entropy

MDPI

Article

DNN Intellectual Property Extraction Using Composite Data

Itay Mosafi [1,*], Eli (Omid) David [1], Yaniv Altshuler [2] and Nathan S. Netanyahu [1,3]

[1] Department of Computer Science, Bar-Ilan University, Ramat-Gan 5290002, Israel; mail@elidavid.com (E.D.); nathan@cs.biu.ac.il (N.S.N.)
[2] MIT Media Lab, 77 Mass. Ave., E14/E15, Cambridge, MA 02139-4307, USA; yanival@media.mit.edu
[3] Department of Computer Science, College of Law and Business, Ramat-Gan 5257346, Israel
* Correspondence: itay.mosafi@gmail.com

Abstract: As state-of-the-art deep neural networks are being deployed at the core level of increasingly large numbers of AI-based products and services, the incentive for "copying them" (i.e., their intellectual property, manifested through the knowledge that is encapsulated in them) either by adversaries or commercial competitors is expected to considerably increase over time. The most efficient way to extract or steal knowledge from such networks is by querying them using a large dataset of random samples and recording their output, which is followed by the training of a *student* network, aiming to eventually mimic these outputs, without making any assumption about the original networks. The most effective way to protect against such a mimicking attack is to answer queries with the classification result only, omitting confidence values associated with the softmax layer. In this paper, we present a novel method for generating composite images for attacking a *mentor* neural network using a student model. Our method assumes no information regarding the mentor's training dataset, architecture, or weights. Furthermore, assuming no information regarding the mentor's softmax output values, our method successfully mimics the given neural network and is capable of stealing large portions (and sometimes all) of its encapsulated knowledge. Our student model achieved 99% relative accuracy to the protected mentor model on the Cifar-10 test set. In addition, we demonstrate that our student network (which copies the mentor) is impervious to watermarking protection methods and thus would evade being detected as a stolen model by existing dedicated techniques. Our results imply that all current neural networks are vulnerable to mimicking attacks, even if they do not divulge anything but the most basic required output, and that the student model that mimics them cannot be easily detected using currently available techniques.

Keywords: deep learning; cybersecurity; artificial intelligence; swarm intelligence; adversarial AI; information theory; entropy; models; neural networks; communication

Citation: Mosafi, I.; David, E.; Altshuler, Y.; Netanyahu, N.S. DNN Intellectual Property Extraction Using Composite Data. *Entropy* **2022**, *24*, 349. https://doi.org/10.3390/e24030349

Academic Editor: Stanisław Drożdż

Received: 29 December 2021
Accepted: 21 February 2022
Published: 28 February 2022

1. Introduction

In recent years, deep neural networks (DNNs) have been used very effectively in a wide range of applications. Since these models have achieved remarkable results, redefining state-of-the-art solutions for various problems, they have become the "go-to solution" for many challenging real-world problems, e.g., object recognition [1,2], object segmentation [3], autonomous driving [4], automatic text translation [5], cybersecurity [6–8], credit default prediction [9], etc.

Training a state-of-the-art deep neural network requires designing the network architecture, collecting and preprocessing data, and accessing hardware resources, in particular graphics processing units (GPUs) capable of training such models. Additionally, training such networks requires a substantial amount of trial and error. For these reasons, such trained models are highly valuable, but at the same time, they could be the target of attacks by adversaries (e.g., a competitor) who might try to duplicate the model and the entire sensitive intellectual property involved without going through the tedious and expensive process of developing the models by themselves. By doing so, all the trouble of data

collection, acquiring computing resources, and the valuable time required for training the models are spared by the attacker. As state-of-the-art DNNs are used more extensively in real-world products, the prevalence of such attacks is expected to increase over the next few years.

An attacker has two main options for acquiring a trained model: (1) acquiring the raw model from the owner's private network, which would be a risky criminal offense that requires a complicated cyber attack on the owner's network; and (2) training a student model that mimics the original mentor model. That is, the attacker could query the original mentor using a dataset of samples and train the student model to mimic the output of the mentor model for each of the samples. The second option assumes that the mentor is a black box, i.e., there is no knowledge of its architecture, no access to the training data used for training it, and no information regarding the trained model's weights. We only have access to the model's predictions (inference) for a given input. Thus, such a mentor would effectively teach a student how to mimic it by providing its output for different inputs.

In order for mimicking to succeed, a key element is to utilize the certainty level of a model on a given input, i.e., its softmax distribution values [10,11]. This knowledge is highly important for the training of the student network. For example, in case of a binary classification, classifying an image as category i with 99% confidence and as category j with 1% confidence is much more informative than classifying it to category i with, say, 51% confidence and to category j with 49% confidence. Such data are valuable and often much more informative than the predicted category alone, which in both cases is i. This confidence value (obtained through the softmax output layer) also reveals how the model perceives this specific image and to what extent the predictions for categories i and j are similar. In order to protect against such a mimicking attack, a trained model may hide this confidence information by simply returning only the index with the maximal confidence, without providing the actual confidence levels (i.e., the softmax values are concealed, while the output contains merely the predicted class). Although such a model would substantially limit the success of a student model using a standard mimicking attack, we provide in this paper a novel method by querying the mentor with *composite* images, such that the student effectively elicits the mentor's knowledge, even if the mentor provides the predicted class only.

Contributions: This research possesses various contributions to the domain of DNN intellectual property extraction.

1. It is possible to extract the intellectual property of a model with no access to the original data (inputs and labels) used for training it.
2. All classification models are vulnerable, maximum protection of the model was assumed, and still, the composite method managed to extract the intellectual property.
3. A novel composite method using unlabeled data was described for knowledge extraction, which can be applied on any model as long as unlabeled data are available.
4. The state-of-the-art watermarking methods are not able to identify a student model once it contains the knowledge of the mentor model, which was protected using watermarks.

The rest of the paper is organized as follows. Section 2 reviews previous methods used for network distilling and mimicking. Section 3 describes our new approach for a successful mimicking attack on a mentor, which does not provide softmax outputs. Section 4 presents our experimental results. Finally, Section 5 makes concluding remarks. This paper is based on a preliminary version published in [12].

2. Background

2.1. Threats to Validity

We included the studies that (1) deal with methods to attack machine learning or deep learning models, (2) protect models' intellectual property from attacks or provide methods to identify stolen models, and (3) discuss the mentor–student training schema and its limitations, such as the number of layers reduction, speedup gain, and accuracy

reduction. We have used multiple combination strings such as 'DNN distillation', 'mentor student training', 'teacher student training', 'DNN attacks', 'machine learning attacks', 'watermarking in DNN', 'DNN protection', 'DNN intellectual property', and 'ML and DL models protection' to retrieve the peer-reviewed articles of journal conference proceedings, book chapters, and reports. We have targeted the five databases, namely IEEE Xplore, SpringlerLink, Scopus, arXiv digital library, and ScienceDirect. Google Scholar was also largely used for searching and tracking cited papers based on the topics of interest. The title and abstract were screened to identify potential articles; then, the experimental results were carefully reviewed in order to identify relevant baselines and successful methods.

2.2. Motivation

There already exist secondary markets for the resale of stolen identities, such as www.infochimps.com (accessed on 19 November 2021) or black market sites and chat rooms for the resale of other illegal datasets [13,14]. It also reasonable to assume that a digested "learned" data would be worth more to such buyers than the raw data itself, and that models learned through the use of more data and higher computational resources might be priced differently than more basic ones. After all, why work hard when one can employ the high-quality results of a learning process executed by others [15–18]?

We note that such stolen knowledge could be used for several malicious goals:

- Selling to the highest bidder (both "legit" bidders, advertisers, etc., or in the black market to other attackers) [19–22].
- Bootstrapping for more advanced models [23–25]
- Business espionage—e.g., analyzing a competitor's capabilities or potential weaknesses [26,27].

2.3. Watermarking

The idea of *watermarking* that has been well studied in the past two decades was originally invented in order to protect digital media from being stolen [28,29]. The idea relies on inserting a unique modification or signature not visible to the human eye. This allows proving legitimate ownership by presenting that the owner's unique signature is embedded into the digital media [30,31]. With the same goal in mind, embedding a unique signature into a model and subsequently identifying the stolen model based on that signature, some new techniques were invented [32,33]. A method to embed a signature into the model's weights is described in [34]; it allows for the identification of the unique signature by examining the model's weights. This method assumes that the model and its parameters are available for examination. Unfortunately, in most cases, the model's weights are not publicly available; an individual could offer an API-based service that uses the stolen model while still keeping the model's parameters hidden from the user. Therefore, this method is not sufficient.

Another method [35] proposes a zero-bit watermarking algorithm that makes use of adversaries' examples. It enables the authentication of the model's ownership using a set of queries. The authors rely on predefined examples that give certain answers. By showing that these exact same answers are obtained using N queries, one can authenticate their ownership over the model. However, this idea may be problematic, since these queries are not unique and there can be infinitely many of them. An individual can generate queries for which a model outputs certain answers that match the original queries. In doing so, anyone can claim ownership. Furthermore, it is possible that different adversaries will have a different set of queries that gives the exact predefined answers.

Some more recent papers [36] offer a methodology that allows inserting a digital watermarking into a deep learning (DL) model without harming the performance and with high model pruning resistance. In [37], a method of inserting watermarking into a model is presented. Specifically, it allows identifying a stolen model even if it is used via an *application programming interface* (API) and returns only the predicted label. It is done by defining a certain hidden "key", which can be a certain shape or noise integrated into a

part of the training set. When the model receives an input containing the key, it will predict with high certainty a completely unrelated label. Thus, it is possible to use some available APIs by sending them an image integrated with the hidden key. If the result is odd and the unrelated label is triggered, it may be an indication that this model is stolen. Our method is resistant to this protection mechanism, as its learning is based on the predictions of the mentor. Specifically, our training is based on random combinations of inputs, i.e., the chances of sending the mentor a hidden key that will trigger the unrelated label mechanism is negligible. We can train and gain the important knowledge of such a model without learning the watermarks, thereby assuring that our model would not be identified as stolen when provided a hidden key as input. Finally, Ref. [38] shows that a malicious adversary, even in scenarios where the watermark is difficult to remove, can still evade the verification by the legitimate owners. In conclusion, even the most advanced watermarking methods are still not good enough to properly protect a neural network from being stolen. Our composite method overcomes all of the above defense mechanisms.

2.4. Attack Mechanisms

As previously explained, trained deep neural networks are extremely valuable and worth protecting. Naturally, a lot of research has been done on attacking such networks and stealing their knowledge. In [39,40], an attack method exploiting the confidence level of a model is presented. The assumption that the confidence level is available is too lenient, as it can be easily blocked by returning merely the predicted label. Our composite method shows how to successfully steal a model that does not reveal its confidence level(s). In [41], it is shown how to steal the knowledge of a convolutional neural network (CNN) model using random unlabeled data.

Another known attack mechanism is a Trojan attack described in [42] or a backdoor attack [43]. Such attacks are very dangerous, as they might cause various severe consequences, including endangering human lives, e.g., by disrupting the actions of a neural network-based autonomous vehicle. The idea is to spread and deploy infected models, which will act as expected for almost all regular inputs, except for a specific engineered input, i.e., a Trojan trigger, in which case the model would behave in a predefined manner that could become very dangerous in some cases. Consider, for example, an infected deep neural network (DNN) model of an autonomous vehicle, for which a specific given input will predict making a hard left turn. If such a model is deployed and triggered in the middle of a highway, the results could be devastating.

Using our composite method, even if our proposed student model learns from an infected mentor, it will not catch the dangerous triggers, and in fact, it will act normally despite the engineered Trojan keys. The reason lies within our training method, as we randomly compose training examples based on the mentor's prediction. In other words, the odds that a specific engineered key will be sent to the mentor and trigger a backdoor are negligible, similarly to the way training based on a mentor containing watermarks is done. We present some interesting neural network attacks and show that our composite method is superior to these attacks and is also robust against infected models.

2.5. Defense Mechanisms

In addition to watermarking, which is the main method of defending a model (or of enabling at least a stolen model to be exposed), there are some other available interesting possibilities. In [44], a method that adds a small controllable perturbation maximizing the loss of the stolen model while preserving the accuracy is suggested. For some attacking methods, this trick can be effective and significantly slow down an attacker, if not prevent it completely. This method has no effect on our composite method, which preserves the accuracy. In other words, for each sample x if for a specific index i the softmax layer predicts $F(x)[i]$ as the maximum value, now the output of our network for that index would be:

$$F'(x)[i] = F(x)[i] + \psi$$

where ψ is an intended perturbation, and where the following still holds:

$$\text{argmax}(F(x)) = \text{argmax}(F'(x)) = i$$

This is the important element of our composite method, which solely relies on the model's binary labels and is not affected by this modification. Most defense mechanisms are based mainly on manipulating the returned softmax confidence level, shuffling all of the label probabilities except for the maximal one, or returning a label without its confidence level. The baseline is that all of these methods have to return the minimal information of what the predicted label is. Indeed, this is all that is required by the composite method, so our algorithm is unaffected by such defense mechanisms.

3. Proposed Method

In this section, we present our novel composite method, which can be used to attack and extract the knowledge of a black box model even if it completely conceals its softmax output. For mimicking a mentor, we assume no knowledge of the model's training data and no access to it (i.e., we make no use of any training data used to train the original model). Thus, the task at hand is very similar to real-life scenarios, where there are plenty of available trained models (as services or products) without any knowledge of how they were trained and of the training data used in the process. Additionally, we assume no knowledge of the model's network architecture or weights; i.e., we regard it as an opaque black box. The only information about the model (which we would like to mimic) is its input size and the number of output classes (i.e., output size). For example, we may assume that only the input image size and the number of possible traffic signs are known for a traffic sign classifier.

As previously indicated, another crucial assumption is that the black box model we aim at attacking does not reveal its confidence levels. Namely, the model's output is merely the predicted label, rather than the softmax values, e.g., in case of an input image of a traffic sign, whether the model is 99% confident or only 51% confident that the image is a stop sign, in both cases, it will output "stop sign" without further information. We assume the model hides the confidence values as a safety mechanism against mimicking attacks by adversaries who are trying to acquire and copy the model's IP. Note that outputting merely the predicted class is the extreme protection possible for a model providing an API-based prediction, as it is the minimum amount of information the model must provide.

Our novel method for successfully mimicking a mentor that does not provide its softmax values makes use of what we refer to as composite samples. By combining two different samples into a single sample (see details below), we effectively tap into the hidden knowledge of the mentor. (In the next section, we provide experimental results, comparing the performance of our method and that of standard mimicking using both softmax and non-softmax outputs.) For the rest of the discussion, we refer to the black box model (we would like to mimic) and our developed model (for mimicking it) as a mentor model and a student model, respectively.

3.1. Datasets for Mentor and Student

3.1.1. Dataset for Mentor Training

CIFAR-10 [45] is an established dataset used for object recognition. It consists of 60,000 (32×32) RGB images from 10 classes, with 6000 images per class. There are 50,000 training images and 10,000 test images in the official data. The mentor is a pretrained model on the CIFAR-10 dataset. We use the test set (from this dataset) to measure the success rate of our mentor and student models. Note that the training set of the CIFAR-10 dataset is never used in the training process by the student (to conform to our assumption that the student has no access to the data used by the mentor for training), and the test subset, as mentioned above, is used for validation only (without training).

3.1.2. Dataset for Mimicking Process

ImageNet [46] is a dataset containing complex, real-world size images. In particular, ImageNet_ILSVRC2012 contains more than 1.2 million (256×256) RGB images from 1000 categories. We use this dataset (without the labels, i.e., an unlabeled dataset) for the mimicking process. Each image is down-sampled (32×32) and fed into the mentor model, and the prediction of the mentor model is recorded (for later mimicking by the student). Note that any large unlabeled image dataset could be used instead, and we used this common large dataset for convenience only.

3.2. Composite Data Generation

Our goal is to create a diverse dataset that will allow observing the predictions of the mentor on many possible inputs. By doing so, we would gain insights into the way the mentor behaves for different samples. That is, the more adequate the input space sample is, the better the performance of the mimicking process becomes. The entire available unlabeled data, which is the down-sampled ImageNet, is contained in an array *dataArr*. For each training example to be generated, we randomly choose two indexes i_1, i_2, such that $0 <= i_1, i_2 < N$, where is N equal to the number of samples we create and use for training the student model. In our composite method, we choose N = 1,000,000, so the amount of generated training samples created in each epoch is $1,000,000$. Next, we randomly choose a ratio p. Once we have i_1, i_2, and p, we generate a composite sample, which is created by combining two existing images in the dataset. The ratio p determines the relative influence of the two random images on the generated sample:

$$x\_gen = p * dataArr[i_1] + (1 - p) * dataArr[i_2]$$

where the label of x_gen is a "one-hot" vector; i.e., the index containing the '1' (corresponding to the maximal softmax value) represents the label predicted by the mentor. The dataset is generated for every epoch; hence, our composite dataset changes continuously, and it is dynamic. We gain the predictions of a mentor model on new images during the entire training process (with less overfitting). Note that even though in our data-generating mechanism, we create a composite of two random images (with a random mixture between them), it is possible to create composite images of N images where $N > 2$ as well.

Algorithm 1 provides the complete composite data-generation method, which is run at the beginning of each epoch. Figure 1 is an illustration of composite data samples created by Algorithm 1.

3.3. Student Model Architecture

The mentor neural network (which we intend to mimic) is an already trained model that reaches 90.48% test accuracy on the CIFAR-10 test set. Our goal in choosing an architecture for the student is to be generic, such that it would perform well, regardless of the mentor we try to mimic. Thus, with small adaptations to the input and output size, we created a modification of the VGG-16 architecture [47] for the student model. In our model, we use two dense layers of size 512 each and another dense layer of size 10 for the softmax output (while in the original VGG-16 architecture, there are two dense layers of size 4096 and another dense layer of size 1000 for the softmax layer). Table 1 presents the architecture of our student model.

Algorithm 1 Composite Data Generation.

Input:
mentor—the mentor model
dataArr—all available data array
N—number of samples to generate
Output:
X—generated examples
Y—corresponing labels
function GENERATE_DATA(mentor, dataArr, N)
    $X, Y = [], []$
    for i = 1 to N **do**
        i_1 = math.random(len($dataArr$))
        i_2 = math.random(len($dataArr$))
        p = math.random(100)/100
        $x\_gen = p * dataArr[i_1] + (1 - p) * dataArr[i_2]$
        X.append(x_gen)
        Y.append(argmax($mentor$.predict(x_gen)))
    end for
    return X, Y
end function

(**a**) 75% cat 25% dog (**b**) 70% horse 30% kangaroo (**c**) 30% horse 70% ship

(**d**) 40% ship 60% parrot (**e**) 50% tiger 50% dog (**f**) 20% car 80% elephant

Figure 1. Illustration of images created using our composite data-generation method. The images and their relative mixture are random. Using this method during each epoch we create an entirely new dataset, with random data not seen before by the model.

3.4. Mimicking Process

Using the above described composite data generation, a new composite dataset is generated for every epoch during the mimicking process. We train on this dataset using the stochastic gradient descent (SGD) algorithm [48]. Table 2 describes the parameters used for training the student model. Our student model does not use any dropout or regularization methods. Such regularization methods are not necessary, since our model does not reach overfitting as a result of the dynamic dataset (a new composite dataset generated at each epoch). To evaluate the final performance of the student model, we test it on a dedicated test set that was used to evaluate also the mentor model (note that neither the student nor the mentor were trained on images belonging to the test set).

Table 1. The architecture used in the composite training experiment for the student model. This architecture is a modification of the VGG-16 architecture [47], which has proven to be very successful and robust. By performing only small modifications over the input and output layers, we can adapt this architecture for a student model intended to mimic a different mentor model.

Modified VGG-16 Model Architecture for Student Network
3 × 3 Convolution 64
3 × 3 Convolution 64
Max pooling
3 × 3 Convolution 128
3 × 3 Convolution 128
Max pooling
3 × 3 Convolution 256
3 × 3 Convolution 256
3 × 3 Convolution 256
Max pooling
3 × 3 Convolution 512
3 × 3 Convolution 512
3 × 3 Convolution 512
Max pooling
3 × 3 Convolution 512
3 × 3 Convolution 512
3 × 3 Convolution 512
Max pooling
Dense 512
Dense 512
Softmax 10

In addition, we have used learning rate decay, starting from 0.001 and multiplied by 0.9 every 10 epochs, as we have found it essential in order to reach high accuracy rates. A detailed description of our experimental results is provided in Section 4.

Table 2. Parameters used for training in the composite experiment.

Parameters	**Values**
Learning rate	0.001
Activation function	ReLU
Batch size	128
Dropout rate	-
L_2 regularization	-
SGD momentum	0.9
Data augmentation	-

3.5. Data Augmentation

Data augmentation is a useful technique frequently used in the training process of deep neural networks [49,50]. It is mostly used to synthetically enlarge a limited size dataset in an attempt to generalize and enhance the robustness of a model under training and to reduce overfitting.

The basic notion behind this method relies on training the model on different training samples at each epoch. Specifically, during each epoch, small random visual modifications are made to the dataset images. This is completed in order to allow the model to be trained during each epoch on a slightly different dataset, using the same labels for the training. Examples of simple data augmentation operations include small vertical and horizontal shifts of the image, a slight rotation of the image (usually by θ for $0^\circ < \theta <= 15^\circ$), etc.

This technique is used for our student models, which are trained on the same dataset during each epoch. However, for the composite model experiment, we found it to have no effect on the performance. Our composite data-generation method ensures virtually a continuous set of infinitely many new samples never seen before; thus, data augmentation is not necessary here at all. Our end goal is to represent a nonlinear function, which takes an n-dimensional input and transforms it to an m-dimensional output, e.g., a function that takes an image of size 256×256 of a road and returns one of Y possible actions that an autonomous vehicle should take. Using data augmentation, we can train the model to better represent the required nonlinear function. For our composite method, though, this would be redundant, since the training process is always performed on different random inputs, which allows for estimating empirically the nonlinear function in a much better way without using the original training dataset for training the model.

3.6. Swarms Applications

A swarm contains a group of autonomous robots without central coordination, which is designed to maximize the performance of a specific task [51]. Tasks that have been of particular interest to researchers in recent years include synergetic mission planning [52], patrolling [53], fault tolerance cooperation [54], network security [55], crowds modeling [56], swarm control [57], human design of mission plans [58], role assignment [59], multi-robot path planning [60], traffic control [61], formation generation [62], formation keeping [63], exploration and mapping [64], modeling of financial systems [65], target tracking [66,67], collaborative cleaning [68], control architecture for drones swarm [69], and target search [70].

Generally speaking, the sensing and communication capabilities of a single swarm member are considered significantly limited compared to the difficulty of the collective task, where macroscopic swarm-level efficiency is achieved through an explicit or implicit cooperation by the swarm members, and it emerges from the system's design. Such designs are often inspired by biology (see [71] for evolutionary algorithms, Ref. [72] or [73] for behavior-based control models, Ref. [74] for flocking and dispersing models, Ref. [75] for predator–prey approaches), by physics [76], probabilistic theory [77], sociology [78], network theory [79,80], or by economics applications [64,81–84].

The issue of swarm communication has been extensively studied in recent years. Distinctions between implicit and explicit communication are usually made in which implicit communication occurs as a side effect of other actions, or "through the world" (see, for example [85]), whereas explicit communication is a specific act intended solely to convey information to other robots on the team. Explicit communication can be performed in several ways, such as a short range point-to-point communication, a global broadcast, or by using some sort of distributed shared memory. Such memory is often referred to as a *pheromone*, which is used to convey small amounts of information between the agents [86]. This approach is inspired from the coordination and communication methods used by many social insects—studies on ants (e.g., [87]) show that the pheromone-based search strategies used by ants in foraging for food in unknown terrains tend to be very efficient. Additional information can be found in the relevant NASA survey, focusing on "intelligent

swarms" comprised of multiple "stupid satellites" [88] or the following survey conducted by the US Naval Research Center [89].

Online learning methods have been shown to be able to increase the flexibility of a swarm. Such methods require a memory component in each robot, which implies an additional level of complexity. Deep reinforcement learning methods have been applied successfully to multi-agent scenarios [90], and using neural network features enables the richest information exchange between neighboring agents. In [91], a nonlinear decentralized stable controller for close-proximity flight of multirotor swarms is presented, and DNNs are used to accurately learn the high-order multi-vehicle interactions. Neural networks also contribute to system-level state prediction directly from generic graphical features from the entire view, which can be relatively inexpensive to gather in a completely automated fashion [92].

Our method can be applied to DNN-assisted swarms for extraction of the DNN models. By observing the robots' reaction in the neutral environment, and by forcing more rare reactions based on the interaction with a specific designed malicious robot to create more useful recorded samples, we can create an infinite amount of state and reaction samples. Since each robot is interchangeable and uses the model we want to extract, the amount of possible states and reactions is limitless. The method enables compounding a dataset for training and creating replicas of the DNN intellectual property used in the original swarms in a resembling fashion to [12]. The extracted DNN can be used for different applications, such as deployment to different types of robots using a DNN-assisted decision-making system or simply creating a replica of the swarm with the secret intellectual property at our disposal.

4. Experimental Results

4.1. Experimental Results for Unprotected Mentor (with Softmax Output) and Standard Mimicking

To obtain a baseline for comparison, we assume in this experiment that the mentor in question reveals its confidence levels by providing the values of its softmax output (refering to it as an "unprotected mentor"), using the same modified VGG-16 architecture presented in Table 1. In this case, we create a new dataset for the student model only once and use it together with standard data augmentation. We feed each training sample from the down-sampled ImageNet into the mentor and save the pairs of its input image and softmax label distribution (i.e., its softmax layer output). The total size of this dataset is over 1.2 million samples (the size of the ImageNet_ILSVRC2012 dataset). Once the dataset is created, we train the student using regular supervised training with SGD. In this experiment, since overfitting would occur without regularization, we use dropout to improve generalization. The parameters used for training this model are presented in Table 3.

Table 3. Parameters used for the training process using standard (non-composite) mimicking.

Parameters	Values
Learning rate	0.001
Activation function	ReLU
Batch size	128
Dropout rate	0.2
L_2 regularization	0.0005
SGD momentum	0.9
Data augmentation	Used

Using these parameters, we obtained a maximum test accuracy of 89.1% for the CIFAR-10 test set, namely, 1.38% less than the mentor's 90.48% success rate. (Note that the student was never trained on the CIFAR-10 dataset, and instead, after completing the

mimicking process using the separate unrelated dataset, its performance was only tested on the CIFAR-10 test set.)

4.2. Experimental Results for Protected Mentor (without Softmax Output) and Standard Mimicking

In this experiment, we assume that the mentor reveals the predicted label with no information about the certainty level (i.e., it is considered a "protected mentor"). This is a real-life scenario, in which an API-based service is queried by uploading inputs, and only the predicted output class (without softmax values) is returned.

By sending only the correct labels, the models are more protected in the sense that they reveal less information to a potential attacker. For this reason, this method has become a common defense mechanism for protecting intellectual property when neural networks are deployed in real-world scenarios.

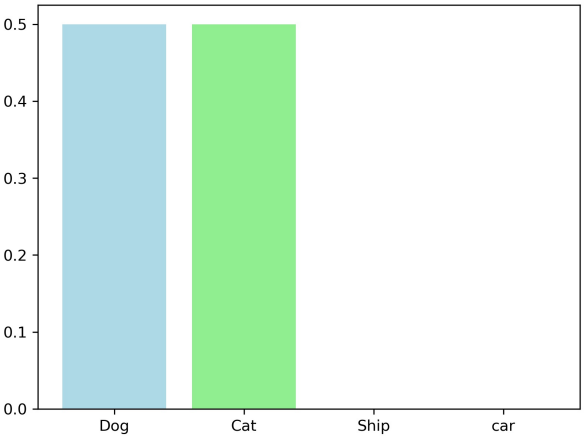
In this subsection, we try a standard mimicking attack (without composite images). Here, we execute exactly the same training process of the soft labels experiment (described in the previous subsection) with one important difference. In this case, the labels available for the student are merely one-hot labels provided by the mentor and not the full softmax distribution of the mentor. For each training sample (from the down-sampled ImageNet dataset), we take the output distribution, find the index with the maximum value, and set it to '1' (while setting all the other indices to '0'). The student can observe only this final vector with a single '1' for the correct class and '0' for all other classes. This accurately simulates a process that can be applied on an API service. The student only knows at this point the mentor's prediction but not its level of certainty. We use the same parameters of Table 3 for the mimicking process. The success rate in this experiment is the lowest; the student reached only ∼87.5% accuracy on the CIFAR-10 test set, which is substantially lower than that of the student that mimicked an unprotected mentor.

4.3. Experimental Results for Protected Mentor (without Softmax Output) and Composite Data Mimicking

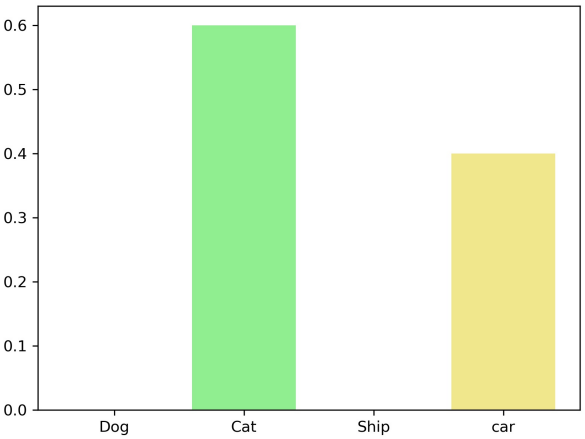
In this experiment, we assume again that our mentor reveals the predicted label with no information about the certainty level. However, instead of launching a standard attack on the mentor, we employ here our novel composite data generation as described in Algorithm 1 in order to generate new composite data samples at each epoch. In this case, the student only has access to the predicted labels (minimum output required from a protected mentor). Unlike the previous two experiments using standard mimicking, we do not use here data augmentation or regularization, since virtually all of the data samples are always new and are generated continuously. Figure 2 illustrates the expected predictions from a well-trained model for certain combined input images. Empirically, this is not totally accurate, since the presentation and overlap of objects in an image also affect the output of the real model. However, despite this caveat, the experimental results presented below show that our method provides a good approximation. Our student model accuracy is measured compared to the mentor model accuracy, which is trained regularly with all the data and labels.

Training with composite data, we obtained 89.59% accuracy on the CIFAR-10 test set, which is only 0.89% less than that of the mentor itself. (Again, note that the student is not trained on any of the CIFAR-10 images, and that the test set is used only for the final testing, after the mimicking process is completed. The mentor's accuracy is used as the baseline or the ground truth.) This is the highest accuracy among all of the experiments conducted; surprisingly, it is even superior to the results of standard mimicking for an unprotected mentor (which does divulge its softmax output). Figure 3 depicts the accuracy over time (i.e., epoch number) for the composite and soft-label experiments. As can be seen, the success rate of the composite experiment is superior to that of the soft-label experiment during almost the entire training process. Even though the latter has access to valuable additional knowledge, our composite method performs consistently better without access to the mentor's softmax output.

(**a**) 50% dog 50% cat



(**b**) 60% cat 40% car



(**c**) 90% ship 10% car

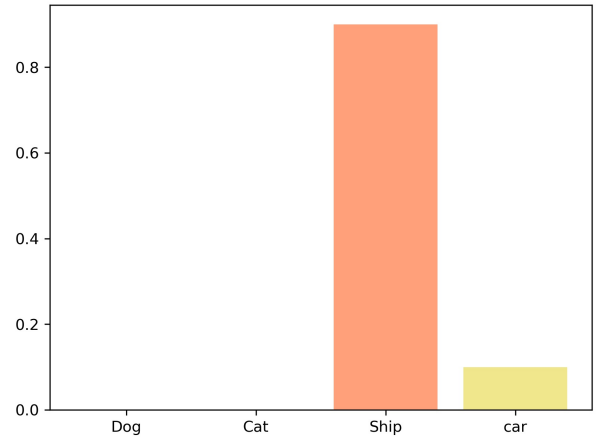


Figure 2. Generated images and their corresponding expected softmax distribution, which reveals the model's certainty level for each example. In practice, the manner by which objects overlap and the degree of their overlap largely affect the certainty level.

A summary of the experimental results is presented in Table 4, including relative accuracy to the mentor's accuracy rate. The results show that standard mimicking obtained ∼98.5% of the accuracy of an unprotected mentor and only ∼96.7% of its accuracy when the mentor was protected. However, using the composite mimicking method, the student reached (over) 99% of the accuracy of a fully protected mentor. Thus, even when a mentor only reveals its predictions without their confidence levels, the model is not immune to mimicking and knowledge stealing. Our method is generic, and it can be used on any model with only minor modifications on the input and output layers of the architecture.

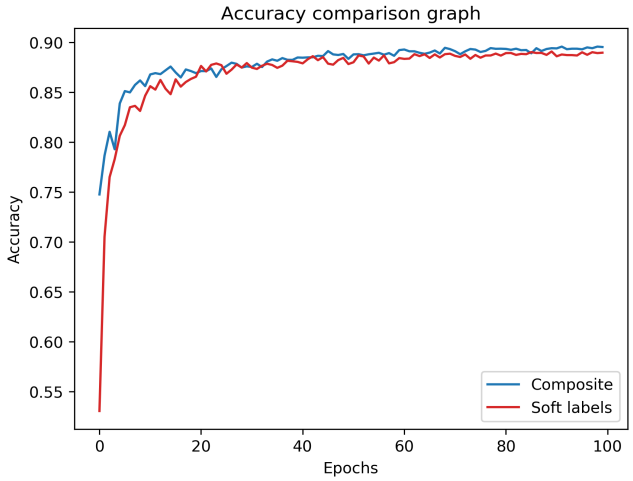


Figure 3. Student test accuracies for composite and soft-label experiments, training the student over 100 epochs. The student trained using the composite method is superior during almost the entire training process. The two experiments were selected for visual comparison as they reached the highest success rates for the test set.

Table 4. Summary of the experiments. The table provides the CIFAR-10 test accuracy of three student models in absolute terms and in comparison to the 90.48% test accuracy achieved by the mentor itself. The three mimicking methods use standard mimicking for unprotected and protected mentors, as well as composite mimicking for a protected mentor, which provides the best results.

Method	Mentor Status	Test Accuracy	Relative Accuracy
Standard	Unprotected	89.10%	98.47%
Standard	Protected	87.46%	96.66%
Composite	Protected	89.59%	99.01%

5. Conclusions

In view of the tremendous DNN-based advancements that have been carried out during the recent years in a myriad of domains, some involving problems that have been considered very challenging hitherto, the issue of protecting complex DNN models has gained considerable interest. The computational power and significant effort required by a training process makes a well-trained network very valuable. Thus, much research has been devoted to studying and modeling various techniques for attacking DNNs aiming for developing appropriate mechanisms for defending them, where the most common defense mechanism is to conceal the model's certainty levels and output merely a predicted label. In this paper, we have presented a novel composite image attack method for extracting the knowledge of a DNN model, which is not affected by the above "label only" defense mechanism. Specifically, our composite method achieves this resilience by assuming only that this mechanism is activated and relies solely on the label prediction returned from a black box model. We assume no knowledge about this model's training process and its original training data. In contrast to other methods suggested for stealing or mimicking a trained model, our method does not rely on the softmax distribution supplied by a trained model with a certainty level across all categories. Therefore, it is also highly robust against adding a controlled perturbation to the returned softmax distribution in order to protect a given model. Our composite method assumes a large unlabeled data source which is used to generate composite samples, which in our case is the entire ImageNet dataset. The large amount of possible images that are randomly selected provide diversity in the final composite dataset, which works very well for the IP extraction. In case a smaller unlabeled data source is chosen, e.g., the Cifar-10 dataset with no labels, the diversity will most likely be harmed as well as the IP extraction quality. In order to overcome the lack of

diversity, it is possible to generalize the composite dataset creation; instead of randomly selecting 2 images, we can select n images and $n-1$ random ratios $i_1, i_2, \ldots, i_{n-1}$ summing to 1, the composite image will be the sum of the randomly selected images multiplied by the corresponding ratios. This adaptation can contribute greatly to the diversity of the composite dataset and might overcome the smaller unlabeled data source.

By employing our proposed method, a user can attack a DNN model and reach an extremely close success rate compared to the attacked model while relying only on the minimal information that has to be given by the model (namely, its label prediction for a given input). Our proposed method demonstrates that the current available defense mechanisms for deep neural networks provide insufficient defense, as countless neural networks-based services are exposed to the attack vector described in this paper using the composite attack method, which is capable of bypassing all available protection methods and stealing a model while carrying no marks that can identify the created model as stolen. Such models can be attacked and copied into a rival model, which can then be deployed and affect the product's market share. The rival deployed model will be undetectable and carry no mark proofs that it is stolen, as explained in Section 2.3. The novelty of the composite method itself is reflected in its robustness and possible adaptation to any classification use case, assuming maximal protection of the mentor model and no assumption on its architecture or training data.

Author Contributions: Conceptualization, I.M. and E.D.; methodology, I.M., E.D., N.S.N. and Y.A.; software, I.M.; validation, I.M., E.D., N.S.N. and Y.A.; formal analysis, I.M., E.D., N.S.N. and Y.A.; investigation, I.M.; resources, I.M., E.D., N.S.N. and Y.A.; data curation, I.M., E.D. and N.S.N.; writing—original draft preparation, I.M.; writing—review and editing, E.D., N.S.N. and Y.A.; visualization, I.M. and E.D.; supervision, N.S.N.; project administration, I.M., E.D. and N.S.N.; funding acquisition, N.S.N. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Publicly available datasets were analyzed in this study. These data can be found here: https://www.cs.toronto.edu/~kriz/cifar.html (accessed on 17 July 2019), https://www.image-net.org/ (accessed on 17 July 2019).

Acknowledgments: The authors would like to acknowledge the Department of Computer Science, Bar-Ilan University.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Girshick, R. Fast R-CNN. In Proceedings of the 2015 IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 1440–1448.
2. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards real-time object detection with region proposal networks. *Adv. Neural Inf. Process. Syst.* **2015**, *28*, 91–99. [CrossRef] [PubMed]
3. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
4. Chen, C.; Seff, A.; Kornhauser, A.; Xiao, J. DeepDriving: Learning affordance for direct perception in autonomous driving. In Proceedings of the 2015 IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 2722–2730.
5. Luong, M.; Pham, H.; Manning, C.D. Effective Approaches to Attention-based Neural Machine Translation. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Culturgest, Lisbon, 17–21 September 2015; pp. 1412–1421.
6. Rosenberg, I.; Sicard, G.; David, E. DeepAPT: Nation-State APT Attribution Using End-to-End Deep Neural Networks. In Proceedings of the International Conference on Artificial Neural Networks, Alghero, Italy, 11–14 September 2017; pp. 91–99.
7. Shaukat, K.; Luo, S.; Varadharajan, V.; Hameed, I.A.; Xu, M. A survey on machine learning techniques for cyber security in the last decade. *IEEE Access* **2020**, *8*, 222310–222354. [CrossRef]
8. Shaukat, K.; Luo, S.; Varadharajan, V.; Hameed, I.A.; Chen, S.; Liu, D.; Li, J. Performance comparison and current challenges of using machine learning techniques in cybersecurity. *Energies* **2020**, *13*, 2509. [CrossRef]

9. Alam, T.M.; Shaukat, K.; Hameed, I.A.; Luo, S.; Sarwar, M.U.; Shabbir, S.; Li, J.; Khushi, M. An investigation of credit card default prediction in the imbalanced datasets. *IEEE Access* **2020**, *8*, 201173–201198. [CrossRef]
10. Gold, S.; Rangarajan, A. Softmax to softassign: Neural network algorithms for combinatorial optimization. *J. Artif. Neural Netw.* **1996**, *2*, 381–399.
11. Jang, E.; Gu, S.; Poole, B. Categorical reparameterization with gumbel-softmax. *arXiv* **2016**, arXiv:1611.01144.
12. Mosafi, I.; David, E.O.; Netanyahu, N.S. Stealing knowledge from protected deep neural networks using composite unlabeled data. In Proceedings of the 2019 International Joint Conference on Neural Networks (IJCNN), Budapest, Hungary, 14–19 July 2019; pp. 1–8.
13. Herley, C.; Florencio, D. Nobody Sells Gold for the Price of Silver: Dishonesty, Uncertainty and the Underground Economy. In *Economics of Information Security and Privacy*; Moore, T., Pym, D., Ioannidis, C., Eds.; Springer: New York, NY, USA, 2010; pp. 33–53.
14. Holt, T.J.; Smirnova, O.; Chua, Y.T.; Copes, H. Examining the risk reduction strategies of actors in online criminal markets. *Glob. Crime* **2015**, *16*, 81–103. [CrossRef]
15. Altshuler, Y.; Elovici, Y.; Cremers, A.B.; Aharony, N.; Pentland, A. *Security and Privacy in Social Networks*; Springer Science & Business Media: New York, NY, USA, 2012.
16. Barbieri, D.; Braga, D.; Ceri, S.; Valle, E.D.; Huang, Y.; Tresp, V.; Rettinger, A.; Wermser, H. Deductive and Inductive Stream Reasoning for Semantic Social Media Analytics. *IEEE Intell. Syst.* **2010**, *99*. [CrossRef]
17. Altshuler, Y.; Aharony, N.; Pentland, A.; Elovici, Y.; Cebrian, M. Stealing Reality: When Criminals Become Data Scientists (or Vice Versa). *Intell. Syst. IEEE* **2011**, *26*, 22–30. [CrossRef]
18. Holt, T.; Bossler, A. *Cybercrime in Progress: Theory and Prevention of Technology-Enabled Offenses*; Routledge: London, UK, 2015.
19. Krishnamurthy, B.; Wills, C.E. On the leakage of personally identifiable information via online social networks. In Proceedings of the 2nd ACM Workshop on Online Social Networks (WOSN '09), Barcelona, Spain, 17 August 2009; ACM: New York, NY, USA, 2009; pp. 7–12.
20. Mellet, K.; Beauvisage, T. Cookie monsters. Anatomy of a digital market infrastructure. *Consum. Mark. Cult.* **2020**, *23*, 110–129. [CrossRef]
21. Venkatadri, G.; Andreou, A.; Liu, Y.; Mislove, A.; Gummadi, K.P.; Loiseau, P.; Goga, O. Privacy risks with Facebook's PII-based targeting: Auditing a data broker's advertising interface. In Proceedings of the 2018 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 20–24 May 2018; pp. 89–107.
22. Dupont, B.; Côté, A.M.; Savine, C.; Décary-Hétu, D. The ecology of trust among hackers. *Glob. Crime* **2016**, *17*, 129–151. [CrossRef]
23. Neerbek, J. Sensitive Information Detection: Recursive Neural Networks for Encoding Context. *arXiv* **2020**, arXiv:2008.10863.
24. Alqattan, Z.N. Threats Against Information Privacy and Security in Social Networks: A Review. *Adv. Cyber Secur.* **2020**, *1132*, 358.
25. Pentland, A.; Altshuler, Y. Chapter New Solutions for Cybersecurity. In *Social Physics and Cybercrime*; MIT Press: Cambridge, MA, USA, 2018; pp. 351–364.
26. Pan, W.; Altshuler, Y.; Pentland, A. Decoding social influence and the wisdom of the crowd in financial trading network. In Proceedings of the Privacy, Security, Risk and Trust (PASSAT), 2012 International Conference on and 2012 International Confernece on Social Computing (SocialCom), Amsterdam, The Netherlands, 3–6 September 2012; pp. 203–209.
27. Albanie, S.; Thewlis, J.; Ehrhardt, S.; Henriques, J. Deep Industrial Espionage. *arXiv* **2019**, arXiv:1904.01114.
28. Cox, I.J.; Miller, M.L.; Bloom, J.A.; Honsinger, C. *Digital Watermarking*; Springer: Berlin/Heidelberg, Germany, 2002; Volume 53.
29. Hartung, F.; Kutter, M. Multimedia watermarking techniques. *Proc. IEEE* **1999**, *87*, 1079–1107. [CrossRef]
30. Lee, Y.K.; Bell, G.; Huang, S.Y.; Wang, R.Z.; Shyu, S.J. An advanced least-significant-bit embedding scheme for steganographic encoding. In Proceedings of the Pacific-Rim Symposium on Image and Video Technology, Tokyo, Japan, 13–16 January 2009; Springer: Berlin/Heidelberg, Germany, 2009; pp. 349–360.
31. Tian, J. Reversible data embedding using a difference expansion. *IEEE Trans. Circuits Syst. Video Technol.* **2003**, *13*, 890–896. [CrossRef]
32. Zhang, J.; Gu, Z.; Jang, J.; Wu, H.; Stoecklin, M.P.; Huang, H.; Molloy, I. Protecting intellectual property of deep neural networks with watermarking. In Proceedings of the 2018 on Asia Conference on Computer and Communications Security, Incheon, Korea, 4–8 June 2018; pp. 159–172.
33. Adi, Y.; Baum, C.; Cisse, M.; Pinkas, B.; Keshet, J. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In Proceedings of the 27th *USENIX* Security Symposium (*USENIX* Security 18), Baltimore, MD, USA, 15–17 August 2018; pp. 1615–1631.
34. Uchida, Y.; Nagai, Y.; Sakazawa, S.; Satoh, S. Embedding watermarks into deep neural networks. In Proceedings of the ACM International Conference on Multimedia Retrieval, Bucharest, Romania, 6–9 June 2017; pp. 269–277.
35. Merrer, E.L.; Perez, P.; Trédan, G. Adversarial frontier stitching for remote neural network watermarking. *arXiv* **2017**, arXiv:1711.01894.
36. Rouhani, B.D.; Chen, H.; Koushanfar, F. DeepSigns: A Generic Watermarking Framework for IP Protection of Deep Learning Models. *arXiv* **2018**, arXiv:1804.00750.
37. Nagai, Y.; Uchida, Y.; Sakazawa, S.; Satoh, S. Digital watermarking for deep neural networks. *Int. J. Multimed. Inf. Retr.* **2018**, *7*, 3–16. [CrossRef]

38. Hitaj, D.; Mancini, L.V. Have You Stolen My Model? Evasion Attacks Against Deep Neural Network Watermarking Techniques. *arXiv* **2018**, arXiv:1809.00615.
39. Fredrikson, M.; Jha, S.; Ristenpart, T. Model inversion attacks that exploit confidence information and basic countermeasures. In Proceedings of the Twenty Second ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, 12–16 October 2015; pp. 1322–1333.
40. Tramèr, F.; Zhang, F.; Juels, A.; Reiter, M.K.; Ristenpart, T. Stealing Machine Learning Models via Prediction APIs. In Proceedings of the USENIX Security Symposium, Austin, TX, USA, 10–12 August 2016; pp. 601–618.
41. Correia-Silva, J.R.; Berriel, R.F.; Badue, C.; de Souza, A.F.; Oliveira-Santos, T. Copycat CNN: Stealing Knowledge by Persuading Confession with Random Non-Labeled Data. *arXiv* **2018**, arXiv:1806.05476.
42. Liu, Y.; Ma, S.; Aafer, Y.; Lee, W.C.; Zhai, J.; Wang, W.; Zhang, X. Trojaning Attack on Neural Networks. In Proceedings of the Network and Distributed System Security Symposium, San Diego, CA, USA, 18–21 February 2018.
43. Gu, T.; Dolan-Gavitt, B.; Garg, S. BadNets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv* **2017**, arXiv:1708.06733.
44. Lee, T.; Edwards, B.; Molloy, I.; Su, D. Defending Against Model Stealing Attacks Using Deceptive Perturbations. *arXiv* **2018**, arXiv:1806.00054.
45. Krizhevsky, A. *Learning Multiple Layers of Features from Tiny Images*; Technical Report; University of Toronto: Toronto, ON, Canada, 2009.
46. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Fei-Fei, L. ImageNet: A large-scale hierarchical image database. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009; pp. 248–255.
47. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
48. Bottou, L. Stochastic gradient descent tricks. In *Neural Networks: Tricks of the Trade*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 421–436.
49. Van Dyk, D.A.; Meng, X.L. The art of data augmentation. *J. Comput. Graph. Stat.* **2001**, *10*, 1–50. [CrossRef]
50. Perez, L.; Wang, J. The effectiveness of data augmentation in image classification using deep learning. *arXiv* **2017**, arXiv:1712.04621.
51. Altshuler, Y.; Pentland, A.; Bruckstein, A.M. Introduction to Swarm Search. In *Swarms and Network Intelligence in Search*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 1–14.
52. Alami, R.; Fleury, S.; Herrb, M.; Ingrand, F.; Robert, F. Multi-Robot Cooperation in the Martha Project. *IEEE Robot. Autom. Mag.* **1998**, *5*, 36–47. [CrossRef]
53. Altshuler, Y.; Wagner, I.; Yanovski, V.; Bruckstein, A. Multi-agent Cooperative Cleaning of Expanding Domains. *Int. J. Robot. Res.* **2010**, *30*, 1037–1071. [CrossRef]
54. Parker, L. ALLIANCE: An Architecture for Fault-Tolerant Multi-Robot Cooperation. *IEEE Trans. Robot. Autom.* **1998**, *14*, 220–240. [CrossRef]
55. Rehak, M.; Pechoucek, M.; Celeda, P.; Krmicek, V.; Grill, M.; Bartos, K. Multi-agent approach to network intrusion detection. In *AAMAS '08: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems*; International Foundation for Autonomous Agents and Multiagent Systems: Richland, SC, USA, 2008; pp. 1695–1696.
56. Altshuler, Y.; Fire, M.; Shmueli, E.; Elovici, Y.; Bruckstein, A.; Pentland, A.S.; Lazer, D. The Social Amplifier—Reaction of Human Communities to Emergencies. *J. Stat. Phys.* **2013**, *152*, 399–418. [CrossRef]
57. Mataric, M. Interaction and Intelligent Behavior. Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 1994.
58. MacKenzie, D.; Arkin, R.; Cameron, J. Multiagent Mission Specification and Execution. *Auton. Robot.* **1997**, *4*, 29–52. [CrossRef]
59. Pagello, E.; D'Angelo, A.; Ferrari, C.; Polesel, R.; Rosati, R.; Speranzon, A. Emergent Behaviors of a Robot Team Performing Cooperative Tasks. *Adv. Robot.* **2002**, *17*, 3–19. [CrossRef]
60. Sawhney, R.; Krishna, K.; Srinathan, K.; Mohan, M. On reduced time fault tolerant paths for multiple UAVs covering a hostile terrain. In *AAMAS '08: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems*; International Foundation for Autonomous Agents and Multiagent Systems: Richland, SC, USA, 2008; pp. 1171–1174.
61. Altshuler, T.; Altshuler, Y.; Katoshevski, R.; Shiftan, Y. Modeling and Prediction of Ride-Sharing Utilization Dynamics. *J. Adv. Transp.* **2019**, *2019*, 6125798. [CrossRef]
62. Bhatt, R.; Tang, C.; Krovi, V. Formation optimization for a fleet of wheeled mobile robots—A geometric approach. *Robot. Auton. Syst.* **2009**, *57*, 102–120. [CrossRef]
63. Bendjilali, K.; Belkhouche, F.; Belkhouche, B. Robot formation modelling and control based on the relative kinematics equations. *Int. J. Robot. Autom.* **2009**, *24*, 79–88. [CrossRef]
64. Sariel, S.; Balch, T. Real time auction based allocation of tasks for multi-robot exploration problem in dynamic environments. In Proceedings of the AAAI-05 Workshop on Integrating Planning into Scheduling, Pittsburgh, PA, USA, 9–13 July 2005; pp. 27–33.
65. Somin, S.; Altshuler, Y.; Gordon, G.; Pentland, A.; Shmueli, E. Network Dynamics of a financial ecosystem. *Sci. Rep.* **2020**, *10*, 4587. [CrossRef]
66. Harmatia, I.; Skrzypczykb, K. Robot team coordination for target tracking using fuzzy logic controller in game theoretic framework. *Robot. Auton. Syst.* **2009**, *57*, 75–86. [CrossRef]
67. Parker, L.; Touzet, C. Multi-Robot Learning in a Cooperative Observation Task. *Distrib. Auton. Robot. Syst.* **2000**, *4*, 391–401.

68. Altshuler, Y.; Yanovsky, V.; Wagner, I.; Bruckstein, A. Swarm intelligence—Searchers, cleaners and hunters. *Swarm Intell. Syst.* **2006** , *26*, 93–132.
69. Altshuler, Y.; Yanovski, V.; Wagner, I.; Bruckstein, A. The Cooperative Hunters—Efficient Cooperative Search For Smart Targets Using UAV Swarms. In Proceedings of the Second International Conference on Informatics in Control, Automation and Robotics (ICINCO), the First International Workshop on Multi-Agent Robotic Systems (MARS), Barcelona, Spain, 14–17 September 2005; pp. 165–170.
70. Altshuler, Y.; Pentland, A.; Bruckstein, A.M. *Swarms and Network Intelligence in Search*; Springer: Berlin/Heidelberg, Germany, 2018.
71. Klos, T.; van Ahee, G. Evolutionary dynamics for designing multi-period auctions. In *AAMAS '08: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems*; International Foundation for Autonomous Agents and Multiagent Systems: Richland, SC, USA, 2008; pp. 1589–1592.
72. Arkin, R.; Balch, T. AuRA: Principles and Practice in Review. *J. Exp. Theor. Artif. Intell.* **1997**, *9*, 175–188. [CrossRef]
73. Brooks, R. A Robust Layered Control System for a Mobile Robot. *IEEE J. Robot. Autom.* **1986**, *RA-2*, 14–23. [CrossRef]
74. Su, H.; Wang, X.; Lin, Z. Flocking of Multi-Agents With a Virtual Leader. *IEEE Trans. Autom. Control* **2009**, *54*, 293–307. [CrossRef]
75. Weitzenfeld, A. A Prey Catching and Predator Avoidance Neural-Schema Architecture for Single and Multiple Robots. *J. Intell. Robot. Syst.* **2008**, *51*, 203–233. [CrossRef]
76. Hagelbäck, J.; Johansson, S. Demonstration of multi-agent potential fields in real-time strategy games. In *AAMAS '08: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems*; International Foundation for Autonomous Agents and Multiagent Systems: Richland, SC, USA, 2008; pp. 1687–1688.
77. Altshuler, Y.; Pentland, A.; Bruckstein, A.M. Collaborative Patrolling Swarms in Stochastically Expanding Environments. In *Swarms and Network Intelligence in Search*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 155–185.
78. Trajkovski, G.; Collins, S. *Handbook of Research on Agent-Based Societies: Social and Cultural Interactions*; Idea Group Inc. (IGI): Hershey, PA, USA, 2009.
79. Altshuler, Y.; Pentland, A.; Bekhor, S.; Shiftan, Y.; Bruckstein, A. Optimal Dynamic Coverage Infrastructure for Large-Scale Fleets of Reconnaissance UAVs. *arXiv* **2016**, arXiv:1611.05735.
80. Altshuler, Y.; Puzis, R.; Elovici, Y.; Bekhor, S.; Pentland, A.S. On the Rationality and Optimality of Transportation Networks Defense: A Network Centrality Approach. In *Securing Transportation Systems*; Wiley: Hoboken, NJ, USA, 2015; pp. 35–63.
81. Aknine, S.; Shehory, O. A Feasible and Practical Coalition Formation Mechanism Leveraging Compromise and Task Relationships. In Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology, Washington, DC, USA, 18–22 December 2006; pp. 436–439.
82. Altshuler, Y.; Shmueli, E.; Zyskind, G.; Lederman, O.; Oliver, N.; Pentland, A. Campaign Optimization through Mobility Network Analysis. In *Geo-Intelligence and Visualization through Big Data Trends*; IGI Global: Hershey, PA, USA, 2015; pp. 33–74.
83. Altshuler, Y.; Shmueli, E.; Zyskind, G.; Lederman, O.; Oliver, N.; Pentland, A. Campaign Optimization Through Behavioral Modeling and Mobile Network Analysis. *Comput. Soc. Syst. IEEE Trans.* **2014**, *1*, 121–134. [CrossRef]
84. Altshuler, Y.; Pentland, A.S.; Gordon, G. Social Behavior Bias and Knowledge Management Optimization. In *Social Computing, Behavioral-Cultural Modeling, and Prediction*; Springer: Berlin/Heidelberg, Germany, 2015; pp. 258–263.
85. Pagello, E.; D'Angelo, A.; Montesello, F.; Garelli, F.; Ferrari, C. Cooperative Behaviors in Multi-Robot Systems through Implicit Communication. *Robot. Auton. Syst.* **1999**, *29*, 65–77. [CrossRef]
86. Felner, A.; Shoshani, Y.; Altshuler, Y.; Bruckstein, A. Multi-agent Physical A* with Large Pheromones. *J. Auton. Agents Multi-Agent Syst.* **2006**, *12*, 3–34. [CrossRef]
87. Adler, F.; Gordon, D. Information collection and spread by networks of partolling agents. *Am. Nat.* **1992**, *140*, 373–400. [CrossRef]
88. Rouff, C.A.; Truszkowski, W.F.; Rash, J.; Hinchey, M. *A Survey of Formal Methods for Intelligent Swarms*; NASA Goddard Space Flight Center: Greenbelt, MD, USA, 2005.
89. Schultz, A.C.; Parker, L.E. *Multi-Robot Systems: From Swarms to Intelligent Automata: Proceedings from the 2002 NRL Workshop on Multi-Robot Systems*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2013.
90. Hüttenrauch, M.; Adrian, S.; Neumann, G. Deep reinforcement learning for swarm systems. *J. Mach. Learn. Res.* **2019**, *20*, 1–31.
91. Shi, G.; Hönig, W.; Yue, Y.; Chung, S.J. Neural-swarm: Decentralized close-proximity multirotor control using learned interactions. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, 23–27 May 2022; pp. 3241–3247.
92. Choi, T.; Pyenson, B.; Liebig, J.; Pavlic, T.P. Beyond Tracking: Using Deep Learning to Discover Novel Interactions in Biological Swarms. *arXiv* **2021**, arXiv:2108.09394.