Fake Image Classification and Artifact Detection

Abstract

Recently, there have been significant advancements in image generation technology, fueled by the rapid progress of AI. Advanced generative models, such as Adobe Firefly, Stable Diffusion, and MidJourney, have enabled the creation of highly realistic images, some of which exhibit such a remarkable level of detail that they can even pass the Turing Test. Techniques like Diffusion models and Generative Adversarial Networks (GANs) have been instrumental in achieving this level of realism. Our objective is to develop a classifier that accurately distinguishes between real and AI-generated (fake) images while providing reasoning for its classification. The classifier aims to identify artifacts commonly associated with AI-generated images, including anatomy-related inconsistencies, material-related inconsistencies identified through geometric cues, implausible mechanical or aerodynamic structures, irregular proportions, scale inconsistencies, blurred boundaries, jagged edges, textural anomalies, lighting and shadow mismatches, depth and lighting inconsistencies, as well as stylistic effects such as excessive cinematization or artificial smoothness.

ACM Reference Format:

. 2024. Fake Image Classification and Artifact Detection. In . ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

The rapid advancement of generative AI, driven by models like Generative Adversarial Networks (GANs) and Diffusion models, has heightened cybersecurity threats, particularly through AI-generated fake images. These hyper-realistic synthetic images are increasingly difficult to differentiate from authentic ones, posing significant risks such as disinformation campaigns, identity theft, and financial fraud. As a result, trust in digital media is eroding, image verification processes are becoming more complex, and concerns about privacy and the reliability of digital evidence are growing. To mitigate these threats, it is essential to develop effective detection mechanisms and establish ethical guidelines for the responsible use of generative AI technologies. In this context, we have two key objectives set: 1) to accurately classify real and fake images, and 2) to identify the artifacts present in images detected as fake and provide explainable insights into these findings.

A standard approach to designing a real vs. GAN or Diffusion fake image classifier involves training a binary classifier on a dataset of GAN-generated images or Diffusion models from pre-trained models. However, in real-world applications, access to the specific generative model used by adversaries is often unavailable. Consequently, ensuring the classifier's ability to generalize across diverse generative models is a critical objective. Furthermore, accurately classifying images in datasets containing both real and fake samples poses additional challenges.

To address these limitations, we experimented with a range of direct CNN classifiers, including ResNet50 and a customized EfficientNetV2 model, achieving accuracies of 97.98% , 98%, respectively. Additionally, we explored ConvNeXt, a convolutional neural network inspired by transformer architectures, which demonstrated decent results. Advanced generative techniques, such as those employed in GANs and Diffusion models, leave subtle but detectable fingerprints in the images they generate. To capture these artifacts, we further experimented with transformer-based models, including Swin Transformer and Vision Transformer (ViT).

In parallel, we investigated the frequency domain to detect generative artifacts, leveraging high-frequency representations and wavelet-based attention mechanisms. After extensive experimentation, we propose it as the optimal approach, balancing accuracy and robustness to address the outlined challenges.

Our architecture combines frequency domain processing and spatial features to capture these artifacts effectively. By transforming images into the frequency domain using the Fast Fourier Transform (FFT) [2] and suppressing low-frequency components, the model emphasizes high-frequency features that are typically more indicative of generative inconsistencies. The integration of the Discrete Wavelet Transform (DWT) [1] further enhances detection by decomposing features into multiple scales, isolating critical frequency components while preserving spatial details. This hybrid approach ensures a robust distinction between real and fake images, leveraging the unique fingerprints of generative models in the frequency spectrum.

Existing fake image detection models predominantly focus on binary classification, often without offering interpretability for non-expert audiences. This lack of explainability undermines the credibility of authenticity assessments and risks obscuring potential biases within the models. To mitigate these limitations, we emphasize the incorporation of explainability into our approach. Our goal is to identify artifacts, such as anatomical inconsistencies, stylistic aberrations, shadowing errors, perspective distortions, depth anomalies, and lighting inconsistencies, in images classified as fake, and to provide interpretable explanations for these classifications.

To achieve this, we explored multiple explainability techniques, including SHAP (SHapely Additive exPlanations), LIME (Local Interpretable Model-agnostic Explanations) , and GradCAM, to extract relevant features for interpretation. Additionally, we experimented with leveraging the capabilities of Large Multimodal Models (LMMs) like LLaMA 3.2 11b Vision Model , Pixtral 12b, and LLaVA, testing

their utility for explainability in fake image classification without fine-tuning. However, the results were not very promising, as the image size was 32x32 pixels. Subsequently, we fine-tuned the Vision-Language Models (VLMs) to improve performance. For feature extraction, we incorporated handcrafted methods, such as residual domain analysis, leveraging the distinct chrominance patterns in synthetic images, and texture-based methods, including GLCM(Gray Level Co-occurence Matrix), LBP(Linear Binary Patterns), and HOG(Histogram of Oriented Gradients).

We also explored state-of-the-art Vision-Language (VLM) models like CLIP, SigLIP, PaliGemma and LLaVA, leveraging their strengths in zero-shot learning, contrastive learning, and visual question answering (VQA). These models offered significant advantages in capturing high-level semantic features. After rigorous experimentation and an in-depth analysis of the individual capabilities of these models and techniques, we propose a custom ResNet50 with frequency components as our approach. For the artifact classification and explanation tasks we use a pre-trained CLIP for classifying artifacts, and pre-trained Idefics2 with 8 billion parameters for explanation of the artifacts.

2 Key Challenges

The dataset provided was CIFAKE, a synthetic dataset derived from the CIFAR-10 dataset, designed specifically for training and evaluating fake image detection models. CIFAKE consists of 32x32-pixel images. The low resolution of the dataset posed significant challenges for feature extraction, as subtle artifacts left by generative models are harder to detect in such small images.

Since the test images were generated from a variety of models, we tried to ensure generalizability in our approach. Additionally, the dataset included adversarial perturbations, which were challenging to address. To handle these issues, we tried various experimental methods, including frequency-domain analysis to detect generative model fingerprints and refining our pipeline to improve robustness.

We also tried extracting artifacts from the images, which proved to be another difficult hurdle. Leveraging VLP models like CLIP helped significantly in identifying inconsistencies by aligning textual and visual representations. Despite CLIP's effectiveness, we tried fine-tuning it to enhance performance further. For explainability, we explored VLMs such as LLaVA and tried fine-tuning them to better extract interpretable artifacts and provide human-understandable explanations for classifying images as fake. This iterative process ultimately led us to the comprehensive final pipeline addressing all the posed challenges.

3 Experiments and Methodology for Task 1 (Classification) and Task 2 (Explainability)

Prior to arriving at our robust solution, we explored multiple approaches to address **Task 1 (Classification)** and **Task 2 (Explainability)**. Below, we detail our initial experiments and their outcomes.

3.1 Classification of Real and Fake Images

Our initial approach focused on training convolutional neural network (CNN) classifiers to distinguish between real and fake images. To this end, we experimented with state-of-the-art architectures, including ResNet-50, ConvNeXt-Tiny, and ResNeXt-50. The methodology and corresponding results of these experiments are presented in the following sections.

3.1.1 ***ResNet-50.*** The ResNet-50 model was implemented using PyTorch, leveraging the pretrained weights available in the `torchvision` library. The final fully connected layer was replaced with a linear layer comprising two output nodes to perform binary classification. The model was trained using the CrossEntropyLoss function and the Adam optimizer.

A subset of 20,000 images from the CIFAKE dataset was utilized for training. All images were preprocessed to align with the input requirements of the model. Specifically, the images were resized to 224×224 pixels using bicubic interpolation and normalized with a mean of $[0.485, 0.456, 0.406]$ and a standard deviation of $[0.229, 0.224, 0.225]$.

A dynamic learning rate scheduler was employed to optimize training. The scheduler reduced the learning rate by a factor of 0.1 whenever the validation performance plateaued for 5 consecutive epochs. The minimum learning rate was capped at 1×10^{-6}, with a relative threshold of 0.001. The model was trained for a total of 20 epochs.

Principal outcomes: training accuracy is 100%, validation accuracy is 97.15%

3.1.2 ***ConvNextTiny.*** The ConvNeXt-Tiny model was implemented using TensorFlow and Keras, utilizing pretrained weights from the `tf.keras.applications` module. To adapt the model for binary classification, the top fully connected layers were removed, enabling the addition of a custom classification head. The pretrained layers were initially frozen to preserve the features learned from the ImageNet dataset.

The custom head was designed with the following components:

- A `GlobalAveragePooling2D` layer for dimensionality reduction.
- A `LayerNormalization` layer to stabilize training.
- A fully connected layer with 1024 neurons and ReLU activation for feature extraction.
- A final output layer with 2 neurons and softmax activation to compute class probabilities for real and fake images.

The model was compiled using the AdamW optimizer with a learning rate of 5×10^{-5} and a weight decay of 1×10^{-8}. Binary cross-entropy loss was used as the loss function, and accuracy was selected as the evaluation metric.

A subset of the CIFAKE dataset, containing real and fake images, was used for training and validation. The images were preprocessed by resizing them to 224×224 pixels and normalizing the pixel values to the range $[0, 1]$. A validation split of 20% was applied to the dataset. The training process was conducted with a batch size of 32, ensuring compatibility with the ConvNeXt input format. The model was trained for 100 epochs.

Principal outcomes: training accuracy is 78.72%, validation accuracy is 79%

3.1.3 ***ResNext50_32x4d***. The ResNeXt50_32x4d model was implemented using pretrained weights from the `torchvision.models` package (`ResNeXt50_32x4d_Weights`). The CIFAKE dataset was employed as the primary dataset, with preprocessing performed using `Transform.Compose`. The preprocessing pipeline included resizing images to 224×224 pixels and applying random flipping and rotation to enhance generalization.

To optimize the model, a custom class (`CustomResNeXt50`) was developed. The training process utilized the following configuration:

- **Batch size**: 128
- **Optimizer**: Adam with a learning rate of 3×10^{-3}
- **Loss function**: BCEWithLogitsLoss

To prevent overfitting, a `ReduceLROnPlateau` scheduler was employed to dynamically adjust the learning rate when validation loss plateaued. The model, comprising over 23 million parameters, was trained for 50 epochs.

Principal outcomes: training accuracy is 99.19% and validation accuracy is 97.98%

The primary challenge encountered during the training of these models was the high risk of overfitting to the dataset. Another significant hurdle was the small image size of 32×32, which posed difficulties for effective feature extraction by the models.

3.1.4 ***FreqNET and Resnet***. The proposed architecture integrates spatial and frequency domain processing to extract rich and complementary feature representations, combining high-frequency analysis, wavelet attention mechanisms, and a ResNet-inspired backbone for robust feature extraction. The sequential flow of the architecture is as follows:

1. Input Processing. The input is an RGB image with dimensions $(B, 3, H, W)$, where B is the batch size, H is the height, and W is the width of the image. The High-Frequency Representation of Image (HFRIBlock) processes the input as follows:

- The input image is transformed into the frequency domain using the Fast Fourier Transform (FFT).
- Low-frequency components are suppressed to retain high-frequency details.
- The frequency representation is converted back to the spatial domain using the inverse FFT (iFFT), producing an image enriched with high-frequency features.

2. Frequency-Net (FreqNet). FreqNet is a sub-network designed for frequency-domain feature extraction. It consists of:

- **HFRFSBlock (High-Frequency Representation of Features Across Spatial):** Processes feature maps in the spatial frequency domain by applying FFT along spatial dimensions, suppressing low-frequency information, and passing the result through a convolutional layer.
- **FCLBlock (Frequency Convolution Layer):** Decomposes feature maps into amplitude and phase components in the frequency domain. These components are processed separately through convolutional layers, recombined using polar coordinates, and converted back to the spatial domain with iFFT.
- Alternating **HFRFSBlock** and **FCLBlock** layers progressively extract complex frequency-domain features. A stride-2 convolutional layer at the end reduces dimensions.

3. Wavelet Attention Mechanism. Wavelet-based attention mechanisms are incorporated to capture multi-scale frequency information:

- **Wavelet Attention Block:** Decomposes feature maps using Discrete Wavelet Transform (DWT) into low-frequency and high-frequency components. A softmax-based attention mechanism highlights critical frequency regions by weighting high-frequency components.
- **Wavelet Attention Stride:** Integrates the attention-enhanced wavelet-transformed features with a 1x1 convolutional layer for dimensionality reduction, retaining enriched frequency features.

4. ResNet-Inspired Backbone. The backbone processes spatial features and integrates wavelet-based enhancements:

- **Initial Convolution and Wavelet Attention Pooling:** The input passes through a 7x7 convolutional layer followed by Wavelet Attention Stride for initial spatial feature extraction and downsampling.
- **Residual Blocks:** Four stages of residual blocks are included, each containing three convolutional layers (1x1, 3x3, and 1x1), batch normalization, and ReLU activations. Identity connections retain earlier feature information.
- **Wavelet Attention in Deeper Layers:** Wavelet Attention Stride is applied at deeper stages to enhance multi-scale frequency representation.

5. Integration of FreqNet and ResNet Outputs. The merging of ResNet and FreqNet features happens before the final output, at multiple intermediate points within the network. This allows the model to leverage spatial and frequency domain features together throughout the feature extraction process, enabling a more comprehensive representation for the task at hand.

6. Feature Aggregation and Classification.

- **Adaptive Average Pooling:** Reduces spatial dimensions to a fixed size, regardless of input size.
- **Fully Connected Layer:** The pooled features are flattened and passed through a fully connected layer for binary classification.

Summary. The proposed architecture integrates spatial and frequency domain analysis, combining FreqNet, wavelet-based attention mechanisms, and a ResNet-inspired backbone to extract fine-grained patterns and global characteristics. This hybrid approach is particularly well-suited for high-frequency sensitive tasks such as image forgery detection.

3.2 GradCAM

We have looked into various XAI methods but found GradCAM [4] to be most useful.

GradCAM was employed to interpret the decision-making process of a pretrained classification model designed to distinguish between 'real' and 'fake' images:

- **Last Convolutional Layer Focus:** Visualization was focused exclusively on the last convolutional layer, as it captures high-level spatial features most relevant to the model's predictions.
- **Forward Hook and Activations:** A forward hook was applied to the last convolutional layer to extract its activations during the forward pass.
- **Gradient Calculation:** Gradients of the predicted class score were calculated with respect to the activations to quantify their importance. The gradients were averaged spatially to derive channel-wise weights, which scale the activation maps.

Class Activation Map (CAM) Generation. The resulting Class Activation Map (CAM) was computed to visually represent the critical features contributing to the model's prediction:

- **CAM Calculation:** The weighted activation maps were aggregated, followed by the application of a ReLU activation to retain only positive contributions.
- **Normalization and Resizing:** The CAM was normalized to the range [0, 1] and resized to match the input image's dimensions (224x224 pixels).
- **Colormap Application:** A colormap was applied to the CAM, creating a heatmap that visually emphasizes regions of the image influencing the model's predictions.

Heatmap Overlay and Model Transparency. Finally, the heatmap was overlaid onto the original image to identify artifacts in 'fake' images:

- **Overlay Process:** The heatmap was overlaid onto the original image to highlight regions influencing the model's decision.
- **Interpretability and Evaluation:** This approach enhances model transparency and allows for qualitative evaluation of its reliability in detecting manipulated or anomalous regions within the images.

Though this method was quite robust but it wasn't able to map the exact features effectively(verified with some manually mapped artifacts).

3.3 Explainability

3.3.1 ***VLM Models.*** As GradCAM didn't give good results we moved on to VLM models like CLIP and BLIP. We tried leveraging CLIP's zero shot learning on the provided list of artifacts.

The methodology for detecting artifacts in images utilizes the **CLIP (Contrastive Language-Image Pre-training)** model [3], a state-of-the-art model trained on large datasets containing paired images and textual descriptions. CLIP is capable of processing both visual and textual information, making it highly effective for multimodal tasks like artifact detection. The model is designed to compute **similarity scores between textual descriptions and images**, facilitating the identification of specific visual anomalies based on predefined categories.

Detection Using CLIP Model. The artifact detection process leverages the CLIP model, where a list of predefined artifact descriptions is passed as text labels alongside the images. This approach enables the model to detect various artifacts in the images based on the corresponding textual cues. The model was selected due to its ability to handle diverse inputs, making it effective for detecting artifacts in complex images.

Image Preprocessing. The preprocessing of input images begins by converting them to RGB format to standardize the color channels, ensuring consistency for the model. The images are then resized to 224x224 pixels, the optimal resolution for the CLIP model's processing pipeline. In addition to image preprocessing, the list of predefined artifact categories is tokenized, ensuring that both the images and the textual descriptions are in a compatible format for feature extraction. This step is essential for aligning and normalizing the data, enabling the model to generate accurate similarity scores.

Model Computation and Probability Scoring. After preprocessing, the CLIP model computes similarity scores between the image and each artifact description. The model generates logits, which are raw scores representing the correlation between the image and each artifact label. These logits are then passed through a softmax function to convert them into probabilities, which quantify the likelihood that a specific artifact is present in the image. These probability scores are crucial for ranking the artifacts based on their relevance to the visual content, helping identify the most likely artifacts in the image.

Thresholding and Result Filtering. To ensure that only the most probable artifacts are detected, a threshold is applied to the probability scores. Artifacts with scores above the threshold are considered detected, while those below the threshold are discarded. This thresholding step minimizes false positives and ensures that the detected artifacts are the most significant.

Results Storage and Batch Processing. The results of the artifact detection process are saved in a structured text file for each image, with the filename corresponding to the image. These text files contain the detected artifacts and their associated confidence scores, facilitating easy inspection and further analysis. Additionally, the methodology supports batch processing, allowing the system to iterate over multiple images in a directory, perform artifact detection, and generate a separate text file for each image. This batch processing capability makes the methodology highly scalable, enabling the efficient detection of artifacts in large datasets with hundreds or thousands of images.

The limitation in explainability of CLIP is addressed in the model **Idefics**

Multimodal Input Handling. The **Idefics2** model is specifically designed to handle multimodal tasks by processing both images and text together. It accepts inputs comprising images and a textual query, such as a request to identify the differences between two images. To ensure compatibility between modalities, the model uses a processor that formats the input data appropriately.

Computation. Once the input is formatted, the model computes the relationship between the images and the text. This involves leveraging learned embeddings to evaluate how the textual description aligns with the visual content. This process enables the model to generate a response that is conditioned on both the textual prompt and the visual data.

Generation and Decoding. After processing the input, the model generates a text-based response, such as a detailed explanation of the differences between the images. The response is grounded in the patterns and knowledge the model has acquired during training. This output is then decoded into human-readable text, forming the final result.

Functionality. The **Idefics2** model demonstrates a robust ability to integrate and interpret visual and textual data. This capability makes it well-suited for tasks such as visual question answering and image comparison, where a deep understanding of multimodal inputs is essential.

Though idefics provides us with both artifact identification and explainability, the main limitation in using this directly is that the vision encoder model in this is trained on **224x224 pixels**.

3.3.2 ***Custom Trained Idefics Model.*** The Idefics model integrates **sigLIP's Vision Transformer** [6] with **Mistral as its Language Model**. SigLIP, which is inherently trained on images of resolution 224x224 pixels, serves as the foundation for the visual component of this multimodal architecture.

Initially, we trained the sigLIP Vision Transformer as a binary classifier to distinguish between 'real' and 'fake' images. This training phase achieved an accuracy of approximately 94%. Following this, we extended the model's capabilities to handle 70 distinct classes by reducing the learning rate, enabling robust multiclass classification.

Next, the integration of sigLIP with Mistral was undertaken. During this phase, the Vision Transformer was frozen, and only the Mistral language model was trained. This allowed the system to leverage sigLIP's pretrained visual capabilities while refining Mistral for textual tasks.

The final system operates in two stages. First, an image identified as 'fake' is passed through sigLIP, which performs multiclass classification to detect the corresponding artifacts. Then, the Mistral language model provides explainability for the detected artifacts, enhancing the interpretability and usability of the system.

The hyperparameters used during training are as follows: the model was configured with a learning rate of 2×10^{-4}, mixed-precision training (fp16) was enabled to optimize memory usage, and a per-device batch size of 2 was selected for both training and evaluation. Gradient accumulation steps were set to 8 to simulate a larger batch size. Other key parameters included disabling data loader pin memory, limiting the number of saved checkpoints to 3, and employing a step-based saving strategy. Evaluations were performed every 10 steps, checkpoints were saved every 25 steps, and metrics were logged every 5 steps. Training was capped at 25 steps, and unused columns were removed from the data pipeline.

3.3.3 ***Dataset formation and annotation*** . Though results from CLIP for extracting the artifacts were good, there's still an issue of explainability due to the size of the images. This prompts for the requirement for training a VLM from scratch for which we require a dataset and a customised dataset is made accordingly.

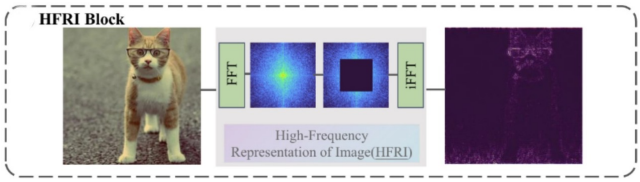

Figure 1: High Frequency Representation of Image

The dataset used for fine-tuning the Vision-Language Model (VLM) was created by blending authentic and AI-generated photos. A total of 150 authentic photographs, obtained from an open-source stock photo website, were uniformly allocated across the 10 classes of CIFAR-10, with 15 images per class.

Photos for synthetic data were generated using the following sources:

- **Stable Diffusion**: 1,500 images
- **Adobe Firefly**: 257 images
- **Midjourney**: 288 images

This resulted in a collection of 2,045 synthetic photos. Of these, 637 images were manually annotated with the assistance of ChatGPT. The annotations focused on identifying artifacts in the generated photos, with the findings archived in JSON files named after the respective images.

The dataset facilitated the assessment of the model's capacity to differentiate between authentic and AI-generated data, thereby enhancing the experiment's robustness.

4 Final Approach Methodology for Task1 (Classification) and Task2 (Explainability)

4.1 Pipeline for Task 1 and Task 2

4.1.1 ***Task 1: FreqNet and ResNet Integration.*** The proposed architecture for Task 1 integrates spatial and frequency domain processing to extract rich and complementary feature representations. This is achieved by combining high-frequency analysis, wavelet attention mechanisms, and a ResNet-inspired backbone for robust feature extraction. The sequential flow of the architecture is outlined below:

1. Input Processing. The input is an RGB image with dimensions $(B, 3, H, W)$, where B is the batch size, H is the height, and W is the width of the image. The High-Frequency Representation of Image (HFRIBlock) processes the input as follows:

- Transform the image into the frequency domain using Fast Fourier Transform (FFT).
- Suppress low-frequency components to retain high-frequency details.
- Convert the frequency representation back to the spatial domain using inverse FFT (iFFT) to produce an image enriched with high-frequency features.

2. Frequency-Net (FreqNet). FreqNet is designed for frequency-domain feature extraction, featuring:

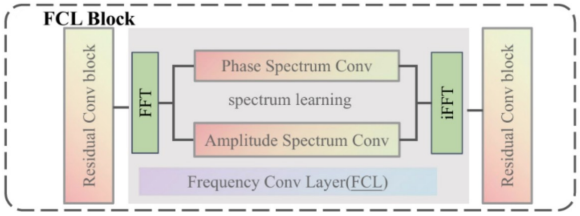

Figure 2: Frequency Conv Layer Block [5]

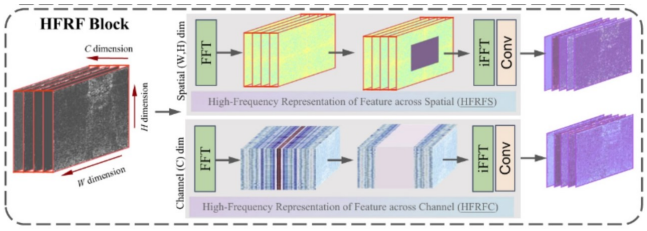

Figure 3: HFRF BLOCK [5]

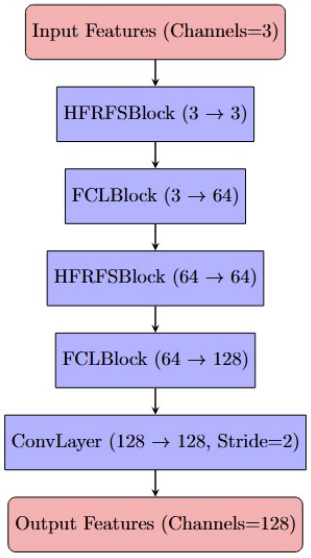

Figure 4: FREQNET ARCHITECTURE [5]

- **HFRFSBlock:** Processes feature maps in the spatial frequency domain by applying FFT, suppressing low-frequency information, and passing the result through a convolutional layer.
- **FCLBlock:** Decomposes feature maps into amplitude and phase components in the frequency domain. These components are processed separately through convolutional layers, recombined using polar coordinates, and converted back to the spatial domain using iFFT.
- Alternating HFRFSBlock and FCLBlock layers extract complex frequency-domain features. A stride-2 convolutional layer reduces dimensions at the end.

3. Wavelet Attention Mechanism. Wavelet-based attention mechanisms enhance multi-scale frequency information:

- **Wavelet Attention Block:** Decomposes feature maps into low- and high-frequency components using Discrete Wavelet Transform (DWT). A softmax-based attention mechanism highlights critical frequency regions by weighting high-frequency components.
- **Wavelet Attention Stride:** Integrates attention-enhanced wavelet-transformed features with a 1x1 convolutional layer for dimensionality reduction.

4. ResNet-Inspired Backbone. The backbone processes spatial features and incorporates wavelet-based enhancements:

- **Initial Convolution and Wavelet Attention Pooling:** The input passes through a 7x7 convolutional layer followed by Wavelet Attention Stride for spatial feature extraction and downsampling.
- **Residual Blocks:** Four stages of residual blocks are included, each with three convolutional layers (1x1, 3x3, 1x1), batch normalization, and ReLU activations. Identity connections retain earlier feature information.

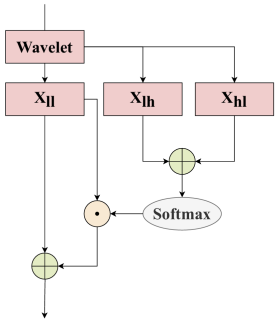

Figure 5: WAVELET ATTENTION BLOCK [7]

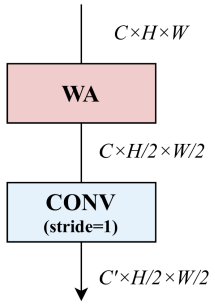

Figure 6: WAVELET ATTENTION STRIDE [7]

- **Wavelet Attention in Deeper Layers:** Wavelet Attention Stride is applied in deeper stages to enhance multi-scale frequency representation.

5. Integration of FreqNet and ResNet Outputs. The outputs from ResNet and FreqNet are merged at multiple intermediate points, allowing the model to leverage spatial and frequency domain features simultaneously throughout the extraction process.

6. Feature Aggregation and Classification.

- **Adaptive Average Pooling:** Reduces spatial dimensions to a fixed size.
- **Fully Connected Layer:** Flattened features are passed through a fully connected layer for binary classification.

Summary. The hybrid architecture integrates FreqNet, wavelet attention mechanisms, and ResNet to extract both fine-grained patterns and global characteristics, making it particularly effective for high-frequency sensitive tasks such as image forgery detection.

4.1.2 *Task 2: Vision-Language Model (VLM) for Explanation Generation.*

For Task 2, we extended the pipeline to include the Vision-Language Model (VLM), **Idefics2** of model size 8 billion parameters, to generate explanations for images detected as fake by the Task 1 pipeline. The process is as follows:

Pipeline Overview. For each image identified as fake from the previous Task 1 pipeline, the image is passed into the Task 2 pipeline, which consists of the following steps:

- The image is first processed through a CLIP model with TextEncoder of size 76 million parameters and ImageEncoder of 115 million parameters, which extracts a list of relevant artifacts associated with the image.
- These artifacts, along with the image, are then passed into the pretrained **Idefics2** model, which generates detailed explanations for the image, outlining why it is detected as fake.

Pre-trained Models and Setup. The models used in Task 2 are pre-trained, eliminating the need for further training:

- CLIP Model: Used for extracting relevant artifacts associated with the input image.
- Idefics2 Model: Pretrained model for generating explanations based on the image and its corresponding artifacts.

Evaluation and Logging. During this phase, the focus is on local evaluations with optimized configurations:

- Evaluation Steps: Evaluations are conducted every 10 steps.
- Logging Steps: Logs are generated every 5 steps to monitor the process.
- Checkpoint Management: Limited checkpoint saving to ensure efficiency.

Unused columns were removed to enhance processing efficiency. The model was not pushed to the Hugging Face hub during this phase, and reporting was disabled to focus on local evaluations.

Conclusion. The trained Idefics2 model demonstrates its applicability when fine-tuned on domain-specific datasets. This training process provides a strong foundation for deploying the model in downstream tasks, leveraging its integrated vision and language processing capabilities.

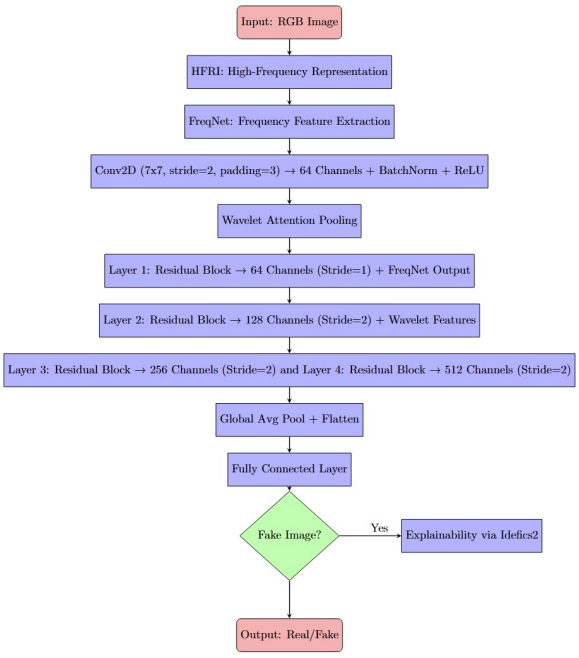

Figure 7: FULL PIPELINE

5 Final Results

5.1 Task 1

We have trained the above mentioned model for **100 epochs**. The training and testing accuracies at 100th epoch are **97.44%** and **95.98%** respectively. The training and testing loss are **0.0678** and **0.1112** respectively.

5.2 Task 2

In the second phase, the model correctly identified the reasons for classifying the images as fake, this was corroborated through visual inspection. However, the detected artifact was being inaccurately attributed to the wrong class. After using pre-trained CLIP for classifying the artifacts and pre-trained Idefics2 for the explanation the final results were not satisfactory due to unstable behavior of the LLM. Therefore the results were manually cleaned before submission.

6 Discussion

Though we have tried to train Idefics2 model from scratch specifically to cater for our problem but though we have trained, there was some issue with the inference code on the trained model though the explanations verified via visual inspections. So we have used the pretrained models.

The hyperparameters used during training are as follows: the model was configured with a learning rate of 2×10^{-4}, mixed-precision training (fp16) was enabled to optimize memory usage, and a per-device batch size of 2 was selected for both training and evaluation. Gradient accumulation steps were set to 8 to simulate

a larger batch size. Other key parameters included disabling data loader pin memory, limiting the number of saved checkpoints to 3, and employing a step-based saving strategy. Evaluations were performed every 10 steps, checkpoints were saved every 25 steps, and metrics were logged every 5 steps. Training was capped at 25 steps, and unused columns were removed from the data pipeline.

We trained this model using LoRA and PEFT.The GPU used for training was NVIDIA A6000 which has 48GB of vram.

7 Hypothesis for identifying artifacts in a fake image

The location of the artifacts can be detected using multiple ways, which includes using models like PaliGemma which also detects bounding boxes and segmentation masks. Another approach can be to set the patch-size of the vision transformer that is used in the VLM equal to four. Using the embeddings of each patch along with the generated explanation, we'll predict their similarity using cosine-similarity from CLIP. The patch with the highest similarity score will most likely have the artifact.

8 Further Scope

Improving Artifact Attribution:

- Enhance the explainability pipeline by refining artifact classification and attribution to the correct classes. This can involve:
  - Developing more robust preprocessing techniques.
  - Fine-tuning Vision-Language Models (VLMs) such as Idefics2 to better align visual features with artifact descriptions.
  - Exploring alternative architectures for multimodal alignment, including advanced CLIP-based models.

Explainability and Interpretability:

- Incorporate advanced explainability techniques such as Grad-CAM++ or Integrated Gradients to provide more accurate artifact localization.
- Develop a user-friendly visualization tool that highlights detected artifacts and explains the classification decision in detail, aimed at non-expert audiences.

Robustness Against Adversarial Attacks:

- Study the robustness of the pipeline against adversarial attacks targeting fake image detection systems. Introduce adversarial training or defenses to mitigate such vulnerabilities.

Using Core Image Processing Principles:

- Leverage classical image processing principles, such as edge detection, texture analysis, gradient analysis, segmentation based feature extraction, and color space transformations, to enhance the feature extraction process and improve artifact detection.
- Integrate these techniques with modern deep learning architectures to build hybrid models that capture both low-level and high-level features effectively.

Ethical Considerations:

- Address ethical implications and biases in fake image detection. Develop fairness metrics to ensure unbiased detection across different contexts and image types.

References

[1] Ingrid Daubechies. Orthonormal bases of compactly supported wavelets. *Communications on Pure and Applied Mathematics*, 41(7):909–996, 1988.

[2] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Prentice Hall, 2nd edition, 2002.

[3] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763, 2021.

[4] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-CAM: Visual explanations from deep networks via gradient-based localization. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 618–626, 2017.

[5] Chuangchuang Tan, Yao Zhao, Shikui Wei, Guanghua Gu, Ping Liu, and Yunchao Wei. Frequency-aware deepfake detection: Improving generalizability through frequency space learning. *To be added (Check publication details)*, 2024. Contact: tanchuangchuang, yzhao, shkwei@bjtu.edu.cn, guguanghua@ysu.edu.cn, pino.pingliu@gmail.com, wychao1987@gmail.com.

[6] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. *To be added (Check publication details)*, 2024. Contact: xzhai, basilm, akolesnikov, lbeyer@google.com.

[7] Xiangyu Zhao. Wavelet-attention cnn for image classification. *To be added (Check publication details)*, 2024. Contributing authors: imagzhaoxy@njust.edu.cn.