From Real to Synthetic: A Pipeline for Fake Image Detection and Model Identification

Cocca Flavia
Università degli Studi di Modena e Reggio Emilia
253218@studenti.unimore.it

Cosimi Tommaso
Università degli Studi di Modena e Reggio Emilia
338969@studenti.unimore.it

De Bellis Elena Maria
Università degli Studi di Modena e Reggio Emilia
256249@studenti.unimore.it

Contents

Abstract

In recent years, AI-generated images have become a powerful asset for the creative community, such as illustrators and designers, to assist their workflow. However, this same technology can also be misused by individuals with harmful intentions, who might leverage its capabilities to craft deceptive content.

The idea is to develop a pipeline to understand if a picture is real or generated, and if so, which kind of model generated it.

The proposed pipeline feeds data from the frequency spectrum rather than the original RGB image to a model which has to be able to distinguish between real and fake images, and in the latter case discriminate which generator was used in the creation of the original image itself.

1. Introduction

The widely adopted use of Generative models (GANs and Diffusion Models) has made it increasingly difficult to distinguish genuine content from synthetic one just by looking at it. While instruments such as Midjourney or Stable Diffusion offer vast creative possibilities, they also pose critical risks related to disinformation and deepfakes.

The adoption of synthetic content in traditional media has already triggered significant public controversy. Major brands such as Coca-Cola and McDonald's have faced strong consumer backlash following the release of advertising campaigns created entirely or partially using artificial intelligence [1, 2]. Similar criticism has emerged in other sectors, including publishing, where Tor Books was subject to scrutiny for using generative art on the cover of a novel [3], and fashion where Vogue employed artificially generated models in its productions [4].

Even more alarming is the use of these technologies in the development of hyper-realistic deepfakes aimed at fraud or disinformation. A striking example is the recent case involving Brad Pitt, whose likeness was cloned to promote online scams, clearly demonstrating the risks posed by such systems [5]. These episodes highlight the need for robust automated solutions for deepfake detection and source attribution, in order to restore human trust in digital visual media.

Recent scientific literature suggests that, although generated images may appear flawless in the spatial domain, they often have distinctive fingerprints in the frequency domain. These spectral fingerprints are frequently introduced by architectural components of generative models,

such as upsampling operations and convolutional filters.

Motivated by these findings, the adopted approach discards the traditional RGB analysis to focus exclusively on the spectral characteristics. This study proposes a personalized CNN (Convolutional Neural Network) that takes as input a four channel tensor, which does not represent the original image in the spatial domain as in traditional approaches, and exploits the Discrete Fourier Transform - thus the frequency domain of the original image - to gather different characteristics of the image itself:

- Magnitude of the Discrete Fourier Transform of the Image;
- Phase (decomposed in sine and cosine in separate channels) of the Discrete Fourier Transform of the Image;
- Autocorrelation of the Image in the frequency domain.

2. Related Work

The DeepFake Detection problem has been addressed with different methodologies. Approaches based on the spatial domain use standard CNNs (e.g. ResNet[6], EfficientNet[7]) trained on huge datasets of RGB images [8]. Although effective, these models tend not to be robust on specific visual artifacts (e.g. blurred background, eyes asymmetries, image compression).

To overcome these limitations, more recent approaches focus on the frequency domain. Studies such as those by Wang et al.[9], Frank et al.[10] and Durall et al.[11] have shown that analysis conducted in the frequency domain can make the network detect hidden periodic patterns invisible to the human eye.

More recently, efforts have focused on improving cross-model generalization. Ojha et al.[12] proposed universal fake image detectors designed to remain effective across multiple and unseen generators. However, these approaches typically frame the task as binary classification, neglecting the problem of source attribution, which is increasingly relevant in accountability-driven scenarios.

Our work fits into the frequency-domain paradigm, distinguishing itself by the use of a composite representation of the input that includes not only the magnitude spectrum, but also the phase components (fundamental for the image structure) and the autocorrelation, capturing repetitive patterns in the frequency domain. These features are then processed by a custom convolutional architecture, allowing for both detection of generated images and generator-specific source attribution.

3. Data

The dataset used as a base in this project is ELSA_D3[13, 14]. It is a multimodal dataset designed for deep fake detection. Among other information, the data structure provides, for each example, a URL pointing to a real image along with a description of such image, which is then used as a prompt to generate four synthetic variants:

- `image_gen0` → DeepFloyd/IF-II-L-v1.0;
- `image_gen1` → CompVis/stable-diffusion-v1-4;
- `image_gen2` → stabilityai/stable-diffusion-2-1-base;
- `image_gen3` → stabilityai/stable-diffusion-xl-base-1.0.

This organization proves to be particularly suitable for our setting, since it supports at the same time both the division between real and generated images and the multi-class source attribution.

Figure 1. Handpicked example of interesting data from one row of the ELSA_D3 Dataset

3.1. Data Acquisition and Management

Given the possibly large dimension of the dataset, we implemented a script that streams dataset chunks, downloading real images through HTTP requests using the provided URL, and retrieving generated ones directly from the dataset chunk itself.

During this acquisition phase, a validity check is performed on URLs and content types to ensure that only valid images are processed, discarding any corrupted or unreachable links, and the related generated images.

To optimize memory usage and enable scalable storage, all images are converted to raw bytes and written progressively in Parquet chunks.

In our configuration, the acquisition process collects a total of 16.384 dataset entries, each containing five images (one per class), making our custom dataset of 81.920 images perfectly balanced.

4. Preprocessing

The preprocessing phase is focused on frequency analysis.

Every image is initially converted to greyscale, and then resized to a fixed resolution of 256×256 pixels, to ensure consistency across the dataset.

Subsequently, a 2D Discrete Fourier Transform is applied to the image to switch from the spatial domain to the frequency domain. From the resulting Fourier transform, we derive a four-channel spectral representation of the original image, which constitutes the input to the neural network.

Each channel has the spatial resolution 256×256, resulting in a tensor of shape $4 \times 256 \times 256$.

Given the definition of the 2D Discrete Fourier Transform adapted to our 256×256 greyscale input:

$$\begin{aligned} \mathrm{DFT}(I) &= \sum_{x=0}^{255} \sum_{y=0}^{255} I(x,y) e^{-j2\pi\left(\frac{ux}{256} \frac{vy}{256}\right)} \\ &= \Re + j\Im \end{aligned} \quad (1)$$

Where:

- I is the image;
- $u \in [0, 255]$ are the horizontal frequency indices;

- $v \in [0, 255]$ are the vertical frequency indices.

Then the channels are defined as follows:

1. **Magnitude Spectrum**
   We use the Log Magnitude, which is computed as:
   $$\text{Magnitude} = 20 \times \log_{10}(\sqrt{\Re^2 + \Im^2})$$
2. **Phase Spectrum (sine)**
   To avoid losing directional and structural information, computed as:
   $$\text{Sine Phase Spectrum} = \sin\left(\arctan\left(\frac{\Im}{\Re}\right)\right)$$
3. **Phase Spectrum (cosine)**
   Complementarily we define the cosine of the phase angle to enable a continuous and stable encoding of the phase information, without relying on complex values:
   $$\text{Cosine Phase Spectrum} = \cos\left(\arctan\left(\frac{\Im}{\Re}\right)\right)$$
4. **Autocorrelation**
   The autocorrelation is computed using the Inverse 2D DFT of the power spectrum, useful for highlighting repeated periodic patterns typical of generative convolutions:
   $$\text{Autocorrelation} = DFT^{-1}((\Re^2 + \Im^2) * (\Re^2 - \Im^2))$$

In Figure 2 a sample of what the different computed channels look like.

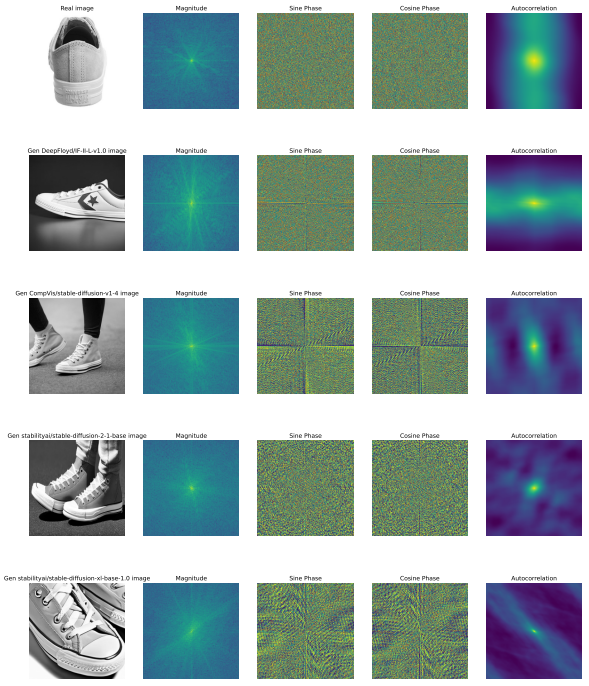


Figure 2. Example run of the Preprocessing pipeline over one row of the Dataset

5. Model

The proposed architecture takes inspiration from ResNet [6], its variants and the Inception Block from GoogleNet[15]. It's a fully custom solution trained from scratch. The Model is composed by a combination of the following building blocks.

5.1. Channel Expansion Layers

The network begins with a 3×3 convolution that expands the feature space to 16 channels, this is then followed by an initial residual block to expand the channels to 64.

In Figure 3 a visualization of the aforementioned initial layers.

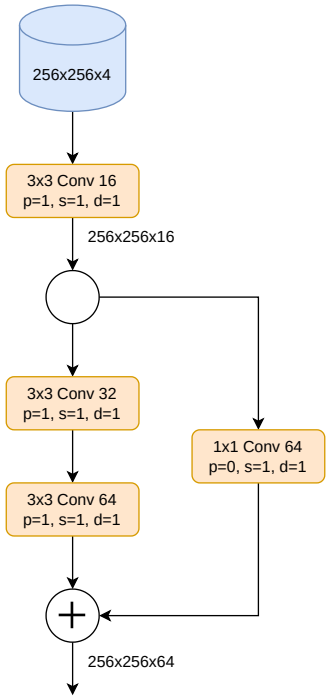


Figure 3. Initial layers of the architecture

5.2. Operational Blocks

The core of the network consists of four instances of the Operational Block, each implementing a parallel multi-branch convolutional architecture combined with residual connections. Each block contains:

- **Branch 1**
  A sequence of four 3×3 convolutions
- **Branch 2**
  A sequence of two 3×3 convolutions with a dilation factor of 2

The outputs of the branches are concatenated along the channel dimension and summed to a residual connection.

This design enables the network to capture both fine-grained and more coarse spectral patterns. Each block is followed by max-pooling to progressively reduce spatial resolution and increase receptive field size.

In Figure 4 a visualization of an example of the operational block.

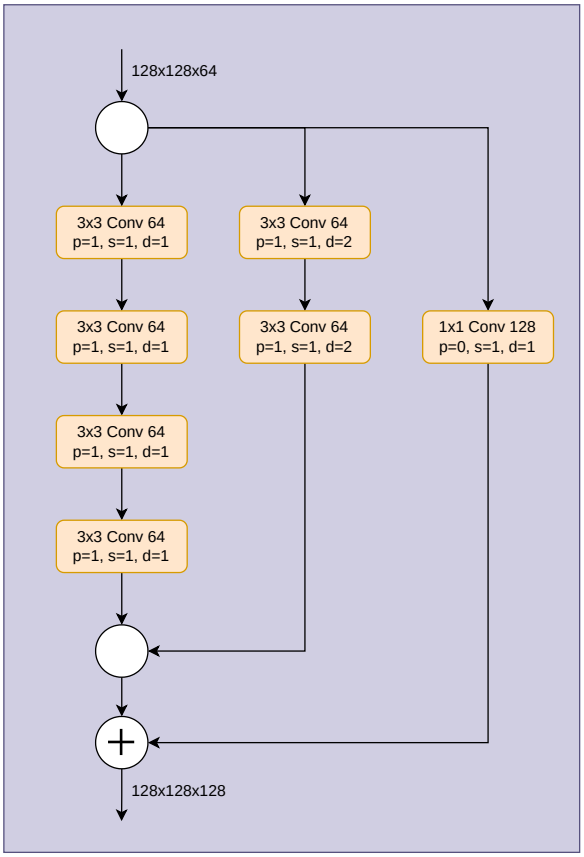


Figure 4. Example of a functional Operational Block

In the overall model structure it is possible to find a sequence of the above blocks alternating with MaxPooling layers, which halve the spatial resolution as shown in Figure 5.

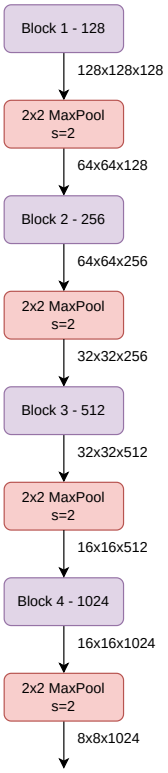


Figure 5. Sequence of Operational Blocks

5.3. Channel Compression Layers

Before proceding with the Global Pooling Layers a residual block, consisting of a series of three 3×3 convolutions that keeps intact the number of channels, is implemented. This is then followed by three convolutional layers which actuate the channel compression from 1024 to 512 to 256 to a final number of channels of 128.

In Figure 6 a visualization of these layers.

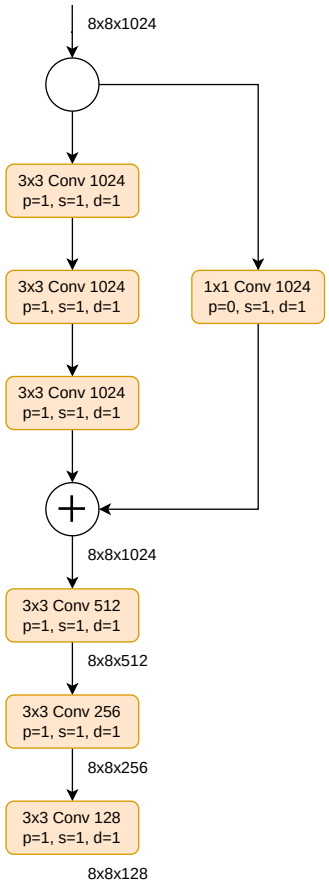


Figure 6. Last layers of the architecture before the Global Pooling Pyramids

5.4. Global Pooling Pyramids

Taking inspiration from Zhao et al.[16], to be able to capture spectral characteristics at multiple scales, the Global Average Pooling (GAP) and the Global Max Pooling (GMP) operators are ran at different kernel sizes:

- 1×1 Global Pooling
  Sums up each 8×8 feature map in one value (either the average or the highest one).
- 2×2 Global Pooling
  Sums up each 8×8 feature map in one value for each quadrant (either the average or the highest one).
- 4×4 Global Pooling
  Sums up each 8×8 feature map in one value for each quarter of every quadrant (either the average or the highest one).

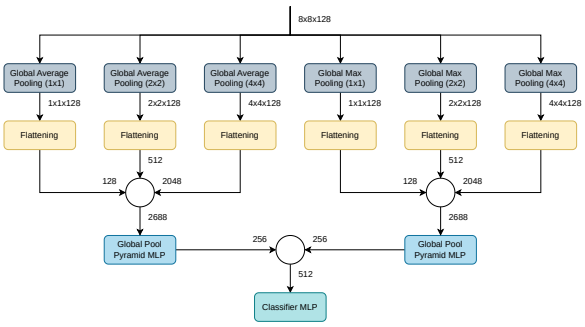


Figure 7. Global Pooling Pyramids in Model Architecture

These feature maps are then flattened in order to be concatenated and sent as an input to the respective Global Pooling Pyramid MLP (Multi-Layer Perceptron) shown in Figure 8.

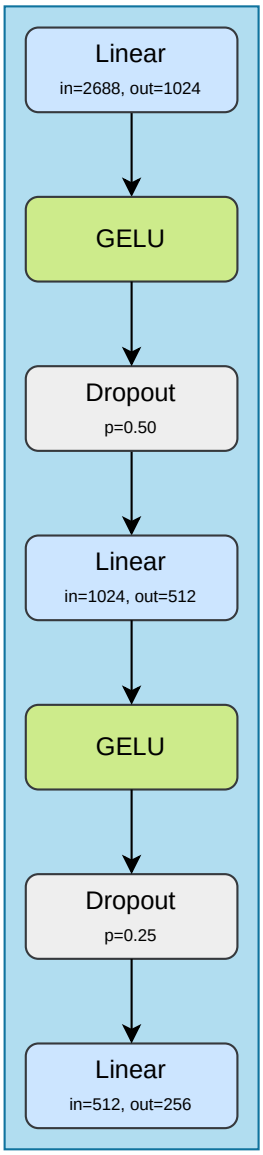


Figure 8. Global Pooling Pyramid MLP Architecture

The results coming from the Global Pooling Pyramid MLPs are concatenated to be passed as input to the final classifier.

5.5. Classifier

After the above block, a final MLP with three cascading linear layers along with their respective GELU activation function is added. It gradually compresses the feature vector from a length of 512 elements to 5 (the number of classes: real image and the four generative sources).

Dropout with a variable probability p is applied between Linear layers to prevent overfitting.

In Figure 9 the described MLP is depicted.

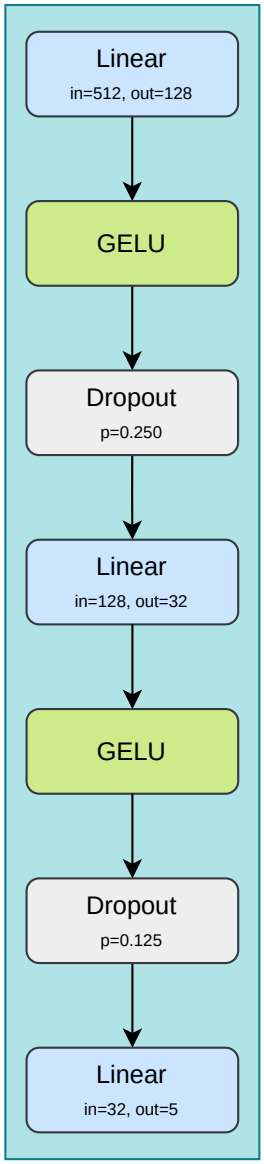


Figure 9. MLP for final classifier

5.6. Overall Model Architecture

The final custom architecture is represented in Figure 10, and is a composition of the various building blocks described above.

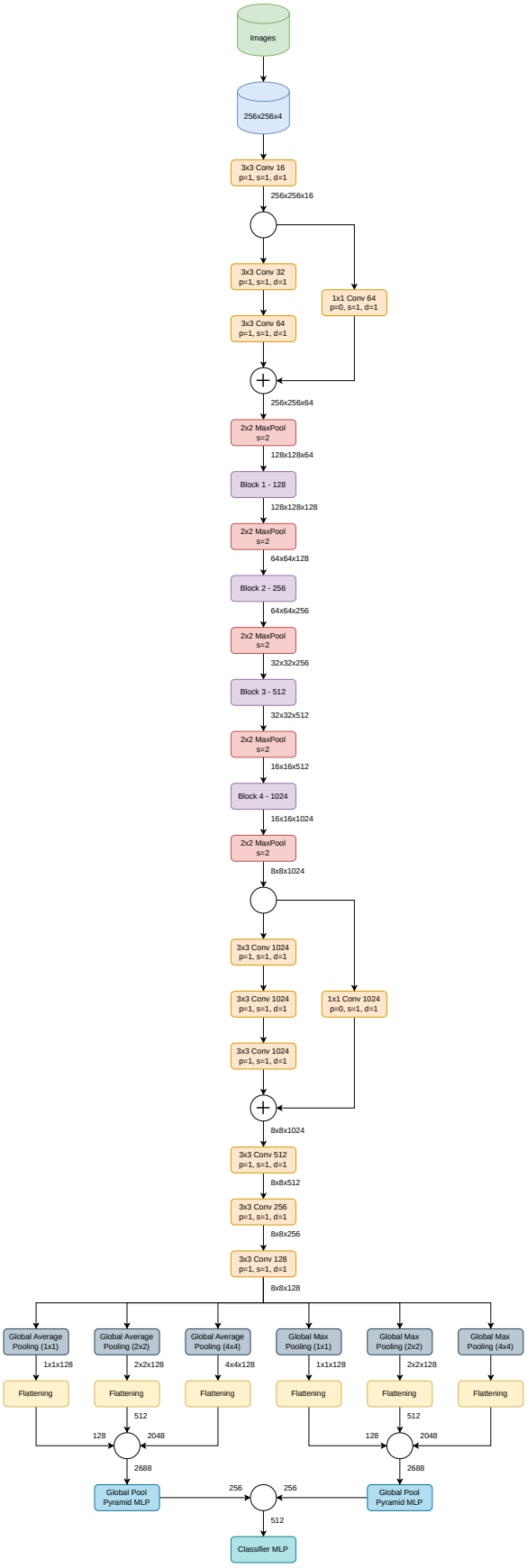


Figure 10. Overall Model architecture

6. Experiments

6.1. Experimental Setup

To guarantee robust results, we adopted a Stratified K-Fold Cross-Validation procedure with three folds for the training pipeline.

Some of the most important training parameters used in our experiment are:

- **Optimizer**: Adam;
- **Loss Function**: Cross Entropy;
- **Batch Size**: 32 images;
- **Training Precision**: `float32` with Automatic Mixed Precision (AMP) to use `bfloat16` when possible;
- **Epochs**: 100 Epochs implementing early stopping with a patience of 10 Epochs.

Additionally, a checkpointing mechanism has been employed to save both the best performing model (minimizing the validation loss) and the last trained model's weights for each Cross-Validation Fold.

6.2. Evaluation Strategy

In the testing phase an Ensemble Soft Voting strategy has been implemented.

6.2.1. Predictions

The predictions produced by the three folds are passed through a SoftMax operation and added together to then be averaged. The final class is assigned using ArgMax on the resulting averaged probabilities. This approach should reduce error variance and improve generalization.

The averaged SoftMax outputs are used in order to reduce the possibility of the final prediction being influenced too much by just one of the models of the ensemble.

6.2.2. Test Loss

To compute the test loss, the average logits of the predictions are used in order to be consistent with previous notations.

To guarantee the reliability of the evaluation, predictions and test loss were computed using the best checkpoint from each model instance.

6.3. Metrics

The final performance is evaluated using multiple metrics:

- Accuracy;
- Precision (per-class, with macro, micro and weighted averages);
- Recall (per-class, with macro, micro and weighted averages);
- F1-Score (per-class, with macro, micro and weighted averages);
- Per-class Confusion Matrices.

These metrics were chosen to investigate both general and per-class performances.

Given the fully balanced nature of our dataset, macro and weighted averages yield nearly identical results, as do micro averages with respect to the overall model accuracy.

6.4. Results

6.4.1. Loss

Looking at Figure 11a it is possible to observe both training and validation losses across the three folds.

During training the loss values decrease smoothly from initial values around 1.26 to values that vary between 0.6 and 0.35 depending on the fold. The stable downward trend suggests a stable optimization of the network during the training phase.

Occasional spikes in the validation loss are observed, along with a typical overfitting pattern starting around the 25^{th} epoch.

Soft Voting during the test run ensures better generalization, with respect to validation, as depicted by the lower loss value of the red line in the rightmost plot of Figure 11a.

6.4.2. Accuracy

The classification performance shown in Figure 11b represents the accuracy metric.

It is possible to observe that during training the accuracy metrics increase rapidly from values near 40% to 70%, then they improve steadily but more slowly up to 90%.

Validation accuracy, exhibiting more variance between the results, shows a clear difficulty in the multiclass attribution task.

Despite the validation metrics, test results do not drift a lot with respect to training performance, highlighting the effectiveness of the Ensemble Soft Voting technique.

6.4.3. Micro, Macro and Weighted Averaging

- **Micro**
  Micro averaging metrics - as anticipated - exhibit a similar trend to the one of the overall accuracy due to the total balancing of the dataset.
- **Macro**
  Macro averaging exhibits some form of instability during the validation run, further reinforcing the already mentioned difficulties in multiclass classification.
- **Weighted**
  Weighted averaging metrics show exactly the same results as Macro averaging, on account of the total balance of the dataset used.

These findings are shown in Figure 12.

6.4.4. Per Class Scores

While synthetic classes are more difficult to discern from one another as shown in Figure 13, real images are easily classified exhibiting substantially higher metrics (except for the class "`Generator 0`", which actually exhibits the highest performance metrics overall).

These results suggest strong deepfake detection performance on real images; however, there is still room for improvement in the model's ability to classify generator outputs.

These considerations are perfectly reflected in the Confusion Matrices.

Class "`Generator 0`" - DeepFloyd/IF-II-L-v1.0

The Class "`Generator 0`" is easily identified by the model with a good precision. As seen in Appendix A the generator has a strong digital fingerprint in the frequency domain, making it the only generator that benefits from a denoising process. This allows to better isolate meaningful patterns.

Class "`Generator 1`" - CompVis/stable-diffusion-v1-4

The Class "`Generator 1`" is a previous version of Stable Diffusion and it shares many characteristics with "`Generator 2`". Due to this, accuracy and F1 score metrics are slightly inferior with respect to the "`Real Image`" class since the classifiers has difficulties distinguishing between these models and others based on stable diffusion.

Class "`Generator 2`" - stabilityai/stable-diffusion-2-1-base

The Class "`Generator 2`" exhibits spectral patterns which contain crucial, but fragile information at very high frequencies. This is partially explained by the slight loss in performance shown in Appendix A with the denoising experiment. It is also possible to notice a difficulty in discerning between the Class "`Generator 1`" and this specific class, as explained in the previous analysis.

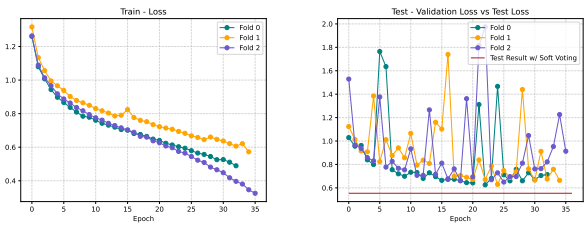
Class "`Generator 3`" - stabilityai/stable-diffusion-xl-base-1.0

Although the Class "`Generator 3`" belongs to the same family as "`Generator 1`" and "`Generator 2`", its performance is higher. This is likely because its underlying model differs slightly, making it easier for the classifier to distinguish between "`Generator 3`" and the others.

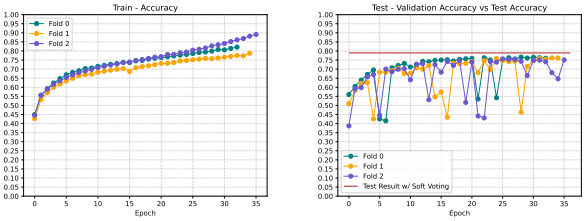
Class "`Real Image`"

The Class "`Real Image`" can be considered the most robust class in terms of generalization, showing minimal fold-to-fold variability and strong ensemble stability, making the model suitable for fake image detection.

6.4.5. Plots

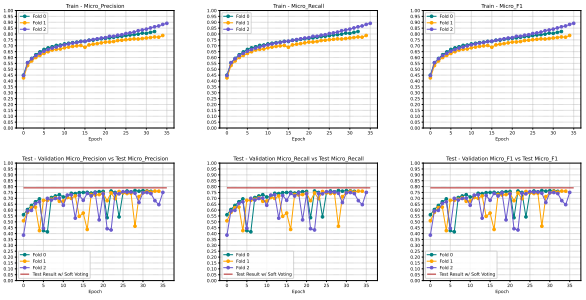


(a) Loss performances over epochs

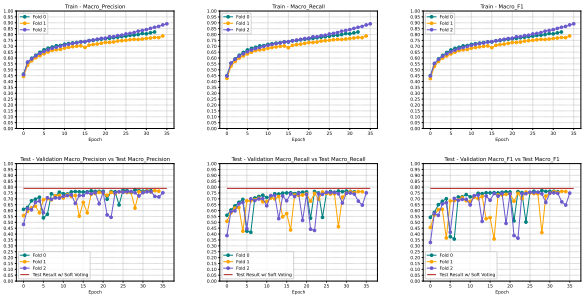


(b) Accuracy performances over epochs

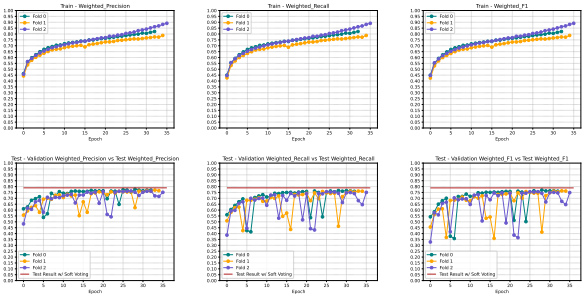
Figure 11. General metrics performances



(a) Micro performances over epochs

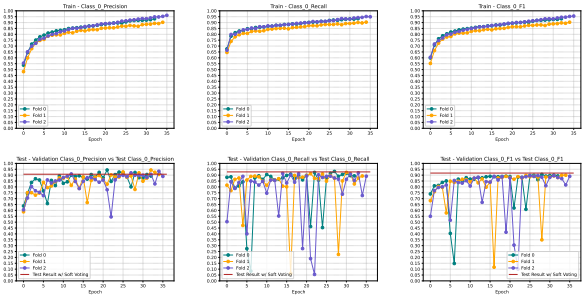


(b) Macro performances over epochs

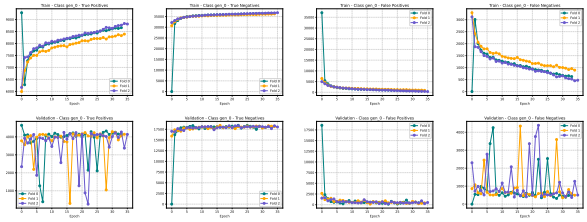


(c) Weighted performances over epochs

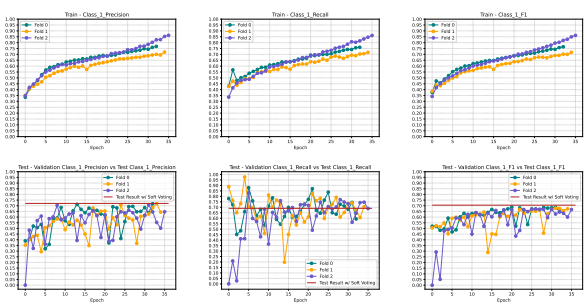
Figure 12. Weighted general metrics performances



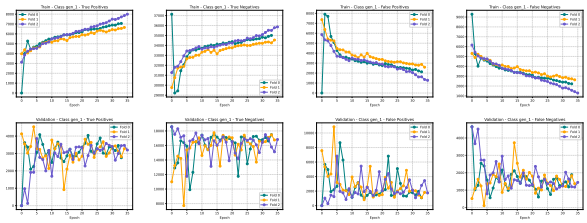
(a) Class “Generator 0” performances over epochs



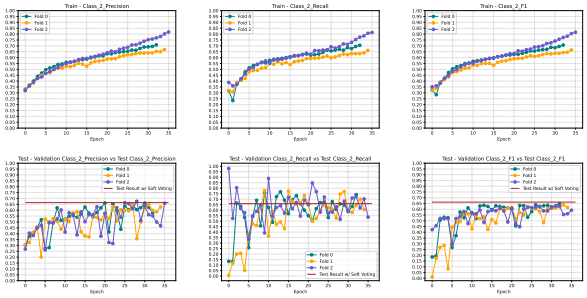
(b) Class “Generator 0” confusion matrices over epochs



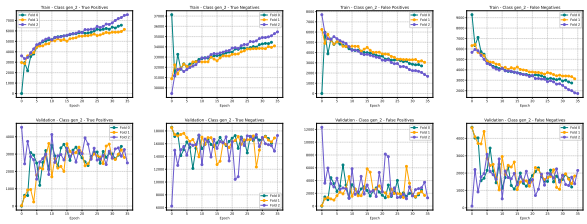
(c) Class “Generator 1” performances over epochs



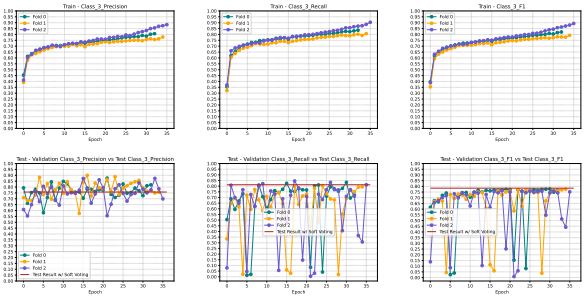
(d) Class “Generator 1” confusion matrices over epochs



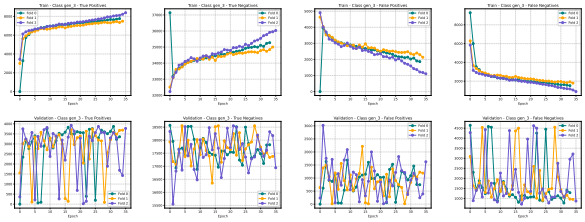
(e) Class “Generator 2” performances over epochs



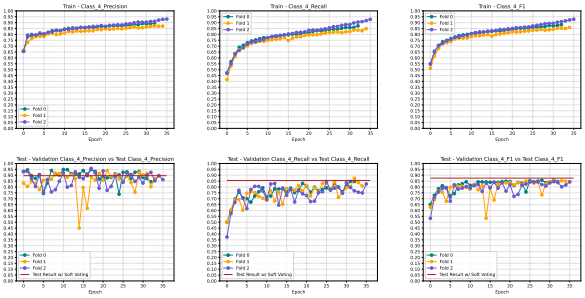
(f) Class “Generator 2” confusion matrices over epochs



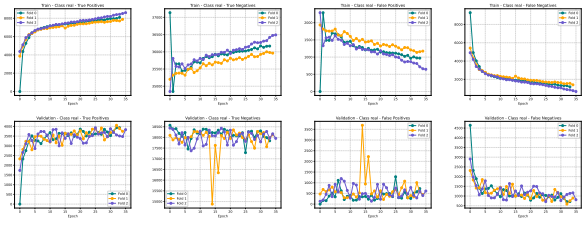
(g) Class “Generator 3” performances over epochs



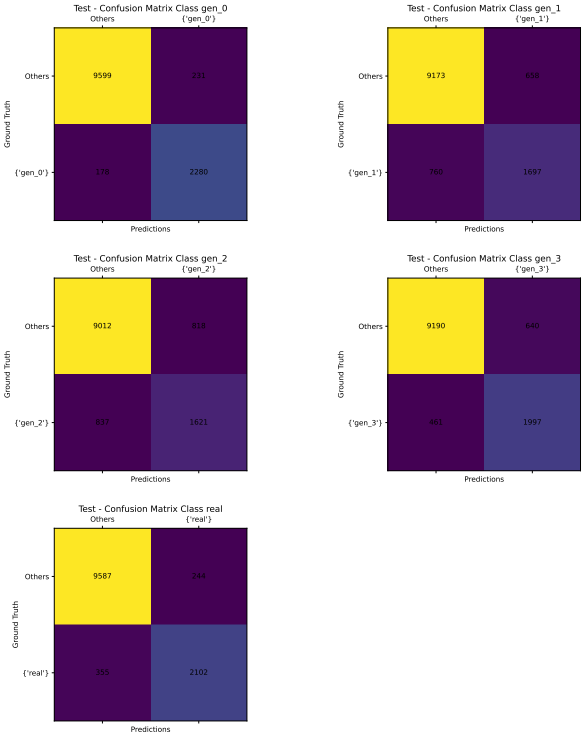
(h) Class “Generator 3” confusion matrices over epochs



(i) Class “Real Image” performances over epochs



(j) Class “Real Image” confusion matrices over epochs



(k) Confusion matrices in the Test phase

Figure 13. Per-class metrics performances

7. Conclusions

In conclusion, the Ensemble Classifier exhibits satisfactory performances especially when distinguishing between real and synthetic images. The difficulties in discerning between synthetic images coming from different generators but belonging to the same family - after observing steady improvements during the training and validation plots across epochs - could be partly attributed to the relatively small size of the working dataset.

7.1. Future Works

Future improvements could involve expanding the range of models to discriminate from, thereby increasing the number of classes. This would require augmenting the current dataset, but could yield more comprehensive results as newer and different generators are introduced. Moreover, an additional class “`Others`” could be implemented in order to not only recognize that the image is fake, but also that it is generated by an unknown model. Our experiment demonstrated that it is possible to work in the frequency domain to discriminate between real and fake images with satisfactory accuracy, even under suboptimal conditions.

Improving preprocessing techniques further could better the results of the network.

Attempts with a light denoising pass over the greyscale images were made. The aim was to preserve the majority of the high-frequency noise patterns in order to keep the discriminant parts of the generated images, but the results were similar to the ones showed above with no denoising applied (see Appendix A).

Another improvement could be to try and average batches

of real images belonging to one class, in an effort to try and enhance the common features between them. This last option has not been explored due to the small size of the dataset.

Finally, treating different semantics from the input channels separately could be another area to explore, thus increasing the model complexity and implying the necessity of a larger preprocessed portion of the Elsa_D3[13, 14] Dataset.

A. Note on Denoising Images

During our experment phase another preprocessing technique was also applied in the form of image denoising.

The denoising process has been achieved using OpenCV as follows; right before moving to the frequency domain:

```
denoised_img =
    ↪  cv2.fastNlMeansDenoising(
    src=resized_greyscale_img,
    dst=None,
    h=3.0,
    templateWindowSize=7,
    searchWindowSize=21
)
```

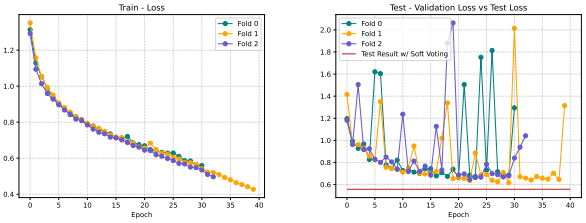
This approach should preserve most of the high-frequency noise characteristics typical of image generators.

Below are the results coming from a complete re-training of the same network with the denoised image dataset.

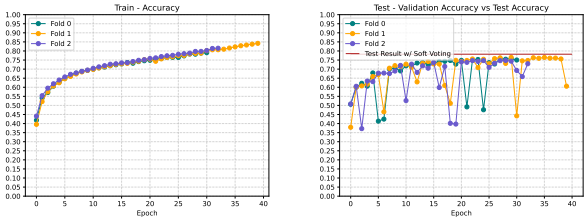
As shown in the following plots it is noticeable that the class of the Generator 0 is the only one that could benefit from the denoising process. This is probably due to examples coming from the Generator 0 being particularly noisy, and the denoising process helping in isolating significant patterns better.

Other classes either get marginal differences or - as is the case for the class of the Generator 2 - a more significant penalty, with the latter loosing in the F1-Score metric from the denoising process, probably due to wiping important information through the additional preprocessing step.

All of these observations could still be in the margin of error since their magnitude is in the $0.001\% - 0.01\%$ range.

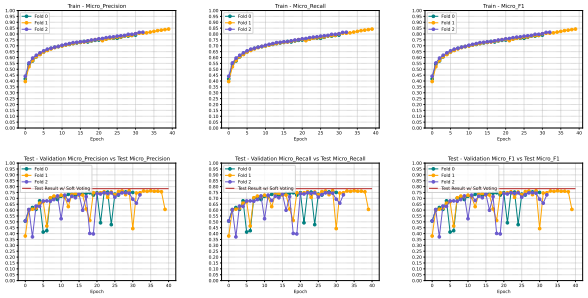


(a) Loss performances over epochs

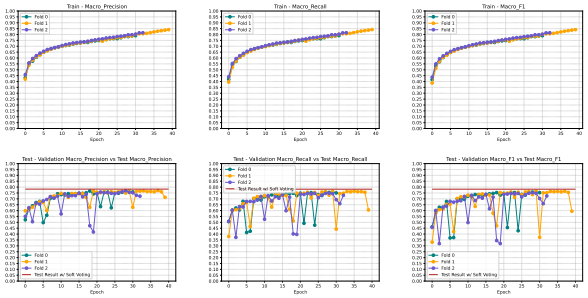


(b) Accuracy performances over epochs

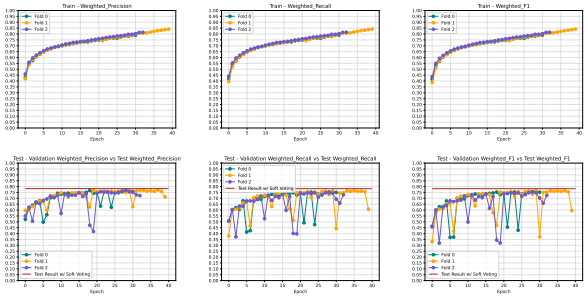
Figure 14. General metrics performances



(a) Micro performances over epochs

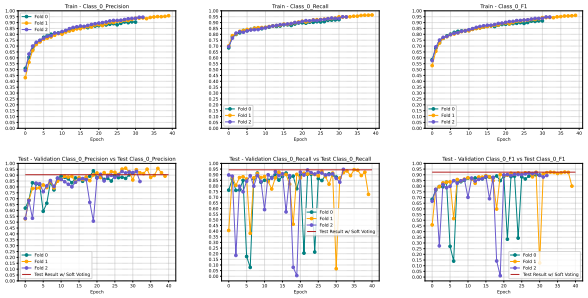


(b) Macro performances over epochs

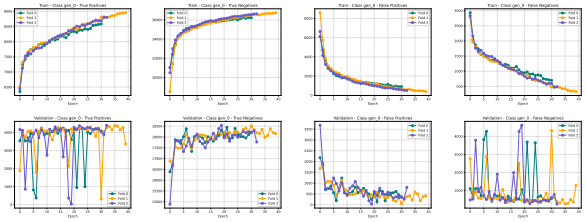


(c) Weighted performances over epochs

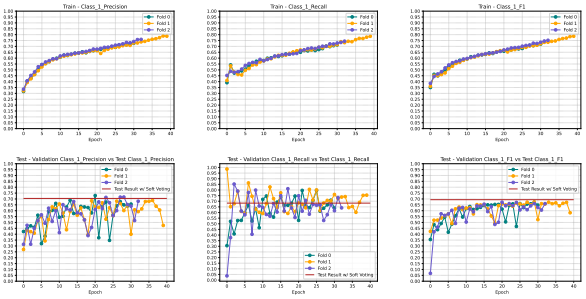
Figure 15. Weighted general metrics performances



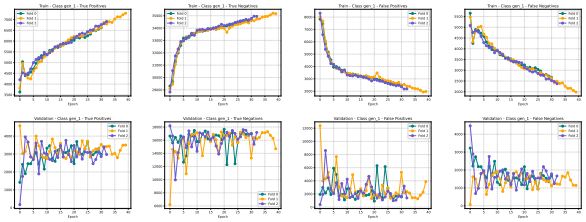
(a) Class "Generator 0" performances over epochs



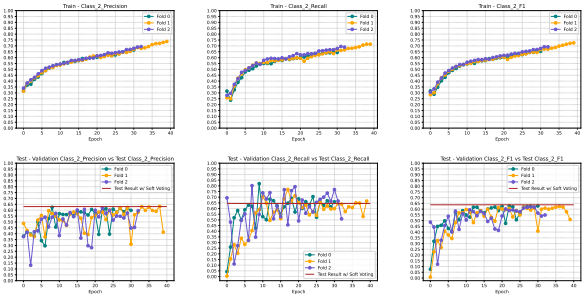
(b) Class “Generator 0” confusion matrices over epochs



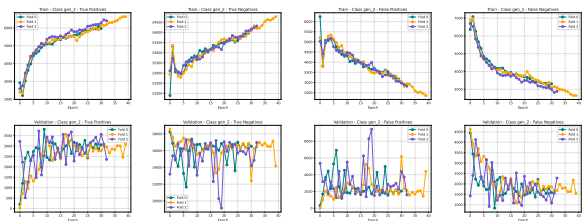
(c) Class “Generator 1” performances over epochs



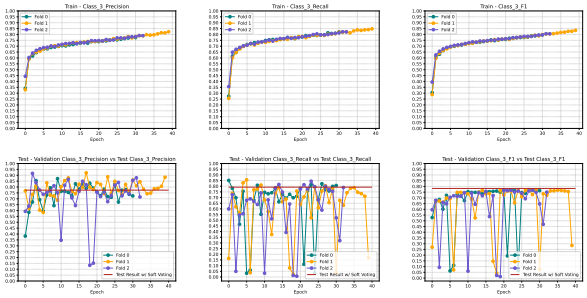
(d) Class “Generator 1” confusion matrices over epochs



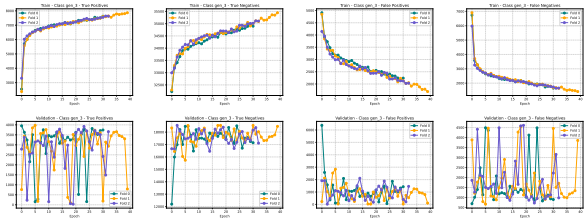
(e) Class “Generator 2” performances over epochs



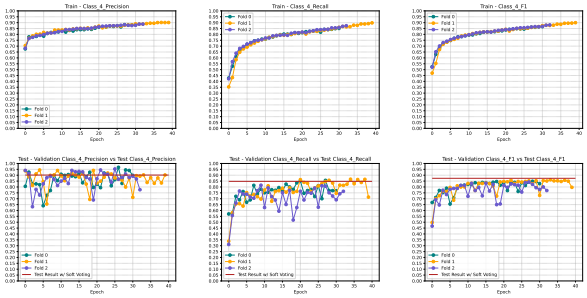
(f) Class “Generator 2” confusion matrices over epochs



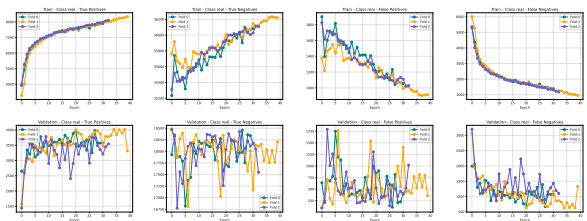
(g) Class “Generator 3” performances over epochs



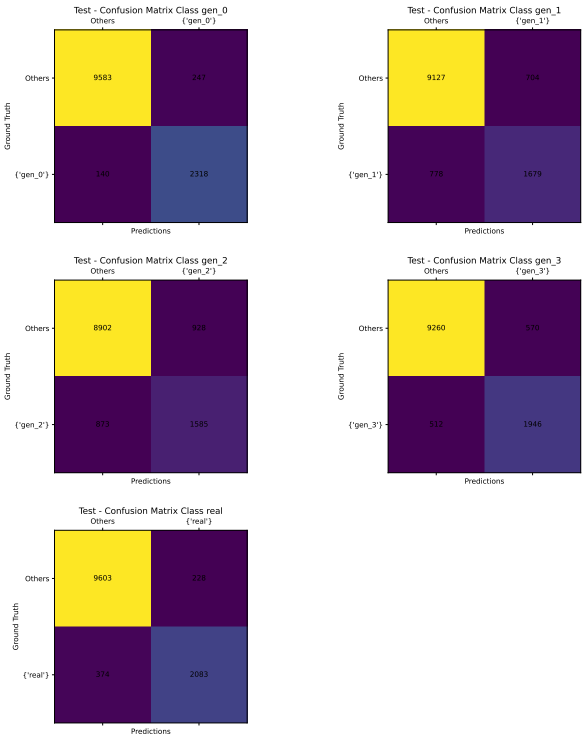
(h) Class “Generator 3” confusion matrices over epochs



(i) Class “Real Image” performances over epochs



(j) Class “Real Image” confusion matrices over epochs



(k) Confusion matrices in the Test phase

Figure 16. Per-class metrics performances

References

[1] Bruna Horvath - NBC News. Coca-Cola causes controversy with AI-made ad, 11 2024. Article available at: https://www.nbcnews.com/tech/innovation/coca-cola-causes-controversy-ai-made-ad-rcna180665.

[2] Elmira Aliieva - NBC News. Not lovin' it: McDonald's pulls AI-generated Christmas ad after social media backlash, 12 2025. Article available at: https://www.nbcnews.com/world/europe/mcdonalds-ai-generated-christmas-advert-social-media-backlash-rcna248590.

[3] Sophia Stewart - Publishers Weekly. Tor Books Criticized for Use of AI-Generated Art in "Gothikana" Cover Design, 02 2024. Article available at: https://www.publishersweekly.com/pw/by-topic/industry-news/publisher-news/article/94388-tor-books-imprint-bramble-criticized-for-use-of-ai-generated-art-in-gothikana-cover-design.html.

[4] Moin Roberts-Islam - Forbes. Vogue's AI-Generated Models Spark Reader Fury And Industry Panic, 07 2025. Article available at: https://www.forbes.com/sites/moinroberts-islam/2025/07/29/vogue-erupts-ai-generated-models-spark-reader-fury-and-industry-panic/.

[5] Laura Gozzi - BBC. AI Brad Pitt dupes French woman out of €830,000, 01 2025. Article available at: https://www.bbc.com/news/articles/ckgnz8rw1xgo.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[7] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 2020.

[8] Andreas Rössler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Nießner. Faceforensics++: Learning to detect manipulated facial images, 2019.

[9] Sheng-Yu Wang, Oliver Wang, Richard Zhang, Andrew Owens, and Alexei A. Efros. Cnn-generated images are surprisingly easy to spot... for now, 2020.

[10] Joel Frank, Thorsten Eisenhofer, Lea Schönherr, Asja Fischer, Dorothea Kolossa, and Thorsten Holz. Leveraging frequency analysis for deep fake image recognition, 2020.

[11] Ricard Durall, Margret Keuper, and Janis Keuper. Watch your up-convolution: Cnn based generative deep neural networks are failing to reproduce spectral distributions, 2020.

[12] Utkarsh Ojha, Yuheng Li, and Yong Jae Lee. Towards universal fake image detectors that generalize across generative models, 2024.

[13] Lorenzo Baraldi, Federico Cocchi, Marcella Cornia, Lorenzo Baraldi, Alessandro Nicolosi, and Rita Cucchiara. Contrasting deepfakes diffusion via contrastive learning and global-local similarities, 2024.

[14] AImageLab - UniMORE. ELSA_D3 Dataset, 07 2024. Dataset available at: https://huggingface.co/datasets/elsaEU/ELSA_D3.

[15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014.

[16] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network, 2017.