DeepFake Detection via Multi-Modal Deep Learning Architectures: A Comparative Analysis

43729 48032 47914

Abstract

With the growing progress in synthetic image generation and manipulation techniques, namely Generative Adversarial Networks (GANs), the detection of deepfakes has become a critical challenge in maintaining digital authenticity. This work presents a comprehensive evaluation of deep learning architectures for distinguishing authentic and synthetic facial images, focusing on 3 approaches: (1) a YOLO-assisted convolutional neural network (CNN), (2) transfer learning with Inception-ResNet V2, and (3) ShuffleNet V2. Our analysis reveals that pretrained models achieved better accuracy, with Inception-ResNet V2 outperforming ShuffleNet V2 and the custom CNN baseline. However, ShuffleNet offered faster inference with a minimal accuracy drop. YOLO-based preprocessing demonstrated no significant impact, suggesting region-of-interest (ROI) cropping may be redundant in architectures with inherent spatial attention and establishes transfer learning with architectural optimisation, rather than preprocessing pipelines, as the driver of state-of-the-art performance. With Grad-CAM visualisations providing interpretability for model decisions, these insights lay out a blueprint for deploying explainable deepfake detection mechanisms in real-world scenarios.

1. Introduction

In the digital age, social media platforms have become ubiquitous conduits of information, shaping global discourse and individual worldviews. While these networks democratize access to knowledge and foster connectivity, they also enable novel threats to informational integrity, most notably through AI-generated synthetic media termed 'deepfakes.' These algorithmically crafted videos and images, often indistinguishable from authentic content, increasingly distort public perception of individuals, communities, and critical socio-political issues.

"Deepfakes are videos, picture or audio clips made with artificial intelligence to look real." (BBC Newsround, 2024)

While deepfake technology can have positive applications (TechInformed, 2024) in education, marketing and healthcare, we more often see it's disadvantages and there has risen a need for deepfake detection algorithms. This calls for robust machine learning techniques capable of identifying even the smallest manipulations in images to categorise them as fake or real. We propose the use of convolutional neural networks, CNNs, to address this need as this was seen to be the most successful method in multiple models(found through our literature review). One of its benefits is it's ability to utilise transfer learning (in the case of pretrained CNN's) which improve the efficiency of the model (Rafique et al., 2021).

We evaluate three CNN-based approaches: (1) a custom CNN with YOLO for joint face localisation and classification, (2) Inception-ResNet V2 for transfer learning, and (3) ShuffleNet V2 for transfer learning and lightweight inference. Using the *140k Real and Fake Faces* dataset (XHLULU, 2022) sourced from Kaggle, we demonstrate that pretrained models significantly outperform custom architectures. Inception-ResNet V2 achieved $\mathbf{98.98}\%$ test accuracy, surpassing ShuffleNet V2 ($\mathbf{98.19}\%$) and our YOLO-CNN baseline ($\mathbf{83.23}\%$). Notably, YOLO-based preprocessing provided marginal utility despite computational overhead, suggesting that modern CNNS inherently localise discriminative regions (e.g., eyes, mouth) without explicit guidance.

Figure 1. Data Preview of 140k Real and Fake Faces

Our key contributions include:

- Quantifying the speed-accuracy trade-off: ShuffleNet V2 achieves $\mathbf{2}\times$ faster inference than Inception-ResNet V2 with only $\mathbf{0.078}\%$ accuracy loss, enabling real-time deployment on edge devices.
- Demonstrating redundancy of preprocessing (YOLO face cropping) in architectures with spatial attention mechanisms.
- Providing Grad-CAM visualisations to interpret model decisions, addressing the "black box" critique of deep learning in forensic applications.

This work bridges the gap between theoretical accuracy and practical deployability, offering actionable insights for implementing explainable deepfake detectors in resource-constrained environments.

2. Related Work

Recent advances in deepfake detection have been driven almost entirely by deep neural networks, which automatically learn subtle spatial, temporal, and frequency-domain features introduced by modern synthesis algorithms (Mary & Edison, 2023). Convolutional Networks, often augmented with attention or combined with temporal models, remain foundational (Rössler et al., 2019) .Capsule network architectures encode hierarchical part–whole relationships to improve robustness against adversarial attacks (Nguyen et al., 2019). Transformer-based and hybrid CNN–ViT models offer enhanced global reasoning and cross-dataset generalisation (Mary & Edison, 2023), while self-supervised and anomaly-detection frameworks further strengthen performance under limited supervision (Larue et al., 2023). Below, we review these approaches, discussing the methodologies, outcomes and areas of improvement while presenting our work, which involves a comparative analysis of fine-tuned and custom neural network architectures.

2.1. Convolutional Network Methods

Early deepfake detectors demonstrated that conventional CNNs can reliably distinguish fake from real frames by exploiting mesoscopic and frequency features. Rössler et al. introduced FaceForensics++, a benchmark demonstrating that XceptionNet trained on compressed (c23) videos achieves over 98 percent frame-level accuracy (Rössler et al., 2019). Afchar et al. proposed the MesoInception-4 network, a lightweight CNN focusing on mesoscopic image properties, attaining 92 accuracy on individual DeepFake images (Afchar et al., 2018). Recent work integrates pretrained CNNs (e.g., EfficientNet-B4) with ensemble classifiers like XGBoost, achieving 90.7 percent accuracy on merged Celeb-DF and FaceForensics++ datasets (Ismail et al., 2021b). Pan et al. further enhanced CNN-based detection via a cost-sensitive training, attaining 98 percent accuracy with XceptionNet on Celeb-DF v2 (Pan et al., 2020).

2.2. Spatio-Temporal Deep Models

To capture temporal inconsistencies, such as irregular blinking or lip movements, several studies augment CNN's with recurrent or 3d-convolutional modules. Li et al. first exploited abnormal blink patterns to expose forged videos via a CNN+LSTM pipeline (Li et al., 2018). Xu and Yayilgan compared CNN+RNN pipelines against standalone CNNS, revealing that sequence-level modelling improves robustness under cross-dataset mismatches on FaceForensics++ (Redmon et al., 2016). Hybrid 3D CNN architectures (e.g., Inflated 3D ConvNets) have also been applied, showing strong generalisation across multiple manipulation methods by simultaneously modelling spatial textures and short-term motion (Larue et al., 2023). Güera and Delp applied Long Short-Term Memory (LSTM) networks atop per-frame CNN features, demonstrating an RNN-based detector with 90% accuracy on AVSS-2018 deepfake videos (Güera & Delp, 2018).

2.3. Capsule network and Attention-Enhanced Architectures

Capsule-Forensics employs capsule layers to preserve hierarchical relationships between facial parts, mitigating the loss of spatial information common in pooling operations. Nguyen et al. demonstrated that a capsule-based model outperforms equivalently sized CNNs on both image and video deepfake benchmarks, particularly under compression features(Nguyen et al., 2019). Incorporating attention modules enables fine-grained localisation of manipulated regions. Ciamarra et al. presented a multi-attention network that applies CondenseNet feature enhancement alongside multi-scale artefact detectors, yielding high true-positive rates even when identity cues vary substantially (Rössler et al., 2019).

2.4. Transformer-Based and Self-Supervised Approaches

Vision transformers (ViTs) and hybrid CNN–ViT architectures have recently set new state-of-the-art results. Mary and Edison (Mary & Edison, 2023) outlined a hybrid model in which parallel EfficientNet-B4 and XceptionNet extract local patches that are then processed by a transformer encoder, achieving 95 percent accuracy on Celeb-DF while maintaining real-time inference capability. In the self-supervised domain, Larue et al. introduced SeeABLE, which learns one-class anomaly scores via a bounded contrastive loss; the model localises spatial and frequency-domain perturbations without requiring negative samples, achieving state-of-the-

art results on multiple benchmarks (Larue et al., 2023).

2.5. Positioning Our Work

While prior studies focus on single architectures (e.g., XceptionNet (Rössler et al., 2019), MesoInception-4 (Afchar et al., 2018)), we conduct a comparative analysis of 3 approaches: (1)Inception-ResNet V2: Leverages transfer learning with frozen ImageNet layers. (2)YOLO Hybrid: Integrates YOLOv8n (Redmon et al., 2016) for face localisation, enhancing data-preprocessing and interpretability. (3)ShuffleNet V2: Balances accuracy and efficiency via fine-tuning using a custom CNN as a baseline model.

Our work includes the YOLO-based preprocessing pipeline, which localises faces before classification, a step intended to reduce noise and improve focus on facial features. However, our experiments revealed that this preprocessing step did not significantly enhance performance compared to end-to-end models. Despite theoretical advantages, the YOLO hybrid achieved only 83.2 percent test accuracy, lagging behind Inception-ResNet V2 (99 percent) and ShuffleNet V2 (98 percent). We attribute this to two factors: (1) localisation inaccuracies in low-resolution or occluded faces, which introduced false positives (FPR: 7 percent), and (2) computational overhead from dual-model inference, which slowed training without commensurate accuracy gains. This finding contrasts with prior hybrid frameworks like CNN+RNNs (Güera & Delp, 2018), where temporal integration improved robustness.

3. Proposed Models

Besides a custom CNN as our baseline model, we explored three deep learning models, based on the literature reviewed.

1. Yolo (You look only Once)
2. Inception-ResNet V2
3. Shuffle Net

These models will be evaluated using the 140k faces dataset (XHLULU, 2022), which was chosen due to it's availability, providing insights into trade-offs between accuracy, speed, and complexity, a gap in existing literature dominated by single-model evaluations (Mary & Edison, 2023),(Ismail et al., 2021b).

It consists of 70k REAL faces from the Flickr dataset collected by Nvidia, as well as 70k fake faces sampled from the 1 Million FAKE faces, generated by StyleGAN. The downloaded dataset consisted of images resized to 256×256 pixels and split into training, validation, and test sets, accompanied by CSV files for streamlined data handling.

3.1. Baseline CNN

For comparison we first bit a Baseline CNN which is particularly simple in architecture. Specificaly, the network initiates with a pair of convolutional blocks: a Conv2d layer with 16 filters and a stride of 2, followed by a ReLU activation, and a second Conv2d layer with 32 filters, also using a stride of 2 to progressively downsample feature maps.

Ultimately we use two different baseline models, one with dropout for the second to last layer and one without. Here on forth, 'Baseline model' or 'Model 4' will refer to that with dropout seeing as it had a higher validation accuracy. see table 2

The hyperparameters for this baseline are as follows. Training employed the Adam optimizer with a reduced learning rate of 0.0001, paired with binary cross-entropy loss with logits (BCEWithLogitsLoss). The model was trained over 50 epochs on RGB facial images resized to 256×256. The dataset was preprocessed externally, with no internal data augmentation.

Ultimately Model 4 acquired a validation accuracy of nearly 90 percent. later on we consider the Inception-ResNet50 and Shuffle Net to be superior classification architectures given their greater success in this metric

3.2. YOLO with custom CNN

YOLO (You Only Look Once) Methods have recently been applied to deep fake face detection tasks. Most notably is a paper produced by (Ismail et al., 2021a) who applied YOLO towards deep fake video detection. YOLO architecture was initially developed by Redmon et al. in a 2016 paper and have since seen large subsequent developments in use.

3.2.1. Architecture

YOLO is primarily implemented with a Darknet backbone which is trained to produce bounding boxes identifying specific objects. Per its name, You Only Look Once it is designed to be extremely fast and efficient for image detections making its use appealing for complicated classification tasks and network architectures. It is able to be quite computationally efficient as it only looks at the image for one pass, a result of its "single pass architecture" In addition to its relative efficiency for the task this method enables one to isolate specific components of an image, specifically faces. This then isolated component of the image can be passed to subsequent layers in a model. Theoretically this approach can prove to be more effective for deep-fake detection as it simplifies inputs to the large model making both training and classification more efficient. Additionally because parts of the network are fed simpler images It may enable more accurate identification.

Our application of YOLO follows a similar architecture to that used by (Ismail et al., 2021a). However we fine tune this architecture specifically for processing a single image. Therefore we forgo the use of a bidirectional LSTM.

We first tune a preexisting YOLO architecture, specifically Yolov8 on a dataset containing 17,000 noisy (as in containing multiple objects including faces) images. This data set can be found on kaggle (Elmenshawii, 2020). Once trained this model is then implemented as the first layer in a larger classification network. Since the YOLO model outputs bounding boxes this first layer involves identifying the face and then cropping and resizing the contents of the output bounding box. This cropped image is then passed to the rest of the model.

To better isolate the impact of employing YOLO this large model is kept relatively simple containing just two 2D convolutional layers of RGB input, 16 output channels with kernel size 3 and 2 strides. The subsequent 2D convolutional layer contains 32 output channels with kernel size 3 and 2 strides. Both these layers implement ReLU activation functions as these have well established success in the deep learning literature. The convolutions are then flattened with 128 outputs and a ReLU activation function followed by a dropout of 50 percent and then the final binary classification output layer. Table 1 summarizes this

Model	Epochs	Dropout	YOLO Layer
Model 1	10	No	Yes
Model 2	10	No	No
Model 3	50	Yes	Yes
Model 4	50	Yes	No

Table 1. Model configurations for custom CNNs

3.2.2. Training Methods / Experimental Setup

As noted, four different models are trained for either 10 and 50 epochs. Firstly exists a model 1 with the aforementioned architecture simply without dropout trained for 10 epochs. Model 2 is then trained without both the initial YOLO layer and without dropout for 10 epochs to try and isolate the impact of YOLO. Model 3 is then trained for 50 epochs with dropout and the fourth and final without YOLO but with dropout for 50 epochs.

To train all of these architectures an Adam optimizer is used with a learning rate of 0.0001 and using a binary cross entropy loss function. Dropout is implemented in the third and fourth model and validation and training accuracy data supports it's implementation

3.3. Inception-ResNet V2

InceptionResNetV2 is a hybrid architecture combining the multi-scale feature extraction of Inception modules with residual connections, introduced by Szegedy et al. in "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning," 2016. While originally designed for large-scale image classification, its ability to capture fine-grained spatial and hierarchical features makes it highly effective for deepfake detection. (Szegedy et al., 2017) Unlike traditional CNNs, InceptionResNetV2 reduces computational overhead through factorized convolutions (e.g., asymmetric 1×3 and 3×1 filters) and leverages residual scaling to stabilize training. For this project, we adapt the pretrained InceptionResNetV2 backbone, freezing early layers to retain generic feature representations and fine-tuning deeper layers to specialize in detecting features indicative of synthetic media.

3.3.1. Architecture

The model uses a pretrained InceptionResNetV2 backbone, initialized with weights from ImageNet to exploit its robust feature extraction capabilities. To preserve low- and mid-level feature detectors (e.g., edge filters, texture analyzers), the first 450 layers are frozen, ensuring stability in early training phases. Input images are standardized to 224×224×3 resolution, aligning with the model's original training configuration. The base model processes inputs through its hierarchical Inception-ResNet blocks, culminating in 8×8×1536 feature maps that encode high-level spatial and channel-wise patterns. These maps are condensed into a 1536-dimensional vector via a GlobalAveragePooling2D layer, which retains discriminative features while discarding spatial redundancies.

A custom classification head is appended to adapt the model to deepfake detection. First, a Dense layer (512 units) with ReLU activation introduces nonlinearity, augmented by L2 regularization ($\lambda = 0.01$) to penalize overly large weights and curb overfitting. BatchNormalization follows to stabilize training dynamics by normalizing activations and mitigating internal covariate shift. A Dropout layer (rate = 0.5) further regularizes the network by randomly deactivating (50% of neurons during training, enforcing redundancy in learned representations. The final Dense layer employs sigmoid activation for binary classification, producing a probability score for real vs. fake detection.

With 25.3 million total parameters, the architecture emphasizes computational efficiency: only 3% of parameters (layers 451 onward) are trainable, focusing fine-tuning on high-level hierarchical patterns critical for identifying deepfake features, such as inconsistent facial landmarks, unnatural texture blending, or anomalous frequency distributions. By freezing the majority of the base model, the design

balances transfer learning (preserving generic feature extractors) with task-specific adaptation, optimizing both accuracy and resource utilization. This approach ensures robust performance on high-resolution inputs while maintaining compatibility with modern GPU hardware.

3.3.2. TRAINING

Some data augmentation strategies included horizontal flips, random rotations (±20°), and zoom (10%) to simulate viewpoint and scale variations. Pixel values were normalised to the [0, 1] range to align with the base model's pretraining distribution. Early experiments revealed diminishing returns from augmentation, likely due to the dataset's inherent diversity, prompting its retention primarily for regularisation rather than performance gains.

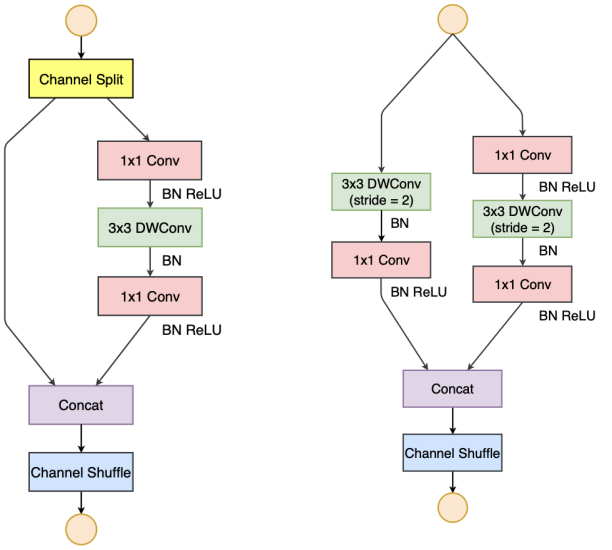
The model was then fine-tuned using the Adam optimizer with a reduced learning rate of 0.0001, selected to balance gradient stability in the frozen base layers with adaptation of the unfrozen task-specific layers. This configuration minimized perturbations to pretrained feature extractors while enabling controlled updates to high-level representations. Training leveraged batches of 32 samples, constrained by GPU memory limitations, with the 'tf.data' API prefetching data to optimize pipeline efficiency through parallelized loading and computation. Training was governed by early stopping (patience = 3 epochs), terminating the process after 8 epochs upon validation loss plateauing to prevent overfitting. To address computational and class imbalance constraints, a random subset of 20,000 samples was utilized, preserving the original data distribution while reducing training time by 60%.

3.4. ShuffleNet

ShuffleNet is a computationally efficient CNN architecture designed specially for mobile devices with very limited computing power first introduced by Zhang et al. in "ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices", 2018. Rafique et al. proposed the use of Shufflenet for extraction of features and then classification using either KNN or SVM for deep fake detection. However, this approach proved to be computationally expensive. So instead we propose to use ShuffleNet V2 for the complete model as it is more computationally efficient given constraints such as memory access costs (Ma et al., 2018).

3.4.1. ARCHITECTURE

ShuffleNet V2 has been pretrained using Imagenet for classification as was done in ShuffleNet. The basic units for ShuffleNet V2 are illustrated in Fig 2. In the basic unit seen in 2a, the input of feature channels are split into 2 branches. Branch 1 remains as identity (no changes made to it. Branch 2 consists of three convolutions - pointwise convolution, depthwise convolution and again pointwise convolution. Then the 2 branches are concatenated and a channel shuffle operation occurs which "enables information communication between the two branches." (Ma et al., 2018) In the spatial downsampling unit seen in 2b, the 2 branches process the whole input in parallel. Branch 1 consists of a depthwise convolution followed by pointwise convolution. Branch 2 consists of three convolutions - pointwise convolution, depthwise convolution and again pointwise convolution. The depthwise convolution layer has stride = 2 which is where the downsampling occurs. The 2 branches are then concatenated and a channel shuffle operation occurs. The number of output feature channels of this block is double the number of input channels due to the parameters given in the pointwise convolution. The ShuffleNet V2 model being proposed for use in this project consists of 5 stages and a fully connected layer which acts as the classifier. These stages are built by which are built by stacking the units in Fig 2. The first 2 stages are frozen and stage 3, 4 and 5 are trained using the training dataset sourced from Kaggle. Stage 1 consists of an initial convolution layer (followed by batch normalisation and ReL activation) and a max pooling layer. Stage 2 consists of 4 blocks . Block 0 is the spatial downsampling unit and Blocks 1 - 3 are the basic unit stacked. In the PyTorch implementation used in our model - ' pytorch/vision:v0.10.0', 'shufflenet_v2_x1_0' , Stage 2 by default outputs 116 channels. So each branch of the downsampling unit takes an input of 24 and outputs 58. (Other versions of shufflenet_v2 in PyTorch output 48).



(a) Basic Unit

(b) Unit for spatial down sampling

Figure 2. ShuffleNet V2 Architecture. Reproduced from (Ma et al., 2018) [DW stands for Depthwise Convolution]

Stage 3 consists of 8 blocks : Block 0 is a spatial downsampling unit and Blocks 1-7 are the basic unit stacked. The output of this stage is 232 channels Stage 4 consists of 4 blocks: Block 0 is the spatial downsampling unit and Blocks 1 - 3 are the basic unit stacked. The output of this stage is 464 channels. Stage 5 is a convolution layer with input 464 and output 1024 (followed by batch normalisation and ReLU activation). Finally the last layer of the model is a fully connected layer which acts as the classification layer : takes in 1024 features and returns 2 labels.

3.4.2. Training methods / Experimental Setup

As mentioned in Section 3.4.1, we are using the pretrained ShuffleNet V2 Model available via PyTorch. (pytorch/vision:v0.10.0', 'shufflenet_v2_x1_0'). Stages 3,4 and 5 have been unfrozen to train the model better as without this, the model was overfitting. Thus the parameters/optimisers used were only for these stages. For the optimiser, Adam was chosen as it was found to work better than SGD and RMSprop based on empirical performance. This could be due to its adaptive learning rate. A learning rate of 0.001 was also chosen based on empirical performance. Due to memory limitations and computational time constraints, this experiment was restricted in terms of size of dataset used and training duration. A random subset of 20,000 samples from the training dataset was selected, as it yielded high validation accuracy while keeping resource usage manageable. Increasing the dataset size beyond this point resulted in minimal improvements in validation performance, despite a substantial increase in training time. Similarly, applying data augmentation extended the training time without producing a significant gain in accuracy. The number of training epochs was limited to 10 to reduce computational overhead; empirical results indicated that validation accuracy stabilized around this point.

4. Numerical Results

All the models discussed have been evaluated using the 140k faces dataset (XHLULU, 2022), with the primary metric used for evaluation being the test accuracy. Given that the dataset was balanced, it was realised that accuracy was a good choice for evaluation. Further, some multiple models gave similar accuracies, so other evaluation metrics such as precision, recall, f1 score, area under receiver operating curve, false positive rate, false positive rate and false negative rate were considered. The definitions of the used metrics are as follows

- **Precision:** The proportion of correctly predicted positive samples out of all samples predicted as positive.

$$\text{Precision} = \frac{TP}{TP + FP}$$

- **Recall:** The proportion of correctly predicted positive samples out of all actual positive samples.

$$\text{Recall} = \frac{TP}{TP + FN}$$

- **F1 Score:** The harmonic mean of precision and recall.

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

- **AUC-ROC:** Area under the Receiver Operating Characteristic curve. Measures the model's ability to distinguish between classes across all thresholds. The closer to 1, the better the model

- **False Positive Rate (FPR):** The proportion of negative samples incorrectly classified as positive.

$$\text{FPR} = \frac{FP}{FP + TN}$$

- **False Negative Rate (FNR):** The proportion of positive samples incorrectly classified as negative.

$$\text{FNR} = \frac{FN}{TP + FN}$$

The baseline model chosen, custom CNN neural network with dropout gave an accuracy of $\approx 90\%$

4.1. Custom CNN with vs without YOLO

As mentioned, the primary task at hand is identifying the impact of implementing a YOLO layer in a simple neural network architecture. Of the two models trained without dropout, the one which delivered the greatest validation accuracy was the one with a YOLO layer, respectively 89 percent compared to 81 percent for the last epoch. However, a certain degree of overfitting can be observed, justifying the addition of a dropout layer. Later, when training for 50 epochs with dropout, the model which saw the greatest success was actually without the YOLO layer, achieving a validation accuracy of 90 percent and a test accuracy of 89.53 percent. This was relative to a validation accuracy of 84 percent and a test accuracy of 83.23 percent for the model with YOLO. Given such it could be argued that the implementation of the YOLO layer saw a decrease in test accuracy of about 6 percent. Additionally, to aid in the argument that this method finds validity in efficiency the speed of single image classification for all models can be seen at a near negligible value. With the total time for roughly 20,000 images taking roughly 2 ms for a batch of 1024 images. These results are presented in Table 2 and Table 3

Model	Validation Accuracy	Description
Model 1	∼89%	Custom CNN without dropout, with YOLO
Model 2	∼81%	Custom CNN without dropout and without YOLO
Model 3	∼84%	Custom CNN with dropout, with YOLO
Model 4	**∼90%**	**Custom CNN with dropout, without YOLO**

Table 2. Validation accuracy and descriptions for each model.

Metric	Model 1	Model 3
Test Accuracy	89.53%	83.23%
Total Inference Time	39.23 ms	31.53 ms
Avg. Time per Batch (1024)	2.18 ms	1.58 ms
Avg. Time per Image	0.01 ms	0.00 ms

Table 3. Test performance comparison between Model 1 and Model 3

When working with a custom CNN,the most successful model was that without YOLO, with dropout and trained for 50 epochs. However we want to make the case that the implementation of YOLO may have yielded better results than that of the non YOLO had it been trained for further epochs given the trend of the validation accuracy for the third model was yet to plateau after 50 epochs. Further supporting this assertion is that in the first two models that which did have the YOLO layer far outperformed that without the YOLO. Without dropout the models seemed to have trained faster

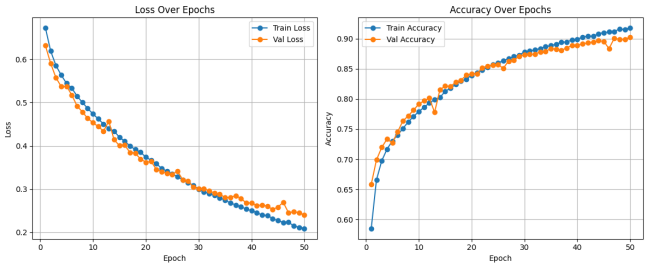


Figure 3. Baseline with dropout (Model 4)

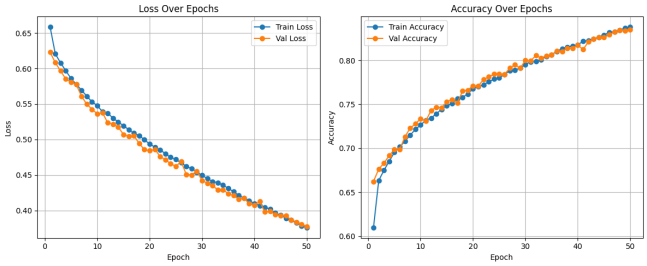


Figure 4. Yolo with dropout (Model 2)

Upon considering the respective accuracies, Inception-resNet and Shufflenet delivered better results, hence, for further comparison, additional evaluation metrics were assessed for the two models

Metric	Inception-ResNet v2	ShuffleNet v2
Test Accuracy	0.9897	0.9819
Precision	0.9925	0.9850
Recall	0.9868	0.9809
F1 Score	0.9896	0.9830
AUC-ROC	0.9994	0.9986
FPR	0.0075	0.0149
FNR	0.0132	0.0191

Table 4. Evaluation metrics for ShuffleNet v2 and Inception-ResNet v2

4.2. Inception-ResNet v2

The InceptionResNetV2 model, trained on a random subset of 20,000 images, achieved exceptional performance on the test set (unseen during training). With 98.97% test accuracy and a near-perfect AUC-ROC of 0.9994, the model demonstrated near-flawless discrimination between ”fake” and ”real” classes.

Key metrics included precision (99.25%) and recall (98.68%) , with an F1 score of 98.96reflecting balanced performance across both classes. Misclassification rates were minimal: 0.75% of real images were falsely flagged as fake (FPR = 0.0075), and 1.32% of deepfakes were misclassified as real (FNR = 0.0132). The test loss of 0.0332 confirmed stable convergence, with training/validation curves indicating no overfitting. (5)

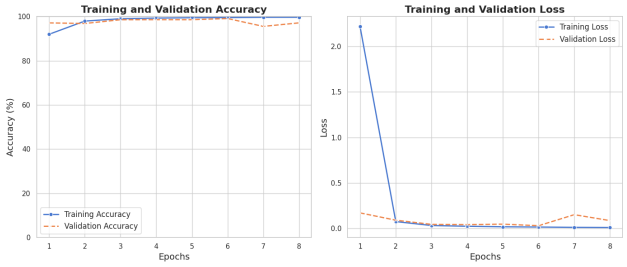


Figure 5. Inception-ResNet v2 - Accuracy and Loss (Training vs Validation)

4.3. ShuffleNet v2

The ShuffleNet v2 Architecture was found to perform well when trained with a random sample of 20,000 images from the training dataset and validated using the entire validation dataset over 10 epochs. When using a larger sample, the model sometimes overfit to the training data. A total of

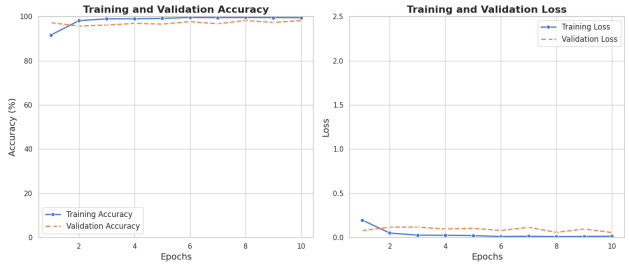


Figure 6. ShuffleNet v2 - Accuracy and Loss (Training vs Validation)

2,278,604 parameters can be found in ShuffleNet v2 but only approximately 1,500,000 of these are trainable as only Stages 3,4 and Conv5 were unfrozen. As accuracy was found to be high (i.e. 98.19%), other evaluation metrics were calculated which can be seen in Table 4. It is clear to see that this model performs well as the model correctly identifies 98.09% of all deepfakes (seen through the recall metric) i.e only 1.91% of deepfakes were classified as real. Only 1.49% of real images was incorrectly flagged as fake. We also see that the area under the receiver operating characteristic curve (AU-ROC) is 0.999 which is almost 1 which signifies the model is nearly perfect indicating it can differentiate between the 2 classes well.

5. Interpretation

As seen in Section 4, the models that perform the best are the pre-trained CNNs, ShuffleNet v2 and Inception ResNet v2 which have had the last few layers unfrozen to train with the given dataset. These models differ in computational cost.

ShuffleNet v2 offers a high accuracy with 2.2 million parameters as compared to the 25.3 million parameters in Inception-ResNet v2 model. It's use of channel splitting enables fast inference and specifically efficiency on low-resource devices. (Ma et al., 2018) Despite it's simplicity, ShuffleNet achieved a test accuracy of 98.19% with $AU - ROC \approx 0.999$ with only a sample of the training set and no data augmentation, highlighting its capability to generalise well with minimal computational cost. On comparing the plots for training vs validation accuracy and loss for two models, it can be seen that Inception-ResNet v2 has a more stable convergence. This is beneficial for large-scale image classification tasks. Although the test precision and $AU - ROC$ of the Inception-ResNet v2 model were higher, the model took longer to run using the T4-GPU available on Google Colab. (ShuffleNet v2- 45 minutes; Inception-ResNet v2 - 120 minutes) Therefore there is a trade-off between performance and efficiency that must be considered while choosing a model.

Explainable AI Tools

Due to resource constraints, we were only able to perform a sanity check using the explainable AI tool - Gradient-weighted Class Activation Mapping (Selvaraju et al., 2017) on the ShuffleNet v2 model. We visualised activation heat maps on a sample of 3 images each from both classes - real and deepfake to ensure that the models focus on the facial features and not the background as this could lead to a bad performance of the model on other datasets. It is clear from 7 that the model primarily focuses on facial features such as eyes in the case of real images to classify the image. Thus the model should perform well when faced with another unseen dataset

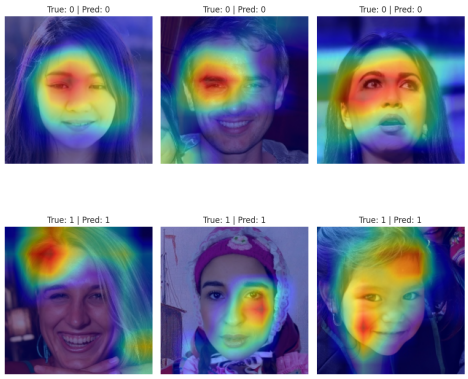


Figure 7. Using GradCam to show most useful features used by model ShuffleNet v2

6. Conclusion

In this study, we explored multiple deep-learning architectures for deepfakes detection. While the YOLO based approaches looked promising in terms of computational time, the test accuracy did not compare with that of the other models. It has been demonstrated here that this methodology may have the capacity to enable better detection. Furthermore, it demonstrates that the greater speed of YOLO methods can be leveraged to decrease identification time. Despite the benefits of the computational CNN with YOLO model, we consider the ShuffleNet v2 and Inception-ResNet V2 models to be better models due to their significantly higher test accuracy. On experimentation, we also found that these models exhibit high precision and recall, making them suitable for applications where deepfake detection is critical.

Further research could attempt to improve the computational time of these models and possibly leverage more explainable AI tools to improve model interpretability.

Statement of Individual Contributions

This project was done with equal contribution from each of it's authors.

- 43729 worked on ShuffleNet v2 model and presented it in the paper. They also co-wrote other sections of the paper.
- 48032 worked on Inception-ResNet v2 model and presented it in the paper. They also co-wrote other sections of the paper.
- 47914 worked on custom CNNs with and without YOLO and presented it in the paper. They also integrated all models into one comprehensive script.

References

Afchar, D., Nozick, V., Yamagishi, J., and Echizen, I. Mesoinception-4: A compact facial video forgery detection network. In *Proceedings of the IEEE Winter Applications of Computer Vision Workshops (WACVW)*, 2018.

BBC Newsround. Deepfake technology: What is it, how does it work, and what can it be used for? https://www.bbc.co.uk/newsround/69009887, May 2024. Accessed: 2025-04-30.

Elmenshawii, F. Face detection dataset. https://www.kaggle.com/datasets/fareselmenshawii/face-detection-dataset, 2020. Accessed: 2025-05-01.

Güera, D. and Delp, E. J. Deepfake video detection using recurrent neural networks. In *Proceedings of the IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pp. 1–6, 2018. doi: 10.1109/AVSS.2018.8639163.

Ismail, A., Elpeltagy, M., Zaki, M., and ElDahshan, K. A. Deepfake video detection: Yolo-face convolution recurrent approach. *PeerJ Computer Science*, 7:e730, 2021a. doi: 10.7717/peerj-cs.730. URL https://doi.org/10.7717/peerj-cs.730.

Ismail, A., Elpeltagy, M., Zaki, S., and Eldahshan, K. A new deep learning-based methodology for video deepfake detection using xgboost. *Sensors*, 21(16):5413, 2021b. doi: 10.3390/s21165413.

Larue, L., Sztyler, T., Steuwe, H., Knopp, J., Rosenhahn, B., and Riess, C. Seeable: Soft discrepancies and bounded contrastive learning for exposing deepfakes. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2023.

Li, Y., Chang, M.-C., and Lyu, S. In ictu oculi: Exposing ai generated fake face videos by detecting eye blinking. arXiv preprint arXiv:1806.02877, 2018.

Ma, N., Zhang, X., Zheng, H.-T., and Sun, J. Shufflenet v2: Practical guidelines for efficient cnn architecture design, 2018. URL https://arxiv.org/abs/1807.11164.

Mary, A. and Edison, A. Deep fake detection using deep learning techniques: A literature review. In *Proceedings of the International Conference on Control, Communication and Computing (ICCC)*, pp. 1–6, 2023. doi: 10.1109/ICCC57789.2023.10164881.

Nguyen, H. H., Yamagishi, J., and Echizen, I. Capsule-forensics: Using capsule networks to detect forged images and videos. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019.

Pan, D., Sun, L., Wang, R., Zhang, X., and Sinnott, R. O. Deepfake detection through deep learning. In *Proceedings of the IEEE/ACM International Conference on Big Data Computing, Applications and Technologies (BDCAT)*, pp. 134–143, 2020. doi: 10.1109/BDCAT50828.2020.00001.

Rafique, R., Nawaz, M., Kibriya, H., and Masood, M. Deepfake detection using error level analysis and deep learning. In *2021 4th International Conference on Computing Information Sciences (ICCIS)*, pp. 1–4, 2021. doi: 10.1109/ICCIS54243.2021.9676375.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, 2016.

Rössler, A., Cozzolino, D., Verdoliva, L., Riess, C., Thies, J., and Nießner, M. Faceforensics++: Learning to detect manipulated facial images. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019.

Selvaraju, R. R., Das, A., Zitnick, C. L., and Parikh, D. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *IEEE International Conference on Computer Vision (ICCV)*, pp. 618–626, 2017. doi: 10.1109/ICCV.2017.74.

Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. A. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, volume 31. AAAI Press, 2017. doi: 10.1609/aaai.v31i1.11231. URL https://ojs.aaai.org/index.php/AAAI/article/view/11231.

TechInformed. Deepfakes for good? how synthetic media is transforming business. https://techinformed.com/deepfakes-for-good-how-synthetic-media-is-transforming-business/, 2024. Accessed: 2025-05-01.

XHLULU. 140k real and fake faces. https://www.kaggle.com/datasets/xhlulu/140k-real-and-fake-faces, 2022. Accessed: 2025-04-01.

Zhang, X., Zhou, X., Lin, M., and Sun, J. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6848–6856, 2018.