DFP-Net: An explainable and trustworthy framework for detecting deepfakes using interpretable prototypes

Fatima Khalid[1], Ali Javed[2*], Khalid Mahmood Malik[3], Aun Irtaza[1]
[1]Department of Computer Science, University of Engineering and Technology-Taxila, Pakistan
[2]Department of Software Engineering, University of Engineering and Technology-Taxila, Pakistan
[3]Department of Computer Science and Engineering, Oakland University, Rochester, MI, USA
*ali.javed@uettaxila.edu.pk

2023 IEEE International Joint Conference on Biometrics (IJCB) | 979-8-3503-3726-6/23/$31.00 ©2023 IEEE | DOI: 10.1109/IJCB57857.2023.10448842

Abstract

The rise of deepfake videos poses a serious threat to the authenticity of visual media, as they have a potential to manipulate public opinion, mislead individuals or groups, harm reputation, etc. Traditional methods for detecting deepfakes rely on deep learning models, which lack transparency and interpretability. To gain the confidence of forensic experts in AI-based deepfakes detector, we present a novel DFP-Net for detecting deepfakes using interpretable and explainable prototypes. Our method makes use of the power of prototype-based learning to generate a set of representative images that capture the essential features of genuine and deepfake images. These prototypes are then used to explain our model's decision-making process and to provide insights into the features most relevant for deepfake detection. We then use these prototypes to train a classification model that can detect deepfakes accurately and with high interpretability. To further improve the interpretability of our method, we also utilize the Grad-CAM technique to generate heatmaps that highlight the regions of the image that contribute the most towards the decision of the model. These heatmaps can be used to explain the reasoning behind the model's decision and provide insights into the visual cues that distinguish deepfakes from real images. Experimental results on a large-scale FaceForensics++, Celeb-DF and DFDC-P datasets demonstrate that our method achieves state-of-the-art performance in deepfakes detection. Moreover, the interpretability and explainability of our method make it more trustworthy to forensic experts by allowing them to understand how the model works and makes predictions.

Keywords: Deepfakes detection, DFP-Net, Interpretable prototypes, Explainable AI, FaceForensics++.

1. Introduction

In recent years, deepfake technology has gained widespread attention due to its ability to generate highly convincing fake media content, such as videos, images, and audio. While this technology has various applications, including entertainment and creative expression, it poses significant risks to individuals, organizations, and society. Deepfakes can be used to spread false information, manipulate public opinion, and perpetrate fraud, among other malicious activities. Therefore, the need for reliable deepfake detection methods has become more critical than ever. Traditional deepfake detection methods have relied on handcrafted feature-based techniques as well as end-to-end deep learning-based techniques that are trained on large datasets of real and fake content. However, these methods have limitations, such as being vulnerable to adversarial attacks, lacking interpretability and explainability, and having limited generalization ability.

Deep neural networks (DNNs) have proved successful in many computer vision tasks, but their "black box" nature makes it challenging to understand their decision-making process. Recently, there has been a growing interest in developing interpretable and explainable machine learning models for deepfake detection to counter this limitation of DNN. Machine learning models rely heavily on the concepts of interpretability and explainability. The ability to understand how a model works and how it makes predictions is referred to as interpretability. It means that the cause and effect can be determined, and the model can take the inputs and produce the same outputs on a regular basis [1]. Explainability, on the other hand, refers to a model's ability to provide a human understandable explanation of how it works and why it makes certain predictions. As a broader concept, explainability includes interpretability as well [2]. Like, in [3], Supervised Contrastive Learning (SCL) for deepfake detection was presented, which aims to improve the generalization and explainability of deepfake detection models. This method trains a deepfake detection model using a supervised contrastive loss function to classify real and fake samples and generates class activation maps, which highlight the regions of the input that are most relevant to the classification decision. However, this method requires access to both real and fake videos of the same person during training, which may not be feasible. In [4], attention-based architecture was presented for deepfakes detection. Also, an ensemble of different models was employed to improve the detection performance and

generate focus attention maps using Grad-CAM explanations. However, this approach requires significant computational power during training. In [5], pairwise learning and complementary information from various color space representations were utilized to detect deepfakes. A multi-channel XceptionNet was used to classify real and fake images by analyzing pairs of facial components. The method also incorporated t-SNE and attention maps to explain how the decision-making process works. However, this method is not generalizable to real-world scenarios. In [6], a graph neural network is presented for identifying deepfakes by dividing images into nodes and creating a graph by connecting adjacent nodes based on low-level features such as color and texture. The resulting graph structure is used to gain insight into the model's decision-making process. However, computing the adjacency matrix can be time-consuming for large graphs, which limits scalability when dealing with extensive datasets. These techniques employ diverse methods of explanation to achieve different levels of interpretability. For instance, grad-cams are used to explain the inner working of a model, activation maximization helps in visualizing neurons, while deconvolution or up-convolution can explain and visualize the layers of the architecture. While these models have made significant efforts in achieving accurate predictions, there remains considerable scope for enhancing the interpretability and explainability of these models.

One of the prominent approaches to explain DNNs is posthoc analysis via gradient [3-6]. This approach provides insights into the model's behavior but does not modify the underlying architecture and building frameworks that are interpretable by design with a built-in ability of self-explanation. Instead, another line of research allows for more intuitive and understandable explanations for non-experts by representing interpretability as general concepts rather than raw inputs. Chen et al. [7] presented a deep learning model for image recognition that uses prototypes obtained through a clustering algorithm on similar patches from training images. This method is easily extendable to new classes without retraining and requires no human intervention. In [8], a hierarchical prototype-based method was introduced for object classification within a predefined taxonomy. This method selects the most similar prototype at each level to make predictions, enabling the classification of previously unseen classes. In [9], natural language explanations of prototype representation for a class were generated using gradient and optimization techniques to improve interpretability. Nauta et al. [10] presented a ProtoTree for fine-grained image recognition that uses prototype learning and decision trees. However, its effectiveness depends on the prototypes' ability to represent the class well, and it may not always provide accurate local explanations. Trinh et al. [11] employed dynamic prototypes to distinguish real videos from deepfakes by capturing their unique characteristics. A deep neural network (DNN) based encoder was applied to produce these dynamic prototypes, which were then utilized to calculate similarity scores for the test videos. However, the method requires large amount of training data.

Current literature focuses on detecting deepfakes but lacks attention to interpretability and trustworthiness. These methods only label faces as real or fake or give the probabilities, it would be more useful to explain how the model arrived at its decision. An explainable and interpretable deepfake detector is needed that not only detects deepfakes but also provides understandable explanations for humans to understand and trust the system's decision-making process. To counter these issues, we propose an explainable and interpretable prototype-based network DFP-Net for the detection of deepfakes. DFP-Net works by creating a set of prototypical representations by analyzing the features of real and fake samples and grouping the similar features together. After creating these prototypical representations, the proposed method uses them to make predictions about testing samples. It compares the features of the testing sample to the prototypical representations and identifies which prototypical representations are most similar to the testing sample. Finally, to make it easier for experts to understand what the model has learned, the prototypes are projected onto representative image patches from the training dataset. This allows us to see what the prototypes look like in a more tangible way and better understand the differences between real and fake samples that the proposed method has identified. The main contributions of this work are:

- We propose a novel prototype based DFP-Net method for the detection of deepfakes.
- We propose an interpretable and explainable deep learning model for reliable deepfakes detection.
- Rigorous experimentation was performed including the cross corpora evaluation to show the significance of our method.

2. Methodology

The following section provides an in-depth explanation of the proposed DFP-Net for detecting deepfake videos. The architecture of the proposed methodology is shown in Figure 1.

2.1. Pre-processing

We utilized the Multi-task Cascaded Convolutional Neural Networks (MTCNN) [12] face detector during pre-processing stage to identify and extract the facial region of 224 × 224 from the input video. MTCNN is able to recognize facial landmarks like eyes, nose, and mouth, in a

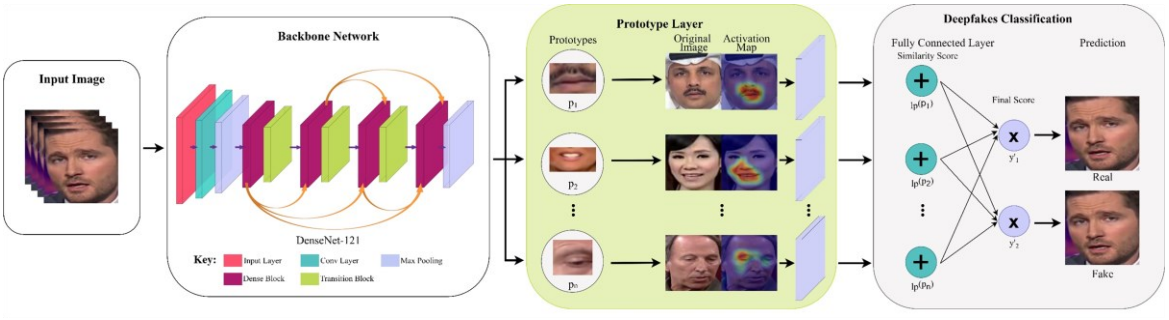


Figure 1: Architecture of DFP-Net.

progressive manner from coarse to fine details. We opted for the MTCNN as it accurately recognizes faces even in the presence of occlusion and varying illumination conditions, unlike other facial detectors [12].

2.2. DFP-Net architecture

The framework of the proposed architecture is shown in Figure 1. Our DFP-Net model consists of three main components: the backbone network $\boldsymbol{f}$, the prototype-layer $\boldsymbol{l_p}$, and the fully connected layer $\boldsymbol{l_h}$. The network is designed to take input in the form of the video frames. Let $\boldsymbol{I} = \{(\boldsymbol{s_i}, \boldsymbol{y_i})\}$ be the video frames of the dataset, where $\boldsymbol{s_i}$ represents the samples and $\boldsymbol{y_i}$ as their corresponding labels. The components are explained in the subsequent sections.

Backbone network. DFP-Net utilizes the convolutional layers of DenseNet-121, augmented by two extra 1 × 1 convolutional layers, as its backbone network. We have employed the Swish activation function for the convolutional layers, until the last, which used the sigmoid function. We have chosen Swish due to its non-monotonic nature, which can enhance the representational power of the network. This property can aid in detecting complex and subtle patterns in deepfakes, where small features play a vital role in accurate classification. The convolutional layers of the proposed method extract significant features $\boldsymbol{f(s)}$ from the input image $\boldsymbol{s}$, which are utilized for prediction. The convolutional output $\boldsymbol{f(s)}$ has a spatial dimension of $\boldsymbol{H = W = 7}$, and the number of output channels $\boldsymbol{D}$ in the additional convolutional layers is chosen from 128, 256, or 512 using cross-validation for our dataset.

Prototype layer. The prototype layer is the main component of our DFP-Net architecture. The prototype layer is able to effectively learn a discriminative representation of each class using only a few labeled samples. DFP-Net utilizes $\boldsymbol{n}$ prototype vectors $(\boldsymbol{p_1}, \boldsymbol{p_2}, \ldots, \boldsymbol{p_n})$ of shape $(\boldsymbol{1}, \boldsymbol{1}, \boldsymbol{D})$ to capture distinct activation patterns in the convolutional feature maps. The prototype layer $\boldsymbol{l_p}$ computes the squared distance $\boldsymbol{l^2}$ between each prototypical vector $\boldsymbol{p_j}$ and every spatial patch $(\boldsymbol{1}, \boldsymbol{1}, \boldsymbol{D})$ in the input feature map $\boldsymbol{x}$, and then inverts the distance to generate $\boldsymbol{n}$ similarity maps for each prototype. These maps indicate the presence of prototypical parts in the image. The activation map of similarity scores for each prototype unit is then globally max-pooled to a single similarity score that represents the strength of the prototypical part in some patch of the input image. The prototype vectors' shape corresponds to the smallest facial patch in $\boldsymbol{x}$, and $\boldsymbol{l_p}$ calculates $\boldsymbol{n}$ similarity scores as:

$$l_{p_n}(x) = \max_{x' \in patches(x)} \frac{1}{1+\|x' - l_{p_n}\|} \quad \textbf{(1)}$$

Where $\boldsymbol{l_{p_n}(x)}$ is the similarity score between the prototype $\boldsymbol{p_n}$ and feature map $\boldsymbol{x}$.

Fully connected layer. In this network, a fully connected layer calculates the weighted sums of similarity scores as $\boldsymbol{l_h = w l_p(x)}$, where the weights are denoted by $\boldsymbol{w \in \mathbb{R}^{k \times n}}$, and $\boldsymbol{k = 2}$ is the number of classes. The resulting values are passed through a SoftMax function to obtain the predicted probabilities as:

$$\hat{y}' = \frac{exp(a_i)}{\sum_{j=1}^{k} exp(a_j)} \quad \textbf{(2)}$$

To represent each class effectively, we allocate $\boldsymbol{n_k}$ prototypes for every class $\boldsymbol{k}$ in the range of $\{0,1\}$. This means that the final model contains $\boldsymbol{n_k}$ prototypes for each class.

2.3. DFP-Net Training

Our goal with DFP-Net is to acquire a significant representation of forgeries that guarantees the proximity of

prototype vectors to input image patches, distinctness between real and fake artifacts, and model interpretability. We train DFP-Net by employing a loss function for all layers except fully connected, prototype projections, and optimization of the fully connected layer. These training stages are repeated multiple times in a cycle.

Loss function. During the initial training phase, our objective is to create a latent space where the image patches are clustered around prototypes that represent similar semantic classes. The clusters associated with prototypes from both classes must be well-separated in terms of $\boldsymbol{l^2}$ distance. To accomplish this goal, we use loss functions to optimize the convolutional layer parameters as well as the prototypes in the prototype layer. Let $\boldsymbol{I} = \{(\boldsymbol{s_i}, \boldsymbol{y_i})\}$ be the training dataset, where s_i represent the image samples and $\boldsymbol{y_i}$ are their labels. The objective function that we aim to minimize involves hyperparameters $\boldsymbol{\mu_c}$, $\boldsymbol{\mu_s}$, and $\boldsymbol{\mu_d}$. It includes four loss functions: cross entropy, clustering, separation, and diversity. It is calculated as:

$$\mathcal{L}(I,\theta) = CrossEntropy(I,\theta) + \mu_c R_{clus}(I,\theta) + \mu_s R_{sep}(I,\theta) + \mu_d R_{div}(I,\theta) \quad \textbf{(3)}$$

Where $\boldsymbol{R_{clus}}$, $\boldsymbol{R_{sep}}$, and $\boldsymbol{R_{div}}$ represents the clustering, separation, and diversity loss functions. $\boldsymbol{\theta}$ represents the training parameters of the backbone network $\boldsymbol{f}$ and prototype layer $\boldsymbol{l_p}$. The cross-entropy function ensures classification accuracy and is calculated as:

$$CrossEntropy(I,\theta) = \frac{1}{n}\sum_{i,k=1}^{n,k} -1[y_i = k]\, log(\hat{y}') \quad \textbf{(4)}$$

On the other hand, the clustering loss $\boldsymbol{R_{clus}}$ minimizes the squared distance $\boldsymbol{l^2}$ between a latent patch from a training sample and its closest prototypical vector of the same class. The expressions for these loss functions are given below:

$$R_{clus} = -\frac{1}{n}\sum_{i=1}^{n} \min_{p_i \epsilon p_{y_i}, x \in patches(s_i)} \left\| x - p_j \right\|_2^2 \quad \textbf{(5)}$$

$\boldsymbol{p_{y_i}}$ refers to the collection of prototypical vectors assigned to class $\boldsymbol{y_i}$. The $\boldsymbol{R_{sep}}$ separation-loss promotes the distance between each patch of an altered training video and the genuine prototypes, and vice versa. It is calculated as:

$$R_{sep} = -\frac{1}{n}\sum_{i=1}^{n} \min_{p_i \notin p_{y_i}, x \in patches(s_i)} \left\| x - p_j \right\|_2^2 \quad \textbf{(6)}$$

$\boldsymbol{R_{div}}$ enables the penalization of similarity between prototypes up to a certain threshold, which results in representations that are more expressive and diverse and is expressed as follows:

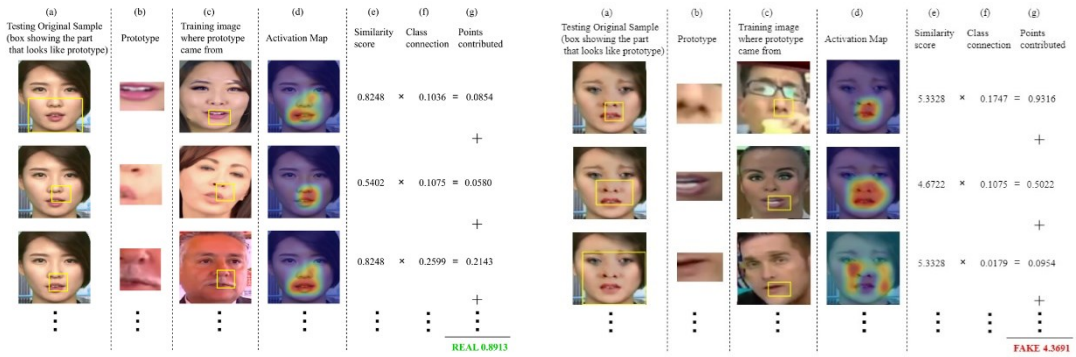
$$R_{div} = \sum_{k=1, i \neq j \rightarrow p_i, p_j \in p_k}^{K} max(0, cos(p_i, p_j) - s_{max}) \quad \textbf{(7)}$$

Prototype Projection. During the training process, we visualize the prototypes by periodically performing a projection step. This involves projecting prototype vectors to real image patches from the training set, specifically for all prototype vectors of a given class. The projection step finds the closest latent representation of a manipulated or genuine image patch within the same class. By doing this, test predictions are based on the similarities between the test sample and the learned prototypes. The prototype projection has the same temporal complexity as a convolutional layer feedforward computation followed by global average pooling. This is due to the fact that the projection step takes the shortest distance across all prototype-sized patches, whereas global average pooling takes the average of dot products across all filter-sized patches. As a result, the prototype projection adds no additional time complexity to the network training process.

Optimization function. During the training, we apply optimization $\boldsymbol{\mu}$ to the weighted matrix $\boldsymbol{w_h}$ of the fully connected layer $\boldsymbol{l_h}$, to achieve sparsity in the proposed methodology. This sparsity property reduces reliance on negative reasoning processes. The optimization problem is solved as:

$$\min_{w_h} \frac{1}{N}\sum_{i=1}^{N} CrossEntropy(l_h \circ l_p \circ f(s_i), y_i) + \mu \sum_{k=1, j:\, p_j \notin p_k}^{K} \left| {w_h}^{k,j} \right| \quad \textbf{(8)}$$

As we maintain all parameters constant in the convolutional and prototype layers, the optimization is convex. During this process, we apply $\boldsymbol{\mu}$ to the weights of the fully connected layer $\boldsymbol{l_h}$ to encourage sparsity. Specifically, $\boldsymbol{\mu}$ is a sum of the absolute values of the weights that connect to prototypes that are not associated with the current input sample. It encourages the network to use only a small subset of the prototypes for each input sample, which reduces the reliance on negative reasoning processes and can improve accuracy. It only impacts the weights of the fully connected layer $\boldsymbol{l_h}$, without altering the prototypes or the learned latent space. Therefore, the prototypes and the learned latent space remain fixed throughout the optimization process, and the improvement in accuracy is achieved purely through the sparsity-inducing $\boldsymbol{\mu}$ without changing the prototypes.



(A) (B)

Figure 2: The explainability (reasoning process) of the DFP-Net for real (A) and fake (B) class.

2.4. Visualization of the prototypes

To determine the corresponding patch of $\boldsymbol{s}$ that corresponds to the given prototype $\boldsymbol{p_n}$, DFP-Net uses the latent patch of $\boldsymbol{s}$ as $\boldsymbol{p_n}$ during prototype projection in training. This identifies the image patch that is strongly activated by $\boldsymbol{p_n}$ and uses it as the representation of $\boldsymbol{p_n}$. This is because the patch of $\boldsymbol{s}$ that corresponds to $\boldsymbol{p_n}$ should have the highest activation from $\boldsymbol{p_n}$. To locate this patch, we feed $\boldsymbol{s}$ through a DFP-Net that has been trained and up sampled the activation map generated by the prototype unit $\boldsymbol{l_{p_n}}$(prior to maxpooling) to the same size as $\boldsymbol{s}$ as shown in Prototype layer in Figure 1. The region of high activation in the upsampled activation map indicates the most strongly activated patch of $\boldsymbol{s}$ by $\boldsymbol{p_n}$. To visualize $\boldsymbol{p_n}$, we identify the region of the input image $\boldsymbol{s}$ that corresponds to the prototype by using the smallest rectangular patch. This patch contains the pixels with the highest activation values in the upsampled activation map generated by the prototype unit $\boldsymbol{l_{p_n}}$. This rectangular patch corresponds to the latent patch of $\boldsymbol{s}$ that is used as $\boldsymbol{p_n}$ during prototype projection in training. The purpose of this process is to locate the specific region of the input image that is most strongly activated by the prototype unit $\boldsymbol{l_{p_n}}$, and use it as the representation of the prototype $\boldsymbol{p_n}$. The rectangular patch that is identified contains the most relevant pixels for the prototype and can be visualized to better understand the features that the prototype is representing.

3. Experimental setup

3.1. Dataset

We evaluated our proposed methodology's performance by utilizing three distinct datasets: FaceForensics (FF)++ [13], Celeb-DF [14] and Deepfake Detection Challenge-Preview (DFDC-P) [15]. The FaceForensics++ dataset comprises over 1,000 videos, both original and manipulated, along with corresponding ground truth masks that indicate the manipulated frames. The manipulated videos were produced using four different techniques: Deepfakes, Face2Face, FaceSwap, and NeuralTextures. The videos have diverse individuals wearing glasses and having various illumination levels, making it challenging to differentiate between genuine and fake samples. The FaceForensics++ dataset has become a standard benchmark for evaluating deepfakes detection algorithms.

Celeb-DF dataset consists of more than 600,000 videos along with corresponding ground truth labels indicating whether each video is a deepfake or genuine. The genuine videos were extracted from YouTube and contain interviews of celebrities from various ethnicities, genders, and age groups. The fake videos were generated using a variety of deepfake generation techniques like face swapping and facial re-enactment. The dataset features a diverse range of subjects, including politicians, celebrities, and individuals from the general population, making it one of the most challenging and extensive datasets for deepfake detection.

DFDC-P dataset contains more than 5000 videos from paid actors, which were synthesized using deep learning techniques to create both real and manipulated faces in a variety of lighting, angles, and expressions. The dataset features a diverse range of different scenarios and subjects, including news footage, political speeches, and social media posts.

3.2. Performance evaluation of proposed method

Our proposed DFP-Net was evaluated on real and fake samples from FF++, Celeb-DF, and DFDC-P datasets through a two-stage experiment to assess its efficacy for the detection of deepfakes. In the first stage, we used our DFP-Net model to differentiate real and fake samples from each subset of the FF++ dataset. We compared real samples against fake samples from Deepfakes (DF), FaceSwap (FS), Face2Face (F2F), NeuralTextures (NT), and FaceShifter (SH) sets of FF++. The results are shown

in Table 1, where the FaceSwap set obtained the highest accuracy of 98.2%, and the FaceShifter set had the lowest accuracy of 89.1% compared to other sets. Our DFP-Net model demonstrated excellent results on all subsets of FaceForensics++ dataset, indicating its capability of detecting identity (FaceSwap, Deepfakes, FaceShifter) and expression (Face2Face and NeuralTextures) swap generated faces.

Table 1 Performance evaluation on the FaceForensics++ dataset.

	DF	FS	F2F	NT	SH
Accuracy	98.2%	98.4%	92.3%	91.1%	89.1%
AUC	0.97	0.98	0.93	0.87	0.84

Secondly, for the Celeb-DF and DFDC-P dataset, we also tested the real samples against fake samples, and the results are displayed in Table 2. The high accuracy of 96.05% achieved on the real and fake samples of the Celeb-DF dataset indicates that the DFP-Net model can accurately distinguish highly realistic face swapped samples with minimal color discrepancy and temporal flickering. This is an important achievement since highly realistic face swaps are the most challenging types of deepfakes to detect due to their high quality and minimal artifacts. Moreover, DFP-Net achieved the accuracy of 92.53% indicating that the model can differentiate between low illumination and side pose angled samples. These types of samples are difficult to classify accurately because they contain less discriminative features and pose challenges for most deepfake detection models. The high accuracy achieved on these samples shows that the DFP-Net model has learned to extract meaningful features and can classify even the most challenging samples with high accuracy.

Table 2 Performance evaluation on Celeb-DF and DFDC-P datasets.

	Celeb-DF	DFDC-P
Accuracy	96.05%	92.53%
AUC	0.95	0.93

Overall, the high accuracy and AUC scores achieved on FaceForensics++, Celeb-DF and DFDC-P datasets demonstrate the robustness and generalizability of the proposed DFP-Net model. The model can accurately detect deepfakes across different datasets, each with their unique traits and generative algorithms. This highlights the potential of the DFP-Net model to be used in real-world applications for deepfake detection.

3.3. Cross dataset evaluation

We conducted a cross-dataset experiment on FF++, Celeb-DF and DFDC-P datasets to assess the transferability and generalizability of our proposed method. This experiment consists of following scenarios: (i) training on the entire FF++ dataset and testing on Celeb-DF, DFDC-P, and vice versa, (ii) training on Celeb-DF and testing on DFDC-P, and (iii) training on DFDC-P and testing on Celeb-DF. The results of this experiment are shown in Table 3. It can be seen that FF++ trained model has shown higher accuracy on the test sets of Celeb-DF and DFDC-P as compared to other datasets. FF++ dataset is a diverse and standardized dataset comprising videos with varying lighting conditions, backgrounds, age, gender, and diverse ethnicities, featuring both expression and identity swap techniques. The DFDC-P trained model has shown lower accuracies as compared to both datasets, it may be due to the presence of very low illumination and side posed angles. Each dataset possesses unique traits and generative algorithms, making it important to test the transferability and generalizability of our proposed method across multiple datasets. The results of our cross-dataset experiment demonstrate the effectiveness of our method in achieving convincing accuracies across different datasets.

Table 3 Performance evaluation on cross-datasets.

	Test Dataset		
Train Dataset	FF++	Celeb-DF	DFDC-P
FF++	97.9%	81.26%	78.6%
Celeb-DF	74.71%	96.05%	70.04%
DFDC-P	71.97%	67.2%	92.53%

3.4. Explainability analysis of the testing image

We presented an analysis to show the explainability power and decision-making process of our DFP-Net. To achieve this, we examined how DFP-Net classifies a real sample test image using its learned prototypes. Figure 2 illustrates the decision-making process of the model, where the latent features $\boldsymbol{f(s)}$ of the test image $\boldsymbol{s}$ (Figure 2a) are compared with the learned prototypes (Figure 2b) to find evidence for its belonging to a certain class $\boldsymbol{k}$. The latent patches representation of $\boldsymbol{s}$ is compared with each prototype $\boldsymbol{p_n}$ of class $\boldsymbol{k}$ to obtain similarity scores, as depicted in Figure 2e. These scores are then used to generate an activation map, which highlights the regions of the image that are activated by each prototype (Figure 2d). In the original sample column (Figure 2a), a bounding box represents the most activated image patch of the given image for each prototype (Figure 2b). For instance, the first prototype of the real class strongly activates the lower facial area of the testing image, while the second prototype most strongly activates the right side of the nose and lips. The similarity scores between image patches and prototypical features are then weighted and averaged to obtain an overall score for the sample belonging to a specific class. This process is repeated for each class until the network correctly recognizes the testing image as a real sample. Our findings indicate that DFP-Net focuses on specific facial features, such as nose and lips, to accurately

detect the real and fake samples. By providing insights into the decision-making process of DFP-Net, our study improves the transparency and interpretability of our model.

3.5. Comparative analysis with the state-of-the-art methods

To evaluate the effectiveness of the proposed DFP-Net for deepfake detection, we compared it with contemporary methods. In this comparative analysis, we focused solely on those methods that emphasize the interpretability of deepfake detection. Specifically, we compared the performance of DFP-Net with [3, 5, 6] on FaceForensics++ dataset, and with [4, 6] on Celeb-DF, and DFDC-P datasets. The results, presented in Table 4, show that DFP-Net outperforms or matches the performance of existing methods. Notably, DFP-Net achieves the highest accuracy on subsets based on expression swapping, namely Face2Face and NeuralTextures, with accuracy gains of 27.7% and 35.6%, respectively, compared to [3]. These subsets are particularly challenging to detect due to subtle semantic changes, but our model demonstrates superior capability in distinguishing them compared to contemporary methods. In contrast, [3] using contrastive learning shows the lowest accuracy on all subsets of FF++ due to sensitivity to negative sample selection and requires a more complex model that demands greater computational resources. Our DFP-Net has shown relatively lower accuracy on FaceShifter set, which is attributed to the difficulty of detecting faces generated by this algorithm with minimal semantic changes.

When our DFP-Net is compared with methods [4,6] evaluated on Celeb-DF and DFDC-P, it achieved the highest accuracy for both the datasets as shown in Table 4. Compared to [4], DFP-Net achieved a higher accuracy of 2.65% on Celeb-DF, while on DFDC-P, it outperformed [6] with an accuracy gain of 0.53%. It is worth noting that in [4], an ensemble of weakly supervised models was used for the detection of deepfakes involving training using limited labeled data and unlimited unlabeled data. This model achieved the accuracy of 93.64% and 92.4% on Celeb-DF and DFDC-P, respectively. However, DFP-Net achieved higher accuracy than [4] on both datasets.

DFP-Net has achieved the highest accuracy among most state-of-the-art methods on FaceForensics++, Celeb-DF, and DFDC-P datasets. The superior performance indicates that the DFP-Net can adapt well to different types of manipulations. The deepfake techniques used to create manipulated images can vary in terms of the level of detail, complexity, and type of manipulation applied. For instance, some deepfake techniques may involve subtle changes in facial expressions, while others may involve more drastic changes, such as replacing an entire face with another person's face. DFP-Net's ability to detect these manipulations with high accuracy demonstrates its versatility and robustness in detecting various deepfake techniques.

Table 4 Comparative analysis with the state-of-the-art methods.

Paper	DF	FS	F2F	NT	SH	FF++	Celeb-DF	DFDC-P
[3]	83.9	49.7	64.6	55.5	-	-	-	-
[6]	98.9	98.0	62.4	75.0	97.7	97.1	93.9	92.0
[4]	-	-	-	-	-	-	93.64	92.4
[11]	-	-	-	-	-	98.2	-	-
DFP	98.2	98.4	92.3	91.1	89.1	97.9	96.05	92.53

4. Ablation study

An ablation study was carried out on the FaceForensics++ dataset to showcase the efficacy of different convolution neural network backbone architectures, the impact of various activation functions on backbone network, and prototype layer for the detection of deepfakes.

4.1. Choosing backbone architecture

This experiment was designed to assess the accuracy of various convolutional neural networks (CNNs) in detecting deepfakes. The experiment was performed on the FaceSwap and Deepfakes set of FaceForensics++, where real samples were compared with fake samples to analyze the effectiveness of the model. A range of pre-existing neural networks including VGG-16, ResNet-101, ResNet-152, DenseNet-121 and DenseNet-201 with their corresponding frameworks (ReLu activation function) were employed in the backbone architecture, and the findings are presented in Table 5. Compared to VGG-16, which achieved a moderate accuracy of 73.4% and 73.98% on FaceSwap and Deepfakes subsets respectively, ResNet-152 demonstrated better performance with accuracy of 80.15% and 82.83% on the same subsets. Similarly, ResNet-101 exhibited an accuracy of 79.33% and 80.61% on the FaceSwap and Deepfakes subsets, respectively. However, DenseNet-121 outperformed all other models, achieving the highest accuracy of 84.1% and 86.97% on FaceSwap and Deepfakes subsets. DenseNet-201 also showed promising results with an accuracy of 84.26% and 87.08% on the respective subsets.

The exceptional performance of DenseNet-121 and DenseNet-201 may be due to the feature reuse property, which connects each layer to every other layer in a feedforward manner. This aids in the efficient propagation of information across the network, allowing the model to achieve faster convergence and learn more discriminative features for deepfake detection. In contrast, VGG-16, ResNet-101 and ResNet-152 are not as efficient in propagating information across the network, which can limit their ability to capture subtle patterns in the data. Furthermore, the parameter efficiency of DenseNet-121

and DenseNet-201 contributes to improved generalization performance and reduced risk of overfitting. These results validate our choice of DenseNet-121 as the backbone network in our proposed network and suggest that DenseNet-201 shows potential for deepfake detection.

Table 5 Performance evaluation on various backbone architectures.

Network	FaceSwap	Deepfakes
VGG-16	73.4%	73.98%
ResNet-152	80.15%	82.83%
DenseNet-121	**84.1%**	**86.97%**
ResNet-101	79.33%	80.61%
DenseNet-201	84.26%	87.08%

4.2. Choosing activation function for backbone architecture

This experiment was aimed to analyze the impact of various activation functions on the backbone network. We considered three commonly used activation functions; ReLu, GeLu and Swish. These functions were considered due to their prevalence in the literature and their ability to improve the network's performance. The real and fake samples of FaceSwap and Deepfakes set were compared with each other to evaluate the performance of the activation function on the backbone architecture (DenseNet-121). It can be seen from the results in Table 6 that the Swish activation function outperformed the other functions in terms of accuracy. Swish has a smoother curve than ReLu and GeLu, making it easier to optimize and avoid the dying ReLu problem, resulting in faster convergence during training. Furthermore, Swish's non-monotonic behavior assists the model in learning more complex patterns in the data, which contributes to its superior performance when compared to ReLu and GeLu.

Table 6 Performance evaluation on various activation function for backbone network.

Activation Function	FaceSwap	Deepfakes
ReLu	84.1%	86.97%
GeLu	90.13%	94.3%
Swish	**98.4%**	**98.2%**

4.3. Choosing activation function for prototype layer

This experiment was carried out to analyze the impact of various activation functions on prototype layer of DenseNet-121 having Swish function. The performance of the Logarithmic and Linear activation function was evaluated using the real and fake samples of FaceSwap and Deepfakes set of FF++. These activation functions were considered because they are commonly used and have different characteristics that can affect the performance of the prototype layer. The results shown in Table 7 indicate that the Logarithmic function outperformed the Linear function. This is due to the Logarithmic function's ability to compress a wide range of input values into a narrow range of output values, which is especially useful for prototypes aiming to represent input data with a small number of prototype vectors. The prototype layer plays a vital role in learning a condensed and informative representation of the input data, where the Logarithmic activation function can capture significant patterns and features while keeping the output values manageable. The Linear activation function, on the other hand, lacks nonlinearity, making it difficult to capture intricate relationships between input and output, resulting in a less effective representation of input data in the prototype layer. Hence, Logarithmic activation function is a viable option for compressing input values and capturing intricate relationships between input and output, resulting in a more effective representation of input data in the prototype layer.

Table 7 Performance evaluation on various activation function for prototype layer.

Activation Function	FaceSwap	Deepfakes
Logarithmic	**98.4%**	**98.2%**
Linear	91.99%	90.36%

5. Conclusion

In this paper, we have presented a novel DFP-Net that uses prototype-based learning to generate an interpretable and explainable deepfake detection method. We developed a deepfakes detection system by leveraging the power of the prototype-based learning technique, which provides insights into the model's decision-making process, making it more transparent and trustworthy. Our experimental findings on FaceForensics++, Celeb-DF and DFDC-P datasets, including cross-dataset experiments, show that our method outperforms traditional black-box machine learning models in deepfake detection. Our research results emphasize the significance of developing transparent and interpretable deepfake detection methods to ensure the authenticity and trustworthiness of video content in the digital age. In the future, we intend to investigate more interpretability techniques and the application of our method for the detection of unified audio-visual deepfakes.

Acknowledgment This work is supported by the grant of Punjab Higher Education Commission (PHEC) Pakistan via Award No. (PHEC/ARA/PIRCA/20527/21), National Science Foundation (NSF) via Award No. 2231619 and Michigan Transnational Research and Commercialization (MTRAC), Advanced Computing Technologies (ACT) via Award No. 292883.

References

[1] Jonathan Johnson, Interpretability vs Explainability: The Black Box of Machine Learning, accessed on April 11, 2023, Available at https://www.bmc.com/blogs/machine-learning-interpretability-vs-explainability/

[2] A. Erasmus, T. D. Brunet, and E. Fisher. What is interpretability? Philosophy & Technology, 34(4): 833-862, 2021.

[3] Y. Xu, K. Raja, and M. Pedersen. Supervised contrastive learning for generalizable and explainable deepfakes detection. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 379-389, 2022.

[4] S. H. Silva, M. Bethany, A. M. Votto, I. H. Scarff, N. Beebe, and P. Najafirad. Deepfake forensics analysis: An explainable hierarchical ensemble of weakly supervised models. Forensic Science International: Synergy, 4:100217, 2022.

[5] Y. Xu, K. Raja, L. Verdoliva, and M. Pedersen. Learning Pairwise Interaction for Generalizable DeepFake Detection. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 672-682, 2023.

[6] F. Khalid, A. Javed, H. Ilyas, and A. Irtaza. DFGNN: An Interpretable and Generalized graph neural network for deepfakes detection. Expert Systems with Applications, 222:119843, 2023.

[7] C. Chen, O. Li, D. Tao, A. Barnett, C. Rudin, and J. K. Su. This looks like that: deep learning for interpretable image recognition. Advances in neural information processing systems, 32, 2019.

[8] P. Hase, C. Chen, O. Li, and C. Rudin. Interpretable image recognition with hierarchical prototypes. In Proceedings of the AAAI Conference on Human Computation and Crowdsourcing, 7:32-40, 2019.

[9] M. Nauta, A. Jutte, J. Provoost, and C. Seifert. This looks like that, because... explaining prototypes for interpretable image recognition. In Machine Learning and Principles and Practice of Knowledge Discovery in Databases: International Workshops of ECML PKDD 2021, Virtual Event, 441-456,2022.

[10] M. Nauta, R. Van Bree, and C. Seifert. Neural prototype trees for interpretable fine-grained image recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 14933-14943, 2021.

[11] L. Trinh, M. Tsang, S. Rambhatla, and Y. Liu. Interpretable and trustworthy deepfake detection via dynamic prototypes. In Proceedings of the IEEE/CVF winter conference on applications of computer vision, 1973-1983, 2021.

[12] J. Xiang, and G. Zhu. Joint face detection and facial expression recognition with MTCNN. In 2017 4th international conference on information science and control engineering (ICISCE), IEEE, 424-427, 2017.

[13] A. Rossler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Nießner. Faceforensics++: Learning to detect manipulated facial images. In Proceedings of the IEEE/CVF international conference on computer vision, 1-11, 2019.

[14] Y. Li, X. Yang, P. Sun, H. Qi, and S. Lyu. Celeb-df: A large-scale challenging dataset for deepfake forensics. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 3207-3216, 2020.

[15] B. Dolhansky, R. Howes, B. Pflaum, N. Baram, and C. C. Ferrer. The deepfake detection challenge (dfdc) preview dataset. arXiv preprint arXiv:1910.08854, 2019.