EC5

JoliGEN user manual for style transfer

Document reference number for ANR

contact@confiance.ai | www.confiance.ai

Document reference: XXX

Contributors

	Name	Organisation	Role
Responsible for the deliverable	Yoann RANDON Hassan SLEIMAN	IRT-SystemX Renault SAS	Data Scientists
Scientific responsible	Amélie Bosca	Sopra Steria	Data Scientist
Co-authors			

Document control

Revision	Date	Commentary	Author
1.0	21/12/2023	OK	Pierre Pereira & Antoine Jacquet (Jolibrain)

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 Specific challenges

This document specifically addresses the challenge of acquiring high-quality data in a cost-effective manner. Conventional data collection methods employing real-world sensors may not adequately encompass the full spectrum of use case requirements. The purpose of this document is to provide guidance to novices in the field of deep learning, particularly those working with synthetic image generation, to facilitate the training of models on proprietary data utilizing the JoliGEN framework.

A.3 Objectives of the document

This document delineates the capabilities of the JoliGEN framework, a rebranded iteration of JoliGAN, which is centered around the generation of images through the application of neural network architectures. Within the scope of this framework, two distinct classes of generative models are explicated: Generative Adversarial Networks (GANs) and Diffusion models. The primary objective of this document is to furnish comprehensive insights into the framework's codebase, thereby empowering users to initiate and execute model training protocols pertinent to style transfer applications.

The discourse herein will predominantly concentrate on GAN-based algorithms, owing to the ongoing integration process of Diffusion models within the JoliGEN framework. It is imperative to acknowledge that the framework's code is accompanied by a dedicated documentation resource, accessible at https://www.joligen.com/doc/ . The documentation provides an in-depth explanation of the JoliGEN code at the commit version available at https://github.com/jolibrain/joligen/tree/204f48132b3e1f37b89dba04631402311a9123f0.

To generate the documentation of this specific commit version, one can check the "docs" folder within JoliGEN git version, where the documentation generation script is readily available.

B. Concept GAN

Generative Adversarial Networks (GANs) [1] constitute a sophisticated class of generative models that evolve through the adversarial interplay between two distinct neural network models: a generator and a discriminator. The generator is tasked with the synthesis of data, while the discriminator evaluates the authenticity of the generated data. The training process of a GAN involves a dynamic competition between the generator and the discriminator, orchestrated by a loss function. This loss function acts as a judge in this competition, rewarding or penalizing each model based on their performance. The generator strives to create data that is indistinguishable from real data, aiming to fool the discriminator. Conversely, the discriminator works to accurately distinguish between the genuine data and the fabrications of the generator. They are both driven to improve by the feedback from the loss function: if the generator succeeds in deceiving the discriminator, the loss function will penalize the discriminator for its error. If the discriminator correctly identifies the generator's data as fake, the loss function will penalize the generator. Through this tug-of-war, where the loss function score reflects the current state of play, both models continually adapt and enhance their strategies to outperform the other.

The training of a GAN is inherently unstable due to its adversarial nature. To mitigate this instability, the implementation of multiple discriminators and the addition of various regularization loss functions can be employed. These techniques help to stabilize the training process by ensuring that neither the generator nor the discriminator becomes too powerful too quickly, maintaining a balance that allows for gradual learning and adaptation on both sides.

GANs play a vital role in translating images without the need for direct pairings, facilitating the transformation of image styles autonomously. By mastering the intrinsic data distributions, GANs adeptly produce images that authentically resemble the desired domain. This innovation is particularly valuable in areas where acquiring matched datasets is challenging, providing substantial advantages across various applications.

In the following We present a more detailed explanation for each component mentioned: generator, discriminator, and loss function.

B.1 Generator

Within the architecture of Generative Adversarial Networks (GANs), the generator is conceptualized as a neural network whose primary objective is the synthesis of images. The foundational approach for a generator is to commence with a noise vector (z), which is an n-dimensional vector of stochastic values. This vector subsists within a multidimensional space known as the latent space, which encapsulates the entire spectrum of potential values for the vector.

The generator's training involves the development of a mapping function from the latent space to the data distribution of a given dataset. As the generator iterates through the training process, it refines this mapping, enabling the production of images that are statistically similar to the original data distribution. Consequently, the generator can yield novel images that are variations of the training dataset.

For applications involving style transfer, the role of the generator is adapted to transform the stylistic elements (such as texture and colour schemes) from a source dataset A to align with those of a target dataset B. To facilitate this, the generator incorporates an autoencoder framework. The encoder portion of the generator takes images from dataset A and encodes them into a condensed feature vector. Subsequently, the decoder portion of the generator takes this feature vector and reconstructs it into an image that embodies the style characteristics of dataset B.

This dual-phase process enables the generator to not only learn a representation of the original data but also to transpose stylistic attributes from one dataset to another, effectively accomplishing style transfer. The autoencoder's ability to encode and then decode the data is central to this process, allowing for the transformation of stylistic elements between datasets.

In our case, we want to train the generator for a style transfer issue, which mean we want to change the style (textures, colour, ...) of a source dataset A to target dataset B. So, instead of starting from a noise vector, we will use an autoencoder structure, which will map images from A to a feature vector with the encoder part of the generator, then the decoder part will map feature vector to an image from dataset B.

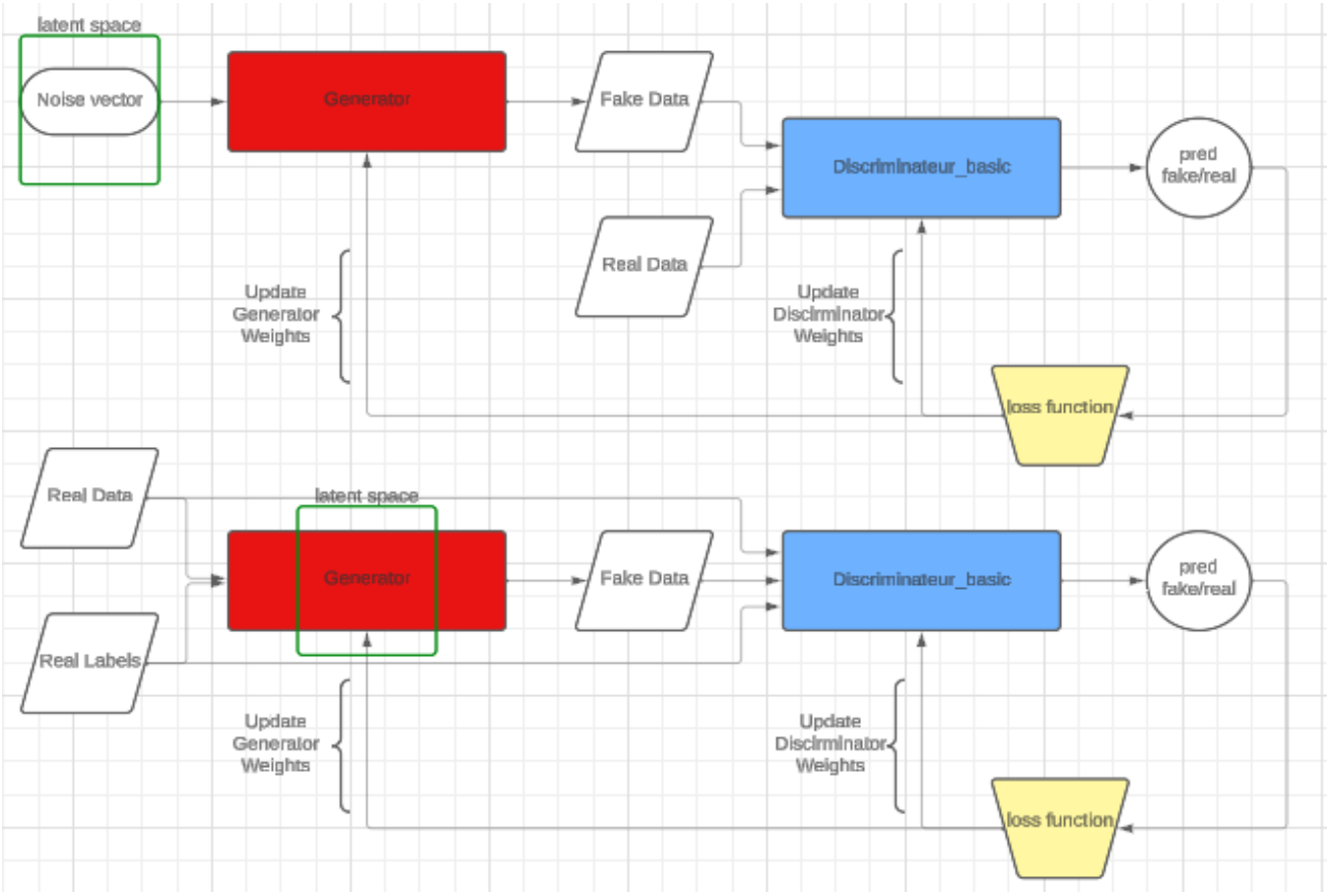

Figure 1: architecture GAN noise2image et image2image

B.2 Discriminator

The discriminator, an integral counterpart within the Generative Adversarial Network (GAN) schema, is instantiated as a neural network with the express function of differentiating between images that either

conform to or deviate from a specified target style. In essence, the discriminator operates as a binary classifier, utilizing a series of convolutional layers to process and scrutinize the stylistic attributes of input images.

Upon receiving an image, the discriminator engages its convolutional network to extract features and patterns pertinent to the style in question. It then assesses these features to ascertain whether the image exhibits the target style characteristics. The discriminator's output is a probabilistic estimate reflecting the likelihood that the image adheres to the stylistic criteria of the target dataset.

To augment the robustness of the generative process and impose more stringent constraints on the generator, it is feasible to deploy an ensemble of discriminators. Each discriminator within this ensemble is tailored to evaluate specific facets of the generated images, such as texture, color distribution, or other stylistic nuances. The collective assessment provided by multiple discriminators ensures a more nuanced and comprehensive evaluation of the generated data, thereby enhancing the fidelity of the style transfer. Some of those discriminators will be presented in a future part.

B.3 Loss functions

Loss functions serve as a pivotal mechanism to impose regulatory constraints on the adversarial agents within a Generative Adversarial Network: the generator and the discriminator. A prevalent loss function employed in this adversarial context is Binary Cross Entropy (BCE), which is formally defined as follows:

$$BCE = -\frac{1}{N}\sum_{i=0}^{N} y_i \cdot log(\hat{y}_i) + (1 - y_i) \cdot log(1 - \hat{y}_i)$$

Here, (N) represents the number of samples, (y_i) denotes the ground truth label, and ($\hat{y}_i$) is the predicted probability for the ith sample.

The choice of loss function extends beyond BCE, with a multitude of alternatives available, each tailored to the specific characteristics of the discriminator's architecture and the desired properties of the generator's output. The incorporation of multiple loss functions into the adversarial training regime introduces a spectrum of constraints, guiding the networks towards more complex and nuanced models of data representation. The diversity of loss functions correlates with the degree of constraint: the greater the number of loss functions applied, the tighter the control exerted on the adversarial dynamics. The selection and integration of these loss functions are contingent on the discriminator's structure, with the aim of optimizing both the authenticity of the generated images and the accuracy of the discriminator's evaluations.

C. State of the art

In this section, we shall concentrate exclusively on a triad of sophisticated GAN architectures: CycleGAN introduced by Zhu et al. in 2017 [E], Contrastive Unpaired Translation (CUT) introduced by Park et al. in 2020 [D], and RecycleGAN introduced by Bansal et at. [G] in 2018. These architectures have been selected for their novel contributions to the field, each presenting distinctive mechanisms and algorithms that address and surmount the nuanced challenges associated with their respective image translation tasks.

C.2 CycleGAN

CycleGAN [E] is an algorithm used for image-to-image translation tasks where paired training examples are not available. This ground-breaking method was developed by Jun-Yan Zhu et al. in 2017. The core idea behind CycleGAN is to learn a mapping G: X → Y that converts an image from a source domain X to a target domain Y and the inverse mapping F: Y→ X . To achieve this without paired examples, the algorithm employs two key components: an adversarial loss and a cycle consistency loss.

The adversarial loss ensures that the distribution of the generated images G(X) is indistinguishable from the distribution of the target domain Y as well as F(Y) for domain X. This is achieved by introducing adversarial discriminators DY and DX, which are trained to classify whether the translated images belong to the target domain or not.

The cycle consistency loss is what sets CycleGAN apart from other image-to-image translation methods. It enforces that an image from domain X, when translated to domain Y through G and then back to X through the reverse mapping F: Y → X, should result in an image close to the original. In mathematical terms, it ensures F(G(X)) ≈ X and G(F(Y)) ≈ Y, thus forming a cycle. This consistency acts as a regularizer, providing an additional constraint that helps guide the learning process and prevents the model from producing nonsensical translations.

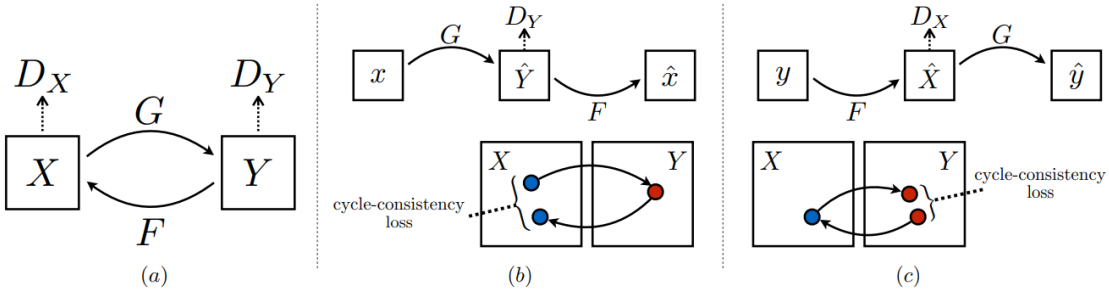

Figure 2: The CycleGAN model employs two generators, G: X → Y and F: Y → X, with corresponding adversarial discriminators, DY and DX, and enforces fidelity through forward and backward cycle-consistency losses to ensure that each domain translation, when reversed, approximates the original input [E].

For its network architecture, CycleGAN utilizes a generator network based on the architecture proposed by Johnson et al. [G], comprising convolutional layers, residual blocks, and fractional-strided convolutions. The discriminator networks are based on the PatchGAN architecture, which operates at the scale of patches of an image, helping to focus on high-frequency details and making the model applicable to images of any size.

CycleGAN has been successfully applied to various applications, including artistic style transfer where photos are rendered in the style of famous painters like Monet or Van Gogh, object transfiguration like transforming horses into zebras, and even seasonal transformation of landscapes. Its versatility is further highlighted in tasks such as photo enhancement, where it can take smartphone photos and generate versions that look like they were taken with DSLR cameras.

Despite its successes, CycleGAN does have limitations, particularly with tasks that require significant geometric changes between the source and target domains. Additionally, while it can approximate the results of supervised methods that use paired examples, there remains a performance gap that future research might address. Nonetheless, CycleGAN's ability to leverage unpaired image data is a significant achievement in the field of machine learning and computer vision.

C.3 CUT

Contrastive Learning for Unpaired Image-to-Image Translation is an approach proposed by Taesung Park et al [D] in 2020, which aims to address the problem of image-to-image translation in a way that preserves content between the input and output images despite changes in domain-specific appearances.

The key idea behind this approach is to maximize mutual information between corresponding patches in the input and output images using a contrastive learning framework. In contrast to traditional methods that may use adversarial losses to enforce appearance and cycle-consistency losses to preserve content, this method directly encourages patches that correspond across the input and output to map to similar points in a learned feature space. This is achieved by using a contrastive loss function, specifically the InfoNCE loss, which learns an embedding that brings corresponding patches closer together while pushing non-corresponding patches (negatives) apart.

The algorithm makes use of a multi-layer, patch-based approach. Instead of operating on entire images, it focuses on small patches, which allows for more detailed and localized learning of correspondences. Additionally, negatives are drawn from within the input image itself rather than from an external dataset, which further reinforces the content preservation aspect by making the patches preserve the internal content structure.

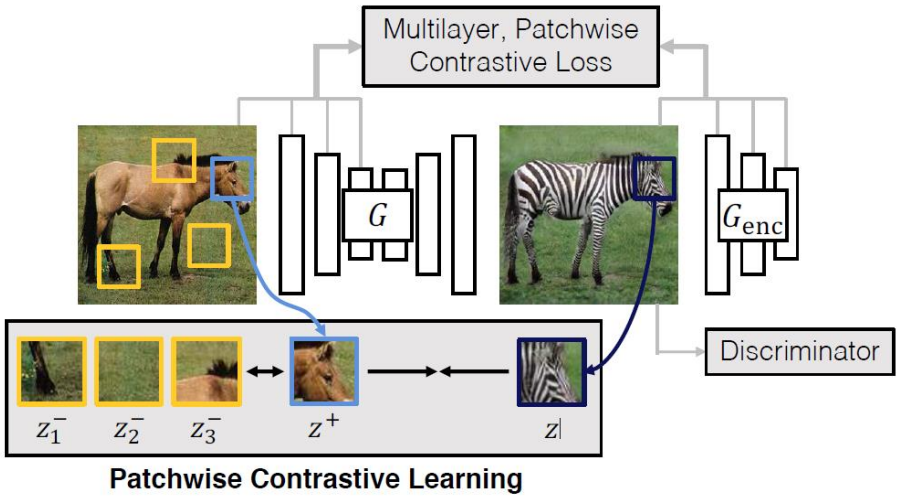

Figure 3: Illustration of patchwise contrastive learning in one-sided translation, where a multilayer contrastive loss ensures a generated output patch is closer to its corresponding input patch than to any other non-corresponding patches [D].

Some key technical details of the proposed method include:

- The use of a multilayer, patch-wise contrastive loss that maximizes mutual information between corresponding input and output patches.
- The employment of a multilayer approach where features are extracted from different layers of an encoder network to represent patches at various scales.
- Sampling negatives from within the same input image, which encourages the preservation of content by leveraging the internal statistics of the image.
- Avoiding the need for additional components such as memory banks or specialized architectures, simplifying the training procedure and reducing computational requirements.

The authors also demonstrate that this framework enables one-sided translation in the unpaired setting and can be extended to training scenarios where each domain consists of only a single image.

Overall, this approach to unpaired image-to-image translation uses contrastive learning to effectively maintain content correspondence while allowing for domain-specific appearance changes, leading to improved quality and reduced training time compared to previous methods.

C.4 RecycleGAN

Recycle-GAN: Unsupervised Video Retargeting is a cutting-edge technique introduced by Aayush Bansal et al. from Carnegie Mellon University and Facebook Reality Lab in 2018 for transferring content from one domain to another in a style-preserving manner [G]. This method is particularly useful in applications where

both spatial and temporal information are significant, such as face-to-face translation, flower blooming processes, and environmental changes like sunrise and sunset.

The core algorithm leverages a combination of spatial and temporal constraints along with adversarial losses to achieve content translation and style preservation without the need for paired or labeled training data. Unlike previous methods that relied solely on spatial information, Recycle-GAN incorporates temporal information, which provides additional constraints to the optimization process, leading to better local minima and more accurate style learning.

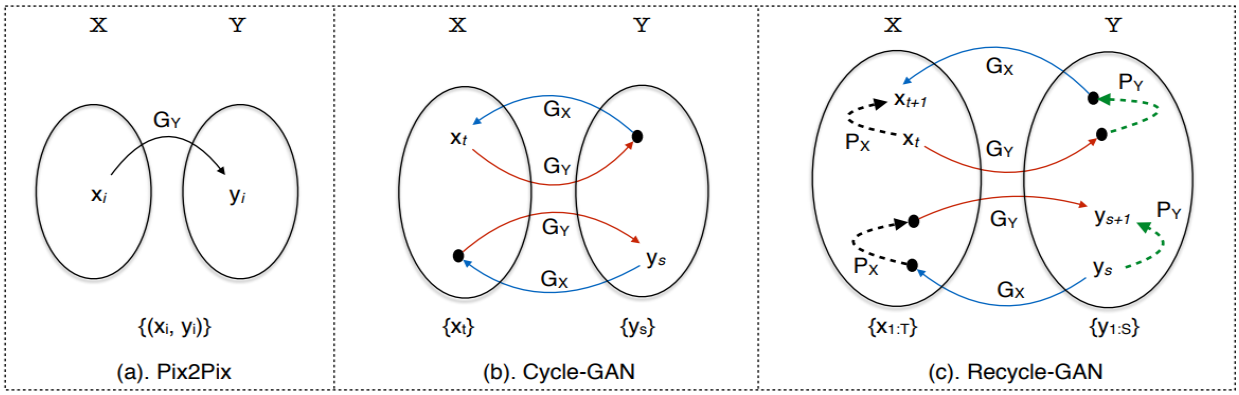

Figure 4: Overview of image-to-image translation approaches contrasting Pix2Pix, Cycle-GAN, and Recycle-GAN [G]

Recycle-GAN introduces a novel approach that integrates spatiotemporal cues with conditional generative adversarial networks (GANs) for video retargeting. The model consists of two main components: a mapping function (GY) that transforms frames from one domain to another, and a temporal predictor (PY) that ensures the temporal coherence of sequences.

The key contributions of Recycle-GAN are:

- A new formulation that combines spatial and temporal constraints for improved video retargeting.
- The use of publicly available video data to learn transformations between domains without manual supervision.
- Demonstrations of the approach across various natural processes, such as face-to-face translation and environmental changes.

The experiments conducted by the authors demonstrate that Recycle-GAN outperforms Cycle-GAN, which only uses spatial information. The introduction of temporal constraints helps to avoid common issues such as perceptual mode collapse and spatially tied inputs, leading to more natural and realistic video translations.

D. Dataset format

Before starting the training, it is important to verify our data have the corresponding format. We will present in the following parts the dataset format to perform style transfert with unlabeled data, class conditioned data, bounding box as label data and mask as label data in JoliGEN framework.

JoliGEN is a versatile framework that supports datasets in various formats, both labeled and unlabeled. The utilization of labels, referred to as semantic constraints, narrows down the range of potential pixel combinations for image generation, enhancing control over the output. The dataset format is available in the documentation https://www.joliGEN.com/doc/datasets.html .

D.1 Unlabelled data

Training with unlabelled data is the simplest form of dataset preparation within JoliGEN. This approach is employed when metadata is not available or not required for the generation task. To prepare an unlabelled dataset, the first step is to divide the collection of images into training and testing subsets. Typically, this split follows the convention of allocating 80% of the images to the training set and reserving the 20% balance for the test set. These subsets should then be placed in corresponding folders.

The structure for an unlabelled dataset is straightforward and consists of two primary subdirectories within the data folder: trainA and trainB. These subdirectories contain images from domain A (the source) and domain B (the target), respectively. When evaluation is needed, additional subdirectories testA and testB shall be created to house the test images.

Here is an example of how to organize an unlabeled dataset for a domain transfer task such as converting horses to zebras without any accompanying metadata:

- horse2zebra/
  - trainA/: Contains 80% of horse images for training.
  - trainB/: Contains 80% of zebra images for training.
  - testA/: Contains 20% of horse images for testing.
  - testB/: Contains 20% of zebra images for testing.

The absence of labels makes this dataset format the most accessible for users who are new to image generation or for tasks where metadata is not crucial. The next section will explore the use of datasets conditioned with class labels, which introduce metadata into the training process for more controlled image generation outcomes.

D.2 Conditioned by class

When utilizing conditioned datasets with class labels in JoliGEN, an additional channel is incorporated into each image to designate its class affiliation. This channel is an image-sized layer where every pixel holds the value corresponding to the class number. For instance, if an image is categorized under class 2, the added channel will be a matrix with the same dimensions as the image itself, and every element of this matrix will be set to the value 2.

This method of dataset preparation is crucial for tasks that require a clear association between the input and output images based on their class labels. It allows the JoliGEN framework to maintain the integrity of class-specific characteristics during the image generation process.

The conditioned dataset should be organized into two primary directories, trainA and trainB, similar to the structure for unlabeled datasets. Within trainA, subdirectories are created for each class, and these contain the images belonging to their respective classes.

Here's an illustrative example of a dataset structure for class-conditioned data, using a task that transforms images of one font to another:

- font_conversion/
  - trainA/
    - 0/: Contains images of the numeral 0 in the original font.
    - 1/: Contains images of the numeral 1 in the original font.
  - trainB/
    - 0/: Contains images of the numeral 0 in the target font.
    - 1/: Contains images of the numeral 1 in the target font.

In this setup, JoliGEN allows to effectively learn to convert the numerals from one font to another while preserving the numeral's identity, thanks to the class channel that provides additional information to the model.

The next section will elaborate on datasets that include bounding boxes as a form of data conditioning, allowing for even more precise control in the image generation process.

D.3 Bounding box

In JoliGEN, datasets incorporating bounding boxes are essential for scenarios where images contain multiple elements and precise identification of each is required. Bounding boxes enable the specification of the class to which each element belongs by encasing it within a rectangular outline. This spatial annotation is particularly useful when the model needs to distinguish between different items within the same image.

The bounding box format in JoliGEN is specified as 'cls minx miny maxx maxy', where:

- cls is the class identifier starting from 1, whereas 0 is reserved for the background.
- minx and miny denote the coordinates of the top-left corner of the rectangle.
- maxx and maxy represent the coordinates of the bottom-right corner.
- The x and y variations correspond to the width and height of the image, respectively, and increase from the top left to the bottom right of the image.

During the training process, the JoliGEN parser associate images with their corresponding bounding boxes through the utilization of a text file referred to as "paths.txt". This file serves the purpose of linking each image file to its corresponding bounding box file. Each line of the "paths.txt" file includes the path of the image as well as the path of its associated bounding boxes. The semantic masks utilized by the model for guiding image generation are constructed through the utilization of these bounding boxes.

Here's how to structure a dataset with bounding boxes for a task like converting Super Mario to Sonic, while preserving the character's position and action:

- mario2sonic/
  - trainA/
    - imgs/: Contains image files of Mario.
    - bbox/: Contains text files with bounding box details for each image.
  - trainB/
    - imgs/: Contains image files of Sonic.
    - bbox/: Contains text files with bounding box details for each image.

Each Mario and Sonic image will have an associated text file in the bbox folder detailing the bounding boxes in the mentioned format.

The subsequent section will address the usage of masks in datasets, which allows for even more detailed semantic segmentation in the JoliGEN training model.

D.4 Mask

In JoliGEN, mask-based training comes into play when fine-tuned semantic segmentation is desired, particularly with images that contain intricately shaped objects. Masks are used to demarcate specific regions within an image, allowing the model to learn and generate nuanced details with precision.

A mask is provided as a PNG file, which is an image format that supports a single-channel, grayscale representation. Within this channel, pixel values are used as labels to distinguish between different classes present in the image. These values range from 0 to n-1, where n represents the total number of unique classes identified in the dataset.

This approach to segmentation ensures that the model has a detailed understanding of the various parts of the image to be generated or modified. By delineating each class with a unique label, the model can accurately recreate or transform specific regions within the image without affecting the others.

To structure a mask-based dataset for JoliGEN, you would organize it similarly to other datasets but with the inclusion of mask directories. Here's an example for a task involving the addition of eyewear to faces:

- faces_with_glasses/

- trainA/
  - imgs/: Contains the source images, for example, faces without glasses.
  - masks/: Contains corresponding mask files, each defining the region around the eyes.
- trainB/
  - imgs/: Contains the target images, for example, the same faces with glasses.
  - masks/: Contains corresponding mask files, each defining the region where the glasses are located.

Each mask file correlates with an image file and is used during training to guide the model in generating or altering the image in a targeted manner. By leveraging masks, JoliGEN can facilitate detailed and context-sensitive modifications, resulting in highly accurate and realistic image generation outcomes.

E. Training

This section outlines the process of initiating and managing model training using the joliGEN framework. We will begin by detailing the preparation of the training environment, followed by how to execute training commands with key options. Additionally, we will provide guidance on starting a training session and conclude with a brief introduction to using Visdom for real-time training visualization.

E.1 Environment

The requisite libraries for deploying the joliGEN codebase are enumerated within the "requirements.txt" file, located in the project's GitHub repository. This list facilitates the establishment of a suitable environment by delineating the necessary dependencies.

In instances where prior installations of libraries may conflict with those required by joliGEN, the use of a Docker container is advisably employed. Documentation pertinent to the acquisition and deployment of the joliGEN Docker image is comprehensively available. The utilization of Docker ensures a consistent and isolated environment, advantageous for reproducibility and mitigating dependency-related issues.

Subsequent to environment setup, it is necessary to initialize a Visdom server instance. Visdom serves as a tool for real-time monitoring of the model's training progression across epochs. To ensure persistent operation, irrespective of server connectivity, the Visdom server should be executed as a background process utilizing tools such as nohup, screen, or systemd. Training durations are contingent on variables such as dataset length and training parameters; thus, background operation is essential to prevent process termination upon user disconnection.

The subsequent sections will expound upon the configurable options within joliGEN.

E.2 Options

The efficacy of model training is significantly influenced by the selection of training options. The joliGEN framework provides an extensive suite of configurable parameters, exceeding one hundred in number, tailored to a variety of training scenarios. While it is not feasible to elucidate each option individually, the ensuing discussion will focus on salient options pivotal to the training process. Comprehensive documentation of all available options can be accessed at joliGEN Documentation: https://www.joliGEN.com/doc/options.html

- Options for train.py file:

	Description
checkpoints_dir	Directory for saving models and associated training data.
dataroot	The file path leading to the dataset.
gpu_ids	Specification of GPU identifiers for computational allocation during training.
model_type	Determination of the model type to be employed in training.

name	Designation of the training session, forming a subdirectory within checkpoints_dir.

These options predominantly pertain to storage and computational resource allocation, which are foundational for model training and development.

- Discriminator

	Description
D_netDs	A list of discriminators to be engaged during training.

The discriminator options influence the constraints imposed on the generative model, with a diversity of discriminators introduced to augment training robustness.

- Generator

	Description
G_netG	Specification of the generator model, with the Contrastive Unpaired translation model frequently employed in GAN architectures.

The option netG refer to the generator model, the most utilize GAN Generator model is the Contrastive Unpaired translation [F].

- Dataset handling

	Description
data_load_size	Initial size for loading images.
data_crop_size	Dimension to which images are cropped post-loading.
data_dataset_mode	Dataset mode, typically unaligned for unpaired datasets.
data_num_thread	An increased thread count can expedite data loading operations.

The framework JoliGEN use a crop and resize function to simulate a biggest amount of data. Those data are load to a specific size data_load_size then they are cropped to size data_crop_size. Depending on the Use Case you might use a specific dataset mode data_dataset_mode. In a way of time computation optimization, it may be useful to increase the number of thread data_num_thread to load data faster.

- Visdom

	Description
output_display_visdom_server	Server IP address for Visdom display, defaulting to localhost.
output_display_visdom_port	Network port for Visdom access, with a default value of 8097.

Visdom is an important tool in the framework JoliGEN. We need to define in which server output_display_visdom_server it's instance will be create and port output_display_visdom_port will be used to access it. A presentation of this tool will be done bellow.

E.3 Initiating Training with joliGEN

Prior to the initiation of the model training process utilizing the joliGEN framework, systematic verification is required to ensure that all prerequisites for a successful training session are met. The following checkpoints must be addressed:

- Data Validation: Confirm that the dataset adheres to the specified format as outlined in the "Dataset Format" section of the documentation. The format must be compatible with the joliGEN framework's requirements to facilitate accurate processing and analysis.
- Configuration File Appropriateness. Retrieve the configuration file from the "examples" directory provided with joliGEN. It is imperative to ascertain whether the parameters within this file are congruent with the use case it is intended to serve. Adjustments should be made to reflect the specific needs of the training objectives.
- Visdom Instance Activation: Establish that a Visdom server instance is actively running. Visdom is instrumental for real-time monitoring of the model's training progression, thus its operational status is crucial.

Once the criteria are satisfied, training can be initiated by executing the train.py script. This procedure is explained in detail within the joliGEN Training Documentation. It is noteworthy that any options specified at the command line will supersede the corresponding default settings in the configuration file.

To resume training from a previously saved epoch, the --train_continue flag should be employed. This flag instructs the framework to continue training from the last saved checkpoint. While the default behavior of joliGEN is to save the model upon the completion of each epoch, customization of this behavior and other training parameters is achievable through modifications of the configuration file.

A more detailed explanation on training on BDD100K for style transfer on meteorological data is available on JoliGEN documentation website https://www.joliGEN.com/doc/tutorial_styletransfer_bdd100k.html or in the experimental document [H] produced by Confiance.ai.

E.4 Visdom

Visdom is a Python-based visualization library that facilitates the creation of interactive and real-time visualizations. Originally developed by Facebook Research, its primary application is the visualization of data throughout the neural network model training process, with a particular focus on models implemented using the PyTorch framework.

E.4.1 Overview of Visdom's Functionality

Visdom offers the capability to generate dynamic visualizations across a diverse array of data types, including but not limited to images, graphs, and plots. The interactive nature of these visualizations allows for an engaging analysis experience, proving especially beneficial in the following contexts:

- Loss Curve Plotting: Visdom enables the plotting of loss curves, providing an intuitive representation of model performance over time.
- Image Display: It can display images directly related to the training process, allowing for visual assessment of model outputs.
- Embedding Visualization: Visdom supports the visualization of embeddings, offering insights into the feature space learned by the model.
- Metric Observation: Other training metrics can be monitored, aiding in the comprehensive assessment of model behavior.

The web-based interface provided by Visdom ensures that data can be visualized and monitored seamlessly as the machine learning model undergoes training. This interface is designed for ease of use, allowing users to explore and manipulate visualized data effectively.

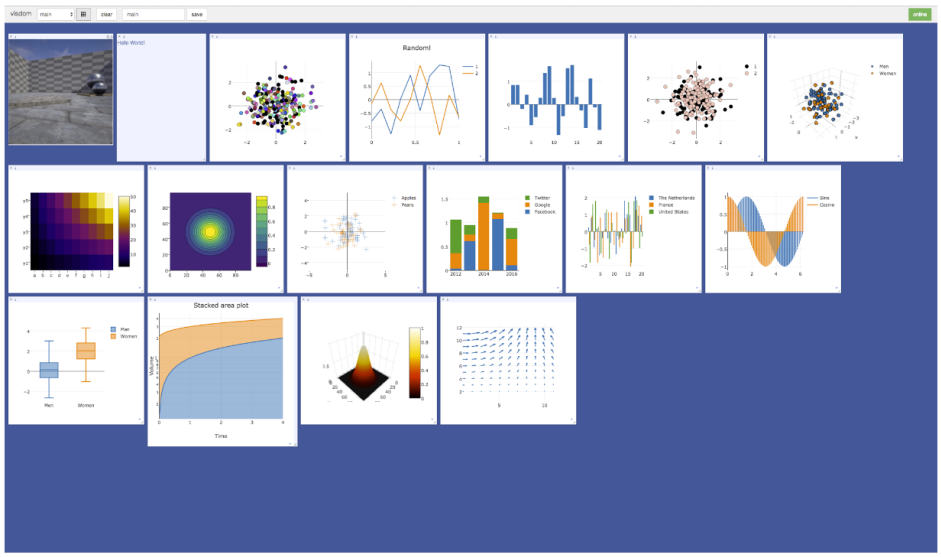

Figure 5: Visdom interface

E.4.2 Integration with joliGEN

Within the joliGEN framework, Visdom is utilized to graphically represent loss functions alongside the visualization of masks and images pertinent to the model currently under training. This tool is instrumental in verifying the correctness of the training procedure, as it provides evidence of whether the model is learning as expected.

However, for individuals new to the domain—referred to as neophytes—the mastering of Visdom's functionalities can present challenges. The presence of numerous loss functions, potentially in the order of ten or more, necessitates a robust understanding of the role each model plays in the training process. Without such knowledge, identifying issues, barring those that are overtly indicative of model failure, can be a complex task.

E.4.3 Resources and Accessibility

The Visdom source code, along with its documentation and illustrative examples, is accessible via its GitHub repository. This repository is an asset to the research and development community engaged in machine learning endeavours, particularly for those who utilize PyTorch and seek to gain deeper insights into their data and model performance over the course of training. This repository is an asset to the research and development community engaged in machine learning endeavours, particularly for those who utilize PyTorch and seek to gain deeper insights into their data and model performance over the course of training.

The Visdom code repository is available at: Visdom GitHub Repository: https://github.com/fossasia/visdom

F. Inference

The inference stage is a critical component of the machine learning workflow where a trained model is applied to new data to make predictions or generate outputs. In the context of joliGEN, this process is facilitated by leveraging a specifically formatted configuration file that encapsulates the model's training parameters.

- Configuration File: train_config.json

At the conclusion of a model's training phase, joliGEN automatically generates a configuration file named train_config.json. This file embodies the settings and hyperparameters that were utilized during the training process, and it is stored within the model's designated directory. The integrity of the inference process is predicated on the congruence of conditions between training and inference, making the train_config.json file an indispensable artifact for maintaining this consistency.

- Utilization During Inference

The gen_single_image.py script, which is executed for the purpose of inference, necessitates the presence of the train_config.json file in the same directory as the model file indicated by the --model-in-file option. The script relies on this configuration file to instantiate the model with the appropriate parameters, thereby assuring that the model operates as expected during the inference phase.

- Template Scripts for Model Integration

The Python scripts made available by joliGEN, such as gen_single_image.py, are not only functional but also serve as illustrative templates for developers who aim to incorporate joliGEN's models into alternative codebases. These scripts provide exemplars of how to correctly load and execute a pre-trained model for inference-related tasks. They are amenable to customization and can be modified to accommodate the requisites of various projects or research endeavors.

- Leveraging Trained Models for Synthetic Image Generation

By adhering to the protocols delineated in these scripts and ensuring the availability of train_config.json, practitioners can harness the capabilities of joliGEN's trained models. The models can be utilized to generate synthetic images that embody the insights and knowledge gleaned during the training phase. The applications of this technology are manifold, ranging from the augmentation of visual content through style transfers to the synthesis of training datasets for machine learning models across diverse domains.

G. Conclusion

To conclude, JoliGEN is a diverse tool that aims to collect a wide variety of models for solving the synthetic image generation problem. While the tool offers control over every parameter of each collected model, this abundance of options may discourage new users from utilizing the tool.

Having a good understanding of the model being trained is crucial for effective training. This knowledge allows users to identify, through the loss functions chart provided by visdom, whether any parameter changes are necessary.

One of the main strengths of JoliGEN is its utilization of the crop and resize function, which expands the number of input images. However, this can also be a weakness if one of the source or target domains lacks the necessary characteristics on each possible patch, resulting in a partial failure of the style transfer.

It is important to note that the code is still in development, and improvements and bug fixes are expected to be implemented in the near future.

H. Bibliography

[A] A.Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2020). Generative adversarial networks. Communications of the ACM, 63(11), pages 139-144.

[B] A.Liu, R., Wang, X., Lu, H., Wu, Z., Fan, Q., Li, S., & Jin, X. (2021). SCCGAN: style and characters inpainting based on CGAN. Mobile networks and applications, 26, 3-12.

[C] A.Pachade, S., Porwal, P., Kokare, M., Giancardo, L., & Meriaudeau, F. (2021). NENet: Nested EfficientNet and adversarial learning for joint optic disc and cup segmentation. Medical Image Analysis, 74, 102253.

[D] Park, Taesung, et al. "Contrastive learning for unpaired image-to-image translation." Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IX 16. Springer International Publishing, 2020.

[E] Zhu, Jun-Yan, et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks." Proceedings of the IEEE international conference on computer vision. 2017.

[F] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In ECCV, pages 694–711. Springer, 2016. 2, 3, 4

[G] Bansal, Aayush, et al. "Recycle-gan: Unsupervised video retargeting." Proceedings of the European conference on computer vision (ECCV). 2018.

[H] Sleiman, H. JoliGEN application to weather style transfer on images. Confiance.ai Batch3.

Title : JoliGEN user manual for style transfer

Keywords : JoliGEN, GAN, Deep learning, Computer vision, Style transfer

The burgeoning integration of Artificial Intelligence (AI) into critical systems necessitates a foundational trustworthiness to gain widespread acceptance. This document explores the challenges in establishing such trust, including the intrinsic design of trustworthy AI models, detailed characterization, hardware integration, advanced data engineering, verification, certification, and the need for a holistic methodological framework. The Confiance.ai program aims to tackle these challenges. This document focuses on the specific challenge of acquiring high-quality, cost-effective data, highlighting the inadequacies of traditional data collection methods. It introduces the JoliGEN framework, designed to assist novices in deep learning, particularly in synthetic image generation for model training. The document details how to use JoliGEN, a rebranded version of JoliGAN, and expounds on the use of Generative Adversarial Networks (GANs) and Diffusion models for style transfer applications within this framework. Although the integration of Diffusion models is ongoing, the document concentrates on the GAN-based algorithms and directs users to the framework's comprehensive documentation and codebase, underscoring the importance of the JoliGEN framework in the AI development lifecycle.

Our partners