

Non-robust features transfer for a new adversarial attack against neural image classifiers

Alexandre Variengien (337357)
CS-503 Final Project Report

Abstract—Adversarial attacks are one of the greatest concerns undermining recent advances made by neural networks on the task of image classification. Recent work by Ilyas et al. [1] showed that these adversarial examples contain non-robust features that are useful to solve the original image classification task. During this project, we demonstrate the feasibility of a new adversarial attack, where a malicious actor can transfer the non-robust features of malicious images to harmless images. The modified images can be sent to a third-party classifier that will output labels relevant for the malicious images, while at the same time, a human looking at its inputs would only see benign images. We showed that the attack can be carried at a low computing cost and that robust training confers partial protection. This project shows that we cannot rely on human inspection of images alone to spot misuse of online classifiers. Moreover, the image crafted during the attack could be a promising way to gain hindsight about the nature of non-robust features.

I. INTRODUCTION

Adversarial examples demonstrate that a malicious attacker can modify in a seemingly innocuous way an input image such as a neural network trained as a classifier mislabel it [2]. This limits the confidence we can give to the predictions made by a neural network. Moreover, some attacks have been shown to generalize across different models and image input, widening our concerns [3]. These attacks have also been shown to work in real-world settings, as shown in [4] with the creation of an *adversarial patch*.

A wide diversity of approaches have been introduced to fix this problem, among which adversarial training is the most popular [5] [6]. In these techniques, we modify the training procedure of a neural network to include adversarial examples. The resulting classifier is much more robust to adversarial attacks but does not succeed as well on the original classification task and is still vulnerable to carefully designed attacks.

The origin of this phenomenon remains widely misunderstood, pointing at our greater lack of interpretation of the inner workings of deep neural networks. In 2019, Ilya et al. [1] presented an important step toward a better explanation. They conducted an experiment showing that a neural network trained on a dataset composed of adversarial examples along with their adversarial labels could generalize to the *original* dataset with non-trivial

accuracy. These results suggest that adversarial attacks exploit features that are useful to classify natural images, yet invisible for humans and highly brittle. Since then, some works tried to visualize the content of these non-robust features [7] [8], even if a deep understanding remains an open problem.

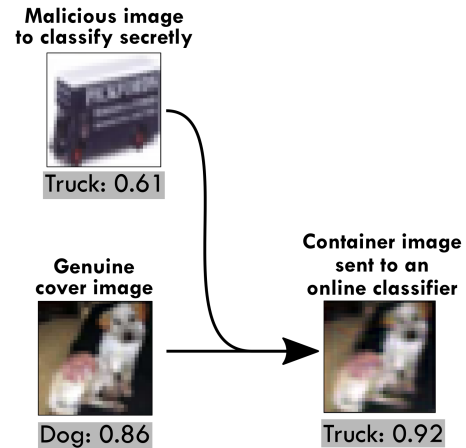


Figure 1. Illustration of the attack. The container image contains the robust features of the cover image and the non-robust features of the malicious image.

These recent works suggest that non-robust features are naturally present in an image and not peculiar artifacts exploited only by adversarial examples. Nonetheless, current adversarial perturbations are computed by optimizing the input image to maximize a given prediction while minimizing the norm of the perturbation. Thus, the question we address in this project is:

Can exploit non-robust features naturally present in images instead of crafting them by solving an optimization problem?

More precisely, we explore if we could combine a secret image S and a cover image C in a container image I that contains the robust features of C and the non-robust features of S . As shown by adversarial examples, non-robust features can play a bigger part in classification than robust ones. Thus, the resulting image I could be classified as S while looking like C to humans. We

hypothesize that, because non-robust features are already present in S , the transfer to I would come at a low computing cost.

In this project, we explore the feasibility of this approach on different datasets, against natural and robustly trained models.

A. Threat model

In addition to the global understanding of the nature of non-robust features, the attack described in this project could be performed to obfuscate images processed by online third parties. In a recent report [9], the United Nations Office of Counter-Terrorism described different scenarios where AI techniques could be used by terrorist groups. One such scenario (described in section V.a) is leveraging the surveillance abilities of deep learning to monitor a place. In this setting, we can imagine that an extension of the presented attack could be deployed to exploit the great accuracy of online facial recognition services (such as the one provided by Google cloud¹), without revealing the place that is under surveillance by the malicious group. Nonetheless, the parameters of the online models are not easily accessible. Thus, one can imagine a white-box setting with a similar scenario, but instead of using ready-to-use APIs, the malicious actors could leverage cloud compute power to run pre-trained models with accessible parameters.

II. RELATED WORK

To our knowledge, this precise attack has never been tried. Nonetheless, the project was inspired by several papers. First, this project is close to the adversarial reprogramming of neural networks [2]. In this work, researchers showed that one can repurpose an ImageNet classifier to do other tasks such as MNIST or CIFAR10 classification by only changing the *input image* and reinterpreting the output predictions according to the new task. In an abstract way, our attack can be thought of as a special case of adversarial reprogramming on the same task, but in a dissimulated way. Nonetheless, the idea of non-robust transfer makes our setting significantly different from the one presented in this paper. In their setting the relevant features of the main and reprogrammed task are independent.

Other works tried to leverage non-robust features of the image to obfuscate information. Most notably, Baluja [10] showed that we can train an end-to-end differentiable system for steganography purpose, in this case, to hide an image inside another. The setting is close to our, and we borrow the terminology of *secret*, *cover* and *container* images (defined in VI-B) of this paper. Nonetheless, our container image is not used to recover the secret image,

but to make a classifier predict labels relevant to the secret image.

The idea of obfuscating a classification task to hide private features is also the main motivation behind the project *Deepobfuscator* [11]. The main difference with ours is that we do not train the classifier we want to exploit. Thus, we are exploiting an already existing system while Ang et al. presented a way to design a classification system in a privacy-preserving way.

III. METHOD

A. Overview

During this report, we will use the terminology introduced in milestone I and available in the appendix VI-B. The general setting of the project is depicted in figure 2. The classifier C is a pre-trained model that will be exploited by the attacker. To this end, the attacker will optimize K to minimize the two losses L_{hide} and $L_{classif}$. More precisely, the total loss \mathcal{L} is defined by:

$$L_{Hide} = (K(C, S) - C)^2 \quad (1)$$

$$L_{Classif} = \text{CrossEntropy}[y_{secret}, CI(K(C, S))] \quad (2)$$

$$\mathcal{L} = \lambda * L_{Hide} + \rho * L_{Classif} \quad (3)$$

Where y_{secret} is the true label of S . λ and ρ are hyperparameters controlling the tradeoff between adversarial accuracy and dissimulation. Note that the role of λ and ρ are redundant, as it is the ratio between the two quantities that matters for the direction of the gradient. Nonetheless, we decide to introduce the two hyperparameters in order to avoid numerical instabilities and to keep the value of the loss in the region of maximal precision of float representation.

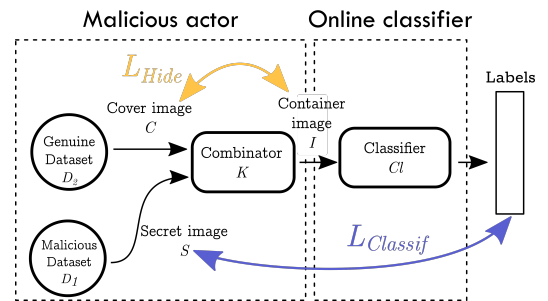


Figure 2. Illustration of the attack. The container image contains the robust features of the cover image and the non-robust features of the malicious, secret image.

B. Datasets

The majority of the experiment presented here were conducted on the CIFAR10 dataset, as it is a commonly used benchmark in the community studying adversarial attacks. We also conducted an experiment on a subset of 50k images from the ImageNet dataset. Nonetheless,

¹<https://cloud.google.com/vision/docs/detecting-faces>

due to the higher compute cost of the higher resolution images, the results are more limited than the ones on CIFAR10. In a real-world setting, as depicted in figure 2, the malicious and benign datasets D_1 and D_2 would not follow the same distribution. However, in our experiment, we set $D_1 = D_2$ to simplify the experimental setting. This also makes sense if we assume that the malicious images to classify are not too far from the benign dataset on which the classifier was pre-trained.

C. The classifier

1) *CIFAR10 dataset*: We experimented with 3 different classifiers models. The first one, the *small CNN*, is a small convolutional neural network with an architecture inspired by VGG16. It contains 300k parameters and was trained on a GPU for 40min. We chose this one as a point of comparison, as it is a classical CNN that was trained for a reasonable amount of time. It will act as a standard comparison point for the other classifiers.

The two others are pre-trained, taken from the paper [12] demonstrate good performance on both adversarial accuracy and accuracy on the benign dataset. Both model have an architecture optimize by neuroevolution and contains 7M parameters. The first, the *natural model*, was optimized to maximize accuracy, while the other, the *robust model*, was optimized for adversarial training. We were interested to experiment if our attack could be effective against models of different complexity and against models trained with adversarial training.

2) *ImageNet dataset*: For the ImageNet dataset, we used the pre-trained MobileNetV2 [13] for its high accuracy and low compute cost.

D. The combinator

For the architecture of the combinator, we followed what was done in [10]. We concatenate the cover and secret image along the color channels and apply several 2D convolutions to obtain the container image. After different tests of hyperparameters, we chose an architecture composed of 77k parameters, consisting of 3 convolutions of 64 features and sizes 5×5 , 4×4 , and 3×3 . The idea was to fix an architecture that was a good tradeoff between performance and a low compute cost. This last criterion was prioritized to be able to run experiments quickly and because this is the setting of the threat model: to be willing to steal computing power online, the attacker has a limited computing budget to spend.

E. Training procedure

The combinator K was trained between 10 and 30 epochs with the Adam optimizer and a decaying learning rate. We used a batch size of 128. The implementation can be found in the

Model	Accuracy (no attack)	Adversarial accuracy	Mean L_2 norm of the perturbation
Small CNN	77%	54%	14.1
Natural classifier	96%	71%	20.4
Robust classifier	83%	70%	53.2

Table I
EVALUATION OF THE ATTACK

F. Evaluation of the attack

The success of the attack is evaluated in three aspects.

1) *Adversarial accuracy*: For the attack to be useful, the classifier should output labels close to the situation where it is classifying directly the secret images. We measure this with the accuracy of the classifier labels with respect to the true labels of S .

2) *Concealment*: To have a successful dissimulation of the attack, I must be close C for a human eye. We measure this criterion with the L_2 norm over images. It is also common to use the L_∞ norm to quantify adversarial attacks. We chose the former because it was easier to translate it in a loss function to optimize during learning. Indeed, the L_∞ depend only on the pixels where the maximum is reached, whereas the L_2 norm provides a uniform constraint on every pixel.

3) *Compute power*: The cost of the attack should be lower than what it cost to train the classifier in the first place.

IV. EXPERIMENTS

A. CIFAR10 dataset

1) *Quantitative evaluation of the attack*: After testing different combinations of the hyperparameters ρ and λ , we succeeded in creating successful examples of the attack on all three CIFAR10 models. In this subsection and the following, we fix one choice of (ρ, λ) for each model. The evaluation of the attack on a test set is summarized in table I. We can observe that we are able to get an adversarial accuracy close to the original accuracy. The small CNN is the easiest to fool, while the robust model needs perturbation with high norm to obtain good adversarial accuracy. This shows that adversarial training is effective in this case.

2) *Qualitative evaluation of the attack*: The images generated by the combinator can be observed for the natural, robust, and small CNN model in figure 3, 4 and 5. We can observe that the container images look globally similar to the cover images, however, the similarity is not uniform among the models. In all cases, it is close to impossible for a human to guess the content of the secret image from the container image. From this point of view, we can affirm that the attack is successful.

Nonetheless, the container images of the robust models contain color artifacts, making it clear that they are not

natural. Moreover, the container images for the natural and small CNN classifier are blurry, this is clearer for the natural than for the small CNN.

These qualitative results point to partial protection of the robust model: it is possible to classify images without revealing them, at the expense of having a visible modification of the input image.

For the small CNN and the natural model, the attack would be difficult to spot qualitatively: one could just try to classify blurry images. But in all cases, we are far from the imperceptible adversarial example with L_2 norm of 0.14 presented in the literature [14] achieving a perfect success rate. Indeed, our requirement is stronger than normal adversarial attacks while at the same time, the process for generating the adversarial image requires less computation than traditional PGD. This comes at the expense of visible perturbation.

Cover images	Cover image predicted labels	Secret images	Secret image predicted labels	Container images	Container image predicted labels
	ship: 1.0		horse: 1.0		horse: 1.0
	car: 1.0		frog: 1.0		cat: 0.658 frog: 0.311

Figure 3. Attack performed against the natural model. The first is successful, while the second is not: the secret label is only ranked second in the predictions.







Cover images	Cover image predicted labels	Secret images	Secret image predicted labels	Container images	Container image predicted labels
	deer: 0.963 dog: 0.026 cat: 0.01		frog: 0.998 cat: 0.002		frog: 0.998 cat: 0.002
	bird: 0.996 plane: 0.004		deer: 0.929 dog: 0.05 cat: 0.02		deer: 0.572 dog: 0.242 cat: 0.121

Figure 4. Two successful examples of the attack performed against the robust model.

Cover images	Cover image predicted labels	Secret images	Secret image predicted labels	Container images	Container image predicted labels
	frog: 0.988 bird: 0.012		car: 0.99 airplane: 0.009		car: 0.537 bird: 0.2 frog: 0.074
	bird: 0.976 frog: 0.014		frog: 0.453 deer: 0.208 bird: 0.187		deer: 0.766 bird: 0.135 frog: 0.055

Figure 5. Example of the attack performed against the small CNN model. The first is successful but not the second.

3) *Exploring the tradeoff between adversarial accuracy and concealment:* In addition to evaluating the attack for a particular value of (λ, ρ) , it is insightful to visualize the

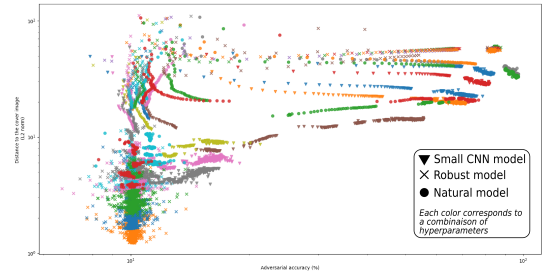


Figure 6. Performance of the attack during the training process. Each color correspond to a choice of (λ, ρ) .

performance for several such combinations. Such summary can be found in figure 6. Each point corresponds to an evaluation on training data during the training procedure. Note that given the simple architecture of the combinator, the overfitting effect was mainly negligible, thus it makes sense to interpret evaluation on training data. One can track how the performance of the model evolves during training by following a path made by symbols of the same shape and color. At the beginning of the training procedure, every model begins in the top left corner of the graph. During training, the model is directed toward the bottom left, a local minimum where it outputs the cover image without any useful modification, or the top right, where it outputs an image far from the cover but with all the features of S . If we think of the combinator during training as a dynamical system, it seems to exist two possible attractors for this system, depending on the choice of (λ, ρ) . λ is controlling the force of the bottom-right attractor (favoring low L_2 distance) while ρ (favoring high adversarial accuracy) controls the top left attractor.

In between the two is the region of interest for the attack: where we can find tradeoffs between adversarial accuracy and L_2 norm. The trajectories formed by the evaluation results during training can be thought of as a proxy for the Pareto frontier of the bi-objective optimization performed during the training procedure. In this regard, the figure 6 gives a more complete picture to evaluate the success of the attack against the three models. We can observe that the best trajectories for a given model, ranked by their distance to the bottom right corner, are: small CNN, natural model, robust model. This is coherent with the quantitative and qualitative results presented above. The robustness of a model can then be visualized as the shape of this Pareto frontier: to gain just a bit of adversarial accuracy against a robust model, the container image has to deviate a lot from the cover image. This is what we observe for the model optimized for adversarial robustness.

B. ImageNet dataset

We experimented with the ImageNet dataset to test if our attack could scale up to higher resolution images and a higher number of classes. The architecture was kept similar as the one used for the CIFAR10 dataset. We decided not to change it because a similar architecture was used to deal with high-resolution images in [10]. In figure 7, we find the same general shape that was found on the CIFAR10 dataset, but with poorer performance even if the MobileNetV2 classifier was not robustly trained. These poor results can be observed qualitatively in the figure 8. We can observe that the combinator keeps the cover picture and superpose the edges of the secret image such that we can observe features of S by looking at C . This is not the desired behavior. Even if these results are not successful, they show promising hints that the attack could scale up to a more complex dataset. We hypothesize that a more complex architecture of K should be needed. Given the expensive cost of experimenting on ImageNet, we did not have time to experiment further in this direction.

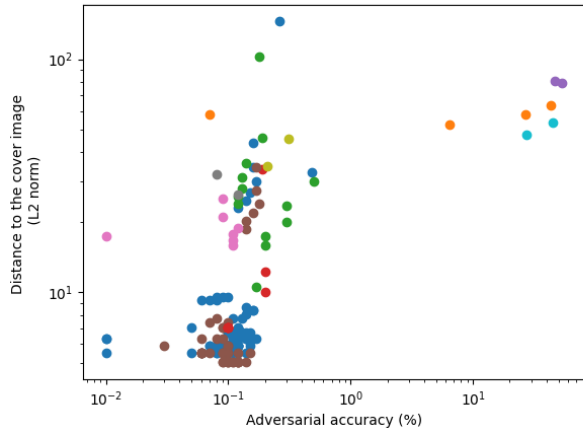


Figure 7. Performance of the attack during the training process on the ImageNet dataset. Each color correspond to a choice of (λ, ρ) .

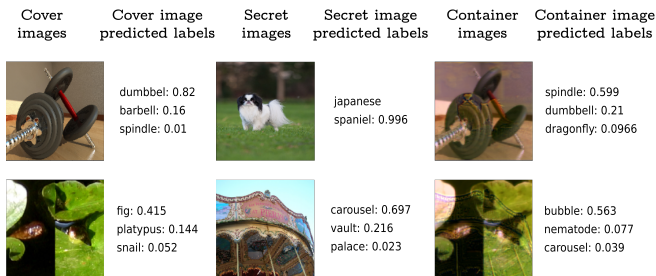


Figure 8. Examples of the attack for an adversarial accuracy of 8% and a mean L_2 norm of 41. Both are examples are failed, but they nonetheless lead to misclassification.

Model	Number of parameters	Time to train on a GPU (s)	Inference time per sample on a GPU (ms)
Small CNN	360k	2.4×10^3	0.74
Natural classifier	7.2M	8.6×10^4	1.8
Robust classifier	7.7M	1.2×10^6	2.7

Table II
CLASSIFIER COMPUTE COST

Model	Time to train the combinator on a GPU (s)	Attack time per sample on a GPU (ms)	Targeted PGD attack time per sample on a GPU (ms)
Small CNN	4.8×10^2	1.3	120
Natural classifier	2.5×10^3	1.3	663
Robust classifier	3.1×10^3	1.3	940

Table III
COMPUTE COST OF THE ATTACK

C. Computing cost

One crucial point to evaluate to consider the effectiveness of the attack is its compute cost compared to the classifier. If the attack is more expensive than training a classifier from scratch, then it becomes useless! The cost of training and inference for the classifier and the combinator are summarized in the tables II and III. For the attack cost, we also included the projected gradient descent, a standard adversarial attack [15] implemented by the ART library ². We can see that the cost of our attack is orders of magnitude lower than usual adversarial attacks. Moreover, the training time of the combinator is orders of magnitude lower than the one of the classifier. Note that for these results, the neuroevolution technique used by [12] to train the robust and natural models was maybe inefficient. For comparison, 94% accuracy on CIFAR10 can be achieved on a GPU for a ResNet architecture in the order of 10^4 s.³

As for inference time, the combinator is slower than the small CNN but faster than the natural and robust models. Nonetheless, a significant amount of computation needs to be performed from the attacker's side.

All in all, these tables show that the attack is computationally feasible, as it avoids the computing cost that comes with the training of performant image classifiers.

V. CONCLUSION AND LIMITATIONS

In this project, we demonstrate the feasibility of a new white box attack against neural image classifiers relying on the transfer of non-robust features naturally present in images. This attack requires a low compute cost such that it become possible to exploit the computing power of

²<https://adversarial-robustness-toolbox.readthedocs.io/>

³<https://dawn.cs.stanford.edu/benchmark/CIFAR10/train.html>

third parties machine learning models without revealing the image to classify. Nonetheless, the protection provided by this method is still to be determined, as it seems possible to extract features of the secret images from the container image (see VI-A). Moreover, misuse of online classifiers is still susceptible to being caught by human inspection. One can observe the decorrelation between the robust features of the input images and the outputted predictions. The applications of this new attack are not restricted to malicious use: it could also be a first step toward converting the current machine learning models in the cloud to a more privacy-preserving usage.

REFERENCES

- [1] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry, "Adversarial examples are not bugs, they are features," *arXiv preprint arXiv:1905.02175*, 2019.
- [2] G. F. Elsayed, I. Goodfellow, and J. Sohl-Dickstein, "Adversarial reprogramming of neural networks," *arXiv preprint arXiv:1806.11146*, 2018.
- [3] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1765–1773.
- [4] T. B. Brown, D. Mané, A. Roy, M. Abadi, and J. Gilmer, "Adversarial patch," *arXiv preprint arXiv:1712.09665*, 2017.
- [5] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.
- [6] T. Bai, J. Luo, J. Zhao, B. Wen, and Q. Wang, "Recent advances in adversarial training for adversarial robustness," *arXiv preprint arXiv:2102.01356*, 2021.
- [7] K.-A. A. Wei, "Understanding non-robust features in image classification," Ph.D. dissertation, Massachusetts Institute of Technology, 2020.
- [8] J. Kim, B.-K. Lee, and Y. M. Ro, "Distilling robust and non-robust features in adversarial examples by information bottleneck," *Advances in Neural Information Processing Systems*, vol. 34, 2021.
- [9] "Algorithms and terrorism: The malicious use of ai for terrorist purposes," <https://www.un.org/counterterrorism/sites/www.un.org.counterterrorism/files/malicious-use-of-ai-uncct-unicri-report-hd.pdf>, United Nations Office of Counter-Terrorism (UNOCT) and the United Nations Interregional Crime and Justice Research Institute (UNICRI), 2021.
- [10] S. Baluja, "Hiding images in plain sight: Deep steganography," *Advances in Neural Information Processing Systems*, vol. 30, pp. 2069–2079, 2017.
- [11] A. Li, J. Guo, H. Yang, and Y. Chen, "Deepobfuscator: Adversarial training framework for privacy-preserving image classification," *arXiv preprint arXiv:1909.04126*, 2019.
- [12] M. Sinn, M. Wistuba, B. Buesser, M.-I. Nicolae, and M. Tran, "Evolutionary search for adversarially robust neural networks," *Safe Machine Learning workshop at ICLR*, 2019.
- [13] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.
- [14] J. Rony, L. G. Hafemann, L. S. Oliveira, I. B. Ayed, R. Sabourin, and E. Granger, "Decoupling direction and norm for efficient gradient-based l2 adversarial attacks and defenses," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4322–4330.
- [15] F. Croce and M. Hein, "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks," in *International conference on machine learning*. PMLR, 2020, pp. 2206–2216.
- [16] N. Cammarata, S. Carter, G. Goh, C. Olah, M. Petrov, L. Schubert, C. Voss, B. Egan, and S. K. Lim, "Thread: Circuits," *Distill*, 2020, <https://distill.pub/2020/circuits>.

VI. APPENDIX

A. Visualizing the non-robust features of container images

We conducted an experiment to visualize the features of the secret image that were transferred to the container image. To this end, we trained a decoder network to reconstruct the secret image from the container image alone. We did not obtain significant results with the natural and robust network. A qualitative evaluation of the decoder for the small CNN model is presented in figure 9. Even if the reconstructed images are far from the secret ones, they show that certain features, such as the edges of the object, are encoded in the container images. Nonetheless, the color of the secret image seems lost, as the reconstructed images seem composed only of grayscale nuances. This result is in line with what was shown by [7] (section 4.1): the non-robust features seem to contain only little information about the color. By looking at the differences between the container and cover images, we can see that both the edges of the secret and the cover image are present. We hypothesize that the edges are highly informative features, as it has been shown in [16]. The combinator is smoothing the edges of the cover image to diminish their importance while transferring the edges of the secret image in an obfuscated way. We hypothesize that this general mechanism could also hold true for the robust and natural model. This could explain the blurriness of the container images presented above.

This experiment shows a promising way to visualize the non-robust features of an image.

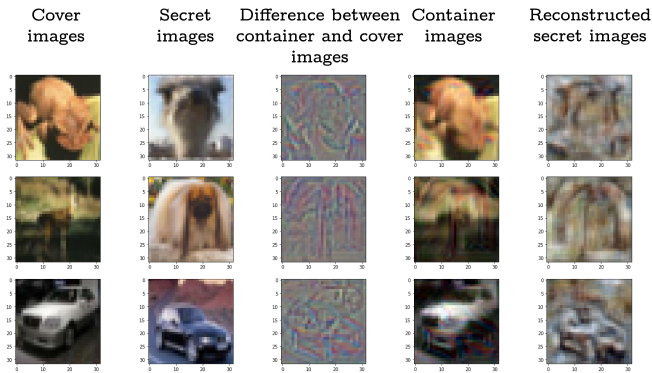


Figure 9. Reconstructed secret images from the container images, for the small CNN model. We also show the difference between the container and the secret image.

B. Definitions

To ease the communication, we introduced a specific terminology for the project, inspired from [10].

- *Secret image*: The image we want to classify without revealing it to the owner of the classifier.
- *Cover image*: The image that should be close to the input of the classifier for a human eye.

- *Container image*: The image given to the classifier. It is a combination of the secret and the cover image.
- *Adversarial accuracy*: The accuracy of the classifier for predicting the true labels of the secret images. This is the metric we want to maximize.
- *Cover accuracy*: The accuracy of the classifier for predicting the true labels of the cover images. If the attack on the CIFAR-10 dataset is perfect, this accuracy should be 10%: a random guess because secret and cover images are chosen at random, uniformly, and independently.

C. Code of the project

The attack can be run on a Google Colab notebook, and can be found in the GitHub repository <https://github.com/aVariengien/non-robust-feature-transfer>.