AI-Detector

Technical Report

Student: Gianluca Giuseppe Maria De Nardi

Abstract

To identify patterns in AI-generated images, this method focuses on extracting the image's fingerprint based on the difference in entropy between low-texture and high-texture regions. These regions can exhibit various features, such as regular pattern presence and repetitive structures. These features can help detect artifacts such as black patches or uneven colors. If there are many areas with uneven color or inconsistent "noise," this could indicate that the image is fake. Although fingerprints in image analysis are not new, it has traditionally relied on noise to determine the tool used to capture or create the image. This approach uses high-pass filters to identify inter-pixel correlations in different regions by classifying the transformed high-frequency domain instead of the spatial domain and passing it through a sophisticated classifier as a convolutional neural network or a residual neural network.

MODEL

First, the image is divided into 8x8 pixel patches. For each patch, the texture diversity function is applied to distinguish areas rich in detail from those poor in detail. 64 patches rich in detail and 64 patches with the slightest detail are selected. These patches are then constructed into two separate images and sent to the fingerprint model, which applies a high-pass filter to extract high-frequency features from the images. Subsequently, they are passed through a convolution block to:

- Strengthen the relevant features extracted by the high-pass filters, to further emphasize these important features by passing these maps through additional convolutions.
- Allow the model to combine and create more complex representations of the basic features extracted to help the model better understand the structures and patterns within the image.

The processed image's fingerprints and their respective labels are passed to the classifier.

FINGERPRINT

To get the fingerprint of the image, the high-pass filters are convoluted to the image made of rich and poor patches; the idea is to increase the difference between them and intensify the contrast of the colors that an image generates by AI, emphasizing features such as uneven colors and black patches.

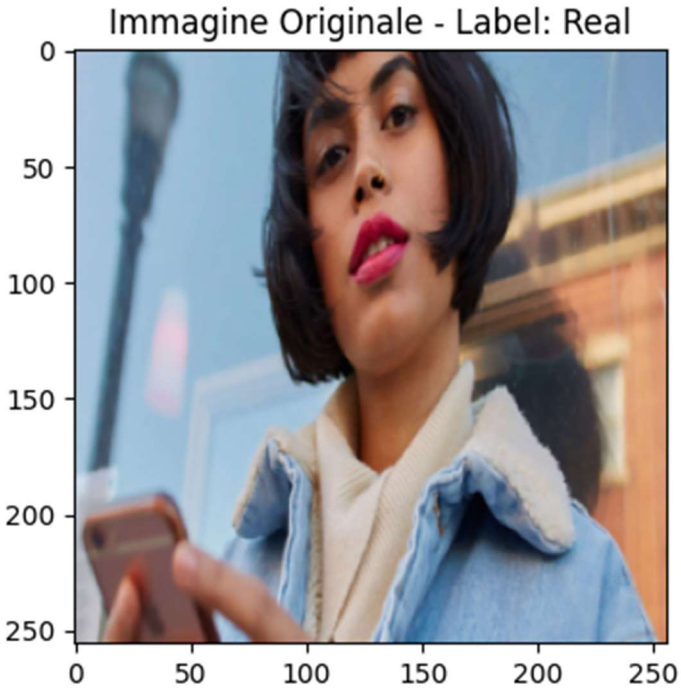
High-pass Filter

The high-pass filter has characteristics that leave the components of the high-frequency signal unchanged and modify the low-frequency ones. The masks implemented and applied to the image are **30** filters with a shape of **5x5** with **1** channel (gray). The rules for a high-pass filter are

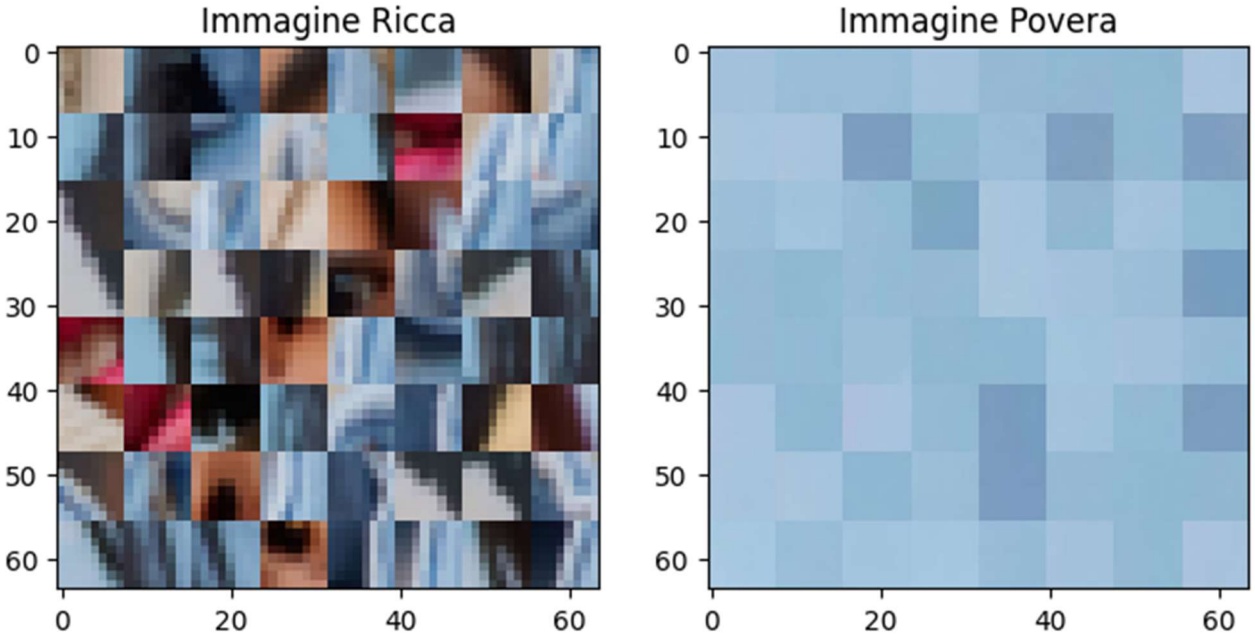
- All filter coefficients can be positive or negative.
- The sum of all coefficients must give 0.

EXAMPLE

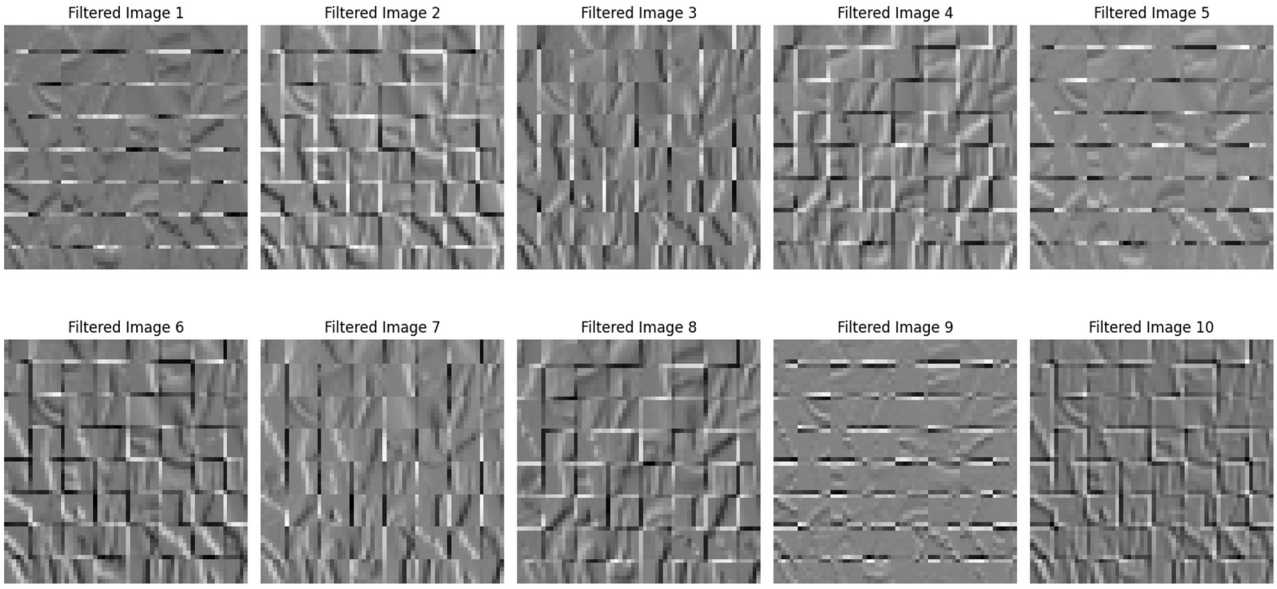
Given this image from the dataset at the fingerprint model :



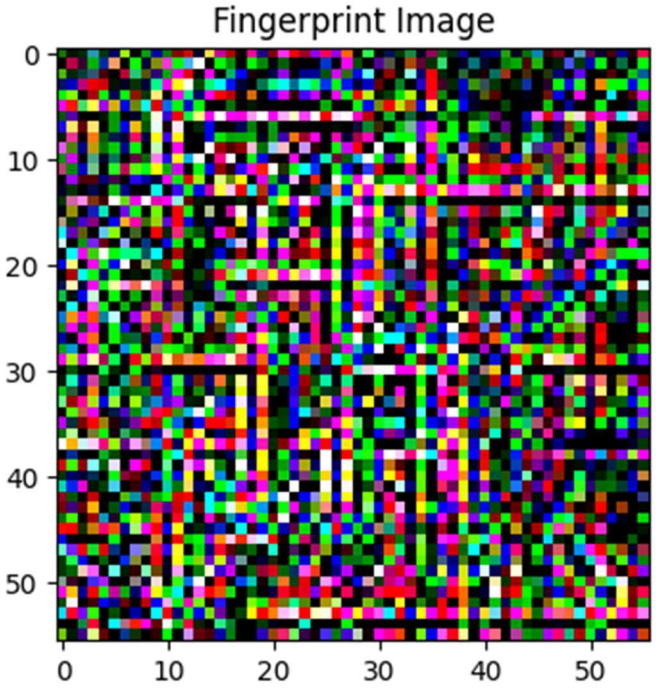
Smashed in patch 8x8:



Convoluted to the filters:



Final image with the high-frequency patches:



Now, with this image, we can be fed at the **classificator** to detect the differences between a real image and a generated one.

CLASSIFICATOR

The model has been tested on 4 classificators:

CNN: with four convolutional layers.

- **Conv1:** 3x3, 32, stride 1
- **MaxPool:** 2x2, stride 2
- **Conv2_x:** [3x3, 32; 3x3, 32] x **2**
- **Conv3_x:** [3x3, 32; 3x3, 32] x **2**
- **Conv4_x:** [3x3, 32; 3x3, 32] x **2**
- **Average Pooling, FC Layer** (1-way sigmoid)

ResNet50: It is a variant with 50 layers. It is often used as a good balance between complexity and performance. Its architecture consists of:

- **Conv1**: 7x7, 64, stride 2
- **MaxPool**: 3x3, stride 2
- **Conv2_x**: [1x1, 64; 3x3, 64; 1x1, 256] x **3**
- **Conv3_x**: [1x1, 128; 3x3, 128; 1x1, 512] x **4**
- **Conv4_x**: [1x1, 256; 3x3, 256; 1x1, 1024] x **6**
- **Conv5_x**: [1x1, 512; 3x3, 512; 1x1, 2048] x **3**
- **Average Pooling**, **FC Layer** (1000-way softmax)

ResNet101 It is a deeper variant with 101 layers, designed to capture more complex representations.

- **Conv1**: 7x7, 64, stride 2
- **MaxPool**: 3x3, stride 2
- **Conv2_x**: [1x1, 64; 3x3, 64; 1x1, 256] x **3**
- **Conv3_x**: [1x1, 128; 3x3, 128; 1x1, 512] x **4**
- **Conv4_x**: [1x1, 256; 3x3, 256; 1x1, 1024] x **23**
- **Conv5_x**: [1x1, 512; 3x3, 512; 1x1, 2048] x **3**
- **Average Pooling**, **FC Layer** (1000-way softmax)

ResNet151: It is one of the deeper variants with 152 layers. It maximizes learning capabilities and captures the most detailed and complex representations:

- **Conv1**: 7x7, 64, stride 2
- **MaxPool**: 3x3, stride 2

- **Conv2_x**: [1x1, 64; 3x3, 64; 1x1, 256] x **3**
- **Conv3_x**: [1x1, 128; 3x3, 128; 1x1, 512] x **8**
- **Conv4_x**: [1x1, 256; 3x3, 256; 1x1, 1024] x **36**
- **Conv5_x**: [1x1, 512; 3x3, 512; 1x1, 2048] x **3**
- **Average Pooling**, **FC Layer** (1000-way softmax)

DATASETS

The most challenging aspect is identifying the dataset. Because of the limited computational power and the absence of large, well-organized datasets, I prioritized image quality over quantity. We then used data augmentation techniques to increase the amount of data points.

API KAGGLE

I established a link to Kaggle through an API connection and successfully downloaded two datasets. The datasets were managed as follows:

Dataset: AI Generated Images vs Real Images:

- **AiArtData**: Label = 0.
- **RealArt**: Label = 1.

This dataset is a captivating ensemble of images sourced from two distinct channels: web scraping and AI-generated content. The content covers many subjects; however, special emphasis was placed on these topics: people, animals, portraits, scenery, and psychedelics.

Key Features:

Web-Scraped Images: These images are harvested from various online sources. The web-scraped images offer a glimpse into the vast spectrum of digital imagery available online, ranging from landscapes, paintings, psychedelic trips, and portraits.

The dataset comprises *538* images labeled as "*0*" and *435* images labeled as "*1*", encompassing various formats such as '**.jpg**', '**.jpeg**' and '**.png**'. Managing this dataset required extensive preprocessing to tackle data inconsistency and specific issues. To address this, the images are consolidated into a unified folder, and labels are assigned to distinguish fake images. Subsequently, the data are partitioned into training, test, and validation sets in preparation for the *DataLoader*.

Dataset: Camera Photos vs Ai generated Photos Classifier:

- **Ai_Images**: Label = 0.
- **Camera_images**: Label = 1.

his dataset has been created to classify AI-generated photos and camera-captured photos. It consists of two distinct categories: camera-captured and AI-generated photos, each containing over 300 high-resolution images in JPG format. High resolution is a critical factor for this model.

DIMENSIONS

- **Input**: The input images have dimensions of a tensor of [3, 128, 128] (3 channels, height 128, width 128).

- **Output**: The output of the fingerprint model has the same dimensions [3, 128, 128] but represents high-frequency features.

TRAINING

For the training phase, the parameters applied are:

Criterion = nn.BCELoss()

The loss function chosen for this model is Binary Cross-Entropy Loss (BCELoss). Let's analyze it in detail:

What is it?

Is a loss function used for binary classification tasks, where the goal is to classify an input into one of two classes (e.g., "fake" or "real"). It is the model's prediction (a probability between 0 and 1). The loss is calculated as the average loss over all examples in the batch.

What are the vantage points?

- **Compatibility** with Sigmoid: The sigmoid produces an output between 0 and 1, which is interpreted as a probability.
- **Suitability** for Binary Classification: It is specifically designed for binary classification tasks, making it a natural choice for distinguishing between two classes.

Optimizer = Adam

Adam is a stochastic optimization algorithm that leverages adaptive estimates of first and second-order moments. It combines the advantages of the AdaGrad and RMSProp optimizers.

What is it?

The optimizer used to update the model's weights is Adam (Adaptive Moment Estimation). Let's analyze it in detail:

How does it work?

Adam uses adaptive estimates of first and second-order moments of the gradient to dynamically adjust the learning rate for each parameter in the model.

Why use it?

- **Adaptability**: Adam adjusts the learning rate for each parameter, making it highly effective across a wide range of problems.

- **Robustness**: It combines the advantages of *AdaGrad* and *RMSProp*, offering faster and more stable convergence.
- **Bias Correction**: Adam includes mechanisms for bias correction in moment estimates, improving optimization effectiveness.

Using a minimal learning rate was crucial for making small parameter adjustments, avoiding large oscillations, and ensuring stable convergence.

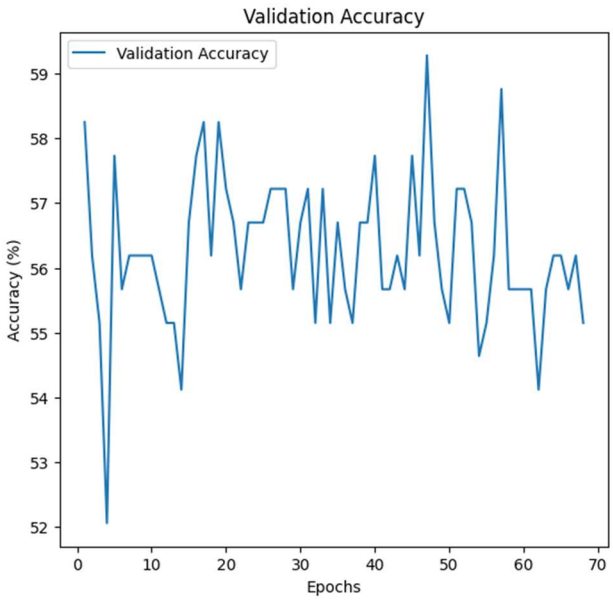
***scheduler.step*:** It effectively reduced the learning rate during the training.

EVALUATION

CNN

Throughout our model training process, the loss and accuracy of the training and validation sets are monitored. This helped to assess how well the model was learning and generalizing to new data. After **600 epochs**, the training loss started at 0.541 and showed minor fluctuations between **0.534** and 0.549 over the epochs. Similarly, the validation loss began at 0.716 and displayed similar fluctuations, remaining within the range of 0.699 to 0.755. This indicates that the model may have difficulty generalizing to the validation data. The validation accuracy started at 58.25% and varied between 52.06% and **59.28%**, with no discernible trend of improvement, indicating that the model needs help to enhance its predictive capability on the validation data. Both the training and validation losses exhibit significant fluctuations without a clear trend of improvement, potentially indicating that the model has difficulty converging to an optimal solution. There is no definitive evidence of overfitting, as the training and validation losses remain close. However, the lack of continuous improvement suggests potential underfitting, indicating that the model may need to be more complex or that the optimization is impractical. The stability in the fluctuations of loss and accuracy implies that the model has reached a plateau but is not at an optimal performance level.

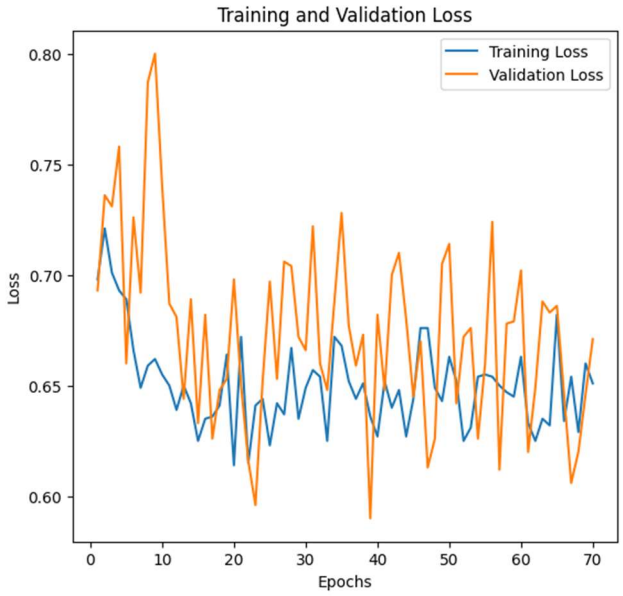
Two graphs illustrate the training and validation loss trends and validation accuracy over the last **70 epochs** out of **600 epochs** to provide a more precise visual representation. The loss graph demonstrates how the training and validation losses change over time, with fluctuating lines suggesting that the model is not converging as expected. The validation accuracy graph shows the trend over time, highlighting the model's difficulties in improving its performance on the validation data.

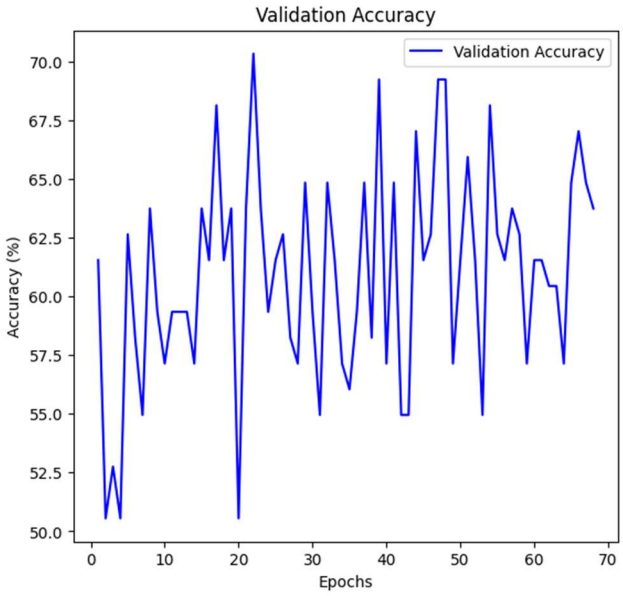


ResNet50

The model displayed interesting and complex learning behavior over the **70 epochs**. Initially, the training loss decreased rapidly, indicating that the model was learning from the training data. However, this trend must be consistently reflected in the validation loss and accuracy, which raised concerns about overfitting or the model's generalization capabilities. The loss decreased significantly during the initial epochs, indicating that the model was learning from the training images. However, some epochs had noticeable fluctuations, suggesting that the model might need to adapt more rigidly to the training data. The validation loss showed considerable variability, with noticeable peaks and drops. This behavior indicated that the model might not generalize well from the training data to the validation data, a clear sign of overfitting.

The validation accuracy fluctuated significantly between epochs, reaching high peaks, such as **70.33%** in some epochs, but also dropping to around 50%. This suggested inconsistency in the model's ability to classify validation images correctly. The fact that accuracy did not consistently increase as training loss decreased indicated that the model might be learning noise or specific details from the training data not present in the validation data.

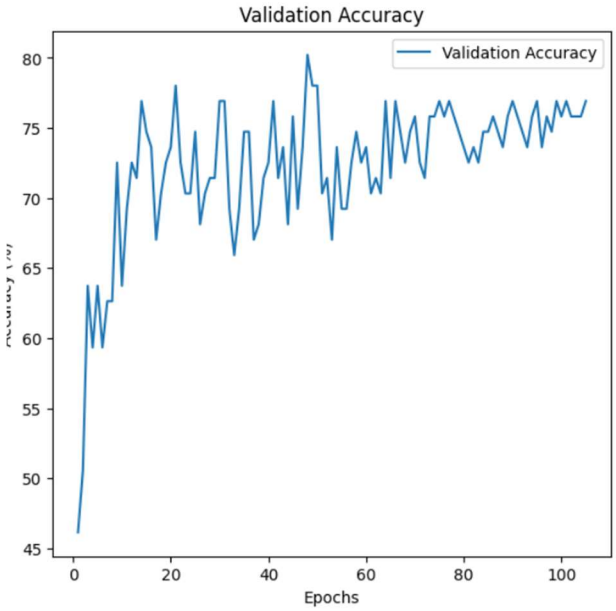




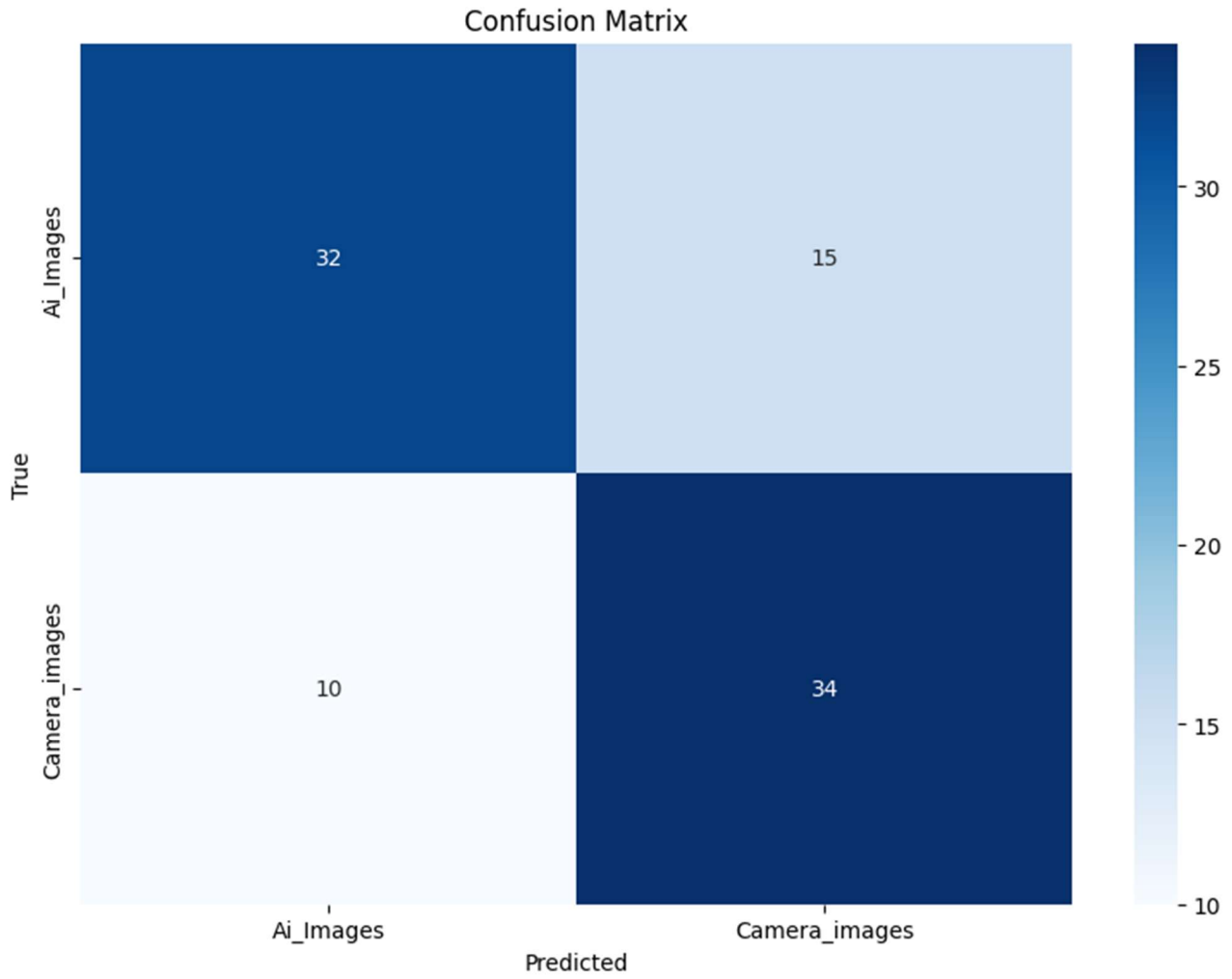
ResNet101 (Better results)

The model seems balanced, as the training and validation losses remain consistently close without significant divergence. However, the fluctuation in the validation loss may indicate that the model is sensitive to variations in the validation set. The model demonstrates stability regarding training loss, validation loss, and validation accuracy, suggesting it has achieved good convergence. The final validation accuracy is around **73-79%**, representing good performance. Nonetheless, the fluctuations in the validation loss suggest there may be room for further improvements through additional regularization techniques or parameter optimization. The trends in loss and accuracy indicate that the model benefited from a lengthy training period (**100 epochs**), but it may not require many more epochs as its performance has stabilized.

In conclusion, the model exhibits good learning and generalization capabilities. However, the fluctuations in the validation loss indicate that exploring further stabilization and regularization techniques might be beneficial. The validation accuracy is satisfactory but could be enhanced through fine-tuning and additional experiments.



Confusion Matrix



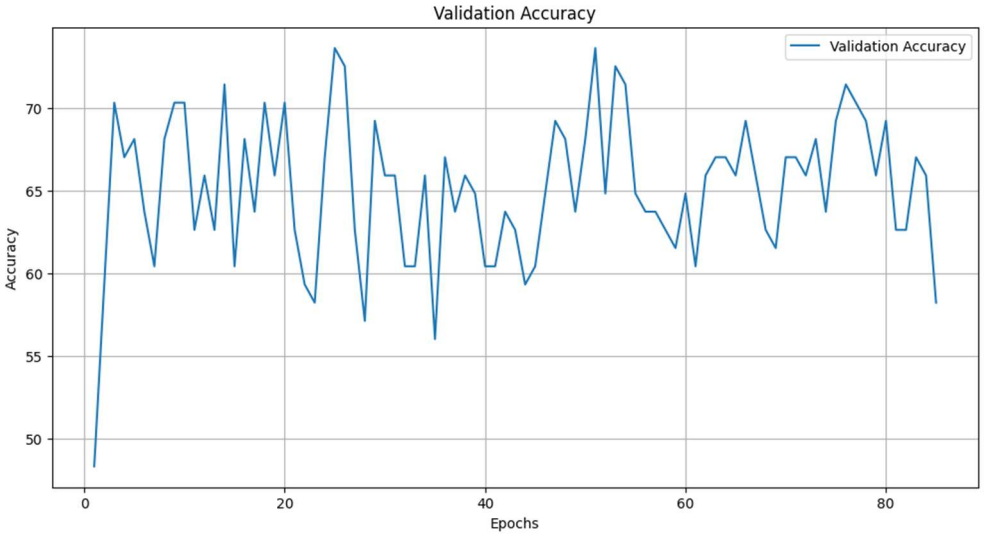
The confusion matrix and associated metrics, including *precision*, *recall*, and *F1-score*, comprehensively assess the model's effectiveness. When precision and recall are harmoniously balanced, it indicates that the model is performing well.

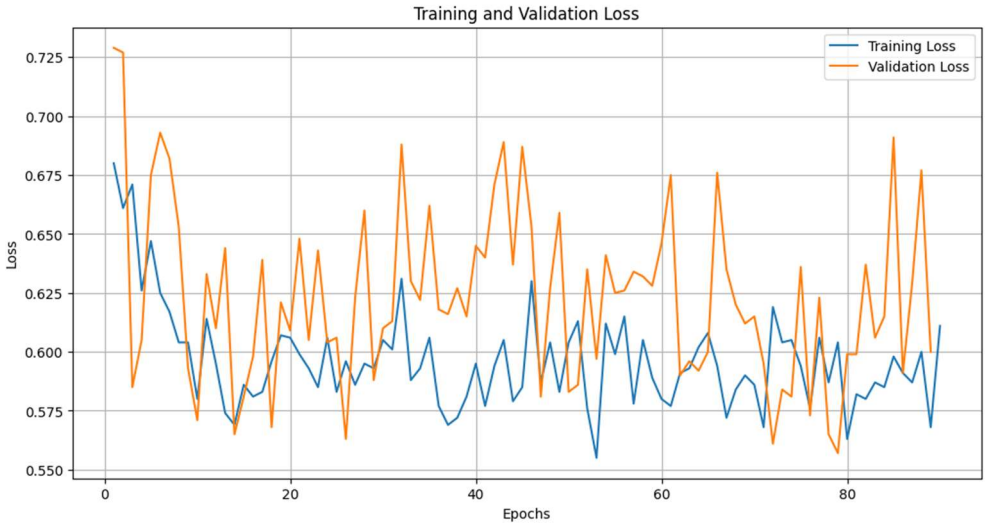
Performance evaluation

- **Precision**:
  - AI Images:0.68
  - Camera Images: .77
- **Recall**:
  - AI Images:0.76
  - Camera Images:0.69
- **F1-Score**:
  - AI Images: 0.72
  - Camera Images: 0.73

RESNET151

Throughout the **100 epochs**, the model displayed varying learning behaviors, showing improvements and challenges. The training loss gradually decreased over the epochs, starting at 0.680 in the first epoch and fluctuating around lower values, such as 0.563 by epoch 80. In contrast, the validation loss showed significant variability. While it decreased in some epochs, it increased in others, suggesting that the model does not always generalize well to the validation data. Validation accuracy improved in various epochs, reaching a peak of **73.63%** in epoch 51. However, there were also periods of stagnation and decline, with values like 48.35% in the first epoch and subsequent fluctuations. The model is learning, but its generalization ability varies. Overall, the model demonstrates an improvement trend, with the training loss consistently decreasing and validation accuracy reaching significant values in some epochs. The model architecture can learn from the training images, but more data is needed.

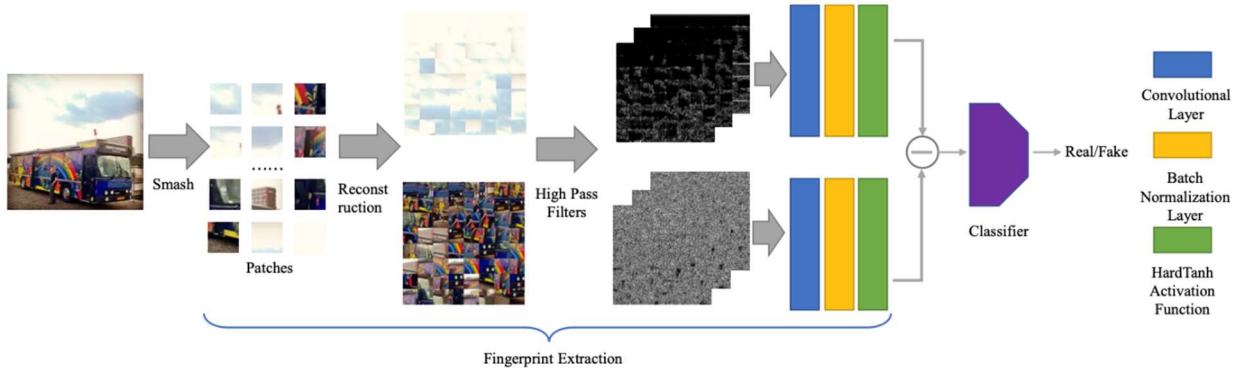




TABLE

Classifier	*Accuracy*	*Loss*	*Epochs*
CNN	*59.28%*	*0.53*	***600***
ResNet50	*70.33%*	*0.65*	*70*
ResNet101	***79%***	***0.52***	*100*
ResNet151	*73.63%*	*0.56*	*100*

Model baseline



The model baseline idea has three steps:

1. Extract rich and poor texture regions.
2. Get fingerprints from these two parts.
3. Classification of the fingerprint image based on artifacts detected.

CONCLUSION

The proposed method is effective, especially with ResNet101, but relies on complex information extraction models and substantial amounts of data. It is crucial to ensure data quality, and during the training phase, prioritizing image quality over quantity is advisable. I believe incorporating a Swin Transformer could significantly improve the accuracy and predictive values of the classifier. In particular, the sliding window function could introduce attention to patches before applying high-pass filters to emphasize high-frequency differences between patches, simplifying the classifier's task and ensuring efficient model performance even with limited datasets. However, validating this hypothesis would necessitate high-performance GPUs. This refined approach highlights both the successes and areas for improvement in model performance, offering valuable insights to guide future research and optimization endeavors.

References

1. Durall, Ricard, Margret Keuper, and Janis Keuper. "Watch your up-convolution: Cnn based generative deep neural networks are failing to reproduce spectral distributions." *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020.

2. Frank, Joel, et al. "Leveraging frequency analysis for deep fake image recognition." *International conference on machine learning*. PMLR, 2020.

3. https://github.com/dustin-nguyen-qil/AI-Generated-Image-Detection

4. Alaeddine, Hmidi, and Malek Jihene. "Deep residual network in network." *Computational Intelligence and Neuroscience* 2021.1 (2021): 6659083.

5. Susladkar, Onkar, et al. "ClarifyNet: A high-pass and low-pass filtering based CNN for single image dehazing." Journal of systems architecture 132 (2022): 102736.