Team 91
From Pixels to Proof: Leveraging Vision and Language to Detect AI-Generated Artifacts

Abstract

This work addresses the problem of detecting and analyzing AI-generated images, with a focus on identifying artifacts and explaining their origins. The task is divided into two primary sub-tasks: (1) detecting AI-generated artifacts and (2) providing detailed explanations for the identified artifacts. For the first task, we utilize the RINE with TinyCLIP model to detect inconsistencies and anomalies in AI-generated images. The TinyCLIP model, fine-tuned on 32x32 images, performs exceptionally well in identifying subtle artifacts such as texture irregularities, lighting inconsistencies, and unrealistic reflections.

Task 2 extends this analysis by not only detecting the presence of AI-generated artifacts but also providing insightful explanations for why these artifacts occur. For this task, the LLaVA (7B)[] model was used, which, despite challenges related to the low resolution of 32x32 images, successfully interprets noisy visual data and generates clear, coherent explanations. By integrating natural language processing capabilities, LLaVA enables the model to describe and articulate the reasons behind the visual anomalies, linking them to potential flaws in the image generation process.

The use of 32x32 images in both tasks presents a unique challenge, as the low resolution limits the model's ability to detect fine-grained details. However, this trade-off was necessary to reduce computational complexity and expedite training. Additionally, the synthetic SynArtifact dataset, generated using ChatGPT, was employed to supplement the training data due to the absence of real-world high-quality datasets. Despite these challenges, the models demonstrated promising results in both artifact detection and explanation, providing valuable insights into the nature of AI-generated images.

Overall, the integration of RINE with TinyCLIP for artifact detection and LLaVA for explanation generation presents a comprehensive approach to understanding and analyzing AI-generated images, with the potential for significant improvements in both model accuracy and interpretability in future iterations.

Keywords

Vision-Language Model, Adversarial robustness, Image manipulation, Explainability, Digital transparency, Generative AI

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

ACM Reference Format:
. 2024. Team 91 From Pixels to Proof: Leveraging Vision and Language to Detect AI-Generated Artifacts . In . ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

The rise of AI-generated images has led to significant advancements in creative and artistic domains, as well as practical applications in media, entertainment, and design. However, the increasing realism of these images also raises concerns regarding their authenticity, posing challenges in fields such as security, forensics, and misinformation detection. Detecting AI-generated images, particularly identifying subtle artifacts or inconsistencies, is a crucial task in distinguishing them from real images. In this work, we address two primary objectives related to AI-generated image detection: (1) detecting artifacts and (2) explaining the nature of these artifacts.

To achieve these objectives, we propose a novel approach that combines state-of-the-art models in computer vision and natural language processing. For artifact detection, we leverage TinyCLIP, a robust model trained with adversarial techniques. The model, fine-tuned on 32x32 resolution images, effectively identifies inconsistencies such as texture irregularities, lighting distortions, and unrealistic reflections, which are indicative of AI generation.

Building on the artifact detection capability, the second task explaining why an image is AI-generated—extends the analysis by providing clear, interpretable explanations of the identified artifacts. For this task, we employ the LLaVA (7B)[5] language model, which integrates natural language processing to generate detailed narratives about the possible causes behind each artifact. Despite challenges posed by the low resolution of 32x32 images, LLaVA is capable of converting noisy, artifact-laden visual data into human-readable explanations, enhancing the model's interpretability.

The work also introduces challenges such as the limited resolution of 32x32 images, which inherently makes it harder to detect fine-grained details. Furthermore, the absence of high-quality, real-world datasets necessitated the creation of a synthetic dataset, SynArtifact, annotated using the ChatGPT API. This synthetic data, while useful, has limitations in realism and variety when compared to real-world AI-generated images.

This paper presents a comprehensive framework for detecting and explaining AI-generated artifacts, showcasing the potential of combining computer vision and natural language processing models to provide a deeper understanding of AI-generated content. The outcomes of this work contribute to advancing research in AI authenticity detection and can have significant implications for future advancements in both detection accuracy and model interpretability.

2 Task 1 - Real/Fake Image Detection

2.1 Key Challenges

In tackling the problem of real/fake AI detection using 32x32 images, several key challenges were encountered. The most significant challenge stemmed from the limited spatial resolution of the images, which restricted the amount of detail available for differentiating between real and fake content. At such a small resolution, subtle features, textures, or anomalies in images that could aid classification became harder to capture, making it difficult for traditional models to discern fine-grained differences. Additionally, the small image size presented difficulties in data augmentation, where typical transformations like rotation or flipping were less impactful on such small images. This necessitated the development of more sophisticated techniques to generate a diverse set of training samples. Another hurdle was that simple CNN architectures were insufficient for achieving good performance across various types of generators, requiring us to explore more complex or customized models. Lastly, the lack of publicly available datasets for real/fake image classification at the 32x32 resolution compounded these challenges. Most existing datasets for this task are focused on larger image sizes, forcing us to either downscale high-resolution datasets or create a custom dataset, adding another layer of complexity to the project.

2.2 Background and Related Work

With the rapid proliferation of generative models like GANs and diffusion models, detecting fake images has become an increasingly critical challenge. A significant body of research has been dedicated to developing methods that can generalize across different types of generative models and reliably distinguish real from fake content.

One prominent study, *Towards Universal Fake Image Detectors*[6] that Generalize Across Generative Models, highlights the limitations of traditional deep learning-based fake image classifiers, which are often tuned to detect specific types of synthetic images, such as those generated by GANs. These models struggle to generalize to newer generative models, such as diffusion-based or autoregressive models. The authors propose a novel approach of performing real-vs-fake classification without explicit training, utilizing feature spaces from pretrained vision-language models like CLIP. Their method, which involves nearest-neighbor and linear probing, demonstrates surprising generalization power. When tested on unseen generative models, the technique outperforms state-of-the-art methods, achieving significant improvements in accuracy. This research underscores the ability of large, pretrained models to generalize well across diverse generative models, a key point for building effective real/fake detection systems.

Another important contribution comes from the paper *Leveraging Representations from Intermediate Encoder-Blocks for Synthetic Image Detection*[4], which focuses on leveraging intermediate layers of the CLIP model for synthetic image detection. CLIP has been shown to excel in cross-modal tasks, but its high-level representations often miss fine-grained details crucial for detecting fake images. This study proposes utilizing intermediate Transformer blocks to extract more detailed image features, improving the model's ability to detect forgeries. Their method shows an average performance improvement of +10.6% over state-of-the-art methods and achieves excellent results with minimal training time. Additionally, the use of the ten-crop validation technique on datasets like Synthbuster—focused on synthetic images—demonstrates that training on diffusion-based data yields substantial performance gains, further reinforcing the effectiveness of CLIP's intermediate layers for robust fake image detection.

In line with leveraging CLIP's capabilities, the *TinyCLIP*[9] paper introduces a novel distillation method to make the CLIP model more efficient and suitable for deployment on edge devices. By utilizing techniques like affinity mimicking and weight inheritance, TinyCLIP reduces the model's size by 50%, while maintaining comparable performance to the original CLIP model. This is particularly valuable for real-time applications on devices with limited computational resources. The paper also demonstrates that TinyCLIP can achieve impressive accuracy in tasks like image classification, surpassing the original CLIP model despite using only a fraction of the parameters. This lightweight model aligns well with our goal of deploying AI detection models on edge devices while harnessing the power of CLIP's knowledge distillation for effective fake image detection.

Lastly, adversarial attacks[1] such as FGSM, PGD, and CW [8]are a critical concern for the robustness of AI models, especially in security-sensitive applications like fake image detection. These attacks involve subtly altering input images to deceive the model into making incorrect predictions. Adversarial training, which involves augmenting the training set with adversarially perturbed examples, has proven to be an effective strategy for improving model robustness. By incorporating these adversarial examples during training, models become better at recognizing and resisting adversarial attacks, ensuring more reliable performance in real-world scenarios. This robustness is especially important for real/fake detection tasks, where the goal is not only to identify synthetic images but also to ensure that the model remains resistant to manipulative attempts designed to bypass detection systems.

2.3 Final Model Architecture

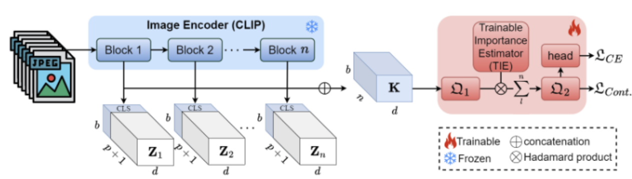

Figure 1: The RINE architecture.

The final model for real/fake AI detection leverages **TinyCLIP**, an adversarially trained version of CLIP, integrated with the architecture of **Intermediate Encoder-Blocks for Synthetic Image Detection (RINE)**. This combination enhances the model's resilience to adversarial attacks while maintaining computational efficiency, making it ideal for deployment on edge devices. The architecture captures both high-level semantic features and fine-grained low-level details, enabling superior generalization across diverse generative models.

2.3.1 Model Overview. The **RINE** architecture is built on the **Vision Transformer (ViT)** variant of **CLIP**. The model processes a batch of images $X \in \mathbb{R}^{b \times 3 \times w \times h}$, where b is the batch size, w is the width, and h is the height of the images. The images are first flattened into patches, with each patch reshaped into a sequence of size p, where $p = \frac{w \times h}{P^2}$ and P is the patch side length. These flattened patches are then linearly projected into d-dimensional embeddings, which are subsequently concatenated with the learnable **CLS token** and enriched with positional embeddings to form the input sequence $Z_0 \in \mathbb{R}^{b \times (p+1) \times d}$.

The architecture uses **Transformer encoder blocks**[7] to process the input sequence through multiple layers, each consisting of **Multi-head Self Attention (MSA)** and **Multi-layer Perceptron (MLP)** blocks with **Layer Normalization (LN)**. After n successive Transformer encoder blocks, the output of each block, denoted as $Z_l \in \mathbb{R}^{b \times (p+1) \times d}$, is used to extract **Representations from Intermediate Encoder-Blocks (RINE)**. Specifically, the **CLS token** from each block is concatenated across all layers to create the representation $K \in \mathbb{R}^{b \times n \times d}$.

2.3.2 Trainable Modules and Feature Processing. Once the **RINE** representation K is obtained, it is passed through a series of **trainable modules** for further processing:

(1) **Projection Network (Q1)**: The concatenated **RINE** features are passed through a projection network Q_1, which consists of several layers. Each layer applies a linear transformation followed by a **ReLU** activation, resulting in a new representation $K_m \in \mathbb{R}^{b \times n \times d'}$, where d' denotes the reduced dimensionality.
(2) **Trainable Importance Estimator (TIE)**: To fine-tune the influence of each feature, a **Trainable Importance Estimator (TIE)** is introduced. This module learns the importance of features at different stages of processing (i.e., different Transformer blocks). It uses a learnable variable $A \in \mathbb{R}^{n \times d'}$ to weight the importance of each feature at the lth Transformer block. The importance-adjusted features are then averaged across blocks to produce a final representation $\tilde{K} \in \mathbb{R}^{b \times d'}$.
(3) **Second Projection Network (Q2)**: The importance-weighted features are passed through a second projection network Q_2, which further refines the feature representation before it is processed by the **classification head**.
(4) **Classification Head**: The classification head consists of two dense layers, each with **ReLU** activation, followed by a final dense layer that produces the logits, representing the model's prediction on whether the image is real or fake. The output is a probability score between 0 and 1, indicating the likelihood of the image being fake.

2.3.3 Objective Function. The model is trained using a combination of two loss functions:

(1) **Binary Cross-Entropy Loss**: This loss directly optimizes the real-vs-fake classification task:

$$L_{CE} = -\sum_{i=1}^{b} \left[y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i) \right]$$

where y_i is the ground truth label (0 for real, 1 for fake) and $\hat{y}_i$ is the predicted probability.
(2) **Supervised Contrastive Loss (SupContrast)**: This loss helps the model by encouraging the separation of features from different classes while pulling together features from the same class:

$$L_{Cont.} = -\sum_{i=1}^{b} \frac{1}{G(i)} \sum_{g \in G(i)} \log \frac{\exp(z_i \cdot z_g / \tau)}{\sum_{a \in A(i)} \exp(z_i \cdot z_a / \tau)}$$

where $G(i)$ is the set of indices with the same class as y_i, and $A(i)$ is the set of all other samples. The temperature parameter τ controls the sharpness of the distribution.

The two loss functions are combined using a tunable factor ξ:

$$L = L_{CE} + \xi \cdot L_{Cont.}$$

2.3.4 Benefits of Using TinyCLIP. **TinyCLIP** offers several advantages, particularly in terms of computational efficiency and deployment feasibility, making it an ideal choice for edge-device applications. Key benefits include:

- **Reduced Latency**: By using the TinyCLIP model, which is a compressed and distilled version of the original CLIP, we achieve significantly reduced inference times. The distillation process retains much of CLIP's performance but reduces the number of parameters, leading to faster model inference.
- **Lower Memory Consumption**: TinyCLIP achieves memory efficiency by reducing the model size by up to 50%, making it suitable for deployment on resource-constrained devices like smartphones and embedded systems. This lower memory footprint ensures that the model can run on devices with limited computational resources without compromising performance.
- **Efficiency in Training**: The use of knowledge distillation, through affinity mimicking and weight inheritance, allows TinyCLIP to train faster compared to models trained from scratch. This translates to significant training speed-ups (up to 7.8x faster), making it feasible to fine-tune the model quickly.
- **High Transferability**: Despite the reduction in model size, TinyCLIP maintains a high level of performance on downstream tasks, such as fake image detection. It achieves impressive zero-shot performance, meaning it can generalize well to new data without requiring retraining on task-specific datasets.
- **Optimized for Edge Devices**: TinyCLIP's small size and fast inference times make it ideal for real-time applications on edge devices. This ensures that fake image detection can be performed efficiently on devices with limited power, enabling real-world deployment of AI-based detection systems.

2.3.5 Conclusion. By combining **TinyCLIP** with the **RINE architecture**, the final model achieves a powerful and efficient solution for synthetic image detection. The use of intermediate encoder-block representations enhances feature extraction, while the addition of the **Trainable Importance Estimator (TIE)** allows for adaptive refinement of feature contributions. This results in a model that not only performs well across various generative models but is also highly efficient, making it suitable for deployment on edge devices. The model's ability to generalize across different synthetic image generators and perform real-time detection on resource-constrained devices marks a significant advancement in the field of real/fake AI image detection.

2.4 Experimental Setup

2.4.1 ***Dataset Generation and Preprocessing.*** For the dataset generation process, we began by curating a diverse set of image datasets to ensure a comprehensive collection for the real/fake AI detection task. The following steps outline how we acquired and processed the data:

- **Latent Diffusion Training Data:** We then downloaded the Latent Diffusion Training dataset, which provided images from another powerful generative model. Latent diffusion models are increasingly popular for their ability to generate high-quality images and were included to further expand the range of synthetic content.
- **ProGAN & GAN-Based Datasets:** We downloaded GAN-based datasets, including the Deepfake, Low-level-vision, and Perceptual loss test sets. These GAN-based datasets were primarily used for testing the model's ability to detect synthetic content. Additionally, the ProGAN dataset provided us with the LSUN real dataset, which consists of real images to serve as the counterpart to the synthetic images for training. GANs are one of the most widely used generative models for image creation, and incorporating these datasets ensured we were addressing the most prevalent types of synthetic images.
- **Synthbuster Dataset**: The next step was to acquire the Synthbuster dataset, which focuses on deepfake and AI-generated content. This dataset is specifically designed for the detection of synthetic media, making it an essential addition to our training data for the real/fake detection task.
- **MSCOCO Dataset:** Finally, we obtained the MSCOCO dataset, which includes a wide variety of real-world images. To match the 32x32 image resolution required for the project, each image was downsampled to 32x32 pixels, ensuring consistency across all images in the dataset.
  After downloading and organizing all the datasets into a structured directory under data/, we proceeded with the preprocessing step of resizing the images to the required 32x32 resolution. The datasets were finalized to include both GAN-based and diffusion-based synthetic images, as these represent the majority of the generative models used worldwide for image creation. With the newly generated dataset ready, we started the training process to develop the real/fake AI detection model. This diverse collection, containing both real and synthetic images from various sources, ensured a well-rounded approach to training and testing the model's ability to generalize across different types of generative techniques.

2.4.2 ***Implementation Details.*** The training process for our model follows a structured approach aimed at optimizing the performance of the TinyCLIP-based architecture under adversarial conditions. We began with two TinyCLIP variants for adversarial fine-tuning: "TinyCLIP-ViT-8M-16-Text-3M-YFCC15M" with a 256-dimensional feature projection and "TinyCLIP-ViT-40M-32-Text-19M-LAION400M" with a 512-dimensional feature projection. Both models were pre-trained and subsequently fine-tuned to enhance their robustness against adversarial attacks. The training was conducted for 1 epoch, with plans to experiment with 3, 5, 10, and 15 epochs to assess potential benefits from further training. A batch size of 64 was used to balance training stability and computational efficiency. The learning rate was set to 1×10^{-4} for the Adam optimizer and was reduced at the 6th epoch using a predefined schedule. The backbones used were "TinyCLIP-ViT-8M-16-Text-3M-YFCC15M" with a projection dimension of 256 and "TinyCLIP-ViT-40M-32-Text-19M-LAION400M" with a projection dimension of 512. Two projection networks were applied with $nprojs = 2$, and the contrastive loss weight ξ was varied across values $\{0.1, 0.2\}$. Data augmentation included random JPEG compression, resizing to 224x224 pixels, and a random horizontal flip with a probability of 0.5. For validation and testing, the images were resized and center-cropped to 224x224 pixels. The experiments were conducted on NVIDIA GPUs using PyTorch and the `torchattacks` library for adversarial attack generation. The model was trained with the Adam optimizer, and the training configuration included a learning rate schedule to reduce the learning rate at the 6th epoch. These settings allowed us to fine-tune TinyCLIP models with adversarial examples and improve their robustness in detecting fake images while maintaining efficiency for deployment on edge devices.

2.4.3 ***Evaluation Protocol:*** We evaluate the proposed architecture with accuracy (ACC) and average precision (AP) metrics on each test dataset, following previous works for comparability purposes. For the calculation of accuracy no calibration is conducted, we consider 0.5 as threshold to all methods. We also report the average (AVG) metric values across the test datasets to obtain summary evaluations.

2.5 Results:

In table 1 and table 2, we present the performance scores (ACC AP respectively) of our models. Table 3 summarizes the Latency of the model on the CPU.

2.6 Discussions:

Training models on 32x32 images presents unique challenges and trade-offs, particularly in terms of model performance, accuracy, and computational efficiency. The reduced image size significantly limits the amount of fine-grained information available to the model, making it more difficult for the network to capture high-level semantic features, especially for complex tasks such as real/fake image detection. This limitation in spatial resolution leads to a decrease in accuracy compared to models trained on higher-resolution images, as finer details critical for distinguishing between real and fake images are often lost. Despite this, the smaller image size offers notable advantages in terms of reduced computational cost, allowing for faster training times and lower memory usage. These benefits are particularly important when deploying models on edge devices with limited resources. However, this efficiency comes at a cost: the trade-off between the reduction in input resolution and the accuracy of feature extraction results in a less robust model when evaluated on more challenging, real-world datasets. Additionally,

Generator	Accuracy (ACC)	Average Precision (AP)
DALL-E 2	62.4	69.7
DALL-E 3	46.4	47.4
Stable Diffusion 1-3	52.6	55.3
Stable Diffusion 1-4	51.4	54.3
Stable Diffusion 2	47.6	48.1
Stable Diffusion XL	48.1	48.8
GLIDE	74.5	83.6
Firefly	57.1	53.8
MidJourney V5	50.7	52.3
ProGAN	49.0	48.7
StyleGAN	51.1	50.3
StyleGAN 2	46.3	45.9
BigGAN	49.7	50.6
CycleGAN	46.2	45.6
StarGAN	45.5	42.1
GauGAN	47.2	47.0
DeepFake	48.0	48.7
SeeingThroughFog	49.7	49.7
SAN	50.0	49.7
CRN	47.9	47.6
ImageNet	48.9	49.1
Mean	51.0	51.9

Table 1: Generator Accuracy (ACC) and Average Precision (AP) Results "TinyCLIP-ViT-8M-16-Text-3M-YFCC15M"

Generator	Accuracy (ACC)	Average Precision (AP)
DALL-E 2	62.8	70.2
DALL-E 3	47.1	44.4
Stable Diffusion 1-3	62.2	69.8
Stable Diffusion 1-4	61.3	69.7
Stable Diffusion 2	53.8	58.5
Stable Diffusion XL	54.3	55.6
GLIDE	78.9	90.6
Firefly	54.4	59.0
MidJourney V5	51.3	53.1
ProGAN	48.5	47.1
StyleGAN	55.3	63.0
StyleGAN 2	54.4	52.8
BigGAN	53.5	54.4
CycleGAN	43.9	41.3
StarGAN	43.1	41.4
GauGAN	48.0	43.3
DeepFake	50.4	48.3
SeeingThroughFog	50.8	50.8
SAN	50.5	50.7
CRN	51.6	51.7
ImageNet	51.0	47.1
Mean	53.5	55.1

Table 2: Generator Accuracy (ACC) and Average Precision (AP) Results "TinyCLIP-ViT-40M-32-Text-19M-LAION400M"

when working with smaller images, parameter efficiency becomes crucial, as larger models may overfit or struggle to generalize effectively. In our experiments, we observed that reducing the model's size via techniques such as distillation (e.g., using TinyCLIP) was critical to maintaining a balance between model complexity and real-time performance on edge devices. Overall, while the use of 32x32 images significantly reduces the task's difficulty in terms of computational load, it also necessitates careful consideration of trade-offs between accuracy and efficiency, requiring robust architectures capable of extracting meaningful features from the limited input size.

3 Task 2: Identifying and Explaining the Reasons for Image Fakery

In Task 2, the goal is to provide a deeper understanding of why an image is identified as AI-generated, beyond simple detection. While the task of detecting AI-generated images involves identifying specific artifacts or inconsistencies in the image, explaining[2] the reasons behind this classification requires a more detailed analysis. This includes detecting discrepancies in the image's pixels, textures, and structures that are often indicative of synthetic content. These inconsistencies may arise from the generative model's inherent limitations, such as failure to replicate natural lighting, complex textures, or object distortions.

However, one significant challenge in our task is the small image size (32x32 pixels). This size poses several limitations in identifying and explaining AI-generated artifacts effectively. At such a low resolution, many finer details and subtle artifacts, which are typically used for detecting AI generation, may be lost. Features such as unrealistic lighting, unnatural skin tones, or inconsistent object shapes that could be easily spotted in higher-resolution images become increasingly difficult to detect in 32x32 images. Additionally, smaller image sizes result in lower capacity for capturing the contextual relationships between objects or understanding the overall scene, which complicates the task of offering a comprehensive explanation for why an image is AI-generated.

Model	Latency (ms)
TinyCLIP-ViT-8M-16-Text-3M-YFCC15M	22.42
TinyCLIP-ViT-40M-32-Text-19M-LAION400M	27.54

Table 3: Model Latency Table on CPU

These limitations often lead to a trade-off between model accuracy and computational feasibility. With a resolution of 32x32, the model is forced to focus on very low-level features, potentially missing more complex, high-level indicators of AI generation. This presents a fundamental challenge in both the detection and interpretation tasks, as the model may struggle to provide sufficient detail or justification for its predictions. The reduced image size also impacts the model's ability to generalize effectively across different types of AI-generated content, particularly when different generative models exhibit unique artifacts that are typically visible in higher resolutions.

Thus, while Task 2 aims to provide not just a detection but an explanation of AI-generated images, the 32x32 image size imposes significant obstacles that need to be addressed in the model architecture and training process.

3.1 Methodology

3.1.1 ***Dataset Preparation:*** The SynArtifact[3] dataset is a specialized collection designed to address the task of detecting and explaining artifacts in AI-generated images. Unlike general-purpose datasets, SynArtifact consists of high-resolution images (768x768 pixels) that were specifically selected for their potential to exhibit a wide range of artifacts commonly found in AI-generated images, such as texture irregularities, unnatural lighting, and geometric distortions. To facilitate the detection of localized features and allow for more focused analysis, these images were downscaled to 32x32 pixels. This resolution, while aiding in simplifying the detection task, presents challenges by reducing the visibility of finer details that are often crucial for distinguishing synthetic artifacts from real image characteristics.

The dataset is annotated using ChatGPT API, which generates detailed textual descriptions of possible visual artifacts found in each image. The annotations include specific explanations for detected issues like misaligned features, irregular proportions, texture inconsistencies, and unnatural lighting, among others. These annotations serve not only to identify the artifacts but also to explain the underlying reasons for their appearance, such as the limitations of the AI model used in generating the image. By focusing on images that have been intentionally crafted to include synthetic artifacts, the SynArtifact dataset provides a robust foundation for training models to both detect and explain why certain images may be classified as AI-generated, despite the challenges posed by the low-resolution format. This dataset is invaluable for advancing the field of AI image forensics, particularly in the context of identifying subtle or localized inconsistencies in AI-generated content.

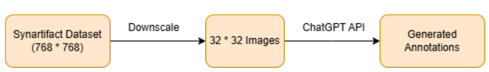

Figure 2: Dataset Preparation

3.1.2 ***Model Selection:*** For the artifact detection task, the LLaVa-7B language model was chosen as the foundation. Task 2 presented a significant challenge: extracting valuable insights from low-resolution, artifact-laden 32x32 images. The LLaVA (7B) model proved to be an effective tool, converting noisy visual data into clear, understandable narratives.

3.1.3 ***Why LLaVa?***

- **Multimodal Understanding:** LLaVA's distinctive ability to process both visual and textual information distinguishes it from traditional language models like LLaMA and Moondream. This multimodal feature enables it to explore visual content more thoroughly and produce more precise and informative descriptions.
- **Semantic Abstraction:** LLaVA is adept at creating concise and informative abstracts, even from low-quality and noisy images. This skill is vital for applications such as image categorization, search, and retrieval.
- **Robustness to Noisy Data:** Thanks to its extensive pretraining on large-scale vision-language datasets, LLaVA can effectively manage noisy and low-resolution images with impressive resilience. This robustness is crucial for real-world applications where data quality may be less than ideal.
- **Efficient Fine-Tuning:** LLaVA can be fine-tuned successfully on relatively small datasets, making it a practical option for various applications. This flexibility allows it to be customized for specific tasks and domains.
- **Computational Efficiency:** The 7B model achieves a balance between performance and computational cost, making it suitable for deployment across a variety of hardware platforms.

To improve its ability to detect and interpret artifacts, the LLaVa-7B model underwent fine-tuning on the SynArtifact dataset. During this training, the model learned to:

- Identify and categorize specific features that indicate AI-generated images.
- Provide clear explanations for its classifications, ensuring transparency and building trust.

3.2 Results

The results of the proposed solution demonstrate its efficacy in detecting and interpreting artifacts in images, with key performance metrics summarized below:

Accuracy and Loss Function: The model's training utilized the **Language Modeling Loss(Cross Entropy Loss)** loss function, achieving high accuracy in distinguishing real images from AI-generated ones. This performance highlights the robustness of the model in identifying subtle and complex artifacts.

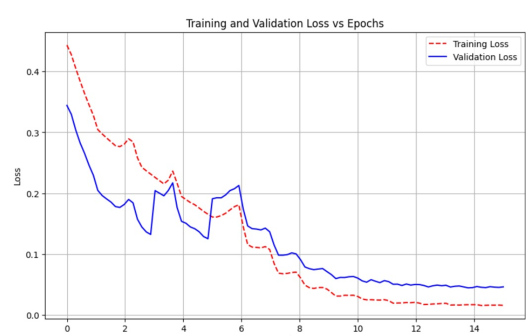

Figure 3: Loss v/s Epochs

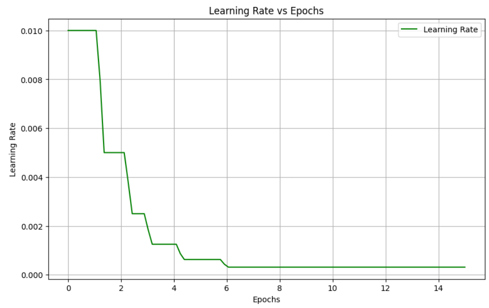

Figure 4: LR v/s Epochs

Qualitative Example:

"ImageId": 2.png

"Question": Analyze the given image and identify any deepfake artifacts. List all artifacts present, referencing their specific locations or manifestations in the image. Provide a detailed explanation of why each artifact is identified as a deepfake feature.

"Answer":

- *Inconsistent object boundaries:* The boundaries of the airplane in the image appear somewhat unclear and merge with the background, which is common in AI-generated images, especially at low resolutions. This can occur due to the AI’s difficulty in distinctly defining edges at a small scale.
- *Floating or disconnected components:* The airplane’s engines appear slightly disconnected from the wings, likely due to the low resolution and the AI’s handling of complex overlapping structures, where it can fail to accurately connect all parts visually.
- *Asymmetric features in naturally symmetric objects:* The wings of the airplane show slight asymmetry; this could be due to the AI’s rendering process, which sometimes struggles with maintaining symmetry in objects that should be uniformly shaped.
- *Texture bleeding between adjacent regions:* The color and texture of the airplane seem to bleed into the sky, indicating a lack of clear separation between different textures. This is typical in AI-generated images where different material properties are not cleanly separated.
- *Over-smoothing of natural textures:* The surface of the airplane lacks detailed textural qualities that one would expect from a high-resolution image, appearing unnaturally smooth and homogeneous. This is often a result of the AI’s processing, which tends to smooth out details to avoid noise.
- *Loss of fine detail in complex structures:* Due to the low resolution, finer details such as the airplane’s windows, landing gears, and surface textures are either completely lost or overly simplified, which is characteristic of downscaled AI-generated images.

3.3 Future Scope

future scope of this research involves several key areas for improvement and expansion:

- **Incorporating Higher-Resolution Images:** Moving from 32x32 to higher-resolution images (e.g., 224x224 or 512x512) to capture finer details and improve artifact detection accuracy.
- **Expanding and Enhancing the Dataset:** Curating larger, diverse datasets that represent various AI generation techniques and artifacts. This includes both synthetic and real-world images.
- **Leveraging Advanced Models and Architectures:** Exploring more powerful models (e.g., Vision Transformers, GPT-4) and hybrid architectures combining computer vision and NLP for better artifact detection and explanation.
- **Adversarial Robustness and Generalization:** Improving model resilience against adversarial attacks and ensuring better generalization across different image domains and artifact categories.
- **Explainability and Interpretability:** Developing techniques such as saliency maps and attention visualization to make the model’s decision-making process more transparent and interpretable.
- **Real-Time Detection and Deployment:** Optimizing models for real-time detection in large-scale platforms, using techniques like model pruning and quantization for speed and efficiency.
- **Ethical Considerations and Fairness:** Ensuring the model is fair, unbiased, and ethical in detecting AI-generated content across diverse datasets and real-world applications.

3.4 Challanges

- **Fine-Tuning on 32x32 Images:**
  - The limited resolution of 32x32 images presented difficulties in capturing fine-grained details, such as texture irregularities, subtle lighting inconsistencies, and intricate distortions typically associated with AI-generated images.
  - Low-resolution images are dominated by pixelation, which hampers the model’s ability to interpret features, making it harder to identify artifacts like blurry edges, noise, or inconsistent textures.
  - The model had to rely on abstract reasoning and pattern recognition rather than direct visual cues, reducing the accuracy and effectiveness of artifact detection.
- **Lack of Original Datasets:**
  - The absence of high-quality, original datasets forced the use of synthetic data generated with ChatGPT. This dataset was helpful but lacked the complexity and variety of real-world AI-generated images.
  - The synthetic dataset may not fully capture the diversity of artifacts in real-world images, which could limit the model’s generalizability and performance on actual AI-generated images.
  - The synthetic data lacked the nuanced details and subtle inconsistencies found in natural AI-generated images, which further impacted the model’s artifact detection capabilities.
- **Resource Constraints – Limited GPU Availability:**
  - Limited access to high-performance GPUs slowed down the fine-tuning process, as the model required substantial

time and computational power for processing the 32x32 images.
- The lack of computational resources hindered rapid iterations, model optimization, and hyperparameter tuning, leading to slower experimentation and performance improvements.
- The limited GPU resources constrained the ability to conduct extensive testing and fine-tune the model to its fullest potential.

- **Challenges in Analysis with 32x32 Images:**
  - Traditional image analysis techniques such as texture analysis, edge detection, and object recognition struggled with 32x32 images due to the pixel-level abstraction and loss of detail during downscaling.
  - Subtle AI-generated artifacts, such as lighting discrepancies, unrealistic reflections, or unnatural object contours, were harder to detect at low resolutions, reducing the accuracy of artifact detection.
  - The model had to rely on global patterns instead of pixel-level details, which led to less accurate or incomplete detection of artifacts.

References

[1] Naveed Akhtar and Ajmal Mian. 2018. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access* 6 (2018), 14410–14430.

[2] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador Garcia, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. 2020. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Inf. Fusion* 58 (June 2020), 82–115.

[3] Bin Cao, Jianhao Yuan, Yexin Liu, Jian Li, Shuyang Sun, Jing Liu, and Bo Zhao. 2024. SynArtifact: Classifying and Alleviating Artifacts in Synthetic Images via Vision-Language Model. *arXiv* 2402.18068 (2024). https://doi.org/10.48550/arXiv.2402.18068 arXiv:2402.18068 [cs.CV].

[4] Christos Koutlis and Symeon Papadopoulos. 2024. Leveraging Representations from Intermediate Encoder-blocks for Synthetic Image Detection. *arXiv* 2302.10174 (2024). https://doi.org/10.48550/arXiv.2302.10174 arXiv:2302.10174 [cs.CV].

[5] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. 2024. Improved Baselines with Visual Instruction Tuning. *arXiv* 2310.03744 (2024). https://doi.org/10.48550/arXiv.2310.03744 arXiv:2310.03744 [cs.CV].

[6] Utkarsh Ojha, Yuheng Li, and Yong Jae Lee. 2023. Towards Universal Fake Image Detectors that Generalize Across Generative Models. *arXiv preprint arXiv:2302.10174* (2023). arXiv:2302.10174 https://doi.org/10.48550/arXiv.2302.10174 https://doi.org/10.48550/arXiv.2302.10174.

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. (June 2017). arXiv:1706.03762 [cs.CL]

[8] William Villegas-Ch, Angel Jaramillo-Alcázar, and Sergio Luján-Mora. 2024. Evaluating the robustness of deep learning models against adversarial attacks: An analysis with FGSM, PGD and CW. *Big Data Cogn. Comput.* 8, 1 (Jan. 2024), 8.

[9] Kan Wu, Houwen Peng, Zhenghong Zhou, Bin Xiao, Mengchen Liu, Lu Yuan, Hong Xuan, Michael Valenzuela, Xi (Stephen) Chen, Xinggang Wang, Hongyang Chao, and Han Hu. 2023. TinyCLIP: CLIP Distillation via Affinity Mimicking and Weight Inheritance. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 21970–21980.