Sandia Laboratories Project-AnomalyGPT

Team SAM

SeungHoon Yoo
Georgia Institute of Technology
syoo97@gatech.edu@gatech.edu

Amit Trikha
Georgia Institute of Technology
amittrikha@gatech.edu

Mukta Bisht
Georgia Institute of Technology
mbisht6@gatech.edu

Abstract

For this study, we utilized the AnomalyGPT model [1] to specialize in detecting anomalies in our custom Artwork (paintings) dataset. Large vision-language models (LVLM) excel in recognizing common objects due to their extensive training data, they often struggle with domain-specific knowledge and fine-grained details within objects. This limitation impedes their effectiveness in domain specific tasks such as Artwork (or painting) Anomaly Detection. So,we investigated adapting the AnomalyGPT model to our custom dataset to tackle the domain specific (i.e artwork/painting) problem. Our approach utilizing the AnomalyGPT models,introduces a methodology leveraging Large Vision-Language Models (LVLMs) to enhance anomaly detection capabilities in artwork settings. By training AnomalyGPT on our specific dataset,we aimed to improve anomaly detection accuracy by integrating domain-specific knowledge directly into the model. This represents a departure from traditional anomaly detection methods by offering a more automated and precise approach to distinguishing anomalies. Rather than relying solely on predefined thresholds,the model learns nuanced features and patterns inherent in anomalies,thereby facilitating more effective anomaly detection without extensive manual intervention. Through this study,we try to demonstrate the feasibility and effectiveness of adapting LVLMs like AnomalyGPT for specialized anomaly detection tasks. As part of our exploration into advanced anomaly detection techniques,we also investigate alternative large vision models beyond AnomalyGPT. Specifically,models such as CLIP (Contrastive Language-Image Pretraining) and AnomalyCLIP were scrutinized for their suitability and efficacy for the artwork dataset. Each model underwent comprehensive training on the dataset,followed by evaluation using established performance metrics to gauge their detection capabilities and overall performance.

1. Introduction

The increasing prevalence of digital imagery in various sectors necessitates robust and reliable anomaly detection systems. Anomaly detection is a critical task in ensuring the integrity and quality of visual data,which can have significant implications across fields such as security, healthcare,and art curation. This project report delves into the development and evaluation of three innovative models—AnomalyGPT, AnomalyCLIP,and CLIP—designed for the classification and detection of anomalies in images. Each model leverages different aspects of advanced machine learning techniques to achieve accurate and efficient anomaly detection.The primary objective of this project was to build and compare the performance of three distinct models for anomaly detection in images.Scope of Images were restricted to 2 classes, Mona Lisa and Girl with pearl earring. Training images were around 50-100 and test images had 34 anomaly images and 16 non anomaly ones.

- **CLIP Model**: Utilizes the power of the CLIP (Contrastive Language–Image Pre-training) model to classify images as either anomaly or non-anomaly.Trained to classify images into anomaly and non-anomaly categories based on their visual content
- **AnomalyCLIP Model**: An advanced iteration of the CLIP model, specifically designed to not only classify images but also detect the presence of anomalies within them.Enhanced to detect the presence of anomalies within the images, providing a more nuanced understanding of the data

- **AnomalyGPT Model**: A sophisticated model that extends the capabilities of AnomalyCLIP by providing descriptive insights into the images and indicating whether an anomaly is present. This model features an interactive user interface similar to ChatGPT,enhancing user interaction and accessibility. Equipped with natural language processing capabilities to describe the images and identify anomalies,making it a versatile tool for detailed image analysis.

2. Data

2.1. Base Dataset

As discussed in the proposal and the midterm report,the team discovered *Best Artworks of All Time* dataset containing over 3,000 images of artworks painted by 50 influential artists,including Claude Monet, Edgar Degas, and Andy Warhol [2]. The team's initial plan was to utilize the entirety of over 3,000 images as "good", non-anomalous data and generate anomalous paintings to test on by artificially adding synthetic anomalies.

2.2. Adding Synthetic Anomalies

The team developed Python codes using the OpenCV library to artificially add anomalies to the original paintings dataset. To ensure variety in resulting anomalies,the team generated anomalies with:

- Different shapes including boxes,lines,letters,circles, and dots.
- Random variations in size,location,and thickness
- Colors that match the overall color scheme and blend more with the artwork

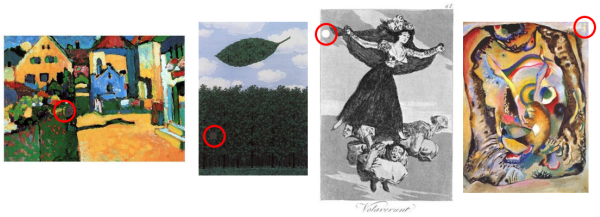

Figure 1. Synthetic Anomaly Images

2.3. Pivoting to *the Mona Lisa* and *the Girl with a Pearl Earring*

As the team deepened the understanding of the anomaly detection models, the team realized that the initial plan to use the entire paintings dataset was not realistic. This is because anomaly detection models need to be trained on numerous "good",non-anomalous images of the same or similar data. For example, AnomalyGPT model is trained on the MVTec-AD dataset which contains many images of cross sections of bottles,cable wires,etc that largely look similar in naked eyes [1]. As each artwork is unique and distinct by nature, no similar artworks exist and therefore no enough images are available for adequate model training. Thus,the team decided to use multiple copies of the same painting for training, but doing so for 3,000 artworks would require excessive compute and storage capacity. As a result,the team decided to use only *the Mona Lisa* and *the Girl with a Pearl Earring*.

In addition to adding synthetic anomalies to the two artworks as described in 2.2,the team also included in the test dataset fake replicas and artistic variations (*artistic reinterpretations*) of the two artworks.

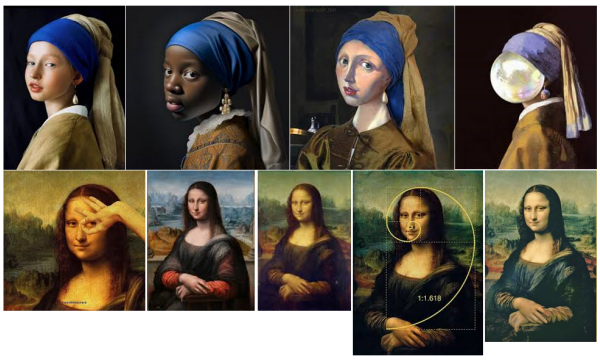

Figure 2. Tested Variations of *the Mona Lisa* and *the Girl with a Pearl Earring*

3. Infrastructure

As a part of this project,we need to work with a large dataset and train anomaly detection models using GPUs. This requires a high amount of RAM and CPU. For this reason, we have leveraged Google Collab Pro. Later,we were also provided Linux Machine from Georgia tech to use.Our infrastructure requirements were below.

- Storage Capacity: 400 GB
- Storage capacity details:ImageBind - 5GBLlama - 20 GB (7B)
- Llama - 55 GB (13B)
- vicuna-7b-delta-v0 - 15 GB
- vicuna-13b-delta-v0 - 30 GB
- Combine Weights 7b - 40 GB (estimate)
- Combine Weights 13b - 90 GB (estimate)
- PandaGPT - 1GB

- AnomalyGPT weights:
  - Unsupervised on MVTec-AD - 225 MB
  - Unsupervised on VisA - 225 MB
  - Supervised on MVTec-AD, VisA, MVTec-LOCO-AD and CrackForest-225 MBDataset
- Artworks - 2GB (compressed)
- MVTec-AD - 5GB (compressed)
- VisA - 2GB (compressed)
- PandaGPT - 15 GB (compressed)

Since this project requires specific infra requirements as above and GPU for training the models(AnomalyGPT and AnomalyCLIP),it makes it challenging to get hold of GPU with considerable amount of compute units without them getting exhausted quickly because of the weights generation and training requiring high amounts of those.Google collab provides GPUs which were bought time to time.The other alternative was the linux machine with GPUs provided by GT,whose setup was easy but it only provided CLIs and no UI like google drive/collab to interact properly. Weight files being very huge (few GBs) would have to be scp to the server.Which will be time consuming because of the size.

4. Models Overview

4.1. AnomalyGPT

AnomalyGPT represents a revolutionary approach in Large Vision-Language Models (LVLMs) for its ability to autonomously identify anomalies in industrial images,marking a significant advancement in Industrial Anomaly Detection (IAD) [1]. The model achieves this without the dependency on manually defined thresholds and offering a efficient approach to anomaly detection in complex industrial environments. Unlike existing IAD methods that offer anomaly scores requiring manual threshold configuration,and conventional LVLMs incapable of image anomaly detection,AnomalyGPT excels in pinpointing anomalies' presence,location,and providing contextual image details. Leveraging a pre-trained image encoder and LVLM. The model aligns IAD images with textual descriptions via simulated anomaly data. This approach includes a lightweight,visual-textual feature-matching image decoder for localization,and employs a prompt learner to enhance LVLM semantic understanding via prompt embeddings,enabling anomaly detection even for novel items with minimal normal sample data.

Large Language Models (LLMs) are now being used for tasks that involve both images and text. AnomalyGPT uses a Llama weights that refer to the parameters utilized in models within the Llama family provided by META. Weight is a fundamental concept in neural networks,including the transformer-based language models like Llama [2]. Weights are the adjustable parameters that the model learns during training,enabling it to capture patterns in the data and perform well on natural language processing (NLP) tasks. The transformer architecture 4,which has become the prevalent design for state-of-the-art language models,organizes these weights into a specific structure named "multi-head self-attention". MiniGPT4 [3] does this by connecting BLIP-2's [4] image part with the Vicuna [5] model through a special layer,refining its performance with lots of image and text data. Similarly,PandaGPT [6] links ImageBind [7] with the Vicuna [5] model to handle different kinds of information together. While these models show promise in handling various types of tasks,they often lack specific knowledge about particular fields because they are trained on broad datasets that cover many topics. To tackle this issue,AnomalyGPT presents a new approach in this research. It uses simulated abnormal data,incorporates advanced techniques for interpreting images, and employs prompt embeddings. AnomalyGPT is designed to detect anomalies in data without needing predefined rules,and it can quickly learn from just a few examples in specific situations.

Figure 3 shows the architecture of AnomalyGPT. When provided a query image $x \in \mathbb{R}^{H \times W \times C}$, the image encoder extracts its final features $F_{\text{img}} \in \mathbb{R}^{C1}$. These features are then passed through a linear layer to create the image embedding $E_{\text{img}} \in \mathbb{R}^{C_{\text{emb}}}$, which is then input into the Large Language Model.

In an unsupervised scenario,the image encoder's intermediate layers extract features from patches,which are then used by the decoder along with text features to pinpoint anomalies at the pixel level. The model operates without labeled training data. The image encoder extracts patch-level features,which are combined with text features in the decoder to generate detailed,pixel-level anomaly maps. This approach relies solely on the intrinsic properties of the data to find anomalies. In a few-shot scenario, features from normal samples' patches are stored in memory banks. To find anomalies,the system calculates the distance between query patches and their closest matches in the memory bank. These localization results are then converted into prompt embeddings using a prompt learner,which becomes part of the input for the large language model (LLM). The LLM uses image input,prompt embeddings,and user-provided textual input to detect anomalies and identify their locations,thus generating responses for the user. The model utilizes a small number of labeled examples to enhance its performance. Normal sample features are stored in memory banks, which serve as a reference for the query image patches. By comparing these patches to the stored features, the model can more accurately pinpoint anomalies. The distances between the query patches and their closest matches

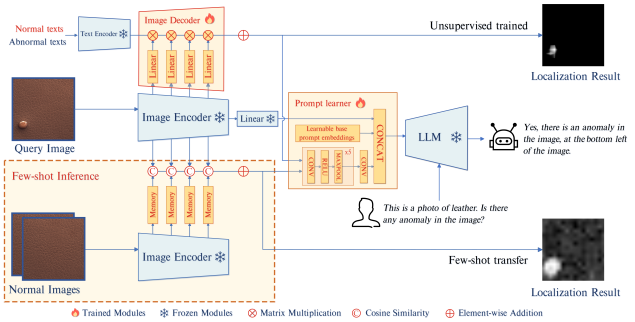

Figure 3. Architecture of AnomalyGPT.

in the memory bank are used to determine the anomaly scores. These scores are then transformed into prompt embeddings by the prompt learner.

The combination of these embeddings with the initial image and any textual input from the user forms the complete input to the LLM. It processes the comprehensive input to detect and localize anomalies within the image and then generates detailed,informative responses for the user,explaining the nature and location of the detected anomalies. This architecture allows it to function effectively in both unsupervised and few-shot learning scenarios.

4.2. CLIP

Additionally,regular vision models like Contrastive Language-Image Pretraining[8] was also tried to see how this model performed on our dataset. CLIP in general,tries to maximize the "cosine similarity" between correct image-caption vector pairs,and minimize the similarity scores between all incorrect pairs. In inference,it calculates the similarity scores between the vector of a single image with a bunch of possible caption vectors,and picks the caption with the highest similarity. Note that CLIP is not a caption generation model,it can only tell you if some existing text caption fits well with an existing image or not. A contrastive function that will modify the weights of the model such that correct image-caption pairs get a high similarity score,and incorrect pairs get low similarity scores.[9] gives a good summary of it. We used CLIP with few different CNN architectures like [10] which is a convolutional neural network (CNN) that's a member of the ResNet (Residual Networks) family of models. It's a supervised learning algorithm that's trained to predict labels or outputs based on input images. ResNet50 is often used for image classification tasks.Loss function like [11] was used. Other algorithms like [12],which detects anomalies using binary trees were used with CLIP. The algorithm has a linear time complexity and a low memory requirement,which works well with high-volume data. It uses decision trees to efficiently isolate anomalies by randomly selecting features and splitting data based on threshold values. This approach is effective in quickly identifying outliers,making it well-suited for large datasets where anomalies are rare and distinct.[13] was another algorithm which was used in conjunction with CLIP,it is an unsupervised anomaly detection method which computes the local density deviation of a given data point with respect to its neighbors. It considers as outliers the samples that have a substantially lower density than their neighbors.[14] tells what happens in the background. We used supervised learning approach with all algorithms providing test data with the actual labels.

4.3. AnomalyCLIP

AnomalyCLIP is an adaption of the CLIP model specifically designed for anomaly detection in images. It is a zero-shot anomaly detection (ZSAD) model that tends to demonstrate an enhanced zero-shot recognition ability compared to CLIP because it is trained to detect and capture generic normality and abnormality rather than focusing on the object semantics such as the foreground objects in the image. For example,in the case of *the Mona Lisa* painting,the regular CLIP model would focus on the women in portrait as the most foreground element. However,the AnomalyCLIP would largely ignore the women and look for general pattern that either fits as normal or deviates as abnormal.[15]

This ability to identify normality and abnormality is a major characteristic of AnomalyCLIP, a ZSAD model. The AnomalyCLIP model is trained using auxiliary data that do not necessarily resemble or are directly related to the target dataset its model is tested and used on. This is an important strength that makes this model more versatile and applicable to various unseen cases. In the paper that first introduced AnomalyCLIP, the regular CLIP model was adapted by fine-tuning with the MVTec AD dataset and its ZSAD performance was tested on non-MVTec AD dataset such as VisA and MPDD. Similarly, in this project, the team leveraged the existing pre-trained AnomalyCLIP model to evaluate its ZSAD performance against the unseen paintings data—*the Mona Lisa and the Girl with a Pearl Earring*.[15]

5. Model Setup and Training

We trained 3 different models, AnomalyGPT, CLIP and AnomalyCLIP for our dataset. This was done to see how different version of anomaly detection models behave and respond to our dataset. All three models were trained on the same dataset comprising original same images of two famous paintings, M̈ona Lisaänd G̈irl with Pearl Earring.̈ However, Anomaly GPT and Anomaly CLIP required additional GPU infrastructure for training due to their complexity, this setup was essential to handle the computational complexity and large datasets required for these models,while the standard CLIP model could be run locally without such requirements. ¡FOR EACH MODEL ADD FEW SENTENCES WHICH WAS DONE SPECIFI-

CALLY FOR THAT MODEL DURING TRAINING AND SETUP¿

5.1. AnomalyGPT

Our implementation for AnomalyGPT begins with replicating the methodology outlined in the original AnomalyGPT research paper, which focuses on utilizing industrial images for effective anomaly detection and then subsequently training the model on a new dataset. The initial step involved securing access to the model weights and integrating them into both the training and inference phases of our project. This process ensures that we leverage the established capabilities of AnomalyGPT in our specific domain of interest that is industrial dataset. Later, we sourced a new Artworks dataset, expanding beyond the industrial context studied in the original paper. Artwork dataset is then augmented with anomalies, introduced manually or through automated procedures (python notebooks), to simulate real-world scenarios where anomalies may manifest in the imagery. The goal was to train the AnomalyGPT model using this new dataset, enabling it to perform anomaly detection tasks within the domain of Artworks.

We ensured the proper environment configuration by installing the required packages listed in the repository's requirements file. We downloaded essential pre-trained image encoders and a Large Language Model (LLM) to align images with their textual descriptions, creating simulated anomaly data. These pre-trained model weights and repositories were obtained, and the scripts were run initially using VS Code to prepare the weights in the necessary Hugging Face and combined formats. The model employs a specialized image decoder that matches visual and textual features to achieve localization results in the images. Additionally, it incorporates a prompt learner that enriches the LLM's understanding by adding detailed meanings and adjusting the LVLM with prompt embeddings. This initial setup ensures that we capture the capabilities of AnomalyGPT within our specific industrial dataset domain. The following steps were undertaken:

- **Setting up the Cloud environments**: To successfully begin running the demo, the first step involved setting up the essential cloud environments. AnomalyGPT relies on multiple model weights and requires substantial storage capacity. Additionally, the loading and training of these weights necessitate powerful GPUs. Therefore,the entire codebase and weights had to be transferred to a cloud storage solution like Google Drive. Google Colab was selected for utilizing GPUs for efficient computation and also for its capability to support complex model training and inference tasks. This provided the necessary computational power without requiring local hardware upgrades, enabling seamless execution of the demo. By leveraging Google Colab, the setup process streamlined access to GPU resources, crucial for handling the computational demands of AnomalyGPT's training and deployment phases. It provided a robust foundation for handling the intensive computational requirements of AnomalyGPT, ensuring smooth operation and efficient utilization of resources throughout the project lifecycle.

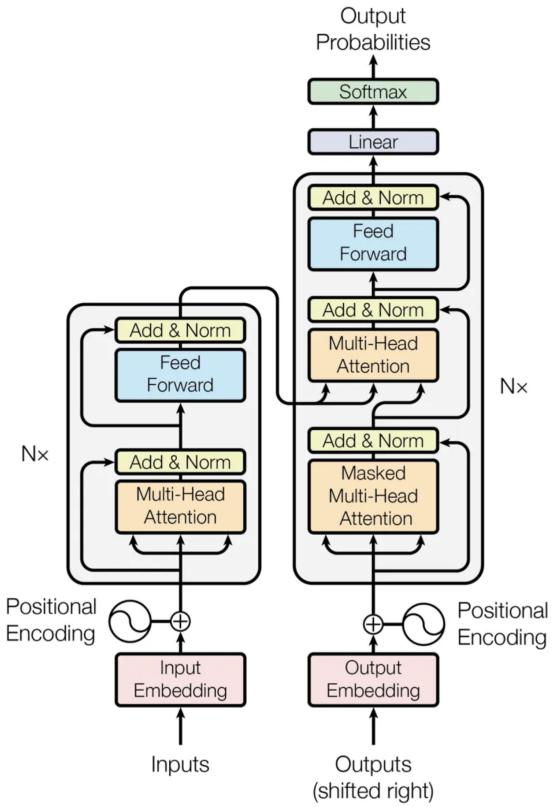

Figure 4. The Transformer Model Architecture

- **Cloning the Repository**: In order to replicate AnomalyGPT, we started by cloning the Anomaly GitHub repository [16]. The repo was loaded to cloud storage and executed from Colab. Additionally, we installed all required packages for the demo as outlined in the repository's requirements.txt file. This GitHub repository encompassed the complete codebase and directory structure essential for running the demo.

- **Downloading and preparing checkpoint weights**: **Prepare ImageBind Checkpoint**: We downloaded the pre-trained ImageBind model [7] and stored it in the specified directory for the Demo. ImageBind is a method designed to create a unified representation across six types of data: images,text,audio,depth, thermal,and IMU data. This method shows that achieving this unified representation doesn't require training on every possible combination of paired data. Training

solely with image-paired data suffices to link all these diverse data types together.

Prepare Vicuna Checkpoint: In AnomalyGPT,the language decoder depends on Vicuna version 0, which needs to be integrated with the LLAMA weights from Meta. Llama weights refer to the parameters utilized in the models within the Llama family based on Transformer Architecture 4. Due to the distribution license of LLaMA,we had to manually restore the Vicuna weights after obtaining approval from Meta. Initially,we restored the 7B version of Vicuna v0. Once we acquired the 7B LLaMA weights,we followed instructions to convert them into the Hugging Face format. Later,we downloaded the delta weights of Vicuna provided by the original authors, specifically using the 7B Vicuna model's delta weights (`lmsys/vicuna-7b-delta-v0`). The final step involved combining these two sets of weights using the Fastchat tool provided by the Vicuna team. We also attempted to use Vicuna-13B (version 0) weights, but encountered irreconcilable errors when trying to combine them with the delta weights. Therefore,we decided to proceed with Vicuna-7B instead.

Prepare Delta Weights for AnomalyGPT: We obtained pre-trained parameters from PandaGPT to initialize our model. Using Vicuna-7B and the `pandagpt_7b_max_len_1024` (`pytorch_model.pt`),we placed these weights in the specified directory.

AnomalyGPT weights: Finally,below weights for different dataset were were downloaded and added in the ./code/ckpt/ directory in the Google Drive

Setup and Datasets	Weights Address
Unsupervised MVTec-AD	train_mvtec
Unsupervised VisA	train_visa
Supervised MVTec-AD, VisA, MVTec-LOCO-AD,CrackForest	train_supervised

Table 1. Weights addresses in AnomalyGPT

- **Running AnomalyGPT DEMO**: After completing the previous tasks,we proceeded to run the demo locally by navigating to the code directory and executing the `web_demo.py` command. Figures 5 showcase images and results from the AnomalyGPT Demo user interface. From the results,it is evident that the application correctly identifies anomalous and non-anomalous images.

- **Training on Artwork Dataset**: After successfully running the demo, the next step was to train the model on a new dataset. For this, we selected the Artwork dataset, focusing on two iconic images: "Mona Lisa" and "Girl with a Pearl Earring." Our goal was to explore how the model performs with art-related data and to investigate its ability to recognize and adapt to various transformations applied to these images. We began by developing a script to generate different transformations of the selected images. These transformations included blurring, inverting, tilting,saturation changes, and other alterations. The purpose was to create a diverse training set that would help the model learn to identify and handle various modifications and distortions of the original artworks.

  Once the transformed images were prepared, we strated working on developing the training scripts. These scripts were specifically developed for the Artwork dataset and were designed to optimize the training process on Google Colab's powerful GPUs,specifically the A100 or L4, depending on availability. The training was conducted over three different durations: 15,20,and 25 epochs. This range allowed us to observe how the model's performance evolved with varying amounts of training. For training,we set a learning rate of 1e-3 and experimented with two batch sizes: 8 and 16. These parameters were chosen based on standard practices and initial experimentation to ensure a balance between training speed and model accuracy. During this training process, only the decoder and prompt learner underwent parameter updates. All other parameters were kept frozen to focus the learning process on these specific components of the model.

- **Training Configurations** The table 2 shows the training hyperparameters used in our experiments. The hyperparameters are selected based on the constraints of our computational resources, i.e. A100 or L4 GPUs Google Colab.

Model	Epochs	Batch	Learning Rate	MaxLength
Vicuna-7B	15	8	1e-3	1024
Vicuna-7B	15	16	1e-3	1024
Vicuna-7B	20	16	1e-3	1024

Table 2. Training hyperparameters used in our experiments

Throughout the training process,we closely monitored the model's performance,particularly its ability to recognize and accurately classify the transformed images. By applying various transformations to the images,we aimed to simulate real-world scenarios where artworks might be subjected to different viewing conditions or alterations. This approach helped in assessing the robustness and adaptability of the model. The Demo application was then updated to use the newly created Artwork training weights. Figures 6, 7 display the out-

comes of various test cases, showcasing the application's performance with the new weights.

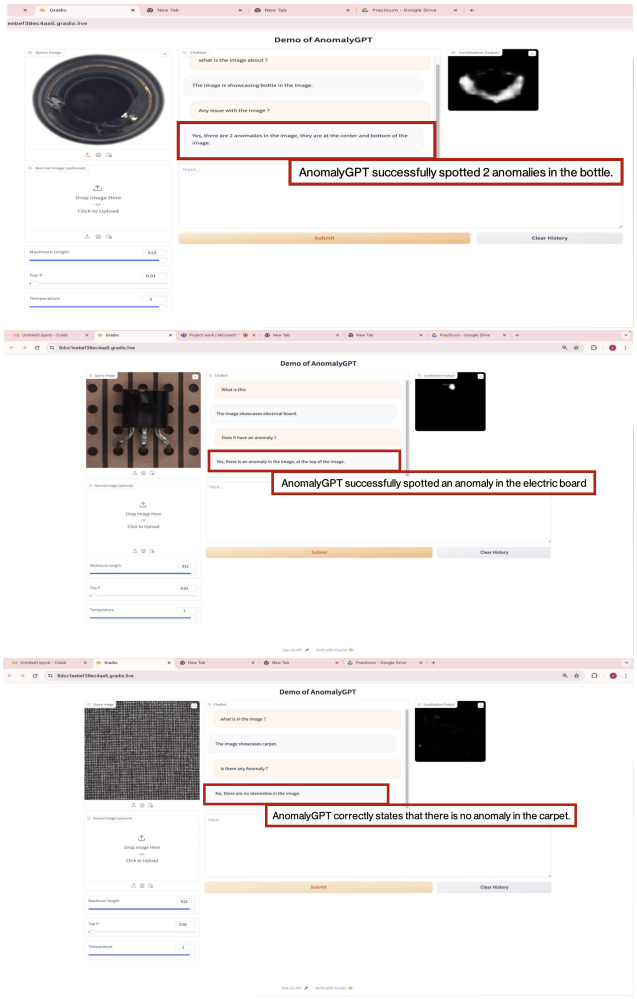

Figure 5. AnomalyGPT Demo UI - Image of Bottle, Transistor with Anomaly and Carpet without Anomaly

5.2. CLIP

For the training purposes, we used 50-100 non anomaly images for model to get trained on.Since our focus has been drilled down to 2 classes mainly ie: Mona Lisa and Girl with pearl earring, we only had these 2 classes for model to learn from. One project has all the python files that would need to be run individually to see the output. Each of them creates some output folder to store the classified images as anomaly vs. non anomaly.ReadME.md has all details on what each file does,input,output and execution instructions in it.We referred [17] and [18] to start with.All the training is mainly done with more than one epoch to let model learn better from non anomaly images for both classes.With algorithms like LOF,Isolation Forest and resnet50 architecture using cross entropy function,main approach was to adjust the parameters like threshold,contamination etc which helps model to determine how to differentiate in order to segregate between anomaly and non anomaly after it calculates cosine similarity.Hence,as we expect some manual tweaks are required for these parameters to adjust them for model to learn better.Out of all algorithms,resnet50 was slowest and was taking around 5-10 min to finish the run.Others took 2 min on an average to finish.

5.3. AnomalyCLIP

As described previously,AnomalyCLIP is a zero-shot anomaly detection model whose strength lies on how the model can successfully generalize to target dataset that is not part of the data used for training the model. Thus,the team decided to use the pre-trained AnomalyCLIP weights without re-training it on the paintings dataset and evaluate if it shows expected ZSAD performance.

However,it is still worthwhile to briefly discuss how the scientists behind the AnomalyCLIP model implemented the model. The AnomalyCLIP team based its model on the publicly available regular CLIP model, specifically VIT-L/14@336px. The model parameters of the CLIP model are kept unchanged during the training of the AnomalyCLIP model. The length of learnable text prompts is set to 12. The learnable token embeddings are attached to the first 9 layers of the text encoder for refining the textual space, and their length in each layer is set to 4. To further fine-tune the model, the scientists used the industrial MVTec AD dataset. It is this AnomalyCLIP model,fine-tuned with the MVTec AD data, that the team used to test its ZSAD performance on *the Mona Lisa* and *the Girl with a Pearl Earring*.[15]

In order to use the pre-trained AnomalyCLIP model,the team followed the instructions in the model developers' GitHub repository: https://github.com/zqhang/AnomalyCLIP. The repository includes all scripts needed to run the model on test images of various industrial anomaly datasets. The team modified the script mvtec.py to ensure it is compatible with the structure of the folder containing normal and abnormal images of *the Mona Lisa* and *the Girl with a Pearl Earring*. The modified python scripts are titled monalisa.py and pearl_earring.py, respectively. Similarly,the original test.sh file was modified to create monalisa_test.sh and pearl_earring_test.sh shell files.

6. Model Testing

All the three models were tested using same test images which comprised 34 anomaly and 16 non anomaly images. Below is the brief summary of testing for each model.

6.1. AnomalyGPT

To test our system,we implemented an anomaly detection mechanism for images of artworks using a pre-trained model,OpenLLAMAPEFTModel. This model leverages few-shot learning to enhance detection capabilities with limited examples. We initialized the OpenLLAMAPEFTModel with specified parameters and loaded pre-trained checkpoints from designated paths. The model was then set to evaluation mode and moved to the GPU.

The prediction function generates responses and anomaly maps for input images using the model. It constructs a prompt based on the input and the history of previous interactions. The predicted anomaly maps are compared with ground truth masks to calculate pixel-level and image-level anomaly detection metrics.

- **Precision Metrics**:
  Pixel-level AUROC (Area Under the Receiver Operating Characteristic Curve): This measures the model's ability to distinguish between normal and anomalous pixels across the entire image.

  Accuracy: calculates the percentage of correct predictions out of the total predictions made by the model for each class.

  For each class, we captured the number of correct and incorrect predictions, pixel-level AUROC and finally computes the average pixel-level AUROC and accuracy across classes.

Paintings	Pixel-Level AUROC	Accuracy
Mona Lisa	62.3	61.2
Pearl Earring	59.7	58.2

Table 3. Artwork Performance For AnomalyGPT

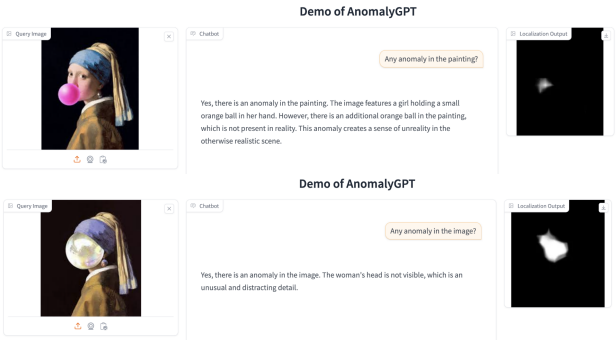

Figure 6. Artwork Trained Anomaly Detection: Girl with a Pearl Earring

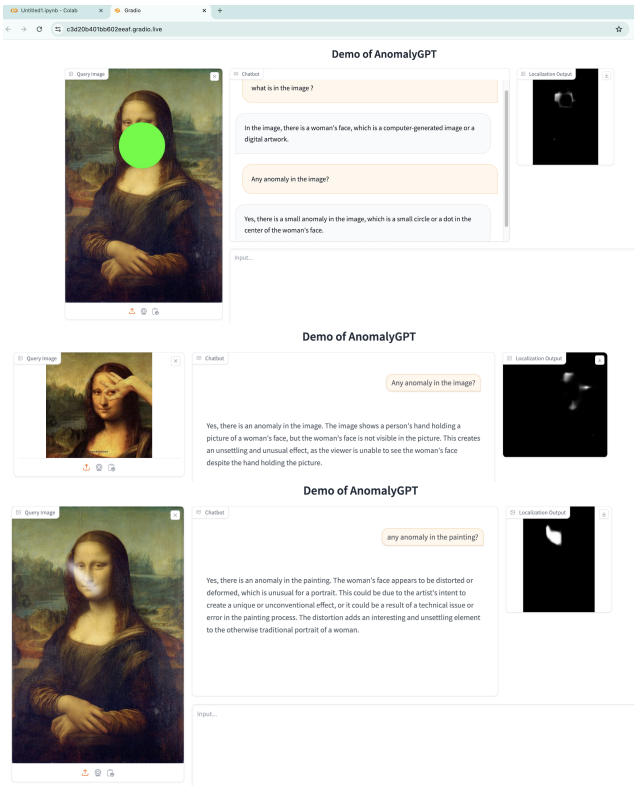

Figure 7. Artwork Trained Anomaly Detection - Mona Lisa

6.2. CLIP

Test dataset consisted of 34 anomaly images and 16 non anomaly images for both classes. The structure for test dataset was set up differently as compared to train, where it had only folders for each class with clean images. Here,we had 2 folders for each label,anomaly and non_anomaly with each having both classes inside them. This is essential for model to the real labels in order to calculate the accuracy against test data.The test data consists of diverse anomalies from both classes. Once the model runs on few epochs, it matches checks the threshold value vs score it got,vs similarity score etc. Since each algorithm we used with CLIP, logically works different,the classification depends on their own logic. Below is the brief summary of each.

- **CLIP(CrossEntropy loss function)**: This loss function commonly used for classification tasks, measures the difference between two probability distributions: the predicted probability distribution (output by the model) and the actual distribution (true labels). It quantifies how much the predicted probabilities diverge from the true labels. It calculates the loss for a single instance as -log(P(y)),where P(y) is the predicted probability for the true class y. In training neural networks,this loss is minimized using optimiza-

tion techniques like gradient descent. This minimization process adjusts the model weights to improve prediction accuracy. In our model code,anomaly detection is performed using the CLIP model and a custom classifier. Once the CLIP model is loaded,and the device (CPU or GPU) is set up,features extracted,transformations are done to preprocess the images (resize and convert to tensors). Image datasets are created from the specified directories,and dataloaders are set up to iterate through the data in batches. A custom classifier is defined,consisting of two fully connected layers with ReLU activation. Optimizer used is Adam for optimizing the classifier's weights. A custom function generates synthetic anomalies by applying random transformations (color jitter and Gaussian blur) to the training images. The classifier is trained on the combined dataset of original non-anomaly images and synthetic anomalies: Overall,It extracts features using the CLIP model,Labels are created (0 for non-anomaly and 1 for synthetic anomaly),loss is calculated using CrossEntropyLoss. The model is optimized using backpropagation. Predictions are made using the classifier.Accuracy is calculated by comparing predictions with true labels. Images are saved into respective output directories based on the predictions (anomaly or non-anomaly). The model was able to correctly classify 25 non anomaly images as non anomaly but 8 were misclassified as non anomaly. All 16 non anomaly images were correctly classified as non anomaly. Overall it was able to achieve,83.67% accuracy.

- **CLIP(resnet50)**: ResNet-50 is a deep convolutional neural network that is 50 layers deep. It is part of the ResNet (Residual Networks) family,which introduced the concept of residual learning to tackle the problem of vanishing gradients in very deep networks. Our code uses a combination of a pre-trained ResNet-50 model(to extract features) and the CLIP model (to get the image-text embedding) to classify test images as either "anomaly" or "non-anomaly." The classification is based on the cosine similarity between the test image embeddings and the embeddings of the original paintings ("Mona Lisa" and "Girl with a Pearl Earring"). The classification results are saved in separate folders for anomalies and non-anomalies. The model was able to correctly classify all non anomaly images as non anomaly and all anomaly images were correctly classified as anomaly. Overall it was able to achieve,100% accuracy.This model used the check where similarity measure has to be greater than equal to 1 to be classified as non anomaly.

- **CLIP(iso forest)**: IsoForest is an anomaly detection algorithm that is particularly well-suited for high-dimensional datasets. It works by isolating observations in a dataset. The basic premise is that anomalies are few and different,thus they are more susceptible to isolation. The algorithm creates multiple decision trees, known as isolation trees or iTrees,by randomly selecting a feature and then randomly selecting a split value between the minimum and maximum values of the selected feature. This process of randomly selecting features and split values continues recursively until each data point is isolated or the tree reaches a predefined height. The number of splits required to isolate a data point is termed as the path length. Anomalies,being few and different,are more likely to be isolated sooner (i.e, in fewer splits) compared to normal data points. For instance,if a data point is isolated by splitting on one or two features,it is likely an anomaly. The anomaly score is based on the average path length from all the trees. The shorter the average path length,the more anomalous the point is considered. The anomaly score for a point is computed as:(x,n)=2$\hat{}$(E(h(x)c(n)), where E(h(x)) is the average path length for point x and (n).(n) is a normalization factor representing the average path length of an unsuccessful search in a Binary Search Tree. A threshold is set for the anomaly score. Points with scores above the threshold are classified as anomalies. Raw pixel values of images are high-dimensional and may not be directly useful for anomaly detection. Instead, meaningful features are extracted using a pre-trained model like CLIP. The extracted features represent the image in a lower-dimensional space that captures the important characteristics of the image. These features are used to train the Isolation Forest model.In our code,it checks the anomaly score to the threshold , if its less than threshold then its anomaly else not. With just 10th percentile of npṗercentile(anomaly_scores 10),it is not able to pick up all anomalies we have in the test images. When set to 60 it performed much better. Also contamination part was updated from 0.2 to 0.4. This is all done to pick up all anomalies that can reside in any part of test image. The model was able to classify all non anomaly images correctly but 4 images were misclassified as non anomaly when they were actually anomalies.Overall it was able to achieve,91.11% accuracy.

- **CLIP(LOF)**: The Local Outlier Factor (LOF) algorithm is an unsupervised anomaly detection method. It identifies anomalies based on the local density deviation of a data point compared to its neighbors. Local density calculation(LDC)For each data point,the algorithm calculates the local density by considering the distance to its nearest neighbors.Local Reachability Density (LRD) is the reachability distance of a point A

with respect to point B is the maximum of the distance between A and B and the distance to the k-th nearest neighbor of B. LRD of a point is the inverse of the average reachability distance from its neighbors. The density is estimated based on the inverse of the average distance to the k-nearest neighbors. The LOF score for a point is the average ratio of the LRD of its neighbors to its own LRD. A higher LOF score indicates a greater likelihood of being an outlier,as it suggests that the point has a significantly lower density compared to its neighbors. A threshold is set for the LOF score to classify points as anomalies. Points with scores above this threshold are considered anomalies. In our code,LOF is used to detect anomalies in image features extracted by the CLIP model. The CLIP model provides a meaningful representation of the images,and LOF identifies images that deviate significantly from the normal patterns in these representations. The training dataset is augmented with random horizontal flips,rotations, and color jitter,followed by the preprocessing steps required by the CLIP model.The test dataset is only preprocessed without augmentation. Data loaders are created for the training and test datasets with batch sizes of 32. The feature extraction function iterates over the data loader, passes the images through the CLIP model to extract features,and collects these features and their corresponding labels. A Local Outlier Factor (LOF) model is initialized and trained on the extracted training features.n_neighbors is set to 20, novelty to True (allowing the model to be used for prediction),and contamination to 0.45 (indicating the expected proportion of anomalies in the dataset). True labels for the test dataset are created based on the presence of the word 'anomaly' in the file paths. The LOF model predicts anomalies in the test features.The predictions are converted to binary labels (1 for anomaly, 0 for non-anomaly). Accuracy is calculated and results are are copied into appropriate directories (anomaly or non-anomaly) based on the predictions. The model was able to classify all non anomaly images correctly but only 1 image was misclassified as non anomaly when they were actually anomalies. Overall it was able to achieve,97.62% accuracy.

6.3. AnomalyCLIP

The team tested the AnomalyCLIP model on the Mona Lisa and the Girl with a Pearl Earring. The scripts included in the original AnomalyCLIP repository return the Area Under the Receiver Operating Characteristic Curve (AUROC) as the performance metrics. We report these results here along with an additional metric, the model accuracy.

The metrics above show combined results of all types of various anomalies ranging from synthetic dots, boxes, lines, and letters to reinterpretations of the artworks. Reinterpretations,shown in Figure 2, are much more subtle and nuanced than synthetic anomalies in Figure 1. And this seems to have affected the performances of the AnomalyCLIP model in capturing anomalies. While the model was able to detect many of the synthetic anomalies, it was highly likely to fail to detect subtle variations in the artworks, often identifying normal regions of the image as abnormal. In the case of the Girl with a Pearl Earring, the model detected the pearl earring itself as anomalous,similar to how it detected circles as anomalous. Also,even amongst the synthetic anomalies, the ones that have colors and sizes that are less visible and obvious against background and surrounding regions (e.g. brown boxes located in the Mona Lisa's dark silk dress) were less likely to be detected by the model.

Paintings	**AUROC**	**Accuracy**
Mona Lisa	70.5	75.6
Pearl Earring	76.4	66.7

Table 4. ZSAD Performance of AnomalyCLIP

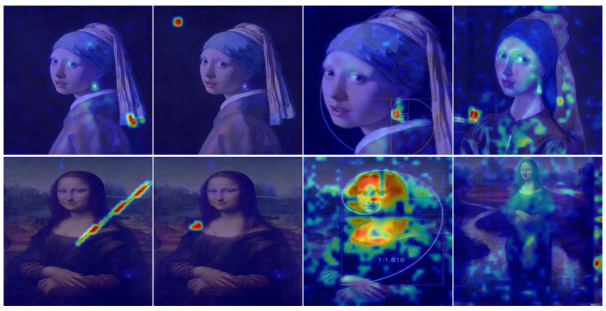

Figure 8. Results of Anomaly Detection by AnomalyCLIP

Also,it is important to note that AnomalyCLIP made false detections, even in images for which the model correctly detected anomalies. Although these instances were considered detections in the accuracy measure, addressing the false positives would be a challenging obstacle in using the AnomalyCLIP model for more sophisticated scenarios.

7. Challenges

Our dataset comprises of images that have paintings from 2 classes, while trained images did have same original image to be trained on for model to learn the correct non anomaly image,the test dataset comprised of various variations of anomalies which were not limited to just one type , rather random and different. Since the anomaly could be present anywhere in the image and could be anything,it was challenging overall for the models to get trained which could result in a good accuracy as well as their detection. While some models like CLIP were able to classify them, but that would have risk of getting overfit and also require

manual intervention for tweaking hyperparameters used in order to classify images better. Below is the brief summary of challenges faced for each.

7.1. AnomalyGPT

- **Infrastructure set-up**: We subscribed for Google Colab Pro as it provides GPUs needed for training AnomalyGPT and running the web demo. However,this gives only a limited number of compute runtime and can exhaust easily. Also,Google Drive automatically stores the weights generated and other files created during the set-up process but can stop if there is not enough RAM left for storage. Once mounted to Google Colab,files in Google Drive only stays for 12 hours,which means some tasks need to be repeated to amend for lost jobs.

- **Large Weight Files**: The weights for pretrained checkpoints like ImageBind Checkpoint,Vicuna,and delta are large (4GB files). During the generation of these files,we had technical errors related to version requirements and dependencies. Some had to be upgraded/changed to a new version. Some errors pertained to OS library,zip directory,and pickle error. Also,while running the AnomalyGPT demo,we had to re-upload the weights for pandasGPT.

- **13B Weights**: We attempted to utilize the 13B weights instead of the 7B but encountered multiple errors when combining them with the Vicuna delta weights. Also,13B weights required a considerably larger amount of space and computing power compared to 7B,adding to the challenges encountered during testing.

- **Vast Training Times**: Training the AnomalyGPT model requires vast training times due to several factors. The model employs a complex architecture that integrates pre-trained components like OpenLLAMAPEFT and image processing layers,that needs heavy computation. The training process involves large datasets with high-resolution images and that further increasing computational demands. Fine-tuning the model on specific anomaly detection tasks also requires significant iterations to achieve optimal performance.

7.2. CLIP

There were no challenges in order to prep or set up regular vision model as they can use CPU and also,do not require any different infrastructure set up as our previous mentioned models do. It can be run locally on a VScode or pycharm with required requirements.txt file. Anomalies constituted of marks like circle,dot,line,different color,different face,fuzziness etc. Any minor deviation from original painting should be detected by model. Since most of them required manual tweaks for model to learn more appropriately given the anomalies could have resided anywhere in the whole painting. We had to try different threshold,contamination percentage etc. to achieve a better accuracy. Model was not introduced to any of the anomaly images rather only clean images. Anomalies were diverse and given the nature of our dataset,it was not straightforward to train the model to detect any anomaly which it was not introduced before. With all the regular vision model CLIP used along with algorithms mentioned highly depended on setting threshold,contamination parameter etc. There is no dynamic thresholding,rather static. Which means the model has to be set in such a way that it is able to gauge the anomalies anywhere in the image. This requires manual intervention. Which can also challenge on the generalization part of model. Once tested on given test images for anomalies,it may or may not perform that well for new anomalies if the new image is drastically different from what it was trained and tested on earlier.Which also means, shifting the threshold and other parameters. This kind of model would easily get overfit and may not generalize well for diverse images.

7.3. AnomalyCLIP

Infrastructure set-up: Similar to AnomalyGPT,GPU is needed to run the AnomalyCLIP model. Thus,we subscribed for Google Golab Pro. However,this gives only a limited number of compute runtime and can exhaust easily. This would be a challenge if we were to continue to amend and improve the model for repeat uses.

Performance on subtle anomalies: As noted in 6.3, AnomalyCLIP did not show a strong performance in detecting subtle and nuanced anomalies. Also,the model made false detections even in images for which it correctly detected anomalies. This would be a major challenge to overcome in order for this model to be used widely.

Sourcing data for fine-tuning: Although AnomalyCLIP model's main strength is its zero-shot anomaly detection ability that generalizes to unseen dataset, the model can benefit from fine-tuning with images from the same or related domains However,obtaining such relevant images is a challenge at times due to data privacy and security concerns. The scientists behind the AnomalyCLIP model also faced this challenge when applying the model to medical images. The team explored various options and eventually addressed the issue by combining multiple medical datasets from different sources.

8. Comparison between models

The table 5 offers a detailed comparison of accuracy scores for different models tested on a dataset,

specifically highlighting the performance of the CLIP, AnomalyCLIP,and AnomalyGPT models with various algorithms. The CLIP model is evaluated using ResNet50,CrossEntropy,Isolation Forest,and Local Outlier Factor (LOF) algorithms, achieving the highest accuracy score of 100 with ResNet50. AnomalyCLIP model scores an accuracy of 71.11, suggesting it is less effective than the best-performing CLIP configurations. AnomalyGPT records an even lower accuracy score of 59.7 indicating a notable performance gap between it and the other models. The substantial differences in accuracy scores between AnomalyGPT and the top-performing CLIP model underscore the varying effectiveness of different models and algorithms on the test dataset. The test results highlight the complexity of the models and the challenges in training them on datasets. Capturing intricate details proves to be particularly difficult. This complexity arises from the sophisticated nature of the models, requiring extensive computational resources and fine-tuning to achieve accurate results. The training process involves navigating various parameters and configurations to optimize performance,making it a daunting task. Additionally,the models must be capable of recognizing subtle patterns and nuances within the data,which adds another layer of difficulty.

Model	Algorithm	Accuracy score
CLIP	resnet50	100
	crossentropy	83.67
	iso forest	91.11
	LOF	97.62
AnomalyCLIP	-	71.11
AnomalyGPT	-	59.7

Table 5. Accuracy Score Comparison for Different Models on Test Dataset

9. Learning and Future enhancements

9.1. Key Learnings

- **Model Performance and Infrastructure**: One of the most significant learnings from this project was the critical role of infrastructure in model training and deployment. Given the computational demands of models like AnomalyCLIP, AnomalyGPT,GPU availability is essential for efficient training and inference. We encountered several infrastructure issues that highlighted the importance of robust hardware setups and efficient resource management.

- **Parameter Tuning**: The CLIP model,while powerful, requires careful parameter tuning to optimize performance. This manual parameter setting process can be time-consuming and prone to errors,potentially leading to overfitting if not managed properly. This challenge underscored the need for automated parameter optimization techniques to streamline model training. AnomalyGPT and AnomalyCLIP work on huge number of parameters and in our case it was 7B.Not only it will be tricky to manually update the parameters but also time consuming. The models used for these were pretrained on these parameters and weights generated were huge files.

- **Model Overfitting**: The tendency of the CLIP model to overfit on the training dataset was a notable drawback. This overfitting occurs when the model learns the training data too well,including its noise and outliers,at the expense of generalizing to new,unseen data. Identifying and mitigating overfitting through techniques such as cross-validation and regularization is crucial for developing robust models. AnomalyGPT and AnomalyCLIP are not prone to overfitting since they work on huge number of parameters but can take a lot of time to fine tune them before reaching the desired behaviour.

- **Diverse Test Data**: The performance of our models can vary significantly with more diverse and complex test images. While the models performed well on the given dataset,their accuracy and reliability may decrease when exposed to a broader range of anomalies. This observation emphasizes the importance of comprehensive testing on diverse datasets to ensure model generalization and robustness.

9.2. Enhancements and Future Work

- **Infrastructure Improvements**: Investing in more reliable and scalable GPU infrastructure would significantly enhance the efficiency and effectiveness of model training and deployment. Exploring cloud-based solutions could provide the necessary computational power while maintaining flexibility and scalability.

- **Automated Parameter Optimization**: Implementing automated parameter tuning methods, such as grid search,random search,or more advanced techniques like Bayesian optimization,can reduce the manual effort required and help find optimal settings more efficiently. This automation would also help mitigate the risk of overfitting.

- **Regularization Techniques**: To address overfitting, incorporating regularization techniques such as dropout,weight decay, and data augmentation can improve model generalization. Additionally, implementing early stopping based on validation performance can prevent the model from overfitting during training.

- **Expanded Dataset**: To enhance the model's ability to handle diverse and complex anomalies,expanding the dataset to include a wider variety of images would be beneficial. This expansion could involve collecting more images with different types of anomalies and non-anomalies across various contexts.
- **Continuous Evaluation and Feedback Loop**: Establishing a continuous evaluation and feedback loop where the model is regularly tested on new data and fine-tuned based on performance metrics can ensure ongoing improvement and adaptation to new types of anomalies.
- **Integration with User Feedback**: For the AnomalyGPT model, integrating user feedback mechanisms where users can provide corrections or confirmations of the model's predictions can help in continuously refining and improving the model's accuracy and reliability.

By addressing these areas,we can enhance the robustness, accuracy,and generalizability of our anomaly detection models,making them more effective tools for real-world applications.

10. Conclusion

We worked on 3 different versions of anomaly detection models with different settings and configurations and algorithms. Each worked differently and provided different classification results depending on the hyperparameter tuning done for each. With 3 different models we have tried to compare the behaviour,accuracy.Each model was adding a new capability,with CLIP being a regular vision model which just classified the paintings versus,anomaly CLIP highlighting anomalies in those as well and in the end,anomalyGPT providing a chatbot approach to not only detect anomalies but also describe the image and interact with the user. We also feel that it might largely depend on the dataset as well which in turn decides which algorithm/model approach would eventually be best for that. Here in our case,we had different variations of anomalies tested and each model provided different output based on how they work. It also depends what is the need of consumer which ultimately decides what model would be best for that scenario,keeping its requirements/challenges in mind as discussed in report.

References

[1] Z. Gu, B. Zhu, G. Zhu, Y. Chen, M. Tang, and J. Wang, "Anomalygpt: Detecting industrial anomalies using large vision-language models," 2023. [Online]. Available: https://arxiv.org/abs/2308.15366 1, 2, 3

[2] M. Authors, "Llama weights: An ultimate guide 2024," 2024, llama Weights: An Ultimate Guide 2024. 2, 3

[3] D. Zhu, J. Chen, X. Shen, X. Li, and M. Elhoseiny, "Minigpt-4: Enhancing vision-language understanding with advanced large language models," 2023. [Online]. Available: https://arxiv.org/abs/2304.10592 3

[4] J. Li, D. Li, S. Savarese, and S. Hoi, "Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models," 2023. [Online]. Available: https://arxiv.org/abs/2301.12597 3

[5] W.-L. Chiang, Z. Li, Z. Lin, Y. Sheng, Z. Wu, H. Zhang, L. Zheng, S. Zhuang, Y. Zhuang, J. E. Gonzalez, I. Stoica, and E. P. Xing, "Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality," March 2023. [Online]. Available: https://lmsys.org/blog/2023-03-30-vicuna/ 3

[6] Y. Su, T. Lan, H. Li, J. Xu, Y. Wang, and D. Cai, "Pandagpt: One model to instruction-follow them all," 2023. [Online]. Available: https://arxiv.org/abs/2305.16355 3

[7] R. Girdhar, A. El-Nouby, Z. Liu, M. Singh, K. V. Alwala, A. Joulin, and I. Misra, "Imagebind: One embedding space to bind them all," 2023. [Online]. Available: https://arxiv.org/abs/2305.05665 3, 5

[8] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," 2021. [Online]. Available: https://arxiv.org/abs/2103.00020 4

[9] J. Boschman, "Clip paper explained easily in 3 levels of detail," https://medium.com/one-minute-machine-learning/clip-paper-explained-easily-in-3-levels-of-detail-61959814ad13, 2023, cLIP Summary. 4

[10] P. Potrimba, "What is resnet-50?" https://blog.roboflow.com/what-is-resnet-50/, 2024, resNet50. 4

[11] P. Subedi, "Clip by openai explained," https://medium.com/@pragyansubedi/clip-by-openai-explained-1e4c38644356, 2023, crossentropy. 4

[12] A. 416, "Anomaly detection using isolation forest – a complete guide," https://www.analyticsvidhya.com/blog/2021/07/anomaly-detection-using-isolation-forest-a-complete-guide, 2024, iso forest. 4

[13] P. kumar, "Understanding lof (local outlier factor) —perspective for implementation," https://medium.com/@pramodch/understanding-lof-local-outlier-factor-for-implementation-1f6d4ff13ab9, 2020, lOF. 4

[14] V. Jayaswal, "Local outlier factor (lof) — algorithm for outlier identification," https://towardsdatascience.com/local-outlier-factor-lof-algorithm-for-outlier-identification-8efb887d9843, 2020, lOF Maths. 4

[15] Q. Zhou, G. Pang, Y. Tian, S. He, and J. Chen, "Anomalyclip: Object-agnostic prompt learning for zero-shot anomaly detection," 2024. [Online]. Available: https://arxiv.org/abs/2310.18961 4, 7

[16] A. Authors, "Anomalygtp github code repository," https://github.com/CASIA-IVA-Lab/AnomalyGPT.git, 2023, anomalygtp Github Code Repository. 5

[17] E. Betzalel, "image outlier detection," https://github.com/eyalbetzalel/image_outlier_detection.git, 2022, code for CLIP model. 7

[18] J. Wook, "Clip," https://github.com/openai/CLIP.git, 2021, code for CLIP model. 7

[19] ICARO, "Best artworks of all time," https://www.kaggle.com/datasets/ikarus777/best-artworks-of-all-time, 2019, accessed: 2024-05-23.