Adobe Mid Prep Final Term Report

Team 97

Abstract

In this study, we address the dual challenges of detecting the authenticity of images and generating coherent explanations for classification decisions. The first task is to differentiate between artificial intelligence-generated and actual visuals. In order to take advantage of multimodal embedding capabilities and improve accuracy in identifying subtle aspects of AI-generated images, we use a hybrid architecture that combines a Vision Transformer (ViT) with Contrastive Language-Image Pretraining (CLIP). The second task is to come up with comprehensible justifications for the classification results. We extract important visual elements using Explainable AI (xAI) techniques and combine them with Large Language Models (LLMs) to provide human-readable explanations based on the model's predictions and previous answers. This two-stage framework aims to establish trust in automated decision-making by combining state-of-the-art image classification methods with robust explanation generation, ensuring transparency and reliability in the detection of AI-generated media. The proposed solution holds potential applications in combating misinformation, enhancing digital media authentication, and fostering responsible AI usage.

1 Introduction

The rapid advancement of generative models has led to an unprecedented rise in AI-generated images, posing significant challenges in distinguishing authentic content from synthetic visuals. This capability is particularly critical in fields such as media verification, content moderation, and digital forensics. Equally important is the need for understanding the rationale behind AI classifications, as it ensures transparency, accountability, and trust in AI-driven systems. This project aims to address these challenges by proposing a dual-task framework: detecting AI-generated images and generating human-interpretable explanations for the classification outcomes.

Task 1 focuses on developing a model that accurately identifies AI-generated images, with particular emphasis on those generated using diffusion models and other state-of-the-art techniques. To this end, we have designed a robust classification model that leverages a hybrid architecture combining CLIP (Contrastive Language-Image Pretraining)[1] with a Vision Transformer (ViT)[2, 3]. This model is trained on a diverse dataset that includes CIFAKE [4–6], Diffusion Forensics [7], and WildDeepFake [8], ensuring exposure to a wide range of real and AI-generated content. Such a varied dataset enables the model to generalize effectively across different generative techniques and image styles.

Task 2 aims to go beyond mere identification by developing a system that not only detects distinguishing artifacts within images but also provides interpretable explanations for its classifications. Interpretability is emphasized through the integration of Grad-CAM (Gradient-weighted Class Activation Mapping) [9], which highlights the Regions of Interest (ROIs) critical to the classification decision. These highlighted regions are subsequently processed by a Large Language Model (LLM), which maps the extracted artifacts to predefined parameters, generating coherent, contextually relevant explanations. To enhance the LLM's interpretability, the dataset for this task is annotated with common generative artifacts, enabling the model to produce explanations that align with human reasoning.

This project not only achieves high accuracy in distinguishing real from AI-generated images but also bridges the gap between AI decision-making and human understanding. By offering a scalable solution, it holds significant potential for applications in combating misinformation and promoting responsible AI deployment.

2 Key Challenges

The rapid evolution of generative models has given rise to a diverse array of synthetic image artifacts, each with unique characteristics. This variation poses a significant challenge in developing a robust classification system capable of handling such diversity. Datasets such as

CIFAKE, Diffusion Forensics, and WildDeepFake represent a broad spectrum of real and AI-generated images, making it crucial for the model to generalize effectively across different generative techniques and image styles. In addition to the diversity of generative methods, AI-generated images often contain subtle, fine-grained artifacts that are nearly indistinguishable to the human eye. Identifying and leveraging these minute details requires highly sophisticated feature extraction and analysis techniques to ensure precise classification.

Furthermore, for practical deployment, the model must exhibit the ability to generalize beyond the training datasets to correctly identify images generated by novel techniques or from real-world datasets. This generalization capability is essential for ensuring the model's relevance and effectiveness in dynamic environments.

Another critical challenge arises from the inherent black-box nature of deep learning models, which complicates the interpretation of their decisions and undermines trust in their predictions. To address this, it is imperative to generate transparent and interpretable explanations for the model's outputs.

Finally, producing coherent and meaningful explanations that align with human reasoning requires the integration of both visual and linguistic insights. The process must seamlessly link identified visual cues with contextual textual explanations to foster user confidence in the system's decisions. These challenges must be addressed comprehensively to develop a system that is not only accurate but also trustworthy and explainable. Our proposed framework, combining CLIP+ViT for classification and an xAI + LLM pipeline for explanation generation, aims to tackle these issues by providing both high classification accuracy and interpretable decision-making.

3 Literature Review

3.1 Contrastive Language-Image Pre-Training

The CLIP (Contrastive Language-Image Pretraining) model, pretrained on large-scale image-text datasets, uses contrastive learning to align image and text embeddings. By maximizing cosine similarity for matching pairs and minimizing it for non-matching pairs, CLIP adapts well to a wide range of tasks, including AI-Generated Image (AIGI) [10] detection. It leverages ResNet [11] or Vision Transformers for image encoding and Transformer networks for text embeddings. While CLIP excels in generalization and domain-shifting scenarios, it struggles with content-specific tasks, such as object counting, due to its content-agnostic nature.

3.1.1 CLIP Model. CLIP relates image and text embeddings through contrastive learning, optimizing similarity for matching pairs and dissimilarity for non-matching pairs. Its robust architecture, using either ResNet or Vision Transformer for image processing, makes it suitable for detecting AIGIs. While effective for general tasks, CLIP's performance can decline in content-specific applications that require detailed domain knowledge.

3.1.2 Datasets. The dataset for fine-tuning includes 1,000 AI-generated images from diffusion models and GANs [12], and real images from the "bedroom" subset of [13]. This diverse dataset ensures the model is exposed to a wide range of generative methods, enhancing its ability to generalize across different types of synthetic images.

3.1.3 CLIP Fine-Tuning. Fine-tuning the pretrained CLIP model (with a ResNet101 encoder) involves adding descriptive captions for each image, specifying its authenticity and generative model. These captions help the model learn to distinguish between real and synthetic images. The fine-tuning process pairs image and text embeddings, optimizing them with cross-entropy loss and the Adam optimizer.

3.1.4 Inference. The fine-tuned CLIP model outperformed traditional models like CNNDet and DIRE[14], achieving over 98% accuracy in detecting AIGIs. It excelled in both binary classification (real vs. synthetic) and generative model identification. CLIP also demonstrated superior computational efficiency, processing large datasets with less GPU memory compared to DIRE.

The results highlight the effectiveness of large-scale pre-trained models like CLIP for AIGI detection. CLIP's generalization ability, efficiency, and adaptability to different generative methods make it a valuable tool for combating synthetic media. This work underscores the potential of pre-trained models in specialized tasks, offering a scalable solution for AI-generated image detection.

3.2 Generalizing Adversarial Explanations with Grad-CAM

Grad-CAM (Gradient-weighted Class Activation Mapping) is a widely used technique for explaining CNN-based models by generating a gradient-based heatmap that highlights the regions of an image that influence the model's decision. However, Grad-CAM does not generalize well to the overall behavior of CNNs [15]. To address this, three new metrics have been introduced to better understand model behavior under adversarial conditions: **Mean Observed Dissimilarity (MOD)** [16], **Variation in Dissimilarity (VID)** [16], and **Normalized Inverted Structural Similarity Index (NISSIM)** [16].

Deep learning models are vulnerable to adversarial attacks, where small, often imperceptible perturbations to the input image can cause misclassification. These attacks, which can be either targeted (forcing a specific wrong prediction) or non-targeted (any incorrect prediction), are difficult for humans to detect and can transfer across different models. Adversarial attacks can be black-box (with no knowledge of the model) or white-box (with full knowledge of the model's architecture and training).

One common adversarial attack is the **Fast Gradient Sign Method (FGSM)** [17], which adds noise to the image based on the gradient of the loss function:

$$\text{adv}_x = x + \epsilon \cdot \text{sign}(\nabla_x J(\theta, x, y))$$

where ϵ controls the magnitude of the perturbation.

3.2.1 Metrics for Model Behavior. Invert Structural Similarity Index (SSIM) and normalise it. The range of SSIM is (-1,1]. Invert SSIM to get dissimilarity and normalise it to get NISSIM with range of (0, 1]. Ideally we want NISSIM to be close to 0.

$$NISSIM_i = \frac{1 - SSIM_i}{2}$$

The MOD metric calculates the mean NISSIM value across adversarial examples, quantifying the overall shift in model focus.

$$MOD_{\text{advset}} = \frac{1}{N} \sum NISSIM_i$$

VID measures the variance of NISSIM values, indicating the stability of the model's focus under adversarial conditions.

$$m_h = \frac{1}{\text{eps}} \sum \text{NISSIM}_{\text{eps}}$$

$$VID = \sqrt{\frac{\sum(\text{NISSIM}_{\text{eps}} - m_h)^2}{\text{eps}}}$$

3.3 Artifact Feature Purification

The Artifact Purification Network (APN) [18] improves cross-domain detection of AI-generated images by purifying artifact features, addressing the limitations of existing methods that overfit to specific generators or scenes. APN combines explicit and implicit purification techniques, followed by a classifier, for robust domain-agnostic artifact detection.

APN's explicit purification involves separating artifact-related features in the frequency and spatial domains. In the frequency domain, the Discrete Fourier Transform (DFT) identifies suspicious frequency bands, and artifact-related components are reconstructed in the spatial domain. The spatial domain uses orthogonal decomposition to separate artifact-related and unrelated features. Implicit purification minimizes mutual information between these features, ensuring domain-independence.

The results show that APN outperforms existing methods, improving accuracy by 5.6% to 16.4% on GenImage and 1.7% to 50.1% on DiffusionForensics. Cross-scene detection shows only a 0.1% accuracy drop between in-domain and out-of-domain samples.

In summary, APN enhances artifact detection across domains, making it a significant advancement in AI-generated image detection.

4 Methodology

4.1 Dataset Accumulation

4.1.1 Extracting Tar Files. The first step involves extracting .tar.gz files from a directory using os.walk() to handle nested structures and tarfile.open() for decompression. The choice of .tar.gz simplifies the process by focusing on a single compression format, ensuring efficient dataset handling. This method avoids the need to process multiple archive types and handles deeply nested files effectively.

4.1.2 Restructuring Dataset for Organization. After extraction, the dataset is reorganized for easier access during training. Images from nested subfolders (e.g., "**REAL**" and "**FAKE**") are consolidated into single folders for each class using shutil.copy(). This flat structure avoids complexities from deep hierarchies, simplifies data processing, and preserves original files. Missing directories are created with os.makedirs(), and source-destination paths are printed for transparency.

4.1.3 Interdependencies in Data Organization. The two-step process-extraction followed by restructuring-ensures that compressed archives are transformed into a simplified format suitable for machine learning tasks. This modular approach can be reused across datasets requiring similar preprocessing.

4.1.4 Labeling the Dataset. Labels are assigned based on directory names: images in "**REAL**" are labeled as **0**, and those in "**FAKE**" as **1**. The folder structure inherently provides class information, with image paths and corresponding labels stored in lists for use during training. Numeric labels ensure compatibility with machine learning algorithms and loss functions like cross-entropy.

4.1.5 Design Decisions. Using folder names for labels is straightforward and aligns with supervised learning requirements. Numeric labels are computationally efficient, and preserving the directory structure maintains clarity while simplifying the labeling process.

4.2 Training Strategy

4.2.1 Dataset Design and Loading. The dataset includes **CIFAKE**, **WildDeepFake**, and **DiffusionForensics**, organized into **REAL** and **FAKE** directories labeled as **0** and **1**, respectively. The **SuperResolutionDataset** class preprocesses images using transformations like resizing, normalization, and augmentation (e.g., RandomResizedCrop and RandomHorizontalFlip) to improve generalization. A stratified train-test split, implemented with train_test_split, ensures balanced class distributions in training and validation sets.

4.2.2 Model Setup and Architecture. CLIP with a Vision Transformer (ViT) backbone is adapted for binary classification by replacing the classifier head with a linear layer $f : \mathbb{R}^d \to \mathbb{R}^2$, mapping extracted features to logits for the REAL and FAKE classes:

$$z = Wx + b$$

where $W \in \mathbb{R}^{2\times d}$ is the weight matrix, $b \in \mathbb{R}^2$ is the bias, and $x \in \mathbb{R}^d$ represents the extracted features.

The **Cross-Entropy Loss** function is used for training:

$$\mathcal{L}CE(y, \hat{y}) = -\sum_{i=1}^{C} y_i \log(\hat{y}_i)$$

where y is the true label (one-hot encoded), $\hat{y}$ is the predicted probability, and $C = 2$ for binary classification. This loss encourages the model to output probabilities close to 1 for the correct class.

4.2.3 Training Loop and Optimization. The training loop uses the **Adam optimizer**:

$$\theta_{t+1} = \theta_t - \eta \frac{m_t}{\sqrt{v_t} + \epsilon}$$

where m_t and v_t are the first and second moment estimates of the gradients, η is the learning rate (set to $1e^{-4}$), and ϵ is a small constant for numerical stability. Gradients are reset using optimizer.zero_grad() to avoid accumulation.

Mini-batches (batch_size $= 8$) are used for efficiency, with loss computed as:

$$\mathcal{L}_{\text{batch}} = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_{\text{CE}}(y^{(i)}, \hat{y}^{(i)})$$

where N is the batch size. Features are extracted by forwarding inputs through the model in train mode, enabling dropout and batch normalization.

4.2.4 Validation, Early Stopping, and Model Saving. Validation tracks performance using the validation set:

$$\text{Accuracy} = \frac{\text{Correct Predictions}}{\text{Total Predictions}}$$

During validation, the model operates in evaluation mode (disabling dropout).

Early stopping halts training if validation accuracy does not improve for a set number of epochs p (patience), avoiding overfitting. If validation accuracy improves, the model's state is saved:

$$\text{torch.save}(\theta, \text{"best\_model.pth"})$$

4.2.5 Integration and Interdependence. Augmentation enhances generalization, while validation, early stopping, and model saving ensure computational efficiency. This pipeline enables robust training and effective detection of fake images by harmonizing all steps, from dataset preparation to model validation.

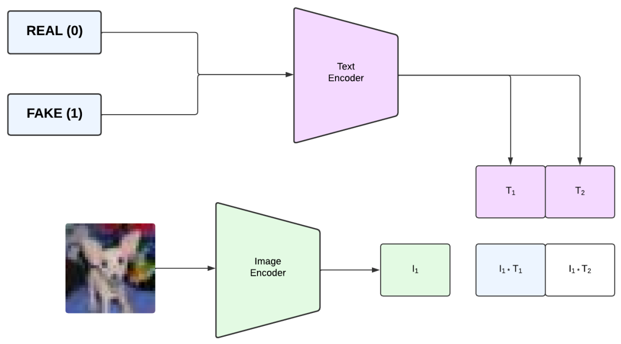


Figure 1: CLIP Architecture

4.3 Inference Pipeline

The inference pipeline of the model is structured to perform two distinct yet integrated tasks: **Task 1, which classifies images as REAL or FAKE (AI-generated)**, and **Task 2, which provides human-readable explanations for the model's classification decisions**. The two tasks are closely interconnected, ensuring both accurate predictions and transparent interpretability of the model's outputs.

4.3.1 ***Task 1: Detecting Real vs. AI-Generated Images.*** The first step of Task 1 involves the preprocessing of the input image to ensure compatibility with the model. The image is resized to **224x224 pixels** to match the input requirements of the Vision Transformer (ViT) model, and the pixel values are normalized to align with the CLIP (Contrastive Language-Image Pretraining) architecture. This preprocessing step guarantees consistent input quality, minimizing variations caused by size or format discrepancies.

Once preprocessed, the image is passed through the **CLIP+ViT** model for feature extraction. CLIP, with its multimodal capabilities, aligns textual and visual features, while the ViT component captures global contextual information from the image. This enables the model to detect subtle patterns such as texture irregularities, unnatural lighting, and edge inconsistencies—artifacts typically introduced by generative models like GANs or diffusion models. These features are critical for distinguishing real images from AI-generated ones.

The extracted features are then passed through a classification head, which outputs a **binary prediction**: a value of **0** for **REAL** images and **1** for **FAKE** (AI-generated) images. This decision is derived from representations learned across multiple datasets, including **CIFAKE**, **Diffusion Forensics**, and **WildDeepFake**, ensuring that the model is robust to a wide array of generative models.

The result of Task 1 is a binary classification, indicating whether the image is real or AI-generated.

4.3.2 ***Task 2: Generating Explanations for the Classification.*** To enhance the interpretability of the model, Task 2 is designed to generate human-readable explanations for the classification results. The first step in Task 2 is the identification of **Regions of Interest (ROIs)** using **GradCAM (Gradient-weighted Class Activation Mapping)**[1]. GradCAM is applied to the **CLIP+ViT** model to generate a **heatmap** that highlights the image regions most influential in the model's decision-making process. For real images, the model typically focuses on natural features such as edges, textures, or lighting. In contrast, for AI-generated images, the heatmap may highlight irregularities, such as blurred transitions, unnatural patterns, or inconsistent lighting.

The ROIs identified by GradCAM are then encoded into a compact representation that captures the visual cues from these highlighted regions. This encoding retains critical information necessary for interpretation in downstream steps.

[1] https://github.com/jacobgil/pytorch-grad-cam

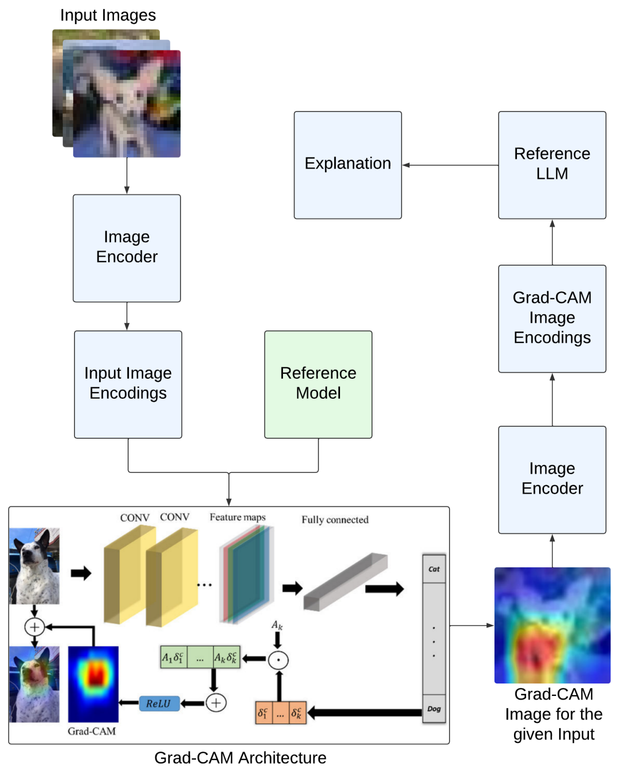


Figure 2: Grad-CAM + LLM Architecture for xAI

Subsequently, a predefined list of common generative artifacts (e.g., edge smoothness, texture uniformity, pattern anomalies) is compiled. These artifacts are incorporated into the prompt, alongside the encoded ROIs, and passed to the **Mistral LLM (Large Language Model)**. Mistral, a powerful text generation model, processes this input and generates a coherent textual explanation, detailing the reasoning behind the classification decision. For instance, the LLM might describe how a "smooth yet unnatural texture" or "lighting inconsistencies" led to the classification as AI-generated.

The output of Task 2 is a detailed textual explanation, which provides transparency into the classification process, making it more interpretable and understandable to the end-user.

4.3.3 Overall View of the Pipeline. The pipeline begins with the input image, which is processed by Task 1 to classify the image as either real or AI-generated. The preprocessed image is passed through the **CLIP+ViT** architecture, which outputs a binary classification result: **0** for **REAL** images and **1** for **FAKE (AI-generated)** images. This result triggers Task 2, where **Grad-CAM** identifies the **ROIs** in the image that influenced the classification decision. These regions, along with predefined artifact descriptions, are used by the **Mistral LLM** to generate a human-readable explanation for the classification.

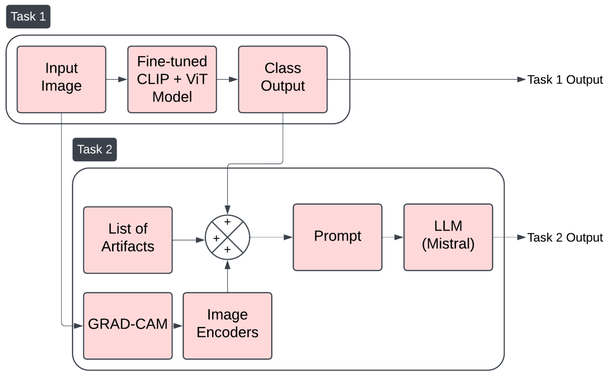


Figure 3: Overall Pipeline

The final output of the pipeline consists of two components: the **Classification** result (REAL or FAKE (AI-generated)) and the **Explanation** result (a detailed, human-readable description of the decision process). This dual-output system ensures that the model not only classifies images but also provides actionable insights into the reasoning behind the classification.

4.4 Overview of Algorithms

Table 1: Summary of Algorithms

Task	Algorithms Used	Alternatives
Classification	CLIP+ViT	ResNet50
Explainability	Grad-CAM + Mistral LLM	SHAP

5 Results and Evaluation

5.1 Inference Notebooks

Training has been done on a total of six different models tailored to specific datasets.

Table 2: Models Overview

Model	Train Accuracy	Test Accuracy (CIFAKE)	Test Accuracy (Diffusion)	Inference	#Parameters	Size
CIFAKE_e10	81.82%	83.86%	48.66%	419ms	150M	570MB
CIFAKE_e30	83.54%	84.66%	49.06%	413ms	150M	570MB
CIFAKE_e50	84.47%	86.4%	49.6%	408ms	150M	570MB
Consolidated_131_e10	77.34%	50.53%	49.86%	930ms	430M	1.59GB
Consolidated_114_e10	77.22%	81.33%	49.46%	930ms	430M	1.59GB
Consolidated_131_e30	85.66%	50.66%	53.46%	961ms	150M	218MB
Quantized_CIFAKE_e10	$\approx 81\%$	48.8%	39.73%	320ms	26M	218MB
Quantized_CIFAKE_e30	$\approx 83\%$	48.8%	52.4%	307ms	26M	218MB
Quantized_CIFAKE_e50	$\approx 84\%$	43.86%	56.66%	314ms	26M	218MB
Quantized_Consolidated_131_e10	$\approx 77\%$	49.46%	56%	2.49s	39M	520MB
Quantized_Consolidated_114_e10	$\approx 77\%$	49.46%	49.86%	2.58s	39M	520MB
Quantized_Consolidated_131_e30	$\approx 85\%$	51.33%	52.4%	591ms	26M	520MB

Table 3: Dataset Description of Models

Model	Dataset Description
CIFAKE_e10	CIFAKE (100%)
CIFAKE_e30	CIFAKE (100%)
CIFAKE_e50	CIFAKE (100%)
Consolidated_131_e10	CIFAKE(80%), Diffusion Forensics(100%), WildDeepFake(12%)
Consolidated_114_e10	CIFAKE(80%), Diffusion Forensics(35%), WildDeepFake(12%)
Consolidated_131_e30	CIFAKE(80%), Diffusion Forensics(100%), WildDeepFake(12%)

Test Dataset is a combination of CIFAKE and Diffusion Forensics. We have subsequently quantized these models to create versions compatible with mobile deployment. The quantization process reduces the model size and computational demands without significantly compromising performance, making them suitable for use in mobile environments where resources are limited.

5.2 Quantitative Results

5.2.1 ***Accuracy and Classification Metrics.*** The performance of Task 1 (Classification) is evaluated using standard metrics such as Accuracy on the Testing Dataset and on the Testing Dataset for Diffusion Forensics. The model was trained on an **NVIDIA GeForce RTX 2080 Ti GPU**, leveraging its **11GB GDDR6** memory for efficient handling of computational tasks. The training process utilized optimized hyperparameters, including a learning rate, batch size, and epochs, to ensure robust convergence. Additionally, the training was accelerated using mixed-precision computations and performed in a CUDA-enabled environment to maximize the hardware's potential.

Dataset Split for Training Model: Training (80%) and Validation (20%) on consolidated dataset i.e. CIFAKE + WildDeepFake + Diffusion-Forensics. The best models (including "on-mobile" quantized models) are reported as follows:

Table 4: Precision/Recall and F1 score

Model	#Parameters	F1 score	Recall
Quantized_CIFAKE_e50	26M	0.29	0.27
CIFAKE_e50	150M	0.86	0.63
Quantized_Con_114e10	39M	0.80	0.60

Depending on the usability and device parameters, one of the three parameters can be used. The three models are equally competitive and provide an equivalent trade-off between file size, inference time and accuracy

Table 5: Comparison with Baselines

Model	Accuracy
CLIP + ViT	86.4%
AlexNet	75%
ResNet50 SDG	77%

*5.2.2 **Inference Time**.*

- Task 1 (Classification): Average inference time per image is **408 ms** for CIFAKE_e50 and **314 ms** for quantized CIFAKE_e50.
- Task 2 (Explainability):
  - Grad-CAM computation: **7.45 seconds/image**
  - LLM inference: **15.39 seconds/sample**

The overall pipeline achieves a total inference time of **23.15 seconds per input**, making it feasible for deployment in near-real-time applications.

5.3 Ablation Study

5.3.1 Effect of Grad-CAM Heatmaps. The implementation of Grad-CAM heatmaps in the processing pipeline significantly enhanced the performance of the LLM providing targeted attention mapping. This technique enabled the LLM to better identify and focus on regions of interest within images, thereby improving its ability to detect and interpret significant artifacts. By stressing the important parts of the image, Grad-CAM facilitated a more refined analysis, allowing the LLM to produce more accurate and contextually relevant explanations for the targeted artifacts.

5.3.2 Sequential Layer implementation in CLIP Processor. In CLIP model's architecture, replacing the standard linear layer with a sequential layer consisting of multiple linear and non-linear activation functions significantly improved model accuracy and learning efficiency. This modification allowed the model to capture more complex image features quickly, showing marked improvements over fewer epochs compared to baseline CLIP models.

6 Discussions

6.1 Limitations of the Current Solution

The CLIP model used in our current approach, while powerful and efficient in many contexts, has limitations when applied to image classification tasks in this specific domain. Notably, the accuracy score of the model is less than 90%, indicating that there is room for improvement in terms of performance and reliability. A contributing factor to this limitation is the image dataset, which contains a mix of variable and skewed features. These inconsistencies in the data make it challenging for the model to generalize effectively, potentially leading to reduced performance during training.

Additionally, a portion of the images in the dataset are blurry, further complicating the task of training a robust model. Image quality is critical for feature extraction, and the presence of blurred images results in incomplete or incorrect feature representations, which directly impacts the model's ability to differentiate between real and AI-generated content. Consequently, the quality and variability of the dataset become significant barriers to achieving higher accuracy and reliable results.

6.2 Observations Regarding the Data

The dataset provided for training and testing consists of images that lack sharp gradients, which makes it particularly difficult for the model to learn effective representations. Sharp gradients in images are essential for accurate edge detection and feature extraction, and their absence creates challenges for the model during both feature extraction and classification tasks. As a result, the overall gradient dataset becomes highly nonlinear and skewed, which makes it harder for the model to distinguish between subtle differences in real and AI-generated images. Moreover, while the explanation output from the model was generally accurate, certain anomalies were observed in the generated explanations.

These inconsistencies can primarily be attributed to the **Mistral LLM** used for explanation generation, which sometimes struggles to interpret image features accurately when provided with input data that deviates from expected norms. The relationship between the visual features extracted by the CLIP model and the textual explanations generated by the LLM requires further refinement to ensure consistency and reduce potential inaccuracies in the explanations.

6.3 Implementation Difficulties

The implementation process was fraught with challenges, particularly in relation to the handling of model outputs during the forward pass in the CLIP model. Initially, we encountered an error that stemmed from a mismatch between the type of value returned by the **CLIP Vision Transformer (ViT)** model and the type expected by downstream processing steps. Specifically, the error message "**AttributeError: 'tuple' object has no attribute 'cpu'**" indicated that the output from the ViT model was a tuple, which was incompatible with the further processing steps that expected a tensor object. Resolving this error required a deeper investigation into the model architecture and output structure.

Once this issue was addressed, another problem arose, which was linked to a minor flaw in the architecture of the model obtained from GitHub. This resulted in a **ValueError**, specifically, "**ValueError: You have to specify pixel values**," indicating that the model's pixel values were not properly defined or passed through the pipeline. This issue was reflective of a limitation in the model's implementation, which had not been fully accounted for during the initial setup.

The primary difficulty encountered in this implementation was the handling of the CLIP Vision Transformer's output, which is a tuple. This necessitated proper handling in both the forward pass and the reshaping function for the transformer's output. Additionally, the need to reshape the output into a spatial format suitable for Grad-CAM further complicated the process. When applying Grad-CAM, an additional error was encountered due to a mismatch in the expected shape of the gradients passed into the Grad-CAM module. Specifically, the error message "**ValueError: Invalid grads shape. Shape of grads should be 4 (2D image) or 5 (3D image)**" indicated that the gradients from the target layer were not in the appropriate form for Grad-CAM to process them correctly. This mismatch led to significant delays and required extensive debugging to resolve.

6.4 Potential Improvements to the Algorithm

Several potential improvements to the algorithm could significantly enhance its performance. One such improvement is replacing the existing **Linear Layer** in the model's classifier with a **Dense Sequential Layer**. This would allow for more complex transformations of the feature space and may provide better accuracy in classifying real and AI-generated images. Additionally, the image dataset used for training could be optimized to be smaller and more concise, focusing on high-quality images that are more representative of the real-world distribution of images. A more focused dataset would reduce noise and improve the model's ability to generalize to unseen data.

Another potential improvement lies in the structure of the dataset itself. Specifically, the gradient of the images could be made linear, which would allow for a more nuanced classification system. Instead of classifying images strictly as either real or fake, we could introduce additional categories, such as "mostly real," "probably real," "probably fake," and "mostly fake." This tiered approach would provide more granular insights into the model's confidence and enhance its ability to deal with ambiguous cases, ultimately improving the robustness of the system and providing more actionable outputs. Such improvements would make the algorithm more efficient and capable of handling a broader range of image types with varying levels of confidence.

6.5 Broader Applications of the Approach in Real-World Scenarios

The approach described in this paper has vast potential for real-world applications, particularly in the detection and analysis of AI-generated content, such as deep fakes, which are becoming an increasing concern on social media platforms. The rise of AI-generated content, especially in the form of videos and images, has created significant challenges for content verification, as these materials can easily be manipulated to deceive viewers. By accurately detecting AI-generated images, the model can be a valuable tool for combating misinformation and preventing the spread of harmful deep fake content across digital platforms.

The potential applications extend beyond social media and content verification. In the field of physical and cyber security, the ability to distinguish between authentic and AI-generated images of individuals or locations is crucial. For example, AI-generated images could be used to create fraudulent identification documents or manipulate video surveillance footage. Our model can play an essential role in identifying these forged images and providing a means of verifying their authenticity, thus strengthening security measures in both physical and digital spaces.

Moreover, the **e-commerce industry** stands to benefit greatly from the adoption of this model. In an online marketplace where product images are crucial for consumer decision-making, our model can be used to determine whether the product images provided by sellers are genuine or if they have been artificially manipulated to mislead customers. This capability would significantly enhance trust between consumers and online vendors, reducing the risk of fraudulent activities.

The **Explainability** feature of the model is another key strength that adds value to these applications. By providing clear, human-readable explanations for the model's decisions, it not only fosters trust in AI-driven tools but also serves as a benchmark for the progress of AI image and artifact detection. The transparency in decision-making ensures that users can better understand how the system arrives at its conclusions, which is crucial in fostering acceptance and trust in AI systems, particularly in high-stakes areas such as security and e-commerce.

7 Conclusion

The proposed model effectively detects and classifies AI-generated images using **CLIP+ViT** for feature extraction and **Mistral LLM** for generating human-readable explanations. It provides robust classification results, but challenges such as suboptimal accuracy and dataset issues (e.g., blurry images, skewed gradients) limit its performance.

The model holds strong potential in real-world applications, such as detecting deep fakes on social media, enhancing physical and cyber security by distinguishing AI-manipulated photos, and verifying product image authenticity in e-commerce. Its **Explainability** feature is crucial for building trust and transparency in AI-driven decisions, especially in sensitive areas like content verification.

In conclusion, while the model shows promise in detecting AI-generated content, further optimization is needed for broader deployment and reliability across multiple domains.

References

[1] 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

[2] N. Park and S. Kim, "How do vision transformers work?" 2022. [Online]. Available: https://arxiv.org/abs/2202.06709

[3] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2021. [Online]. Available: https://arxiv.org/abs/2010.11929

[4] J. J. Bird and A. Lotfi, "Cifake: Image classification and explainable identification of ai-generated synthetic images," *IEEE Access*, vol. 12, pp. 15 642–15 650, 2024.

[5] ——, "Cifake: Image classification and explainable identification of ai-generated synthetic images," 2023. [Online]. Available: https://arxiv.org/abs/2303.14126

[6] A. Krizhevsky, "Learning multiple layers of features from tiny images," *University of Toronto*, 05 2012.

[7] K. Dateset, "Diffusion forensics." [Online]. Available: https://www.kaggle.com/datasets/dngminhli/diffusionforensics-real-and-adm/data

[8] B. Zi, M. Chang, J. Chen, X. Ma, and Y.-G. Jiang, "Wilddeepfake: A challenging real-world dataset for deepfake detection," 2024. [Online]. Available: https://arxiv.org/abs/2101.01456

[9] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," *International Journal of Computer Vision*, vol. 128, no. 2, p. 336–359, Oct. 2019. [Online]. Available: http://dx.doi.org/10.1007/s11263-019-01228-7

[10] A. Moskowitz, T. Gaona, and J. Peterson, "Detecting ai-generated images via clip," 04 2024.

[11] B. Koonce, *ResNet 50*, 01 2021, pp. 63–72.

[12] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014. [Online]. Available: https://arxiv.org/abs/1406.2661

[13] F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao, "Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop," 2016. [Online]. Available: https://arxiv.org/abs/1506.03365

[14] Z. Wang, J. Bao, W. Zhou, W. Wang, H. Hu, H. Chen, and H. Li, “Dire for diffusion-generated image detection,” 2023. [Online]. Available: https://arxiv.org/abs/2303.09295

[15] K. O’Shea and R. Nash, “An introduction to convolutional neural networks,” 2015. [Online]. Available: https://arxiv.org/abs/1511.08458

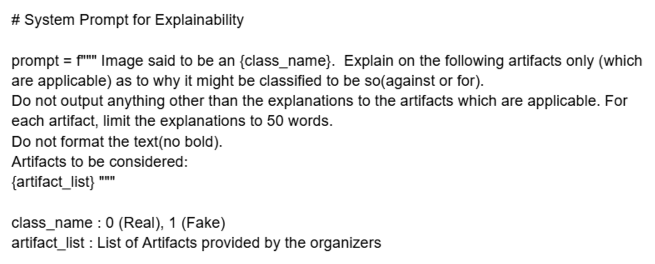
[16] T. Chakraborty, U. Trehan, K. Mallat, and J.-L. Dugelay, “Generalizing adversarial explanations with grad-cam,” 2022. [Online]. Available: https://arxiv.org/abs/2204.05427

[17] J. Sen and S. Dasgupta, “Adversarial attacks on image classification models: Fgsm and patch attacks and their impact,” 2023. [Online]. Available: https://arxiv.org/abs/2307.02055

[18] Z. Meng, B. Peng, J. Dong, and T. Tan, “Artifact feature purification for cross-domain detection of ai-generated images,” 2024. [Online]. Available: https://arxiv.org/abs/2403.11172

A Prompt Templates

A.1 Explainability



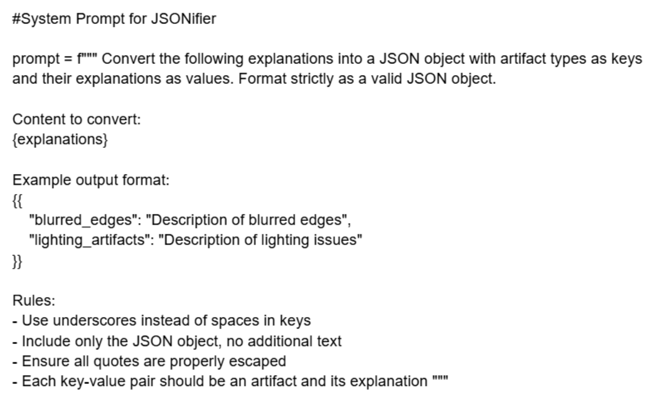
System Prompt for Explainability

prompt = f""" Image said to be an {class_name}. Explain on the following artifacts only (which are applicable) as to why it might be classified to be so(against or for).
Do not output anything other than the explanations to the artifacts which are applicable. For each artifact, limit the explanations to 50 words.
Do not format the text(no bold).
Artifacts to be considered:
{artifact_list} """

class_name : 0 (Real), 1 (Fake)
artifact_list : List of Artifacts provided by the organizers

Figure 4: Prompt Template used for Explainability

A.2 JSONifier



```
#System Prompt for JSONifier

prompt = f""" Convert the following explanations into a JSON object with artifact types as keys
and their explanations as values. Format strictly as a valid JSON object.

Content to convert:
{explanations}

Example output format:
{{
    "blurred_edges": "Description of blurred edges",
    "lighting_artifacts": "Description of lighting issues"
}}

Rules:
- Use underscores instead of spaces in keys
- Include only the JSON object, no additional text
- Ensure all quotes are properly escaped
- Each key-value pair should be an artifact and its explanation """
```

Figure 5: Prompt Template used for JSONifier

B Performance Analysis

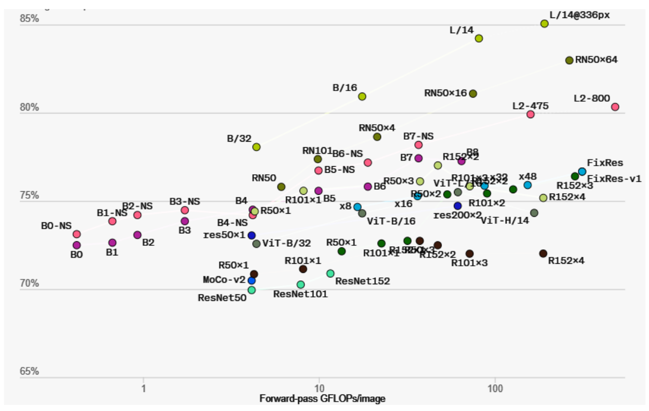
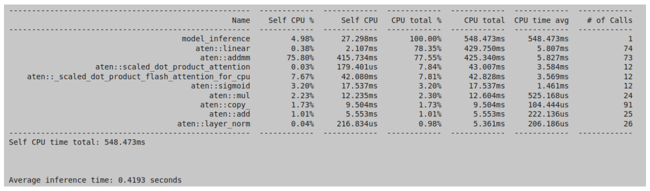


Figure 6: Performance of CLIP in comparison to state-of-art architectures

C Inference Time for Task 1

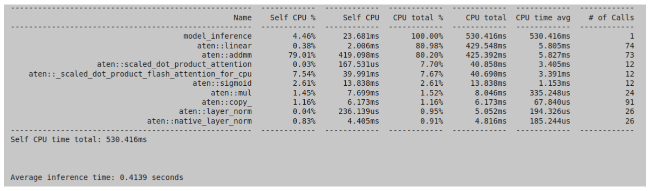
C.1 CIFAKE Datasets



```
-------------------------------------------------  ----------  ----------  ----------  ----------  ----------  ----------
                                             Name  Self CPU %    Self CPU  CPU total %   CPU total CPU time avg  # of Calls
-------------------------------------------------  ----------  ----------  ----------  ----------  ----------  ----------
                                  model_inference       4.98%    27.298ms     100.00%   548.473ms   548.473ms           1
                                     aten::linear       0.38%     2.107ms      78.35%   429.750ms     5.807ms          74
                                      aten::addmm      75.80%   415.734ms      77.55%   425.340ms     5.827ms          73
                  aten::scaled_dot_product_attention       0.03%   179.401us       7.84%    43.007ms     3.584ms          12
   aten::_scaled_dot_product_flash_attention_for_cpu       7.67%    42.080ms       7.81%    42.828ms     3.569ms          12
                                    aten::sigmoid       3.20%    17.537ms       3.20%    17.537ms     1.461ms          12
                                        aten::mul       2.23%    12.235ms       2.30%    12.604ms   525.168us          24
                                      aten::copy_       1.73%     9.504ms       1.73%     9.504ms   104.444us          91
                                        aten::add       1.01%     5.553ms       1.01%     5.553ms   222.136us          25
                                 aten::layer_norm       0.04%   216.834us       0.98%     5.361ms   206.186us          26
-------------------------------------------------  ----------  ----------  ----------  ----------  ----------  ----------
Self CPU time total: 548.473ms


Average inference time: 0.4193 seconds
```

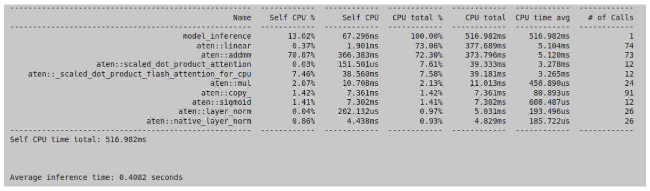
Figure 7: Inference Time for CIFAKE E10



```
-------------------------------------------------  ----------  ----------  ----------  ----------  ----------  ----------
                                             Name  Self CPU %    Self CPU  CPU total %   CPU total CPU time avg  # of Calls
-------------------------------------------------  ----------  ----------  ----------  ----------  ----------  ----------
                                  model_inference       4.46%    23.681ms     100.00%   530.416ms   530.416ms           1
                                     aten::linear       0.38%     2.006ms      80.98%   429.548ms     5.805ms          74
                                      aten::addmm      79.01%   419.098ms      80.20%   425.392ms     5.827ms          73
                  aten::scaled_dot_product_attention       0.03%   167.531us       7.70%    40.858ms     3.405ms          12
   aten::_scaled_dot_product_flash_attention_for_cpu       7.54%    39.991ms       7.67%    40.690ms     3.391ms          12
                                    aten::sigmoid       2.61%    13.838ms       2.61%    13.838ms     1.153ms          12
                                        aten::mul       1.45%     7.699ms       1.52%     8.046ms   335.248us          24
                                      aten::copy_       1.16%     6.173ms       1.16%     6.173ms    67.840us          91
                                 aten::layer_norm       0.04%   236.139us       0.95%     5.052ms   194.326us          26
                          aten::native_layer_norm       0.83%     4.405ms       0.91%     4.816ms   185.244us          26
-------------------------------------------------  ----------  ----------  ----------  ----------  ----------  ----------
Self CPU time total: 530.416ms


Average inference time: 0.4139 seconds
```

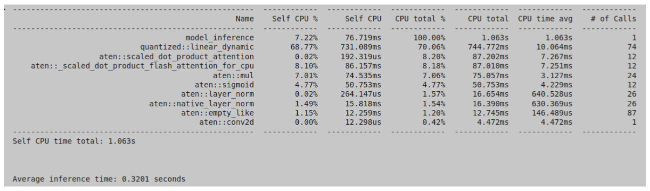
Figure 8: Inference Time for CIFAKE E30



```
-------------------------------------------------  ----------  ----------  ----------  ----------  ----------  ----------
                                             Name  Self CPU %    Self CPU  CPU total %   CPU total CPU time avg  # of Calls
-------------------------------------------------  ----------  ----------  ----------  ----------  ----------  ----------
                                  model_inference      13.02%    67.296ms     100.00%   516.982ms   516.982ms           1
                                     aten::linear       0.37%     1.901ms      73.06%   377.689ms     5.104ms          74
                                      aten::addmm      70.87%   366.383ms      72.30%   373.796ms     5.120ms          73
                  aten::scaled_dot_product_attention       0.03%   151.501us       7.61%    39.333ms     3.278ms          12
   aten::_scaled_dot_product_flash_attention_for_cpu       7.46%    38.560ms       7.58%    39.181ms     3.265ms          12
                                        aten::mul       2.07%    10.708ms       2.13%    11.013ms   458.890us          24
                                      aten::copy_       1.42%     7.361ms       1.42%     7.361ms    80.893us          91
                                    aten::sigmoid       1.41%     7.302ms       1.41%     7.302ms   608.487us          12
                                 aten::layer_norm       0.04%   202.132us       0.97%     5.031ms   193.496us          26
                          aten::native_layer_norm       0.86%     4.438ms       0.93%     4.829ms   185.722us          26
-------------------------------------------------  ----------  ----------  ----------  ----------  ----------  ----------
Self CPU time total: 516.982ms


Average inference time: 0.4082 seconds
```

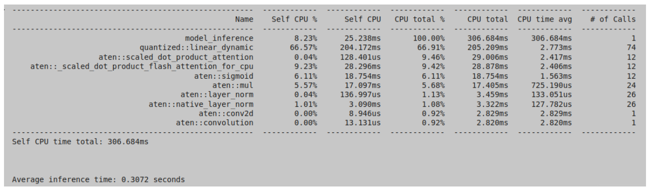
Figure 9: Inference Time for CIFAKE E50



```
--------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
                                              Name    Self CPU %      Self CPU   CPU total %     CPU total  CPU time avg    # of Calls
--------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
                                   model_inference         7.22%      76.719ms       100.00%        1.063s        1.063s             1
                         quantized::linear_dynamic        68.77%     731.089ms        70.06%     744.772ms      10.064ms            74
                aten::scaled_dot_product_attention         0.02%     192.319us         8.20%      87.202ms       7.267ms            12
 aten::_scaled_dot_product_flash_attention_for_cpu         8.10%      86.157ms         8.18%      87.010ms       7.251ms            12
                                         aten::mul         7.01%      74.535ms         7.06%      75.057ms       3.127ms            24
                                     aten::sigmoid         4.77%      50.753ms         4.77%      50.753ms       4.229ms            12
                                  aten::layer_norm         0.02%     264.147us         1.57%      16.654ms     640.528us            26
                           aten::native_layer_norm         1.49%      15.818ms         1.54%      16.390ms     630.369us            26
                                  aten::empty_like         1.15%      12.259ms         1.20%      12.745ms     146.489us            87
                                      aten::conv2d         0.00%      12.298us         0.42%       4.472ms       4.472ms             1
--------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
Self CPU time total: 1.063s


Average inference time: 0.3201 seconds
```

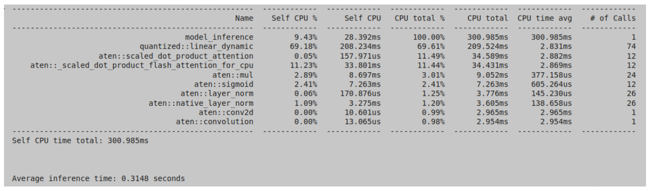
Figure 10: Inference Time for Quantized CIFAKE E10



```
--------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
                                              Name    Self CPU %      Self CPU   CPU total %     CPU total  CPU time avg    # of Calls
--------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
                                   model_inference         8.23%      25.238ms       100.00%     306.684ms     306.684ms             1
                         quantized::linear_dynamic        66.57%     204.172ms        66.91%     205.209ms       2.773ms            74
                aten::scaled_dot_product_attention         0.04%     128.401us         9.46%      29.006ms       2.417ms            12
 aten::_scaled_dot_product_flash_attention_for_cpu         9.23%      28.296ms         9.42%      28.878ms       2.406ms            12
                                     aten::sigmoid         6.11%      18.754ms         6.11%      18.754ms       1.563ms            12
                                         aten::mul         5.57%      17.097ms         5.68%      17.405ms     725.190us            24
                                  aten::layer_norm         0.04%     136.997us         1.13%       3.459ms     133.051us            26
                           aten::native_layer_norm         1.01%       3.090ms         1.08%       3.322ms     127.782us            26
                                      aten::conv2d         0.00%       8.946us         0.92%       2.829ms       2.829ms             1
                                 aten::convolution         0.00%      13.131us         0.92%       2.820ms       2.820ms             1
--------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
Self CPU time total: 306.684ms


Average inference time: 0.3072 seconds
```

Figure 11: Inference Time for Quantized CIFAKE E30

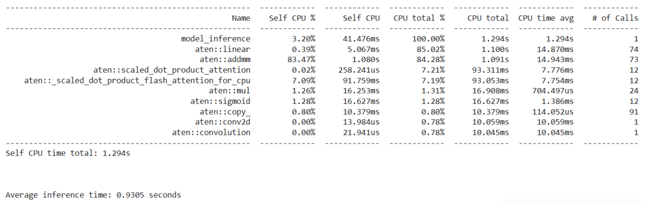


```
--------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
                                              Name    Self CPU %      Self CPU   CPU total %     CPU total  CPU time avg    # of Calls
--------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
                                   model_inference         9.43%      28.392ms       100.00%     300.985ms     300.985ms             1
                         quantized::linear_dynamic        69.18%     208.234ms        69.61%     209.524ms       2.831ms            74
                aten::scaled_dot_product_attention         0.05%     157.971us        11.49%      34.589ms       2.882ms            12
 aten::_scaled_dot_product_flash_attention_for_cpu        11.23%      33.801ms        11.44%      34.431ms       2.869ms            12
                                         aten::mul         2.89%       8.697ms         3.01%       9.052ms     377.158us            24
                                     aten::sigmoid         2.41%       7.263ms         2.41%       7.263ms     605.264us            12
                                  aten::layer_norm         0.06%     170.876us         1.25%       3.776ms     145.230us            26
                           aten::native_layer_norm         1.09%       3.275ms         1.20%       3.605ms     138.658us            26
                                      aten::conv2d         0.00%      10.601us         0.99%       2.965ms       2.965ms             1
                                 aten::convolution         0.00%      13.065us         0.98%       2.954ms       2.954ms             1
--------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
Self CPU time total: 300.985ms


Average inference time: 0.3148 seconds
```

Figure 12: Inference Time for Quantized CIFAKE E50

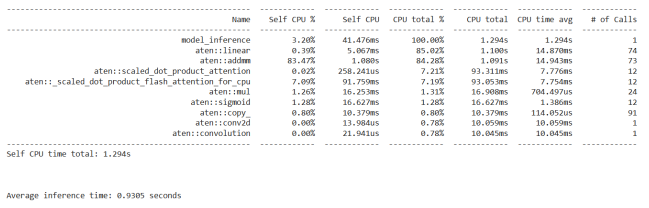
C.2 Consolidated Dataset



```
--------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
                                              Name    Self CPU %      Self CPU   CPU total %     CPU total  CPU time avg    # of Calls
--------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
                                   model_inference         3.20%      41.476ms       100.00%        1.294s        1.294s             1
                                      aten::linear         0.39%       5.067ms        85.02%        1.100s      14.870ms            74
                                       aten::addmm        83.47%        1.080s        84.28%        1.091s      14.943ms            73
                aten::scaled_dot_product_attention         0.02%     258.241us         7.21%      93.311ms       7.776ms            12
 aten::_scaled_dot_product_flash_attention_for_cpu         7.09%      91.759ms         7.19%      93.053ms       7.754ms            12
                                         aten::mul         1.26%      16.253ms         1.31%      16.908ms     704.497us            24
                                     aten::sigmoid         1.28%      16.627ms         1.28%      16.627ms       1.386ms            12
                                       aten::copy_         0.80%      10.379ms         0.80%      10.379ms     114.052us            91
                                      aten::conv2d         0.00%      13.984us         0.78%      10.059ms      10.059ms             1
                                 aten::convolution         0.00%      21.941us         0.78%      10.045ms      10.045ms             1
--------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
Self CPU time total: 1.294s


Average inference time: 0.9305 seconds
```

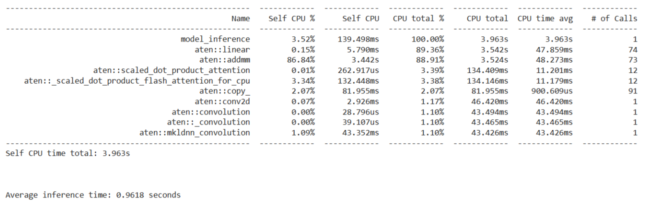
Figure 13: Inference Time for consolidated 114k E10



```
--------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
                                              Name    Self CPU %      Self CPU   CPU total %     CPU total  CPU time avg    # of Calls
--------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
                                   model_inference         3.20%      41.476ms       100.00%        1.294s        1.294s             1
                                      aten::linear         0.39%       5.067ms        85.02%        1.100s      14.870ms            74
                                       aten::addmm        83.47%        1.080s        84.28%        1.091s      14.943ms            73
                aten::scaled_dot_product_attention         0.02%     258.241us         7.21%      93.311ms       7.776ms            12
 aten::_scaled_dot_product_flash_attention_for_cpu         7.09%      91.759ms         7.19%      93.053ms       7.754ms            12
                                         aten::mul         1.26%      16.253ms         1.31%      16.908ms     704.497us            24
                                     aten::sigmoid         1.28%      16.627ms         1.28%      16.627ms       1.386ms            12
                                       aten::copy_         0.80%      10.379ms         0.80%      10.379ms     114.052us            91
                                      aten::conv2d         0.00%      13.984us         0.78%      10.059ms      10.059ms             1
                                 aten::convolution         0.00%      21.941us         0.78%      10.045ms      10.045ms             1
--------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
Self CPU time total: 1.294s


Average inference time: 0.9305 seconds
```

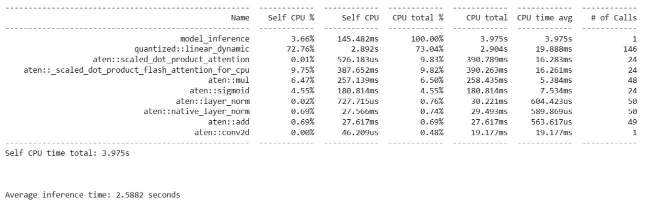
Figure 14: Inference Time for consolidated 131k E10



```
--------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
                                              Name    Self CPU %      Self CPU   CPU total %     CPU total  CPU time avg    # of Calls
--------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
                                   model_inference         3.52%     139.498ms       100.00%        3.963s        3.963s             1
                                      aten::linear         0.15%       5.790ms        89.36%        3.542s      47.859ms            74
                                       aten::addmm        86.84%        3.442s        88.91%        3.524s      48.273ms            73
                aten::scaled_dot_product_attention         0.01%     262.917us         3.39%     134.409ms      11.201ms            12
 aten::_scaled_dot_product_flash_attention_for_cpu         3.34%     132.448ms         3.38%     134.146ms      11.179ms            12
                                       aten::copy_         2.07%      81.955ms         2.07%      81.955ms     900.609us            91
                                      aten::conv2d         0.07%       2.926ms         1.17%      46.420ms      46.420ms             1
                                 aten::convolution         0.00%      28.796us         1.10%      43.494ms      43.494ms             1
                                aten::_convolution         0.00%      39.107us         1.10%      43.465ms      43.465ms             1
                          aten::mkldnn_convolution         1.09%      43.352ms         1.10%      43.426ms      43.426ms             1
--------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
Self CPU time total: 3.963s


Average inference time: 0.9618 seconds
```

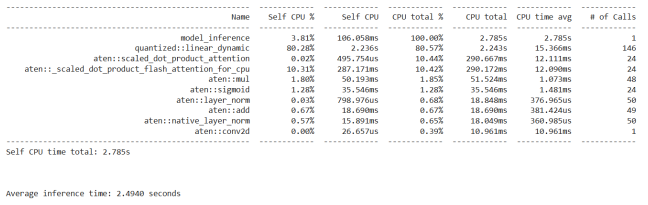
Figure 15: Inference Time for consolidated 131k E30



```
--------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
                                              Name    Self CPU %      Self CPU   CPU total %     CPU total  CPU time avg    # of Calls
--------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
                                   model_inference         3.66%     145.482ms       100.00%        3.975s        3.975s             1
                         quantized::linear_dynamic        72.76%        2.892s        73.04%        2.904s      19.888ms           146
                aten::scaled_dot_product_attention         0.01%     526.183us         9.83%     390.789ms      16.283ms            24
 aten::_scaled_dot_product_flash_attention_for_cpu         9.75%     387.652ms         9.82%     390.263ms      16.261ms            24
                                         aten::mul         6.47%     257.139ms         6.50%     258.435ms       5.384ms            48
                                     aten::sigmoid         4.55%     180.814ms         4.55%     180.814ms       7.534ms            24
                                  aten::layer_norm         0.02%     727.715us         0.76%      30.221ms     604.423us            50
                           aten::native_layer_norm         0.69%      27.566ms         0.74%      29.493ms     589.869us            50
                                         aten::add         0.69%      27.617ms         0.69%      27.617ms     563.617us            49
                                      aten::conv2d         0.00%      46.209us         0.48%      19.177ms      19.177ms             1
--------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
Self CPU time total: 3.975s


Average inference time: 2.5882 seconds
```

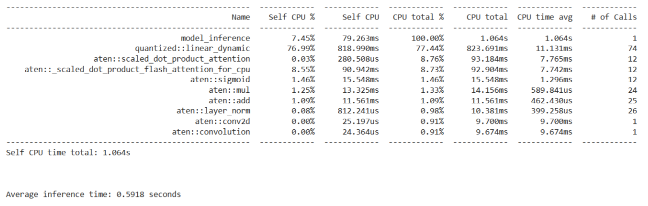
Figure 16: Inference Time for Quantized Consolidated 114k E10



```
--------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
                                              Name    Self CPU %      Self CPU   CPU total %     CPU total  CPU time avg    # of Calls
--------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
                                   model_inference         3.81%     106.058ms       100.00%        2.785s        2.785s             1
                         quantized::linear_dynamic        80.28%        2.236s        80.57%        2.243s      15.366ms           146
                aten::scaled_dot_product_attention         0.02%     495.754us        10.44%     290.667ms      12.111ms            24
 aten::_scaled_dot_product_flash_attention_for_cpu        10.31%     287.171ms        10.42%     290.172ms      12.090ms            24
                                         aten::mul         1.80%      50.193ms         1.85%      51.524ms       1.073ms            48
                                     aten::sigmoid         1.28%      35.546ms         1.28%      35.546ms       1.481ms            24
                                  aten::layer_norm         0.03%     798.976us         0.68%      18.848ms     376.965us            50
                                         aten::add         0.67%      18.690ms         0.67%      18.690ms     381.424us            49
                           aten::native_layer_norm         0.57%      15.891ms         0.65%      18.049ms     360.985us            50
                                      aten::conv2d         0.00%      26.657us         0.39%      10.961ms      10.961ms             1
--------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
Self CPU time total: 2.785s


Average inference time: 2.4940 seconds
```

Figure 17: Inference Time for Quantized Consolidated 131k E10



```
--------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
                                              Name    Self CPU %      Self CPU   CPU total %     CPU total  CPU time avg    # of Calls
--------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
                                   model_inference         7.45%      79.263ms       100.00%        1.064s        1.064s             1
                         quantized::linear_dynamic        76.99%     818.990ms        77.44%     823.691ms      11.131ms            74
                aten::scaled_dot_product_attention         0.03%     280.508us         8.76%      93.184ms       7.765ms            12
 aten::_scaled_dot_product_flash_attention_for_cpu         8.55%      90.942ms         8.73%      92.904ms       7.742ms            12
                                     aten::sigmoid         1.46%      15.548ms         1.46%      15.548ms       1.296ms            12
                                         aten::mul         1.25%      13.325ms         1.33%      14.156ms     589.841us            24
                                         aten::add         1.09%      11.561ms         1.09%      11.561ms     462.430us            25
                                  aten::layer_norm         0.08%     812.241us         0.98%      10.381ms     399.258us            26
                                      aten::conv2d         0.00%      25.197us         0.91%       9.700ms       9.700ms             1
                                 aten::convolution         0.00%      24.364us         0.91%       9.674ms       9.674ms             1
--------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
Self CPU time total: 1.064s


Average inference time: 0.5918 seconds
```

Figure 18: Inference Time for Quantized Consolidated 131k E30