Few-shot generation of images of products with defects by fine-tuning large diffusion models

Avadhut Sardeshmukh

April 25, 2025

1 Problem Statement

Asset maintenance and inspection are critical components of operating power lines and substations. Performing these tasks at scaleacross thousands of kilometers of infrastructureposes significant challenges. Early detection of visible degradation, faults, or other anomalies can lead to substantial cost savings and improved system reliability. However, a key obstacle is the limited availability of anomalous image data for each defect type.

To address this, one promising approach is to augment training data using synthetic images. Toward this end, the present task is to create a conditional image generation model for this purpose by fine-tuning a pre-trained diffusion model on limited labeled data of the target product and defects.

2 Possible Solutions

Conditional image generation is a fundamental problem in computer vision, and has received a lot of attention in recent research. With the rise of models such as generative adversarial networks (GANs) and variational autoencoders (VAEs), deep generative models started showing promise for synthetic image generation. While VAEs model and optimize the data likelihood explicitly, GANs do this implicitly through an adversarial network, leading to sharper images with better perceptual quality. GANs quickly became popular and were adapted for diverse applications (such as generating synthetic speech data and e-commerce visuals). However, GANs are known to be notoriously difficult to train because of the underlying mini-max game. A flurry of research addressed the problems of GAN training stability and other issues such as mode-collapse (e.g. WGAN, WGAN-GP, SpectralNormGAN and so on). Meanwhile, with the advent of better compute and learning architectures, the liklihood (and score)-based modeled regained the attention, resulting in the recent advances in diffusion models. Latest diffusion models such as DDPM, GLIDE, Imagen, ControlNet, and Stable Diffusion 3 are very powerful image generators. While the original DDPM model was unconditional, the follow-up research effectively addressed the problem of conditioning through various ways :

- Adding the conditioning information as extra channels in the ResNet blocks
- Combining conditioning information with timesteps in diffusion
- Using the gradients from a separately trained classifier
- Guidance (e.g. Contrastive Language-Image Pretraining Guidance for alignment with text)
- Classifier free guidance

These models have internalized various semantics of different image types through their extensive training on millions (or even billions in some cases, such as SD3) of images with and without labels. They make a very good choice for making a model for few-shot image generation on a new class of images.

2.1 Fine-tuning methods

Some possible ways to fine tune these pre-trained models for our task are :

1. Full fine-tuning
   The huge parameter space of these models poses a significant challenge in full fine-tuning since the optimizer needs to maintain the state of all the parameters, requiring large GPU memory.
2. Parameter efficient fine-tuning
   Much of recent literature on synthetic image generation for various applications has focused on parameter efficient fine-tuning methods. The notable ones among these include tuning just the biases (BitFit [1]), tuning based on prompts (VPT [2]), low-rank adaptation methods from the language modeling literature (LoRA [3], DriveDitFit [4]), and tuning biases and newly added scales (DiffFit [5]).
3. Augmenting model's conditioning space
   This class of methods is particularly suitable for fine-tuning text-conditional models such as stable diffusion. They are based on the idea of creating new "words" in the embedding space which correspond to new concepts. Most notable approaches include Dreambooth [6] and Textual Inversion [7].
4. Distillation
   These methods train smaller models by leveraging the training signal from the large models, effectively "distilling" the knowledge contained in the larger models.

3 Modeling and Implementation

3.1 Modeling Considerations

3.1.1 Diffusion Model

The task includes class-conditional generation of defective product images. Unconditional models such as DDPM and DDIM are not directly useful here. Further, the latest models such as Imagen, Glide and SD3 are text conditional, so adapting them for this task is not as straightforward. Instead, a recent model from facebook research, called diffusion transformer (DiT), based on the vision transformer (ViT) architecture and trained on the Imagenet dataset for class-conditional generation, is better suited for our task. I chose to fine-tune one of the two largest DiT models, **DiT-XL-2-256**, which are available in the huggingface diffusers pipeline.

3.1.2 Data - The Product Type

The suggested dataset, DefectSpectrum, contains images of various product types such as `pill`, `zipper`, `hazelnut`, `metal nut` and `screw`. For each product, there can be a number of defect types (e.g., for `hazelnut`, the defects are `hole`,`crack`,`cut`, ...). There is a set of images (typically 20) of each product-defect combination. The present task is to choose any one product type and then treat the defect types as classes for class-conditional generation. I chose to experiment with `hazelnut` and `screw`. One of the main reasons behind this particular choise is that there are similar class labels in Imagenet (`buckeye` and `screw`, respectively), and so the learned class embeddings for these classes can perhaps be fine-tuned easily.

4 Experiments

4.1 Fine Tuning

4.1.1 Choosing Parameters

I first implemented a finetuning function using the diffusers implementation of DiTs and the pre-trained weights of DiT-XL-2-256. However, this required a lot of GPU memory, often running into OOM errors. Also, given the small amount of tuning data, full fine-tuning wouldn't be feasible.

I went through some of the parameter efficient fine-tuning methods and found that BitFit and DiffFit were the most suited for this task. BitFit was originally proposed for fine-tuning language models, and the idea of tuning just the biases was extended in DiffFit, where, in addition to the biases, they also tune the kayer norm weights, class embeddings, and learn new scale parameters for each DiT block.

Adapting from DiffFit, **I selected all the biases and layernorm weights for tuning, without adding new scale parameters. In the huggingface**

implementation, the layernorm includes class label embeddings too.

Tuning the class embeddings is perhaps the most important thing since the association of the class labels with Imagenet classes must now be changed to reflect the new classes. **I chose those product types for which a similar class label was available in Imagenet** - for example, screw. And I started the class IDs at the index of this label. My hypothesis is that **in this case, the class embeddings can to be tuned relatively easily.**

4.2 Choosing Hyperparameters

1. **Batch Size** - I used very small batch sizes (tried 1, 2, 4, and 8) to avoid OOM errors and also because the dataset is quite small. With larger batch sizes, I slightly increased the learning rate. However, smaller batch sizes worked better. In particular, batch size 8 resulted in unstable training.

2. **Learning Rate** - I used small learning rate values. I tried $1e-6, 1e-5, 2e-5$ and $3e-5$. Of these, training with $1e-5$ was more stable. With $3e-5$ I got one of the best samples, but the training collapsed after 50 epochs.

3. **Gradient Accumulation** - Since the batch sizes were very small, I used gradient_accumulation_=2. That is, the gradients are accumulated for two steps before applying the weight updates. This gives the benefits of large batches while using small batch sizes.

4. **Noise Scheduler Steps** - The number of of noise scheduler steps in inference (the training steps are fixed. I observed that more steps (> 100) worked better, but only upto an extent (tried 200, 500); 1000 was worse.

5. **Mixed Precision Training** for memory efficiency. Torch automatically applies the optimal precision (autocast), such as fp16 or fp32 to each parameter, allowing mixed precision training.

4.3 Data Preprocessing and Augmentation

1. **Image Size**
   Experimented with 256, 128 and 64. While training with small image size is lighter on the memory, making it possible to use large batches (upto 8), I observed it worked better with 128. This could be because the model was originally trained on 256x256 images.

2. **Normalization**
   Following the normalization scheme in the original paper, I scaled the pixel values in the range $[-1, 1]$.

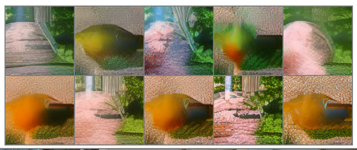
3. **Preprocessing**
   For `hazelnut`, since there was a lot of black background in these images, I cropped the center 800x800 (original images were 1024x1024), and then resized to desired resolution. This was not required for `screw`.

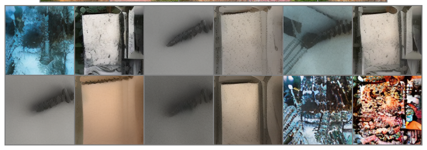
4. **Augmentation**
   I applied the transforms RandomHorizontalFlip, RandomVerticalFlip and RandomRotation (with degree in $[45, 65, 120]$ which is quite different from Flip). These are applied randomly to each image each time, resulting in variation in the data in each epoch, allowing to train for more epochs

5. **Epochs**
   First I trained a model with any new config for 10 epochs, to see if it's working. If yes, then I trained it for 100 epochs. Also tried to implement early stopping (based on just the training loss), but this wasn't working as epected.

4.4 Results

The best generated samples for hazelnut and screw are shown below. Columns correspond to class labels and rows are different samples.





5 Evaluation Metrics

1. Class-wise FID and Inception score

2. Improvement in accuracy of the downstream task model with generated data (various metrics of the downstream model such as accuracy, f1, precision, recall, mAPx% can be used).

6 Further Scope

6.1 Model Improvements

Given more time, following two things can possibly lead to substantially better generations:

1. The huggingface implementation of DiT hides the hyperparameters such as classifier free guidance in the long class hierarchy. While classifier free guidance scale has been implemented, their DiT class does not implement

a "forward_with_cfg" method. Instead, one can work with the original codebase to tune and gain more control over all the hyperparameters

2. A recent paper DriveDitFit [4] proposed a LoRA fine-tuning method for DiT. They propose to replace the class embeddings with closest Imagenet class-embeddings according to a CLIP similarity between corresponding images. This can be used to get good initialization for the class embeddings.

3. Other LoRA approaches of fine-tuning and knowledge distillation.

6.2 Leveraging Extra Information

The DefectSpectrum dataset also includes segmentation masks that demarcate the defects on the product image (e.g. hole in hazelnut). These segmentation masks are images of the same size as the original images. The masks can be used as extra conditional information in addition to the class label and timestep. The recent ControlNet architecture [8] supports conditioning on an image. This could be a very useful guidance for generating small defects, which otherwise could be difficult to generate.

References

[1] Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1–9, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[2] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *European Conference on Computer Vision (ECCV)*, 2022.

[3] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

[4] Jiahang Tu, Wei Ji, Hanbin Zhao, Chao Zhang, Roger Zimmermann, and Hui Qian. Driveditfit: Fine-tuning diffusion transformers for autonomous driving data generation. *ACM Trans. Multimedia Comput. Commun. Appl.*, 21(3), March 2025.

[5] Enze Xie, Lewei Yao, Han Shi, Zhili Liu, Daquan Zhou, Zhaoqiang Liu, Jiawei Li, and Zhenguo Li. Difffit: Unlocking transferability of large diffusion models via simple parameter-efficient fine-tuning. In *2023 IEEE/CVF*

International Conference on Computer Vision (ICCV), pages 4207–4216, 2023.

[6] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. 2022.

[7] Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit H. Bermano, Gal Chechik, and Daniel Cohen-Or. An image is worth one word: Personalizing text-to-image generation using textual inversion, 2022.

[8] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models.