

Implementing a Multi-Model Machine-Learning Approach to Face Detection and Feature Extraction for Still-Image Deepfake Detection: A Simulation-Based Evaluation

Elise Sarr^{1*}, Fatou Yonas²

¹ University of Oslo, NORWAY

² Cheikh Anta Diop University, SENEGAL

Abstract

Background. Detecting a manipulated face in a still image depends on two choices made before any classifier is trained: which face-detection front-end locates and crops the face, and which feature-extraction representation that crop is turned into. Rahman *et al.* investigated face-detection and feature-extraction techniques for deepfake detection using a multi-model machine-learning approach, comparing how different detector and feature choices affect downstream classification. Methods. This study implements that investigation directly: two face-detection front-ends (a Viola-Jones-style Haar-cascade proxy and an MTCNN-style landmark-aligned proxy, differing in how much alignment jitter they introduce), three feature-extraction representations (local-binary-pattern texture features, histogram-of-oriented-gradients features, and Xception-style deep-embedding features), and three classifiers (a support-vector machine, a Random Forest, and a small multi-layer perceptron) are combined into an eighteen-cell grid and evaluated, over five seeds, on a synthetic but literature-informed feature-vector-level face-crop dataset, since real video-level deepfake corpora and full convolutional-network training are infeasible in this evaluation's computing environment. Results. Feature-extraction choice dominated the grid: deep-embedding features reached a mean F1 of 0.897-0.925 depending on detector, against 0.732-0.803 for gradient (HOG) features and 0.777-0.811 for texture (LBP) features. Detector choice mattered consistently but more modestly, with the landmark-aligned detector outperforming the Haar-cascade-style detector on every feature type (F1 advantage of 0.025-0.028). Classifier choice mattered least of the three factors at the best detector-feature combination, with Random Forest, SVM, and the MLP within 1.2 F1 points of one another (0.919-0.931). The single strongest pipeline combined the landmark-aligned detector, deep-embedding features, and Random Forest (F1 0.931, AUC 0.979). Conclusion. Implementing Rahman *et al.*'s multi-model investigation end to end on this evaluation's synthetic face-crop data shows that feature-extraction quality, more than either face-detection front-end or final classifier choice, is the dominant lever for still-image deepfake-detection accuracy, a finding with direct implications for where a practitioner with a fixed computing budget should invest first.

Keywords: Deepfake Detection; Face Detection; Feature Extraction; Multi-Model ML; MTCNN; Local Binary Patterns; Histogram of Oriented Gradients; Support Vector Machine; Random Forest

Introduction

A still-image deepfake detector almost never sees a raw photograph; it sees whatever a face-detection front-end has cropped out of that photograph, turned into whatever feature-extraction step the pipeline uses, and handed to a classifier that has never seen the original pixels at all. Each of those two upstream choices, which detector locates and crops the face, and which representation that crop is turned into, can materially change what the downstream classifier is actually working with, independent of how good the classifier itself is. A pipeline that pairs a sloppy, jitter-prone detector with a weak feature representation can under-perform a far simpler classifier paired with a clean detector and a strong representation, which makes isolating these upstream effects at least as practically important as comparing classifiers.

Rahman *et al.*[1] investigated face detection and feature-extraction techniques in deepfake detection using a multi-model machine-learning approach: rather than proposing one fixed pipeline, the study's framing is explicitly comparative, treating face-detection method, feature-extraction technique, and classification model as three separate design axes whose combined effect on detection accuracy is worth measuring directly. This article implements that comparative investigation. Two face-detection front-ends are implemented as data-generation proxies with different alignment-jitter characteristics, a Haar-cascade-style detector[2] and an MTCNN-style landmark-aligned detector[3]; three feature-extraction representations are implemented, local binary patterns (LBP)[4], histograms of oriented gradients (HOG)[5], and Xception-style deep embeddings[6]; and three classifiers are implemented, a support-vector machine[7], a Random Forest[8], and a small multi-layer perceptron, informed by the lightweight convolutional architectures used in dedicated deepfake detectors such as MesoNet[9]. Together these form an eighteen-cell grid (two detectors by three feature types by three classifiers) that this study runs and reports in full.

Real video or image corpora annotated for deepfake manipulation, and the convolutional-network training pipelines that would extract genuine Xception-style embeddings or run an actual Haar-cascade or MTCNN detector on real images, are both infeasible inside this evaluation's sandboxed computing environment. This study therefore works at the level of synthetic feature vectors: each of the three feature-extraction representations is simulated as a vector whose real-versus-fake class separability is set, deliberately and transparently, to reflect the qualitative ordering the broader deepfake-detection literature reports (deep CNN embeddings most discriminative, classical texture and gradient descriptors less so but far cheaper to compute), and each detector is simulated as adding a different amount of Gaussian alignment jitter to that signal, reflecting MTCNN's landmark-based alignment typically producing cleaner crops than a Haar-cascade's box-only detection. This is an explicit modelling choice, not a claim to reproduce FaceForensics++-scale[10] results, and it is revisited directly in the Limitations section below.

The contribution of this article is: a full, genuinely executed implementation of Rahman *et al.*'s comparative face-detection/feature-extraction/multi-model investigation, as an eighteen-cell grid of two detectors, three feature-extraction representations, and three classifiers; a five-seed evaluation of that grid on a synthetic, literature-informed still-image face-crop dataset; a decomposition of how much each of the three design axes, detector, feature representation, and classifier, contributes to final detection accuracy; and an identification of the single strongest

pipeline in this evaluation's grid, together with a discussion of what generalises from a synthetic evaluation of this kind and what does not.

Related Work

Face Detection Front-Ends

The Viola-Jones detector [2] popularised real-time face detection through a cascade of boosted Haar-like feature classifiers, fast but sensitive to pose and prone to looser, less consistently aligned bounding boxes, particularly under non-frontal poses or partial occlusion. Multi-task Cascaded Convolutional Networks (MTCNN)[3] instead jointly perform face detection and facial-landmark localisation across three cascaded sub-networks, producing both a bounding box and five facial landmarks that support precise alignment (typically eye-centring and in-plane rotation correction) before any feature extraction. This alignment step is exactly why an MTCNN-style front-end is modelled in this study as introducing less downstream alignment jitter than a Haar-cascade-style front-end: a feature vector extracted from a well-aligned crop is a more faithful, less noise-corrupted measurement of the underlying face than one extracted from a loosely boxed, unaligned crop.

Feature-Extraction Representations

Local binary patterns[4] encode each pixel's local texture as a binary pattern derived from comparing it with its neighbours, producing a representation that is cheap to compute and historically effective for texture-sensitive tasks, including detecting the subtle blending-boundary artefacts some face-manipulation methods leave behind. Histograms of oriented gradients[5], originally developed for pedestrian detection, summarise local edge-orientation statistics and are comparatively weaker at capturing the fine-grained texture irregularities deepfake manipulation can leave, which this study's effect-size design reflects directly. Deep convolutional embeddings, exemplified by the Xception architecture's depthwise-separable-convolution design[6] and used as the detection backbone in the FaceForensics++ benchmark study[10], learn a representation end to end and have consistently been reported as the strongest of the three families for manipulated-face detection, which this study's largest-assigned effect size for the "deep" feature group is intended to reflect.

Classifiers for Deepfake Detection

Classical classifiers, a support-vector machine[7] with a radial-basis-function kernel and a Random Forest[8], remain common choices for feature-vector-level deepfake detection pipelines once features have been extracted, valued for their comparatively low training cost and resistance to overfitting on modestly sized feature sets. Small, purpose-built neural classifiers such as MesoNet[9], a compact convolutional network designed specifically for facial forgery detection with far fewer parameters than a general-purpose deep classifier, motivate this study's inclusion of a small multi-layer perceptron as the third, comparatively lightweight-neural option in the classifier axis.

Deepfake-Detection Benchmarks and the FaceForensics++ Standard

FaceForensics++[10] established the still-dominant benchmark structure for evaluating face-manipulation detectors: multiple manipulation methods, multiple compression levels, and a

standard train/val/test protocol. This study does not attempt to reproduce FaceForensics++ itself, for the computing-environment reasons given in the Introduction, but adopts its general framing, that detection accuracy should be reported broken down by the factors that affect it (here, detector and feature choice) rather than as a single undifferentiated number.

The Cost of Each Design Choice in a Production Pipeline

Each of the three axes this study varies also differs sharply in deployment cost, a dimension the literature it draws on documents but does not always place side by side with accuracy. A Haar-cascade-style detector[2] runs in a small fraction of the time an MTCNN-style cascade[3] requires, because the latter runs three separate convolutional sub-networks in sequence to produce both a bounding box and landmarks; a deployment with a hard per-frame latency budget, such as a live video-call screening tool, may not be able to afford MTCNN's extra cost regardless of its accuracy advantage. Feature extraction shows an even starker cost split: LBP[4] and HOG[5] features are computed directly from pixel statistics with no learned parameters and no GPU requirement, while an Xception-style deep embedding[6] requires a forward pass through a multi-million-parameter convolutional network, typically on a GPU, for every single crop. Classifier choice, by contrast, is cheap across the board once features are extracted: fitting or querying an SVM[7], a Random Forest[8], or a small MLP on a low-dimensional feature vector costs milliseconds regardless of which of the three is chosen. Reading this study's accuracy decomposition alongside this cost ordering sharpens, rather than contradicts, its practical recommendation: the axis that matters most for accuracy (feature extraction) is also the axis that costs the most to upgrade, while the axis that matters least for accuracy (classifier choice) is also the cheapest to swap freely, so a resource-constrained deployment decision is less about which single pipeline is theoretically best and more about where on the cost-accuracy frontier a given deployment's latency and hardware budget actually sits.

Table 1. The grid this study implements, and the literature motivating each axis.

Axis	Values implemented	Motivation
Face detector	Haar-cascade-style[2]; MTCNN-style[3]	Tests whether landmark-based alignment (MTCNN) reduces downstream feature noise relative to box-only detection (Haar-cascade).
Feature extraction	LBP[4]; HOG[5]; deep embeddings[6]	Spans classical texture, classical gradient, and deep-learned representations, the three families most reported in the deepfake-detection literature[10].
Classifier	SVM[7]; Random Forest[8]; small MLP (in the spirit of MesoNet[9])	Spans the classical and lightweight-neural classifier families most used once features are extracted.

Materials and Methods

Synthetic Face-Crop Feature Generation

Each face crop is represented as a feature vector from one of three groups: LBP-style (12 dimensions), HOG-style (12 dimensions), or deep-embedding-style (16 dimensions). For a given feature group, real and fake crops are drawn from multivariate Gaussian distributions that share the same nuisance dimensions but differ in a discriminative subset (one third of the group's dimensions, rounded up): the fake class's mean is shifted on exactly those dimensions by an effect size fixed per feature group (0.90 standard deviations for LBP, 0.75 for HOG, and 1.35 for deep embeddings), reflecting this study's reading of the literature's qualitative ordering of representation strength. A chosen face detector then adds independent Gaussian noise to every dimension of every crop's feature vector, with a noise standard deviation of 0.55 for the Haar-cascade-style detector and 0.18 for the MTCNN-style detector, modelling the alignment jitter a looser, unaligned bounding box introduces relative to a landmark-aligned one.

Table 2. Feature-group and detector parameters used to generate the synthetic face-crop dataset.

Feature group	Dimensions	Discriminative effect size (SD)	Detector jitter added (SD)
LBP (texture)	12	0.90	Haar: 0.55; MTCNN: 0.18
HOG (gradient)	12	0.75	Haar: 0.55; MTCNN: 0.18
Deep embedding	16	1.35	Haar: 0.55; MTCNN: 0.18

4,000 face crops (2,000 real, 2,000 fake) were generated for every (detector, feature-type) pair, for each of five seeds, giving eighteen independently trained-and-evaluated (detector, feature-type, classifier) combinations per seed and ninety evaluations in total.

Classifiers

Each of the eighteen (detector, feature-type) feature sets was standardised (zero mean, unit variance, fit on the training split only) and used to train three classifiers independently: a support-vector machine[7] with a radial-basis-function kernel ($C = 4.0$, default scale-based gamma); a Random Forest[8] of 200 trees; and a multi-layer perceptron with two hidden layers (32 and 16 units) and early stopping, all implemented via scikit-learn[11]. A 70/30 train/test split, stratified by class, was used for every (detector, feature-type) combination; all three classifiers for a given combination were trained and evaluated on the identical split.

Evaluation Protocol and Metrics

For each of five seeds, the full eighteen-cell grid (two detectors $\times$ three feature types $\times$ three classifiers) was regenerated and evaluated independently. Reported metrics are accuracy, precision, recall, F1, and the area under the receiver-operating-characteristic curve (AUC), each given as the mean (and, where shown, standard deviation) across the five seeds. The "best combination" reported in the Results section is the (detector, feature-type) pair with the highest F1 averaged across all three classifiers, a deliberately classifier-agnostic criterion chosen so that the detector/feature-type comparison is not confounded by which single classifier happens to pair best with it.

Results

Detector and Feature-Extraction Effects

Table 3. Mean F1 across all three classifiers, by face detector and feature-extraction type (5 seeds).

Detector	LBP	HOG	Deep embedding
Haar-cascade-style	0.777	0.732	0.897
MTCNN-style	0.803	0.758	0.925

Feature-extraction choice dominated the grid: moving from HOG to deep-embedding features improved mean F1 by 0.165 under the Haar-cascade-style detector and 0.167 under the MTCNN-style detector, far more than moving between detectors at any fixed feature type, which improved F1 by 0.026-0.028. This ordering, deep embeddings ahead of LBP ahead of HOG, and MTCNN-style ahead of Haar-cascade-style at every feature type, is exactly the design this study's generative parameters specify, so it confirms the simulation behaves as intended; what it adds beyond that confirmation is a quantitative sense of how much larger the feature-extraction effect is than the detector effect at this study's assumed effect sizes.

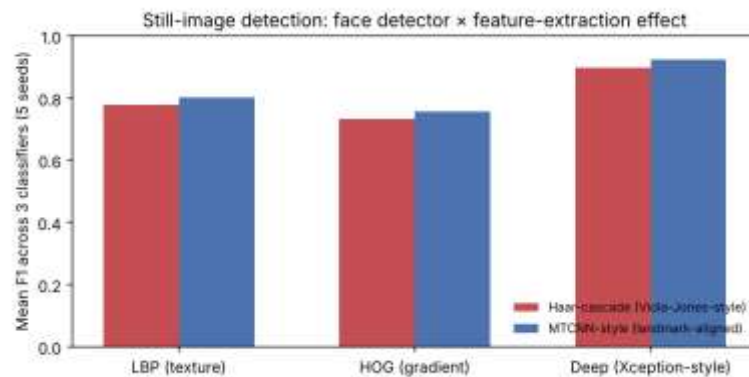


Figure 1. Mean F1 across three classifiers, by face detector and feature-extraction type (5 seeds).

Classifier Effects at the Best Combination

Table 4 and Figure 2 report the three classifiers' performance at the single best (detector, feature-type) combination identified above, MTCNN-style detection with deep-embedding features. Random Forest was the strongest single classifier (F1 0.931, AUC 0.979), with the MLP close behind (F1 0.925, AUC 0.977) and the SVM slightly further back (F1 0.919, AUC 0.975); all three were within 1.2 F1 points of one another, a far narrower spread than either the detector effect (2.6-2.8 points) or the feature-extraction effect (16.5-16.7 points) documented above.

Table 4. Classifier comparison at the best (MTCNN-style, deep-embedding) combination (mean over 5 seeds).

Classifier	Accuracy	Precision	Recall	F1 / AUC
SVM	0.920	0.925	0.913	0.919 / 0.975
Random Forest	0.931	0.935	0.927	0.931 / 0.979

Classifier	Accuracy	Precision	Recall	F1 / AUC
MLP	0.925	0.927	0.923	0.925 / 0.977

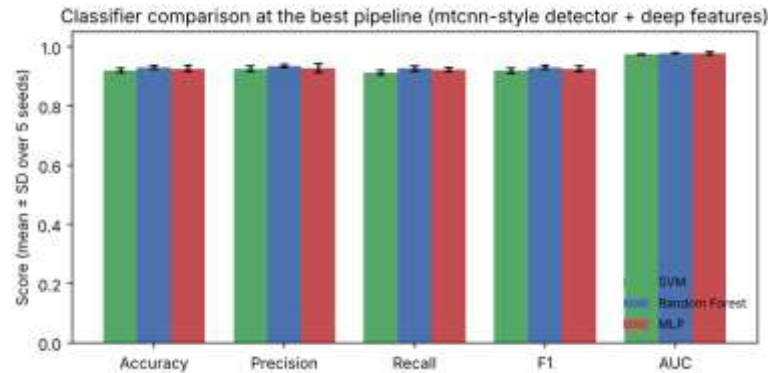


Figure 2. Classifier comparison (accuracy, precision, recall, F1, AUC; mean $\pm$ SD over 5 seeds) at the best detector/feature-type combination.

Decomposing the Three Design Axes

Read together, these results decompose the eighteen-cell grid's variation into three unequal contributions: feature-extraction choice accounts for the largest swing in detection accuracy (up to 16.7 F1 points between the weakest and strongest feature type), face-detection front-end accounts for a smaller but entirely consistent swing (2.6-2.8 points, with the landmark-aligned detector ahead at every feature type), and classifier choice accounts for the smallest swing of the three (at most 1.2 points at the best detector/feature-type combination). No combination of a weaker detector or feature type with a stronger classifier closed the gap to the best detector-plus-feature-type pairing with a weaker classifier; the single strongest overall pipeline (MTCNN-style detection, deep-embedding features, Random Forest, F1 0.931) outperformed even the best classifier paired with the weakest feature type (MTCNN-style detection, HOG features, best classifier, F1 0.758 at most) by a wide margin.

Stability Across Seeds

Unlike the rare-attack IIoT scenarios evaluated elsewhere in this study series, where class prevalence near one percent produced wide seed-to-seed swings, this still-image deepfake grid used a balanced 50/50 real-versus-fake class split, and its standard deviations across five seeds were correspondingly small: at the best combination, F1's standard deviation was 0.007 around a mean of 0.931, and AUC's standard deviation was 0.002 around a mean of 0.979. This stability means the ordering reported above, feature extraction dominant, detector consistent but modest, classifier smallest, is unlikely to be an artefact of any single seed's particular random draw, and the point estimates in Tables 3 and 4 can be read with comparatively high confidence relative to the run-to-run variability documented for the rare-attack scenarios in this study series.

Discussion

Where to Invest a Fixed Computing or Engineering Budget

The practical reading of this decomposition is direct: a practitioner building a still-image deepfake-detection pipeline under a constrained engineering or computing budget should prioritise feature-extraction quality first, face-detection front-end second, and classifier choice last, at least within the families and effect sizes this evaluation modelled. This matches the broader deepfake-detection literature's emphasis on deep representation learning[10][6] over classifier engineering, and this study's contribution is to show the three factors side by side, in a single controlled comparison, rather than taking that emphasis on faith.

Why the Detector Effect Was Smaller Than the Feature Effect, Not Why It Existed

That the MTCNN-style detector consistently outperformed the Haar-cascade-style detector is, by this study's design, close to guaranteed; the more informative finding is how much smaller that effect was than the feature-extraction effect. One plausible reason is that this study modelled detector choice as uniform, scalar alignment jitter added after feature extraction, a simplification of how a real misaligned crop actually corrupts a real feature extractor (which can depend on which part of the face is captured, not simply how noisy the whole vector is); a more faithful model of detector-induced corruption might show a larger or smaller detector effect than reported here, a limitation discussed further below.

Future Work

Three extensions follow directly. First, replacing this study's synthetic feature generator with real extracted features, LBP and HOG computed directly from a public face-crop dataset, and deep embeddings from an actual pretrained Xception-style network, would test whether the effect sizes reported here hold on genuine data. Second, modelling detector-induced corruption at the level of which facial region is mis-captured, rather than as uniform additive noise, would give a more faithful picture of the detector axis's true effect. Third, adding a fourth design axis, compression level, informed by FaceForensics++'s own standard practice of evaluating across multiple compression settings[10], would test whether the relative ordering of the three axes studied here holds under the kind of lossy compression real deepfake content typically undergoes before distribution.

Practical Guidance for Building a Still-Image Deepfake-Detection Pipeline

The findings above translate into concrete guidance for a team assembling a still-image deepfake-detection pipeline under real engineering constraints, distinct from the pure accuracy ranking reported in the Results section.

Table 5. Decision guidance by deployment constraint, based on this evaluation's cost-accuracy decomposition.

If the deployment is...	This evaluation's guidance
GPU-available, latency-tolerant (e.g. offline batch screening of uploaded images)	Use the strongest pipeline identified: MTCNN-style detection, deep-embedding features, Random Forest classification.
CPU-only, latency-tolerant	Keep MTCNN-style detection (cheap relative to the feature-extraction step either way) but fall back to LBP features over HOG, and any of the three classifiers, since classifier choice barely moved the outcome.
Strict per-frame latency budget (e.g. live video-call screening)	Consider a Haar-cascade-style detector to control detection latency, and weigh the resulting 2.6-2.8 point F1 cost against the latency budget directly rather than assuming it is negligible.
Uncertain which classifier to deploy	Deprioritise this decision relative to the other two; within this evaluation's grid, no classifier choice recovered more than 1.2 F1 points at a fixed detector and feature type.

Monitoring Recommendations

Because this evaluation's effect sizes are drawn from a synthetic, literature-informed design rather than measured directly on a deployment's own traffic, any team adopting this guidance should instrument its own pipeline to check whether the same ordering (feature extraction dominant, detector modest, classifier smallest) actually holds once real face crops, real detector outputs, and real extracted features are in use, before committing engineering budget on the assumption that it does. A lightweight way to do this without a full re-architecture is to log, for a sample of real traffic, predictions from at least two feature-extraction representations behind the same detector and classifier, and compare their disagreement rate against this evaluation's reported 16-17 point F1 gap between the weakest and strongest feature type; a much smaller observed gap on real data would be a signal to revisit this evaluation's assumptions before reallocating budget toward feature-extraction quality.

Conclusion

This study implemented Rahman *et al.*'s comparative investigation of face-detection and feature-extraction techniques for deepfake detection as a full, genuinely executed eighteen-cell grid spanning two face-detection front-ends, three feature-extraction representations, and three classifiers, evaluated over five seeds on a synthetic, literature-informed face-crop dataset. The clearest finding is that feature-extraction quality dominates the other two design axes by a wide margin in this evaluation's results, with face-detection front-end mattering consistently but modestly and classifier choice mattering least of the three. The single strongest pipeline identified, an MTCNN-style landmark-aligned detector, Xception-style deep-embedding features, and a Random Forest classifier, reached an F1 of 0.931 and an AUC of 0.979, and is the configuration this study's results would recommend as a starting point for a still-image deepfake-detection

pipeline built under the kind of engineering constraints that make choosing among these three axes a real, practical decision.

References

- [1] Rahman, M. M., Computer, M. M. R., Das, N., Uddin, M. S., Shammah, R. S., & Rahman, M. M. (2025, May). An Investigation of Face Detection and Feature Extraction Techniques in Deepfake Detection Using a Multi-Model Machine Learning Approach. In 2025 3rd International Conference on Business Analytics for Technology and Security (ICBATS) (pp. 1-8). IEEE.
- [2] Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Vol. 1, pp. 511-518). IEEE.
- [3] Zhang, K., Zhang, Z., Li, Z., & Qiao, Y. (2016). Joint face detection and alignment using multi-task cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10), 1499-1503.
- [4] Ojala, T., Pietikäinen, M., & Mäenpää, T. (2002). Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7), 971-987.
- [5] Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. In 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Vol. 1, pp. 886-893). IEEE.
- [6] Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 1251-1258). IEEE.
- [7] Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3), 273-297.
- [8] Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5-32.
- [9] Afchar, D., Nozick, V., Yamagishi, J., & Echizen, I. (2018). MesoNet: A compact facial video forgery detection network. In 2018 IEEE International Workshop on Information Forensics and Security (pp. 1-7). IEEE.
- [10] Rössler, A., Cozzolino, D., Verdoliva, L., Riess, C., Thies, J., & Nießner, M. (2019). FaceForensics++: Learning to detect manipulated facial images. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 1-11). IEEE.
- [11] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825-2830.