From Interpretable Filters to Predictions of Convolutional Neural Networks with Explainable Artificial Intelligence

Shagufta Henna, Senior Member, IEEE, Juan Miguel Lopez Alcaraz,

Abstract—Convolutional neural networks (CNN) are known for their excellent feature extraction capabilities to enable the learning of models from data, yet are used as black boxes. An interpretation of the convolutional filters and associated features can help to establish an understanding of CNN to distinguish various classes. In this work, we focus on the explainability of a CNN model called as CNNexplore that is used for Covid-19 and non-Covid-19 classification with a focus on the interpretability of features by the convolutional filters, and how these features contribute to classification. Specifically, we have used various explainable artificial intelligence (XAI) methods, such as visualizations, SmoothGrad, Grad-CAM, and LIME to provide interpretation of convolutional filters, and relevant features, and their role in classification. We have analyzed the explanation of these methods for Covid-19 detection using dry cough spectrograms. Explanation results obtained from the LIME, SmoothGrad, and Grad-CAM highlight important features of different spectrograms and their relevance to classification.

Index Terms—Deep Learning Model Explanation, XAI, Explainable AI, Transparency in AI, Grad-CAM, LIME, Smooth-Grad.

I. INTRODUCTION

Artificial intelligence (AI) has several applications in various industries to improve goods, methods, and research. Recent years have demonstrated significant success in the AI subfield known as deep learning (DL)/neural networks due to its good performance for multidimensional data. The deep learning model consists of multiple processing layers, to model complex nonlinear input-output relationships, and the ability to perform pattern recognition and feature extraction from low-level input data. Although, these deep learning approaches, can capture non-linear relationships in input data, however, often comes at the cost of limited interpretability. DL models are notorious to elude immediate interpretability by humans, thereby limiting their applications in certain situations, such as medical diagnosis. Explainable artificial intelligence (XAI) aims to explain and interpret the internals of machine learning or deep learning models [1]. The role of XAI in medical diagnosis plays a critical role, due to clinicians who want to consider AI-driven decisions as a second opinion. The explainability of “how” the deep learning model makes a prediction may seem a global explanation of the network, however, the explainability of the “why” is more inclined toward the contribution of features towards classification.

Undoubtedly, XAI provides a better environment for deep learning applications in many ways, and it comes with limitations and challenges. As an example, when it comes to the raw explainable results based on visualization, there are two main concerns as reviewed by [1]. Firstly, there is human attention incapacity to deduce XAI decision-making explanation maps. Secondly, appropriate explainable AI techniques are unavailable to quantitatively provide better interpretability and explanation to augment visualizations.

The challenges that XAI present in healthcare have recently been investigated by various researchers, specifically for the clinical decision support system (CDSS) [2]. A work by [3] found that pathologists who had local model interpretations before the model adoption require more insights on model characteristics for complete trustworthiness. Further, authors in [4] show that identifying the rationale for explainability aids in the selection of XAI approaches that can fill gaps when creating user experiences. Another work by [5] suggest that to obtain user acceptability, a CDSS should have high tractability, which necessitates human reasoning, stepwise inference, explanation capability, and user-familiar terminology. Overall, the explainability for CDSS demands easily understandable terminologies and an explanation of the neural network architecture.

The trend of XAI for medical applications is relatively new, however, there are some efforts in this domain to explain the role of deep neural networks. [6] implemented a gradient-weighted class activation map (Grad-CAM) algorithm for Covid-19 features detection using X-ray images. Another similar work by [7] implemented a layer-wise relevance propagation (LRP) algorithm to highlight breast cancer features in mammogram images. [8] proposed the Grad-CAM algorithm and its derivatives for the explainability of traffic accident anticipation in automated driving systems. Another work in [9] retrieved important job skills from a job classification task with local interpretable model-agnostic explanations (LIME) algorithm. Authors in [10] proposed an explainable algorithm that analyzes the performance of various explainable algorithms by removing noise for disease detection. [11] implemented an explainable algorithm based on dimensionality reduction for traffic classification. [12] presented an explainable method based on a hybrid neural network for acoustic scene classification.

Shagufta Henna and Juan Miguel Lopez Alcaraz are with the Department of Computing, Atlantic Technological University, Donegal, Ireland. Manuscript received April 19, 2021; revised August 16, 2021.
ability of neural networks focus on model internals. There is no work, specifically convolutional neural networks (CNN) to explain the CNN filters to interpret features and their role in Covid-19 classification.

This research focuses on the explainability of CNN, a popular neural network for image classification with a particular focus on its filters. Further, it also investigates various explainability approaches for the local explanation of features interpreted by CNN filters and the decisions made by the classification layer. The major contributions are listed as follows:

- We evaluate the accuracy of CNN for Covid-19 detection using dry cough to show its suitability for explainability. From an application point of view, this is different from [6] and [13] because their approaches detect Covid-19 based on X-ray and CT images respectively. On the other hand, our work focuses on cough sounds that can support a faster and cheaper diagnosis.
- We presented features visualization of CNN to show a positive correlation between the complexity of patterns learned by the network. The proposed convolutional filters visualization is the most relevant pattern in a deep neural network. On the contrary, this work puts an effort to analyze the patterns learned by the filters of each hidden layer of CNN.
- We showed that the visualization of CNN layers helps to understand a clear and distinct characteristic between a positive and a negative class to help interpret the model predictions. Our work focuses on features that the CNN related to each class using the process of activation maximization. This is different from the work of [14] which uses learned patterns to distinguish the classes.
- We evaluated various XAI methods such as Smooth-Grad, Grad-CAM, and LIME to explain the CNN filters and classification when applied for Covid-19 detection using dry cough. This differs from the existing works that focus on the application of saliency maps to explain model internals, and segmentation by occlusion to evaluate the feature importance using inputs and outputs.

II. CONVOLUTIONAL NEURAL NETWORK FOR COVID-19 DETECTION

This section presents the CNN model used for Covid-19 classification using dry cough spectrograms. The model internals as shown in Figure 1 that are later explained using various XAI approaches in Section III.

Figure 1 shows that the CNN takes a spectrogram as input and consists of three convolutional layers with the primary function to extract features from the pixeled images. This is followed by a flattening process converting the two-dimensional image to a single-dimensional matrix suitable for the fully connected layer. The fully connected layer enables learning based on non-linear combinations of extracted features and its output is passed to another fully connected layer that outputs the probabilities of Covid-19 and non-Covid-19 classes.

Algorithm 1 shows the steps of CNN used for Covid-19 detection. The algorithm performs classification as Covid-19 or non-Covid-19 using the spectrogram input. It computes the loss using the predicted label $\hat{y}$ and true label $y$ that is later backpropagated to adjust the weights and bias using an optimizer. The basic steps given in Algorithm 1 are explained in the sections below.

![Fig. 1. Convolutional neural network for Covid-19 detection.](image)

A. Dry Cough Audio to Spectrograms

The neural network in Algorithm 1 takes spectrograms as an input. These spectrograms are the result of conversion from audio to images, representing the meaningful audio information to train a convolutional neural network. Figure 2 shows the wave plot of the dry cough audio. The y-axis...
represents the amplitude of the cough sound, whereas, the x-axis shows the time of the audio using a sampling rate of 44,100 hertz. The wave plot in the Figure shows approximately 308,700 data points for 7 seconds that reflects the information contained in the audio.

Figure 3 shows the scaled Mel spectrogram generated using two steps: magnitude spectrogram generation, and mapping from spectrogram to Mel scale. The magnitude spectrogram is computed by applying the short-time Fourier transform (STFT) \[18\].

\[
Y(m,k) = |S(m,k)|^2
\]  

(1)

SFTF generates a matrix of magnitudes that are later squared to obtain an image representation of spectrograms as given in Equation II-A. In the Equation, \(m\) denotes the signal and \(k\) as the window of the spectrogram computation. Later, the frequencies of magnitude spectrogram are mapped to the Mel scale as given in Equation II-A [15]. The scaled Mel-spectrogram in Figure 3 shows low frequencies after amplification and high frequencies using compression with the help of the Minmax scaling process to fit the values into the 8-bit range, thereby retaining significant information in a smaller size image.

\[
m = 2595\log_{10}\left(1 + \frac{f}{700}\right)
\]  

(2)

Figure 5 shows the scaled Mel spectrogram using inversion of color intensity, where black areas represent high and white as low energies, respectively. This results in a final spectrogram, where the flipping of the image moves the low frequencies at the bottom, making it ready as the input to the CNN.

B. CNN Initialization

Algorithm 1 takes the weights \(w\) and biases \(b\) to train all the layers of the CNN that are later adjusted to minimize the training cost or loss. The weights in convolutional layers are interpreted as kernels, i.e., a weight matrix multiplied with input to extract relevant features. For 2D convolutions, kernels, as used in the Algorithm 1, are also called filters. The bias shifts the activation function by adding a constant value to it. The weights of the neural network are initialized using the Glorot uniform based on the size of the neural network as given in Equation II-B [16]. The Equation randomly initializes the weights of the neural network using the number of input and hidden units in the neural network. Using the 4 input and 5 hidden units, the Equation calculates the random weights in a range of -0.82 to +0.82. The bias is initialized with a value of 0.

\[
GU_{\text{limits}} = \sqrt{\frac{6}{N_{\text{in}} + N_{\text{out}}}}
\]  

(3)

C. Feature Extraction using Convolutional Filters

Figure 5 illustrates the basic process of applying a kernel or filter to the normalized spectrogram. It shows the complete feature map after passing the filter through all the possible positions on the Mel-Spectrogram. The filter represents the weight with the values initialized using the Glorot uniform function given in Equation II-B [16]. We have considered a filter size of 3 × 3. However, the size can be adjusted based on the feature retrieval specific to the area of a Mel-Spectrogram. This filter is later multiplied with the dry cough spectrogram, followed by a dot product with added bias later to generate a value [17]. Finally, the resultant values are added to a feature map. The same filter convolutes from left to right and up to down with the Mel-Spectrogram adding more values to the feature map. The values in the feature map correspond to Mel-Spectrogram pixels while maintaining class-specific pixel correlation.

D. Activation and Max Pooling

Equation II-D shows the activation function for the first three convolutional and the first fully connected layer of the CNN. The activation function is based on the Rectified Linear Unit (ReLU) activation with values between 0 and the input value (Nair and Hinton, 2010).

\[
R(z) = \max(0, z)
\]  

(4)
Algorithm 1 uses two dense layers, each with its weights and same layer. Our model used for explainability as given in Algorithm 1 considers the pool size of 2 × 2, although its size can be adjusted based on the input and convolutional filters.

E. Dropout and Flattening

Once Max pooling is complete, dropout is applied for the removal of input-output connections to regularize the network. In the neural network for Covid-19 classification, the dropout process drops approximately 25% of the features and is applied 4 times in Algorithm 1. Once the dropout process is complete, we consider the flattening process for the CNN that converts the 2-dimensional feature map to a 1-dimensional feature space to feed to the fully connected layer as given in Algorithm 1.

The fully connected layer also known as the dense layer connects each unit of the last layer with all the units of the same layer. Our model used for explainability as given in Algorithm 1 uses two dense layers, each with its weights and bias.

F. Covid-19 Classification and Training

This section illustrates the Covid-19 and non-Covid-19 classification using the dry cough spectrogram using the CNN Algorithm 1 This classification is explained later using various explainable AI approaches. Algorithm 1 shows the use of the SoftMax activation function as given in Equation II-F based on the features extracted by the convolutional filters. SoftMax function assigns a confidence score to each class by applying an exponential function.

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{c} e^{z_j}}$$  \hspace{1cm} (5)

Equation II-F shows how SoftMax function calculates the probability $P$ of each class $y$ at given timestep $t$ from the $k$ classes, i.e., Covid-19 and non-Covid-19 from the input features $f$. The function results in a predicted class called $\hat{y}$.

$$\hat{y}_{t,k} \equiv P(y_t = k | x) = \frac{\exp(w_k^{(5)} h_t^{(4)} + b_k^{(5)})}{\sum_k \exp(w_k^{(5)} h_t^{(4)} + b_k^{(5)})}$$  \hspace{1cm} (6)

Line 16-18 illustrates the loss computation, i.e., the difference between true class label $y$ and predicted class label $\hat{y}$, that is backpropagated from the output layer to the input layer to adjust the weights of each layer using the Adam optimizer. Adam optimizer is based on adaptive moment estimation. It incorporates anticipated values $E$ of previous gradients as given in Equation II-F. Adam optimizer function calculates momentum using various steps consisting of distinct parameters, thereby leading to faster convergence.

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{E[G_{t,ii} + \epsilon]}} \cdot E[g_{t,ii}]$$  \hspace{1cm} (7)

III. EXPLAINABILITY OF CNN FOR COVID-19 DETECTION

This section discusses the explainable AI approaches to help interpret the deep learning model given in Algorithm 1 and explained in Section II. This section starts with the visualization of features to help interpret the role of these features in the Covid-19 classification. After feature visualizations, we have used SmoothGrad Algorithm to highlight the importance of features of a spectrogram to Covid-19 predictions. This is followed by the GradCAM to provide visualization of the spectrogram to indicate the important regions in a feature map in the last convolutional layer as given in Algorithm 1. Finally, explainable AI approach LIME provides the feature importance of a spectrogram using segments.

A. Feature Visualization by Optimization

We have used two approaches of visualization to explain the features and the learning of convolutional neural network given in Algorithm 1. The first approach as given in Algorithm 2 helps to interpret convolutional filters using visualization, whereas the second one given in Algorithm 3 provides an interpretation of dense layers in Algorithm 1. These algorithms involve the use of a loss function to maximize the activation of desired neurons within the selected layer.

Algorithm 2 helps to understand the first convolutional layer filter for simple textures, and the second from Algorithm 3 for complicated textures and patterns from the dry cough spectrogram. The last convolutional filter in Algorithm 1 learns objects or fragments of objects from the spectrogram. In the case of dry cough spectrograms, these objects are various frequencies activated according to the dry cough audio, thereby making its interpretation significantly challenging.

Algorithm 2 presents the steps required for the visualization of convolutional filter. The first step extracts the required convolutional layer from the CNN model. Later, it enables the linear activation function to take the raw input. This is followed by the creation of a new model with the same inputs as the first model and the output of the last layer. Slicing the model into the desired number of convolutional layers transforms it to a new model that does not change the
Algorithm 2: Feature visualization of convolutional filters.

Data: \( \text{cnn} \) from Algorithm 1
Result: Filter \( A_{\text{desired}} \) of desired
1. \( \text{model} \) filters \( \leftarrow \emptyset \)
2. for each layer \( i \) in \( \text{cnn} \) do
   3. if layer \( i \) is convolutional then
      4. \( \text{layer}^\text{conv} \) \( \leftarrow \text{layer} \)
   5. end
   6. \( \text{layer}^\text{conv} \).set activation(linear) \( \triangleright \) set the model activation as linear
   7. \( \text{model} \) filters.inputs \( \leftarrow \text{cnn} \).inputs
   8. \( \text{model} \) filters.outputs \( \leftarrow \text{layer}^\text{conv} \).outputs \( \triangleright \) set the model activation as linear
3. end
4. for each filter \( i \) in \( \text{model} \) filters do
   5. if filter \( i \) is desired then
      6. \( A_{\text{desired}} \leftarrow \text{filter} \)
   7. end
   8. return \( A_{\text{desired}} \) \( \triangleright \) return the desired filter for visualization
9. end

Algorithm 3: Feature visualization of dense layer.

Data: \( \text{cnn} \) from Algorithm 1
Result: \( \text{model} \) dense
1. for each layer \( i \) in \( \text{cnn} \) do
   2. if layer \( i \) is dense then
      3. \( \text{layer}^\text{dense} \) \( \leftarrow \text{layer} \)
   4. end
   5. \( \text{layer}^\text{dense} \).set activation(linear) \( \triangleright \) set the model activation as linear
   6. \( \text{model} \) dense.param \( \leftarrow \text{cnn} \).param
   7. \( \text{model} \) dense.outputs \( \leftarrow \text{layer}^\text{dense} \).outputs \( \triangleright \) set the dense model output as outputs of the dense layer
3. end
4. for each unit \( k \) in \( \text{model} \) dense do
   5. \( \text{img} \) \( \leftarrow \text{createImage} \) (model \( \text{dense} \).inputSample)
   6. \( \text{model} \) dense \( \leftarrow \text{maxActivation}(\text{img} \) \( \text{dense} \) ) \( \triangleright \) maximize activation of Image in each unit of dense layer
7. end

first model by collecting its outputs. Lines 9 to 12 iterates over desired filters for visualization returning an image to distinguish the Covid-19 or non-Covid-19 classes based on specific neural network level patterns. These patterns along with the filters and parameters are learned during the first model training.

Algorithm 4 lists the steps used for the visualization of the last dense layer using the SoftMax classification. The first step in the algorithm extracts the desired layer and changes the SoftMax as an activation function for the analysis. This change brings desirable linearity to the model outputting the probabilities of the Covid-19/non-Covid-19 classes with a sum of 1. The second sets the layer’s activation function to linear. Similar to Algorithm 3 the third creates a new model based on the specification of the first model. This is to prevent the modification to the existing model while explicitly revealing the CNN model internals at this stage. This is followed by an iterative process to apply the activation maximization function that generates a random and optimized image for Covid-19/non-Covid-19 as predicted by the Algorithm 1.

B. Smooth Gradient (SmoothGrad) for \( \text{cnn} \) Model Interpretation

This section presents SmoothGrad to interpret \( \text{cnn} \) in Algorithm 1. The method focuses on the explanation of the feature importance of input dry cough Mel-spectrogram image as given in Figure 3 and outputs a sensitive map. Specifically, the SmoothGrad aims to maximize the class score as given in Equations 8 and 9.

\[
\text{class}(x) = \arg\max_{c \in C} S_c(x) \quad (8)
\]

\[
M_c(x) = \frac{\partial S_c(x)}{\partial x} \quad (9)
\]

Algorithm 4: SmoothGrad.

Data: Image \( x \), noise spread level \( N(0, \sigma^2) \), number of samples \( N \)
Result: SmoothGrad sensitive map \( M'_c \)
1. totalImages \( \leftarrow x.n \)
2. for each image \( x \) in totalImages do
   3. \( x \leftarrow x + N(0, \sigma^2) \)
   4. \( S_c(x) = \text{cnn} \).predict(x)
   5. \( M_c(x) = \frac{\partial S_c(x)}{\partial x} \)
   6. \( M'_c \leftarrow \sum_{i=1}^{N} M_{i}(x) / N \) \( \triangleright \) sensitive map smoothed with noise base
7. end

For the given input features \( x \) extracted from the dry cough Mel-spectrogram, Activation function in Equation 8 selects a class \( c \), i.e, covid-19/non-covid-19 from the set of classes \( C \subseteq \{\text{Covid-19}, \text{non-Covid-19}\} \) to maximize the class score. Equation 9 computes a sensitive map \( M_c(x) \) for the input Mel-spectrogram where \( \partial S_c \) represents the derivative gradient of the class score function for input \( \partial x \). This measures any change in classification score concerning the change in each feature, i.e., pixels of the Mel-spectrogram. This map, however, also includes noise or random pixels in Mel-spectrogram. This is attributed to the local variations in score function \( S_c \) during the backpropagation process.

Equation 10 selects the saliency map \( M'_c(x) \) of a Mel-spectrogram estimating the local average of the gradient values with a stochastic approximation. \( M'_c(x) \) is computed by sampling similar images by adding random noise with a gaussian distribution with a mean of 0 and standard deviation to set \( N(0, \sigma^2) \), then averaging the resultant sensitive maps for each Mel-spectrogram for \( n \) samples.
Algorithm 4 shows steps for sensitive map calculation. Lines 2-3 illustrate the iterative step to apply noise using a Gaussian distribution with a mean of 0 and a standard deviation \( N(0, \sigma^2) \) to each Mel-spectrogram. This augmented Mel-spectrogram \( x \) is then classified by the \( \text{cnn}_{\text{explain}} \) model returning a class score. The application of noise during this process improves sensitive map called as stochastic resonance process. This process replicates the Mel-spectrogram with random noise that strengthens the pixel patterns between the Mel-spectrogram samples without amplifying the random noise. Line 5 calculates the sensitive map with the partial derivatives of the class score \( M'_c(x) \) for the augmented Mel-spectrogram \( x \). Finally, line 6 returns the sensitive map smooth with noise base on the average of sensitive maps \( \frac{\sum^m_{i=1} M_c(x)}{n} \) for each augmented Mel-spectrogram. Later, during the performance evaluation, Algorithm 4 provides meaningful insights for Covid-19/ non-Covid-19 class for the input Mel-spectrogram.

C. Gradient-weighted Class Activation Mapping (Grad-CAM)

Similar to SmoothGrad, Grad-CAM outputs a sensitivity map for a Mel-spectrogram image. However, in contrast to SmoothGrad, it emphasizes the areas of importance by the feature maps in the last convolutional layer for the Covid-19/non-Covid-19 classification. Extracting the importance of pixels in a Mel-spectrogram by the last layer/feature map leads to the collection of both the Covid-19 and non-Covid-19 classes that can be predicted. In our work, we focus on a single class to conduct an analysis of class differentiation using the activation of each sample area.

To collect the importance of each pixel in a spectrogram for image classification, derivatives by backpropagation are required. This process can lead to confusion in the feature’s importance collection during training. However, in this work, we have considered the backpropagation at an inference phase. This is given in Algorithm 5. It computes the global and averaged pooled gradients \( \alpha^i_k \) of each feature map \( A^k \) passed through the ReLU.

Algorithm 5 presents all the steps by the GradCAM to compute the sensitive map. Line 1 passes the Mel-spectrogram using the forward pass in the last convolutional layer of the \( \text{cnn}_{\text{explain}} \) model. Line 2-3 collects the score of the desired class \( c, \) i.e., Covid-19 or non-Covid-19, and sets the score of the other class to 0. Lines 4-6 refer to the backpropagation of the class score for the Mel-spectrogram in the convolutional filters to calculate the gradients, representing the feature importance of the Mel-spectrogram as \( A^k \). Lines 7-9 iteratively perform global average pooling of gradients of each feature map, returning the weights of each feature map. Line 10 computes the weighted sum of the feature maps with their respective weights, called the sensitive map. In the latter step, negative values in the sensitive map are eliminated using the ReLu function, thereby highlighting only the positively correlated features in the Mel-spectrogram.

W have resized the sensitive map to facilitate its visualization and comparison for the corresponding Mel-spectrogram. This is needed as the convolutional filters of the last layer are smaller than the original Mel-spectrogram input because of the image transformation. Algorithm 5 results in map \( GC'_+ \) highlighting the most important areas of a Mel-spectrogram by each feature map \( A \) corresponding to the last convolutional layer \( h_t^{-1} \) contributing significantly to the prediction covid-19 or non-covid-19 class.

D. Local Interpretable Model-Agnostic Explanations (LIME)

The fourth method, we have considered for the explainability of \( \text{cnn}_{\text{explain}} \) is called the LIME algorithm. LIME considers image classification with a focus on specific areas of an image contributing to the Covid-19/non-Covid-19 class. In contrast to SmoothGrad and GradCAM that average sensitive maps and compute a sensitive map from the last filter, the LIME algorithm aims to identify the areas of importance by occlusion. LIME uses an interpretable model to interpret a model by approximating the black-box \( \text{cnn}_{\text{explain}} \). This interpretable model supports straightforward computations to understand than the algorithm trying to explain. LIME is also called the model-agnostic approach due to its functional independence, i.e., zero-intervention with the functionality of model internals. This feature makes LIME the only method to explain the \( \text{cnn}_{\text{explain}} \) without seeing its internal functions.
**Algorithm 6: LIME.**

| Data: | Image $x$, number of features $n$, $c_{explain}$ model |
|-------|----------------------------------------------------------|
| Result: | Essential segments of class $y$, segments $s$ |

1. segments $s = \text{Cluster}(x)$
2. combinations $c = \frac{s_1!}{r!(S-r)!}$, where $r = S-1, S-2$
3. if $(s \in c)$ then
   4. $s \leftarrow$ average as $s'$ for $s \in c$
   end
6. $P_x = 150(c) \cdot s$ \hspace{1cm} \triangleright$perturbed images.
7. $P = c_{explain} \cdot \text{predict}(P_x)$ \hspace{1cm} \triangleright$predictions.$
8. for $P_x \in P_x$ do
   9. $d = \frac{P_x \cdot P_{x'}}{|P_x|}$
   10. $D \leftarrow D \cup d$
   11. return $D$ \hspace{1cm} \triangleright$return distances $D$.
end
end
14. $d_{min} \leftarrow 0$
15. $d_{max} \leftarrow 1$
16. for $d \in D$ do
   17. $W \leftarrow d$
   18. $LM \leftarrow \text{createLinear()}$ \hspace{1cm} \triangleright$create interpretable linear model$
   19. $y \leftarrow LM \cdot \text{predict}(P \cdot W)$ \hspace{1cm} \triangleright$predictions of desired class $c$ using $LM$.$
   20. segments$_y = \text{coefficients}(n, LM)$ \hspace{1cm} \triangleright$n most significant coefficients of LM$
21. end

Algorithm 6 presents the steps required for the explainability by the LIME. The first step segments the input image to be explained with the help of a quick-shift segmentation algorithm. The algorithm computes the segments, also known as super-pixels or features in the image. Line 2 in Algorithm 6 creates the possible combinations of the segments. This involves including or excluding these segments by the factorial product of the total segments $S!$ and segments $r!$ at each combination. The inclusion of segments in each combination provides better differentiation later in the model. Line 3-6 in the algorithm creates 150 perturbed images based on the segment’s activations in 150 random combinations. If a segment is excluded in the combination, an average of the other segments is added to keep the image size excluded from meaningful information.

Line 6 shows the Covid-19/non-Covid-19 class predictions with the $c_{explain}$ on the 150 perturbed images. This returns a class core for each perturbed image for each class. Line 8-13 starts a loop over the set of perturbed images which computes a cosine distance towards the original image. This distance measures the similarity or differences of the permuted image with the original image containing all the feature segments in it. Lines 14 and 15 assign these distances in a range between 0 and 1 to interpret the values as weights of importance for each perturbation image. Step 16 to 19 creates an easily interpretable linear model in which outputs are the predictions towards the covid-19/non-Covid-19 class.

These predictions are based on the perturbed images with their respective importance of weights. Lines 20-21 assign the most significant features for a class in an image, i.e., the number of segments with more significant coefficients in the linear model. The larger the coefficient, the larger the impact on the class prediction. For a visual representation, we will plot only the essential segments regarding the Covid-19 or non-Covid-19 class, and mask the rest of the image.

**IV. EXPLAINABILITY OF CNN$_{explain}$ FOR COVID-19 DETECTION USING DRY COUGH**

This section presents the performance evaluation and analysis of XAI methods for Covid-19 detection using dry cough audio. It describes hardware/software, dataset, preprocessing, model, and hyperparameters.

**A. Experiment Setup**

We have used Linux OS, intel Xeon CPU processor at 2.00GHz with 2 CPU cores, 13GB of RAM, and 24 gigabytes of VRAM from a Tesla K80 graphic card. To implement the CNN model and XAI methods, we have considered libraries including Keras, a deep learning library built on top of TensorFlow, an open-source library for machine learning tasks; tf-Keras-vis, a deep learning visualization toolkit for Keras and LIME for interpretable model-agnostic explanations. Further, we have also considered Librosa [20] which is a package for music and audio analysis.

We have used a public dataset [21] that consists of 160 Covid-19 positive audios. The dataset includes 19 positive classes and 21 random negative classes. The reduction of negative classes at this stage is to address the issue of imbalanced classes, later used for training and predictions.

**B. Pre-processing/Augmentation**

The 40 original audios are augmented using time-stretching the audio from 30% to 190% of the original audio. It is further combined with other techniques including noise aggregation which adds random samples of values dispersed at regular intervals with a mean of 0 and a standard deviation of 1. Moreover, a time-shifting method that moves the sound to the beginning/end of the audio along the time axis with an interval of 15% of the sample rate of 44,100 Hz is also used. These augmentation techniques add 2720 new audios from 40 samples, resulting in a total of 2760 audios.

Further to augmentation, we have also converted the audios to a spectrogram using a sample rate of 44100 hertz and computing the magnitude spectrogram, and mapping it to a Mel scale. The conversion uses 1024 samples per fast Fourier transform window, 512 overlapping samples between successive frames to avoid information loss, and 128 samples of Mels. Mels represent bins of the spectrogram to its height. Finally, the data in the spectrogram is distributed by exchanging black and white colors in the image and flipping them from top to bottom of the spectrogram.
C. Model Parameters and Hyperparameters

The \textit{cnn\_explain} model consists of 3 convolutional layers and two fully connected layers. The input layer takes the spectrogram as $128 \times 820$ pixels along with three channels representing red, green, and blue color space. The convolutional layer is 2D and contains 16 units of filters. The output layer uses the max-pooling process for complexity reduction resulting in an image shape of $64 \times 410$ pixels. We have considered a dropout of 20\% for the output layer for regularizing the \textit{cnn\_explain} to prevent overfitting. The second convolutional layer consists of 32 filters. This layer also has a max-pooling process that reduces the image size to $23 \times 205$ pixels, and a dropout of 20\% of connections. Finally, the third convolutional layer consists of 64 filters followed by a max-pooling with $16 \times 102$ with a dropout of 20\%.

The \textit{cnn\_explain} model uses a total of 6,708,450 trainable parameters. We have conducted experiments with 20 training epochs per iteration with a batch size of 128, and have trained the model using the Adam optimizer. Adam optimizer is known for its best performance for CNN. The learning rate to train the \textit{cnn\_explain} is kept at 0.001. The activation function considered for all the layers is ReLu with the Softmax at the output layer. An early stopping mechanism is used to avoid overfitting and accelerate training, if accuracy does not increases in 5 epochs.

D. Performance Analysis

As it is important to keep a balance between the positive and negative classes to avoid bias during the model training, this section presents an exploratory analysis of data. Further, we have evaluated the performance of the \textit{cnn\_explain} in terms of sparse cross-entropy loss and accuracy as discussed below.

Figure 6 presents a visualization of the image’s shapes in the dataset, precisely the width of images that vary after the augmentation of the audios. The \textit{cnn\_explain} only accepts input images with the same shapes, therefore, this visual analysis provides insights for the best width for the resizing of the images. The localization of the group of images with the same width size is between 500 and 1000 pixels followed by the group between 0 and 500. Although, these groups are close enough for resizing, there are some outliers in categories of 3,000 pixels up to 7,000 pixels. Statistically, the mean of the group is 1,014 pixels and the median 718. Thus, all the images are resized with a width of 820 pixels and a height of 128 pixels.

Figure 7 and Figure 8 shows the distribution of Covid-19 and non-Covid-19 classes in the dataset. The expected distribution of the dataset is equal because the classes are collected in similar distributions during the data collection process. Figure 7 and Figure 8 shows the distribution of classes after a random split of data between train and test set to avoid an imbalanced dataset that can lead to inaccurate learning and predictions by the \textit{cnn\_explain}. Figure 7 and Figure 8 show a good balance of both the positive and negative classes, thereby assuring the efficiency of the learning and testing process.

To evaluate the accuracy of the model, we have changed the number of epochs from 1 to 18. Figure 9 shows that initially, both the training and validation loss is higher, however, it drops significantly after the first few epochs, indicating an optimal weights adjustment by the network for a correct prediction. Finally, it fully converges after 8 epochs. Figure 10 shows the accuracy of the model for the training and validation. Initially, \textit{cnn\_explain} demonstrates low accuracy for both the training and validation. However, after 8 epochs model is fully learned,
V. EXPLAINABILITY RESULTS FOR \textit{cnn}_{explain}

This section presents the explainability results of the \textit{cnn}_{explain} model to understand its learning and predictions and internals. Specifically, we analyze four methods discussed in Section III-A to explain the \textit{cnn}_{explain}.

A. Features Visualization

This section presents feature visualization in layers to explain the \textit{cnn}_{explain} internals. Specifically, it reveals that the last five filters of the three convolutional layers of the \textit{cnn}_{explain} as optimal to learn parameters. Further, there is a visualization of the last dense layer to interpret the most significant features contributing to classification. These features distinguish the characteristics of each class for classification with minimum error.

Figure 11 demonstrates the patterns learned by the first convolutional layer of the \textit{cnn}_{explain}. This layer contains 16 units. The features learned by this layer maximize the representation of images to aid the classification of input images. It can be noted that there is no pattern in the first convolutional layer that humans can identify more than a single color. However, these colors are a fundamental basis of the \textit{cnn}_{explain} to form more complex ways of understanding spectrograms in complex neural networks. Figure 12 shows the last five filters of the \textit{cnn}_{explain} in its second convolutional layer consisting of 32 units of them. Similar to the first convolutional layer, the patterns learned by the \textit{cnn}_{explain} are still simple not to differentiate any clear pattern in a spectrogram. However, in filter 29, the primary color learned by the filter is red, but there are spotted areas with a slightly different color. However, filter number 30 has activated the features like blue color, but a few areas of stronger blue are mainly located in the center and lower part of the channel. Thus, at this stage, the appearance of patterns beyond single colors can be observed.

Figure 13 presents the filters that the CNN learned in the third and last convolutional layer with 64 channels. At this stage, most \textit{cnn}_{explain} filters contain more meaningful information for class distinction. For example, filters 61, 62, and 64 show different colors and distinct shapes. The three images show some line patterns that stain the filter. The visualization shows that the filters in the \textit{cnn}_{explain}’s last layer convert from spots found in previous layers to lines. Figure 14 shows
the important input features that $cnn_{\text{explain}}$ can distinguish to classify each class in the last layer. These features can be seen as a combination of two colors in the spectrograms, i.e., green, and purple. The first noticeable difference between them is the stained part of the Covid-19 class, mainly located in the centre of the image. The localization of this feature is for the non-Covid-19 class with less presence in the surroundings of the spectrogram. The second difference is that the stained feature in the non-Covid-19 class appears mainly in purple, whereas the Covid-19 class has a presence of purple and green colors. The stained feature has a correlated shape with the patterns visualized in the previous layers.

B. Spectrogram Samples

Figure 15 shows four spectrogram samples that are later used for the remaining three explainable methods. For simplicity, four spectrograms are taken, with a balance of classes regarding two spectrograms representing audio of Covid-19 patients and two spectrograms representing audio of patients without Covid-19. The order of these spectrograms remains fixed throughout the following sections.

C. Smooth Gradient (SmoothGrad)

This section analyses the explainability of $cnn_{\text{explain}}$ using SmoothGrad. During the experiments, We have selected 50 smoothed images with a 50% noise spread level. In the visualization of results, the redder parts in the maps symbolize the significant presence of the class activated in the spectrogram, whereas the bluer ones show less. The first image represents the average gradients towards the Covid-19 class, and the second image shows the average gradients toward the non-Covid-19 class regardless of the spectrogram class. Figure 16 presents the saliency maps of the SmoothGrad algorithm for the four images as given in Section V-B towards the Covid-19 class. The SmoothGrad method seems to underperform the distinction of the classes due to its inherent property of averaging gradients. The most likely Covid-19 area by the model is the lower part of the spectrogram in the four spectrograms. However, not all four spectrograms are in the same class. Figure 17 is the second representation of the SmoothGrad algorithm toward the non-Covid-19 class. Again, the algorithm seems to highlight the same areas of the four spectrograms with minor differences. Also, the highlighted areas of this non-Covid-19 class remain the same as the Covid-19 class. Thus, it seems like an underperforming task by SmoothGrad.

D. Gradient-weighted Class Activation Mapping (Grad-CAM)

This section presents the results obtained with the Grad-CAM algorithm. Similarly, as seen in the last section, the explanation of the model by the GradCAM focuses Covid-19 and
non-Covid-19 classes. Figure 18 shows the four spectrograms with activations towards Covid-19, where the red colours represent the areas the Covid-19 relevant features. The first two Covid-19 spectrograms have a slightly less red presence than the last two in the area with more energy indicating non-Covid-19 class. The first two Covid-19 spectrograms contain a stained feature surrounding the area with more energy, a change in contrast to the last two spectrograms of non-Covid-19 that are coloured as blue. Figure 19 shows the four spectrograms with the activation of non-Covid-19 class where the red represent the areas with highlighted non-Covid features. The first two Covid-19 spectrograms contain the strongest blue areas in the image’s high-energy parts, representing the less important areas for non-Covid-19. The last two, non-Covid-19 spectrograms have blue areas as well. As can be observed, the higher activation of red is dominant than the blue, representing the correct classification of non-Covid-19 case.

E. Local Interpretable Model-Agnostic Explanations (LIME)

Similar to other explainability methods, the results with the LIME algorithm remain the same. Figure 20 represents the activation of the Covid-19 class and Figure 21 of non-Covid class for the four spectrograms. Figure 20 shows the four spectrogram’s output from the LIME algorithm for the Covid-19 class. This approach highlights stained segments seen in the first image before and after a high-energy sound, and in the second image at the beginning of the spectrogram. It can be noted that for the non-Covid spectrograms, the areas of Covid-19 relevant features appear to have higher frequencies with fewer frequency segments without clear signs distinction.

Figure 21 shows the results of the LIME algorithm in the four spectrograms towards the non-Covid-19 class. The algorithm highlights segments with relatively low stained areas. In case of non-Covid-19 spectrograms, the algorithm highlights segments that contain waveforms with more energy of sound as seen in the third image. The fourth image also shows areas at higher frequencies.

VI. CONCLUSION

Most of the deep learning approaches considered for medical decisions do not come as ‘out-of-the-box’, thereby limiting their applicability for domains such as medical or malware decisions, where wrong decisions can lead to harmful consequences. In the context of Covid19 detection, full
The CNN model called CNNchine learning models. Our work presents the explainability of by producing more explainable models. It helps humans to interpret, trust and understand the emerging generation of machine learning models. Our work presents the explainability of the CNN model called CNNexplain using four XAI techniques such as visualization, SmoothGrad, Grad-CAM, and LIME. These approaches explain the classification of the CNNexplain model for Covid-19 and non-Covid-19. The interpretability provided by these XAI methods contributes to more reliable decisions by providing insights into an input space’s relevance and analysis of intermediate relevance filters at CNNexplain’s hidden layers. The results by all the XAI methods have yielded encouraging results from our effort in the explanation of CNN, i.e., CNNexplain in terms of its processing, network representation, filter level explanation, and classification.

Initially, we have evaluated the performance of CNNexplain with a training accuracy of 98.8% after a few epochs for Covid-19 and non-Covid-19. This ensures that CNNexplain model fulfills the requirements of good accuracy for Covid-19 detection, thereby making the model a good candidate for the explanation.

The visualization of convolutional filters clearly shows the increasing complexity of patterns learned by the CNNexplain that allows the correct classification of the classes. Also, the dense layer shows the features that most likely represent a class by the CNNexplain. The results show that there is a stained feature that is clearly present in both classes that has a positive correlation with the shapes found in the filters. However, the presence of features is greater in the positive class than in the negative class. Further, the presence of that feature is more prominent for the purple color in the positive class, rather than the green in the negative.

The sensitive maps from the SmoothGrad and Grad-CAM algorithm highlight important features of a sample as does the segments retrieved from the LIME algorithm. SmoothGrad aims to eliminate noise from the sensitive maps, however, it underperforms in its application on spectrograms for the use case, due to its sensitive map showing little distinction for the interpretation of classes. Finally, the remaining two algorithms show some areas that are highlighted for the positive Covid-19 class with stained segments, mainly beside the sounds with high energy. The LIME algorithms seem to highlight areas with waveform in high-energy areas for the negative non-Covid-19 class. Unlike the Grad-CAM algorithm that seems to relate to higher frequency areas for the negative class.

Furthermore, results of the LIME show it as a suitable explanation model for neural network architectures such as convolutional neural networks, or for types of classification problems (e.g., Covid-19 and non-Covid-19). More generally, interpretability by these selected models can contribute to the design of accurate and efficient classifiers, not only by analyzing and leveraging the features’ relevance but also through the analysis of intermediate relevance features “at classifier’s filter’s within hidden layers”. The results from this research have yielded encouraging results from our efforts in the explanation of convolutional neural network processing with a filter-level explanation.

**ACKNOWLEDGMENTS**

This work is supported by the Atlantic Technological University (ATU), Donegal Campus, Ireland.

**REFERENCES**

[1] A. Das and P. Rad, Opportunities and challenges in explainable artificial intelligence (XAI): A survey. arXiv preprint. [Online]. arXiv:2006.11377
[2] A. M. Antoniadi, Y. Du, Y. Guendouz, L. Wei, C. Mazo, and B. A. Becker, “Current challenges and future opportunities for XAI in machine learning-based clinical decision support systems: a systematic review,” Applied Sciences, vol. 11, pp. 5088, 2021.
[3] J. J. Cui, S. Winter, D. Steiner, L. Wilcox, and M. Terry, “Shello AI”: Uncovering the Onboarding Needs of Medical Practitioners for Human-AI Collaborative Decision-Making,” in Proc. ACM on Human-computer Interaction Conf., 2019, pp. 1–24.
[4] Q. V. Liao, D. Gruen, and S. Miller, “Questioning the AI: informing design practices for explainable AI user experiences,” in Proc. CHI Conf. on Human Factors in Computing Systems, 2020, pp. 1–13.
[5] T. Z. Tan, G. S. Ng, and C. Quek, “Improving tractability of Clinical Decision Support system,” in IEEE Proc. Conf. on Neural Networks, 2008, pp. 1997–2002.
[6] J. Y. Jeon, W. S. Choi, N. H. Kim, and K. Y. Lee, “Evaluation of scalability and degree of fine-tuning of deep convolutional neural networks for COVID-19 screening on chest X-ray images using explainable deep-learning algorithm,” Journal of Personalized Medicine, vol. 4, no. 10, pp. 213, 2020.
[7] M. Montebello, M. La Ferla, and D. Seychell, An XAI Approach to Deep Learning Models in the Detection of Ductal Carcinoma in Situ arXiv preprint. [Online]. arXiv:2103.00272
[8] Y. Li, Y. and R. Qin, Towards explainable artificial intelligence (XAI) for early anticipation of traffic accidents. arXiv preprint. [Online]. arXiv:2108.00272
[9] J. H. Kim, and C.K. Lee, “A Study of the Classification of IT Jobs Using LSTM and LIME,” in IEEE Proc. Conf. on Smart Media and Applications, 2020, pp. 248–252.
[10] E. Nigri, N. Ziviani, F. Coppi, I. Zinetti, A. Antunes, and A. Veloso, “Explainable deep CNNs for MRI-based diagnosis of Alzheimer’s disease,” in IEEE Proc. Conf. on Neural Networks (IJCNN), 2020, pp. 1–8.
[11] J. Y. Jeon, W. S. Choi, N. H. Kim, and K. Y. Lee, “Explaining Deep Learning-Based Traffic Classification Using a Genetic Algorithm,” IEEE Access, vol. 9, pp. 4738–4751, 2020.
[12] Z. Ren, Q. Kong, J. Han, M.D. Plumbley, and B. W. Schuller, “CAA-Net: Conditional Attributed CNNs with Attention for Explainable Device-robust Acoustic Scene Classification,” IEEE Transactions on Multimedia., vol. 23, pp. 4131–4142, 2021.
[13] H. Alshazly, C. Linse, E. Barth, and T. Martinetz, “Explainable covid-19 detection using chest CT scans and deep learning,” IEEE Transactions on Multimedia, vol. 21, no. 2, pp. 455, 2021.

[14] G. Gong, L. Lin, Z. Wu, Z., R. Xu, S. Kido, and Y. Hu, “Evolutionary Neural Network and Visualization for CNN-based Pulmonary Textures Classification,” in Proc. IEEE 32nd International Conf. on Tools with Artificial Intelligence (ICTAI), 2020, pp. 555–563.

[15] Y. Cai, “Computing with Instinct: Rediscovering Artificial Intelligence,” in Springer, 2011, pp. 30.

[16] A. Glassner, “Deep Learning: A Visual Approach,” in No Starch Press, 2021, pp. 325.

[17] J. Brownlee, “Deep Learning for Computer Vision,” in Machine Learning Mastery, 2019, pp. 109.

[18] B. Boashash, “Time-frequency signal analysis and processing: a comprehensive reference,” in Academic press, 2015.

[19] V. Bhateja, S. Chandra, Y. D. Zhang, and M. Aradhya, “Intelligent Computing and Communication,” in Springer, 2019, pp. 740.

[20] Librosa.org, “Librosa,” in Available from: https://librosa.org/, [Accessed 22 June 2021].

[21] Kaggle.com, “COVID-19 Cough Recordings,” in Available from: https://www.kaggle.com/himanshu00712/cochcassifier-trial, [Accessed 28 June 2021].

Shagufta Henna is lecturer with the Atlantic Technological University, Donegal, Ireland. She was a post-doctoral researcher with the Waterford institute of technology, Waterford, Ireland from 2018 to 2019. She received her doctoral degree in Computer Science from the University of Leicester, UK in 2013. She is an Associate Editor for IEEE Access, EURASIP Journal on Wireless Communications and Networking, IEEE Future Directions, and Human-centric Computing and Information Sciences, Springer. She is senior member of IEEE. Her current research interests include edge computing, self-supervised learning, attention in deep learning, representation learning, deep reinforcement learning, explainable AI, and AI-driven network optimizations.

Juan Miguel Lopez Alcaraz is a Master student of Big Data Analytics and Artificial Intelligence. His research interests include machine learning, explainable AI, and big data analytics.