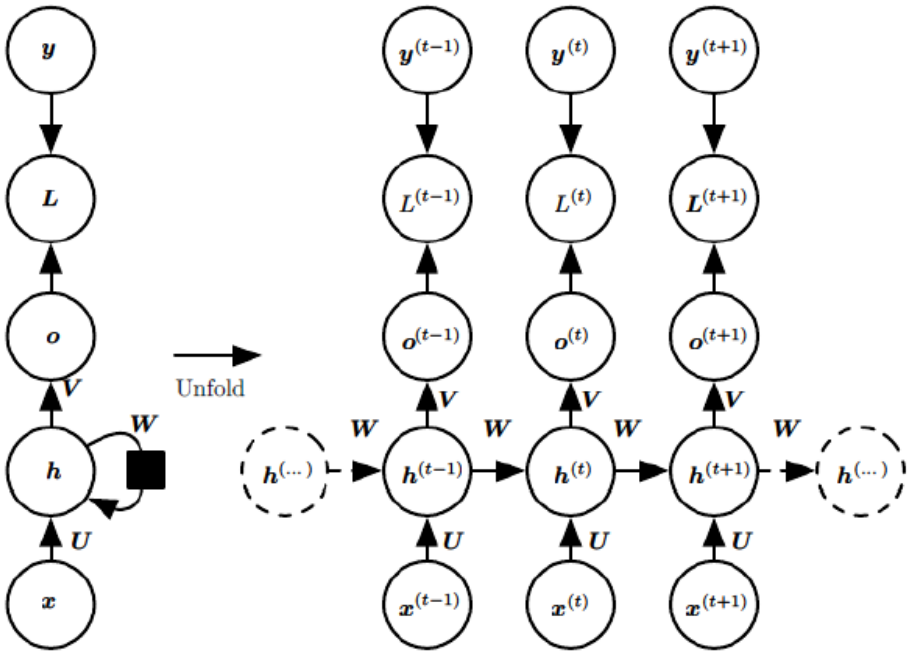


FIGURE 2.7: Recurrent Neural Network (Goodfellow, Bengio, and Courville, 2016).

Positional encoding In Transformers, each word will be processed in parallel making it necessary to remember the position of each of them. For this purpose, Positional encoding is used. The Positional encoding block can be defined as a matrix that contains several vectors encoding a specific position. It uses the sine and cosine functions to compute them, but any function we wish can be adopted.

Hence, starting from the raw sentence, each word is first mapped to an integer, called Token, by the Input Embedding block. This layer is in charged of linking each token to a vector. Therefore, the result will be added to the Positional encoding. The result is the representation of each word with the additional information of the original position in the input sentence.

Attention The key point in Transformers is the use of Attention. The Attention mechanism consists of *Queries*, *Keys* and *Values*. The intuition is that a query vector will be compared with a set of key vectors to determine the correlation between them. At the same time, to each query vector is associated a value vector. The higher the correlation between keys and queries, the more the corresponding value will influence the output of the Attention mechanism. Thus we have three sets: Q, K and V that consists of all the query vectors, key vectors and value vectors, respectively. Notice that all these three components are learned during training by using the output of the summation between the input embedding and the positional encoding.

$$Attention(Q, K, V) = softmax\Big(\frac{QK^T}{\sqrt{d_k}}\Big)V. \tag{2.4}$$

Authors found out that the Attention mechanism seems to suffer for larger value of d_k, namely the dimension of keys. For this reason, they added this scaling factor $\sqrt{d_k}$ to the computation. The Attention's output will be a matrix with an updated representation for each word in the corresponding position.This is the description of

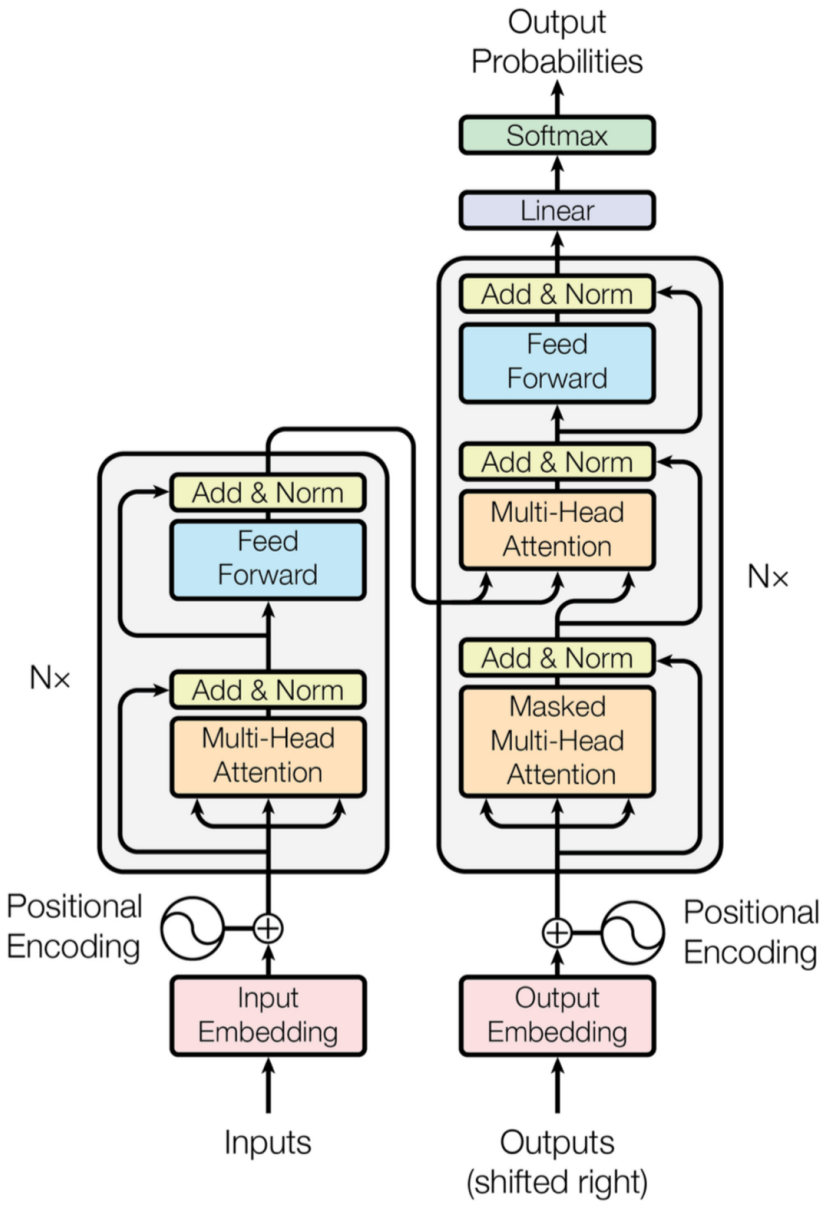


FIGURE 2.8: Transformers architecture.

the process with a single "head", or to make it simply, the computation is done only once.

Multi-head Attention instead performs this computation multiple times because we want to learn multiple representations of the same word at the same time, in parallel. For h Attention heads there are h matrices that will be concatenated and multiplied with another weight matrix to project back the output to the original embedding dimensionality.

$$MultiHead(Q, K, V) = Concat(head_1, head_2, ..., head_h)W^O \tag{2.5}$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

In the decoder, there is also an Attention block but a mask is needed to ensure that the temporal dependencies of the output sentence is respected. The output sentence is sampled one word at a time and the prediction at position t should depend only on the words seen up to position $t-1$. This is accomplished with the use of a sort of a mask during the Attention process. In practice, when the representation at a given position is updated, the decoder should pay no attention to any of the words at positions greater than t. From a mathematical point of view, the mask is a matrix with $-\infty$ above the main diagonal and zeros everywhere else.

One detail to be noted in the Figure 2.8 is that the input of the Multi-Head Attention block in the decoder is the output of the previous block and the final output of the encoder. Hence, these inputs do not come from the same sequence. The queries come from the previous layer of the decode, while the keys and values come from the output of the encoder. This allows the decoder to attend not only to the tokens sampled so far but also to the original input sequence.

GPT-3

As said before, Transformers completely renew the state-of-the-art of NLP making clear that recurrence is not the only possibility to deal with sequences. Moreover, they demonstrated that Attention mechanism can go far beyond expectations. Many researchers started to develop new models using Transformers as architecture but OpenAI made the difference. They proposed a new model, Generative Pre-trained Transformer (GPT), that demonstrated even more the potentialities of Transformers (Radford et al., 2018).

Notice that, depending on the task that we want to perform, we do not necessarily need both encoder and decoder in the Transformer. For instance, if we try to learn a rich sequence representation for a classification task we may use only the encoder part. On the other hand, if we need to generate sequences, a decoder only architecture can be sufficient. That is exactly what OpenAI research team has done (Radford et al., 2018). They realized a model based on a multi-layer *Transformer decoder* (Liu et al., 2018) to generate text-to-text sequences. Besides, they divided the training procedure into two stages: the first one is an unsupervised pre-training in which, given an unsupervised corpus of tokens, the model will apply a multi-head self-attention operation followed by position-wise feedforward layers to produce an output distribution over target tokens. The second stage is a supervised fine-tuning process in which the model is adapted to a discriminative task with labeled data. This model was the first version of GPT: GPT-1.

Few years later, the same authors discussed the fact that many models, until that moment, have been trained to perform single tasks separately. They resembled

	Num. Parameters	Num. Decoder Layers	Num. Hidden Layers	Batch Size
GPT-1	117 M	12	768	64
GPT-2	1.5B	48	1600	512
GPT-3 Small	125M	12	768	0.5M
GPT-3 Medium	350M	24	1024	0.5M
GPT-3 Large	760M	24	1536	0.5M
GPT-3 XL	1.3B	24	2048	1M
GPT-3 2.7B	2.7B	32	2560	1M
GPT-3 6.7B	6.7B	32	4096	2M
GPT-3 13B	13.0B	40	5140	2M
GPT-3 175B or "GTP-3"	175B	96	12288	3.2M

TABLE 2.3: Size, architecture and number of parameters of different GPT versions.

	PPL	ACC
SOTA	99.8	59.23
GPT-1	35.13	45.99
GPT-2	8.63	63.24
GTP-3	3.00	76.2

TABLE 2.4: Performance on LAMBADA dataset.

"narrow experts" rather than "competent generalists" (Radford et al., 2019). That is the reason why they proposed GPT-2, a model able to perform several different tasks without the need of retraining it. It has the same architecture of the original GPT but it is "bigger" than its predecessor. To be precise, it has about 1.5 billion parameters while GPT has only 117 million. Moreover, the model was trained with a dataset that contains a total of 40 GB of text scraped from the Internet, rather than using specific task datasets. In practice, any type of text that we may think of is contained in it. GPT-2 was evaluated on several other datasets for different tasks such as reading comprehension, translation, question&answering, etc. (Radford et al., 2019) and it did great in many of them. It showed that training on larger dataset and having more parameters drastically improve the capabilities of the model.

At this point the question that each of us could ask would be: what if we make it bigger? This is exactly what OpenAI has done with GPT-3 (Brown et al., 2020). GPT-3 has the same model architecture of GPT-2 but it has 175 billion of parameters. Actually, authors provided 8 different models that differ from each other on sizes and learning parameters, and the biggest one was precisely that with 175 billion parameters, named "GPT-3". Table 2.3 lists the different versions of GPT with information about size, architecture and number of parameters.

GPT-3 outperformed the state-of-the-art, including GPT-2, in many different tasks. As an example, in Table 2.4, results obtained with the LAMBADA dataset (Paperno et al., 2016) are reported. LAMBADA dataset tests the ability of systems to model long-range dependencies in text. The task is to predict the final word of sentences which require at least 50 tokens of context for a human to successfully predict (Radford et al., 2019). All the results described in the table are "Zero-shot" results which

means no demonstrations are allowed and the model is only given a natural language instruction describing the task. In some cases, it may also be difficult for humans to understand what the task is asking for. As we can see, GPT-3 results are better both in terms of perplexity (PPL), the human ability to detect model generated texts, and accuracy (ACC).

2.3.3 Image Generation

Generating synthetic but realistic images is what has gained a lot of attention in the research field lately. This can be done by using generative models which are networks capable of generating new content starting from real samples. From a mathematical point of view, we have a dataset of observations, generated according to some probability distribution, and a generative model that tries to mimic this original probability distribution. One of the most used generative model is Generative Adversarial Network (GAN) (Goodfellow et al., 2014). However, recent proposals and studies have brought to light new innovative and interesting solutions.

GAN In GAN, the model consists of two networks: the generator (G) and the discriminator (D). The discriminator learns to determine whether a sample is from the model distribution or the data distribution. GAN are based on a game theoretic scenario in which the generator competes against the discriminator. The generator produces samples $x = g(z; \theta^{(g)})$ while the discriminator attempts to distinguish between samples drawn from the training data and samples drawn by the generator. The discriminator outputs a probability through the function $d(x, \theta^{(d)})$ that is the probability that x is a real training sample rather than a fake sample.

The simplest way to model this problem is through a zero-sum game, in which the function $v(\theta^{(g)}, \theta^{(d)})$ determines the reward of the discriminator. The generator, of course, will receive $-v(\theta^{(g)}, \theta^{(d)})$ as reward. During the learning, each of the gamers tries to maximize its own reward:

$$g^* = argmin_g max_d v(\theta^{(g)}, \theta^{(d)}), \tag{2.6}$$

where v is:

$$v(\theta^{(g)}, \theta^{(d)}) = \mathbb{E}_{x \sim p_{data}} log d(x) + \mathbb{E}_{x \sim p_{model}} log(1 - d(x)). \tag{2.7}$$

Intuitively, this formula will drive the discriminator to distinguish between real and fake examples, while the generator tries to fool the classifier into believing its samples are real. At convergence, the generator's samples are indiscernible from real data, while the discriminator outputs $\frac{1}{2}$ everywhere. Notice that convergence is extremely difficult to be reached.

Diffusion Models For a long period, GANs have been considerate as the state-of-the-art for image generation. However, they have several drawbacks that make them extremely difficult to scale and apply to new domain (Dhariwal and Nichol, 2021). In recent years, diffusion models have been proposed as valid substitutes for GANs as they seem capable of obtaining a higher image sampling quality (Ho, Jain, and Abbeel, 2020, Sohl-Dickstein et al., 2015, Song and Ermon, 2019). In general, these models generate samples by gradually removing noise from the input. On a high level (Figure 2.9), they sample with a noise x_T and produce gradually less-noisy samples x_{T-1}, x_{T-2}... until reaching the final sample x_0, that is the unnoised version of the image. To be precise, each timestep T corresponds to a certain noise level

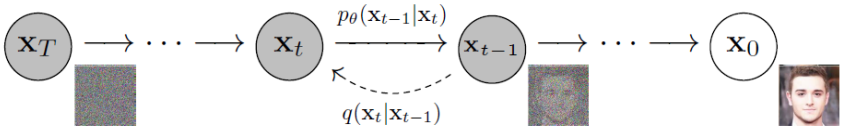


FIGURE 2.9: Directed graphical model of diffusion models

FIGURE 2.10: Diffusion models training. [source]

and the diffusion model learns to produce a slightly more "denoised" x_{T-1} from x_T. Figure 2.10 shows the entire training process which consists of two phases: the fixed forward diffusion process and the generative reverse denoising process. In the first, the noise is added to the original input image, while in the second the model tries to reconstruct the image by removing a little more noise with each pass.

Dall-E 2

This year, on April 2022,a new model known as DALL-E 2 has been proposed for image generation that outperformed all the solutions used until now (Ramesh et al., 2022). DALL-E 2 is able to generate high quality and very realistic images starting from a prompt, or caption, by using diffusion models. The Figure 2.13 shows the impressive results we can achieve with this model. It is evident that the quality and correspondence between text and image is beyond what has been done so far.

DALL-E 2 makes use of an already existing model called Contrastive Language-Image Pre-training (CLIP) (Radford et al., 2021). In general, CLIP is able to learn the correlation between a text and an image. Hence, it learns how much a given caption is related to an image instead of predicting its best caption. To make this possible, CLIP uses two encoders: one will turn the image into image embedding, the other will translate the caption into a text embedding. The goal is to maximize the similarity between the text embedding and the image embedding. The CLIP's ability of learning semantics is the building block of DALL-E 2.

Figure 2.11 depicts the architecture of DALL-E 2. We can identify two parts: above the dotted line there is the CLIP training process through which we can learn a joint representation space for text and images. Below the dotted line, there is the text-to-image generation process. In the next part of the discussion, we will focus mainly of this last process.

Hence we have the decoder that inverts images given their CLIP image embeddings, and a "Prior" block that allows us to learn a generative model of the image embeddings themselves. Putting all together we have a generative model $P(x|y)$ of images x given captions y:

$$P(x|y) = P(x, z_i|y) = P(x|z_i, y)P(z_i|y), \tag{2.8}$$

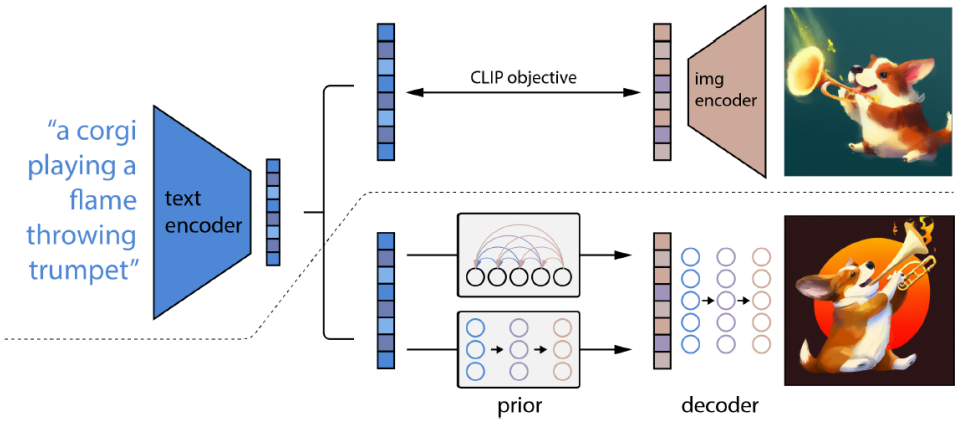


FIGURE 2.11: Dall-2 architecture (Ramesh et al., 2022)

where z_i is the image embeddings of the original image x. Notice that the first equation holds because z_i is a deterministic function of x. This equality means that we can sample from the true conditional distribution $P(x|y)$ by first sampling z_i with the prior, and then sampling x using the decoder.

Prior In DALL-E 2, the Prior is what link the CLIP text embedding with the CLIP image embedding. In the original paper, authors identified two methodologies to implement the Prior:

- Autoregressive approach in which the CLIP embedding is converted into a sequence of discrete codes and predicted autoregressively conditioned on the caption.
- Diffusion approach in which the continuous vector z_i is modelled using a diffusion model conditioned on the caption. They found out that this last one works better for DALL-E 2.

For the diffusion prior, they trained a decoder-only Transformers with a casual attention mask on a sequence consisting of: the encoded text, the CLIP text embedding, the embedding for the diffusion timestep, the noised CLIP image embedding and a final embedding whose output from the Transformer is used to predict the unnoised CLIP image embedding (Ramesh et al., 2022).

Looking at the architecture, one may argue that it would be better, and more efficient, to pass directly the caption, or the text embedding, to the encoder without passing through the Prior block. Authors tested this possibility and they figured out that actually having the prior leads to better results (Figure 2.12). By observing carefully, we can notice that passing the text embedding directly to the decoder gives an acceptable result. However, it was found out that this may lead to loose the capability to generate variations over images.

Decoder The decoder is a diffusion model as well. Even in this case they used another model called GLIDE (Nichol et al., 2021). Unlike a pure diffusion model, in the training procedure, GLIDE provides as input not only the noisy image but also the corresponding caption or text information. In DALL-E 2, CLIP embedding is added as input to the information already passed by GLIDE. This was done to support the image generation. Finally, to generate high resolution images, the decoder has two

FIGURE 2.12: Results for "a hedgehog using a calculator" Ramesh et al., 2022

Model	Zero-shot FID
Dall-E (Ramesh et al., 2021)	28
LAFITE (Zhou et al., 2021)	26.94
GLIDE (Nichol et al., 2021)	12.24
Dall-E 2	10.39

TABLE 2.5: Comparison of FID on MS-COCO 256×256.

diffusion upsampler models: one to upsample images from 64×64 to 256×256 resolution, and another one to further upsample those to 1024×1024 resolution.

In the text-conditional image generation literature, it has become standard practice to evaluate Fréchet Inception Distance (FID) on the MS-COCO (Lin et al., 2014) validation set. Frechet Distance is a measure of similarity between curves that takes into account the location and ordering of the points along the curves. This metric is used to assess the quality of images created by a generative model: the lower the value, the higher quality the image created. As we can see from Table 2.5, the results obtained with DALL-E 2 are quite impressive.

2.4 Discussion

To summarize, in literature social honeypots have already been used as spamming detectors. They can be divided into two broad categories, that are passive or active honeypots. The first ones did not publish any type of content, but just record and analyze users' activity. The latter type of honeypots used tweets differentiated by type of tweets and frequency of publishing. All of the proposed works implemented a machine learning classifier to identify legitimate user from spam accounts.

Regarding machine learning and Artificial Intelligence, there have been huge advances especially in the field of Computer Vision, Natural Language Processing and Image Generation. The latest technologies involve Inception networks, Transformers and Diffusion Models. Despite these great contributions, most of them are not available to the public. In some cases the public can interact with these models through paid APIs, but usually full model access is currently limited to only few highly resourced labs. Furthermore, to access larger models it is necessary to subscribe to a waiting list and the time that you have to wait, before being able to access, could be several months. Nonetheless, there exists several open source models

whose results are quite closed to the state-of-the-art. Among them, Open Pre-trained Transformer model (Zhang et al., 2022) is a decoder-only pre-trained transformers, ranging from 125M to 175B parameters, that is fully shared with other researches. Authors showed that OPT-175B is compatible to GPT-3 and its performance in many tasks, including zero-shot tasks, follow the trend of GPT-3. For image generation, Dall-E 2 can be accessed only upon subscribing the waiting list, but there is an open source version, Dall-E mini (Dayma et al., 2021), that is available to the public. Its generated images are quite good even if they do not achieve the same impressive results of Dall-E 2.

Vibrant portrait painting of Salvador Dalì with a robotic half face.

A close up of a handplam with leaves growing from it.

An espresso machine that makes coffee from human souls, artstation.

A corgi's head depicted as an explosion of a nebula.

A dolphin in an astronaut suit on Saturn , artstation.

A teddybear on a skateboard in times square.

FIGURE 2.13: Some images generated by DALL-E 2 (Ramesh et al., 2022).

Chapter 3

Methodology

Behind this project there was an in-depth preliminary study to carefully design both the implementation of social honeypots and the approach for achieving the goals we have set ourselves. After a brief description of our objectives and the contributions we want to make with this research (section 3.1), we will try to explain as clearly as possible what our approach consists of (section 3.2) together with how the experiments were launched (section 3.3). Some final considerations on the ethical aspects of the whole project will be address and the end of this chapter (section 3.4).

3.1 Motivation

This study aims to propose an innovative and advanced approach for building social honeypots on Instagram. From the beginning, our idea has always been to provide a valid tool that can be a starting point to new researches in this field. Moreover, we wanted to understand how much a social network completely unexplored in literature, such as Instagram, could be exploited for this type of experiment, trying to pave the way towards new developments. Two principles guided the design of this project, that are: the proposed solution must be easily adaptable to different contexts, for instance cybersecurity or marketing sectors, and the management of honeypots has to be automatically.

What convinced us to pursue these objectives was that all the solutions presented in the literature were designed with a specific target, namely spamming detection, and therefore are not easily scalable to other sectors. On the contrary, we believe that a solution should be as general as possible to be easily adaptable to any kind of use. Looking more in detail, the social honeypots proposed in literature achieved better results were those of the "active" type, showing that an approach more similar to what a normal user would have is certainly the most promising possibility. However, it was not well specified how these honeypots generated the tweets. Starting from the assumption that these tweets were generated manually by someone, one of the innovative aspect of the work we are presenting is the use of the latest machine learning technologies for making honeypot management and post generation completely automatic.

3.2 Proposed approach

Since this is the first work in literature with the aforementioned objectives, we decided to focus our project on giving a comprehensive study of different possibilities for building social honeypots with the already discussed characteristics. We have tested them all trying to identify the best strategy, for the management of social honeypots, on which the research should continue. In general, our honeypots are

Instagram accounts characterized by a **topic**, a **post generation strategy** and an **engagement plan**.

3.2.1 Topic

A topic represents the type of content that the Instagram posts, published by a single honeypot, will be based on. It can be an argument of interests or a specific product that we want to advertise or a set of different arguments to capture a specific audience of interest. It is important to choose this topic carefully because it will determine the type of users we want to attract. To give some examples, if we want to advertise the new product of a particular brand, a possible choice for the topic could be the type of product we are advertising. On the contrary, if we want to develop a tool for spamming detection, the chosen topic should be one of interest for spammers so that the honeypot will post attractive content for them. We can even go further and make honeypots with a generic topic that can be used for marketing profiling or social studies. As we can see, the topic is essentially an argument that can be chosen based on the final purpose of the honeypot.

Since we did not have any reference to start from, our choices were based on hashtags: we identified three hashtags with broad, medium and specific coverage. Coverage is a metric that counts the number of posts per hashtag or, to make it simply, the amount of posts, in a specific day, that contain that particular hashtag in the caption. This information can be easily retrieved from Instagram itself. Hence, we chose:

- **Food** with the corresponding hashtag #food counting 493 million of posts (Broad Coverage).
- **Cat** with the corresponding hashtag #cat counting 270 million of posts (Medium Coverage).
- **Car** with the corresponding hashtag #car counting 93,2 million of posts (Specific Coverage).

It should be noted that these numbers are relative to the time the author is writing this thesis.

3.2.2 Post generation strategy

We have developed four models to generate Instagram posts, each of them with specific characteristics and algorithms. They can be divided into two categories: those that generate a new image at the end of the process, and those that use already existing images for creating the final post. In the next sections each of them will be explained starting from the generative ones.

InstaModel

InstaModel is a generative model that makes use of machine learning techniques to generate a new post. Figure 3.1 gives an overview of the process: starting from a post among the top 25 Instagram posts for a specific hashtag, a new caption is generated by taking into consideration both the original image and the original caption. The new generated caption is then used to generate a new image. These two elements will make the final post that the honeypot will use on Instagram.

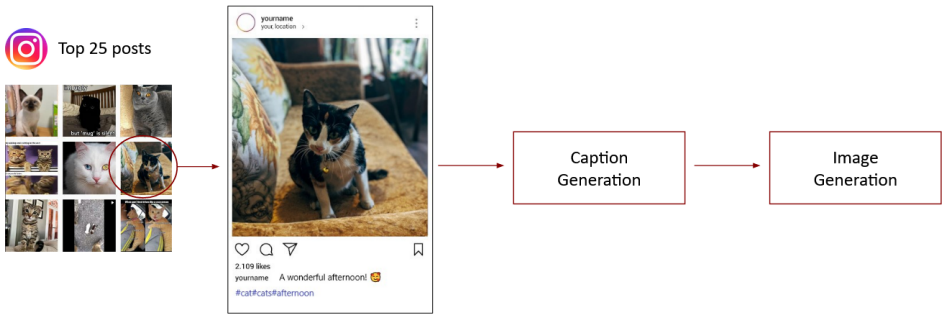


FIGURE 3.1: InstaModel overview.

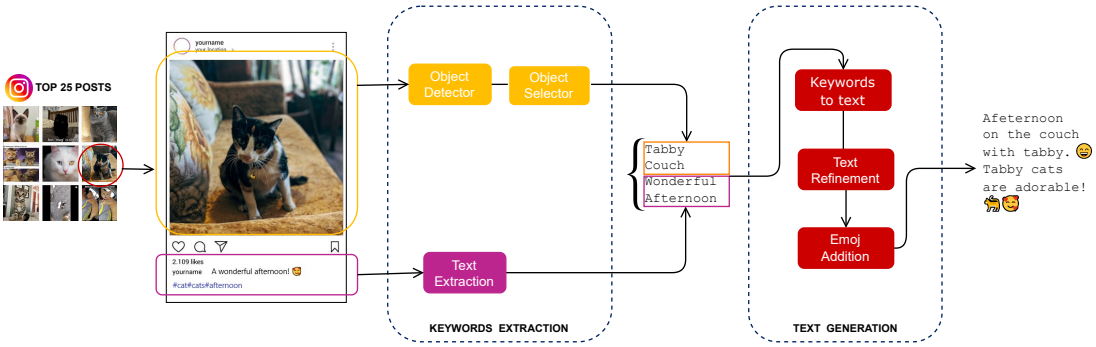


FIGURE 3.2: InstaModel - caption generation.

Caption Geenration Figure 3.2 depicts the main blocks of this process, which is composed of two steps: keywords extraction and text generation. As said before, for generating new captions, both the original image and the original caption are used. An object detector identifies the objects presented in the image, called classes, along with the accuracy with which they were identified. These classes are scanned by an object selector that decides if the specific Instagram post has to be discarded or not. Since not all the Instagram posts have suitable images for this process, we decided that if the highest accuracy achieved is not greater than or equal to 0.25, the post will be discarded. This is to avoid to process images, memes for instance, that will produce bad results or will affect the final image generation process. If the image passes this check, the other detected classes are also scanned and only if their accuracy is greater than 0.5 will they be considered later.

As an example, assume the original post to be the one depicted in Figure 3.2 and let the detected classes, with the corresponding accuracy, be:

$$\begin{cases} \text{Tabby} & 0.52 \\ \text{Couch} & 0.34 \\ \text{Tiger\_cat} & 0.05 \\ \text{Lynx} & 0.03 \end{cases}$$

Since the highest score obtained is 0.52, the post is not discarded and the other classes will be checked as well. However, only the first two, "Tabby" and "Couch", have a precision greater than 0.05 and therefore will be taken into account in the next steps of the generation process. This additional check is done to avoid considering objects with a low precision which may affect the quality of the caption generation. While the original image is processed in the way just explained, the original caption

is scanned by a text extraction block to extract nouns and adjectives. Hence, at the end of these parallel procedures, the result will be a list of keywords which will become the input of the text generation block.

The text generation block uses these keywords to generate a new caption. The first step is to combine these keywords with each other in order to make a preliminary sentence. Considering the example described above, the first generated sentence will be:

```
"Afternoon on the couch with tabby"
```

This sentence is then refined to make a more complex sentence and, as last step, emojis are added to give the classical Instagram caption style.

Image Generation The final caption generated after the caption generation block is then used, as a prompt, to generate the new corresponding image. Once we have both the new caption and the new image, the Instagram post is completed and ready to be published.

One detail that we want to highlight is that this model makes a massive use of machine learning algorithms. Indeed, the state-of-the-art of computer vision, NLP and image generation models is used for object detection, text generation and image generation, respectively.

Further examples generated with this model are presented at the end of this chapter in Figure 3.7.

ArtModel

ArtModel is the second generative model and Figure 3.3 shows its overall structure. As the name suggests, the images generated through it are of the artistic type. This model differs from the previous one because the input for the text generation block is processed in a completely different way: it does not came from Instagram but it is computed automatically by giving a keyword and random style and medium. By selecting randomly the style we will set the style of the artistic picture while the medium will define the artistic support of the picture which can be, for instance, a painting, a sketch or an ink drawing. Once we have this caption, the image generation block works exactly the same as the one in the InstaModel. Several examples generated with this model are presented at the end of this chapter in Figure 3.8.

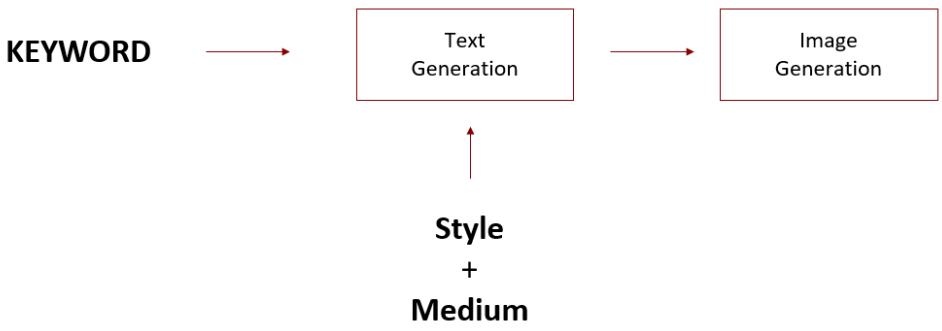


FIGURE 3.3: ArtModel overview.

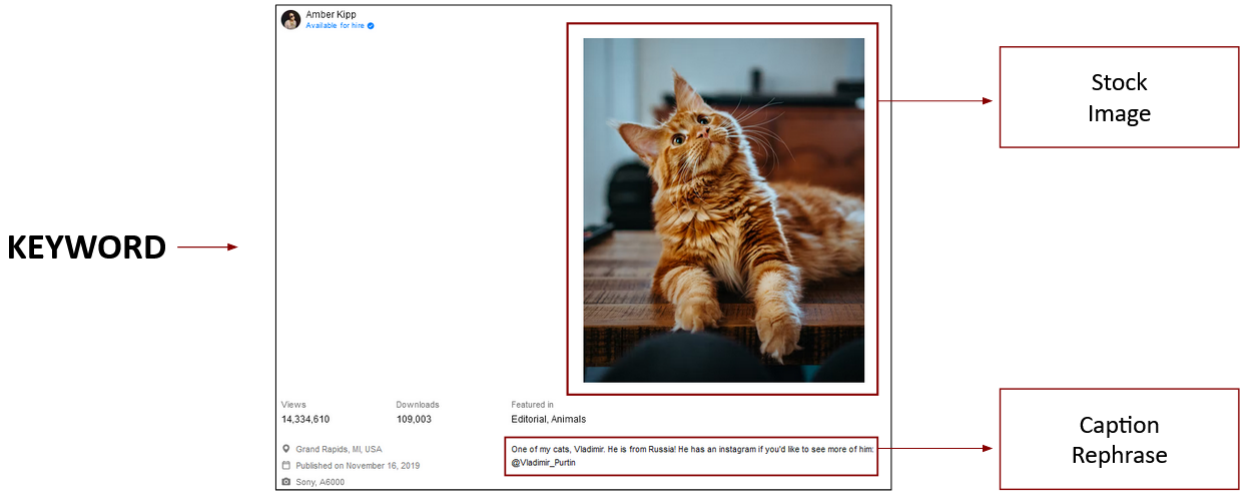


FIGURE 3.4: UnsplashModel overview.

UnsplashModel

UnsplashModel does not belong to the generative group and it uses already existing images. It takes its name from a popular website of stock images, Unsplash. In our case, we chose Unsplash because the stock images uploaded on its website are accompanied with a free to use caption. Actually any type of stock images website can be used and we chose this one as a general example. Looking at the generic representation of its structure (Figure 3.4), we can see that, in this case, we do not have an image generation block since stock images can be used without copyrights issues. Each image is selected by using a specific keyword, that can be for instance the topic of the honeypot, and published directly on Instagram. For the caption, the text generation phase has been substituted with a caption rephrase block that is in charged of rephrasing the original caption that can be found on the website. This phase makes use of machine learning model for text summarization and rephrasing. Several examples generated with this model are presented at the end of this chapter in Figure 3.9.

QuotesModel

QuotesModel is the last model that we have developed and it is not a generative one. Even in this case, as the UnsplashModel, it borrows stock images from popular websites to make the Instagram posts. Looking at the general model in Figure 3.5, it shares some common trait with the previous model: there is no image generation process since the image is chosen from a pool of stock images by using a keyword. However, the difference is that a random quote, from a citation dataset, will be used as caption for the Instagram post. Several examples generated with this model are presented at the end of this chapter in Figure 3.10.

3.2.3 Engagement Plan

Once we have chosen the topic of the honeypot and which post generation strategy it will use, we need a plan to develop and increase the engagement with the users. As mentioned before, our idea is to make active honeypots that not only publish posts on Instagram but are able to generate engagement and therefore attract more users.

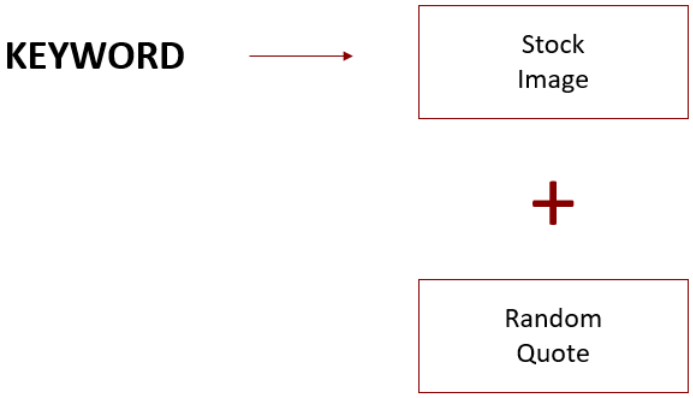


FIGURE 3.5: QuotesModel overview.

For this reason, we have identified three main engagement plans. They go from the simplest to the most complex, both in terms of costs and the amount of engagement that should be generated. Our purpose is to understand which one is more effective combined with different post generation strategies.

PLAN 0 - CTA/Poll/Quiz PLAN 0 adds call-to-actions (CTA) to the generated caption trying to encourage users to make actions such as liking the post or sharing it with their friends. Besides, poll or quiz may be added to the generated caption to involve users in answering some questions in the comment or to leave their opinions about a topic. An example of poll and quiz is shown in Figure 3.6.

PLAN 1 - CTA/Poll/Quiz/Spamming In addition to the techniques of PLAN 0, PLAN 1 makes use also of spamming. Our honeypots, equipped with this engagement plan, leave likes and comments automatically on the top 25 posts for a specific hashtag. In this context, spamming activity is not intended as the classical spam that we may encounter on a social network. The comments that they post automatically are comments that resemble the ones of a regular Instagram user and there is no intention of harming other account with annoying spamming requests. They are supposed to generate engagement with the owner of most appreciated posts in that moment, hoping to redirect this stream on our accounts.

PLAN 2 - CTA/Poll/Quiz/Spamming/Buy Follower/ Sponsored Content This plan combines all the strategies described in PLAN 1, but adds two more paid strategies. One of this, Buy Follower, is not allowed by Instagram, thus we need to lean on external sites. The idea is to boost the honeypots equipped with this engagement plan with initial 100 followers that will be then discarded in the analysis phase. This is because usually users are more prone to generate engagement if the other user has already some followers. We know that, actually, this strategy does not follow the Instagram policy and that most of the time is used for malicious purposes. Nevertheless, we want to test even this possibility to understand if it will have some effects and of which type. The last option is to used the content sponsoring technique proposed by Instagram. It allows specific post to be sponsored for a certain amount of time, targeting a specific group of users based on gender, age, location and interests.

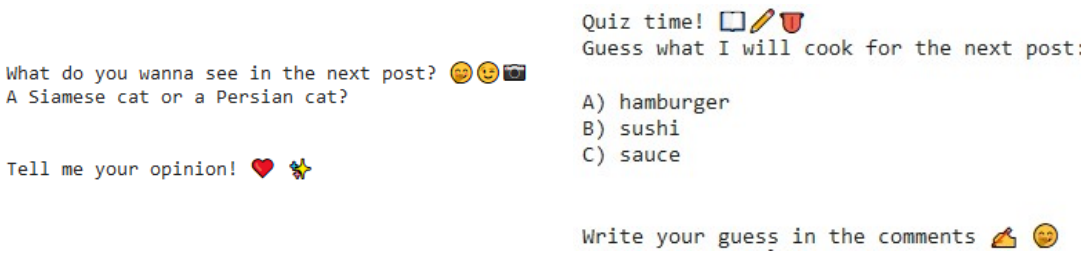


FIGURE 3.6: Poll and quiz examples.

3.3 Experiment

The experiment consists of the deployment of 21 honeypots for at least two months. The 21 honeypots are grouped in this way: seven of them will be based on the topic "FOOD", seven are based on the topic "CAT", and finally the last seven honeypots will use the topic "CAR". For each block of honeypots, there is an internal division that takes into consideration which post generation strategy will be used: for each group, three honeypots will use only generative models (InstaModel and ArtModel), three honeypots will use only the no generative models (UnsplashModel and QuotesModel) and one honeypot will use all of the developed models. Furthermore, there is a second internal division among the honeypot belonging to each topic and to each post generation strategy group, that is which engagement plan they use: one of the three honeypots will use PLAN 0, one honeypot will be equipped with PLAN 1 and the last one with PLAN 2. For the honeypot that uses all the models, we have only PLAN 2. What we obtain is a tree structure in which each leaf represents a honeypot typology that has different behavior both in terms of post generation and how to bring engagement. Table 3.1 lists all the honeypots currently running on Instagram.

The reasoning behind these choices is that we want to understand which combination is able to obtain the best results. In particular:

- we want to figure out if models based on machine learning techniques perform better than the ones that do not use them. Or, on the contrary, if a mixed strategy is more effective.
- we want to find out which combination of engagement plans and post generation strategies is the one to be preferred over the others.
- we want to understand the audience that each honeypot is able to capture.

All of these considerations made clear that we need both quantitative and qualitative data. Thus the data collected, for each honeypot, consist of:

- Total number of followers per day;
- Total number of likes per week;
- Total number of comments per week;
- The post with the most number of likes and comments per week;
- Cities and countries of followers for whom we have demographic data;
- The gender and age distribution of followers for whom we have demographic data.

TABLE 3.1: Honeypots deployed

Name	Topic	Post Generation Strategy	Engagement Plan
h1_cat_ai_0	cat	InstaModel + ArtModel	PLAN 0
h2_cat_ai_1	cat	InstaModel + ArtModel	PLAN 1
h3_cat_ai_2	cat	InstaModel + ArtModel	PLAN 2
h4_cat_no_ai_0	cat	UnsplashModel + QuotesModel	PLAN 0
h5_cat_no_ai_1	cat	UnsplashModel + QuotesModel	PLAN 1
h6_cat_no_ai_2	cat	UnsplashModel + QuotesModel	PLAN 2
h7_cat_mixed_2	cat	All Models	PLAN 2
h8_food_ai_0	food	InstaModel + ArtModel	PLAN 0
h9_food_ai_1	food	InstaModel + ArtModel	PLAN 1
h10_food_ai_2	food	InstaModel + ArtModel	PLAN 2
h11_food_no_ai_0	food	UnsplashModel + QuotesModel	PLAN 0
h12_food_no_ai_1	food	UnsplashModel + QuotesModel	PLAN 1
h13_food_no_ai_2	food	UnsplashModel + QuotesModel	PLAN 2
h14_food_mixed_2	food	All Models	PLAN 2
h15_car_ai_0	car	InstaModel + ArtModel	PLAN 0
h16_car_ai_1	car	InstaModel + ArtModel	PLAN 1
h17_car_ai_2	car	InstaModel + ArtModel	PLAN 2
h18_car_no_ai_0	car	UnsplashModel + QuotesModel	PLAN 0
h19_car_no_ai_1	car	UnsplashModel + QuotesModel	PLAN 1
h20_car_no_ai_2	car	UnsplashModel + QuotesModel	PLAN 2
h21_car_mixed_2	car	All Models	PLAN 2

3.4 Ethical considerations

Classical honeypot, by definition, has no real "user", and thus contains no personal communications or Personally Identifiable Information (PII) (Dittrich, 2015). On the contrary, social honeypots may be filled with personal information and potentially private communications. Unfortunately, this relevant aspect is often not considered in related researches. For instance, among all the reviewed solutions, just few of them paid attention to this research aspect (Lee, Eoff, and Caverlee, 2011, Yang, Zhang, and Gu, 2014, De Cristofaro et al., 2014). Nevertheless, authors of some of the them made just an assumption relaying on the fact that since their honeypots do not interact with users, there is no reason for a legitimate user to be tempted (Lee, Eoff, and Caverlee, 2011,Yang, Zhang, and Gu, 2014). Only one study (De Cristofaro et al., 2014) focused on this aspect claiming that they only collected openly available data, thus no personal information was extracted, and only aggregated statistics were analyzed.

Before starting this work, we decided to carefully analyze this side of social honeypots and take all necessary precautions. Following the approach of (De Cristofaro et al., 2014), to protect the privacy of every user who interacts with the honeypots, we are only collecting openly available statistical data with the Instagram APIs, thus no user can be identified by them. Furthermore, to increase the security of the process, only a small number of people have access to the data and all information are kept confidential and no-redistributed. Upon completion of this study, all collected data will be deleted. This approach complies with the General Data Protection Regulation (GDPR).

In addition to the issue of data collection, there is another issue that usually comes along with social honeypots: the use of deception. At the beginning, users were not informed that they are interacting with no real accounts for research purposes. This was necessary to understand how users would behave and the effectiveness of the honeypot itself. Similar to previous works (Quercia et al., 2011; Hanson et al., 2013), we could not request informed consent to prevent participants from (in)voluntarily changing their behavior, causing the Hawthorne effect (Franke and Kaul, 1978). For this reason we will do our best to inform the deceived people at the end, perhaps by publishing an informative post about the study that has been carried out.

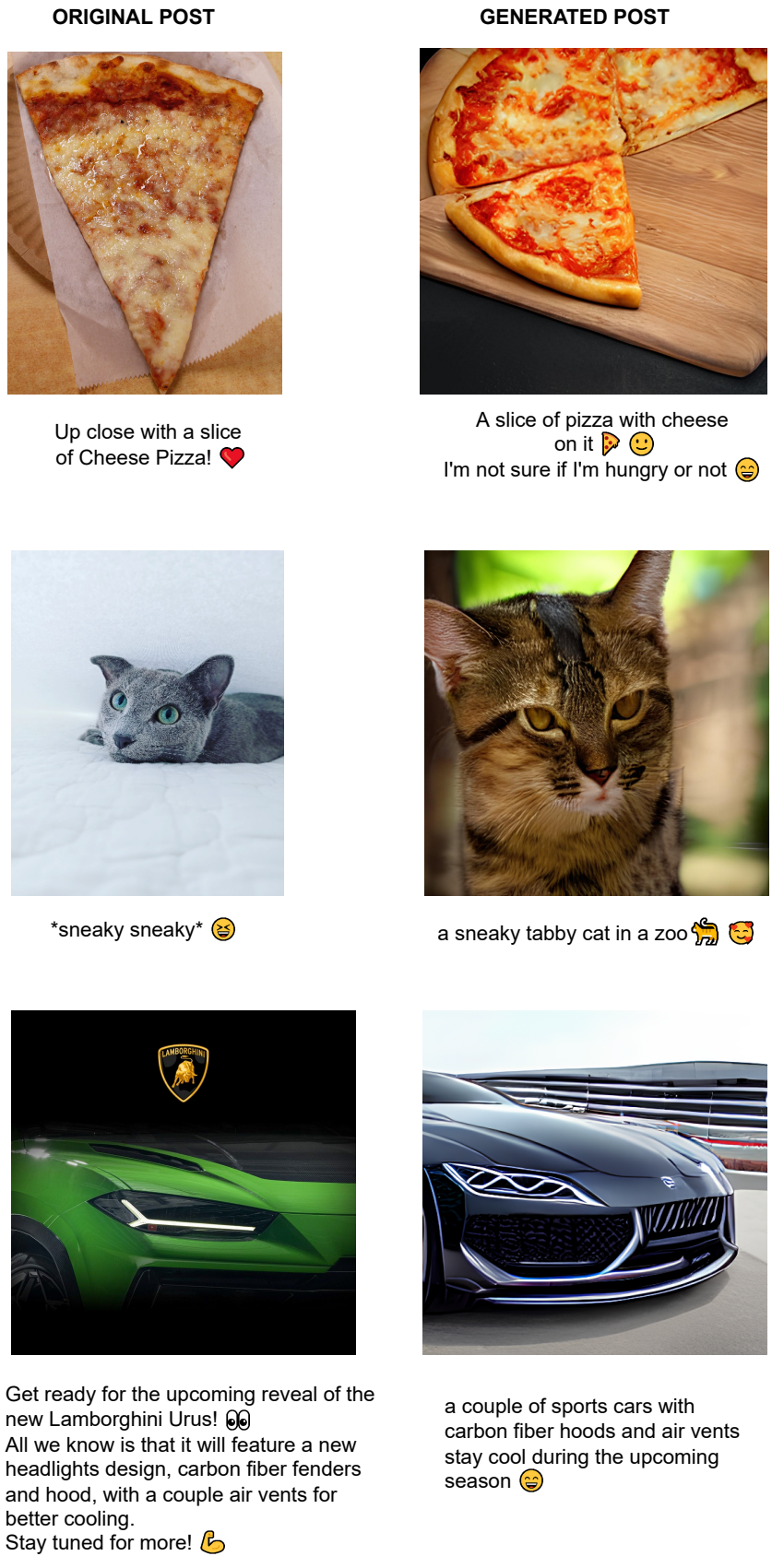


FIGURE 3.7: InstaModel samples.

FIGURE 3.8: ArtModel samples.

GENERATED POST

Home made pasta and risotto. 😁❤️

The cat is sleepy. 🐈 😀

There is a big 4x4 car and a beautiful landscape in the picture, but I don't remember where it was taken. 😻

FIGURE 3.9: UnsplashModel samples.

GENERATED POST

"If you lose something, do not worry."

Lailah Gifty Akita

"When one door of happiness closes, another opens; but often we look so long at the closed door that we do not see the one which has been opened for us."
Helen Keller

"It is in your moments of decision that your destiny is shaped."
Tony Robbins

FIGURE 3.10: QuotesModel samples.

Chapter 4

Implementation

In this chapter, the project implementation will be addressed in detail in order to understand the technological dynamics behind the results that will be shown in chapter 5. After an initial brief description in section 4.1 of the technical features needed for the implementation, section 4.2 will explain the Instagram API chosen, while the models' implementation is address in section 4.3. Finally, section 4.4 will delineate some implementation aspects of the spamming activity.

4.1 Prerequisites

This project is implemented in python 3.7, on a Windows machine with Windows 10 operating system. It makes use of Instagram Graph API to retrieve information about the honeypot itself and to publish new posts. Moreover, since the four models are based on machine learning techniques that normally involves a lot of computational resources, Google Colab with the GPU was used to implement and run all of them. In the following sections, each aspect of the implementation will be described in detail.

4.2 Instagram Graph API

There are two types of Instagram API that are Instagram Basic Display API and Instagram Graph API. The difference between them is that Instagram Basic Display API allows to get basic profile information together with information about photos and videos in our Instagram account. Hence, the API only provides read-access to basic data. On the contrary, as we can read from the official documentation: the Instagram Graph API allows Instagram Businesses and Creators accounts to manage their presence on Instagram. The API can be used to get and publish their media, manage and reply to comments on their media, identify media where they have been @mentioned by other Instagram users, find hashtagged media, and get basic metadata and metrics about other Instagram Businesses and Creators accounts. The API is intended for Instagram Businesses and Creators who need insight into, and full control over, all of their social media interactions. For this project, therefore, we have used the latter type of Instagram API.

Long-access Token A Token is needed in order to use this APIs and the procedure to retrieve it involves Instagram, Facebook and Facebook for Developers platforms. To be precise, we need to get access to an Instagram Business Account, a Facebook page connected to that account, a Facebook Developer account that can perform tasks on that page and a registered Facebook App.

Obtaining the token requires many steps and several procedures that must be followed carefully. Since this token is the building block for the development of all other parts of the project, we will try to be as precise as possible in explaining how to recover it. Therefore, the first thing to do is to register on Instagram, create a regular account and then switch it to a Business account. Note that there are two possibilities we can choose from: Business Account or Creator Account. For the kind of experiments we will be doing, Business is the one that needs to be chosen. Once we have the Instagram Business account, we need to register on Facebook, create a new page and, in the settings, link this page with the newly created Instagram Business account.

At this point, we can login in Facebook for Developers platform and register a new Facebook App. In the settings of this new App, two components has to be added: Facebook login and Instagram Graph API. We can add these products from the App Dashboard. After all these steps, the User Access Token can be obtained through the exploration tool for API graph that can be found among the tools provided by Facebook for Developers. Notice that, to perform the tasks implemented in this project, the Facebook App needs these permissions:

- `read_insights`;
- `pages_show_list`;
- `instagram_basic`;
- `instagram_manage_comments`;
- `instagram_manage_insights`;
- `instagram_content_publish`;
- `groups_access_member_info`;
- `instagram_manage_messages`;
- `pages_read_engagement`;

The last step before being able of using the Instagram Graph API is to retrieve the id of the Instagram Business account. For this purpose, the id of the Facebook page linked to the Instagram Business account is needed and can be found in the settings of the Facebook page itself. The API for recovering the Instagram account id is:

```
https://graph.facebook.com/{graph-api-version}/{facebook-page-id}?
access_token={your-access-token}&fields=instagram_business_account
```

in which the graph api version, for instance v13.0, the Facebook page id and the user access token has to be inserted.

Once we have the Instagram account id, we can start to use all the Instagram Graph APIs. Actually, to be as precise as possible, the access token that we have obtained with this procedure has an expiration date. For this reason, we suggest to request the long-term access token:

```
https://graph.facebook.com/{graph-api-version}/oauth/access_token?
grant_type=fb_exchange_token&client_id={app-id}&client_secret={app-secret}
&fb_exchange_token={your-access-token},
```

where the parameter needed are the client id of the Facebook App, the secret key of the Facebook App, and the access token that we have up to now. The client id and the client secret can be found in the App Dashboard, on Facebook for Developers platform.

At the end of all this procedure, the Instagram Business account id and the long-term access token will be the two values that will be used in the subsequent API calls implemented in this project. The last thing to note is that, from now on, we will refer to the Instagram Business account created as "honeypot" to be consistent with the terminology used so far.

GET APIs Several GET APIs are used in this project, for instance to get basic information about the honeypot, a specific published media, or about hashtags.

Among all the basic honeypot information that can be retrieved, there are the follows count and the followers count. These values are used to understand if the number of follows and followers increases daily, remains stable or decreases. At the same time, there is a variety of information that can be recovered from a specific published media, but what we are most interested in is the count of likes and the count of comments. These values are used to understand the weekly honeypot trend in terms of the total amount of likes and comments.

Finally, as a last useful information for the implementation of the project, we have also collected some statistics on hashtags. In general, the Instagram Graph API allows us to search for the 25 most popular media and the 25 most recent media for a specific hashtag. We used these two API calls to understand which hashtags, for the corresponding topic of the honeypot, are the most used on Instagram and to study the hashtag trend for the 5 most used hashtags in general.

POST APIs Besides the GET APIs, this project makes use also of POST APIs since a honeypot should be able to publish content automatically on Instagram. The official documentation provides some limitations for using these POST APIs:

- Instagram Creator accounts cannot use them. This is the reason why a Business account is needed.
- Accounts are limited to 25 API-published posts within a 24 hour period.
- JPEG is the only image format supported.
- Stories are not supported.

These limitations make clear why we made some implementative choices such as our honeypot only makes posts and not stories and, additionally, all the images generated by our models are converted into JPEG. Actually, there is another limitation with this procedure: images have to be hosted somewhere. More precisely, the image cannot be stored in a local computer and then used as parameter but needs to be reachable by an URL. Publishing a post on Instagram requires two steps: first create a container for the post, and second use the container id to publish. The container is created in this way:

```
https://graph.facebook.com/v5.0/{ig-user-id}/media?image_url={image-url}
&caption={caption}&access_token={access-token}
```

As we can see, it requires two parameters besides the access token and the Instagram account id, which are the URL of the image and the caption. To overcome

this problem, we relied on Discord. Discord is a VoIP social and instant messaging platform where a User can create what is called a "server". A server is a collection of persistent chat rooms and voice channels that can be accessed through invitation links. What convinced us to incorporate Discord into the project is that, firstly, it has a Python library and API that can be easily integrated, and secondly, every time an image is uploaded to a server, the return value will be precisely its URL. For these reasons, we have created a Discord server and implemented a Discord bot which is in charge of uploading photos to this server. Once the bot is done, the returned URL will be used for posting to Instagram.

Given the image URL and the caption generated by our models, we can use the aforementioned POST API for obtaining the container id (also known as creation id) and then use it to publish:

```
https://graph.facebook.com/v13.0/{ig-user-id}/media_publish?creation_id=
{creation-id}&access_token={access-token}
```

As a final consideration, to avoid overloading Instagram with too many posts, we have decided that each honeypot will publish two posts per day, one in the morning and one in the evening. Since this project has to manage 21 honeypots, which means 42 posts each day, we have implemented also a python scheduling script that automatically assigns each post to be published, for each time slot, to each honeypot. Then, a second python script is used to publish all posts, for all honeypots, as needed.

Insights Insights are Instagram Graph APIs to get social interaction metrics for an Instagram user or an Instagram media like an image, a video or a carousel.

As for the information for an Instagram user, the user must be a Business or Creator account in order of being able to use insights on it. In fact, they cannot be used to retrieve information on regular users. For this reason, we have implemented these API calls on our own honeypots to retrieve information about the cities, countries, gender and age of the audience for which demographic data are available. Notice that for having demographic data, at least 100 followers are needed.

For the media insights, they represents social interactions on a specific media object. In particular, it is possible to have information about the engagement that is the sum of likes, comments and saved on a specific photo or the total number of unique Instagram accounts that have seen the specific media object.

4.3 Models Implementation

Once the Instagram APIs have been set, the next step of the project was to implement the four models for generating posts: InstaModel, ArtModel, UnsplashModel and QuotesModel. All of them have different characteristics but, at the same time, share some common functionalities such as adding emojis to the generated caption, append CTA/poll/quiz when needed and selecting the hashtag to be used in the post. For this reason, these shared functionalities will be explained before of the actual implementation of the four models.

Shared functionalities As said before, one of the shared functionalities is adding emojis to the generated text. This python script scans the generated caption trying to find out if there are words that can be translated with the corresponding emoji. If

this is the case, then the corresponding emoji is added at the end of the corresponding sentence. To make this script more effective, it looks also for synonyms of nouns and adjectives found in the text to figure out if any of them can be correlated to a particular emoji. As last operation, the script chooses randomly, from a pool of emojis representing the "joy" sentiment, one emoji for each sentence that will be append at the end of each of them. This pool of emojis has been filled for this project manually by identifying emojis that may represents the sentiment joy like the grinning face with smiling eyes emoji or the smiling face with heart-eyes. To implement this script we suggest to use nltk and emoji python libraries(link).

CTA are simple texts that may encourage a user to do actions. These CTA are sampled randomly from a manually compiled list and then added at the end of the generated caption. For quiz and poll the idea is to engage with users hoping that they will give their opinion to the proposed questions. The options in poll and quiz are chosen randomly from other three lists which contain types of cats, food and cars, corresponding to the topic of the honeypot.

The last shared feature is the selection of hashtags. As said before, through the Instagram Graph API we are able to get the first 25 posts for a specific hashtag and usually the honeypot topic is chosen as the hashtag to search for. From these 25 posts all the hashtags contained in the caption are extracted. This hashtag list is used to update the csv file which contains all hashtags found up to that day for the specific topic. As a final result, we have three csv files, one for each topic, which contain all the hashtags found along with the corresponding number of posts that have entered that particular hashtag in the caption. All these hashtags are sorted by the corresponding post number and so each csv file shows all the hashtags related to the topic, from the most used to the least used. From a high-level perspective, they show us which are the most popular hashtags for a specific topic and which are, on the contrary, the most specific hashtags.

Instagram allows to insert at most 30 hashtags in each posts but we think that this number is too high with respect to the normal user's behavior. For this reason, we decided to choose 15 hashtags that are chosen with this criteria: 8 hashtags are sampled randomly from the first half of the list in the csv file, giving more weight to the top ones, while the other 7 are sampled randomly from the second half of the list, giving more weight to the bottom part of the list. The intuition is that we are selecting the most popular hashtags together with more specific hashtags.

InstaModel Starting from the caption generation, InstaModel uses the Instagram Graph API to retrieve the top 25 posts for a specific hashtag. In practice, the chosen hashtag will be the topic on which the corresponding honeypot is based. Once we have all the 25 posts, they are checked to save only those that have an English caption before being passed to the object detector block. The object detector is implemented by using the InceptionV3 model for object detection tasks. The reasoning follows what has been explained in chapter 3: InceptionV3 detects, in the original image, the object classes with the corresponding accuracy and if the first's class score is not greater than or equal to 0.25, the post will be discarded. Otherwise, the other classes are checked as well and only if their scores are greater than 0.05 will be considered as keywords for the next step. Regarding the original caption, nouns and adjectives are extracted by using nltk python library. Notice that words such as "DM" or "credits" and adjectives such as "double" or similar, are not considered. This is because they usually belong to part of the caption that is not useful for this process. We want to capture the real caption and not sentences like "credits to the owner" or "double tap if you like this post".

Keyword2text is the NLP model that transforms a list of keywords in a preliminary sentence. This preliminary sentence is then used by OPT model to generate the complete text. Considering the computational resources available to us, the model used is OPT with 1.3 billion parameters. Indeed, our project has been implemented on a free-to-use version of Google Colab that does not allow to have access to faster GPU/TPU and more RAM, thus preventing us to use larger models.

We suggest to save the text generated by OPT in a file text because it will be used subsequently to generate the corresponding image. Once we have the complete generated text, emojis are added by using the python script described above together with a CTA sentence that is standard in any post. Up to this point, a poll or a quiz can be added as well. The last step for caption generation is to append hashtags: they will be chosen by sampling from the corresponding csv file with the reasoning mentioned above. As last remark for this caption generation phase, the posts are processed altogether and at the end a summary is made to report all the ones that has been used successfully.

The last step of InstaModel is image generation and for this purpose Dall-E Mini (Dayma et al., 2021) is used. The prompt will be the text generated after the OPT stage, the one that has been save separately. Dall-E Mini first generates four images that should correspond to the prompt and, from the one that we choose, a better version of it will be generated subsequently. The better version consists in some detail refinement or object's border smoothing or color adjustments. It is relevant to highlight that the process with Dall-E Mini is not completely automatic. There should be a person that choose the most suitable image for the giving caption. We did not have alternatives other than Dall-E Mini for image generation. As explained in chapter 2, when we have developed these models there were no free use options of Dall-E 2 or Imagen. The only possibility that gave us acceptable results was Dall-E mini.

ArtModel ArtModel starts from a prompt generated with a python script and uses Dall-E mini, like InstaModel, to generate the corresponding image. The style and the medium are chosen randomly from two lists. Example of styles can be "cyberpunk", "psychedelic", "realistic" or "abstract" while examples of medium are "painting", "drawing", "sketch" or "graffiti". The topic of the honeypot is used as subject of the artistic picture generated by Dall-E Mini. Since Dall-E Mini works well when the descriptions provided are as precise as possible, we have decided to use dishes or anything that could represent "food" in more detail rather than the generic word "food". Even in this case all this type of food are chosen randomly from a list. On the other hand, for cat and car, we did not make any changes and topics have been used as they are. Once the image is generated, the prompt, added of emojis, CTA/poll/quiz and the corresponding hashtags, will be used as Instagram caption.

UnsplashModel UnslashModel does not generate images but uses stock images retrieved from the Unsplash websites. Unsplash has been chosen not only because it gives the opportunity to find images together with the relative captions, but also because it offers API for developers that can be used easily. The only requirement is to register on the websites and create an App to obtain the corresponding token.

To search for a photo, the GET API requires a query tag, meaning what we are looking for, and UnsplashModel uses the topic of the honeypot for it. There are other optional parameters that can be set such as how to sort the photos or which photos we want based on their orientation (landscape, portrait, squarish). To avoid

reusing the same images more than once, each image's id is saved in a text file which will be checked at each iteration. For the caption generation, the original caption is processed by Pegasus model (Zhang et al., 2019) which is an NLP model quite good in the rephrase task. As always, emojis, CTA/poll/quiz and hashtags are added to the final result.

QuotesModel QuotesModel makes use of Pixabay(link) stock images website to avoid reusing Unsplash even for this model. Pixabay gives the possibility to use APIs to retrieve images and the only parameter required is the API key that can be obtained once registered on the website. Even in this case, we use the topic of the specific honeypot as query tag. There are optional parameters which can be set such as image type (photo, illustration, vector), orientation, minimum width and minimum height, etc. As for UnsplashModel, to avoid reusing the same image for different posts, once we have downloaded the image, its id is saved in a text file which will be checked every time needed. For the caption generation, a quote is sampled randomly from a citation dataset(link/cit). In this case, the model does not add emojis to the text because we think that the quote, by itself, can be a valid Instagram caption. On the contrary, as always, CTA/poll/quiz and hashtags are added to the text.

4.4 Spamming

In chapter 3 we have already defined what we mean by spamming: honeypots with PLAN 1 or PLAN 2 engagement plans will automatically interact with the posts of other users. The idea is to retrieve the top 25 Instagram posts for the hashtag corresponding to the specific topic of the honeypot and like and comment each of them. In practice, the spamming task is performed by a Bot, implemented in python.

For the Bot implementation we used Selenium which is a tool to automates browsers and it can be easily installed with pip command. Selenium requires a driver to interface with the chosen browser and in our case, since we chose Firefox, we have downloaded the geckodriver (it can be found here (link)). The Bot is nothing more than a python class which has four main methods: `login`, `like_post`, `comment_post` and `exit`.

The login method is invoked when the honeypot accesses to Instagram. It inserts username and password and close all the popups that show up when you login, such has the cookies popup or remember credentials popup. The like_post method searches, in the DOM, for the button corresponding to the like action and then it clicks it. The comment_post method searches in the DOM for the corresponding comment button and then clicks it. Afterwards, it searches for the dedicated textarea and write a random sampled comment. Finally, it clicks the button to send the comment. The exit method allows the Bot to close the browser and terminate the session.

After describing the general functioning of the Bot, there are several considerations that we want to highlight. First of all each operation needs to be interspersed with a random amount of time by using for instance a `sleep` function. This is done for two reasons: the first one is to give time to the DOM to render each element of the webpage, second to avoid being detected by Instagram. In fact, Instagram tries its best to block any kind of spamming activities. Another detail that should be remembered is that, when dealing with DOM and more specifically with Instagram pages, do not use object css classes to identify elements. This classes are often random string that may change in a month or two. For this reason, we identified each

HTML element thanks to text attribute, such as "Add a comment...", or placeholder attributes in the HTML div.

Chapter 5

Results

The main objective of this thesis is to understand which methodologies and technologies should be used for developing and managing social honeypots on Instagram. For this purpose, the collected data will be analyzed in this chapter in order to understand if our initial assumptions were correct or incorrect. To be precise, the initial hypothesis that we have made are: there will be at least one honeypot whose strategy will work better than the others, the employed machine learning technologies can increase the performance of honeypots and, finally, PLAN 2 should be the preferred engagement plan. These considerations have been done by taking into account some factors. First, machine learning technologies nowadays have reached high level results and even if we cannot use the largest models, due to the computational resources required, the overall generated images are quite good especially with inanimate objects and artistic pictures. Furthermore, as PLAN 2 also includes paid strategies, we are convinced that the results with this engagement plan could be better. About the topic, we do not have any clue on which will be the best one since it is the first time that an experiment like that is conducted on Instagram.

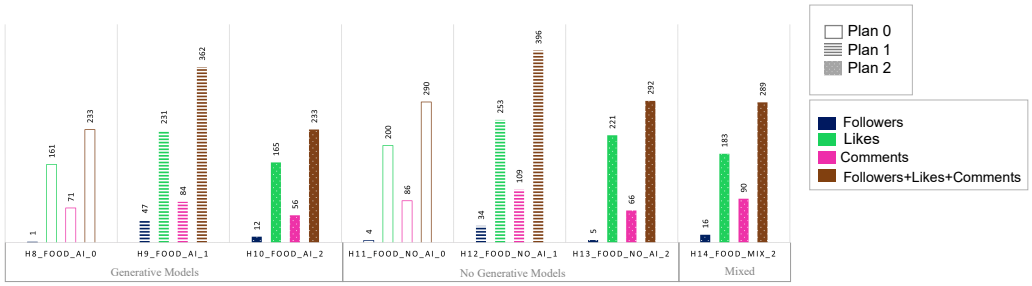
Hence, this discussion will first describe the results obtained in three weeks from a general overview and then move on to a deeper analysis to understand exactly how the experiment is going on. To summarize, the questions that we will try to answer in this chapter are:

- Are we able to generate engagement with our honeypots?
- After three weeks, is there a difference between all the proposed strategies or are they all achieving the same results?
- If there is a difference, are we able to identify the best strategy for building social honeypots on Instagram?
- Is there a significant difference among the four generation models that we have developed?
- Which type of users our social honeypots are able to attract?

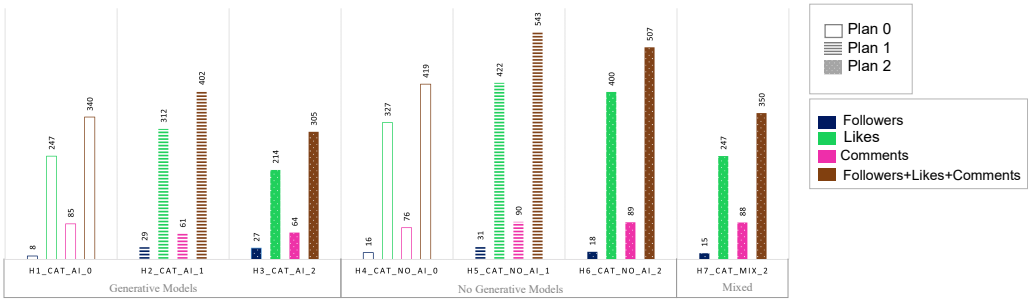
5.1 Quantitative results

1 Are we able to generate engagement with our honeypots?

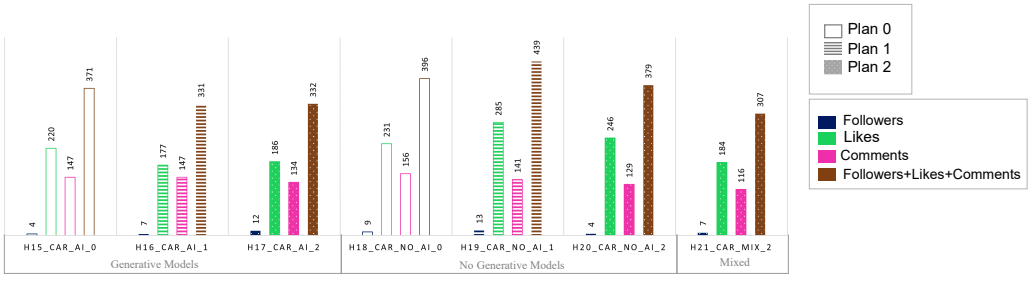
A general overview of the results obtained by honeypots, after three weeks, is shown in Figure 5.1. Precisely, these three graphs report the total amount of likes, comments and followers that honeypots, divided by topic, have earned in three



(A) Followers, likes and comments for honeypots with topic "FOOD".



(B) Followers, likes and comments for honeypots with topic "CAT".



(C) Followers, likes and comments for honeypots with topic "CAR".

FIGURE 5.1: Followers, likes and comments for each honeypot.

weeks. Just by looking at them, it is clear that our honeypots are capable of generating engagement, but we need to understand if they have all behaved the same way or not.

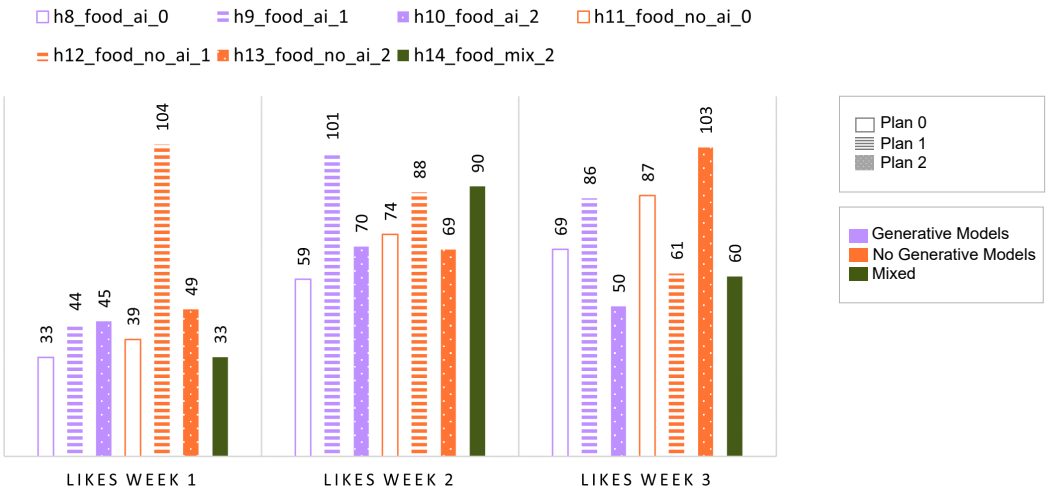
2 After three weeks, is there a difference between all the proposed strategies or are they all achieving the same results?

By looking at the same graphs in Figure 5.1 we can see that not all of them performed at the same level. There are some honeypots that achieved good results while others that were not able to reached considerable values especially for the followers count. For instance, honeypots with PLAN 0 were not able to earned a considerable amount of followers, while those with PLAN 1 achieved higher numbers. Additionally, there are differences in the total amount of engagement generated depending on the topic the honeypots are based on. From just this preliminary overview, we can say that there is a difference among all the strategies implemented and a deeper analysis is required.

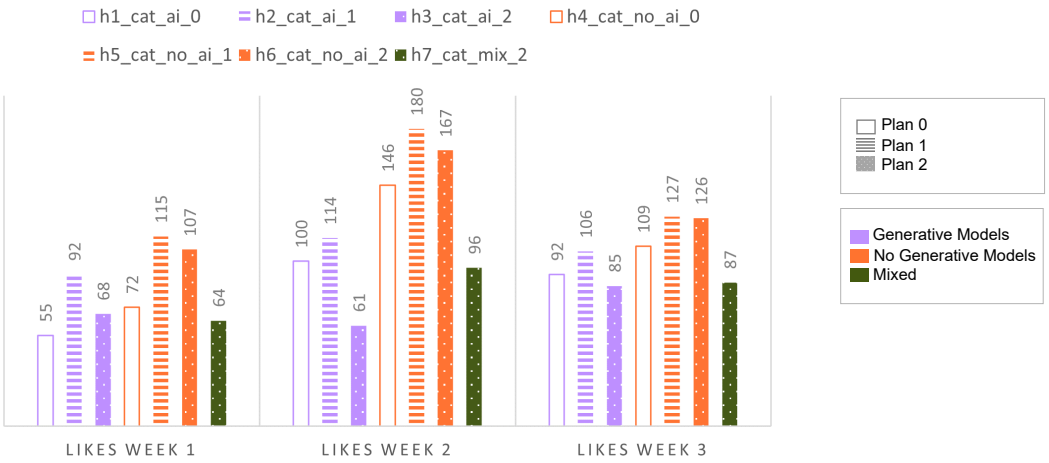
3 If there is a difference, are we able to identify the best strategy for social honeypots on Instagram?

Considering again the same graphs shown previously (Figure 5.1), the general aspect that is in common to all the three graphs is that the amount of likes is higher than the amount of comments and followers. An interpretation for this particular behavior is that for a regular user is easier to give a like to a post, while leaving a comment or starting following a page involve some more efforts. To be clear, likes and comments demonstrate that a specific user likes that specific post or wants to engage with that specific content. Besides, liking requires just a "tap" while commenting involves writing something under the picture. On the contrary, if a user starts following a page, it means that he/she likes the content published by that account and wants to see more of them, not just one post. If we assign a weight to these three actions, obtaining a like is less relevant than gaining a new follower and so we can understand why there was this behavior.

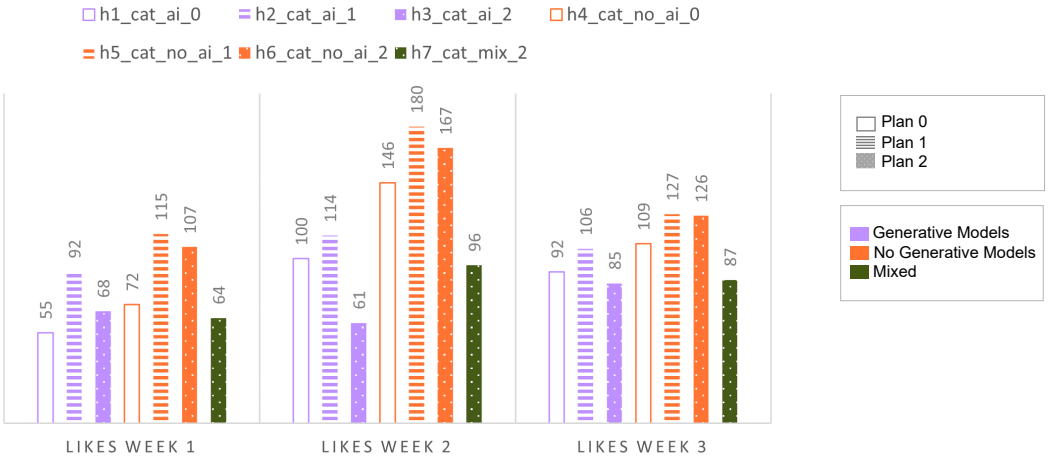
Besides this general aspect, Figure 5.1a reports that, for topic "FOOD", honeypots with no generative models performed better than the others. However, if we consider each honeypot group, PLAN 1 is the engagement plan that has brought the most engagement, both for honeypots with generative models and no generative models. The same study has been conducted for honeypots with topic "CAT" (Figure 5.1b). The first thing that should be noted in this graph is that the amount of total engagement reached by honeypots with topic "CAT" is higher than the total engagement reached by honeypots with a broader topic such as "FOOD". However, this engagement is due mainly to the likes gained during these three weeks, rather than the number of followers. Indeed, honeypots with "FOOD" gained more followers than the ones with "CAT". Furthermore, even in this case honeypots with no generative models achieved the highest values, with the best result from the honeypot with PLAN 1. The last graph in Figure 5.1c shows the engagement generated by honeypots with topic "CAR". The highest values are reached by honeypots with no generative models, with the best results for PLAN 1, that is the same trend as the two previous graphs. The last detail to be highlighted is that even in this case the total amount of engagement is higher with respect to that reached by honeypots with "FOOD". Nevertheless, it is less than that reached by honeypots with topic "CAT".



(A) Amount of likes per week for honeypot with topic "FOOD".



(B) Amount of likes per week for honeypot with topic "CAT".



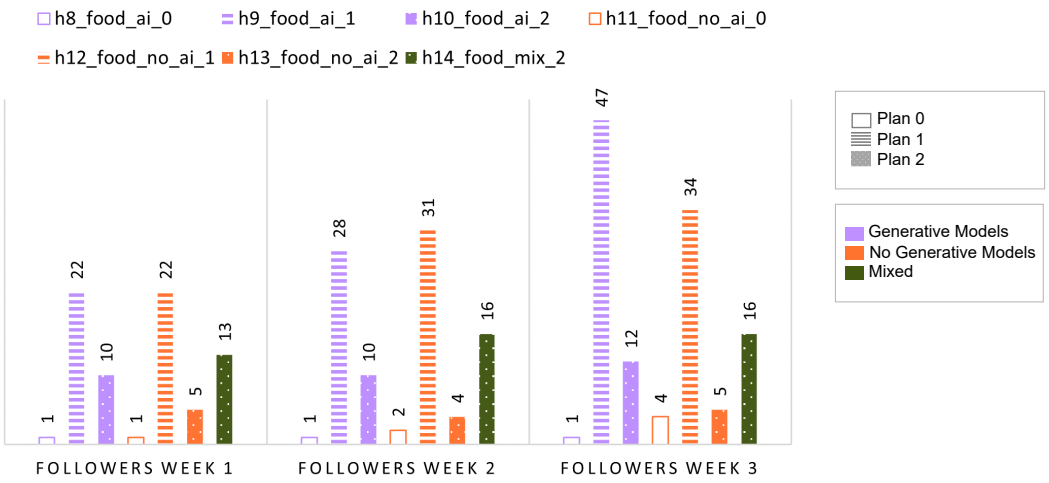
(C) Amount of likes per week for honeypot with topic "CAR".

FIGURE 5.2: Amount of likes gained by each honeypot in each week.

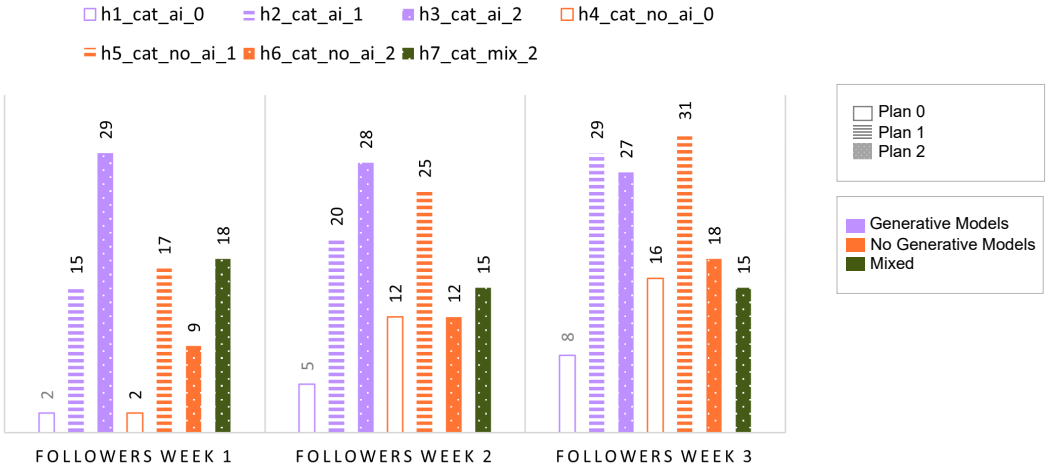
The results shown in these graphs represent the final total engagement achieved, after three weeks, by each honeypot. However, we want to estimate how the engagement has changed in this period of time, not only the final result. The idea is to understand how honeypots, with different strategies, performed in this period of time by looking at the three weeks separately. In particular, we only consider likes and followers because they are the two actions that we believe have very different weights. Starting from likes, Figure 5.2 show the amount of likes gained by each honeypot, in each week. In particular, Figure 5.2a reports the amount of likes that each honeypot with topic "FOOD" has earned in these three weeks. The first thing to notice is that there were more likes in the second week, followed by a slight decrease in the third week. However, if we look in detail, there is no general trend that allows us to identify a specific strategy that worked better than others. In general, honeypots with PLAN 1 and PLAN 2 achieved higher values without major differences between generative and no generative models. For honeypots with the topic "CAT" (Figure 5.2b), it is more evident that those with no generative models and PLAN 1 performed a little better. Even in this case there is no general trend, but the highest number of likes was gained in the second week while in the third week a general slight decrease happened. For "CAR" (Figure 5.2c), the difference between the first week and the subsequent weeks is quite evident. In fact, the amount of likes gained in the first week is less than in the other two periods. Moreover, if we look at the general behavior, we can notice that also here honeypots with no generative models and PLAN 1 performed better than the others.

As said before, we want to understand also how the trends for followers changed in this period of time and, for this purpose, Figure 5.3 shows the cumulative followers earned by each honeypot per week. From Figure 5.3a, we can see that honeypots with "FOOD" have experienced a general increment during the three weeks. This graph supports the results obtained up to now, namely honeypots with PLAN 1 achieved the best results. Furthermore, the increment is faster than the honeypots with PLAN 0 and PLAN 2. To be precise, the honeypot with generative model and PLAN 1 has experienced a very fast increment, ending up to outperform even the honeypot with no generative model in the third week. Actually, this honeypot is the one, among all the 21, that earned the highest amount of followers. For "CAT", Figure 5.3b reports a general increment for honeypots with PLAN 0 and PLAN 1. Looking only at honeypots with PLAN 1, the no generative ones obtained higher values than the generative ones. Notice that the honeypot with generative models and PLAN 2, despite reaching a very high number in the first week, was subject to a slight decrease during the other two weeks. As last case, Figure 5.3c shows the number of followers gained in three weeks by honeypots with "CAR" as topic. While in the previous results, there was a clear trend definition, in this case is not possible to find a general behavior. Some honeypots decreased the number of followers during these three weeks, while other increased. Nevertheless, it should be noted that honeypots with PLAN 1 and no generative models achieved quite good results.

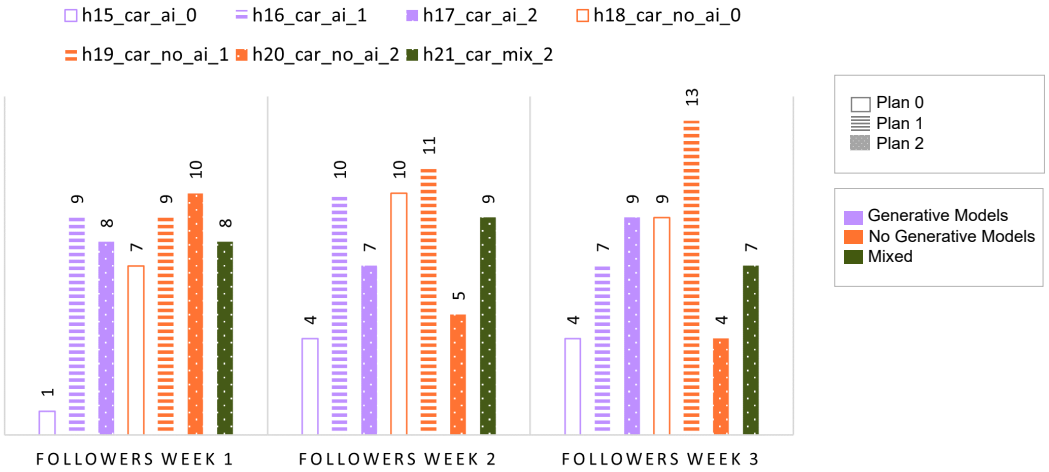
It seems that PLAN 1 works especially well with no generative models but to have a deeper insight about each engagement plan, the graphs in Figure 5.4 show the follower trend for each plan separately. Looking in details, PLAN 0 (Figure 5.4a) reports a general increment for honeypots with "CAT" and an almost constant behavior for honeypots with "FOOD" and "CAR" as topic. In addition, from these trends it is clear that honeypots with no generative models performed a little better than the ones with generative models. Different results can be observed in Figure 5.4b in which honeypots with PLAN 1 and topics "FOOD" and "CAT" experienced a very fast increment. On the other hand, honeypots with "CAR" do not seem to report a



(A) Cumulative followers per week for honeypot with topic "FOOD".

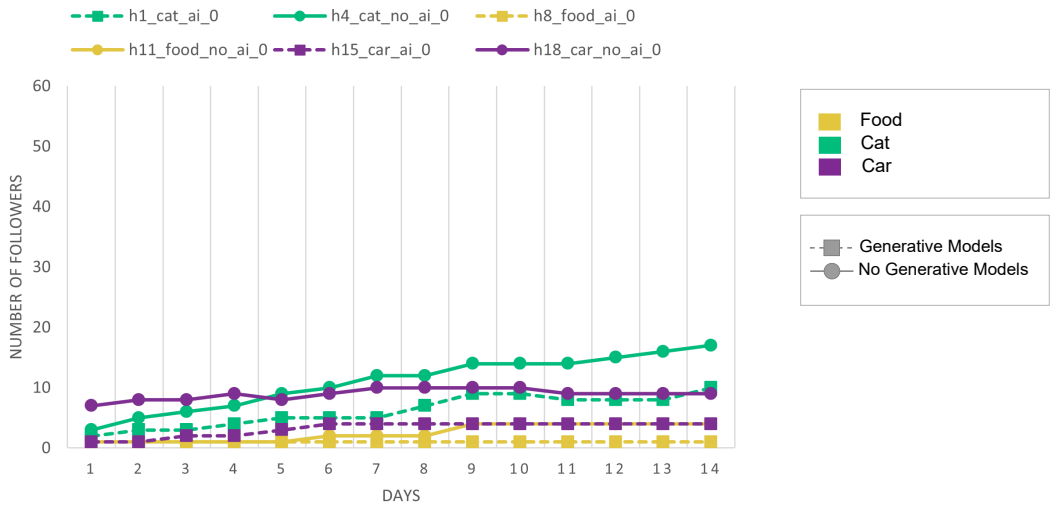


(B) Cumulative followers per week for honeypot with topic "CAT".

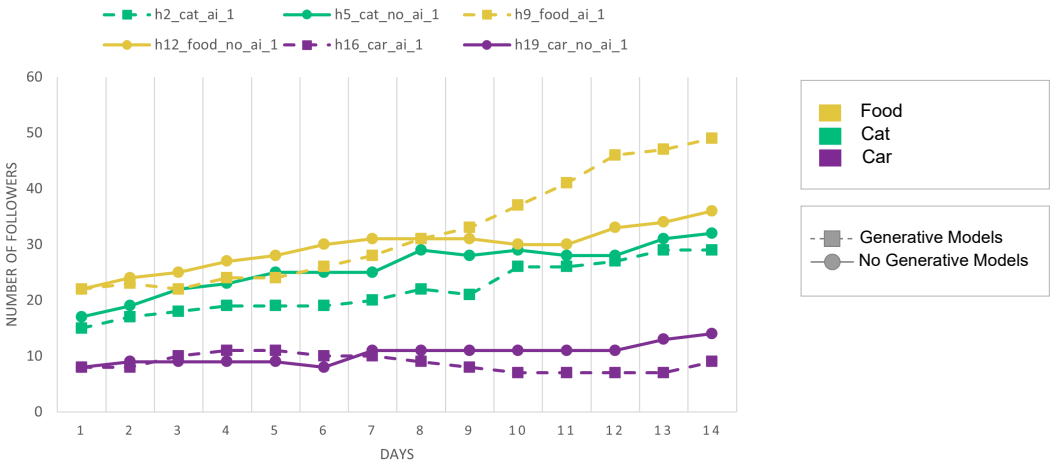


(C) Cumulative followers per week for honeypot with topic "CAR".

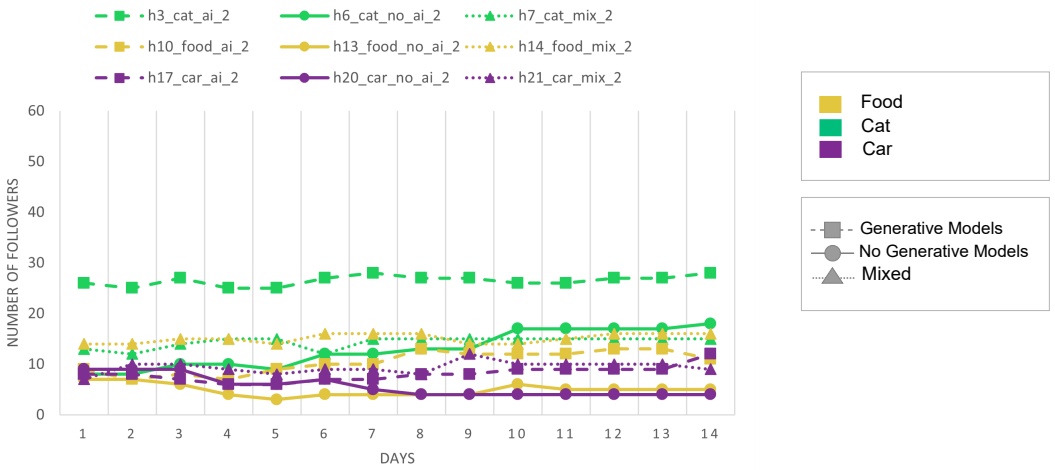
FIGURE 5.3: Cumulative followers per week.



(A) Honeypots with PLAN 0.



(B) Honeypots with PLAN 1.



(C) Honeypots with PLAN 2.

FIGURE 5.4: Followers trend per engagement plan.

relevant behavior. In addition, no generative models achieved better results than the generative models as in the previous graph.The only exception is for "FOOD" where the results are reversed. This honeypot seems to contradict the idea that honeypots with no generative models and PLAN 1 are preferred over generative ones. Looking at Figure 5.4c, while honeypots with PLAN 1 have seen a general increment, honeypots with PLAN 2 seem to remain constant and do not show any type of increment or decrement as for PLAN 0. There is one honeypot that outperformed the others that is the one with "CAT" and generative models. However, even in this case its trend is quite constant.

After this study, it is clear that honeypots with no generative models and PLAN 1, in particular for medium or specific topics, should be preferred over the others. Still, there is this exception for topic "FOOD" that we cannot define precisely. For this reason, we performed some statistical analysis in order to have a clear answer to the main question on which is based this experiment. In particular, we performed two tests, Three-Way Anova and Tukey. The Three-Way ANOVA test is used to determine whether there is a relationship among variables on an outcome. It is often used for gaining knowledge of complex interactions where more than one variable may influence the results. More specifically, it is able to determine whether the variability of the outcomes is due to chance or to the factors in the analysis. On the other hand, Tukey test can be used to find means that are significantly different from each other. It is a pairwise comparison that is able to identify if there is a significant difference between groups of data. After checking the normality of our samples, we used these two tests to understand if one factor, among topic, post generation strategy and engagement plan, can influence the results obtained with the 21 honeypots and in which way. Hence, Table 5.1 shows us the results for the Three-Way Anova test and it tells us that all the three factors influence the data collected in these three weeks. Moreover, Table 5.2 reports the results obtained with the Tukey test and it is now evident how these factors influence the outcome. This last test confirms what we have seen up to now, namely honeypots with no generative models, PLAN 1 and a medium/specific topic are able to achieved the best results. This also demonstrates that the honeypot with topic "FOOD", which seemed to contradict the general results, can be classified as an outlier value that does not change the general conclusions we have reached.

4 Is there a significant difference among the four generation models that we have developed?

Up to now we have understood that no generative models should be preferred over generative models. However, these two categories have models that work differently from each other. We are interested not only in understanding which strategy should be preferred over the other but also to identify which model, at Instagram post level, is able to develop more engagement considering all the 21 honeypots. If our assumption is correct, one among QuotesModel and UnsplashModel should result as the best one or, better, the one that is able to gain more engagement in general. For this reason we have recorded which post obtained more likes and comments in the first, second and third week together with which model has been used to generate that specific post, for all the 21 honeypots.

Table 5.3 shows the results and, from a preliminary analysis, we can observe that the highest values were indeed achieved by QuotesModel, followed by UnsplashModel. This is perfectly inline with the results obtained in the previous analysis.

TABLE 5.1: Anova test with three factors: topic, model, plan.
If the P-Value is less than the significance level (0,05) than the factor influences the data samples.

Source	DF	Adj SS	Adj MS	F-Value	**P-Value**
Topic	2	15723	7861,6	7,85	**0,001**
Model	1	10031	10031,4	10,01	**0,003**
Plan	2	11582	5791,2	5,78	**0,006**

TABLE 5.2: Tukey test with three factors: topic, model, plan.
Means that do not share a letter are significant different.

Models	N	Mean	Grouping	
No Generative	27	142,667	A	
Generative	27	115,407		B

Plan	N	Mean	Grouping	
1	18	149,556	A	
2	18	121,222		B
0	18	116,333		B

Topic	N	Mean	Grouping	
Cat	18	149,556	A	
Car	18	129,778	A	B
Food	18	107,778	B	

However, by observing carefully Table 5.3, we can also understand that, if we consider only honeypots with generative models, InstaModel gained more engagement with respect to ArtModel. Actually, the values obtained by InstaModel are quite close with the ones obtained with UnsplashModel. This is an unexpected behavior that has led us to investigate this aspect in order to understand what is happening. Thus, to have a better insight, we plotted the four distributions in Figure 5.5 which shows another relevant result: QuotesModel, UnsplashModel and InstaModel have quite similar median and shape, in contrast with ArtModel. Furthermore, QuotesModel has several higher values that stretch the shape even more. From this graph we cannot clearly identify which model is the best, perhaps QuotesModel due to its shape, but we can certainly say that ArtModel is the one that has earned the least likes and comments and consequently the least appreciated one.

5.2 Qualitative results

5 Which type of users our social honeypots are able to attract?

The Instagram Graph API that we have used in this project allows us to obtain insight and qualitative results about our honeypots. However, these statistics can be obtained only if the honeypot has at least 100 followers. This means that we cannot show now these insights since none of ours honeypots achieved 100 followers in three weeks. However, we can give some descriptive feature that we have noticed during this period of time. One of the most interesting and unexpected result is that, without the premeditate intention of building spammer detectors, the majority

TABLE 5.3: Maximum number of likes and comments, together with the corresponding model, obtained by each honeypot, per week.

	Week 1		**Week 2**		**Week 3**	
	Likes + Comments	*Model*	*Likes + Comments*	*Model*	*Likes + Comments*	*Model*
h1_cat_ai_0	9	ArtModel	9	InstaModel	**14**	**InstaModel**
h2_cat_ai_1	13	InstaModel	11	ArtModel	13	InstaModel
h3_cat_ai_2	9	ArtModel	7	InstaModel	9	InstaModel
h4_cat_no_ai_0	10	QuotesModel	16	QuotesModel	11	QuotesModel
h5_cat_no_ai_1	**19**	**QuoteModel**	**22**	**QuotesModel**	13	UnsplashModel
h6_cat_no_ai_2	13	UnsplahModel	12	QuotesModel	**14**	**UnsplashModel**
h7_cat_mixed_2	7	UnsplashModel	9	InstaModel	8	QuotesModel
h8_food_ai_0	7	InstaModel	6	ArtModel	9	InstaModel
h9_food_ai_1	8	ArtModel	11	InstaModel	10	InstaModel
h10_food_ai_2	10	InstaModel	**12**	**InstaModel**	**16**	**ArtModel**
h11_food_no_ai_0	8	UnsplashModel	10	UnsplashModel	10	QuotesModel
h12_food_no_ai_1	9	QuotesModel	**12**	**UnsplashModel**	11	QuotesModel
h13_food_no_ai_2	**11**	**QuotesModel**	8	UnsplashModel	10	QuotesModel
h14_food_mixed_2	6	InstaModel	10	UnsplashModel	8	QuotesModel
h15_car_ai_0	7	ArtModel	10	InstaModel	8	InstaModel
h16_car_ai_1	**12**	**InstaModel**	10	InstaModel	6	ArtModel
h17_car_ai_2	6	ArtModel	9	ArtModel	12	ArtModel
h18_car_no_ai_0	7	QuotesModel	**15**	**QuotesModel**	9	UnsplashModel
h19_car_no_ai_1	7	UnsplashModel	12	QuotesModel	10	QuotesModel
h20_car_no_ai_2	**12**	**QuotesModel**	11	UnsplashModel	**13**	**UnsplashModel**
h21_food_mixed_2	7	QuotesModel	14	QuotesModel	6	QuotesModel

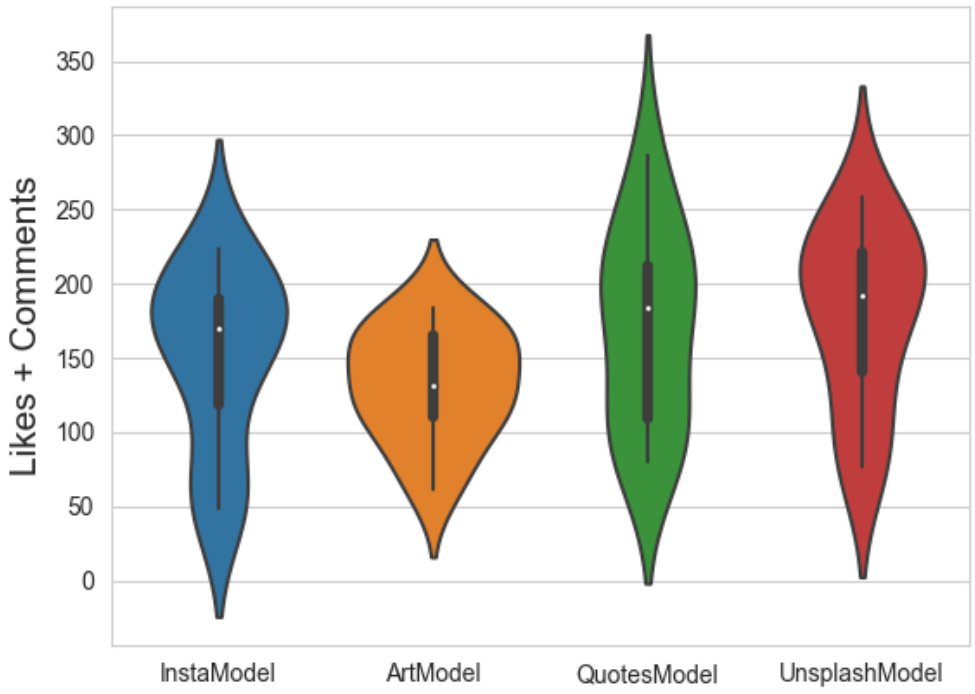


FIGURE 5.5: Distributions of likes and comments earned by each model.

of comments and requests that we have received were from spammers. All these spammer accounts shared similar behavior, namely they used specific words such as "send pic" or "DM me" and were, most of the time, the first comments that we received after publishing a new post. This may indicate that these bots target the most recent published posts, perhaps searching by specific hashtags. This side effect tells us that our honeypots may already be used as spammer detectors.

Due to the lack of information about our followers, we cannot understand completely which accounts we have attracted and their characteristics. Nevertheless, we investigated manually the followers earned by the three honeypots with no generative models and PLAN 1 in order to have an high level overview. In general, about 50% of followers are Instagram pages that has the same topic as ours. For example, the followers of the honeypot "CAT" are mostly Instagram pages whose content consists of cat images/videos. The same has been observed for the other two topics. Despite this trend, there are also some personal accounts, about 40% of the total amount of followers, that resemble real person having a real interest in cats, cars or food. As side effect, there are also accounts with an high probability of being fake accounts/bots but they are fewer with respect to the others two categories, around 10% of the total count.

5.3 Discussion

Contrary to what we expected, these results showed that some of our initial assumptions were incorrect. For instance, we thought that machine learning technologies could make a difference and that paid engagement plans would have generated more engagement than other plans. However, these assumptions have been contradicted by the results reported in this chapter and, thinking about a possible explanation, we can find several reasons why they were wrong. First of all, perhaps machine learning techniques are not ready for this type of experiments on Instagram or maybe the fact that we were forced to use smaller models has affected the final results. Regarding the engagement plan, perhaps we are losing fake accounts while acquiring new real followers and this process did not allow these honeypots to rank up in Instagram algorithms since there is a no clear trend.

we now summarize the outcomes from the different research questions:

1 Are we able to generate engagement with our honeypots?

Yes, our honeypot are able to generate engagement.

2 After three weeks, is there a difference between all the proposed strategies or are they all achieving the same results?

There is a difference since not all the honeypots performed at the same level in terms of likes, comments and followers earned.

3 If there is a difference, are we able to identify the best strategy for building social honeypots on Instagram?

To be effective, a social honeypot on Instagram should:

- Social honeypots' performance are impacted by the topic chosen.

- simple generative process should be preferred to advanced ones such as generative models.
- behave as similar as possible to what a regular account would do without the necessary use of paid strategies.

4 Is there a significant difference among the four generation models that we have developed?

Yes, using quotes as captions seems to be the best option while artistic pictures are the less appreciated.

5 Which type of users our social honeypots are able to attract?

From a preliminary study, honeypots were able to attract spammers, other pages with similar content, and a smaller percentage of real users.

Chapter 6

Conclusions

This thesis aims to demonstrate how social honeypots can be used on Instagram and how to automate them. Social honeypots are intended in this work as Instagram accounts able to lure other users for many different motivations. One of these is to fight against malicious activities that use Instagram to carried out other more dangerous attacks. However, it is not so difficult to imagine social honeypots for purposes that do not belong explicitly to cybersecurity. For instance, they can also be used to understand how the society is evolving and which are nowadays the interests of people, or to profile the audience of a company to promote a new product.

The fact that social honeypots have a wide usage options, makes them extremely flexible and efficient for any needs. However, most of the time they require human intervention to be managed or to record users' activities. The research that we have carried out in this thesis proposes a new way to see at this powerful tool by making it completely automatic. In addition, the latest state-of-the-art Machine Learning techniques were also used in the implementation of these honeypots to understand whether their use helps in improving their performance or not. To be precise, the best achieved in Computer Vision, Natural Language Processing and Image generation techniques has been used to develop content generation strategies able to generate both the image and the caption needed for the classical Instagram post.

Since it is a totally unexplored way of intending social honeypots, the idea that led this project from the beginning was to get some answers to many of the questions that we may ask for. For instance, we wanted to know how a social honeypot behaves on a Online Social Networks as Instagram and which are the characteristics that it should have to perform well. We wanted to understand if our assumption that machine learning could be a plus in this kind of tasks was correct or it is too premature to use it in this type of scenario. For all these reasons we have developed 21 social honeypots, deployed on Instagram, that are differentiated by the topic on which they are based, the post generation strategy and the engagement plan.

Three different topics, based on their coverage on Instagram, have been chosen and four post generation models have been implemented. Two of them are of the generative type, in the sense that they generate the caption and the final image in an automatic manner by using machine learning techniques. The other two make use of already existing images, such as stock images, and modified caption or quotes to make the classical Instagram post. Three engagement plans have been identify, starting from the simplest one, which includes standard call-to-actions, to the more complex one that involves the use of paid strategies such as buying followers or content promotion. Two of these engagement plans involve also spamming techniques, meaning that social honeypots are able to leave likes and comments to the other users' post in a totally automatic way without the need to spend time on searching related post or thinking to which comment to leave.

In only three weeks we were able to demonstrate that indeed social honeypots on Instagram are possible and they can be customized by choosing the appropriate topic for the specific purpose. Our results have shown that social honeypots with no generative models and a behavior that resemble that of a normal user, thus posting content and a simple engagement activity without the help of pay strategies, are more effective than the others. We also have noticed that these social honeypots could already be used as spammer detectors.

One detail we want to clarify is that honeypots with PLAN 2 have not yet used sponsored content. This is because they have only been online for three weeks and we believe it is too early to implement this strategy. However, this method will be certainly employed, in the subsequent weeks, to analyse its impact. Furthermore, as future work, it will be interesting to implement social honeypots equipped with bigger machine learning models and to try them in different scenarios, maybe with more complex topics. We would like also to implement additional features and capabilities such as being able to respond to comments automatically or to share stories.

As last consideration, this study was thought to be a long-term study, thus the 21 social honeypots should be active for at least two or three months to have solid results. This means that the experiments are still going on, during the writing of this thesis, and will continue even after the official graduation date. This is because we want to keep analysing the behavior of the different social honeypots as a whole and understand deeply which factors will influence the final results. With all this in mind, we just have to conclude by saying that the efforts put in this work have led to important results, even if still preliminary, which will certainly open up to new researches and solutions.

Bibliography

Alexa (2022). *Alexa Top Websites*. https://www.expireddomains.net/alexa-top-websites/. Accessed: 2022-08-30.

Brown, Tom et al. (2020). "Language models are few-shot learners". In: *Advances in neural information processing systems* 33, pp. 1877–1901.

Caverlee, James (2008). "A large-scale study of MySpace: Observations and implications for online social networks". In: *Proceedings of the International AAAI Conference on Web and Social Media*. Vol. 2. 1, pp. 36–44.

Chowdhary, K. R. (2020). "Natural Language Processing". In: *Fundamentals of Artificial Intelligence*. New Delhi: Springer India, pp. 603–649. ISBN: 978-81-322-3972-7. DOI: 10.1007/978-81-322-3972-7_19. URL: https://doi.org/10.1007/978-81-322-3972-7_19.

Chung, Junyoung et al. (2015). "Gated feedback recurrent neural networks". In: *International conference on machine learning*. PMLR, pp. 2067–2075.

Dayma, Boris et al. (July 2021). *DALL·E Mini*. DOI: 10.5281/zenodo.5146400. URL: https://github.com/borisdayma/dalle-mini.

De Cristofaro, Emiliano et al. (2014). "Paying for likes? understanding facebook like fraud using honeypots". In: *Proceedings of the 2014 Conference on Internet Measurement Conference*, pp. 129–136.

Dhariwal, Prafulla and Alexander Nichol (2021). "Diffusion models beat gans on image synthesis". In: *Advances in Neural Information Processing Systems* 34, pp. 8780–8794.

Dittrich, David (2015). "The ethics of social honeypots". In: *Research Ethics* 11.4, pp. 192–210.

Franke, Richard Herbert and James D Kaul (1978). "The Hawthorne experiments: First statistical interpretation". In: *American sociological review*, pp. 623–643.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. http://www.deeplearningbook.org. MIT Press.

Goodfellow, Ian et al. (2014). "Generative adversarial nets". In: *Advances in neural information processing systems* 27.

Hanson, Carl L et al. (2013). "Tweaking and tweeting: exploring Twitter for nonmedical use of a psychostimulant drug (Adderall) among college students". In: *Journal of medical Internet research* 15.4, e2503.

He, Kaiming et al. (2015). "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification". In: *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034.

Ho, Jonathan, Ajay Jain, and Pieter Abbeel (2020). "Denoising diffusion probabilistic models". In: *Advances in Neural Information Processing Systems* 33, pp. 6840–6851.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long short-term memory". In: *Neural computation* 9.8, pp. 1735–1780.

Hu, Xia, Jiliang Tang, and Huan Liu (2014). "Online social spammer detection". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 28. 1.

Ioffe, Sergey and Christian Szegedy (2015). "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *International conference on machine learning*. PMLR, pp. 448–456.

Joshi, Prateek and C-C Jay Kuo (2011). "Security and privacy in online social networks: A survey". In: *2011 IEEE international conference on multimedia and Expo*. IEEE, pp. 1–6.

Karl (2022). *The 15 Biggest Social Media Sites and Apps*. https://www.dreamgrow.com/top-15-most-popular-social-networking-sites/. Accessed: 2022-09-11.

Lee, Kyumin, James Caverlee, and Steve Webb (2010). "Uncovering social spammers: social honeypots+ machine learning". In: *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pp. 435–442.

Lee, Kyumin, Brian Eoff, and James Caverlee (2011). "Seven months with the devils: A long-term study of content polluters on twitter". In: *Proceedings of the international AAAI conference on web and social media*. Vol. 5. 1, pp. 185–192.

Lin, Tsung-Yi et al. (2014). "Microsoft coco: Common objects in context". In: *European conference on computer vision*. Springer, pp. 740–755.

Liu, Peter J et al. (2018). "Generating wikipedia by summarizing long sequences". In: *arXiv preprint arXiv:1801.10198*.

Moor, James (2006). "The Dartmouth College artificial intelligence conference: The next fifty years". In: *Ai Magazine* 27.4, pp. 87–87.

Murugan, N Senthil and G Usha Devi (2018). "Detecting spams in social networks using ML algorithms-a review". In: *International Journal of Environment and Waste Management* 21.1, pp. 22–36.

Nichol, Alex et al. (2021). "Glide: Towards photorealistic image generation and editing with text-guided diffusion models". In: *arXiv preprint arXiv:2112.10741*.

Nisrine, Maqrane et al. (2016). "A security approach for social networks based on honeypots". In: *2016 4th IEEE International Colloquium on Information Science and Technology (CiSt)*. IEEE, pp. 638–643.

Paperno, Denis et al. (2016). "The LAMBADA dataset: Word prediction requiring a broad discourse context". In: *arXiv preprint arXiv:1606.06031*.

Quercia, Daniele et al. (2011). "In the mood for being influential on twitter". In: *2011 IEEE third international conference on privacy, security, risk and trust and 2011 IEEE third international conference on social computing*. IEEE, pp. 307–314.

Radford, Alec et al. (2018). "Improving language understanding by generative pre-training". In.

Radford, Alec et al. (2019). "Language models are unsupervised multitask learners". In: *OpenAI blog* 1.8, p. 9.

Radford, Alec et al. (2021). "Learning transferable visual models from natural language supervision". In: *International Conference on Machine Learning*. PMLR, pp. 8748–8763.

Ramesh, Aditya et al. (2021). "Zero-shot text-to-image generation". In: *International Conference on Machine Learning*. PMLR, pp. 8821–8831.

Ramesh, Aditya et al. (2022). "Hierarchical text-conditional image generation with clip latents". In: *arXiv preprint arXiv:2204.06125*.

Richter, Felix (2022). *Social Networking Is the No. 1 Online Activity in the U.S.* https://www.statista.com/chart/1238/digital-media-use-in-the-us/. Accessed: 2022-09-11.

Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams (1986). "Learning representations by back-propagating errors". In: *nature* 323.6088, pp. 533–536.

Sheikhi, Saeid (2020). "An Efficient Method for Detection of Fake Accounts on the Instagram Platform." In: *Rev. d'Intelligence Artif.* 34.4, pp. 429–436.

Simonyan, Karen and Andrew Zisserman (2014). "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556*.

Sohl-Dickstein, Jascha et al. (2015). "Deep unsupervised learning using nonequilibrium thermodynamics". In: *International Conference on Machine Learning*. PMLR, pp. 2256–2265.

Song, Yang and Stefano Ermon (2019). "Generative modeling by estimating gradients of the data distribution". In: *Advances in Neural Information Processing Systems* 32.

Stallings, William et al. (2012). *Computer security: principles and practice*. Vol. 2. Pearson Upper Saddle River.

Stringhini, Gianluca, Christopher Kruegel, and Giovanni Vigna (2010). "Detecting spammers on social networks". In: *Proceedings of the 26th annual computer security applications conference*, pp. 1–9.

Szegedy, Christian et al. (2015). "Going deeper with convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9.

Szegedy, Christian et al. (2016). "Rethinking the inception architecture for computer vision". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826.

Vaswani, Ashish et al. (2017). "Attention is all you need". In: *Advances in neural information processing systems* 30.

Webb, Steve, James Caverlee, and Calton Pu (2008). "Social Honeypots: Making Friends With A Spammer Near You." In: *CEAS*. San Francisco, CA, pp. 1–10.

Yang, Chao, Jialong Zhang, and Guofei Gu (2014). "A taste of tweets: Reverse engineering twitter spammers". In: *Proceedings of the 30th annual computer security applications conference*, pp. 86–95.

Zhang, Caiming and Yang Lu (2021). "Study on artificial intelligence: The state of the art and future prospects". In: *Journal of Industrial Information Integration* 23, p. 100224.

Zhang, Caiming, Xiaojun Xu, and Hong Chen (2020). "Theoretical foundations and applications of cyber-physical systems: a literature review". In: *Library Hi Tech*.

Zhang, Jingqing et al. (2019). *PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization*. arXiv: 1912.08777 [cs.CL].

Zhang, Susan et al. (2022). "Opt: Open pre-trained transformer language models". In: *arXiv preprint arXiv:2205.01068*.

Zhang, Yihe, Hao Zhang, and Xu Yuan (2019). "Toward Efficient Spammers Gathering in Twitter Social Networks". In: *Proceedings of the Ninth ACM Conference on Data and Application Security and Privacy*, pp. 157–159.

Zhou, Yufan et al. (2021). "Lafite: Towards language-free training for text-to-image generation". In: *arXiv preprint arXiv:2111.13792*.

Zhu, Yin et al. (2012). "Discovering spammers in social networks". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 26. 1, pp. 171–177.