EC5

Scientific Contribution on synthetic data generation using diffusion processes

Document reference number for ANR

contact@confiance.ai | www.confiance.ai

Document reference: XXX

Contributors

	Name	Organisation	Role
Responsible for the deliverable	Louis ROUSSEL	IRT St Exupéry	AI engineer
Scientific responsible	Amélie BOSCA	Sopra Steria	Data Scientist
Co-authors	Jean-Philippe BOYER	Thalès	AI engineer

Document control

Revision	Date	Commentary	Author
1.0	01/09/2023	Revue: OK	Jean-Philippe BOYER
1.1	22/12/2023	Add experiments with YoloV8 Revue: OK	Jean-Philippe BOYER Amélie BOSCA

A. Introduction and abstract

A.1 General introduction to trustworthy AI challenges

Trustworthiness in AI within critical systems (systems that can directly or indirectly affect human life and moral entities) is essential for its widespread adoption (by the industry, the decision makers, the general public, etc.) and poses the following significant challenges.

- First, how to design AI models, so that, by construction, they satisfy trustworthy properties (accuracy, robustness…).
- Secondly, how to characterize these AI models, for example to understand and explain their behavior and their adequacy to the operational domain.
- Then, how to implement and embed those AI models on hardware, by making them fit for the target without losing their trustworthy properties.
- Another question is, what methods of data engineering to apply in order to, among other topics, manage important volumes of data and adapt to the evolution of the operational domain.
- At system level, what verification and certification processes to consider specifically for AI-based systems.
- Finally, a federation of all these matters is necessary to build an end-to-end methodological approach, supported by a consistent engineering environment compatible with industrial practices.

These are the challenges, among others, that the Confiance.ai program addresses.

A.2 Problem Introduction and Objectives

This document aims to tackle the challenge of training an objects detector such as a YOLOv5 when dealing with limited data. The objective is to offer a solution to improve performances of objects detectors on real dataset's rare classes using synthetic data. This technical report is destined to be read by Machine Learning Engineers or Researchers facing or interested by such issues.

Term	Definition
Canny Edges	Refers to the boundaries in an image. The Canny Edge detection algorithm aims to highlight significant changes in pixel intensity to identify edges in a given image
Limited Data	In this document when we talk about limited data we talk about a limited number of data (specifically at training time).

Glossary : Definition of Specific terms used in this document

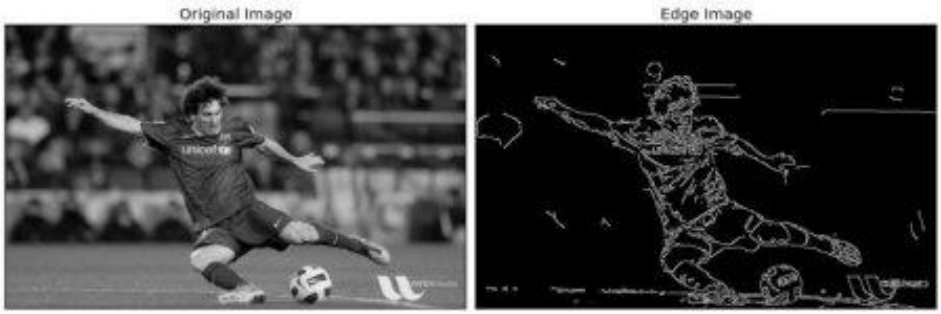

Figure 1 : Canny Edges Detection illustration. From openCV documentation

In real world applications, datasets aren't always balanced and the different classes don't have an almost unlimited amount of instances. The scarcity of certain classes usually leads a detector to underperform on these specific classes. This document offers a methodology to train a diffusion model on limited data and the possibility to then InPaint new instances of a given target class on real images. The resulting synthetic data created could then be used for the training and/or the validation (depending on the aimed use case) of an objects detector.

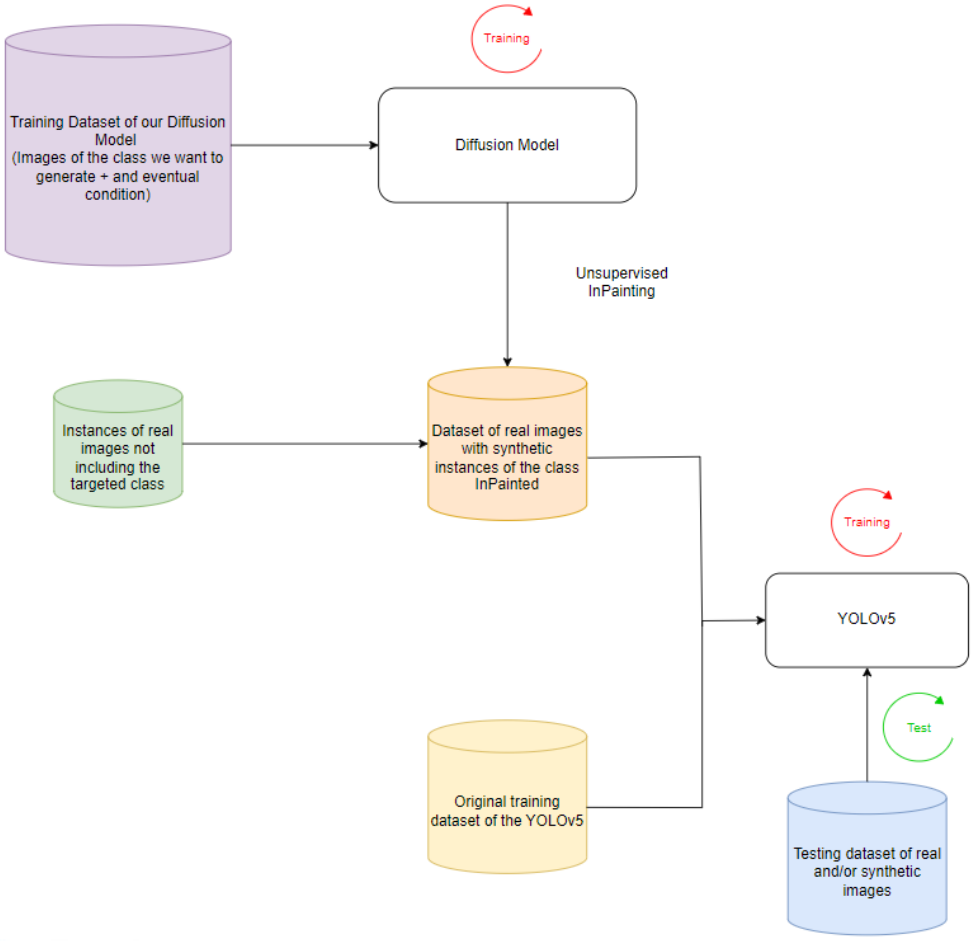

Figure 2: High Level view of our method (here applied for the training of a yolo with synthetic data and its validation)

Our method consists in two main steps. In the first one a diffusion model is trained on a dataset of images containing instances of the class we wish to synthesize. The model can eventually be conditioned, for example by a segmentation map or by canny edges to guide the generation process. This condition can be useful at inference time to have more control on the instances we generate. Once the model trained we use an unsupervised method of InPainting to add synthetic instances of the given class on real images of the training set of the YOLOv5. The code related to this method is available on Gitlab (2 branches, the master one is fit to a specific use case, the « clean » one is more generic).

In a first part, we will explain how diffusion models work. Subsequently, an exploration of a method to train a diffusion model on limited data will be made. Following that, the InPainting method will be reviewed, and ultimately, the maturity of the method and its limitations will be discussed.

B. Diffusion Models State of The Art

The concept of diffusion models is not new, however, the seminal work of Jonathan Ho et al. from 2020[1] brought it to the forefront of the Machine Learning community and sparked a renewed interest in their applications. In this paper the Denoising Diffusion Probabilistic Models (DDPMs), a class of latent variable model inspired by non-equilibrium thermodynamic, are introduced. DDPMs are based on a forward (noise injection) and backward (denoising) process defining a Markov chain.

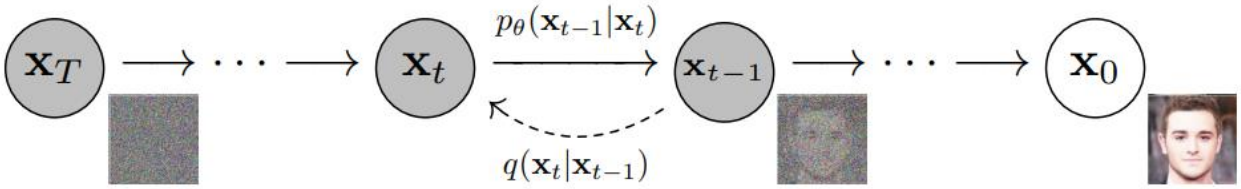

Figure 3: from [1]. Illustration of the considered process

In term of FID, a widely utilized metric for evaluating images generation models, the results obtained in this paper are comparable to the top-performing generative models prevailing at the time of its publication.

Figure 4: From [2]. Samples from a DDPM trained on CelebA-HQ 256

In 2021, some modifications are proposed by Nichol and Dhariwal [2] with the aim of improving the Log Likelihood of these models. Among these improvements, one can find adjustments to the Neural Network employed, a shift from a linear to a cosine noise schedule, and the addition of a learned variance for

$p_\theta(x_{t-1}|x_t)$ alongside the mean. Later, still in 2021, Nichol and Dhariwal [3] declare the end of the GANs and the rising of diffusion models with their paper 'Diffusion Models beat GANs on image synthesis'. In this publication, some improvements are once again formulated (specifically a modification of the neural network) allowing the diffusion models to be State of the Art in term of images generation. In this paper, Classifier Guidance is also introduced. Classifier Guidance refers to the process of guiding the denoising process with the aid of a classifier trained to classify an image at the different noise level of our process, It aims to improve the quality of the generated images at the expense of their diversity (trade off quality/diversity). Classifier Guidance thereby allows the generation of images from a specific class with an unconditionally trained model.

Figure 5: From [3]. Samples from an unconditionned diffusion model with guidance on the "Pembroke Welsh Corgi" class

In 2022, the Classifier Free Guidance is introduced [4], enabling a trade off between quality and diversity without relying on a pre-trained classifier. The diffusion model is conditioned on the classes of the various images in the dataset, during training a « null » class is introduced and used with a probability p. This null class translates to unconditional generation.

Based on these successes, numerous improvements have been brought to the diffusion models ; Jonathan Ho, Saharia et al. [5] offered a method to obtain high quality images through cascaded diffusion (Diffusion for generation in low resolution followed by Super Resolution Diffusion). Kingma et al. [6] introduced Variational Diffusion Models with the aim of improving further more the Likelihood (in this paper the score matching paradigm is used, to learn more about score matching refer to [21], it also implies working with continuous time steps). Jiaming et al. [7] used Denoising Diffusion Implicit Models to reduce the number of time steps needed for infering with a DDPM while also offerring the possibility to use a deterministic backward process.

Diffusion Models have proven beneficial for a multitude of Computer Vision tasks. Palette [8] is a Diffusion Model-based framework for Image-to-Image tasks, while RePaint [9] offers the possibility to do some InPainting with a Diffusion Model trained for general images generation (unsupervised InPainting). In 2022, several Diffusion-based Text-to-Image models gained significant attention, including Dall-e 2 [10], Stable Diffusion [11] and Imagen [12]. Notably, Stable Diffusion is OpenSource

Figure 6: from RePaint paper [9]. InPainting with models trained for images generation

The substantial computational complexity of diffusion models remains a hindrance to their widespread adoption. Indeed, at inference time as many forward passes in the network are necessary as the number of time steps in the forward/backward process. The original DDPM paper used 1000 time steps at inference time. Numerous works tried to tackle this computational complexity issue, with current efforts enabling as low as 20 time steps during inference.

C. Proposed method for synthetic data generation in limited data context

C.1 How Diffusion Models works?

In this section, we briefly describe how diffusion models work. For a more detailed explanation, please refer to the work of Ho et al. [1] and Sohl-Dickstein et al. [14]. This short description has been taken from the work of Prafulla Dhariwal and Alex Nichol [3].

On a high level, diffusion models sample from a distribution by reversing a gradual noising process. In particular, sampling starts with noise x_T and produces gradually less-noisy samples x_{T-1}, x_{T-2}, ... until reaching a final sample x_0. Each timestep t corresponds to a certain noise level, and x_t can be thought of as a mixture of a signal x_0 with some noise ε where the signal to noise ratio is determined by the time step t. We assume that the noise ε is drawn from a diagonal Gaussian distribution, which works well for natural images and simplifies various derivations. A diffusion model learns to produce a slightly more "denoised" x_{t-1} from x_t. Ho et al. [1] parameterize this model as a function $\varepsilon_\theta(x_t, t)$, which predicts the noise component of a noisy sample x_t. To train these models, each sample in a minibatch is produced by randomly drawing a data sample x_0, a timestep t, and noise ε , which together give rise to a noised sample x_t with the following equation :

$q(x_t|x_0) = N(x_t; \sqrt{\overline{\alpha}_t}x_0, (1-\overline{\alpha}_t)I) = \sqrt{\overline{\alpha}_t}x_0 + \sqrt{(1-\overline{\alpha}_t)}\varepsilon, \varepsilon \sim N(0, I)$ with $\overline{\alpha}_t \coloneqq \prod_{s=0}^{t} \alpha_s$ and $\alpha_t = 1 - \beta_t$

β_t being the variance schedule.

The training objective is then $||\varepsilon - \epsilon_\theta(x_t, t)||^2$, i.e. a simple mean-squared error loss between the true noise and the predicted noise.

It is not immediately obvious how to sample from a noise predictor $\epsilon_\theta(x_t, t)$. Recall that diffusion sampling proceeds by repeatedly predicting x_{t-1} from x_t, starting from x_T . Ho et al. show that, under reasonable assumptions, we can model the distribution $p_\theta(x_{t-1}|x_t)$ of x_{t-1} given x_t as a diagonal Gaussian $N(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$, where the mean $\mu_\theta(x_t, t)$ can be calculated as a function of $\varepsilon_\theta(x_t, t)$. The variance $\Sigma_\theta(x_t, t)$ of this Gaussian distribution can be fixed to a known constant or learned with a separate neural network head, and both approaches yield high-quality samples when the total number of diffusion steps T is large enough. Ho et al. observe that the simple mean-squared error objective, L_{simple}, works better in practice than the actual variational lower bound L_{vlb} that can be derived from interpreting the denoising diffusion model as a VAE. They also note that training with this objective using their corresponding sampling procedure is equivalent to the denoising score matching model from Song and Ermon [15], who use Langevin dynamics to sample from a denoising model trained with multiple noise levels to produce high quality image samples. We often use "diffusion models" as shorthand to refer to both classes of models. Alex Nichol and Prafulla Dhariwal [2] showed that learning $\Sigma_\theta(x_t, t)$ with L_{vlb} improved the log-likelihoods of

the models and offered to add a weighted L_{vlb} along L_{simple}, $L = L_{simple} + \lambda L_{vlb}$. For more details concerning the loss and its explicit expression, please refer to the annexes at the end of this document.

C.2 The limited data context

Our method aims to provide a solution for enhancing the performance of an objects detector on a specific class using synthetic data. The original training dataset of the YOLO is assumed imbalanced against this class. In this context, we also assume that the diffusion model is trained with a limited number of examples of this specific class. Diffusion models are usually very data consuming, for reference billions of text-image pairs are used for the training of Stable Diffusion. Taehong Moon et al. [16] offers a method to fine-tune unconditional Diffusion Models on limited data. In this paper, the authors freeze the residual blocks of the U-Net (used as the denoiser model) and fine-tune the Attention Blocks.

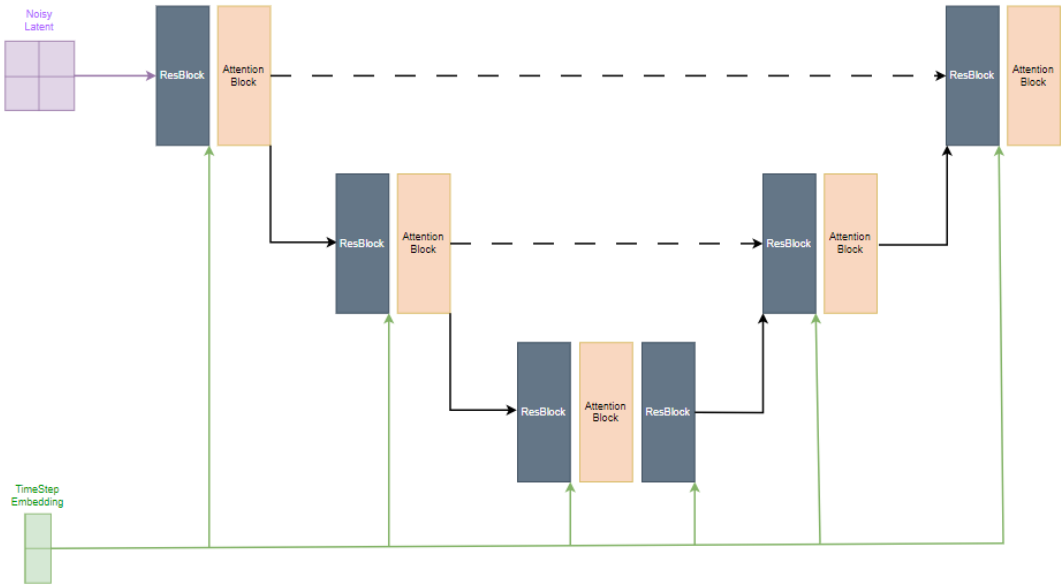

Figure 7 High Level view of the original U-Net architecture used in Diffusion Models

Furthermore, inspired by the adapter-based methods for Language Model, a Time-Fusion module and a Time-Scaling module are added to the Attention blocks. The mixture of these two additional modules creates a 'Time-Aware Adapter' that improves the adaptation of the network to the target data. These modules possess time-awareness, incorporating the time step's embedding into their inputs, alongside the feature tensor z_{in} for the time-fusion module. Ultimately, a weight-sharing technique is employed: the identical Time-

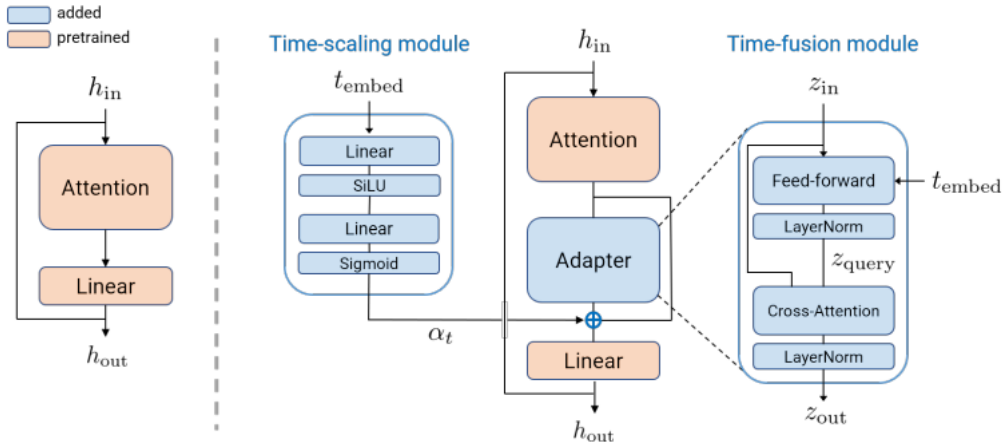

Figure 8: Schematic for the time-aware adapter incorporated in the different attention blocks of the U-Net. Left : Original Attention Block, Right : Attention block modified with the adapter. Scheme taken from [16]

fusion submodule is utilized across multiple attention blocks to minimize parameters overhead. In their publication, the scale of the dataset used for fine-tuning is of approximately 800 to 1000 training images.

Our diffusion model is trained using these adaptations. To add more control at inference time it is also possible to condition the diffusion model on an image or a label, for instance one could use a segmentation map to control the outer shape of the instance generated.

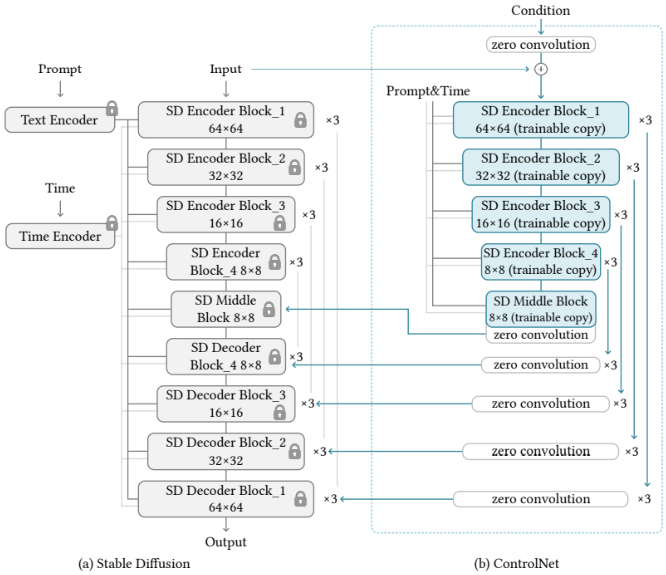

Figure 9 : Illustration of the ControlNet hypernetwork (image taken from [18]). The original Unet is frozen, a copy of the encoder is trained and the different feature maps are summed to the encoder of the frozen model

To do so, inspired by ControlNet [18] we found that even while freezing everything but the Attention Block, we could still manage to fine-tune our unconditional model in a conditional fashion and without hyper-network (as opposed to ControlNet). We thus show the possibility to fine tune an unconditional model for conditional generation with limited data. To do so a light condition encoder is added at the beginning of the network.

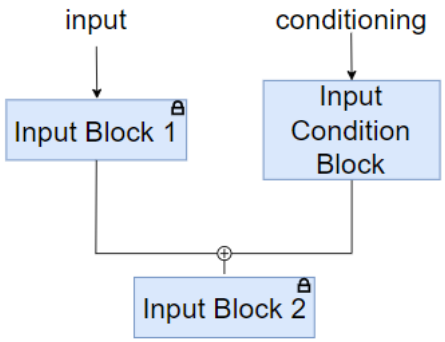

Figure 10:The conditioning image pass through a first block before being added to the first feature map of our network

C.3 Unsupervised Inpainting

Once our diffusion model trained we now wish to InPaint synthetic data on real images. To do so an unsupervised InPainting method is adopted. We choose to retain the RePaint method [9]. In RePaint, Andreas Lugmayr et al. offer a method to perform free form InPainting with DDPM in an unsupervised manner. The idea is, at every diffusion timestep, to use an addition of the known pixels and the inferred unknown pixels to form the final image.

$x_{t-1} = m \odot x_{t-1}^{known} + (1-m) \odot x_{t-1}^{unknown}$ with m being our mask, x_{t-1}^{known} is sampled from q using the known pixel and $x_{t-1}^{unknown}$ is sampled from p_θ using our model.

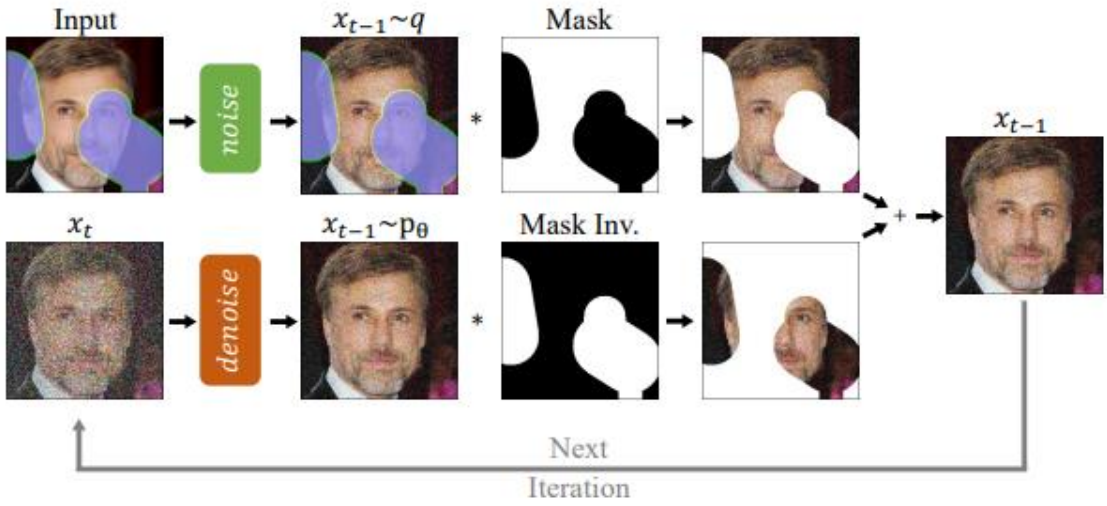

Figure 11: Figure taken from [9]. Overview of the RePaint approach

For more details concerning this method, please refer to [9].

This method offers little control on the inpainted object but the conditioning of the diffusion model can play a role here and one could decide for example to train the diffusion model conditioning it with a segmentation map or with canny edges to enforce a certain shape for the synthetic instance InPainted.

D. Experiments: the XView Dataset (Thalès Satellite Imagery use case)

D.1 Objectives and Available data

The objective of this study is to tackle the challenge of training an object detector like a YOLOv5 when dealing with limited data. Keeping this issue in mind, our approach involves training a diffusion model to generate a particular target class. We aim to enhance the performance of YOLOv5 specifically on this class by augmenting its training set with synthetic data. Our use case is a part of the XVIEW dataset composed of satellite images with various vehicules (truck, cars, planes, boats), our goal is to improve the performance of the YOLOv5 detector on the plane class. The training dataset of our diffusion model is composed of tiles from the XVIEW dataset aswell as a subset of images from the DOTA dataset. More specifically our training dataset consists of 1033 images (811 from DOTA and 222 from XVIEW) and our validation dataset consists of 64 images (45 from DOTA and 19 from XVIEW). A typical image in our dataset contains a centered plane surrounded by airport background (nature, tarmac, passenger gateway, cars, buses etc …). All our images are cropped to a 256x256 pixels resolution (ground resolution of 30cm per pixel).

Figure 12 : Image from our training set

Our objective is to InPaint realistic airplanes on tiles of the XVIEW dataset that dont contains any airplane. These new airplane instances will then be used in the training set of a YOLOv5 detector. Multiple training configurations will be tested :

- Replacement of real airplanes in the training set by synthetic ones to assess the quality of the generated planes
- Add synthetic airplanes in the dataset of real airplanes to augment the number of plane instances in the training set

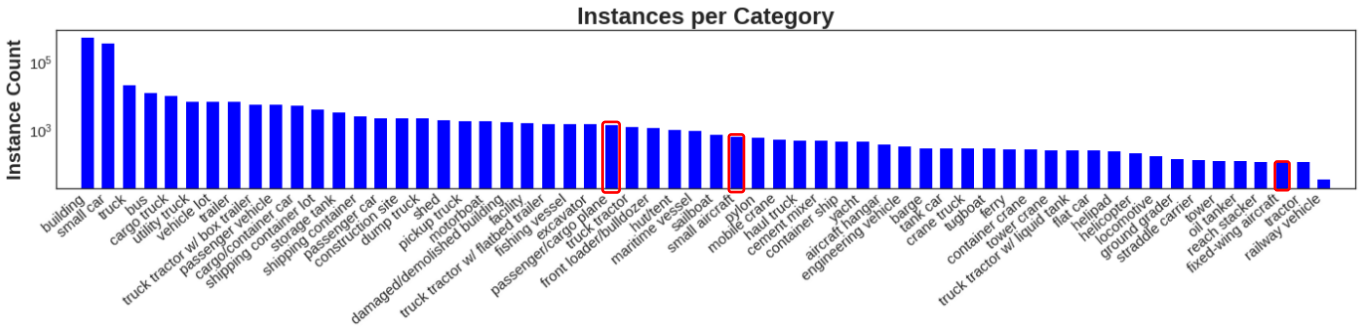

Figure 13: Distribution of the classes in the complete Xview Dataset

In the application on this use case, two diffusion models have been trained, the first one is trained unconditionnaly while the second has been trained with canny edges to guide the generation.

Figure 14: Unconditional Generation (first method)

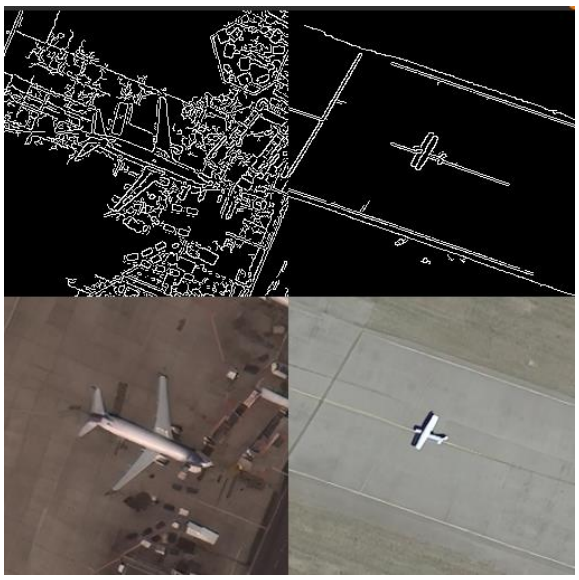

Figure 15: - Top row : Canny edges retrieved from Validation images - Bottom row : images generated by our model (second method)

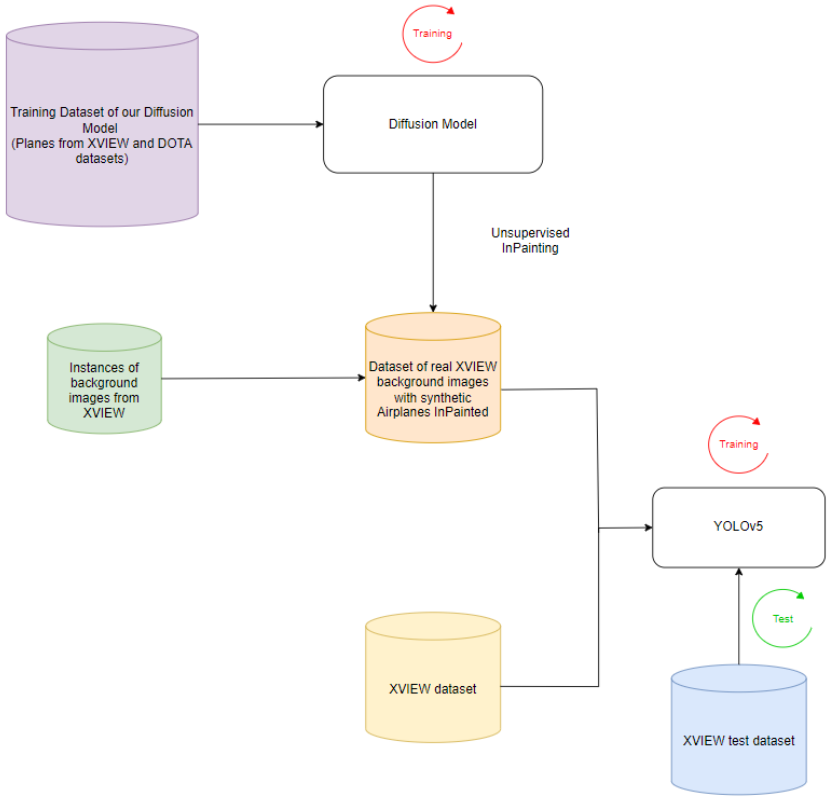

Figure 16 : High Level view of our method with the specificity of our use case taken in count

D.2 Unconditional InPainting

Our unconditional model doesn't offer the possibility to control the size of the plane generated nor the very fact of having a plane inpainted and not just background (see figure 17 for failure cases). Therefore to try to inforce these conditions, we train a classifier to classify an image based on the size of the plane composing it. The size of the plane is evaluated with the size of its bounding box. Thus the label associated to an image containing a plane with a bounding box of size (H_{bbox}, W_{bbox}) is $y = \lfloor 10 * (H_{bbox} + W_{bbox})/(2 * 256) \rfloor$ therefore creating 11 classes from size 0 to size 10. We then use this classifier for guidance during the InPainting process. Classifier Guidance [3] is a method to guide the denoising process toward a specific class using the gradient of a classifier trained on the given class.

Classifier guided diffusion sampling, given a diffusion model $(\mu_\theta(x_t), \Sigma_\theta(x_t))$, classifier $p_\phi(y|x_t)$, and gradient scale s.

Input: class label y, gradient scale s
$x_T \leftarrow$ sample from $\mathcal{N}(0, \mathbf{I})$
for all t from T to 1 **do**
  $\mu, \Sigma \leftarrow \mu_\theta(x_t), \Sigma_\theta(x_t)$
  $x_{t-1} \leftarrow$ sample from $\mathcal{N}(\mu + s\Sigma \nabla_{x_t} \log p_\phi(y|x_t), \Sigma)$
end for
return x_0

Figure 17: Sampling from a DDPM using Classifier Guidance. The pixels of the image infered are modified at every timestep using the gradient of the classifier.

To limit the number of failure cases we also tried an automatic cherry picking mechanism with the training of different binary ResNet, plane vs background and synthetic patch vs real patch. After visual inspection we decided to retain the Plane vs Background cherry picker. A binary classifier for guidance has also been trained on synthetic vs real data, despite being promising this approach doesn't solve the size of the plane being InPainted problem. Our final solution for unconditional InPainting is thus composed of our unconditional generative model, a classifier on the size of the plane generated for guidance and optionally a ResNet18 plane vs background cherrypicker.

Figure 18 Unconditional Generation with classifier guidance over the size of the bounding box. The guidance helps to generate a plane respecting the dimension of the bounding box (first failure case of fig 17)

Figure 19 : Failure case with our unconditional model. Left : The plane generated doesn't fill the InPainting mask, thus the bounding box isn't adapted (solved with guidance). Middle: No plane is generated (solved with guidance). Right; some parts of the generated plane are missing. This last failure case occurs when we generate the plane in a 'random context' : an image with a background totally out of the distribution of the diffusion model's training dataset. With these images we usually chose a non squared inpainting mask to limit the amount of context the model would need to infer. These tighter masks cause in certain instances these failures to generate the whole plane.

D.3 Conditional InPainting

Our conditional model has been trained with Canny Edges. Using Segment Anything [19] we created a dataset of mask and consequently of canny edges based on the planes of the training set. This dataset of masks and associated canny edges is then used at inference time to InPaint planes on different images without planes. During inference we randomly flip the masks and associated edges for diversity.

Figure 20: Xview Tiles InPainted with our conditional model

D.4 Detection experiments

We evaluate the usefulness of our inpainted synthetic planes with two approaches that will be detailed in this section. For each experiment two YOLOv5 are trained, with or without Focal Loss. The base training dataset for our YOLO is composed of 121966 instances of car, 24340 instances of truck, 3025 instances of boat and 658 instances of plane. The different YOLO trained are then tested on a test dataset of real images

D.4.1 Replacement

In our first experiment we replace real planes in real images by synthetic planes. We then train a YOLOv5 network (small version of the network) on a dataset composed of X% of real images with real planes and 100-X% of real images with synthetic planes. We then evaluate our model on the XVIEW test set and compare the results depending on X. For the replacement of images with real planes by images with synthetic planes we discard a given image containing N real planes and we inpaint N images of relevant backgrounds (tarmac) with a synthetic plane.

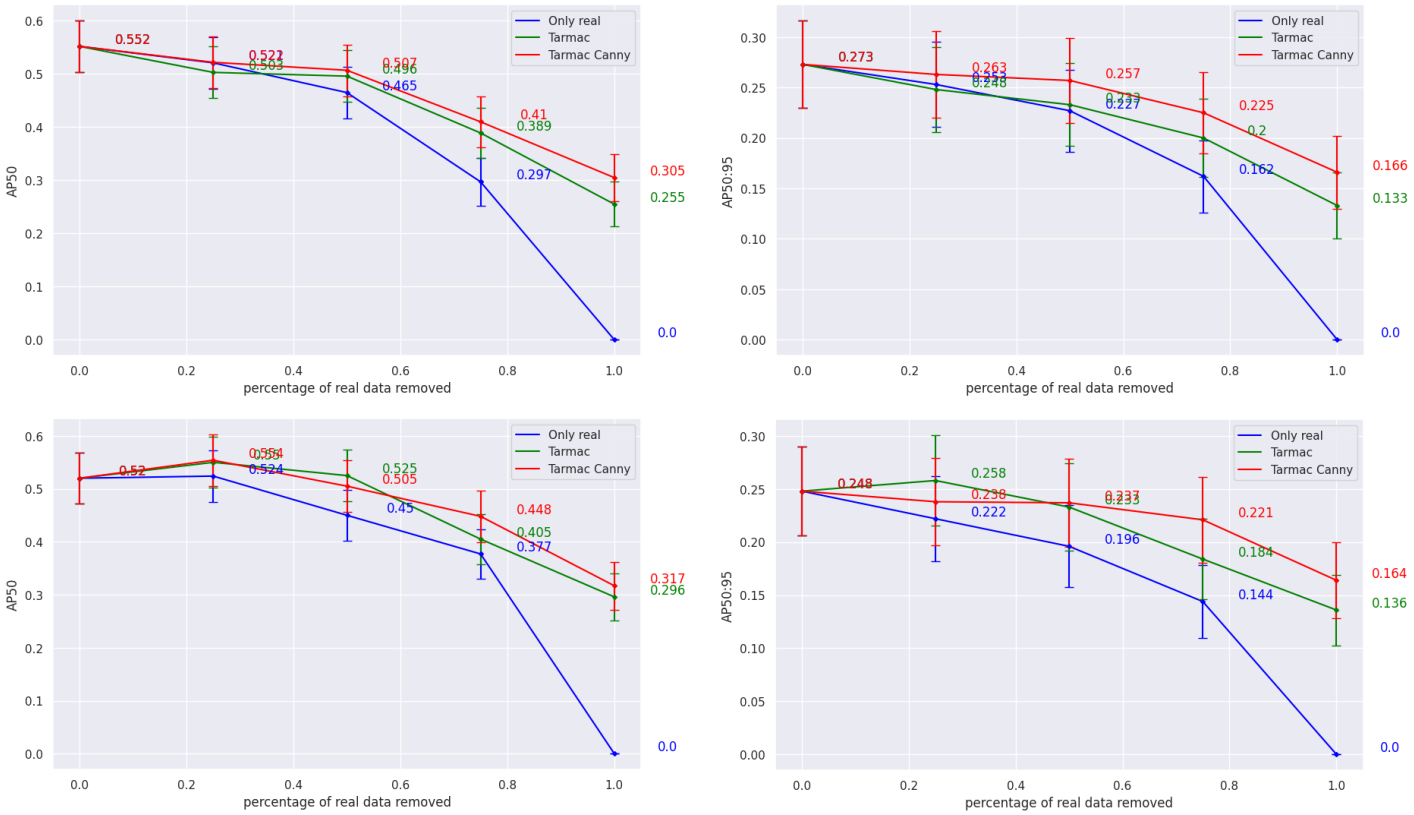

Figure 21 : AP50 and AP50:95 computed on the test set (plane class) and depending on the amount of synthetic data during the training of the YOLO. The blue curve corresponds to a pure ablation of real data without replacement by synthetic data. Confidence Intervals are computed based on the size of the test set. - Top row with Focal Loss - Bottom row without Focal Loss – Left column AP50 metric – Right column AP50:95 metric

For both the AP50 and AP50:95 metrics we observe a degradation in YOLO's performance when reducing the amount of real data in the training set. In extreme cases where the number of real instances becomes critically low (with more than 50% of real data removed), the graph shows that using synthetic data enables us to maintain a decent level of performance.

Relative difference between 100% real data and 100% synthetic data (using Canny Edges) with the same amount of instance.	With Focal Loss	Without Focal Loss
AP50	45%	39%
AP50:95	39%	34%

Some additional experiments have been done using the more recent YOLOv8-Medium for this replacement setup using Canny edges (corresponding to the previous "Tarmac Canny" experiment). All experiment has been averaged over 5 runs.

In the following Figure 21-bis, one can see that the detection performance is already quite close to the optimal with only synthetic planes during training. For mixes with more than 25% of real planes, performance tends to be almost constant. This demonstrates the good representativity/fidelity of the generated synthetical examples.

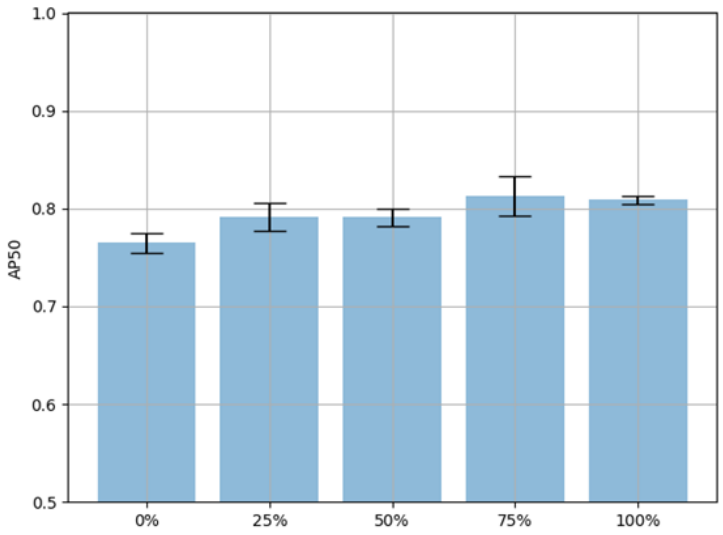

Figure 21-bis: AP50 computed on the test set and depending of the amount of syntehtic data during the training of the YOLOv8. Note that, here, the X-axis corresponds to the percentage of real planes. So, at the left side, no real planes are seen. At the right side, there is 100% of real planes. Confidence intervals are depicted as the standard deviation over 5 runs.

One might note that the latter performances are much higher and more stable than with the YOLOv5 setup. This is probably due to the fact that YOLOv8 has intrinsically higher performance and the use of a COCO pre-training. Also the use of a class-aware sampler during training limits the variability of the performance from one training to another.

D.4.2 Data augmentation

In our second experiment we keep all the real images with real planes, however we add to the training set N images with synthetic planes inpainted. In this second experiment two configurations are explored:

- We inpaint synthetic planes on random context images not containing any real plane. For the unconditional model we also use a ResNet18 plane vs background that cherry pics between 2 inpainted plane for a given image.
- We inpaint synthetic planes on images with relevant background (tarmac).

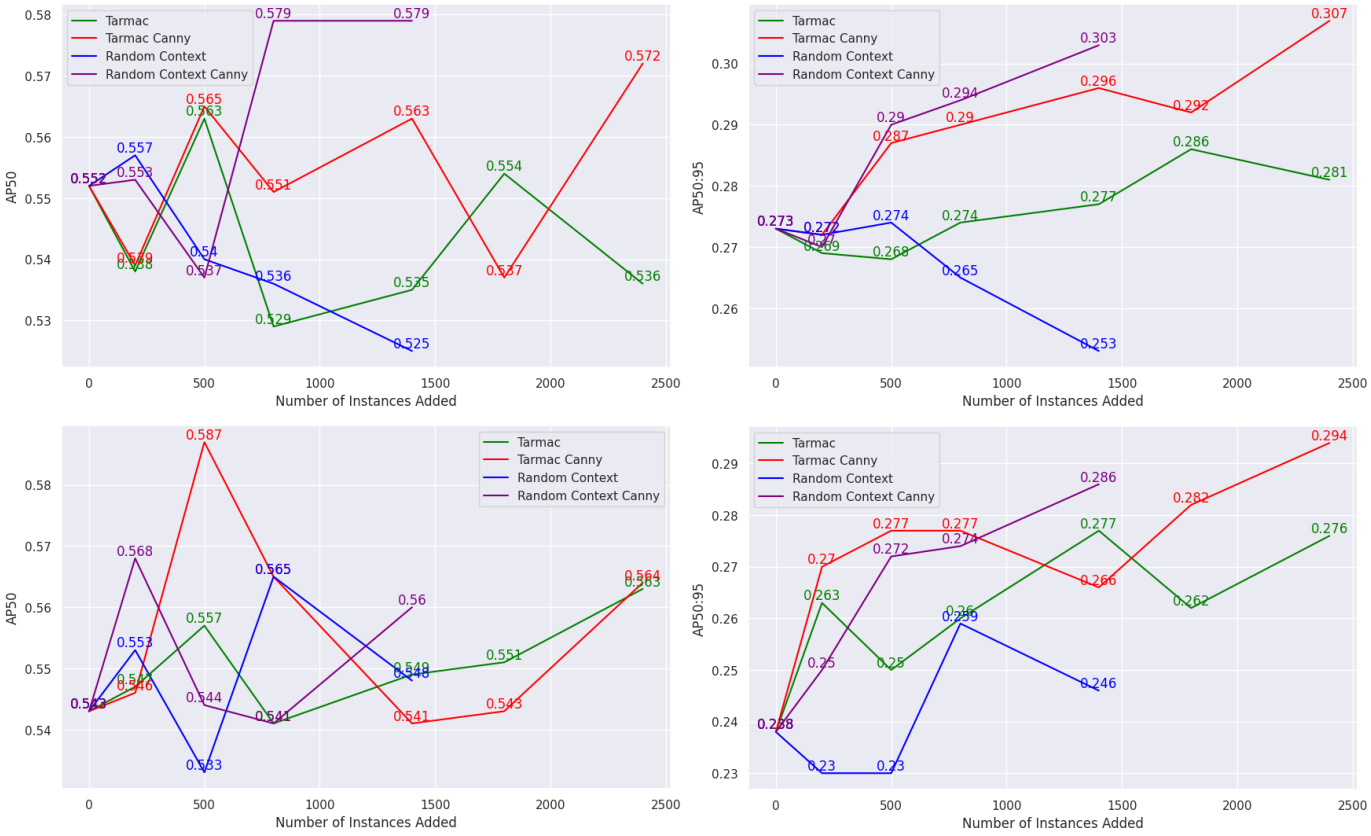

Figure 22 : AP50 and AP50:95 computed on the Test set (plane class), depending on the amount of synthetic data added to the YOLO's training set. The first point, labeled as '0 Number of Instances Added,' corresponds to the base YOLOv5 with the original training set. The top row represents results with Focal Loss, while the bottom row presents results without Focal Loss. The left column shows the AP50 metric, and the right column shows the AP50:95 metric.

The augmentation of the dataset with synthetic data does not show a clear improvement in the case of the AP50 metric, making it difficult to draw any conclusions regarding the benefits of such augmentation at training time. However, a clear trend can be observed in the case of the AP50:95 metric. Unlike AP50, this second metric reflects more accurately the quality of the inferred bounding box. By using synthetic data, we can generate plane with highly precise bounding boxes for the training of the YOLO. It positively impacts the quality of the bounding boxes infered at test time. This performance boost is particularly evident when using the Canny Edges models, where we have full control over the created plane and its location.

Here again, we reproduced part of latter experiments with a YOLOv8-Medium: we compare the baseline setup (*i.e.* the model is trained with all real planes only) and with

- the “Tarmac Canny” setup: during the training, all real planes are used and we add ~2500 new synthetic planes generated on a Tarmac context.
- The “Random Context Canny” setup: during the training, all real planes are used and we add ~2500 new synthetic planes generated on random backgrounds.

All experiment has been averaged over 5 runs. Results are depicted on Figure 22-bis. We can see that performance tends to be a bit higher when adding synthetic data, but the boost remains quite marginal.

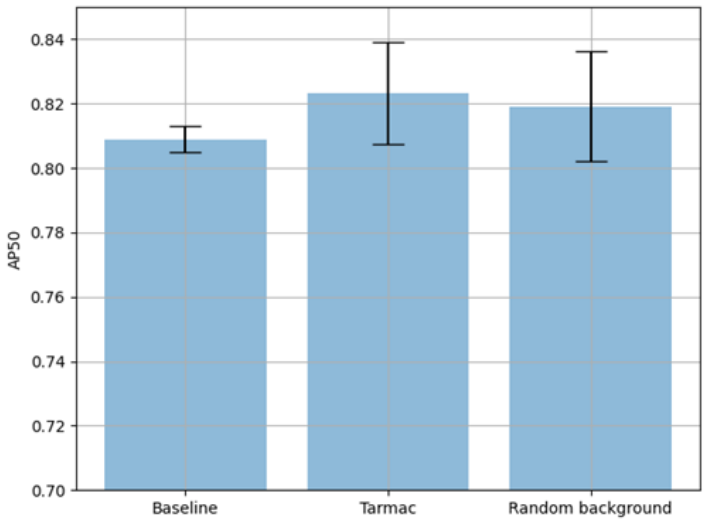

Figure 22-bis: AP50 computed on the baseline setup, Tarmac Canny and Random Context Canny. Confidence intervals are depicted as the standard deviation over 5 runs.

E. Discussions and analysis

The methodology introduced in the report is at the moment a pure innovation project and is still at its very first steps in term of maturity. Its application to a specific use case has however been tested.

The use of diffusion models causes inherent limitations, but this type of model is still relatively new as of today. One of the main limitations is the significant amount of time required for image generation during inference. Generating an image currently involves hundreds of forward passes with the model. However, researchers have made notable progress in reducing this number of steps, with the latest research managing to achieve as few as 20 steps. Some limitations more specific to our method also exist :

- We work in a limited data context, however we still need a certain amount of images of the targeted class to train the diffusion model. Typically in the work of Taehong Moon et al.[16] the order of magnitude for finetuning their diffusion models range from 800 to 1000. This range can fluctuate depending on the complexity of the images, their resolution and the size of the Neural Network used. The training dataset of the diffusion model however doesn't need to contain only images in the context of the dataset of the YOLO. Let's say a YOLO is trained to recognize persons in urban street scenes. The dataset of our diffusion model can also contain humans in a non urban street scene (a person playing golf, a person dancing etc…), that's what will help create diversity at inference time.
- If we train our model in an unconditional fashion, we have little control at inference time on the instance we are going to InPaint. In the worst case scenario the model could fill the scene with background and totally omit the instance of the class we wish to generate. Some tricks exist, classifier guidance for instance could be used to guide the generation towards the class. However we recommend to train the diffusion model with a condition such as a segmentation map, canny edges or human pose annotations to control the generation of the instance at InPainting stage.
- In this report we didn't investigate the adaptations that could be brought to the YOLOv5 for better integration of the synthetic data during the training. We could for example add an adversarial loss on the synthetic data vs real data feature maps of the YOLO.
- In the current implementation of our method the RePaint code is under a non commercial license. For more details concerning the license and its implication please refer to part H at the end of this document.

F. Conclusion

We showed in this report how to train a diffusion model with limited data and the possibility to control its output using extra inputs such as canny edges. The training of differents YOLOV5 and V8 with synthetic data generated by our diffusion model showed multiple limitations but also some interesting results

- With a critically low amount of real data, using synthetic data generated by diffusion allows us to maintain decent performance. In our use case, we show that getting rid of more than half of our 658 instances of real plane at training time leads to a significative drop of performance at test time. Using synthetic data to augment this truncated dataset of less than 400 real images helps to mitigate the drop of performance due to the low amount of data.
- While augmenting a dataset with synthetic data doesn't seem to augment the AP50 (except in the case discussed in the point above). Using synthetic data allows us to easily generate data with high quality label. This use of high quality label is reflected in a significative boost of performance on the AP50:95 metric. The model trained with additional synthetic data infers more accurate bounding box.

These results could be further investigated and different possibilities offered by our method could be assesed.

- The Canny Edges model offers the possibility to generate original example, different from everything the diffusion model trained on. With such possibility one could create a bank of original canny edges to further augment the dataset of mask and canny edges associated used during the creation of synthetic data. These images with original plane could help the YOLO detecting new kinds of planes.

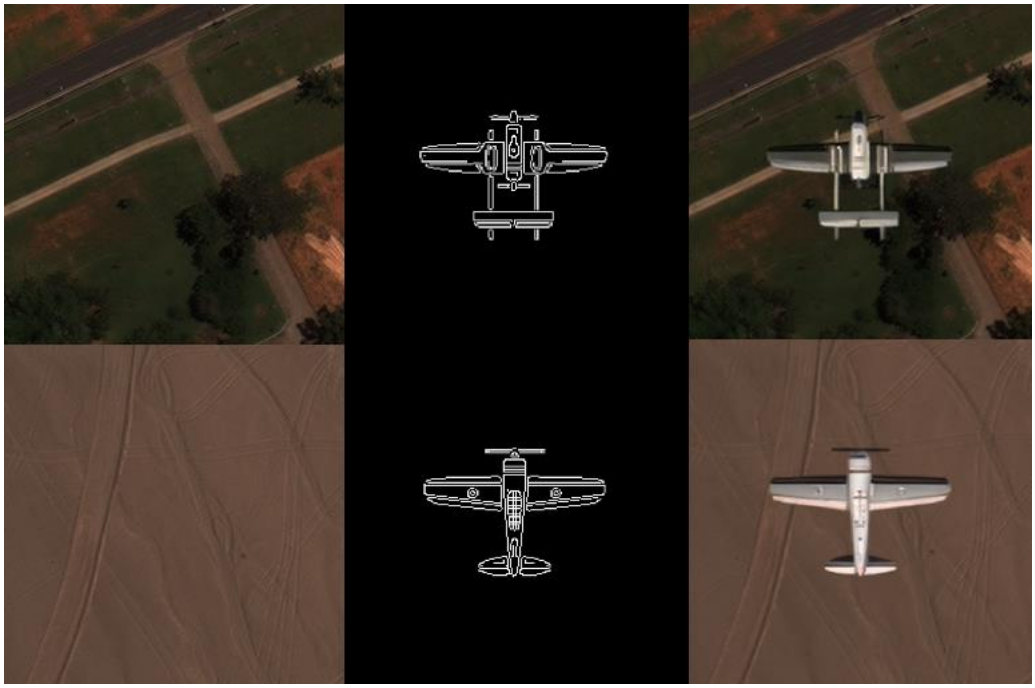

Figure 23: Generation of plane based on original canny edges

- Using Canny edge to condition our model has the advantage at training time to not require any hand made annotation : for each image canny edges can be easily computed using for instance OpenCV. However at inference time this method requires a dataset of planes' canny edges. Furthermore canny edges constitute a strong conditioning and the diversity of planes generated by our model is as wide as the number of planes' canny edges we possess in the corresponding dataset. Conditioning the model by segmentation maps instead of canny edges would give more freedom to our model for a given generation. Of course this solution would require an annotated training dataset for our diffusion model.

G. Annexes

G.1 Training Details

DIFFUSION MODEL	Global Batch Size	Number of iterations	Learning Rate	Optimizer	Schedule Sampler	LR annealing	Diffusion Steps
Unconditional Model	128	25000	0.0002	Adam	Uniform	False	1000
Canny Edges Model	128	25000	0.0002	Adam	Uniform	False	1000

- Models are pretrained on ImageNet 256 (models from [17])
- During the training of our Canny Edges model, canny edges for each image are generated using random threshold (between 1 and 255) with the opencv canny edge detector.
- As [14] we don't use exponential moving average during training. While during training 1000 diffusion steps are used, we use only 250 of them during inference (with the classic DDPM sampler). On a 2080Ti the generation of a 256x256 image with 250 timesteps takes approximatively 6 minutes.
- In our diffusion model training dataset, the small planes are undersampled. Thus we oversample small planes during training

YOLOV5	Global Batch Size	Epochs	LR init	LR final	Optimizer	Momentum	Weight decay
	32	1000	0.01	0.002	SGD	0.937	0.0005

YOLOV5	Box Loss Gain	cls loss gain	IoU training threshold	Anchor multiple threshold	Focal loss gain
	0.05	0.5	0.2	4	0/2

G.2 Loss details

The variational lower bound (VLB) can be written as follow :

$$L_{vlb} \coloneqq L_0 + L_1 + \cdots + L_{T-1} + L_T$$

$$L_0 \coloneqq -\log p_\theta\,(x_0|x_1)$$
$$L_{t-1} \coloneqq D_{KL}(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t))$$
$$L_T \coloneqq D_{KL}(q(x_T|x_0)||p(x_T))$$

Each term of L_{vlb} is a KL divergence between two gaussians and can therefor be evaluated in closed form. To evaluate L_0 for images, it is assumed that each color is divided into 256 bins, the probability of $p_\theta(x_0|x_1)$ landing in the correct bin is then computed. L_{vlb} is used to learned the $\Sigma_\theta(x_t, t)$ component of $p\_\theta(x_{t-1}|x_t)$. ϵ_θ is learned with Lsimple. Lsimple is a simple mean-squared error loss between the true noise and the predicted noise $||\varepsilon - \epsilon_\theta(x_t, t)||^2$. The final loss is $L = L_{simple} + \lambda L_{vlb}$

G.3 Pseudo Codes

Algorithm 1 DDPM sampling

$x_T \sim \mathcal{N}(0, I)$
for t = T,...,1 **do**
  $z \sim \mathcal{N}(0, I)$ if $t > 1$, else $z = 0$
  $x_{t-1} = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t, x_c)) + \sqrt{\Sigma_\theta(x_t, t, x_c)}z$
end for
return x_0

Figure 24: Pseudo code for sampling an image with a trained DDPM

Algorithm 2 InPainting using RePaint

$x_T \sim \mathcal{N}(0, I)$
for t = T,...,1 **do**
  for u=1,...,U **do**
    $\epsilon \sim \mathcal{N}(0, I)$ if $t > 1$, else $\epsilon = 0$
    $x_{t-1}^{know} = \sqrt{\bar{\alpha}_t}x_0 + (1 - \bar{\alpha}_t)\epsilon$
    $z \sim \mathcal{N}(0, I)$ if $t > 1$, else $z = 0$
    $x_{t-1}^{unknown} = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t, x_c)) + \sqrt{\Sigma_\theta(x_t, t, x_c)}z$
    $x_{t-1} = m \odot x_{t-1}^{known} + (1 - m) \odot x_{t-1}^{unknown}$
    if $u < U$ and $t > 1$ **then**
      $x_t \sim \mathcal{N}(\sqrt{\alpha_{t-1}}x_{t-1}, (1 - \alpha_{t-1})I)$
    end if
  end for
end for
return x_0

Figure 25: Pseudo code for InPainting an image using the RePaint method with a trained DDPM

H. About the Licence

In our current implementation available on GitLab the part of the code related to diffusion for image generation has been taken from [3] and modified. This code is under the permissive MIT Licence. However the part of the code related to InPainting is taken from the RePaint repository [9] and this code is under a non commercial licence (CC BY-NC-SA 4.0). Scripts concerned begin with a comment specifying the licence. The diffusers library from HuggingFace offers an alternative with a RePaint pipeline [20], yet this alternative doesn't offer the flexibility of using extra arguments for the model and thus conditioning our model on canny edges or using classifier guidance. The diffuser's code should be directly modified to fit our purpose. Another possibility would be to recode the RePaint code that is under non-commercial licence from scratch.

I. Bibliography

[1] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020.

[2] Alex Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. *arXiv preprint arXiv:2102.09672,* 2021.

[3] Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis. *arXiv preprint arXiv:2105.05233,* 2021.

[4] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *In NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021.

[5] Jonathan Ho, Chitwan Saharia, William Chan, David J. Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation. *Journal of Machine Learning Research*, 2022.

[6] Diederik P. Kingma, Tim Salimans, Ben Poole and Jonathan Ho. Variational Diffusion Model, 2022

[7] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

[8] Chitwan Saharia, William Chan, Huiwen Chang, Chris A Lee, Jonathan Ho, Tim Salimans, David J Fleet, and Mohammad Norouzi. Palette: Image-to-image diffusion models. *arXiv preprint arXiv:2111.05826*, 2021

[9] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, Luc Van Gool. RePaint: Inpainting using Denoising Diffusion Probabilistic Models. 2022

[10] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical Text-Conditional Image Generation with CLIP Latents. In arXiv, 2022.

[11] Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10684–10695, 2022.

[12] Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E., Ghasemipour, S. K. S., Ayan, B. K., Mahdavi, S. S., Lopes, R. G., et al. Photorealistic text-to-image diffusion models with deep language understanding. arXiv preprint arXiv:2205.11487, 2022.

[13] Tero Karras, Miika Aittala, Timo Aila, Samuli Laine. Elucidating the Design Space of Diffusion-Based Generative Models. 2022

[14] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. arXiv:1503.03585, 2015.

[15] Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. arXiv:2006.09011, 2020.

[16] Taehong Moon, Moonseok Choi, Gayoung Lee, Jung-Woo Ha, Juho Lee. Fine-tuning Diffusion Models with Limited Data. Neurips 2022 Workshop on Score-Based Methods, 2022.

[17] https://github.com/openai/guided-diffusion, Codebase for Diffusion Models beat GANS on Image Synthesis

[18] Lvmin Zhang, Maneesh Agrawala. Adding Conditional Control to Text-to-Image Diffusion Models. arXiv:2302.05543, 2023.

[19] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, Ross Girshick. Segment Anything. arXiv:2304.02643, 2023

[20] https://huggingface.co/docs/diffusers/api/pipelines/repaint, RePaint pipeline in the diffusers library by HuggingFace

[21] Score-based generative modeling through stochastic differential equations, Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, Ben Poole. arXiv:2011.13456v2, 2021

Title : Scientific Contribution on synthetic data generation using diffusion processes

Keywords : Diffusion models, Generative Modelling, Data Augmentation, Object Detection, Limited Data, Synthetic Data

This report propose a method to deal with limited data contexts. It shows how to train a diffusion model with limited data and the possibility to control its output using extra inputs such as canny edges. An example is given using Thalès Satellite Imagery use case, performing inpainting of aircrafs.

Our partners