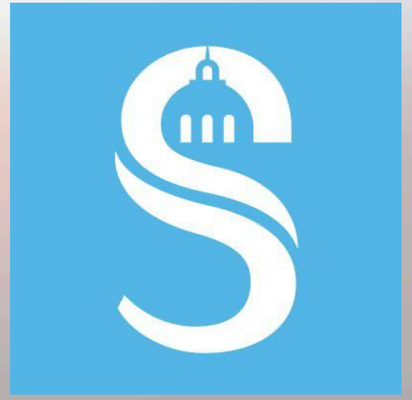
Few-Shot Image Generation for Damaged Products

Leveraging Advanced Dataloading and Flow-Matching Models for Industrial Applications

Hamed Rahimi, PhD - June 2025



Project Overview & Objectives

Develop an efficient pipeline for generating high-quality, class-conditional images of damaged products, particularly in low-data regimes.

Key Objectives:

- Develop a performant and efficient dataloader for image pre-processing.
- Select and fine-tune a suitable pre-trained diffusion model for few-shot generation.
- Implement fine-tuning for generating new class-conditional images of specific damage types.
- Explore advanced techniques like Textual Inversion.
- Devise automated, quantitative metrics for image quality evaluation.

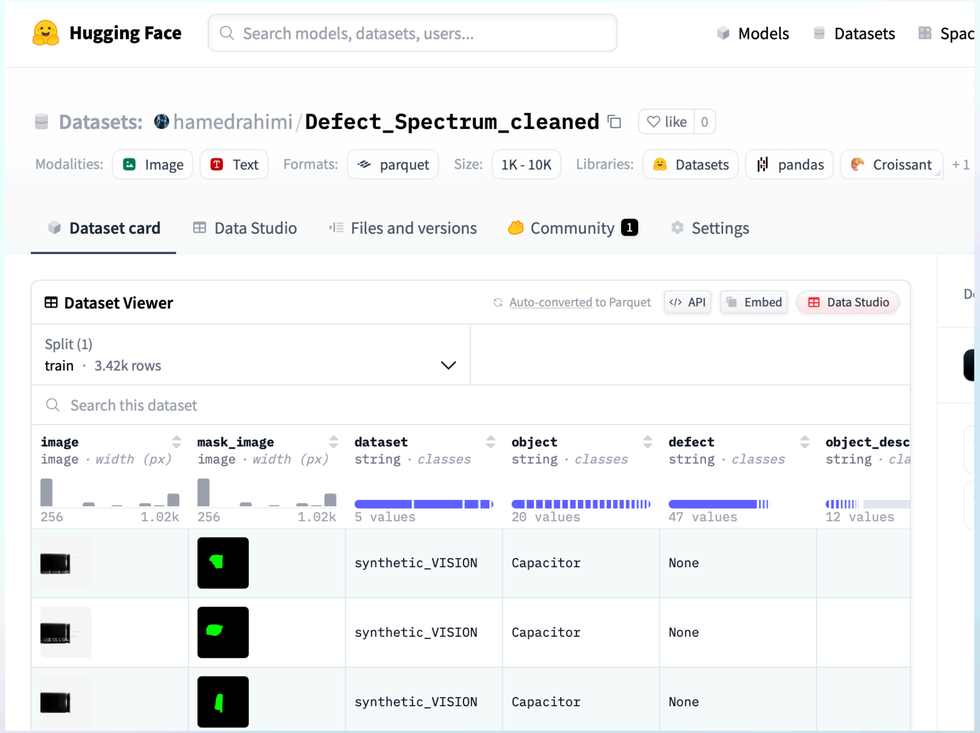
ISIR
INSTITUT
DES SYSTÈMES
INTELLIGENTS
ET DE ROBOTIQUE

Optimized Dataloader for Image Pre-processing

Non-homogeneously ordered metadata in the dataset.

Solution:

- Read all metadata (image directory, object type, defect, descriptions if exist) into a Pandas DataFrame.
- Load DataFrame as a Hugging Face datasets object.
- Push to Hugging Face Hub (*hamedrahimi/Defect_Spectrum_cleaned*) for easy access and reusability.
- **Code available in data_preprocessing notebook**

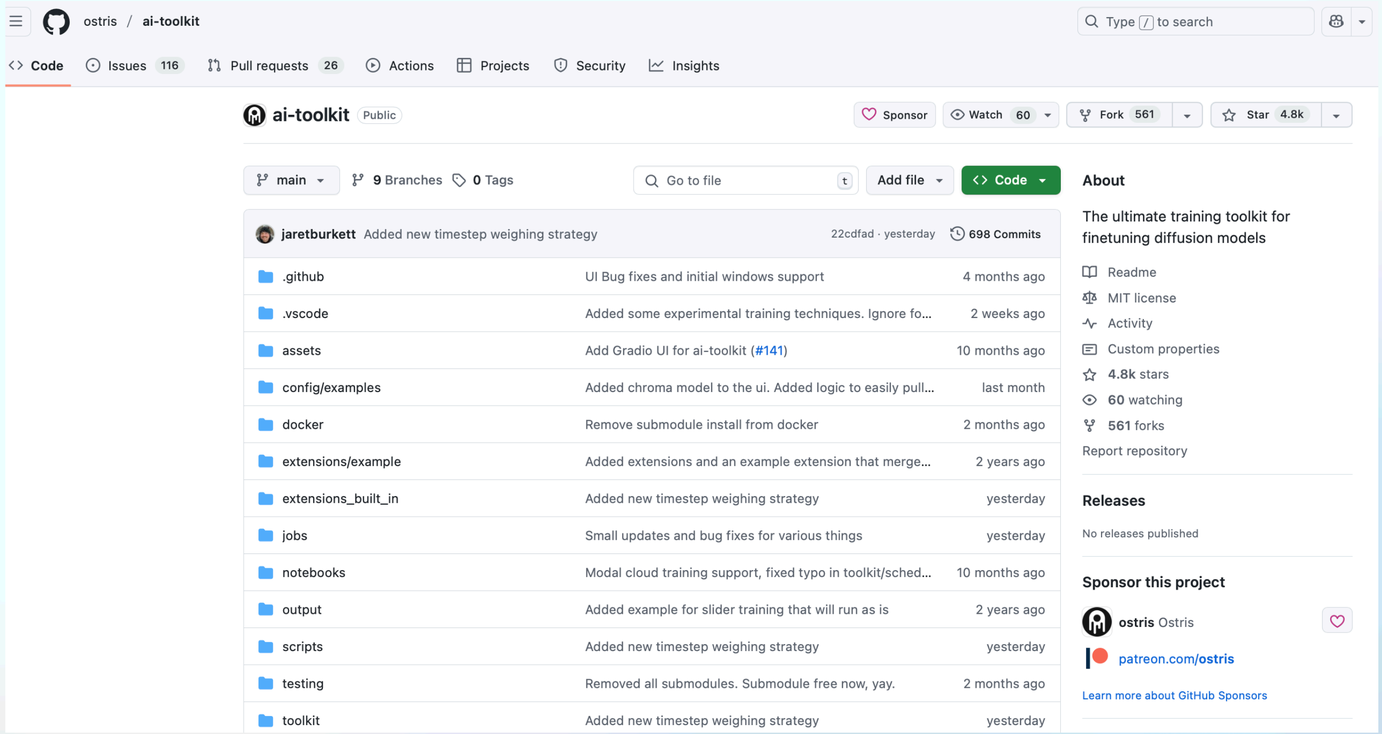


ISIR
INSTITUT
DES SYSTÈMES
INTELLIGENTS
ET DE ROBOTIQUE

Optimization for Multi-GPU:

Automatically done by AI Toolkit by Ostris

- **Distributed Sampler:** Implementing torch.utils.data.distributed.DistributedSampler to ensure each GPU processes a unique subset of the data.
- **Pin Memory:** Using pin_memory=True in DataLoader to speed up data transfer to GPU.
- **Asynchronous Data Loading**: Leveraging multiple worker processes (num_workers > 0) for parallel data loading and pre-processing.
- **Caching:** Implementing caching mechanisms for pre-processed batches in memory or on disk to avoid redundant computation.
- **Mixed Precision Training:** Utilizing torch.cuda.amp.autocast for faster training and reduced memory consumption.
- **Efficient I/O:** Optimizing image loading with libraries like Pillow-SIMD or OpenCV-Python for faster decoding.



ISIR
INSTITUT
DES SYSTÈMES
INTELLIGENTS
ET DE ROBOTIQUE

Pre-trained Diffusion Model Selection & Hyperparameters

Model Choice: FLUX (black-forest-labs/FLUX.1-schnell)

Flow-matching-based text-to-image generation for its efficiency and smaller size, suitable for available compute.

Fine-tuning Method: LoRA (Parameter-efficient fine-tuning, demonstrated strong performance.)

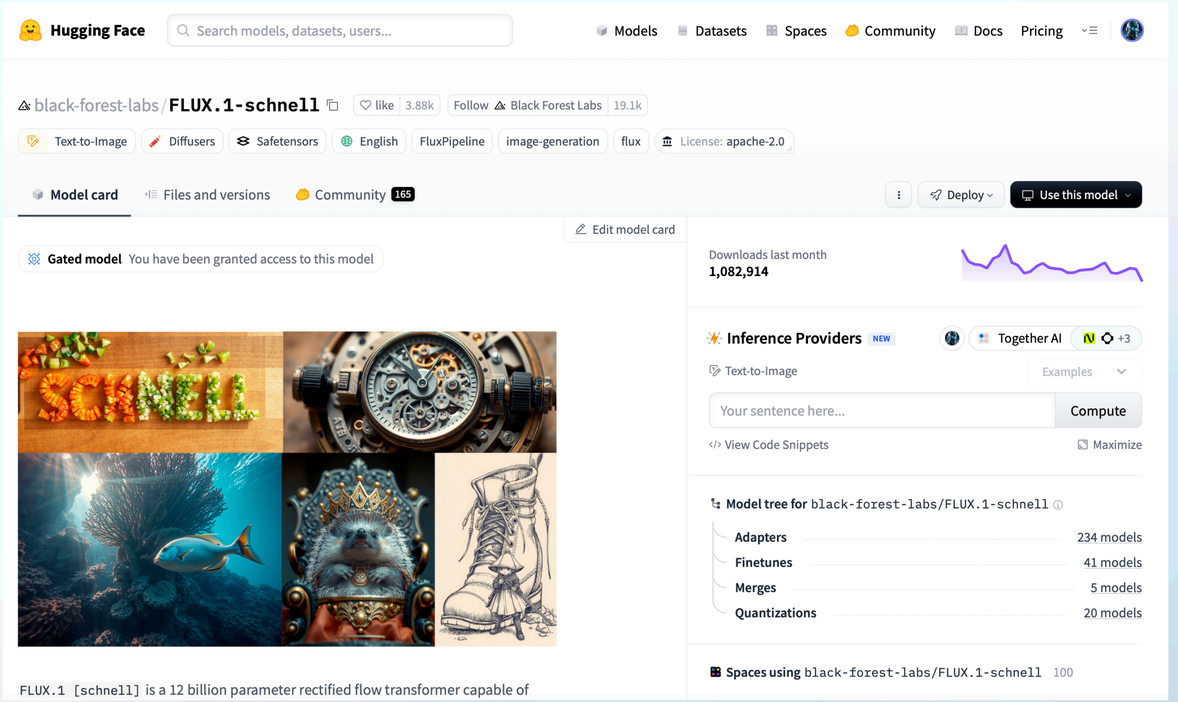
- **LoRA Parameters:**

rank (r): 32 (efficient, comparatively low training cost).

alpha: 32 (typically same as rank).

Optimizer: adamw8bit (Low-cost, fast, efficient for convergence, and good performance.)

Learning Rate (lr): 1e-4



INSTITUT DES SYSTÈMES INTELLIGENTS ET DE ROBOTIQUE

Class-Conditional Image Generation & Data Strategy

- **Strategy for Missing Descriptions:**

For products without descriptions, we should use Vision-Language Models (VLMs) to generate textual descriptions. These VLM-generated descriptions will then be used for further fine-tuning and training.

- **Inference Notebook:**

Have prepared a notebook for easy image generation with existing descriptions.

```python
import torch
from diffusers import FluxPipeline

pipe = FluxPipeline.from_pretrained("black-forest-labs/FLUX.1-schnell", torch_dtype=torch.bfloat16)

pipe.enable_model_cpu_offload()

prompt= " ".join([filtered_ds["train"][0]["object_description"],filtered_ds["train"][0]["defect_description"]])

image = pipe(
    prompt,
    height=1024,
    width=1024,
    guidance_scale=3.5,
    num_inference_steps=50,
    max_sequence_length=512,
    generator=torch.Generator("cuda").manual_seed(0)
).images[0]

image.save("defect.png")
```

Model Fine-tuning Implementation

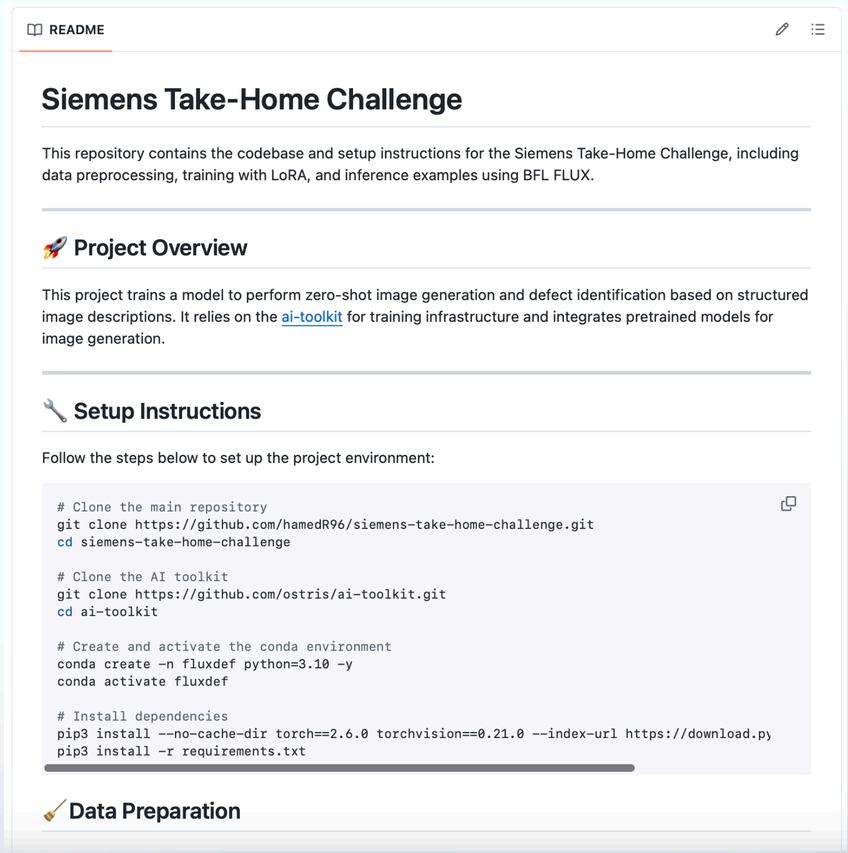
Ostrach AI toolkit

Benefits:

- Simplifies efficient fine-tuning of FLUX, especially with multi-GPU setups.
- Automatically distributes images to each node/GPU.

Process Overview:

- Loading Defect_Spectrum_cleaned dataset.
- Loading pre-trained black-forest-labs/FLUX.1-schnell.
- Attaching LoRA layers to the model (Att & FF).
- Configuring adamw8bit and learning rate scheduler.
- Iterating through batches, compute loss, backpropagate, and update LoRA weights.
- Ostrach AI toolkit handles distributed training automatically.


README

Siemens Take-Home Challenge

This repository contains the codebase and setup instructions for the Siemens Take-Home Challenge, including data preprocessing, training with LoRA, and inference examples using BFL FLUX.

🚀 Project Overview

This project trains a model to perform zero-shot image generation and defect identification based on structured image descriptions. It relies on the ai-toolkit for training infrastructure and integrates pretrained models for image generation.

🔧 Setup Instructions

Follow the steps below to set up the project environment:

```
# Clone the main repository
git clone https://github.com/hamedR96/siemens-take-home-challenge.git
cd siemens-take-home-challenge

# Clone the AI toolkit
git clone https://github.com/ostris/ai-toolkit.git
cd ai-toolkit

# Create and activate the conda environment
conda create -n fluxdef python=3.10 -y
conda activate fluxdef

# Install dependencies
pip3 install --no-cache-dir torch==2.6.0 torchvision==0.21.0 --index-url https://download.py
pip3 install -r requirements.txt
```

🧹 Data Preparation

ISIR
INSTITUT
DES SYSTÈMES
INTELLIGENTS
ET DE ROBOTIQUE

Utilizing Segmentation Masks for Data Augmentation

[Purpose 1] Data Augmentation & Textual Information Generation:

- Using segmentation masks to isolate damaged regions.
- Employing a VLM to generate richer textual descriptions based on the masked regions (e.g., "a deep scratch on the upper left corner of the pill").
- This enriches the dataset, especially for samples with generic or no descriptions.

[Purpose 2] Image-to-Image Generation (e.g., FLUX Kontext):

- Leveraging masks to guide image generation, allowing for more controlled and precise generation of damaged areas.
- This enables generating more diverse and realistic data by manipulating specific regions or introducing new damage types within existing images.

Textual Inversion for Specific Damage Types

Learning a new text embedding for a specific token (e.g., sks_scratch) that represents a particular damage type.

- Ref: https://github.com/AUTOMATIC1111/stable-diffusion-webui/wiki/Textual-Inversion

Implementation (using AUTOMATIC1111 Stable Diffusion Web UI) since it allows for precise control over generating specific visual concepts not inherent in the base model's training data.

- Curating a small set of example images showcasing the sks_scratch damage.
- Initializing an embedding with a base token (e.g., "scratch").
- Supplying the directory of curated images.
- Set training parameters (learning rate, number of steps).
- Usage: Place the trained .pt or .bin embedding file in the UI's embeddings folder.
- Generation Prompt Example: "a photo of a pill with vertical sks_scratch"

Automated, Quantitative Metrics for Image Quality Evaluation

Core Metrics (with Reference Images):

- Fréchet Inception Distance (FID): Measures the similarity between real and generated image distributions (lower is better).
- Learned Perceptual Image Patch Similarity (LPIPS): Quantifies perceptual similarity between images using deep features.
- Structural Similarity Index Measure (SSIM): Assesses structural similarity, luminance, and contrast.

Metrics for Learned Token Effect (e.g., Textual Inversion):

- Prompt Sensitivity Score (PSS): Quantifies how much the learned token influences the generated output.
- Attention Map Analysis: Visualize and analyze attention maps to see if the token correctly focuses on the intended visual concept.

Other Quality Checks (not very common):

- Intra-class Diversity: Ensuring generated images within a class are varied.
- Inter-class Separability: Verifying distinct visual features for different damage types.
- Object Detection/Damage Classification: Using pre-trained models to confirm the presence and type of expected damage.
- BRISQUE (Blind/Referenceless Image Spatial Quality Evaluator): Assessing image quality without reference images.
- Natural Image Statistics: Analyzing statistical properties to detect artifacts or unrealistic patterns.
-


ISIR
INSTITUT
DES SYSTÈMES
INTELLIGENTS
ET DE ROBOTIQUE