Satellite Imagery Noising with Generative Adversarial Networks

Mohamed Akram Zaytar
Ph.D. Candidate
Faculty of Sciences and Technology, Tangier
Email: MedAkramZaytar@gmail.com

Chaker El Amrani
Associate Professor
Faculty of Sciences and Technology, Tangier
Email: celamrani@gmail.com

***Abstract*—Using Satellite Imagery for Supervised Learning Problems can be quite challenging when we don't have complete information due to natural phenomena (Clouds, Fog, Haze, ..). Solving this problem will improve the way we annotate remote sensing data and make use of it in a world where satellite imagery can be a great resource to have in your big data pipeline. In this Paper, we present a Generative Adversarial Network (GANs) Model that can generate natural atmospheric noise that serves as input to Supervised Machine Learning Algorithms.**

I. Introduction

Satellite Imagery and Remote Sensing Data are the cornerstones for modern Environmental Monitoring Systems, for both rule-based and AI-powered systems, we rely more and more on High-Resolution Imagery in domains such as Agriculture, Forestry, Disaster Management, Geology and many more.

In recent years, however, a lot of applications of deep learning on remote sensing data emerged, new state-of-the-art results were published in far-apart domains such as building footprints [1], land use classification [2], iceberg detection [3], deforestation [4], Weather Forecasting [5], and More. This surprising success is linked to the massive amounts of data collected from satellites every day, and, in many occasions, remote sensing imagery data are not RGB-based, but can include up to dozens of visual bands, allowing for rich data mining using deep neural networks.

A common issue when pre-processing Satellite Imagery data to train an agent is the lack of Input (or data features), we only have real (ground-truth) images and we're responsible for creating Input data features to learn a certain pattern, a Prime example is when we want to create an Image-to-Image mapping where the Input image has some missing pixels and the output image is complete, in many more scenarios we are interested in introducing noise to satellite images so that the Neural Network is trained for real-world cases. In Short, pre-processed Input data should simulate the incompleteness of the image data we receive.

We propose a solution based on Generative Adversarial Networks (or GANs). We trained a Model that can learn new vector representations of the data and the underlying noise data structure from pre-processed 50x50 pixel patches containing missing/damaged pixels represented as zeros, and the healthy pixels as ones, After training, the Generator produces samples that are indistinguishable from the real data distribution of satellite noise images.

The Proposed Generative Adversarial Networks is comprised of the principal elements described below.

A. Real Data

A Collected and Pre-processed data set of 1,000,000 50x50 images with natural noise associated with natural features that depend on variables such as atmospheric conditions, weather, sensor quality, satellite position. We receive either a valid pixel measurement represented by 1, or an invalid, missing, or damaged pixel, represented by 0, collected directly from MDEOs [6] Data Pipeline.

B. Random Input

Uniformly Random vectors fed to the generator, sometimes called input entropy, represents the Input to the Generator Network. In Our case, these random vectors are comprised of 100 uniform random values $\in [0, 1]$.

C. The Generator

Responsible for producing vector representations of 50x50 pixel patches with noise ranging from 40% to 60% zero pixels. The Generator is trained using the feedback loop coming from the Discriminator's decisions, in other words, The error is back-propagated using predictions coming from the Discriminator.

D. The Discriminator

A Feed-forward Neural Network Responsible for deciding whether a satellite patch is real or not. simply put, It's a binary classifier, Trained on both real patches coming from the pre-processed data distribution and on fake image patches produced by the Generator.

E. Training Loop

Defines other pre-training functions and hyper-parameters such as the Loss function for the generator and discriminator networks, the Optimizers, the Batch Size, the Number of epochs, parameters that control the training balance between $D(.)$ and $G(.)$, and the actual training loop.

II. Problem

To Properly state the Problem, we define the following entities:

- x: Represents Real or Fake (Generated) Data, a single data point is a matrix of shape 50x50 pixels, each pixel holding a numerical value between 0 and 1 (after normalization).
- z: The Input Noise to the Generator, A Vector of 100 random values between 0 and 1.
- $G(z, \theta_g)$: The Generator Function, or the Generator Neural Network.
- θg: The parameters, or the weights of the Generator Network G.
- $D(x, \theta_d)$: The Discriminator Function, or the Discriminator Neural Network.
- θd: The parameters, or the weights of the Discriminator Network D.

To optimize for a smooth loss function, ground-truth pixel values were transformed so that values between 0.0 and 0.3 represent a missing pixel (instead of simply 0), and values between 0.7 and 1.0 represent an available pixel measurement (instead of 1).

$D(X)$ outputs the probability that X came from the real data distribution (pre-processed satellite patches) rather than p_z (The Generator's learned distribution). We train the Discriminator to maximize the probability of correctly assigning the true label to x and the false label to samples from p_z. We also simultaneously train the Generator to Minimize $log(1 - D(G(z)))$.

$D(.)$ and $G(.)$ play the following two-player minimax game with value function $V(G, D)$ [7]:

$$\min_G \max_D V = \mathbb{E}_{x \sim p_d(x)}[log(D(x))] + \mathbb{E}_{z \sim p_z(z)}[log(1 - D(G(z)))]$$

In Other words, the problem of generating a new noised satellite image is equivalent to the problem of generating a new vector (flattened image) following the "Remote Sensing Noise Probability Distribution" over the 2,500 (50x50) elements of the Vector.

We are solving, actually, the problem of generating a random variable with respect to a specific probability distribution.

III. Solution

We present the following Architectures for the Generator and Discriminator Neural Networks:

A. The Generator

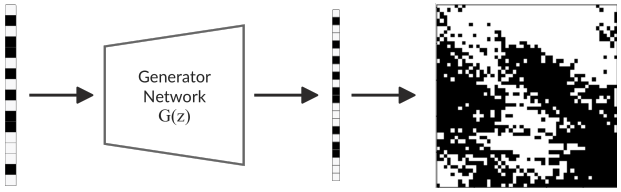

Fig. 1. The Generator learns the distribution of real noise, Output vector is reshaped to a matrix to produce the noise mask.

The Generator is modeled as a feed-forward neural network that takes as input a uniform random variable and returns a random variable that follows the target distribution (real noised images):

- Input: Tensors of Shape $(batch\_size, 100)$, consisting of Initialized Random Values between 0 and 1.
- Output: Tensors of Shape $(batch\_size, 1, 50, 50)$, 1-channel 50×50 generated noise images.
- 1st (Input) Layer: 256 output features.
- 2nd Layer: 512 output features.
- 3rd Layer: 1024 output feature.
- 4th Layer: 2048 output feature.
- 5th (Output) Layer: 2500 (50x50) output features.
- LeakyReLU [8] ($f_\alpha(x) = \max(0, x) - \alpha \max(0, -x)$) : used as an activation function for all layer except the output layer.
- Sigmoid ($S(x) = \frac{1}{1+e^{-x}}$) : used as an activation function for the output layer (since we want values between 0 and 1).
- 1-D Batch Normalization [9] : used for the middle 3 layers to improve the performance and stability of the generator network.

The Network reshapes the final output vector to an image of 50x50 pixels and 1-channel.

B. The Discriminator

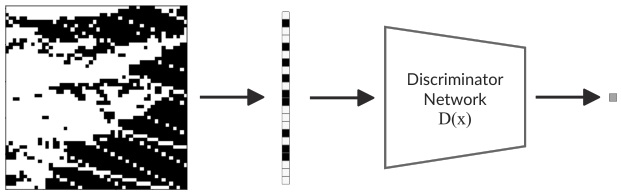

Fig. 2. Example: Takes in an Image and outputs a single value that represents the Probability that the Image is Real.

The Network first flattens Images into vectors of size $2,500$ before feeding them into the first layer:

- Input: Tensors of Shape $(batch\_size, 1, 50, 50)$, consisting of real/fake image patches 0 and 1.

- Output: Tensors of Shape $(batch\_size, 1)$, represents the probabilities that each image patch is real.
- 1st (Input) Layer: 512 output features.
- 2nd Layer: 256 output features.
- 3rd (Output) Layer: 1 output feature (a Probability).
- LeakyReLU : used as an activation function for the first two layers.
- Sigmoid : used as an activation function for the last output layer (to produce a probability).
- 1-D Batch Normalization: used for the middle 2 layers to improve the performance and stability of the discriminator network.

C. Training

1) Optimizers and Loss Function: For the networks' Loss functions, we chose the Binary Cross Entropy function, which measures the distance between the predictions vector and the target labels vector, the loss can be described as:

$$\ell(x, y) = \{l_1, .., l_N\}^\top, l_n = -[y_n \cdot \log x_n + (1 - y_n) \cdot \log(1 - x_n)]$$

Where N is the Batch Size.

As for the Optimizers, we chose the Adam [10] Optimizer, Adaptive Moment Estimation (Adam) computes adaptive learning rates for each parameter. It stores an exponentially decaying average of past squared gradients v_t and an exponentially decaying average of past gradients m_t, where:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

Adam fights zero-related biases in the parameters by computing bias-corrected 1st and 2nd moment estimates, $\hat{m_t}$ and $\hat{v_t}$ [11], the Optimization Algorithm is as follows:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v_t}} + \epsilon}$$

IV. Results

Training was conducted on a Cloud GPU instance with the following Specifications:

- GPUs: A Cloud based Tesla P100 Environment.
- CPUs: 4.
- Memory: 24GB RAM.
- Data Storage: 250 GB SSD.

Training specifications are outlined below:

- Batch Size: 256 images per batch.
- Epochs : 100 epochs for the entirety of the loop.
- η : The Learning Rate of the Optimizer is 0.0002.
- $\beta_1 = 0.5$ and $\beta_2 = 0.999$.

A Training Balance between the Generator and Discriminator was reached by continuously training the Discriminator while training the Generator in Intervals controlled by two parameters: the 'step' and 'n_critic'.

The following algorithm showcases a high-level overview of the training loop:

Algorithm 1: Training Loop

Data: $G(., \theta_{g0})$, $D(., \theta_{d0})$, dataLoader, epochs, gOptimizer, dOptimizer, loss, S [Entropy]
Result: $G(., \theta_g)$, $D(., \theta_d)$
begin
  for $e \in epochs$ **do**
    for $batch \in dataLoader$ **do**
      /* calculate discriminator losses */
      $dLossReal \longleftarrow loss(D(batch), 1s)$
      $dLossFake \longleftarrow loss(D(G(S)), 0s)$
      $dLoss \longleftarrow dLossFake + dLossReal$

      /* Optimize with Back-Propagation */
      $dLoss.backward()$
      $dOptimizer.step()$

      /* Control generator training */
      if *step % n_critic = 0* **then**
        $dDecisions \longleftarrow D(G(S))$
        $gLoss \longleftarrow loss(dDecisions, 1s)$

        /* Optimize with Back-Propagation */
        $gLoss.backward()$
        $gOptimizer.step()$
      $step \longleftarrow step + 1$

After training, we minimized the loss of both the discriminator and generator networks, and the discriminator couldn't distinguish between real noise and generated remote sensing noise.

We improved on the model's inception scores [12] to stabilize the performance of our unconditional GANs, the Kullback-Leibler formula was used to calculate the score, KL divergence measures how similar and different two probability distributions are, to calculate the inception score, we average the exponential KL divergence score over all images:

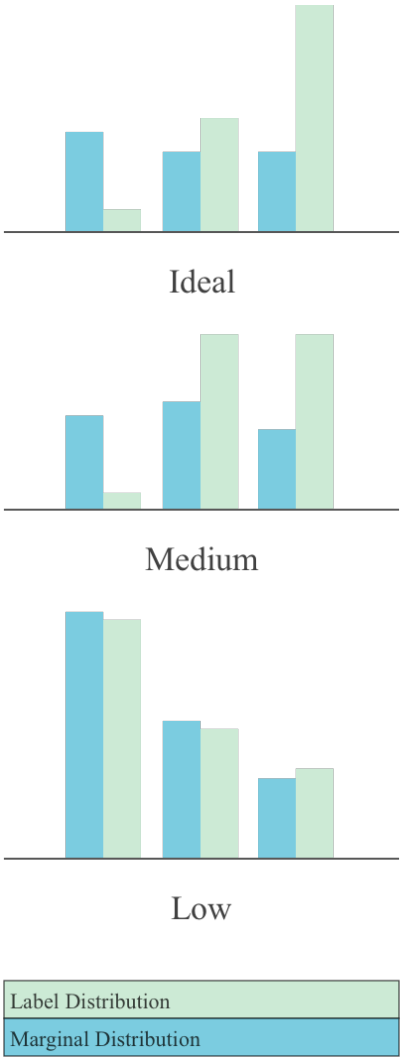

Fig. 3. Example: KL divergence relation to our distributions.

Additionally, we used another pre-trained neural network to measure similarities between the generated sample distribution and real samples coming from the pre-processed satellite imagery.

the following table showcases the final loss values and the accuracy of the discriminator after training:

TABLE I. LOSS AND DISCRIMINATOR ACCURACY

Metric	Score
Generator Loss	0.53
Discriminator Loss	0.77
Discriminator Accuracy	56%

After training the generator to produce 50x50 patches of noise, we use a simple function to mask healthy patches with zeros and produce naturally noised satellite imagery that can be used for several supervised learning problems such as 2-D Interpolation and super-resolution.

V. FURTHER WORK

This model can be considered the first step towards a highly sophisticated noise generator for remote sensing imagery, improvements to the model include specifying the amount of noise (this generator generates 0.4 to 0.6 noise) and generating images of dynamic shapes.

We can achieve parametric noising by using a Conditional Generative Adversarial Network (CGANs) [13], in which we add a conditioning layer as an input to both the generator and discriminator to produce noise in specific percentages, we will change the following terms in the optimization minimax problem:

$$log(D(x)) \Rightarrow log(D(x|y))$$

$$log(1 - D(G(z))) \Rightarrow log(1 - D(G(z|y)))$$

Another interesting approach is to learn representations between noise parameters and the distribution of the noise using InfoGAN [14] by adding more channels to the image containing information about measurements conditions, this can lead to minimizing the ratio of damaged pixels in the future.

A parametric generator will need more data and hyper-parameter tuning to solve the issue of instability and non-convergence of the Loss Function in GANs, this can be addressed in the future.

VI. CONCLUSION

Deep Learning opens the door to limitless applications in domains such as Environmental Science, Agriculture, Pollution and Disaster Monitoring, and many others [15]. Remote sensing data providers [16] also play a central role in the development of the field and in advancing environmental Research using Machine Learning and Data Science, Without Big Data Resources, deep learning methods fall short. Fortunately, we have a wealth of satellite imagery and remote sensing data to train and make use of our models.

We believe that artificial intelligence will revolutionize climate science and lay the ground to build effective solutions that address the environmental problems we are facing today. By using high-resolution imagery, remote sensing data, and satellite sensor-based imagery in general, we can gain a deeper understanding of the atmospheric processes and build systems that contribute in Domains like Energy, Pollution, Agriculture, Oceans, and Climate Science.

ACKNOWLEDGMENTS

The authors are thankful to the Ministry of Higher Education and Scientific Research, and the National Centre for Scientific and Technical Research (CNRST) for funding this study, under the project: PPR/2015/7.

REFERENCES

[1] J. Yuan, "Automatic building extraction in aerial scenes using convolutional networks," *arXiv preprint arXiv:1602.06564*, 2016.

[2] M. Castelluccio, G. Poggi, C. Sansone, and L. Verdoliva, "Land use classification in remote sensing images by convolutional neural networks," *arXiv preprint arXiv:1508.00092*, 2015.

[3] H. Zhao, A. Lall, M. Ogihara, and J. Xu, "Global iceberg detection over distributed data streams," 2010.

[4] J.-F. Mas, H. Puig, J. L. Palacio, and A. Sosa-López, "Modelling deforestation using gis and artificial neural networks," *Environmental Modelling & Software*, vol. 19, no. 5, pp. 461–471, 2004.

[5] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *Advances in neural information processing systems*, 2015, pp. 802–810.

[6] C. El Amrani, G. L. Rochon, T. El-Ghazawi, G. Altay, and T. Rachidi, "System architecture of the mediterranean dialogue earth observatory," in *Geoscience and Remote Sensing Symposium (IGARSS), 2013 IEEE International*. IEEE, 2013, pp. 600–603.

[7] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[8] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[9] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[10] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[11] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[12] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in *Advances in neural information processing systems*, 2016, pp. 2234–2242.

[13] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[14] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," in *Advances in neural information processing systems*, 2016, pp. 2172–2180.

[15] L. Deng, "A tutorial survey of architectures, algorithms, and applications for deep learning," *APSIPA Transactions on Signal and Information Processing*, vol. 3, 2014.

[16] K. D. Klaes, M. Cohen, Y. Buhler, P. Schlüssel, R. Munro, J.-P. Luntama, A. von Engeln, E. Ó. Clérigh, H. Bonekamp, J. Ackermann *et al.*, "An introduction to the eumetsat polar system," *Bulletin of the American Meteorological Society*, vol. 88, no. 7, pp. 1085–1096, 2007.