Generating natural adversarial Remote Sensing Images

Jean-Christophe Burnel, Kilian Fatras, Rémi Flamary, Nicolas Courty

Generating natural adversarial Remote Sensing Images

Jean-Christophe Burnel†, Kilian Fatras†, Rémi Flamary, Nicolas Courty

***Abstract*—Over the last years, Remote Sensing Images (RSI) analysis have started resorting to using deep neural networks to solve most of the commonly faced problems, such as detection, land cover classification or segmentation. As far as critical decision making can be based upon the results of RSI analysis, it is important to clearly identify and understand potential security threats occurring in those machine learning algorithms. Notably, it has recently been found that neural networks are particularly sensitive to carefully designed attacks, generally crafted given the full knowledge of the considered deep network. In this paper, we consider the more realistic but challenging case where one wants to generate such attacks in the case of a black-box neural network. In this case, only the prediction score of the network is accessible, given a specific input. Examples that lure away the network's prediction, while being perceptually similar to real images, are called natural or unrestricted adversarial examples. We present an original method to generate such examples, based on a variant of the Wasserstein Generative Adversarial Network. We demonstrate its effectiveness on natural adversarial hyperspectral image generation and image modification for fooling a state-of-the-art detector. Among others, we also conduct a perceptual evaluation with human annotators to better assess the effectiveness of the proposed method.**

***Index Terms*—Adversarial Examples, Generative models, Remote sensing, Deep Learning.**

I. Introduction

Deep neural networks (DNN) have established as a dominant class of learning models to handle Remote Sensing Images (RSI) on a vast amount of tasks, ranging from detection, classification or segmentation (see *e.g.* [1] for a comprehensive review). Their ability to handle a vast amount of data, both at train and test times, and to work on dedicated processing hardware, makes them ubiquitous nowadays in RSI analysis. Yet, it is well known in the machine learning community that those models are strongly sensitive to carefully crafted attacks also called adversarial attacks, that aim at fooling the network prediction based on imperceptible modifications of the submitted entry [2], [3]. This issue has been vastly underlooked for now in the remote sensing community, despite the fact that it poses significant security issues, as soon as decisions taken from RSI analysis can potentially have tremendous impact on security decision or evolution of public policies. This is notably the case in a military context [4], where an attacker could input malicious images to fool a detector system. The attack can occur both at the level of a computing system [5] (by manipulating the data flowing to the recognition system), or even at the physical level [6], by changing the physical properties of the signal arriving to the remote captor. These modifications could be envisaged both in the geometrical properties of the scene, but also in the spectral domain [7]. Moreover, RSI pose its own challenges with this respect to adversarial image generation because of the heterogeneity of captors, that generally go beyond RGB images. It is worth noting that the question of generating adversarial examples for multi- or hyper-spectral images, SAR, Lidar or even multi-modal inputs is still an open question, each tasks (*e.g.* detection, classification or segmentation) having its own specificities. We propose in this paper **a generic principle for building adversarial examples generators** in all those cases, and whenever the machine learning algorithm used for the analysis is solely known through given input-output pairs. While disposing of such a generator could be used for its first purpose of luring machine learning algorithms, we also foresee its use in cases where one wants to inspect the potential failure cases of a learning algorithm, or to design more robust models that are less sensitive to adversarial attacks.

Turning to the existing literature on adversarial examples, there exists two main strategies to generate such examples. The first strategy is to add a small perturbation to images that are correctly classified. The small perturbation fools the classifier which makes the wrong prediction. The small perturbation is computed from the classifier gradients. Intuitively, the gradient leads toward the direction with the most variation in the prediction. This is the main idea of the Fast Gradient Sign Method (FGSM) algorithm [2]. One problem from this approach is that one needs to know the full classifier architecture to compute the gradients. Another problem is that even a small perturbation can harm the image quality and lead to images that are not similar to original data anymore. Such images can be easily detected as adversarial images [8], [9]. The second paradigm does not require access to the classifier. It looks for adversarial examples only by having access to the classifier's prediction. Most of techniques using the second strategy try to find adversarial examples using Generative Adversarial Networks (GANs) [10]. After training a generator, several recent work [11], [12] look for adversarial vectors in the latent space and then generate adversarial data with the generator. This strategy is appealing as it gives them data which look like the true data but are wrongly classified.

†Equal contribution

JC. Burnel, K. Fatras and N. Courty are with the University Bretagne Sud, CNRS, IRISA, UMR 6074, France e-mail: jean-christophe.burnel@irisa.fr

R. Flamary is with Université Côte d'Azur, Laboratoire Lagrange, Observatoire de la Côte d'Azur.

Our proposed method follows the second strategy but rely on a different approach to generate natural adversarial examples. We train a GAN which is specialized for adversarial data generation. To this end we propose to weight the training data for training our adversarial GAN. The weight values depend on the probability for a datum to be misclassified. This is in opposition to classical GAN strategies that use uniform weights on samples in their dataset. Hence, our method only needs to know the output of a given pre-trained classifier, seen as a "black box" classifier. From the re-weighted distribution of the true data, a generator is trained to generate adversarial examples for the pre-trained classifier. The idea is to create a map between the latent space and the set of natural adversarial images which are present in the true data. Our approach is called ARGAN and stands for Adversarial Reweighted GAN in the following.

The paper is structured as follows: Section 2 introduces the definitions and notations. Then, in section 3, we detail related work on generative modelling in remote sensing and adversarial data generation. After a short introduction on GAN we present our **main contributions** in section 4. We detail our modified loss function which is designed for adversarial examples generation. Section 5 gathers experiments. First, the method is applied on hyperspectral data from the Data Fusion Contest 2018 [13], by generating data which are misclassified by a pre-trained classifier. Then as second experiment, a GAN is trained for image modification through a patch on Potsdam data [14]. Finally the third and last experiment is to train train a GAN in order to fool a state of the art detector on the Potsdam dataset.

Notations. This paragraph details formal notations and definitions. The probability distributions of true data is denoted $\mathbb{P}_r$ and the generated data distribution as $\mathbb{P}_G$. Vectors are expressed in bold, *i.e.,* $\boldsymbol{v}$. The true data of dimensionality d are denoted as $\boldsymbol{x} \sim \mathbb{P}_r$ and the noise vectors of dimensionality $p \ll d$ are $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}_p, \mathbf{I}_p)$. Data lie in a space $\mathcal{X}$. n stands for the number of true data and m for the batch size used during training. For a classifier $\boldsymbol{c}$, the probability to belong to the correct class of a sample $\boldsymbol{x}$ is denoted as $\boldsymbol{c}^y(\boldsymbol{x})$.

II. Related work

Definitions. This paragraph clusters the different kind of adversarial attacks, using notations from related work [2], [11], [12], [15]. We define $\boldsymbol{y}_{pred} = \boldsymbol{c}_\theta(\boldsymbol{x})$ the operation of getting the prediction $\boldsymbol{y}_{pred}$ of the datum $\boldsymbol{x}$ with the classifier $\boldsymbol{c}_\theta$. The classifier is parametrized by weights θ. $\mathcal{I} \triangleq \{[0,1]^n\}$ indicates the set of all normalized images with n pixels and $\mathcal{N}_\mathcal{I} \subseteq \mathcal{I}$ stands for the subset of all natural images. Natural images are meaningfully images similar to original images which make them legible. An oracle o is acknowledged where $o(\boldsymbol{x})$ is the ground truth of the considered problem. Now, we make a distinction about attacks regarding the knowledge of the classifier. A **white box attack** is an attack where there is an access to the classifier's structure and parameters θ. A **black box attack** is an attack where the only knowledge is the outputs $\boldsymbol{y}_{pred}$ for some inputs $\boldsymbol{x}$. Note that we suppose in the following that we have access to both $\boldsymbol{x}, \boldsymbol{y}_{pred}$ but we cannot query the classifier on new samples (since this would allow to compute approximated gradients similarly to FGSM). We now give a detailed description about misclassified example categories.

The idea of **perturbation-based example** is to add a small perturbation to an input in order to create an adversarial example. Most of the time the perturbation is limited by a factor ϵ. We give an example: let us denote $\boldsymbol{\xi}$ a unitary (*i.e.,* $\|\boldsymbol{\xi}\| = 1$) input perturbation, then a new example can be computed as $\boldsymbol{x}_{\text{adv}} = \boldsymbol{x} + \epsilon\boldsymbol{\xi}$. It is an adversarial examples if $c_\theta(\boldsymbol{x}) \neq c_\theta(\boldsymbol{x}_{\text{adv}})$. Formally, it can be written as $\{\boldsymbol{x}_{adv} \in \mathcal{I} \mid \exists \boldsymbol{x}_{test} \in \mathcal{I}, \|\boldsymbol{x}_{adv} - \boldsymbol{x}_{test}\| < \epsilon \wedge o(\boldsymbol{x}_{adv}) = o(\boldsymbol{x}_{test}) = c(\boldsymbol{x}_{test}) \neq c(\boldsymbol{x}_{adv})\}$. **Natural examples** are misclassified data-like images. It can be misclassified training or testing data for instance. Some perturbation-based examples are also natural examples. Formally, it can be written as $\{x_{adv} \in \mathcal{N}_\mathcal{I} \mid o(\boldsymbol{x}_{adv}) \neq c(\boldsymbol{x}_{adv})\}$. The last category is **unrestricted examples**. In this context there are both a pre-trained classifier and an oracle. Unrestricted examples are samples where the oracle and the classifier predict different result. Formally, it can be written as $\{\boldsymbol{x}_{adv} \in \mathcal{I} \mid o(\boldsymbol{x}_{adv}) \neq c(\boldsymbol{x}_{adv})\}$.

We are now ready to describe the attacks against the pre-trained classifier, *i.e.,* the different strategies to fool the classifier. **Untargeted attack** stands for an attack from any adversarial example. **Targeted attack** acknowledges an attack with an adversarial example from a source label, which is classified as a target label, $\boldsymbol{y}_{target} = \boldsymbol{c}(\boldsymbol{x}_{adv}) \neq \boldsymbol{y}_{source} = o(\boldsymbol{x}_{adv})$.

Generative Adversarial Networks. Generative Adversarial Networks have become a popular unsupervised method for data generation. Introduced in [10], several work came out to complete and improve their use. Methods were developed to control the label of generated data [16]. Other approaches focused on developing GANs for super resolution tasks [17]. The use of different statistical distances or regularizations has also been widely investigated in [18], [19], [20], [21], [22], [23]. Recently some work focused on image inpaiting using gans to reconstruct images [24] or even modify them with some given labels [25], lastly GAU-GAN [26] proposed to synthesize an image solely on given labels.

Generated modelling for remote sensing data. Generative modelling for remote sensing data has been investigated in several references. GANs were used by [27] to generate remote sensing data. They used a conditional GAN architecture, where they take as inputs an hyperspectral class and a random noise, to control the class of generated data. They showed that the generated spectra quality is genuine-looking and physically plausible. Furthermore, they experimentally validated that the generated samples can be used for data augmentation strategy to improve classification accuracy. The generation of super resolved remote sensing data was investigated in [28]. To this end they relied on two main subnetworks: an ultradense subnetwork and an edge-enhancement subnetwork combined with an adversarial learning strategy. GANs has also been used to learn unsupervised representations as done in [29]. They used the generator to generate data-like images and the discriminator was used as a feature extractor for classification purposes. [30] also used GANs for learning representation by

incorporating a non-local layer into their architecture. GANs have also been used in remote sensing for domain adaptation on hetereogenous data [31] which might be seen as transfer learning with different type of data.

Generated adversarial data. To generate adversarial examples without requiring the classifier's gradients, [11] uses a GAN and an invertor. The generator is a function which goes from the latent space to the true data space while the invertor is an inverse function. From the true data, they use the invertor to go to the latent space and look for adversarial examples. They add small perturbations to the latent vector z^* which represents a data x^* until they find an adversarial example. This method allows them to have very realistic adversarial examples without using the classifier's gradients. In [12] authors use an AC-GAN [16] to model the data distribution with the assumption that an ideal model could generate all the set of legitimate data. With such a model they search in the latent space for all the adversarial examples. Their search is done by minimizing the confidence score of a classifier while having the auxiliary classifier still predicting the correct class. So the adversarial examples look like true images and are adversarial for the attacked classifier.

III. Adversarial Reweighted GAN (ARGAN)

The section is divided as follows. We start this section with a brief introduction on GANs. After, we describe the link between probability distribution of the true data and image generation. Then, our probability distribution reweighting for adversarial data generation and finally, the use of minibatch to improve training.

A. Generative Adversarial Networks

Generative Adversarial Networks (GAN). The principle of a GAN [10] is to generate realistic data *w.r.t.* a training dataset. It has been expressed as a two player game between two DNNs, a generator and a discriminator. The discriminator tries to predict if an image is real or fake and the generator tries to fool the discriminator with its generated images. Intuitively to fool the discriminator, the GAN tries to minimize the distance between the distributions of generated data and training data. In this formalism the ability of the discriminator to separate generated and real data can be seen as a divergence between the two distributions. This notion of divergence is critical and has led to several variants of GAN. The generator takes as input $z \sim \mathbb{P}_{\boldsymbol{z}}$, to generate data $G(\boldsymbol{z}) \sim \mathbb{P}_G$, it forms the generator's distribution. Then, it tries to reduce the distance with the real data distribution $\mathbb{P}_r$. $\mathbb{P}_z$ is usually a gaussian or a uniform distribution. GANs improve the generated data quality by minimizing the Jensen-Shannon divergence between the distributions $\mathbb{P}_G$ and P_r:

$$\min_{\theta} \max_{\phi} \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_r}[\log \mathcal{D}_\phi(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{z}\sim\mathbb{P}_{\boldsymbol{z}}}[\log(1-\mathcal{D}_\phi(G_\theta(\boldsymbol{z})))]. \tag{1}$$

However as the true distributions $\mathbb{P}_G$ and $\mathbb{P}_r$ are unknown, we only consider their empirical distributions from the available true data and the generated ones.

Wasserstein GAN. There are several statistical distances that can be used for measuring the distance between probability distributions. Wasserstein GAN [18] is a GAN variant which relies on optimal transport tools to compare two probability distributions $(\mathbb{P}_r, \mathbb{P}_z) \in \mathcal{M}_+^1(\mathcal{X}) \times \mathcal{M}_+^1(\mathcal{X})$. Optimal transport seeks a transportation map which minimizes the displacement cost between the distributions with respect to a ground metric on the input space $\mathcal{X}$ [32]. The ground metric is usually taken as the euclidean distance. Formally, the Wasserstein distance W_p that relies on optimal transport between two distributions can be expressed as :

$$W_p(\mathbb{P}_r, \mathbb{P}_z) = \min_{\pi \in \boldsymbol{U}(\mathbb{P}_r, \mathbb{P}_z)} \Big(\int_{\mathcal{X}\times\mathcal{Y}} \|\boldsymbol{x}-\boldsymbol{y}\|_2^p d\pi(\boldsymbol{x},\boldsymbol{y}) \Big)^{1/p}, \tag{2}$$

where $\boldsymbol{U}(\mathbb{P}_r, \mathbb{P}_z)$ is the set of joint probability distribution with marginals $\mathbb{P}_r$ and $\mathbb{P}_z$ such that $\boldsymbol{U}(\mathbb{P}_r, \mathbb{P}_z) = \big\{\pi \in \mathcal{M}_+^1(\mathcal{X},\mathcal{Y}) : \boldsymbol{P}_{\mathcal{X}}\#\pi = \mathbb{P}_r, \boldsymbol{P}_{\mathcal{Y}}\#\pi = \mathbb{P}_z\big\}$. $\boldsymbol{P}_{\mathcal{X}}\#\pi$ (resp. $\boldsymbol{P}_{\mathcal{Y}}\#\pi$) is the marginalization of π over $\mathcal{X}$ (resp. $\mathcal{Y}$). When the ground cost is chosen as the Euclidean distance on $\mathbb{R}^d$, W_p is a metric as well. Notably, W_1 can be rewritten with the Kantorovich-Rubinstein duality [32], giving :

$$W_1(\mathbb{P}_r, \mathbb{P}_z) = \min_{\theta} \max_{f \in \mathtt{Lip}^1(\mathcal{X})} \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_r}[f(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{z}\sim\mathbb{P}_z}[f(G_\theta(\boldsymbol{z}))],$$

where $\mathtt{Lip}^1(\mathcal{X})$ is the set of 1-Lipschitz functions. [18] proposed to use the following approximation:

$$W_1(\mathbb{P}_r, \mathbb{P}_z) = \min_{\theta} \max_{\phi} \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_r}[\mathcal{D}_\phi(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{z}\sim\mathbb{P}_z}[\mathcal{D}_\phi(G_\theta(\boldsymbol{z}))], \tag{3}$$

where $\mathcal{D}$ is the dual potential and is within the set of 1-Lipschitz functions parameterized by weights ϕ. Analogous to GANs, we still call $\mathcal{D}$ the "discriminator" although it is actually a real-valued function. In order to respect the 1-Lipschitz constraint, [18] used a weight clipping trick for the discriminator's weights during the optimization procedure. However this practice leads to unstability during optimization and poor minima. Another approach proposed in [33] involves a gradient penalty and enforces the gradient to be less or equal to one making $\mathcal{D}$ 1-Lipschitz. The WGAN variant used in the rest of the paper is:

$$\begin{aligned}\min_{\theta} \max_{\phi} \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_r}[\mathcal{D}_\phi(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{z}\sim\mathbb{P}_z}[\mathcal{D}_\phi(G_\theta(\boldsymbol{z}))] \\ +\lambda \mathbb{E}_{\hat{\boldsymbol{x}}\sim\mathbb{P}_{\hat{x}}}[(\max\{0, \|\nabla\mathcal{D}_\phi(\hat{\boldsymbol{x}})\|_2 - 1\})^2],\end{aligned} \tag{4}$$

where $\mathbb{P}_{\hat{x}}$ is the distribution of samples along the straight lines between a pair of points from $\mathbb{P}_r$ and $\mathbb{P}_G$ and λ the regularization parameter. In practice, the true distribution $\mathbb{P}_r$ is unknown and only an empirical counterpart $\widehat{\mathbb{P}}_r$ with n *i.i.d.* samples from $\mathbb{P}_r$ is available.

Furthermore, we investigated the use of an AC-GAN which would allow us to control the label of the generated data. Unfortunately, AC-GAN learns a biased distribution [34], the distribution of easy to classify data. Since the learned distribution is the opposite distribution of what the wanted distribution, we chose to extend WGAN instead of AC-GAN.

B. Adversarial Reweighting

To generate data similar to the true data and are diversified, each true data have a uniform weight in the classical GAN

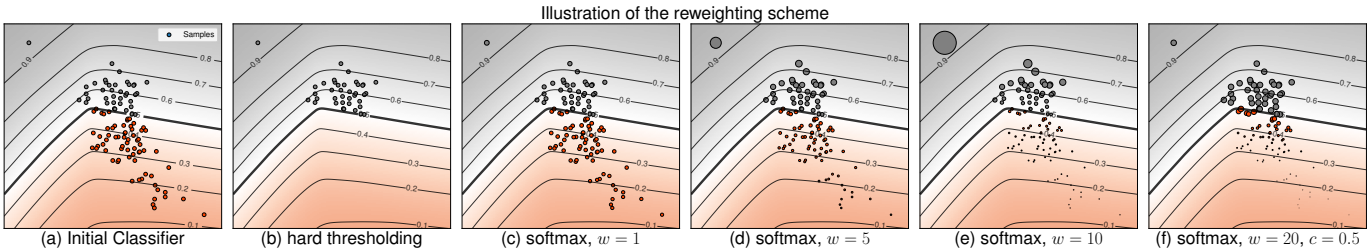


Fig. 1: Illustration of different reweighting strategies for adversarial data generation. (a) is the standard uniform weight between data. (b) is a hard weighting where we only consider misclassified data. (c), (d) and (e) are softmax weighting strategy for different temperatures. (f) is the combination of softmax and clipping strategies.

training. Formally, it means that the empirical distribution $\widehat{\mathbb{P}}_r$ is uniform, i.e $\widehat{\mathbb{P}}_r = 1/n \sum_i^n \delta_{x_i}$. It is customary for empirical distributions to suppose that the samples are drawn *i.i.d.,* from the underlying distribution. One way of changing GAN's goal is to change the empirical distribution $\widehat{\mathbb{P}}_r$. Instead of encouraging the generator to generate data close to the true data, it is encouraged to generate realistic data that make the classifier fail. It means that the true data which are misclassified have a bigger weight in the GAN training than correctly classified data. Our main idea is to reweight a datum's weight according to the classifier's prediction. The resulting weighted distribution, denoted $\widehat{\mathbb{P}}_r^a$, must have the following form: $\widehat{\mathbb{P}}_r^a = \sum_{i=1}^n p_i \delta_{x_i}$, where: $\sum_{i=1}^n p_i = 1$. The GAN is trained to minimize the distance between the generator distribution $\widehat{\mathbb{P}}_G$ and the reweighted true distribution $\widehat{\mathbb{P}}_r^a$. Finally, our new loss function is of the form:

$$\min_\theta \max_\phi \mathbb{E}_{\boldsymbol{x}\sim\widehat{\mathbb{P}}_r^a}[\mathcal{D}_\phi(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{z}\sim\mathbb{P}_z}[\mathcal{D}_\phi(G_\theta(\boldsymbol{z}))] + \lambda \mathbb{E}_{\hat{\boldsymbol{x}}\sim\widehat{\mathbb{P}}_{\hat{x}}}[(\max\{0, \|\nabla \mathcal{D}_\phi(\hat{\boldsymbol{x}})\|_2 - 1\})^2]. \quad (5)$$

Intuitively, the generator is a map between the latent space and the adversarial data area of the true distribution. To the best of our knowledge, it is the first time the distribution is modified for generative purpose. Now we describe and review the impact of different reweighting methods.

C. Reweighting data distributions

We give a list of the different weight strategy for $p(\boldsymbol{x})$. We start by recalling the basic GAN case.

Uniform weight. Having a uniform true data distribution corresponds to the typical GAN setting. Each datum is given the same weight, hence there is no particular reason for the generator to produce adversarial examples. It can be visualized in figure 1.(a) where there is a distribution where all points got the same weight, $p(\boldsymbol{x}) \triangleq 1/n$.

Hard weighting. An intuitive way to generate adversarial examples is to only consider misclassified data, where a constant weight is given to misclassified data and 0 for correctly classified data. Let N_m be the number of misclassified data, then the misclassified data are normalized weights as $1/N_m$. The downside of this weighting strategy is that few data are left for training in the context of a very accurate classifier. And unfortunately, GANs are data hungry which means that this method is not efficient to generate adversarial examples. With $\boldsymbol{c}(\boldsymbol{x})$ the result of the classifier for the targeted class $p(\boldsymbol{x}) \triangleq \lfloor 1 - \boldsymbol{c}(\boldsymbol{x}) \rceil$.

Soft weighting. Another weighting approach is to use the prediction score from the classifier. Such approach could take into account examples which are correctly classified but with a low confidence. We refer to those samples as *soft adversarial examples*. In order to consider the soft adversarial examples, we can use a softmax function. It is defined as:

$$S(\boldsymbol{c}^y(\boldsymbol{x}), w) \triangleq \frac{\exp(w * [\boldsymbol{c}^y(\boldsymbol{x}) - \boldsymbol{c}^y(\boldsymbol{x})_{\max}])}{\sum_{i=0}^{K} \exp(w * [\boldsymbol{c}^y(\boldsymbol{x}_i) - \boldsymbol{c}^y(\boldsymbol{x})_{\max}])}, \quad (6)$$

where $\boldsymbol{c}^y(\boldsymbol{x})$ is the prediction to belong to the correct class, $\boldsymbol{c}^y(\boldsymbol{x})_{\max}$ is the maximal probability among the batch of data to belong to the correct class and w is a temperature coefficient that controls the entropy of the resulting distribution. However as the objective is to generate adversarial data, we consider $1 - \boldsymbol{c}^y(\boldsymbol{x})$ rather than $\boldsymbol{c}^y(\boldsymbol{x})$. The soft weighting strategy can be expressed as $p(\boldsymbol{x}) \triangleq S(1 - \boldsymbol{c}^y(\boldsymbol{x}), w)$. Figure 1.(c) is an example of weighting using the softmax function with a temperature $w = 5$. Note that in the specific case of $w = 0$, the softmax approach corresponds to the WGAN loss function and when w tends to infinity, it becomes the hard weighting strategy.

To demonstrate the relevance of the different reweighting methods, we use a toy dataset to illustrate the softmax approach. See Figure 1. (a) to (e), where we represent data from a certain class with points cloud, the classifier's decision boundary with a black line, so that points under the line are correctly classified and points over the line are misclassified. When we get closer and beyond the classification line, the point clouds become bigger with a bigger weight. However, this might be problematic in the presence of outlier data or in the presence of a variation of misclassified prediction. For instance, in the presence of few misclassified data with high inaccurate predictions and a majority of misclassified data with medium inaccurate prediction, the latter get bigger weights than the former as shown in figure 1.(e).

Weight clipping. In order to make the softmax strategy more robust we decide to apply a clipping function to all prediction vectors $p(\boldsymbol{x})$. Let us denote the threshold κ and the threshold function t. It means that if the prediction vector $p(\boldsymbol{x})$ is above the threshold, it is clipped to the threshold value. The clipping function is applied to the prediction vectors before the

softmax strategy. In our experiments, we selected a threshold of 75%. Putting all together, the weighted distribution is :

$$p(\boldsymbol{x}) \triangleq S(\min(\kappa, (1 - \boldsymbol{c}^y(\boldsymbol{x})), w). \tag{7}$$

D. Minibatch weighting

We numerically found that computing the reweighting directly on the full distributions gives numerical instabilities when batches with small weighted data are selected. A possible solution to this problem is to compute the reweighting strategy on batches from the distributions. The loss function becomes an expectation over mini-batches:

$$\min_{\theta} \max_{\phi} \mathcal{L}(\boldsymbol{x}, \boldsymbol{z}, \theta, \phi), \tag{8}$$

$$\begin{aligned} &\text{where } \mathcal{L}(\boldsymbol{x}, \boldsymbol{z}, \theta, \phi) = \\ &\mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_r^{a} \otimes m}[\mathcal{D}_\phi(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{z} \sim \mathbb{P}_z^{\otimes m}}[\mathcal{D}_\phi(G_\theta(\boldsymbol{z}))] \\ &+ \lambda \mathbb{E}_{\hat{\boldsymbol{x}} \sim \mathbb{P}_{\hat{x}}^{\otimes m}}[(\max\{0, \|\nabla \mathcal{D}_\phi(\hat{\boldsymbol{x}})\|_2 - 1\})^2], \end{aligned}$$

where m is the minibatch size. This strategy is similar to the minibatch Wasserstein distance which was widely used for generative models [23], [35] and studied in [36].

Regarding the extreme cases, two effects can be highlighted from the classifier's performance. In the case of a performing classifier, i.e really high overall accuracy, the softmax term is close to 1 for each datum. In this case the original WGAN is recovered, the generated data look like the true data but they are not adversarial. However if the classifier is not competitive, the generated data are adversarial, showing the trade-off between the generation of adversarial data and the generation of real data.

IV. Experiments and Results

In this section we describe the different experiments and results. ARGAN is applied on synthetic data and compared to the well known WGAN. The first experiment on real-life data is the generation of adversarial hyperspectral images. Then, image modification through a patch is considered on IRRGB images. Finally, the method aims at fooling a state of the art detector with generated data and compared it with a usual WGAN. For all experiments the used optimizer is a RMSProp optimizer [37], with a learning rate of 0.00005 for the generator and 0.0001 for the critic.

A. Synthetic data

The first experiment illustrate ARGAN on the well known two moons toy dataset with 4000 points. On this toy task, we generate data which belong to class 1 but are classified as belonging to class 2, i.e., the white point clouds in figure 2. The classifier used is a pre-trained dense 2-layer with a classification accuracy of 92%. The classifier boundary is represented as a black line. The generator and discriminator share the same dense 3 hidden layers architecture of size respectively of 128, 128 and 64. The input noise dimension $\boldsymbol{z} \sim \widehat{\mathbb{P}}_z$ is 10. The batch size is 256 and the networks is trained for 1000 epochs. We then train WGAN and ARGAN to see in which zone they generate class 1 data. To estimate

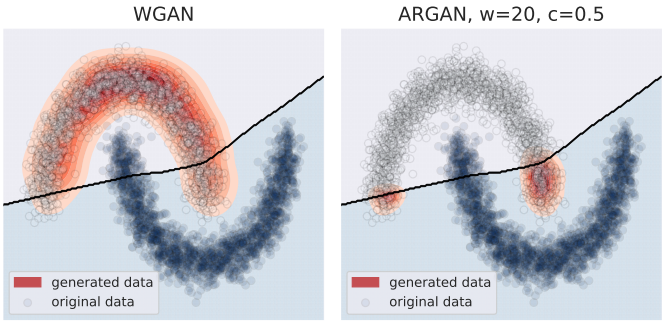


Fig. 2: Adversarial data generation with WGAN (left) and ARGAN (right) on two moons dataset. The black line is the classifier boundary. The kernel density estimation of generated data is in red.

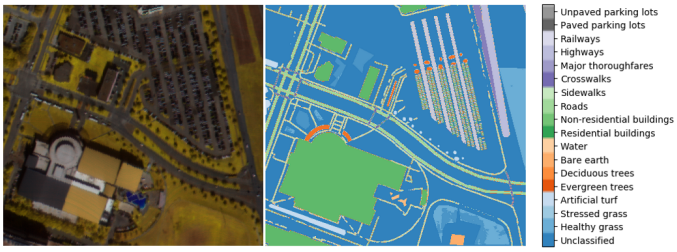


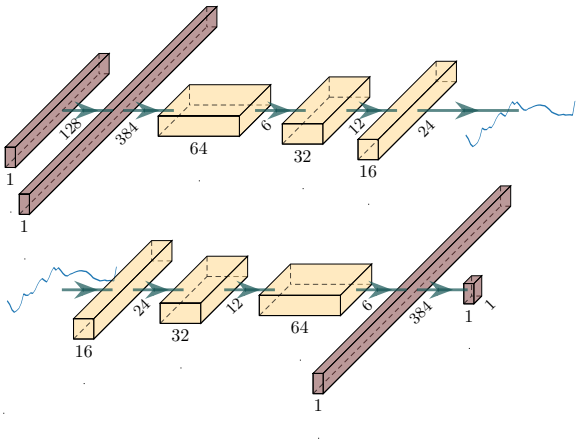
Fig. 3: Data from the DFC2018 dataset. Example of false RGB image (left) and the ground truth (right).

the generation area, 1500 samples are generated and a kernel density estimation of these data is performed. For ARGAN, the temperature w is set to 20 and the clipping value $\boldsymbol{c}$ is 0.5. We see that WGAN generates data all over the class 1 while ARGAN only generates data where the points are misclassified by the classifier. This shows the effectiveness of our method to focus on generating adversarial data for a pre-trained classifier.

B. Generation of hyperspectral spectra

We now investigate the effectiveness of ARGAN on real world data. Unlike traditional RGB images, hyperspectral imagery divides the color spectrum in multiple contiguous bands that can outreach the visible spectrum. In those images a pixel is a spectrum, and we can identify the materials in a pixel with an analysis of the pixel spectral signature. To test our algorithm, a pixel-based classification task on hyperspectral images is first considered. The classifier takes a spectrum as input and gives a material classification probability vector as output.

Dataset and classifier. The DFC2018 dataset [13], which was acquired over the University of Houston campus and its neighbourhoods, is considered in this experiment. The hyperspectral data cover a 380-1050 nm spectral range with 48 bands at a 1-m Ground Sampling Distance. Here the classes do not only cover urban classes (buildings, cars, railways, etc.) but it also covers vegetation classes (healthy or stressed grass, artificial turf, etc.). An overview of the dataset is given with figure 3, where we only take 3 of the 48 bands for visualization purpose, and the ground truth labels are shown in



Generator	Critic
INPUTS: 128×1	INPUTS: 48×1
Dense 128 → 384, ReLU	Conv1D k=3 stride=2, 24×16 ReLU
Reshape 384 → 6×64	Conv1D k=3 stride=2, 12×32 ReLU
Conv1D$^\intercal$ k=3 stride=2, 12×32 ReLU	Conv1D k=3 stride=2, 6×64 ReLU
Conv1D$^\intercal$ k=3 stride=2, 24×16 ReLU	Reshape 6×64 → 384
Conv1D$^\intercal$ k=5 stride=2, 48×1 TanH	Dense 384 → 1

Fig. 4: Generator (Top) and critic (Bottom) 3 convolutional layer architectures to generate adversarial hyperspectral data.

the same figure. The considered classifier is based on [38] and have an overall accuracy of 52% with $\kappa = 0.43$ on disjoint train/test, which is consistent with the result observed in recent reviews [39]. The objective is to generate natural adversarial hyperspectral data for this pre-trained classifier.

Experimental setting. For this experiment, we investigate different adversarial class spectra: adversarial healthy grass spectra, adversarial car spectra and adversarial cross-walk spectra. For the WGAN architecture, we choose 1-dimensional convolution layers for both the generator and the critic as it keeps coherence between close spectral bands. The architecture is detailed in Figure 4. We use the whole dataset with a batch size of 64, λ is set to 20 and in this experiment w is set to 20. Regarding the classifier, its accuracy performance is 82% for the healthy grass spectra on the whole dataset, on the Cars class the classifier has an accuracy of 39% and Crosswalks where the classifier has an accuracy of only 5%. This allows us to discuss the quantity of adversarial generated data according to the classifier performance on a specific class.

Results. Our approach can generate adversarial spectra, so in order to create an image we generate adversarial spectra for all the pixels of the class we attack. A visualization of the adversarial image for healthy grass can be viewed in Figure 5 in false colors. To evaluate the pertinence of ARGAN we need to check two aspects of the generated spectra. We first need to check that the generated spectra indeed belong to the correct class which can be done partially by comparing spectra means and standard deviations. Then we need to check that the classification performance is lower on the generated spectra than the original spectra to know if we are actually able to generate adversarial spectra.

We apply the different methods for healthy grass spectra generation and we check if the generated data belong to the

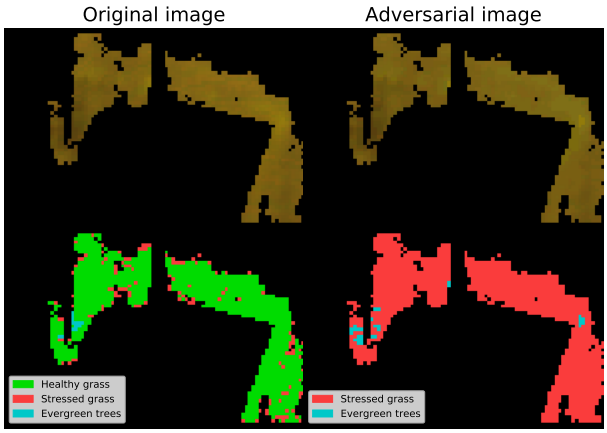


Fig. 5: [Best viewed in color] Healthy grass visualization. (Top) False RBG image built from original spectra (left) and from adversarial spectra (right). (Bottom) Classifier prediction on the original spectra (left) and adversarial spectra (right).

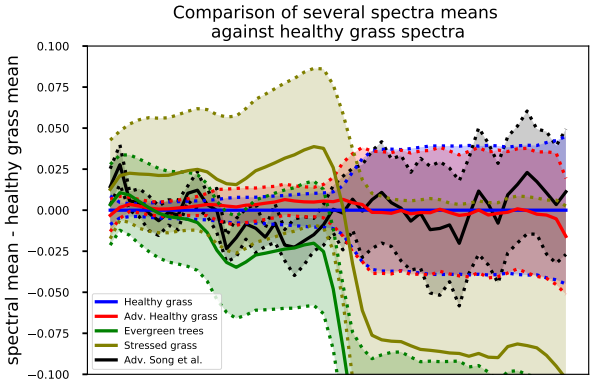


Fig. 6: [Best viewed in color] Comparison between several class spectra means against healthy grass spectra. All means are reported centered around the mean spectrum of healthy grass for better visualization. The spectra means are denoted in plain line and the standard deviations are in dotted lines.

target class. Figure 6 gathers all spectra statistics. We compare the target class against the generated adversarial examples from ARGAN, the generated adversarial examples from [12] and the two most predicted classes by the classifier (evergreen trees and stressed grass classes). In both cases we see that the ARGAN adversarial spectra statistics match the targeted class better than the predicted class and state of the art adversarial data generation. Hence, ARGAN generated data belong to the correct class and is more convincing than the adversarial data from [12].

We now compare the (mis)-classification performance between original data and generated adversarial data. We use the pre-trained network on healthy grass spectra from the original image, where its accuracy is 88.77%. Then, we use the network over the adversarial spectra. An illustration of difference in prediction can be found in Figure 5. The classification performance are gathered in Table I. The pre-trained classifier has an overall accuracy of 82% on real

TABLE I: Classification accuracy for the pre-trained classifier over ARGAN generated spectra and real data (over 10 runs)

	Mean	Std	Real data
Healthy Grass	0.34	0.06	0.82
Car	0.19	0.05	0.39
Crosswalks	0.01	0.009	0.05

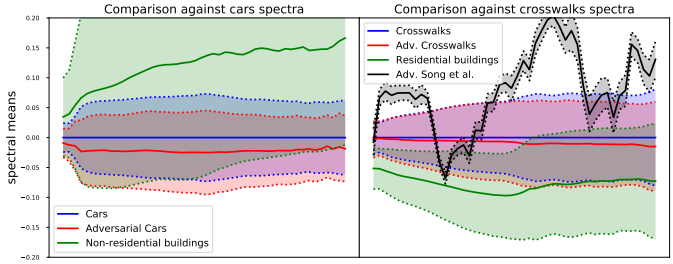


Fig. 7: [Best viewed in color] Comparison between several class spectra means against car and cross-walk spectra. All means are reported centered around the mean spectrum of car or cross-walk for better visualization. The means are in plain and the standard deviations are in dotted lines.

healthy grass spectra while its performance decreases by 50% over our generated spectra. It shows us that the classifier does not succeed to correctly classify ARGAN generated spectra as healthy grass spectra, making ARGAN generated spectra adversarial healthy grass spectra.

We now investigate the performance of ARGAN to generate adversarial cross-walk and cars spectra. The spectra statistics can be found in Figure 7. We compare the target class, ARGAN generated data, the state of the art adversarial data and the most predicted class by the classifier. We can see that ARGAN generated data fits better the target class than the most predicted class by the classifier or the adversarial data when those are available. We deduce that ARGAN generated data indeed belong to the target class. Regarding the classifier accuracy, we see in table I that the classification accuracy of ARGAN generated spectra is smaller than for real data, it decreases by 20% for the car class and 4% for the cross-walk class. This makes ARGAN generated spectra adversarial. We also see that there is a correlation between the classifier performance and the number of adversarial examples per batch. Indeed, better the classifier is, smaller is the number of natural adversarial data. This phenomenon is expected as when the classifier has a big accuracy, the number of adversarial data in the training set is small. Not only there is less misclassified data, but also the margin between well-classified data and the classification boundary tends to increase.

Qualitative comparison with state of the art. We now compare more deeply ARGAN with [12]*. We adapted their method to the same architectures considered here. We used untargeted attacks, which means that we want our spectra to be misclassified but we do not want it to be classified as a specific label. The results are visible with figure 6,7. Regarding the Healthy grass class, the produced adversarial spectrum exhibits a lot of noise and does not fit the target class as good as ARGAN generated spectra. Finally for the Crosswalks class, their algorithm was not able to produce spectra that fit the real spectra unlike ARGAN.

*following their online implementation : https://github.com/ermongroup/generative_adversary

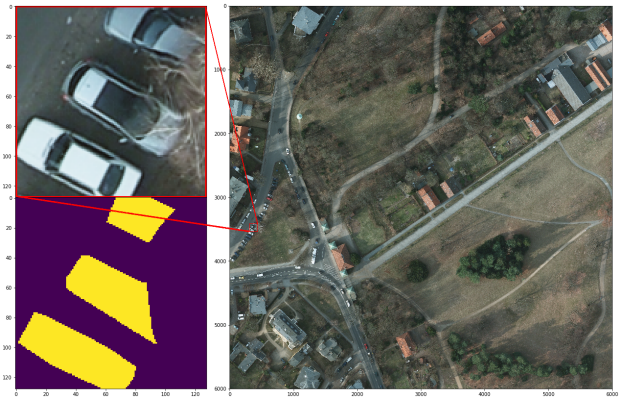


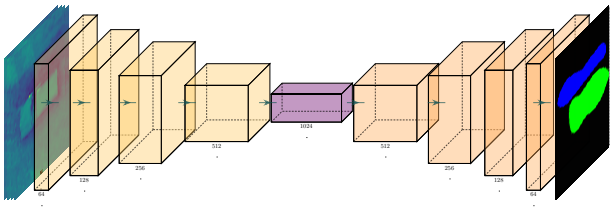
Fig. 8: example of Potsdam dataset patch[14]

C. Mask modification

We now describe the second experiment which objective is not to generate an image but to modify one in order to fool a pre-trained classifier. In order to modify an image, we focus on a mask, denoted $\boldsymbol{M}$, on the center of the image. A mask is a centered square subpart of the image and we aim at modifying what is inside this mask to create an adversarial data. Our images have pixels which are composed of Red-Green-Blue-InfraRed channels. In this task of semantic segmentation each pixel has a label, hence we use both spectral and spatial information unlike the first experiment.

Dataset. We consider the Potsdam dataset [14] which is a dataset composed of 38 6000x6000 pixels patches over the city of Potsdam, Germany. The patches are true orthophotos with four channels red, green, blue, infrared and have a GSD of 5-cm (see example in Figure 8), making it possible to see clearly objects such as cars. We used 28 images as training set and the remaining 10 images as test set. In total, we have 9138 cars in the dataset and from the patches, small images of cars of size 128 by 128 where extracted using center of mass of individual cars on patches. Regarding the learning task, we use a segmentation network (segnet) [40] (see an illustration in Figure 9).A common way to evaluate its performances it to use a confusion matrix (Figure 10).

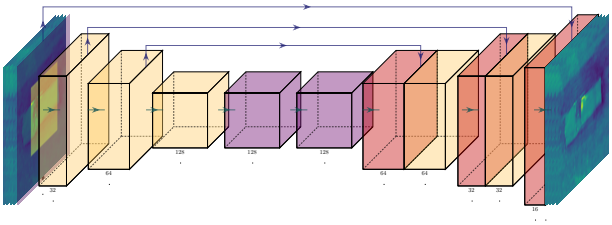
Experimentation. The purpose of this experiment is to modify the car segmentation of a given image in order to make it adversarial for the given pre-trained classifier. In order to achieve this, the car segmentation is within a 64 by 64 mask in the center of the images. The generator is a U-Net [41] and takes as inputs both the image and the mask instead of a latent vector $\boldsymbol{z}$, then it outputs a single new image as shown in figure 11. The generator works as follows: inputs are first compressed in latent information through convolutional and pool layers, then the resulting latent vector is decoded with transposed convolutional layers to give a single new image. With a new designed loss function, the GAN aims at modifying only what is inside the mask. Our modification task can be viewed as



INPUTS: 128×128×4
Conv2D k=3, 128×128×64, ReLU, MaxPool
Conv2D k=3, 64×64×128 ReLU, MaxPool
Conv2D k=3, 32×32×256 ReLU, MaxPool
Conv2D k=3, 16×16×512 ReLU, MaxPool
Conv2D k=3, 8×8×1024 ReLU, upsample
Conv2D k=3, 16×16×512 ReLU, Conv2D k=3, 16×16×512 ReLU, upsample
Conv2D k=3, 32×32×256 ReLU, Conv2D k=3, 32×32×256 ReLU, upsample
Conv2D k=3, 64×64×128 ReLU, Conv2D k=3, 64×64×128 ReLU, upsample
Conv2D k=3, 128×128×64 ReLU, Conv2D k=3, 128×128×32, ReLU
Conv2D k=3, 128×128×6, Softmax

Fig. 9: Segmentation network architecture used for detection on the Potsdam dataset [14]

Fig. 10: Confusion matrix of our segmentation network on the Potsdam dataset [14]



INPUTS: 128×128×5
GatedConv k=5, s=2, 64×64×32, LReLU
GatedConv k=3, s=2, 32×32×64, lrn, LReLU
GatedConv k=3, s=2, 16×16×128, lrn, LReLU
GatedConvDilat k=3, r=2, 16×16×128, lrn, LReLU
GatedConvDilat k=3, r=4, 16×16×128, lrn, LReLU
GatedDeconv k=3, s=2, 32×32×64, lrn, LReLU
GatedConv k=3, 32×32×64, lrn, LReLU
GatedDeconv k=3, s=2, 64×64×32, lrn, LReLU
GatedConv k=3, 64×64×32, lrn, LReLU
GatedDeconv k=3, s=2, 128×128×4, lrn, LReLU
GatedConv k=5, stride=1, 128×128×4, TanH

Fig. 11: Generator architecture designed for mask modification on the Potsdam dataset [14]

image inpainting and following this analogy, we designed a network close to [25] which showed impressive results using dilated convolutions [42].

We now give more details about the training procedure. We used a batch size of 32 and λ is set to 50. Regarding the modified loss function, we add a regularization term to the generator's loss, eq. 9, to ensure that only the mask $\boldsymbol{M}$ is modified and we also add an optional term to favor adversarial segmentation. The latter modification of the loss function, penalizes the GAN when it modified other pixels than the car pixels. Both of these terms can be weighted by constants α and β, and taking only $\alpha = \beta = 1$ gave meaningful results. In this experiments, the modified loss function is equal to:

$$\begin{aligned}\mathcal{L}_2(\boldsymbol{x}, \boldsymbol{M}, \boldsymbol{y}, \boldsymbol{y}_{pred}, \theta, \phi) = \\ \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_r^{a}\otimes m}[\mathcal{D}_\phi(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{z}\sim\mathbb{P}_z^{\otimes m}}[\mathcal{D}_\phi(G_\theta(\boldsymbol{x}, \boldsymbol{M}))] \\ + \lambda\mathbb{E}_{\hat{\boldsymbol{x}}\sim\mathbb{P}_{\hat{x}}^{\otimes m}}[(\max\{0, \|\nabla\mathcal{D}_\phi(\hat{\boldsymbol{x}})\|_2 - 1\})^2] \\ + \alpha\sum_{i=0}^{m}((1-\boldsymbol{M}) * (G(\boldsymbol{x}, \boldsymbol{M})_i - \boldsymbol{x}_i))^2 \\ - \beta\sum_{i=0}^{m}(\boldsymbol{y} * \boldsymbol{M})\log[(1-\boldsymbol{y}_{pred}) * \boldsymbol{M}]. \end{aligned} \quad (9)$$

Results. The results on the test set are gathered in Figure 12. We see in the first line two different cases where ARGAN performed well. In the first case few modifications of the image led to huge difference in segmentation, the differences for the RES column are in red if the car is not segmented anymore or in green if the car is better segmented.

In the second case we have much more modification to erase the car from the segmentation. The second line shows two failure cases, when we have the case where our modification, even if the segmentation is altered, modified the car heavily and it does not look as a natural image. And last, the modification can lead to a better segmentation. Note that these failure cases do not happen often. This can be seen with the following perceptual evaluation that we designed to investigate if our results can be convincing and adversarial.

Perceptual evaluation. To assess the quality of our generated patches we conduct a perceptual evaluation where we compare the ability of the trained classifier and humans to classify adversarial samples. The classifier performance is evaluated using the test set, for both original and mask modified images. And as we can see with the first line of Table II, the classifier looses 28 points of accuracy when it is evaluated on data from ARGAN. However those results are only meaningful if those are real natural adversarial example

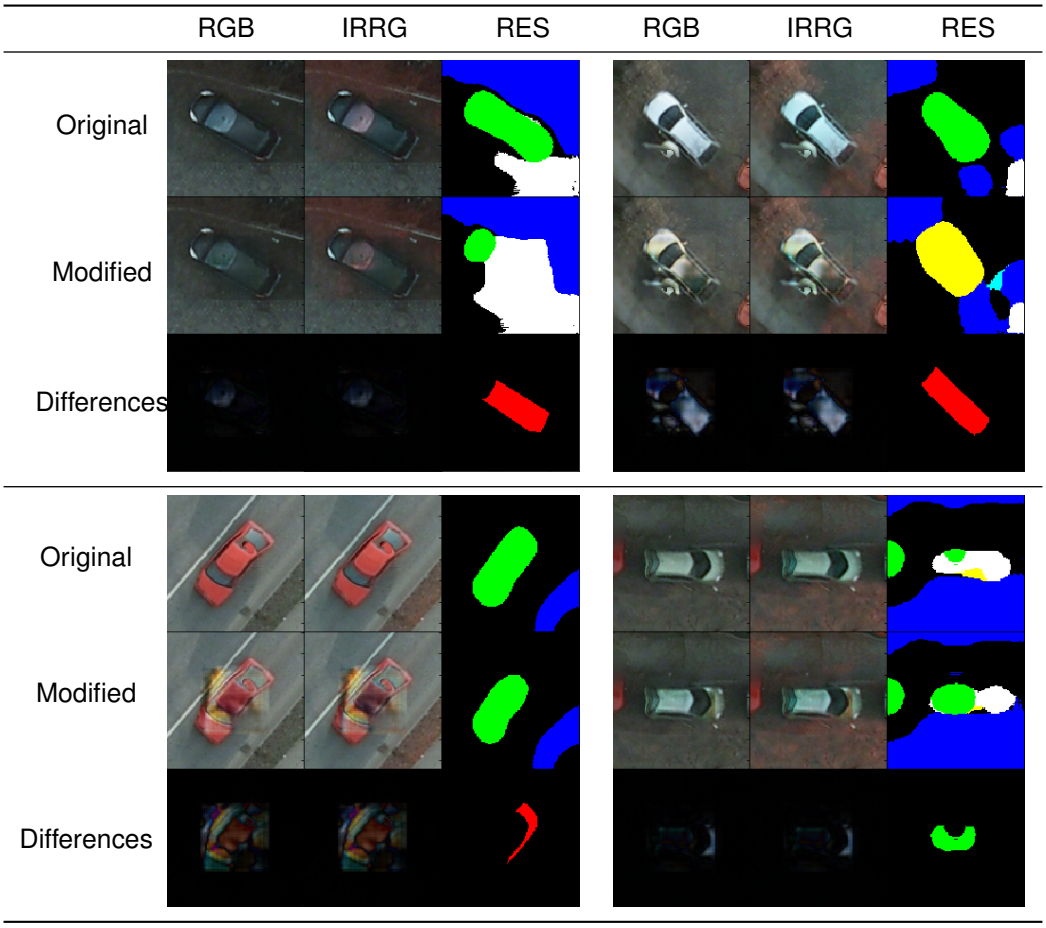


Fig. 12: Results for patches modification, the first line show example where our method worked, the second line show failure cases

as we defined it in the first section, meaning that the generated patches are indeed cars. To this end we conducted a perceptual evaluation similar to [12], with the assumption that if a human can detect a car then its segmentation is trivial. This allows us to have a much simpler task to conduct while having comparable results. We conducted the perceptual evaluation by first taking randomly 50 images where we had cars and 50 images where there were no cars, and then we applied our method to the 50 images with cars, letting us with 150 images in total. We have developed a simple test where 12 images are presented to humans. Among these images 5 are ground truth cars, 5 were modified using our method and 2 do not have cars in them. In this test you must indicate whether you see a car or not and associate this decision with a confidence level. This test was carried out by 74 persons and the results are gathered in the second and third lines of Table II. Humans loose 8 points of accuracy, however their confidence remains of the same order: Moderately confident. When we compare the results, we see a 36% drop of performance for the classifier against 9% for Humans, meaning that our method affected far more the classifier than the humans and that our method produces convincing adversarial examples.

	Original image	Modified image	Image w/o cars
Classifier accuracy	0.762	0.481	/
Human accuracy	0.918	0.835	0.945
Mean confidence	2.524	2.240	2.229

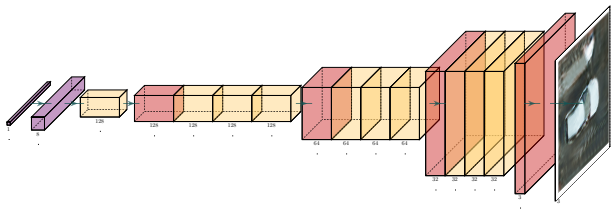
TABLE II: Results of perceptual evaluation

D. Car generation

In the last experiment we aim at evaluating ARGAN's performance when dealing with state-of-the-art predictors. The purpose is to generate adversarial car examples for a very good object detection algorithm. Instead of semantic segmentation where each pixel has a label, the classifier output is bounding boxes surrounding an object. In this experiment, we solely focus on the cars object class object.

Dataset. We use the same Potsdam dataset [14] than in previous experiment. We transformed the potsdam segmentation images to different images with bounding boxes around cars. To train the classifier we used the same train set as the patch modification experiment. However to train the GAN, we used both training and testing sets as done in our first experiment, which allows to increase the number of training data.

Experimentation. The selected detector is a YoloV3 [43] that we trained on the train set. This classifier achieves an overall (Train and Test) accuracy score of 99.5% with an



INPUTS: 128×1
Dense 128 → 8192, ReLU, reshape 8×8×128
GatedConv k=3, s=1, 8×8×128, ReLU
GatedDeconv k=3, s=2, 16×16×128, ReLU
GatedConv k=3, s=1, 16×16×128, ReLU
GatedConv k=3, s=1, 16×16×128, ReLU
GatedConv k=3, s=1, 16×16×128, ReLU
GatedDeconv k=3, s=2, 32×32×64, ReLU
GatedConv k=3, s=1, 32×32×64, ReLU
GatedConv k=3, s=1, 32×32×64, ReLU
GatedConv k=3, s=1, 32×32×64, ReLU
GatedDeconv k=3, s=2, 64×64×32, ReLU
GatedConv k=3, s=1, 64×64×32, ReLU
GatedConv k=3, s=1, 64×64×32, ReLU
GatedConv k=3, s=1, 64×64×32, ReLU
GatedDeconv k=3, s=2, 128×128×3, ReLU
GatedConv k=5, s=1, 128×128×3, ReLU

Fig. 13: Generator used for car generation from the Potsdam dataset [14]

objectness threshold of 0.75. Our classification task can be summarized as the detection of a car on images. This task seems very easy for our classifier as we do not consider the Intersection Over Union. The generator architecture we consider contains three convolutional layers and one dense layer (details in Figure 13). We set λ to 50. We consider two different methods, we first train a WGAN and we evaluate its ability to generate adversarial data. Then we used the WGAN generator as initialization for our method ARGAN, and evaluate the number of generated adversarial data.

Results. State of the art classifiers have high accuracy and only a few natural adversarial examples making it hard to train our methods. Nevertheless, using a pre-trained WGAN generator as generator for our method leads to a high number of generated adversarial data with good image quality. Example of natural adversarial car images can be found in Figure 14. We see that our generated adversarial images have a better quality than natural adversarial images from the ground truth or WGAN generated images. As seen in Table III the considered WGAN generates only 2.3% adversarial data while our methods improve its score by more than 5 points (more than 3 times more examples are adversarial), showing that even in extreme scenarios, our method is still relevant to generate adversarial data. Moreover Figure 14 shows that despite having few good looking natural adversarial examples in the ground truth, our method manages to generate better looking images and has a better success rate.

	Adv. generation rate	std
WGAN	97.4%	±0.94%
Our method	92.2%	±2.5%

TABLE III: Classification accuracy for our pre-trained classifier over generated data from different methods.

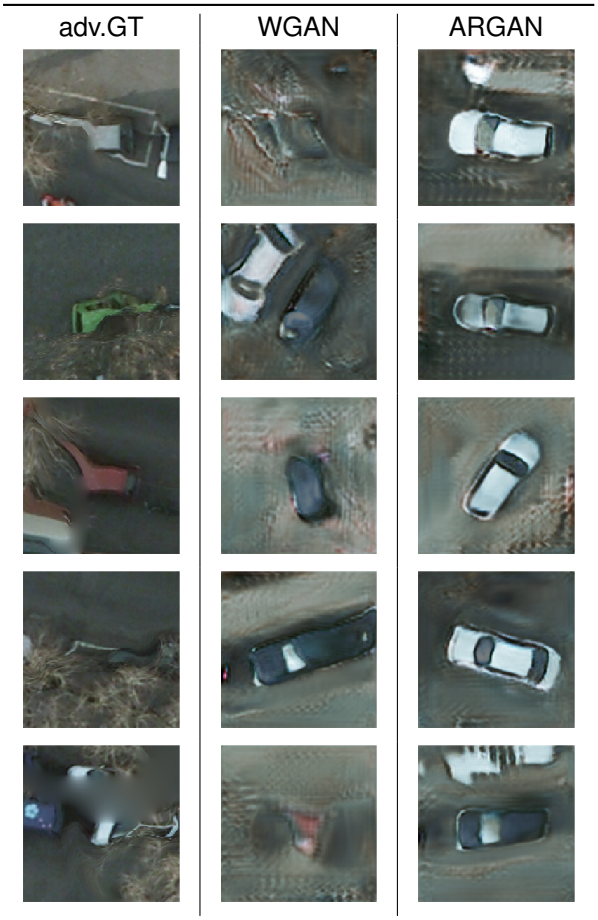


Fig. 14: Generated adversarial car images. The first column are natural adversarial examples in our GT, the second column are adversarial example generated with a classical WGAN, the last column are adversarial example generated with our method

V. Conclusion

This paper tackles the problem of generating natural adversarial images for remote sensing applications. Those images are natural in the sense that they can be considered as a realistic variation of the input image, while being at the same time adversarial *wrt.* a given black box predictor. We propose a novel method to generate such examples, by modifying the WGAN loss function with a re-weighted distribution of the training data based on a pre-trained classifier's prediction. We developed several different weighting strategies to specialize the generator to generate a specifically natural adversarial data. To the best of our knowledge, it is the first time that a work explores a modification of the data distribution for adversarial generative modelling purpose. We have applied our method for several generative modelling tasks such as adversarial hyperspectral generation, image modifications and adversarial image generations for a state-of-the-art classifier.

Future works will consider applying this method to different modalities such as point cloud or SAR data, to further assess the applicability of our method in real scenarios.

ACKNOWLEDGEMENTS

This work is partially funded through the projects OATMIL ANR-17-CE23-0012 and 3IA Côte d'Azur Investments ANR-19-P3IA-0002 of the French National Research Agency (ANR).

REFERENCES

[1] X. X. Zhu, D. Tuia, L. Mou, G.-S. Xia, L. Zhang, F. Xu, and F. Fraundorfer, "Deep learning in remote sensing: A comprehensive review and list of resources," *IEEE Geoscience and Remote Sensing Magazine*, vol. 5, no. 4, pp. 8–36, 2017.

[2] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations*, 2015.

[3] S. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: A simple and accurate method to fool deep neural networks," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2574–2582, 2016.

[4] A. Ortiz, O. Fuentes, D. Rosario, and C. Kiekintveld, "On the defense against adversarial examples beyond the visible spectrum," in *2018 IEEE Military Communications Conference*, pp. 1–5, 10 2018.

[5] L. Chen, G. Zhu, Q. Li, and H. Li, "Adversarial example in remote sensing image recognition," 2019.

[6] W. Czaja, N. Fendley, M. Pekala, C. Ratto, and I.-J. Wang, "Adversarial examples in remote sensing," in *ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pp. 408–411, 2018.

[7] A. Ortiz, A. Granados, O. Fuentes, C. Kiekintveld, D. Rosario, and Z. Bell, "Integrated learning and feature selection for deep neural networks in multispectral images," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1277–127709, June 2018.

[8] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner, "Detecting adversarial samples from artifacts," *arXiv preprint arXiv:1703.00410*, 2017.

[9] T. Pang, C. Du, Y. Dong, and J. Zhu, "Towards robust detection of adversarial examples," in *Advances in Neural Information Processing Systems*, pp. 4579–4589, 2018.

[10] I. Goodfellow *et al.*, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014.

[11] Z. Zhao, D. Dua, and S. Singh, "Generating natural adversarial examples," in *International Conference on Learning Representations*, 2018.

[12] Y. Song, R. Shu, N. Kushman, and S. Ermon, "Constructing unrestricted adversarial examples with generative models," in *Advances in Neural Information Processing Systems*, 2018.

[13] 2018 IEEE GRSS Data Fusion Contest. Online: http://www.grss-ieee.org/community/technical-committees/data-fusion.

[14] "Isprs potsdam 2d semantic labeling dataset." http://www2.isprs.org/commissions/comm3/wg4/2d-sem-label-potsdam.html. Accessed: 2010-09-30.

[15] T. B. Brown, N. Carlini, C. Zhang, C. Olsson, P. Christiano, and I. Goodfellow, "Unrestricted adversarial examples," *arXiv preprint arXiv:1809.08352*, 2018.

[16] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier GANs," in *Proceedings of the 34th International Conference on Machine Learning* (D. Precup and Y. W. Teh, eds.), vol. 70 of *Proceedings of Machine Learning Research*, (International Convention Centre, Sydney, Australia), pp. 2642–2651, PMLR, 06–11 Aug 2017.

[17] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 105–114, July 2017.

[18] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proceedings of the 34th ICML*, Proceedings of Machine Learning Research, PMLR, 2017.

[19] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *Advances in Neural Information Processing Systems*, 2017.

[20] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," in *International Conference on Learning Representations*, 2018.

[21] S. Nowozin, B. Cseke, and R. Tomioka, "f-gan: Training generative neural samplers using variational divergence minimization," in *Advances in Neural Information Processing Systems 29* (D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, eds.), pp. 271–279, Curran Associates, Inc., 2016.

[22] C.-L. Li, W.-C. Chang, Y. Cheng, Y. Yang, and B. Póczos, "Mmd gan: Towards deeper understanding of moment matching network," *arXiv preprint arXiv:1705.08584*, 2017.

[23] A. Genevay, G. Peyre, and M. Cuturi, "Learning generative models with sinkhorn divergences," in *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, 2018.

[24] G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro, "Image inpainting for irregular holes using partial convolutions," 2018.

[25] Y. Jo and J. Park, "Sc-fegan: Face editing generative adversarial network with user's sketch and color," *arXiv preprint arXiv:1902.06838*, 2019.

[26] T. Park, M.-Y. Liu, T.-C. Wang, and J.-Y. Zhu, "Semantic image synthesis with spatially-adaptive normalization," 2019.

[27] N. Audebert, B. Le Saux, and S. Lefèvre, "Generative adversarial networks for realistic synthesis of hyperspectral samples," in *2018 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, July 2018.

[28] K. Jiang, Z. Wang, P. Yi, G. Wang, T. Lu, and J. Jiang, "Edge-enhanced gan for remote sensing image superresolution," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, pp. 5799–5812, Aug 2019.

[29] D. Lin, K. Fu, Y. Wang, G. Xu, and X. Sun, "Marta gans: Unsupervised representation learning for remote sensing image classification," *IEEE Geoscience and Remote Sensing Letters*, vol. 14, pp. 2092–2096, Nov 2017.

[30] Y. Duan, X. Tao, M. Xu, C. Han, and J. Lu, "Gan-nl: Unsupervised representation learning for remote sensing image classification," in *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 375–379, Nov 2018.

[31] C. Voreiter, J.-C. Burnel, P. Lassalle, M. Spigai, R. Hugues, and N. Courty, "A cycle gan approach for heterogeneous domain adaptation in land use classification," 2020.

[32] G. Peyré and M. Cuturi, "Computational optimal transport," *Foundations and Trends® in Machine Learning*, 2019.

[33] H. Petzka, A. Fischer, and D. Lukovnikov, "On the regularization of wasserstein GANs," in *International Conference on Learning Representations*, 2018.

[34] R. Shu, H. Bui, and S. Ermon, "Ac-gan learns a biased distribution," *NIPS Bayesian Deep Learning workshop*, 2017.

[35] T. Salimans, H. Zhang, A. Radford, and D. Metaxas, "Improving GANs using optimal transport," in *International Conference on Learning Representations*, 2018.

[36] K. Fatras, Y. Zine, R. Flamary, R. Gribonval, and N. Courty, "Learning with minibatch wasserstein: asymptotic and gradient properties," in *AISTATS*, 2020 (To appear).

[37] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012.

[38] W. Hu, Y. Huang, L. Wei, F. Zhang, and H. Li, "Deep convolutional neural networks for hyperspectral image classification," *Journal of Sensors*, vol. 2015, 2015.

[39] N. Audebert, B. Le Saux, and S. Lefevre, "Deep learning for classification of hyperspectral data: A comparative review," *IEEE Geoscience and Remote Sensing Magazine*, vol. 7, no. 2, 2019.

[40] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[41] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, p. 234–241, 2015.

[42] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," in *International Conference on Learning Representations*, 2016.

[43] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," 2018.