UNIVERSITY OF THESSALY
SCHOOL OF SCIENCE
Department of Computer Science and Biomedical Informatics

Rule extraction from data in biomedicine

Cholopoulou Eirini

THESIS
Supervisor
Iakovidis Dimitrios
Professor

Lamia, 2021

UNIVERSITY OF THESSALY
SCHOOL OF SCIENCE
Department of Computer Science and Biomedical Informatics

Rule extraction from data in biomedicine

Cholopoulou Eirini

Thesis

Supervisor
Iakovidis Dimitrios
Professor

Lamia, 2021

Με ατομική μου ευθύνη και γνωρίζοντας τις κυρώσεις [(1)], που προβλέπονται από της διατάξεις της παρ. 6 του άρθρου 22 του Ν. 1599/1986, δηλώνω ότι:

1. *Δεν παραθέτω κομμάτια βιβλίων ή άρθρων ή εργασιών άλλων αυτολεξεί* ***χωρίς να τα περικλείω σε εισαγωγικά*** *και χωρίς να αναφέρω το συγγραφέα, τη χρονολογία, τη σελίδα. Η αυτολεξεί παράθεση χωρίς εισαγωγικά χωρίς αναφορά στην πηγή, είναι λογοκλοπή. Πέραν της αυτολεξεί παράθεσης, λογοκλοπή θεωρείται και η παράφραση εδαφίων από έργα άλλων, συμπεριλαμβανομένων και έργων συμφοιτητών μου, καθώς και η παράθεση στοιχείων που άλλοι συνέλεξαν ή επεξεργάσθηκαν, χωρίς αναφορά στην πηγή. Αναφέρω πάντοτε με πληρότητα την πηγή κάτω από τον πίνακα ή σχέδιο, όπως στα παραθέματα.*

2. *Δέχομαι ότι η αυτολεξεί* ***παράθεση χωρίς εισαγωγικά****, ακόμα κι αν συνοδεύεται από αναφορά στην πηγή σε κάποιο άλλο σημείο του κειμένου ή στο τέλος του, είναι αντιγραφή. Η αναφορά στην πηγή στο τέλος π.χ. μιας παραγράφου ή μιας σελίδας, δεν δικαιολογεί συρραφή εδαφίων έργου άλλου συγγραφέα, έστω και παραφρασμένων, και παρουσίασή τους ως δική μου εργασία.*

3. *Δέχομαι ότι υπάρχει επίσης περιορισμός στο μέγεθος και στη συχνότητα των παραθεμάτων που μπορώ να εντάξω στην εργασία μου εντός εισαγωγικών. Κάθε μεγάλο παράθεμα (π.χ. σε πίνακα ή πλαίσιο, κλπ), προϋποθέτει ειδικές ρυθμίσεις, και όταν δημοσιεύεται προϋποθέτει την άδεια του συγγραφέα ή του εκδότη. Το ίδιο και οι πίνακες και τα σχέδια*

4. *Δέχομαι όλες τις συνέπειες σε περίπτωση λογοκλοπής ή αντιγραφής.*

Ημερομηνία: ……/…../20……

Ο – Η Δηλ.

(Υπογραφή)

(1) «Όποιος εν γνώσει του δηλώνει ψευδή γεγονότα ή αρνείται ή αποκρύπτει τα αληθινά με έγγραφη υπεύθυνη δήλωση του άρθρου 8 παρ. 4 Ν. 1599/1986 τιμωρείται με φυλάκιση τουλάχιστον τριών μηνών. Εάν ο υπαίτιος αυτών των πράξεων σκόπευε να προσπορίσει στον εαυτόν του ή σε άλλον περιουσιακό όφελος βλάπτοντας τρίτον ή σκόπευε να βλάψει άλλον, τιμωρείται με κάθειρξη μέχρι 10 ετών.

ΕΞΑΓΩΓΗ ΚΑΝΟΝΩΝ ΑΠΟ ΔΕΔΟΜΕΝΑ ΜΕ ΕΦΑΡΜΟΓΕΣ ΣΤΗ ΒΙΟΙΑΤΡΙΚΗ

Χωλοπούλου Ειρήνη

Τριμελής Επιτροπή:

Δημήτριος Ιακωβίδης, Καθηγητής (επιβλέπων)

Κωνσταντίνος Δελήμπασης, Αναπληρωτής Καθηγητής

Μιχαήλ Σαβελώνας, Επίκουρος Καθηγητής

ABSTRACT

As Deep learning models stay true to their ubiquitous reputation, an increasing demand for explanations behind their reasoning aims to refute some of their obscurity, especially in high-risk domains. Existing methods predominately fulfill this demand estimating a-posteriori explanations, whereas self-explaining models remain in uncharted territory. In this thesis we present an inherently interpretable end-to-end class of CNN models, referred to as ConGAM on the basis of generalized additive models. Our model can be described as an ensemble of jointly trained CNNs, each learning a different image representation which ultimately provides its contribution to the final prediction. We refer to these representations as descriptors because they approximate how we humans intuitively describe objects in images. We demonstrate that our method mainly outperforms other benchmark CNN architectures, while providing illustrative insights behind its decisions on both local and global scale, based on its evaluation on a semantically meaningful dataset we created and two provided endoscopic datasets.

TABLE OF CONTENTS

1 Introduction

Real world applications introduce increasingly complex tasks enabled by the adoption of machine learning systems. Despite their expanding deployment and discriminative power their opaqueness stands beside them as a seemingly unyielding impediment. Surely, dilemmas arise where this opaqueness hides however decisions still have to be made, so the need for explanation is critical. As expressed by the "right to explanation" act (Selbst, et al. 2017) that came into effect by EU's General Data Protection Regulation (GDPR), *'*meaningful information about the logic involved*'* should be provided in the context of automated decision making. Therefore, ensuring interpretability is crucial to establish user's trust since the reasoning behind the model's decision would not be entirely concealed within its black box. In risk sensitive domains the provided insights could contribute to identify bias, provide justifications, improve decision making and ultimately enhance our understanding toward better interpretations. For instance, in medical diagnosis the inability to provide sufficient and consistent explanations for a certain prediction could prevent the implementation of such a model. But how do we convey our need for explanations on these systems? A variety of methods aim to accommodate the emergence of interpretation, and our work is in alignment with that demand.

In the context of intrinsically interpretable models and relevant to our method is the family of Additive models first introduced by (Friedman, et al. 1984), with the name of *'projection pursuit regression'* as a method that modeled a regression surface as a sum of general smooth functions of linear combinations of the predictor variables in an iterative manner. Several methods extended this work, with one of them utilizing neural networks to learn interpretable features resulting in a model with built-in interpretation mechanisms, proposed by (Vaughan, et al. 2018) as Explainable Neural Networks (xNN) and its special case of Adaptive explainable neural networks (AxNNs) by (Chen, et al. 2020). Closer to our work and a further reformulation of AIMs was proposed by (Hastie, et al 1990) introducing the class of generalized additive models by replacing the linear predictor with an additive predictor of smooth functions. Further extensions of GAMs were thereafter introduced including GA^{2}Ms by (Lou, et al 2013) and GAMI-Net by (Zebin Yang, et al. 2020) which can handle pairwise interaction terms, GAMMLI proposed by (Guo, et al 2020) that models user-item interactions for recommendation systems (RecSys) and NAMs proposed by (Agarwal, et al 2020) that parametrize the f_i functions with neural networks, each assigned an input feature to learn non-linear patterns and feature jumping.

1.1 Aim and Contributions

In this thesis we approach the demand for explanations in the context of image classification by proposing ConGAM, an intrinsically interpretable end-to-end framework that shapes its reasoning to approximate how humans understand the different facets of an image in order to make its predictions. ConGAM offers a level of transparency, in a sense that it can indicate how each aspect of information contributes to its decisions on both local and global scale, i.e., offers the feature importance scores of a single image example and of the entire dataset, respectively. This work's contributions entail:

- A novel intrinsically interpretable class of CNN models, referred to as ConGAM, proposed as an extension of the GAM family of models into the computer vision field of applications.
- Two subnetwork CNN architectures utilized within the proposed framework.
- A benchmark dataset for binary image classification used for the subsequent evaluation of ConGAM.
- The comparison of our framework with well-established CNN architectures and the indication of its advantageous performance over them.
- Interpretable representations that bring the reasons behind our model's predictions into illustrative terms, in regard to feature importance.

In the context of this work some of the results are submitted to an international scientific conference in collaboration with Prof.Iakovidis and Dimas.G, as: Dimas, G.,Cholopoulou, E., Iakovidis, D., K.,.(2021). E Pluribus Unum Net: An Interpretable CNN Model. In *Proceedings of the IEEE/CVF International Conference on Computer Vision.(submitted)*.

1.2 Thesis Structure

The rest of the thesis is organized in 4 sections. Section 2 covers the basic concepts of the domains of image analysis and artificial neural networks. Section 3 provides an overview of the existing methods in the context of interpretability by grouping them based on their common characteristics whilst presenting some of their limitations. Section 4 is dedicated to the basic components that comprise the proposed framework, including its structure and interpretable mechanisms. Section 5 presents the quantitative results of our method and the interpretable representations obtained by evaluating our approach on 3 different datasets.

2 Image Analysis and Artificial Neural Networks

In this section we provide a brief overview of some of the basic components that characterize the domains of image analysis and deep learning in the context of artificial neural networks, with the aim of bringing a sufficient level of clarity on the background that establishes the context of this thesis.

2.1 Digital Image Processing

In order to investigate some of the methods encompassed in the field of image analysis we first have to define how images are represented in digital form. Suppose we sample a continuous image of the form $f(s,t)$ with s,t continuous variables, we can obtain the corresponding digital image by implementing the operations of sampling and quantization to reduce it in a discrete and finite form in order to enable its digital processing. We therefore denote images by two-dimensional arrays $f(x,y)$, where (x,y) are the discrete spatial coordinates and $f(.)$ captures the intensity value at any coordinate (x,y) as determined by its bit depth. We express a digital image's numerical representation in the following form:

$$f(x,y) = \begin{bmatrix} f_{(0,0)} & f_{(0,1)} & \cdots & f_{(0,N-1)} \\ f_{(1,0)} & f_{(1,1)} & \cdots & f_{(1,N-1)} \\ \vdots & \vdots & & \vdots \\ f_{(M,0)} & f_{(M,1)} & \cdots & f_{(M,N-1)} \end{bmatrix} \qquad (1)$$

consisting of N rows and M columns to determine the height and width of the image respectively. We could also represent the image as a vector of size $MNx1$ or $1xMN$ as long as we are consistent with the placement of the elements. Each image element $f_{(x,y)}$ is also referred to as pixel, which carries an intensity value and in combination with its neighboring pixels it creates the illusion of a continuous tone image. For example if all pixels are of equal value the image will be appearing with a uniform shade.

2.1.1 Image Filtering

Various techniques exploit the fact that an image is a signal and therefore apply signal processing operations on it. Generally, image processing methods can encompass low-level to high-level procedures, from geometrical transformations and image enhancement to detection and extraction of meaningful features. Therefore, in order to capture the aspects of the image that carry perceptually significant properties we need to address the types of computations they employ. Typically, image filtering is an indispensable part in performing such tasks.

A filter also referred as kernel or mask is the principal component of image filtering methods that encodes different numerical patterns in a matrix form. Its size is indicative of the degree of filtering to be imposed, as larger filters will have a stronger effect but greater loss of image detail information.

The convolution operation provides the basis for image filtering since the obtained filtered image is the result of convolving a selected filter over the original image. Specifically, convolution is performed for each pixel in the image f, where the corresponding value of the filtered image g is obtained by placing the center of the kernel h on the current pixel at (x, y) and summing all the products between the elements of the kernel and their overlapping pixel values. The two-dimensional expression for the convolution between an NxM image denoted as f and the filter h is as follows:

$$g(x,y) = f(x,y) * h(x,y)$$

$$= \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} h(k,l) f(x-k, y-l) \qquad (2)$$

Notably, this is a spatially invariant process as the image is convolved with a constant kernel. Therefore, the kernel can be perceived as the impulse response of a linear space invariant system (LSI) as presented in ***Fig.1*** with input $f(x,y)$ and output $g(x,y)$, which also satisfies the linearity relationship:

$$\alpha f_1(x,y) + \alpha f_2(x,y) \Rightarrow \alpha g_1(x,y) + \alpha g_2(x,y) \qquad (3)$$

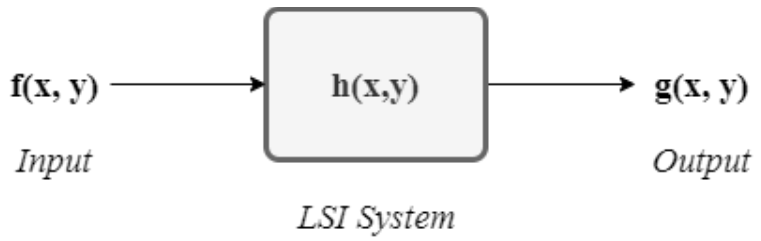


Figure 1: *Depiction of an LSI system with $h(x,y)$ as the impulse response, $f(x,y)$ and $g(x,y)$ the input and output images respectively.*

2.1.1.1 Smoothing Filters:

With the aim of imposing a blurring effect on a given image or reducing the noise it encodes, we apply smoothing filters that can be either linear or non-linear, to exploit their relative capabilities.

Mean Filter:

The linear mean filter is implemented with the use of a spatial averaging operation, where the value of each pixel is replaced by the mean of the pixels that surround it, resulting in the average smoothing of the image. Assuming we want to utilize the mean filter on the f image in the region defined by S_{xy} , the computed value of the filtered image g at a the xy coordinate would be:

$$g(x,y) = \frac{1}{mn} \sum_{(i,j)\ \epsilon\ S_{xy}} f(i,j) \qquad (4)$$

where mn is the spatial filter size and $\frac{1}{mn}$ indicates the values of its coefficients.

The application of this filter on a given image smooths out the local variations within it while maintaining its overall outline therefore it is considered effective in reducing the encoded noise. Notably, the utilized kernel must have non-negative values that sum up to one, ensuring that the brightness of the filtered image is preserved.

Median Filter:

The median filter is a spatially non-linear filter whose response is determined by the ranking of values encompassed in the neighborhood of the pixel its applied on (enclosing the center pixel value as well). The computation therefore entails sorting all the values of a given pixel's neighborhood, determining the median and placing it in the corresponding (x, y) coordinates in the filtered image.

$$g(x,y) = median\{f(i,j)\} \qquad (5)$$

Median filters are proven to be more effective than linear smoothing filters of similar size against certain types of random noise, such as salt and pepper and impulse noise, because they remain unaffected from extreme values within the pixel's neighborhood.

Gaussian Filter:

The Gaussian filter is a low-pass filter which reduces the image's high frequency components, yielding an overall "blurred" result. To achieve this the original image is convolved with a Gaussian mask comprised of elements computed by a Gaussian function (***Fig.2***).

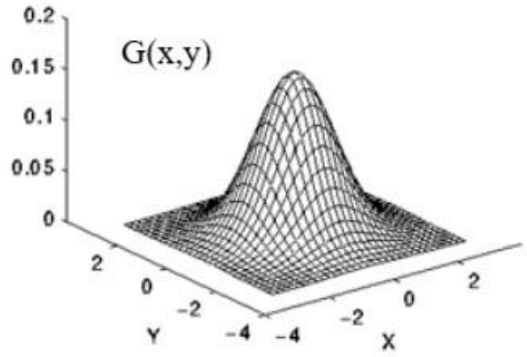


Figure 2: *Illustration of Gaussian function in 3 dimensions.*

At convolution level, the values are replaced with a weighted average of the neighboring pixels, with less weight assigned as the distance from the center pixel increases. Hence, preserving the structure components of the image to an extent. The blur effect is generated by convolving the image with the Gaussian kernel in two dimensions as follows:

$$G(x,y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{x^2+y^2}{2\sigma^2}} \qquad (6)$$

where σ is the standard deviation of the Gaussian distribution. The larger the σ value, the stronger the blur effect. In terms of computation, the two-dimensional kernel operation can be reduced to two separate one-dimensional kernel convolutions, in the vertical and horizontal directions, deriving the same outcome *w*ith fewer calculations.

2.1.1.2 Edge Detection Filters:

In various image processing tasks the identification of semantically meaningful boundaries of objects in a scene can be a powerful tool. The process of edge detection handles that task by identifying sharp discontinuities in intensity of an image which often characterize the physical extent of objects. Through this process the structural components of the image are preserved while a significant amount of redundant information is reduced. Some of the different gradient-based operators built to detect these edges will be briefly mentioned below.

Sobel Edge Detector:

The Sobel detector is a gradient based method which detects edges where the gradient magnitude is high (Sobel, I. 1972). The gradient magnitude is defined as the combination of the gradient approximations over the horizontal and vertical direction and depicts the "strength" of the edge. Notably, these directional operations are relatively computationally inexpensive. The Sobel edge detector entails convolving the input image with two $3x3$ directional kernels, horizontal Gy and vertical Gx, and combining their effect to create a series of gradient magnitudes. Assuming **A** denotes the input image matrix and Gy, Gx the first derivatives, the formula for Sobel edge detection operation is determined as:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * A \qquad (7)$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * A \qquad (8)$$

Note that the weights next to the central pixel are doubled, imposing a higher significance to the center.

The contributions of the horizontal and vertical components are then combined to form the absolute gradient magnitude as computed by:

$$|G| = \sqrt{{G_x}^2 + {G_y}^2} \qquad (9)$$

Or alternatively to be more computationally efficient, an approximate magnitude is computed as:

$$|G| = |G_x| + |G_y| \qquad (10)$$

The gradient's direction can also be calculated as:

$$\theta = arctan(G_y/G_x) \qquad (11)$$

where zero orientation *($\theta = 0$)* indicates a vertical edge with the transition from black to white being from left to right respectively.

Prewitt Edge Detector:

The Prewitt operator is similar to the Sober operator in that it utilizes two 3x3 convolutional kernels, P_x and P_y, designed to respond maximally to edges with vertical and horizontal orientation respectively. The difference is the removed emphasis given to the center pixel in the applied convolutional masks, as shown below:

$$P_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad P_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \qquad (12)$$

Roberts Cross Operator:

Similarly, another differential operator which consists of two convolutional kernels of size 2x2, designed to be sensitive to changes in intensity in a diagonal direction, as shown:

$$R_x = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \qquad R_y = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \qquad (13)$$

The operator is applied to the input image by separately convolving the two kernels over it and then combining their results to form the absolute gradient magnitude at each (x, y) point as defined in *(10)*. It is a rather simple and fast operator but its sensitivity to noise can be restrictive for its application.

2.1.2 Color Models

Our perception of a visual scene and the objects within it is strongly intertwined with the information of color encoded into it. How light is reflected by the objects gives form to our perception regarding colors within the visible part of the electromagnetic spectrum. The frequencies it encloses in its narrow range are captured by three different types of biological sensors, referred to as cones. These types of cones can absorb light differently, by responding to the red, green and blue part of the spectrum respectively as depicted in ***Fig.3***.

To standardize this notion into mathematical terms, in 1931 CIE (Commission internationale de l'éclairage) set the fundamentals of color spaces based on the concept of additive primaries. Specifically, a system of three additive primary colors (red, green and blue) with mathematical properties that enabled the transformations to other color spaces. We define color spaces as different mathematical representations of colors in a standardized form, each mapping individual colors as points within the subspace their coordinate system formulates.

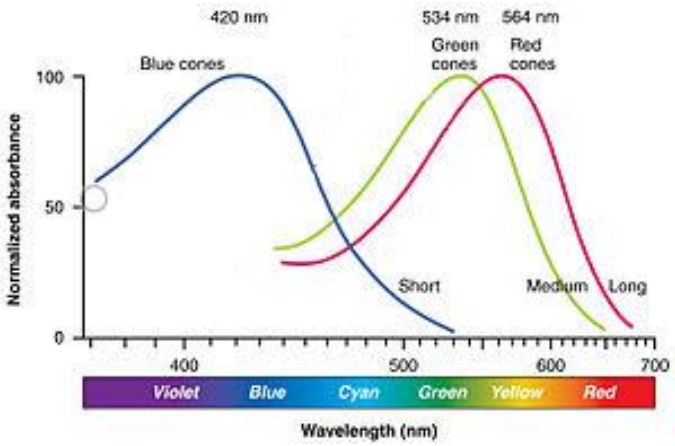


Figure 3: *Sensitivity to light of the red, green and blue cones with respect to the spectral distribution.*

There are several color models utilized today, such as RGB, HSV, CMYK, CIE-L*a*b etc., each characterized by their own attributes, limitations and purposes. We will selectively mention some of them in the following segments.

RGB Color Space:

This is a widely used additive model defined by its primaries, red, green and blue and it's represented in the cartesian coordinate system as a cube, as shown in ***Fig.4***. As we can observe, the axes are labelled as R, G, B to depict the corresponding primaries which are normalized within the range of [0, 1]. The origin *(0,0,0)* is considered black, whereas the furthest corner from it *(1, 1, 1)* is set as white. Notably, the line connecting black and white encodes the grey-scale range where each R, G, B component is of equal value. The color gamut the RGB color model encodes is described by each point configuring this cube.

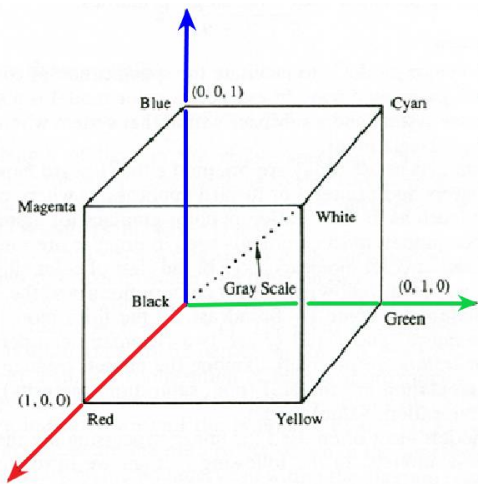


Figure 4: *The RGB color space. Origin indicates black, the line connecting it with white (1, 1, 1) encodes grey-scale and the 3 axis are indicative of the primaries(red, green and blue).*

However, the RGB color space is restrictive in terms of its hardware and brightness dependence and perceptual non-uniformity. Specifically, the displayed results of the RGB model are susceptible to camera modifications, they also encode brightness in every channel which therefore affects chrominance (color-related information) by illumination changes. Moreover, RGB color space is perceptually irrelevant as the changes in color are often characterized by discontinuities occurring because of its too literal approach of recreating the behavior of light.

HSV Color Space:

Attempting to describe colors in a more pragmatic manner, researchers designed the HSV color model to approximate how humans perceive and interpret color. It is an alternate representation of the RGB color model that decouples the intensity component from the color-related information and assumes a hexcone form, as shown in ***Fig.5***. Its components are hue *(H)*, saturation *(S)* and value *(V)*. Hue is the angular dimension that encodes pure color with values in the [0, 360] range. Saturation measures the degree to which a color is diluted by light, indicating its purity. It is measured as the distance from the vertical intensity axis and the range of values it displays are within the 0 to 1 range, indicating no saturation (grayscale) and the purest form of the color respectively. Value is depicted as the vertical axis and it represents the intensity of color from 0 (black) to 1 (white) as the amount of light illuminating a color.

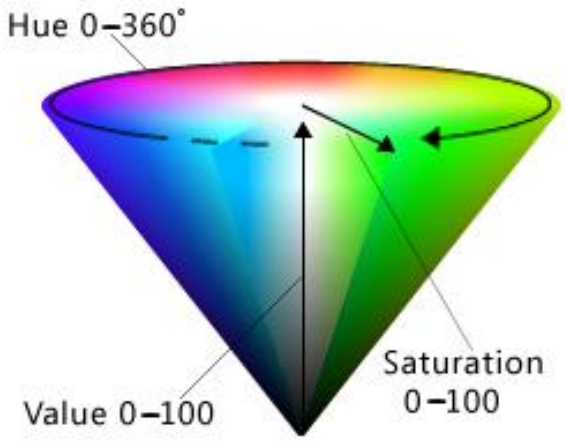


Figure 5: *Cone representation of the HSV color space.*

The HSV color space can take its form in a 3-dimensional coordinate system by mapping each point of the RGB color space to a corresponding point in the HSV model. Given an image in the RGB color format and considering the R, G, B channels to be normalized to the [0, 1] range, the hue *(H)*, saturation *(S)* and value *(V)* components are obtained as follows:

$$V = \max(R, G, B) \tag{14}$$

$$S = \begin{cases} \dfrac{V - \min(R, G, B)}{V} & if\ V \neq 0 \\ 0 & if\ V = 0 \end{cases} \tag{15}$$

$$H = 60 + \begin{cases} \dfrac{G - B}{V - \min(R, G, B)} mod\ 6 & if\ max(R, G, B) = R \\ \dfrac{B - R}{V - \min(R, G, B)} + 2 & if\ max(R, G, B) = G \\ \dfrac{R - G}{V - \min(R, G, B)} + 4 & if\ max(R, G, B) = B \end{cases} \tag{16}$$

If $H < 0$ then $H = H + 360$

The resulting output ranges of the hue, saturation and value components are $0 \leq H \leq 360$, $0 \leq S \leq 1\ and\ 0 \leq V \leq 1$ respectively.

However, the HSV color model can appear problematic in terms the interdependences that characterize it. Specifically between its saturation and value channels, a saturation scale may contain different brightness values. Similarly we note the correlation between its hue and value channels, where fully saturated colors can be described by the same value of lightness, which differs from how we perceive them. These attributes may prevent the utilization of the HSV color model in applications where modifying one dimension should not distort the information in the rest.

*CIE-L*a*b Color Space*:

With the aim of designing a color model that could approximate how we perceive colors, the CIE-Lab Color space took form. It is comprised of 3 channels representing the perceived lightness *L** and the two color-opponent chroma-related channels, *a** and *b**. The perceived lightness dimension *L** aims to capture the non-linear human eye's response to light, carrying values within the range of 0 (black) to 100 (white). The *a** axis represents the color range of green to red, translated into the value range of [-127, 127] and the *b** axis is indicative of the blue to yellow color gamut, translated into a similar value range of [-127, 127] respectively, as shown in ***Fig.6.***

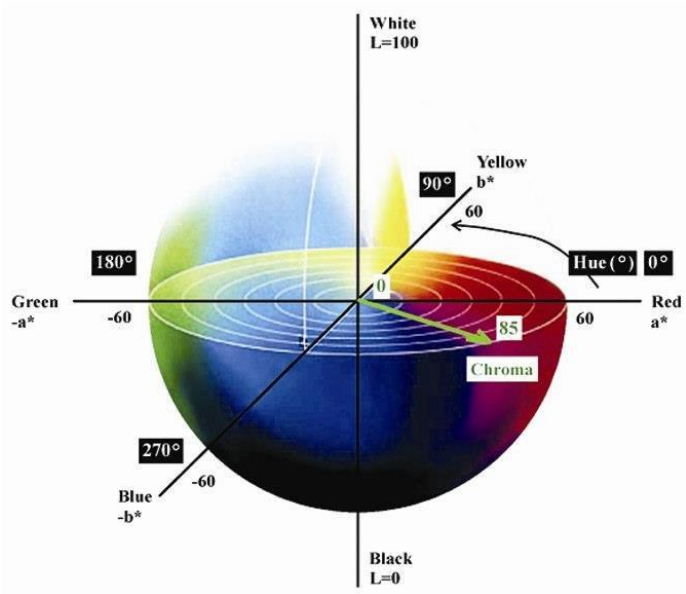


Figure 6: *CIELAB color space 3-dimensional representation.*

The RGB to CIELAB conversion is possible by first transforming the RGB space to the XYZ space, indicating how CIELAB ultimately encompasses a wider range of colors, and then mapping its points to the corresponding CIELAB model color points. Given an image in in the RGB color format and considering the R, G, B channels to be normalized to the [0, 1] range, the steps to obtain the *L**, *a** and *b** components, using *D65* as the white point of reference, are obtained as follows:

The first step of mapping RGB to XYZ is computed as:

$$X = 0.4124R + 0.3575G + 0.1804B \tag{17}$$

$$Y = 0.2126R + 0.7151G + 0.0721B \tag{18}$$

$$Z = 0.0193R + 0.1191G + 0.9502B \tag{19}$$

$$X = \frac{X}{X\mathrm{n}} \tag{20}$$

$$Z = \frac{Z}{Z\mathrm{n}} \tag{21}$$

where $X\mathrm{n} = 0.9504,\ Z\mathrm{n} = 1.0887$

Followed by the L^*, a^* and b^* channel mapping:

$$a^* = 500\big(\mathrm{f(X)} - \mathrm{f(Y)}\big) + \mathrm{delta} \tag{22}$$

$$b^* = 200\big(\mathrm{f(Y)} - \mathrm{f(Z)}\big) + \mathrm{delta} \tag{23}$$

$$L^* = \begin{cases} 116 * Y^{\frac{1}{3}} - 16, & for\ Y > 0.008856 \\ 903.3 * Y, & for\ Y \leq 0.008856 \end{cases} \tag{24}$$

where $delta = 128$ for 8-bit images or $delta = 0$ for floating point images.

Resulting in $0 \leq L^* \leq 100$ from black to white, $-127 \leq a^* \leq 127$ from green to red and $-127 \leq b^* \leq 127$ from blue to yellow along the axes accordingly.

Importantly, CIE L*a*b* (CIELAB) aspires to perceptual uniformity, where a change in a color value produces an analogous visual change (Bansal, et al 2011) and enables the detection of small shifts in chrominance. Despite of L^*'s approximation of the human eye response to light, part of CIE-LAB's connection to human vision is attributed to complimentary color coding. In L*a*b this is depicted as opponent colors along a and b axes where the calculated distance between colors is directly proportional to the difference perceived by the human eye (P. Ganesan, et al 2010). These attributes make the CIELAB color space suitable for applications where perceptual correctness is significant.

2.2 Computer Vision and Deep Learning

The field of computer vision seeks to approximate the tasks human vision performs, by deploying artificial systems to interpret and extract information from images toward reaching that goal to the utmost extend. Although from a high-level standpoint machine vision systems are comprised of a two-step process, feature extraction and decision making, in practice there is a need to unify this process in an automated and effective way.

Image classification is one of the problems computer vision tackles with, usually by constructing discriminative feature extractors to enable algorithms to automatically classify objects in images. Therefore it becomes evident that computer vision approaches share techniques with image analysis and processing, for example, a feature extractor may impose a series of mathematical operations over the input image to isolate its edges.

However, to meet the demand of automation and effectiveness in computer vision tasks as previously mentioned, deep learning approaches are employed. These techniques receive a lot of attention as they reportedly achieve state-of-the-art performance in various tasks, including image classification. Notably, the recent surge of interest and following advancements of deep learning methods can be attributed to two prominent factors, computational power and data availability. Specifically, parallel GPU computing allowed a significant boost in accelerating the training process of deep models, in comparison with the standard CPU-based training. The appearance of publicly available, high-quality labelled datasets like ImageNet (Deng, et al. 2009) further enabled the implementation of more complex models. These two critical developments also aided the resurgence of the neural network models we employ in our methodology as we will further describe in the following sections.

2.2.1 Artificial Neural Networks

Artificial Neural Networks are types of multi-layered, fully-connected networks of neurons that form relationships between variables to extract knowledge from representative sets of data. ANNs are considered non-linear feed-forward mapping structures of a given input to the response variable, that can also learn and generalize well from noisy data. Although, the concealment of their innate complex operations places them in the domain of black-box approaches.

ANNs are typically comprised of an input layer, multiple hidden layers and an output layer, where every node in each layer is connected to every node of the previous layer, as shown in ***Fig.7.*** As we introduce our input data to the neurons of the first layer, the information they convey passes through the network by activation functions that either remain dormant or 'fire' to pass the output signal to the next layers. As these signals pass through the network, they are either inhibited or amplified by weight allocating functions. High-weighted connections indicate more influential nodes and guide more effectively the learning process of the network, to ultimately generate the final output.

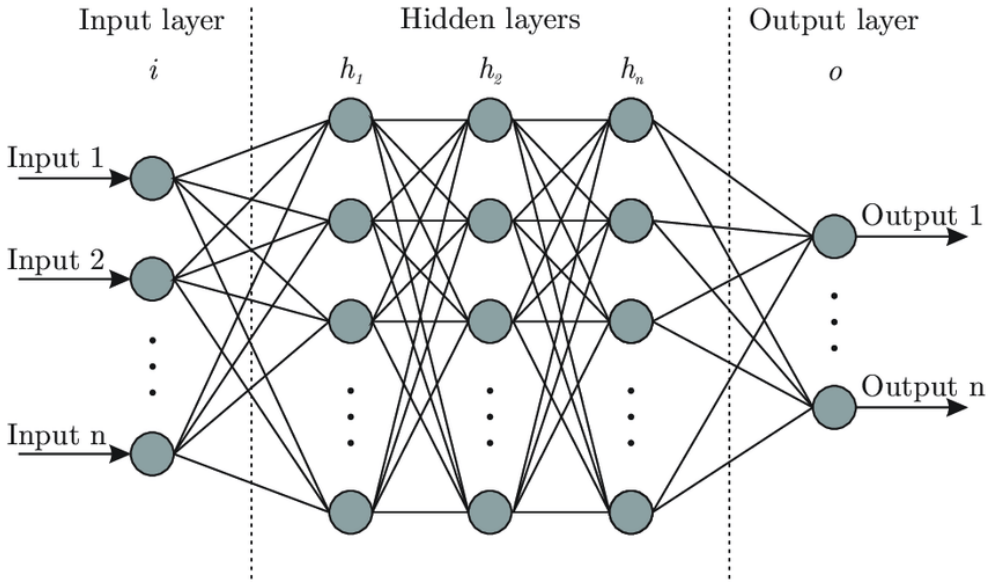


Figure 7: *Artificial Neural Network depiction of 3 distinctive parts: input layer, multiple hidden layers and the output layer.*

Even though ANNs can extract useful information more effectively than traditional machine learning techniques, they require large amounts of data to provide accurate results. Especially for the task of image classification the utilization of ANNs can be quite impractical, as the number of parameters would be computationally prohibitive considering how the entire image would translate into input nodes, only to further increase the number of weights as the network expands. Another limiting aspect in terms of applying ANNs is that the 2D spatial structure of the images would be reduced, due to vectorized images.

2.2.2 Convolutional Neural Networks

In an attempt to combat these limitations, a special type of deep learning model would have to be employed. Convolutional neural networks fulfill this task as they detect features through optimizable extractor functions, 2D-kernels, operating in alignment with the structure of the image. Specifically, the learned filters exploit spatial local correlations for a given input by utilizing the convolution operation repetitively across the image, which results in fewer parameters compared to the standard feedforward neural networks as the weights between the same filters are shared.

Notably, CNNs have demonstrated their effectiveness over traditional machine learning approaches as reported in (Bengio, Y. 2009). From the starting point of their inspiration based on the visual system's structure (Hubel & Weisel, 1962), to their later formulation by (LeCun, et al. 1998) employing the error gradient and yielding state-of-the-art performance results, only to be revolutionized by the AlexNet model (Krizhevsky, et al 2012) which surpassed all other methods by at least 10% in terms of error rate, in the 2012 ImageNet challenge for the task of object recognition in images.

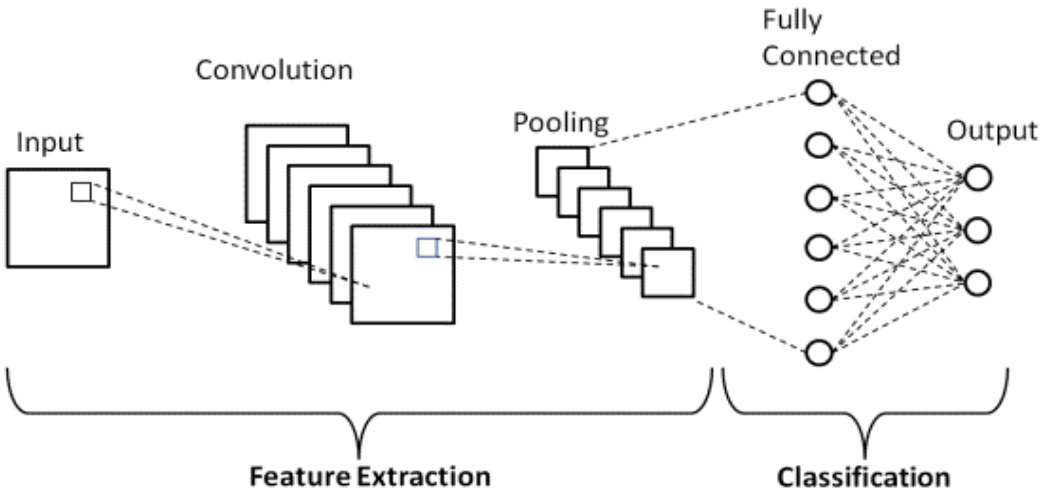


Figure 8: *CNN architecture comprised of the basic building blocks: Convolution layers, Pooling layers and Fully Connected layers.*

Below we present the main components of a Convolutional neural network followed by a brief description of its learning process. A CNN architecture mainly consists of 3 types of building blocks: the convolutional layer, the pooling layer and the fully connected layer. Typically, the first blocks of a CNN are formed by stacking convolutional and pooling layers together which can gradually detect more complex features by mapping each input volume to an output volume under certain rules, leading to one or more fully connected layers which handle the essence of all previously extracted information before reaching a final prediction as shown in ***Fig.8.***

Each type of layer assumes a different role, as we will briefly mention below.

Convolutional layer: The basic building block of a CNN architecture is the convolutional layer. In this context, the convolution operation is applied utilizing various kernels over the input, to produce feature maps. Specifically this includes the multiplication of the input array with a 2-dimentional array of weights, also referred to as kernel or convolutional filter, to obtain a single value that will be placed in the produced feature map as shown in ***Fig.9***. Typically, these filters are smaller than the input, which allows their repeated application over the whole dimension of input width. The information passed to the next layer encloses all the feature maps that were obtained in this process. This allows the network to ultimately learn filters that can detect features across the entire input volume, introducing the concept of translation invariance where the extracted information is regarding the feature's presence rather than its specific location in the image.

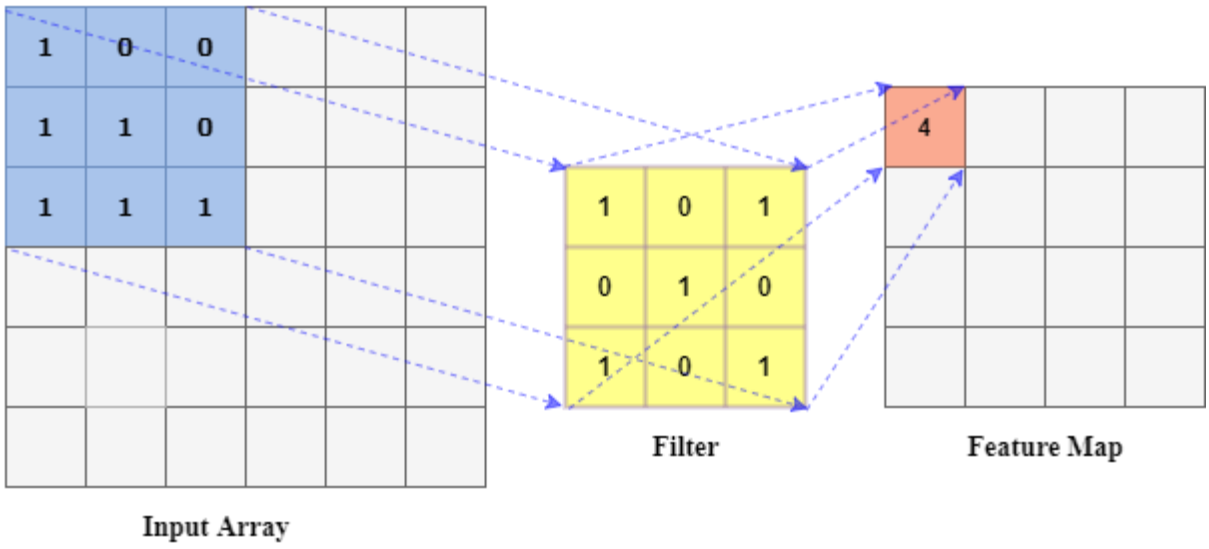


Figure 9: *Example of applying a 3x3 convolutional filter to the input data and the corresponding feature map.*

The convolutional layer is characterized by certain hyper-parameters, namely the kernel-size, stride and depth. Kernel-size refers to the specific height and depth of the 2D convolutional window, commonly used kernel-sizes are *3x3* or *5x5*. The stride parameter specifies the step along the height and width directions when the filter is re-applied, usually takes the value of 1 or 2 which results in overlapping application of filters. Depth indicates the number of output filters of a convolution layer.

Pooling Layer: Another necessary part of the CNN architecture is the pooling layer, also referred to as downsampling as its application ultimately reduces the spatial size of the given input volume. The resulting dimensionality reduction leads to the subsequent decrease of parameters the network has to learn, hence lessening the computational overhead and preventing overfitting. Notably, the pooling layer also imposes invariance to small translations of the input, as the mapping of the features will remain in the same position after downsampling even if their location slightly changes in the input volume.(Zhou, et al. 1988)

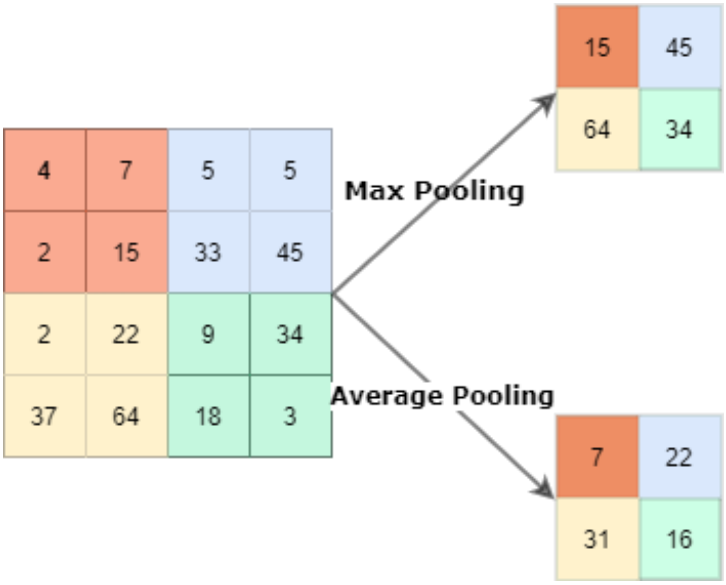


Figure 10: *Pooling operations example of Max pooling and Average pooling cases. Color is indicative of the corresponding responses of a 2x2 stride pooling window.*

The pooling layer is defined by the selected filter size and stride and its applied to each individual feature map separately, leaving the depth dimension unaffected. Typically, the selected filter size of the pooling operation is *2x2* with a stride of 2 which yields a 75% reduction of the input volume. Max-pooling and average-pooling are the most utilized methods ***(Fig.10)***, among other variations such as spectral pooling, stochastic pooling etc. Average-pooling extracts the average value for a given patch in a feature map, while max-pooling keeps the maximum value instead which leads to better performance results as reported in (Scherer, et al. 2010). This is reasonable as the higher responses extracted by max-pooling are also indicative of feature presence, carrying more useful information.

Fully Connected Layer: This type of layer primarily appears at the end of a CNN architecture because it can convey the overall picture of what the previous layers have captured. Each neuron that comprises a fully connected layer takes as input every neuron of the previous layer, hence the name Fully Connected. To formulate such a layer after the several convolutional and pooling layers, a flattening operation must be computed to convert the 2D feature maps to a 1D feature vector. The derived feature vector can then be passed to other fully connected layers, eventually to reach the final neurons of the network which indicate the categories for classification.

Ultimately, the model's goal is to bridge the gap between what it thinks it sees and what is truly depicted. In other words, optimally regulate the weight and bias of neurons to reduce the error of the output in the utmost extent. Among the different loss metrics that quantify the error of the model for the given task of binary image classification, we present the Binary Cross Entropy error which is defined as:

$$E(y,p) = -y\log(p) - (1-p)\log(1-p) \qquad (25)$$

where y is the true label and p the estimated probability of belonging to the positive class.

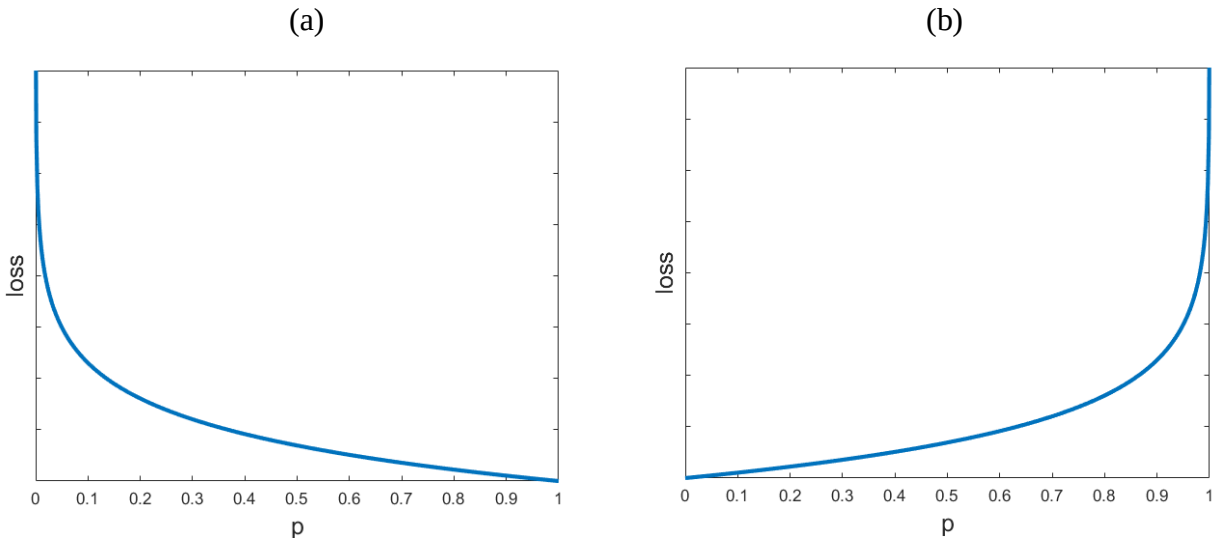


Figure 11: Binary Cross Entropy loss function's both cases depicted as (a) and (b). (a) Case of y=0, Loss $\to \infty$ if p $= 1$ and Loss $= 0$ if p $= 0$;(b) Case of y=1, Loss $\to \infty$ if p $= 0$ and Loss $= 0$ if p $= 1$.

As depicted in ***Fig.11***, the model pays the highest penalty ($Loss \to \infty$) when the estimated probability p and the true label y are in opposition, otherwise it equals zero ($Loss = 0$).

Gradient Descent:

The learning process of the model entails minimizing the above cost function by modifying the weights and biases of the model. In order to clarify this process, we make a brief representation of the underlying mechanism that guides it, Gradient descent (Robbins, H & Monro,S 1951).

For simplification purposes, we use a neural network of L number of layers, as illustrated in ***Fig.12***. We center our attention on the output of the last layer $\boldsymbol{x^L}$, defined as the weighted sum of the previous layer's neuron activations plus a bias, fed into an activation function, as follows:

$$x^L = \sigma(w_0^L x_0^{L-1} + w_1^L x_1^{L-1} + \cdots + w_5^L x_5^{L-1} + b^L) \qquad (26)$$

where σ is the activation function, $w_i$is the weight of the i^{th}neuron and b is the bias.

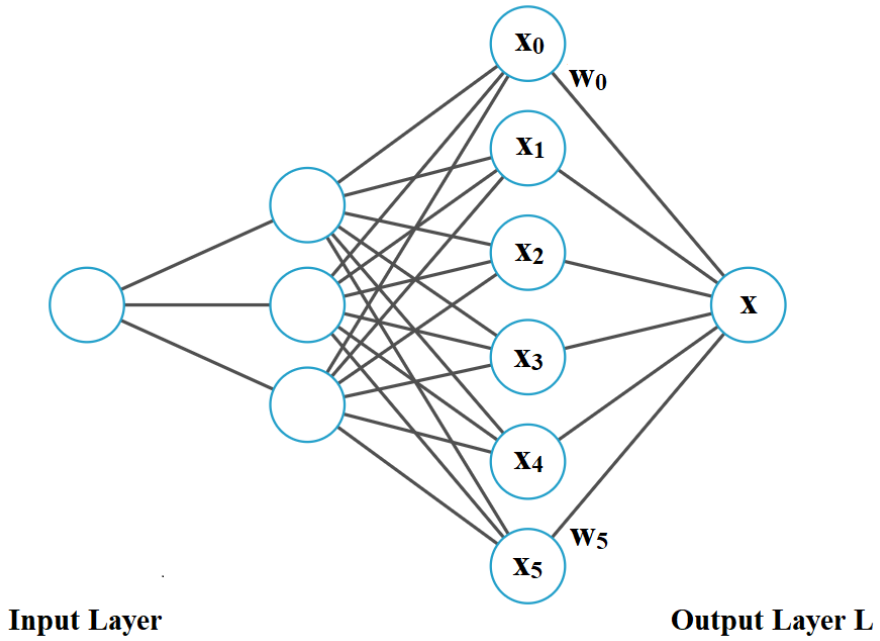


Figure 12: *Neural Network illustration of two hidden layers and one output layer, where x_i the individual neurons and $w_i$their corresponding weights.*

In order for the output $\boldsymbol{x^L}$ to better approximate the ground truth $\boldsymbol{Y}$, its weights w_i and bias should be precisely adjusted in the direction that brings the loss function E closer to a minimum. This is quantified by the partial derivative of the loss with respect to the weights as:

$$\frac{\partial E}{\partial w^L} = \frac{\partial E}{\partial x^L}\frac{\partial x^L}{\partial w^L} \qquad (27)$$

Intuitively, this describes how a change in the weights w^L would evoke a change in the loss function E. Notably, this is an indirect effect, as sifting w^L would first affect the output $\boldsymbol{x^L}$, which would consequently alter the loss function E we aim to minimize. This is referred to as the chain rule and is also applied to modify the biases of the L^{th} layer accordingly:

$$\frac{\partial E}{\partial b^L} = \frac{\partial E}{\partial x^L}\frac{\partial x^L}{\partial b^L} \qquad (28)$$

Back Propagation:

Considerably, the reported gradient computations account for the modification of the last layer. To commence the model's training, gradient descent is recursively applied from the activations of the output layer moving backward for the entirety of the network. This is implemented by the back-propagation algorithm (Rumelhart, et al. 1986). Hence, for a given training example the refinement of the network's weights is attributed to the magnitude of each value in the vector of computed gradients as:

$$W' = W - \eta\frac{\partial E}{\partial W} \qquad (29)$$

where η is the learning rate. Similarly, this process is applied for the update of the biases of the network. Notably, to avoid running back-propagation for all training examples, which is computationally prohibitive, a subset of them is randomly selected referred to as 'mini-batch'. Stochastic gradient descent formulates that technique, hence speeding up the training process significantly while providing a close approximation in terms of error.

2.2.3 Activation functions

As described, the learning process through gradient descent requires the sifting of activations toward the correct prediction. Activation functions aid the model's ability to learn non-linear patterns, as each encloses all the adjusted weights and bias of the previous layer in determining whether a neuron should activate or not.

$$x_n^{j^{th}} = \sigma\left(w_0 x_0^{j^{th}-1} + w_1 x_1^{j^{th}-1} + \cdots + w_{n-1} x_{n-1}^{j^{th}-1} + b\right) \qquad (30)$$

where σ is the activation function, x is the activation of the i^{th} neuron in the j^{th} layer and b is the bias term.

Sigmoid function:

The Sigmoid activation function, also referred as logistic function, assumes an S similar shape and transforms an input value to the range of [0, 1]. It is suitable for probabilistic outputs, usually is utilized in the last classification layer of the model. Sigmoid is defined as:

$$S(x) = \frac{1}{1 + e^{-x}} \qquad (31)$$

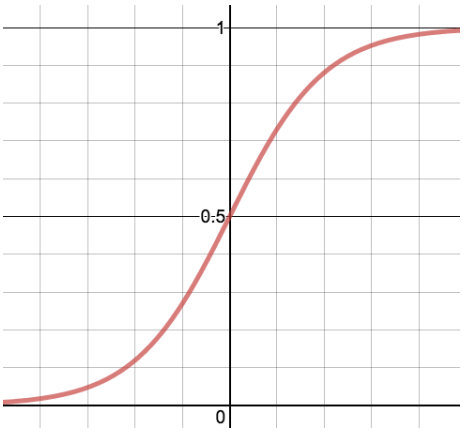


Figure 13: *Sigmoid activation function.*

Rectified Linear Unit:

ReLU (Nair, et al. 2010) is a non-linear activation function in the range of $[0 \rightarrow \infty)$. It is computationally efficient in terms of accelerating the convergence of gradient descent, as it requires only a threshold operation. ReLU outputs x when $x \geq 0$ and 0 when $x < 0$ and is defined:

$$f(x) = \max(0, x) \qquad (32)$$

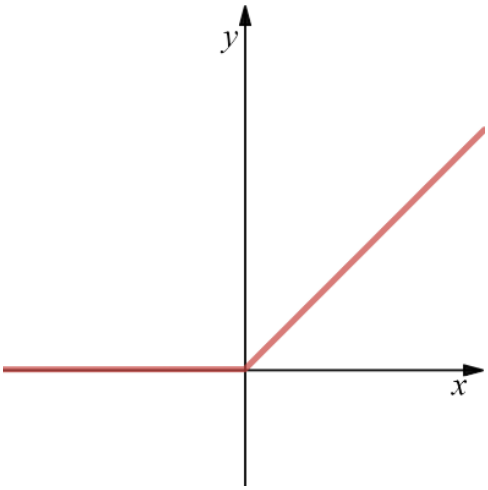


Figure 14: *ReLU activation function*

Hyperbolic tangent:

Tanh is a zero-centered activation function in the range of $[-1, 1]$. It is appropriate for inputs that have strongly negative, neutral or positive values as they will be mapped near -1, 0 or 1 respectively. Tanh is defined as:

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \qquad (33)$$

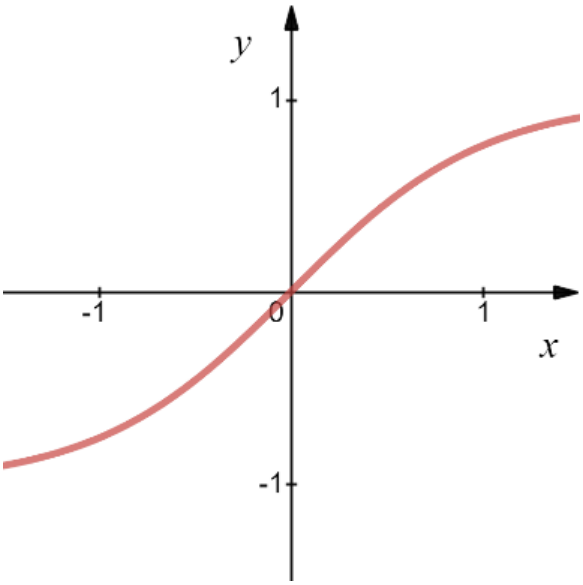


Figure 15: *Tanh activation function.*

2.2.4 Regularization Techniques

The process of training a neural network often entails an essential mechanism to defend it against overfitting. By overfitting we refer to the tendency of the model to learn the noise and random fluctuations in the training set, which in turn limit its ability to generalize on new data and therefore reduce its performance capabilities. Regularization techniques combat this problem by imposing a penalty to the loss function, so as to limit interdependent learning among neurons.

Dropout: One effective regularization technique that prevents overfitting is dropout introduced by (Srivastava, et al. 2014). During training a random subset of neurons are 'switched off' with a probability of p and therefore only the remaining neurons pass information to the next layer for a single iteration. This helps the network generalize better as it is provided with a different 'view' of active neurons at each iteration, unable to form dependencies on a specific subset of them. An illustration is provided in ***Fig.16.***

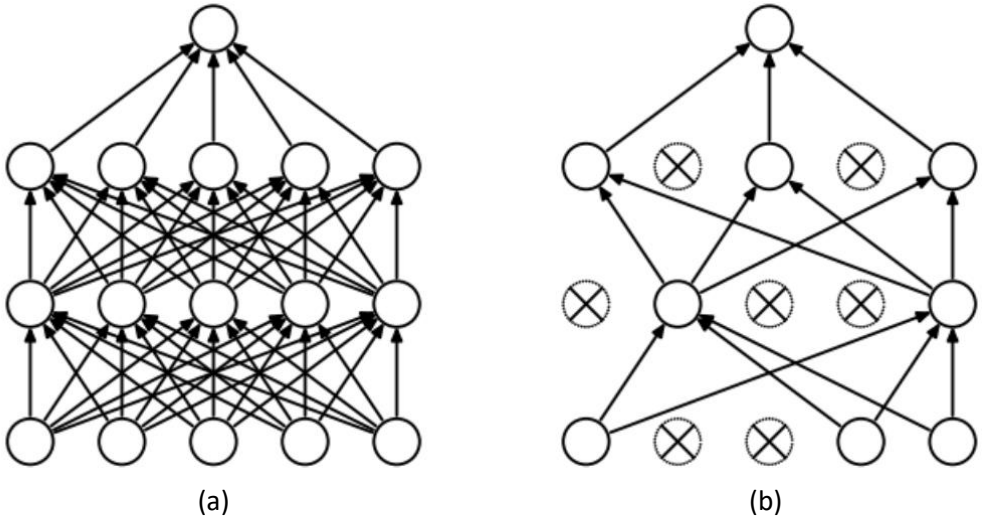


Figure 16: *Dropout technique. (a) no dropout applied, (b) with dropout.*

Batch Normalization: The growing application of deep neural networks prompted the creation of the Batch Normalization algorithm (Ioffe & Szegedy, 2015) in order to speed up convergence as well as make neural networks more robust in terms of hyperparameter tuning. This technique tackles the problem of the "internal covariate shift", which refers to the changes of distributions in the internal layers of the network. These fluctuations are attributed to the constantly changing weights of the network, as we mentioned in the training process, where the input of a given layer is determined by its predecessor's shift of weights. This phenomenon despite of slowing down the learning process is also restrictive in terms of parameter initialization.

By employing batch normalization, the output of the selected hidden layers will be standardized, with zero mean and unit variance as calculated from a given mini-batch. This aids the learning process, as it ensures that the hidden layers will have similar distributions, which in turn sets a more robust ground for the weights of deeper layers. Variations of this basic idea include applying batch normalization prior to the nonlinearity or after.

The implementation of batch normalization for a given activation y_i first computes the mean μ and variance σ of the m^{th} minibatch as shown below:

$$\mu_m = \frac{1}{K}\sum y_i \qquad (34)$$

$$\sigma_m = \sqrt{\frac{1}{K}\sum (y_i - \mu_m)^2} \qquad (35)$$

where *K* denotes the size of the mini-batch.

Then the corresponding normalized activation $\grave{y}$ for the i^{th} sample is given as:

$$\grave{y}_i = \frac{y_i - \mu_m}{\sigma_m} \qquad (36)$$

However, it's important that each normalized output is not fixed to a specific mean of zero and unit variance as the network might find it more suitable to shift these values slightly to better exploit the non-linearity of the utilized activation function. For example if we used the sigmoid activation function the network could respond better to a larger variance than 1.

To impose this notion into batch normalization, the authors inserted the trainable parameters β and γ as the final step of the algorithm to compute the response y_i as follows:

$$y_i = \gamma\, \grave{y}_i + \beta \qquad (37)$$

Despite its effect of stabilizing and speeding up the training process of the network, batch normalization offers a slight regularization effect because the mean and variance are estimated within a small training sample(mini-batch), hence inserting noise to each hidden layer's activations, similarly to dropout.

Early Stopping: This technique (Morgan, N., & Bourlard, H.,1989) enables the model to seize its learning process once its performance stops improving, hence preventing a large, fixed number of epochs leading it to overfitting. The evaluation happens in terms of observing how the model performs on the validation set during the course of learning. Specifically, after having split the training data into a training set and a validation set, we compute an approximation of the generalization error based on the validation set for each example and stop the training process when this error stops decreasing.

Although, because the validation error function contains more than one local minimum ***(Fig.17)*** it is essential to choose the right stopping criteria, to better balance the tradeoff between training time and generalization error (Prechelt, L.,1998). Also, different variations in implementations entail different measures of performance, for example some stop the training of the model when its accuracy metric stops increasing for a given number of subsequent epochs, while others set loss as the performance metric to trigger early stopping.

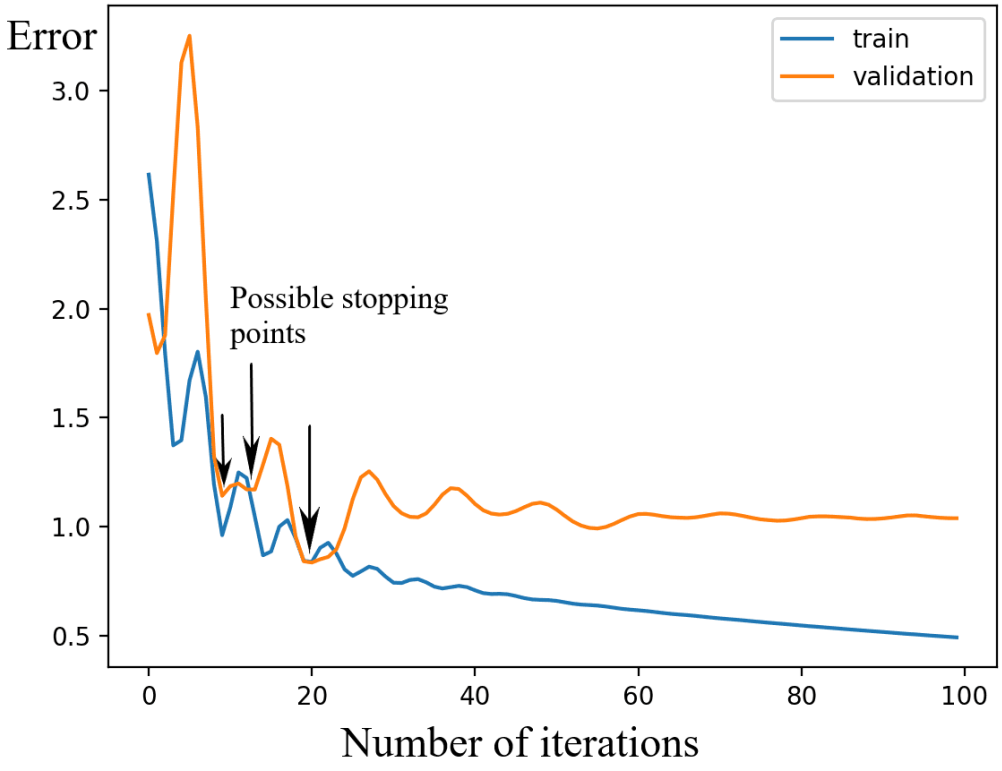


Figure 17: *Training and validation error curves. Number of training iterations displayed on the x-axis. Arrows indicate possible points (minimums) for stopping the training process.*

3 Related Work

A variety of different methods tackle the interpretation problem with the aim of revealing the inner workings of deep learning models. In this section we group them into two main categories, post-hoc and intrinsic, based on whether the explainable aspect of their method relies on an external factor to provide interpretation or its embedded into the model respectively.

3.1 Post-Hoc Methods

3.1.1 Perturbation-based

The explainable algorithms in this category focus on variations in the input feature space, estimating its vulnerable points to explain individual feature attributions that lead to the output response of the network. Therefore, we can observe the model's predictions when impactful aspects of the input are modified. Perturbation can be imposed on the network in various ways, such as masking, substituting features with zero or random counterfactual instances, occlusion, conditional sampling, etc. (Zeiler, et al 2014) omitted different regions of the input image to generate layer-wise visual representations by employing a deconvolutional network. A DeConvNet is comprised of rectification, unpooling and deconvolution operations and can provide illustrations of feature activity by inverting them back to the input space. Meaningful Perturbations (Fong, et al. 2017) proposed a model-agnostic framework that discovers the most impactful regions of the input image which affect the classifier's prediction when perturbed, by utilizing a blurring operation for feature elimination. External Perturbations (Fong, et al. 2017) is an extension of the previous framework *(MP)* that introduces a new area constraint and ensures the application of smooth masks, for the preservation of features and not their removal unlike before. Prediction Difference Analysis (Zintgraf, et al. 2017) proposed is a method for visualizing the regions of the input image that provide evidence of positively or negatively impactful features toward the model's prediction. They estimate a relevance value for each input feature and monitor the output response of the network when removing these features individually or in groups for a specific input.

Notably, these perturbation-based methods extract visual explanations from the input-level view, an approach that disregards the inner workings of models and the possible interpretable insights they could provide. Among them are also techniques that construct proxy models, with similar behavior to the original model yet simplified and impose the perturbation notion on a feature level. LIME is a method proposed by (Ribeiro, et al 2016) that provides explanations by

iteratively perturbing the input data to construct a regularized linear model that serves as the proxy. This proxy is then fitted locally, in order to identify and provide a representation of the most impactful regions of the image around a prediction. SHapley Additive exPlanations (SHAP) is a method proposed by (Lundberg, et al. 2017) that assigns importance scores to features and explores their correlations by removing them, exploiting game-theoretic knowledge to distribute the scores among features fairly for an individual prediction. The authors also proposed several modifications of the baseline such as KernelSHAP, DeepSHAP, etc. to make it more efficient for different applications. Similar with SHAP although seeking to explain the behavior of the model globally (Covert, et al 2020) proposed the SAGE method, which is a model-agnostic approach that accounts for feature interactions.

3.1.2 Gradient-Based

Another set of techniques that reduce the complexity of operations toward interpretability are gradient-based methods which utilize the gradient that is backpropagated from the output prediction back to the input layer. These methods construct saliency maps presenting the most active areas of the network. Saliency maps are explainable visualizations that require a forward and a backward pass of the network to be generated, firstly introduced by (Simonyan et al. 2013) where feature importance was translated to visualizing the gradients. Guided Backpropagation (Springenberg, et al. 2014) offers visual representations of internal network units, by suppressing the negative gradients in the backpropagation process. Class Activation Mapping (CAM) proposed by (Zhou, et al. 2016) produces visualizations of the linear combinations of activations in the last convolutional layer with a single forward pass. Grad-CAM (Selvaraju, et al 2017) improves the CAM technique by incorporating Guided Backpropagation and also displaying negatively highlighted regions of the image. To remove noise imposed by the gradients *(*Smilkov, et al 2017) proposed the SmoothGrad method by adding noise to the input image and averaging the obtained saliency maps. The LRP method introduced by (Bach et al. 2015) passes the relevant scores of the outer layer back to the input layer through back-propagation to visualize the contributions of single pixels. In this context of generating salient image features through backpropagation, a variety of heuristic techniques exist including *DeepLIFT (*Shrikumar, et al 2017***),*** *integrated gradients (*Sundararajan*, et al 2017), BASNet (Qin, et al 2019).* These techniques can display regions where neurons fire the strongest and regions of network sensitivity, however as indicated by (Adebayo, et al 2018) they can be prone to adversarial attacks and input invariances, thus solely relying on

their generated explanations can be misleading. Multimodal Explanations counters that by providing complementary justifications proposed by (D.H.Park, et al. 2018), in terms of visual attention maps as well as natural language explanations that are generated after the model had reached a decision.

3.2 Intrinsically interpretable methods

Opposed to most of the previously mentioned methods which are utilized on pre-trained models, others while remaining in the context of providing transparency are considered inherently interpretable. As these approaches are driven by the shared aim of creating explanation producing systems, their designs vary in terms of training by disentangling representations or reformulating their architecture to be self-explainable.

3.2.1 Disentangling Representations

Approaching interpretability from a different angle, existing methods aim to learn interpretable representations by feature disentanglement. The proposed beta-VAE (Higgins, et al. 2016) method disentangles latent factors from raw image data in an unsupervised manner, as a modification of the variational autoencoder (VAE) framework. Network Dissection (Bau, et al 2017) is a framework that quantifies how hidden units locate semantic concepts across the range of objects, parts, colors and textures that are not explicit in the training phase of the model. A method that disentangles CNN representations to decision trees was proposed by (Zhang, et al 2019), where the constructed decision trees convey the information of which filters were impactful to the final prediction in a semantic level. Addressing the entanglement of intermediate interactions with the aim of obtaining their intrinsic interpretable structure by penalizing interaction orders and reducing a neural network to a generalized additive model with interactions, (Tsang, et al 2018) introduced the Neural Interaction Transparency (NIT) framework. In contrast with these methods, our approach prevents the costly process of disentanglement since each feature is contained within a subnetwork and its individual verdict is separated from the rest in the produced explanations. (X. Chen et al. 2016) introduced InfoGAN as an extension of Generative Adversarial Networks to learn interpretable representations by maximizing the mutual information between the latent codes and observations in an unsupervised manner, with the aim of encoding semantic concepts and minimizing entanglement between latent factors. Capsule Network was proposed by (Sabour et al. 2017) to disentangle and represent higher-level concepts by introducing capsules, as substitutes of neural units, that capture specific semantic concepts encoded in their activity vectors whose

direction represents the instantiation parameters and their length is indicative of the probability an object is present. Proposed as a modification of traditional CNNs by (Zhang, et al 2019) Interpretable CNNs regularize filters by two losses to aid their disentanglement in order for them to encode distinctive object parts in high-convolutional layers and respond to local regions of the image without requiring any annotations, although their practical applicability is limited due to degradation in performance.

3.2.2 Explainable reformulation methods

Receiving much less attention are methods which seek to deploy models with interpretable components taking form within their structure, in order to make the subsequent task of interpretating their decisions more straightforward. (Alvarez-Melis, et al. 2018) proposed a reformulated self-explaining class of neural network models by progressively generalizing a simple linear regression model to learn interpretable concepts while enforcing basic desiredata for interpretability. (Kuo, et al. 2019) offers a different approach toward assigning filter weights in CNNs by eliminating the need for backpropagation and deriving the parameters of a layer based on data statistics form the response of the previous layer. The authors design an interpretable feedforward convolutional network by substituting its convolutional layers with a variant of the principal component analysis (PCA) and its fully connected layers with linear least square regressors. Based on the additive index model, (Vaughan, et al. 2018) proposed the Explainable Neural Network (xNN) structure, that provides insights of how the model utilizes its additive input features which are learned by linear projections and univariate functions, to reach the final prediction. This category of methods also accommodates our proposed approach as an extension of the GAMs family of models.

4 Methodology

In this section, we introduce the architecture and training procedure of our method in the context of binary classification. We further describe the mechanisms behind the model's reasoning and how it justifies its predictions.

4.1 ConGAM Model

To present our approach more concisely we briefly consider the methods that led to its formulation. We initially consider the problem of linear regression where our aim is to approximate the true effect of an unknown quantity as a sum of effects multiplied by covariates. A linear regression model assumes that a constant change in the effects of covariates results in a constant change in the dependent variable $\boldsymbol{y}$. In this context we assume that the response variable follows a normal distribution $y \sim N(\mu, \sigma^2)$ with mean μ and variance σ and that the covariates are independent and of equal variance $y_i|X \sim N(0, \sigma^2)$. Therefore, the response variable $\boldsymbol{y}$ is computed as a sum of linear predictors x_i multiplied by coefficients β_i in the following form:

$$y_i = \beta_0 + \sum \beta_j x_{ij} + \varepsilon \qquad (1)$$

including the constant term of the intercept β_0 and the error term ε which is a random variable that captures other factors that influence $\boldsymbol{y}$.

Notably, this model in its basic form is quite restricted by its assumptions when it comes to capturing more complex types of the response variable. This led to its generalization as formulated by Generalized linear models (GLMs) which allowed the response variable to fall within various types of distributions besides the normal distribution, such as the Poisson or the Bernoulli distributions among others. The GLM relates the mean μ of the selected distribution that generates $\boldsymbol{Y}$ with the linear predictor through a link function as:

$$E(Y|X) = \mu = g^{-1}(X\beta) \qquad (2)$$

where $E(Y|X)$ is the expected value, $X\beta$ denotes the linear predictor with its coefficient β and the link function as $g(.)$.

The link function embodies the relationship between the expected value and the linear predictor, in a sense that now the link function is varying linearly along with the predictor, instead of the response variable y being directly affected by a change of the covariates. Link functions are selected based on the corresponding distribution of $\boldsymbol{Y}$ and the purpose of the application, some commonly used examples are the identity, log and logit functions taking form as:

$$g\big(E(y)\big) = X\beta \qquad (3)$$

where the β parameters are obtained from mathematical methods such as Bayesian or maximum likelihood. Worth observing is how the typical linear regression model we previously mentioned of the form $E(y) = X\beta$ is a GLM with an 'identity' link function and a Gaussian distribution.

Generalized Additive models stand as an extension of the previously mentioned GLMs. They take form by incorporating non-linear functions as part of the predictors to allow smooth arbitrary effects to be estimated from the data in order to better approximate the target variable. Similarly with GLMs the response variable $\boldsymbol{Y}$ can be generated from a set of various distributions and the relationship between the predictors and the expected value of $\boldsymbol{Y}$ is expressed by a link function $g(.)$. GAMs can therefore be presented in the following form:

$$g(E(Y)) = \beta + \sum_i f_i(x_i) \qquad (4)$$

where $E(Y)$ is the expected value of $\boldsymbol{Y}$, each f_i is a univariate smooth function (e.g polynomial), $\boldsymbol{x} = (x_1, x_2 \ \ldots, x_k)^T$ is the input feature vector, β is the intercept and $g(.)$ denotes the link function.

Notably, the differentiating point from GLMs is the utilization of smooth functions wrapped around the predictor variables, that can be thought of as building blocks which are linearly combined to fit the target variable more smoothly. The property of smoothness of a function is intertwined with the calculation of its 2nd order derivatives, which show the rate of change in the slope at any given point, indicating the 'curvature'. Hence, a function is considered smooth if it is differentiable at every point. Different variations of smooth functions are reported in (Buja, et al. 1989) and can generally be viewed as:

$$f_i(x_i) = \sum_k \beta_{ik} b_{ik}(x_i) \qquad (5)$$

where $b_i(.)$ are basis functions to capture non-linearity (e.g b-splines) and β_i the corresponding coefficients which help guide the fitting of the model.

The learning process of GAMs aims to establish a balance between the complexity of the model and how well it fits the data. This is determined by the penalized estimation of the coefficients, which ensures that our model doesn't overfit by imposing a higher penalty when the smooth functions are more complex. This can be described in the form of:

$$L_p(\beta) = L(\beta) - \lambda\beta^T S\beta \qquad (6)$$

where $L(.)$ is the log likelihood measure we seek to maximize, λ controls the penalty and the $\beta^T S\beta$ term expresses the lack of smoothness of the f_i , and S the matrix of known coefficients.

ConGAM provides an alternative formulation in terms of fitting convolutional neural networks to GAMs via gradient based training for the task of image classification. The models we employ are comprised of the feature extraction part that encloses blocks of convolutional layers and the classification part that includes fully connected neural network components. The latter part simulates a fully connected network (FCNN) and therefore ensures the approximation capabilities the f_i functions should entail, as it captures all the univariate activations of the previous layer.

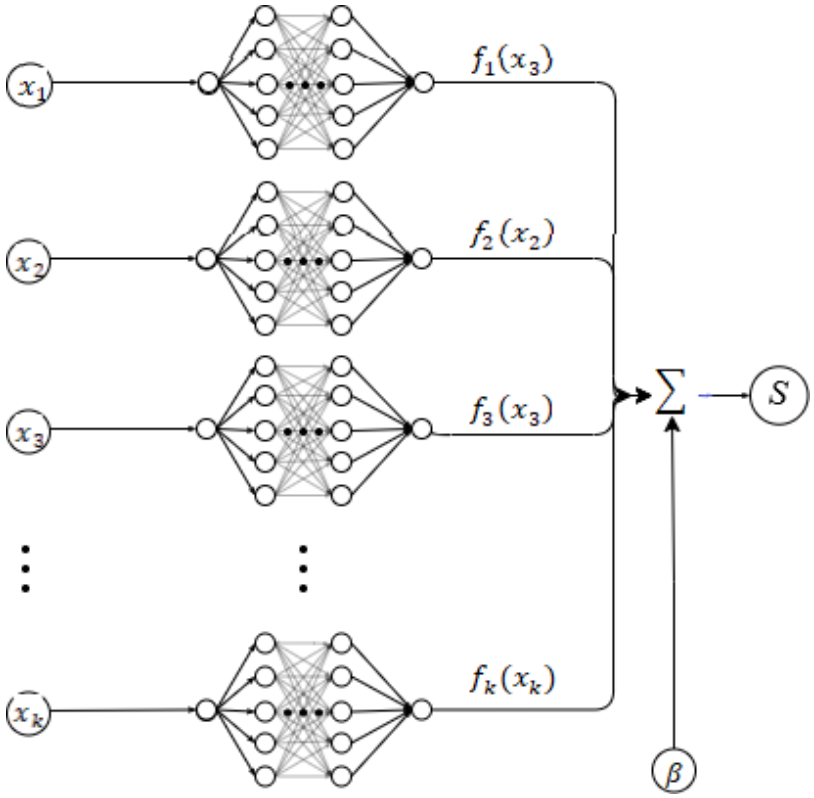


Figure 18: *ConGAM Framework.*

The CNN ensemble is trained jointly through back-propagation and each individual network handles a separate feature. Hence, the interpretation mechanisms of the resulting model are intrinsic to its structure, simplifying the subsequent task of interpretation. Accordingly, ConGAM's architecture is presented in ***Fig.18***. As depicted, each subnetwork is comprised of an input node, multiple hidden layers and an output node that captures the response of the feature assigned to the corresponding subnetwork, all the feature responses along with the bias are then summed before being propagated to the sigmoid function to present the final output.

The proposed method brings forth some advantages over the initial GAMs approach:

- It can exploit the advancements in GPU computing regarding the subnetwork training and make use of the various efficient algorithms for CNNs. Notably, it can utilize mini batch gradient-based methods and profit in terms of memory.
- The high discriminative power of GAMs comes with the cost of millions of decision trees, due to the additive nature of the algorithm, while ConGAM requires a small ensemble of subnetworks without hindering intelligibility.
- Moreover, tree-based GAMs can be restrictive as they require more alterations in their formulation to be employed to different settings, such as multi-label learning, unlike a CNN based approach which is more adaptable.

4.2 Subnetwork Architecture

The presented reformulation of GAMs involves convolutional neural networks since their role in image classification tasks is determinant. As we mentioned in a previous chapter CNNs' building blocks form a feed-forward hierarchy that introduces non-linearity to extract features. In this process the convolution operation is utilized at layer-level where the feature detector applied is constant. In our method we exploit the ability of CNNs to capture spatial information as we employ some of the various operations that comprise their functionality, in order to guide the model's behavior. The process of building the individual subnetworks included a series of steps that will be briefly mentioned in this section before presenting the utilized architectures.

Initially, 12 models were generated by altering the number of blocks of convolutional and pooling layers between 2 or 3, modifying the number of fully connected layers to one of the [1,2,3] values and the number of convolution layers of the last block to one of the possible values in the set of [1,2,3] resulting in *3x2x2* possible combinations as shown in ***Table 1.***

To prevent overfitting, the dropout regularization technique was utilized in all 12 models after each convolutional block and between the last 2 fully connected layers.

***Table 1**: Model combinations. 'conv layers' indicates the number of convolutional layers in each block and 'FC' is the number of fully connected layers. The best one appears in bold.*

Model	Blocks	Conv layers	FC
1	**3**	**2.2.3**	**1**
2	3	2.2.3	2
3	3	2.2.3	3
4	2	2.3	1
5	2	2.3	2
6	2	2.3	3
7	2	2.2	1
8	2	2.2	2
9	2	2.2	3
10	3	2.2.2	1
11	3	2.2.2	2
12	3	2.2.2	3

Next, we introduced the batch normalization technique with 4 different modifications, applied on the best performing model from the previous step. These modifications include the replacement of dropout and parameter alteration of the batch normalization layer as shown in ***Table 2***.:

***Table 2**: Batch normalization parameter combinations. The best one in terms of performance appears in bold.*

Model	Momentum	Scale	Trainable	Dropout after blocks
1	0.9	TRUE	TRUE	YES
2	0.9	TRUE	FALSE	YES
3	0.9	TRUE	FALSE	NO
4	**0.9**	**TRUE**	**TRUE**	**NO**

ConGAM-base:

Several modifications followed regarding different depths of the fully connected layer and first convolutional layer, a slight variation in the number of convolutional layers and the replacement of the dropout operation between the fully connected layers with batch normalization, resulting in the model architecture of the following symbolic form:

$$HxWx1 - [\,2x64C3 - MP - BN\,] - [\,2x128C3 - MP - BN\,] - [3x256C3 - MP - BN] - 128F - F$$

representing an input image of size HxW, two blocks of two convolutional layers with *3x3* filters*(C3)* of 64 and 128 depth respectively, a max-pooling layer*(MP)* and a batch normalization layer*(BN)*, another block with 3 convolutional layers of 256 depth, followed by a fully connected layer*(F)* of 128 neurons and a fully connected output of 2 classes as depicted in ***Fig.19***. We applied the non-linear ReLU activation function for the convolution and fully connected layers and the hyperbolic tangent activation function for the output layer.

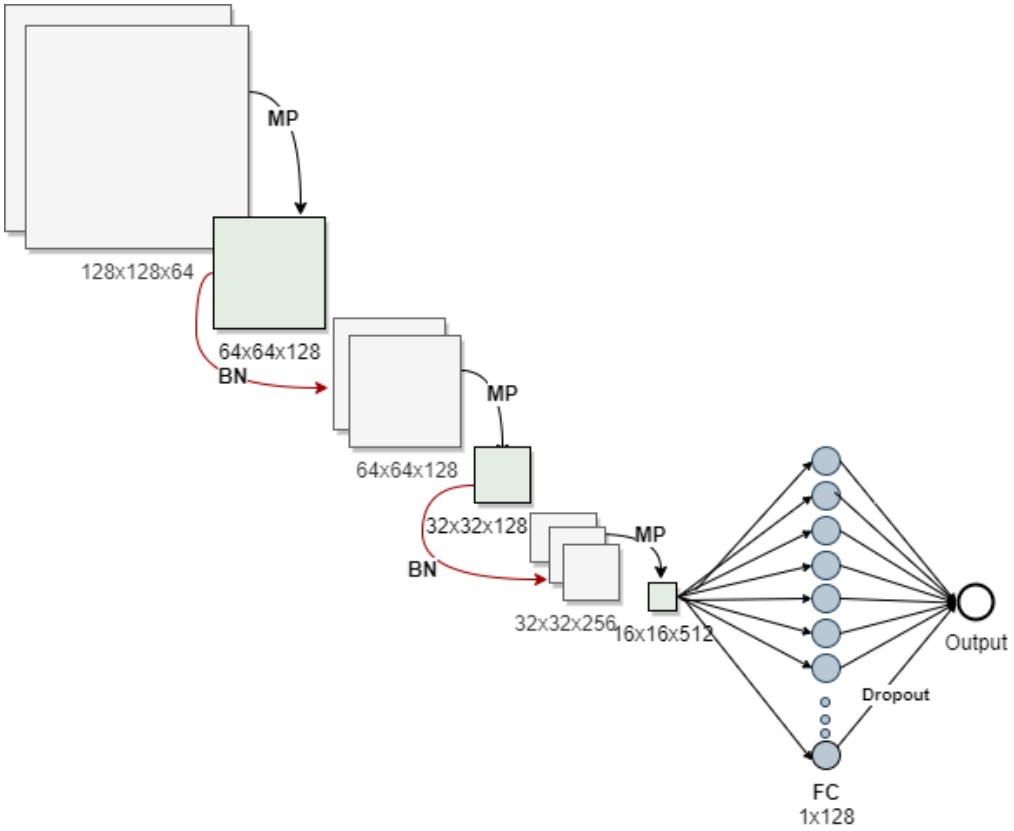


Figure 19**:** *Subnetwork Architecture. The gray squares indicate convolutional layers, the green squares are max pooling layers and the fully connected layer 'FC' is depicted as a set of nodes with circular shape. 'BN' refers to the batch normalization operation. The dimensions of data at each stage are displayed below each scheme.*

ConGAM-Inc:

An alternate subnetwork architecture follows the idea of the GoogLeNet, also known as inception model (Szegedy, et al. 2014). In its proposal the creators deviated from the conventional approach of building a CNN model by stacking layers in a sequential manner and introduced the concept of a sparsely connected architecture. Specifically the structure that imposes that notion is the inception module, where multiple-sized filters operate in parallel to create the input of the next layer after being aggregated. An example of an inception module is illustrated in ***Fig.20*** where to form the output, 3 convolution operations of different sized filters $(1x1, 3x3, 5x5)$ and a max pooling operation are merged together. Also to reduce dimensionality in terms of depth, *1x1* convolutions are computed before the *3x3* and *5x5* convolutions and after the max pooling operation.

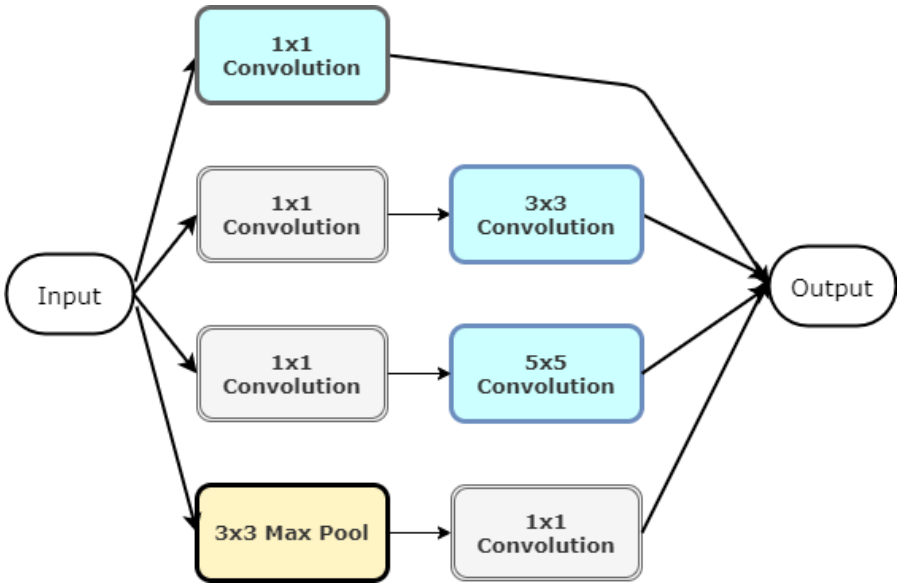


Figure 20: *Inception module that outputs their aggregation of 3 convolutional layers of filter sizes: 1x1, 3x3, 5x5 and a 3x3 max-pooling layer after being reduced by 1x1 convolutions.*

Intuitively, by using the inception module we provide the next layer a variation of information to learn from, enclosing high grained and more abstract details, as captured by the different sized filters. In our method different variations were tested in regard to the placement of the inception module, ultimately distinguishing in terms of performance, the model that placed it in the first block in the formulation that follows:

$$HxWx1 - [32C3 - IM - MP - BN\ ] - [2x64C3 - MP - BN] - [3x128C3 - MP - BN] - 128F - F$$

representing an input image of size $HxWx1$, a block of one convolutional layer with 3x3 filters*(C3)* and depth of 32, one inception module*(IM)*, a max-pooling layer*(MP)* and a batch normalization layer*(BN)*, followed by two blocks with 2 and 3 convolutional layers of *3x3* filters and depth 64 and 128 respectively, max pooling and batch normalization operations, along with a fully connected layer*(F)* of 128 neurons and a fully connected output of 2 classes as depicted in ***Fig.21.*** We applied the non-linear ReLU activation function for the convolution and fully connected layers and the hyperbolic tangent activation function for the output layer.

The inception module denoted as IM in the model's formulation is the one depicted in ***Fig.20*** as:

$$IM = concat[32C1, \quad 32C1 - 64C3, \quad 32C1 - 64C5, \quad MP3 - 32C1]$$

where the parts being aggregated together to form the output are separated by commas enclosed in *'concat'* which denotes the concatenation operation and includes: a convolutional layer with *1x1* filters and a depth of 32, a convolution layer with *3x3* filters of 64 depth after being reduced by a *1x1* convolution, another convolution layer with *5x5* filters of 64 depth after being reduced by a *1x1* convolution and a max-pooling layer of size *3x3* also reduced by a *1x1* convolution.

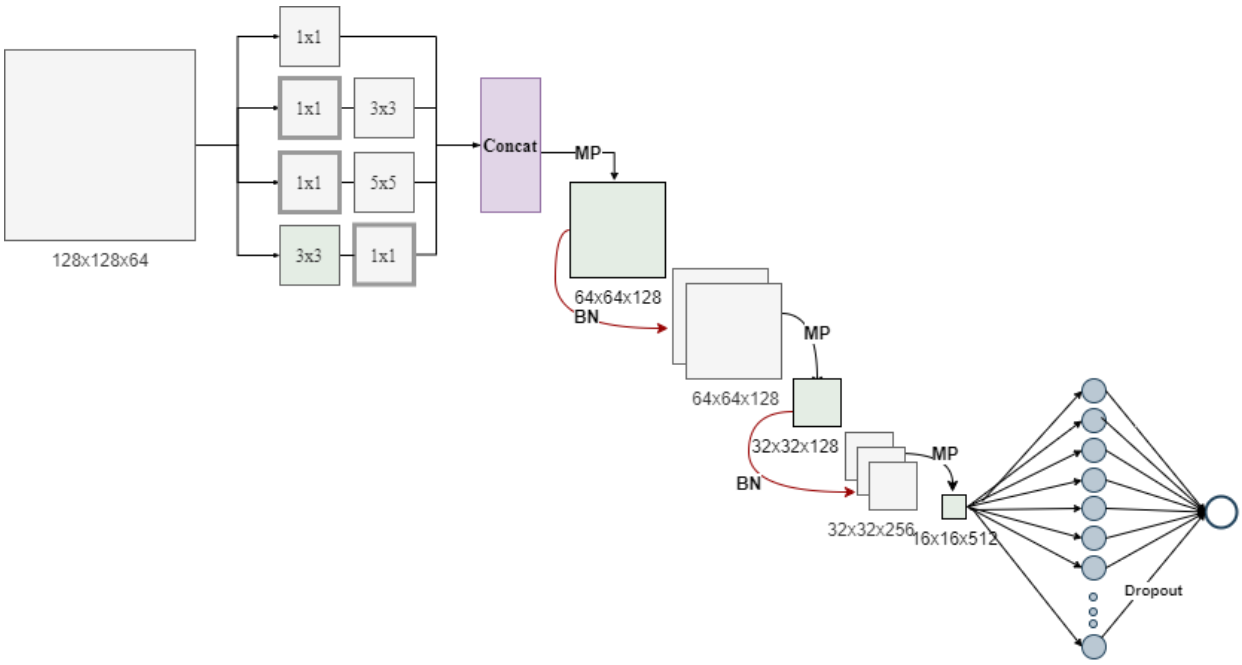


Figure 21*: Subnetwork architecture with inception module after the first convolutional layer. Schemes follow the same meaning as the previous architecture.*

From a high-level perspective, the network is formed as an ensemble of a pre-specified number of subnetworks defined by one of the proposed architectures, each attending to a different feature. The separate networks are trained jointly through back-propagation to learn the right associations toward the true labels. For a given image, the univariate responses of all the subnetworks are summed and passed through the sigmoid function to yield the probabilities for the final prediction.

4.3 Feature Engineering

As ConGAM's intrinsic interpretability resides in its structure, feature engineering utilizes that to guide the model's reasoning towards a prediction. Similarly to how humans understand the different descriptive concepts within an image, features aim to faithfully approximate them. Therefore, enhancing intelligibility since each separate network's response depicts the impact of the extracted feature information toward the final output. This in turn reinforces user's flexibility in selecting appropriate features while assessing their direct responses, depending on the application. In this section we report the different image representations we offered our subnetworks and the information they convey.

Color: In our approach regarding the information passed into the individual networks, color was considered as it constitutes an important attribute providing information on human perception. Notably, the objects of interest within the image fall into a similar color gamut that distinguishes them from other classes. Arguably there is overlap among the ranges of values objects carry, but ultimately their color distributions convey patterns that help differentiate images among the different classes. For example, apples primarily appear red and bananas yellow and although there is some overlap in the green and yellow color ranges (since we find green apples and bananas, as well as yellowish apples), the overall distributions of color we get from each class may guide the learning process of our model. But how do we convey that information in an explicit manner for the model to approximate our own perception in terms of color?

As we have mentioned in a previous section the widely used RGB color space encodes brightness in every channel and therefore any illumination-related modifications will affect the image's chrominance (color-related information). Furthermore, it is perceptually irrelevant as the changes in color are hard to follow due to discontinuities. These properties of brightness dependence and perceptual non-uniformity hinder our purpose of conveying explicit chroma information into the network, thus making the RGB color space unsuitable.

In our method we utilize the RGB to CIELAB color-space conversion that translates this problem into a color space that separates brightness and chromaticity (hue and saturation) into its channels. Specifically, we elicit the information of the a^* and b^* color-opponent dimensions of CIE L*a*b into two separate subnetworks to capture the color distribution of images corresponding to the green-to-red and blue-to-yellow color ranges respectively. In the following ***Fig.22*** we demonstrate an example with the a^* and b^* color channel representations of an image which in our model are conveyed to two separate subnetworks.

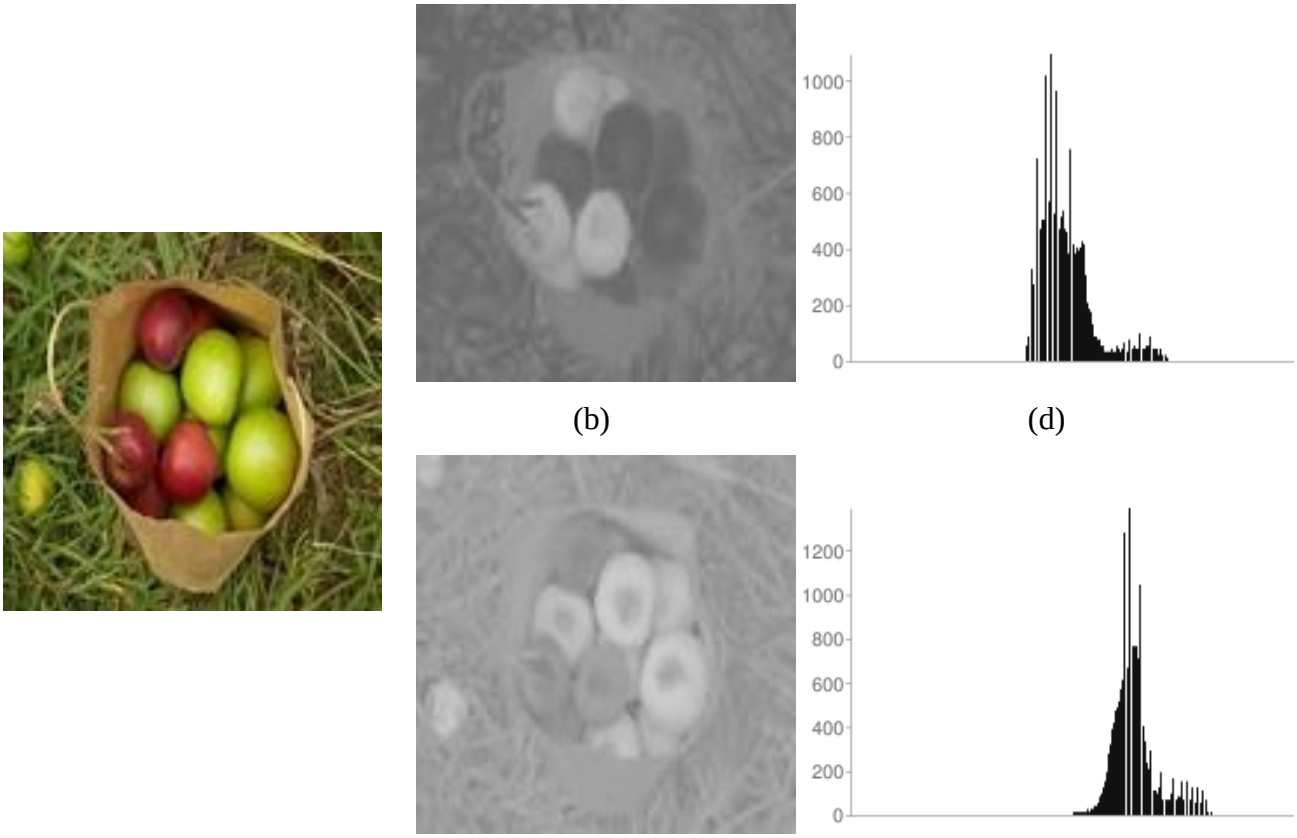


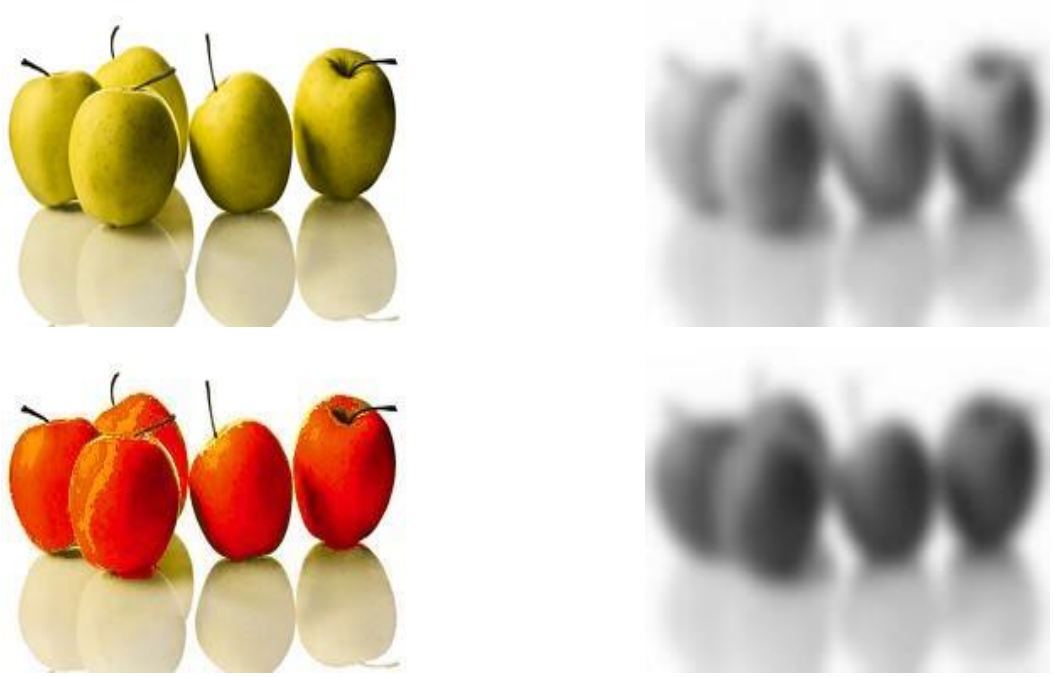
Figure 22: *Visual representation of the a* and b* color components of CIE-Lab. (a) original image, (b): a-channel image representation, (c): b-channel image representation and (d),(e) their corresponding distribution of color(histograms).*

Gaussian Blur:

Another aspect of information fed into an individual network considers the Gaussian blur effect. To simulate that we have applied a gaussian filter on our image data with a standard deviation of the Gaussian distribution equal to a value of five ($\sigma = 5$). As we impose this degradation effect the image's high frequency variations in brightness get smoothed out, encoding an indirect effect of luminance into the subnetwork, as well as its structure related attributes.

In this way we opt for the subnetwork receiving this image representations to extract patterns that differentiate objects based on how they reflect light in an image. Hence, this feature can be considered rather abstract since the information it encodes is not tangible. Specifically, brightness is a descriptor that embodies the notion of achromatic intensity in a *subjective* manner* and regarding the structure of the filtered image, even though somewhat preserved, its basic elements are smeared out, distanced from their true depiction. In ***Fig.23*** we offer an example of two similar images and their filtered representations of abstract luminosity information passed into an individual subnetwork.

it should be noted that the brightness mentioned here differs from the non-linear perceived lightness(L*)* channel of CIE-Lab.



(a) (b)

Figure 23**:** *Gaussian blur on similar images that reflect light differently. Column (a) RGB images, Column (b) Blurred L channel.*

Edge Detection:

Aiming to capture regions of discontinuity in an image, while preserving its structural features led to the implementation of the Sobel edge detection filter to generate a set of image representations that would capture shape-related information.

Following our statements in a previous chapter about edge detection, in our method we aim to detect sharp variations in pixel values that are translated to the physical extent of objects within an image, as captured by the utilized filter. This operation encodes important shape information, thus making the subsequent task of interpreting the subnet's response more intuitive. As illustrated in ***Fig.24***. we depict an example of two different images and their corresponding representations as captured by an individual subnetwork, when the Sobel edge detector is applied.

(a) (b) (c) (d)

Figure 24**:** *Sobel edge detector. (a) and (c) are the original apple and banana images. (b) and (d) are their respective filtered representations.*

In the next section we describe how the mechanisms within the structure of our model help justify its predictions.

4.4 Model Interpretation

The proposed model's interpretability is attributed to its intrinsic structure and to the verdicts it produces for its classification results. ConGAM is considered inherently interpretable as the explanations it generates are part of its explicit training. The linear combination of the subnetworks is presented in a disentangled form, where each independent input feature is handled by an individual network, enclosing different abstractions of the image information. In other words, the network embodies different facets of the image, with their own attributes, into its subnetworks and makes its decision based on their vote, as depicted in ***Fig.25.***

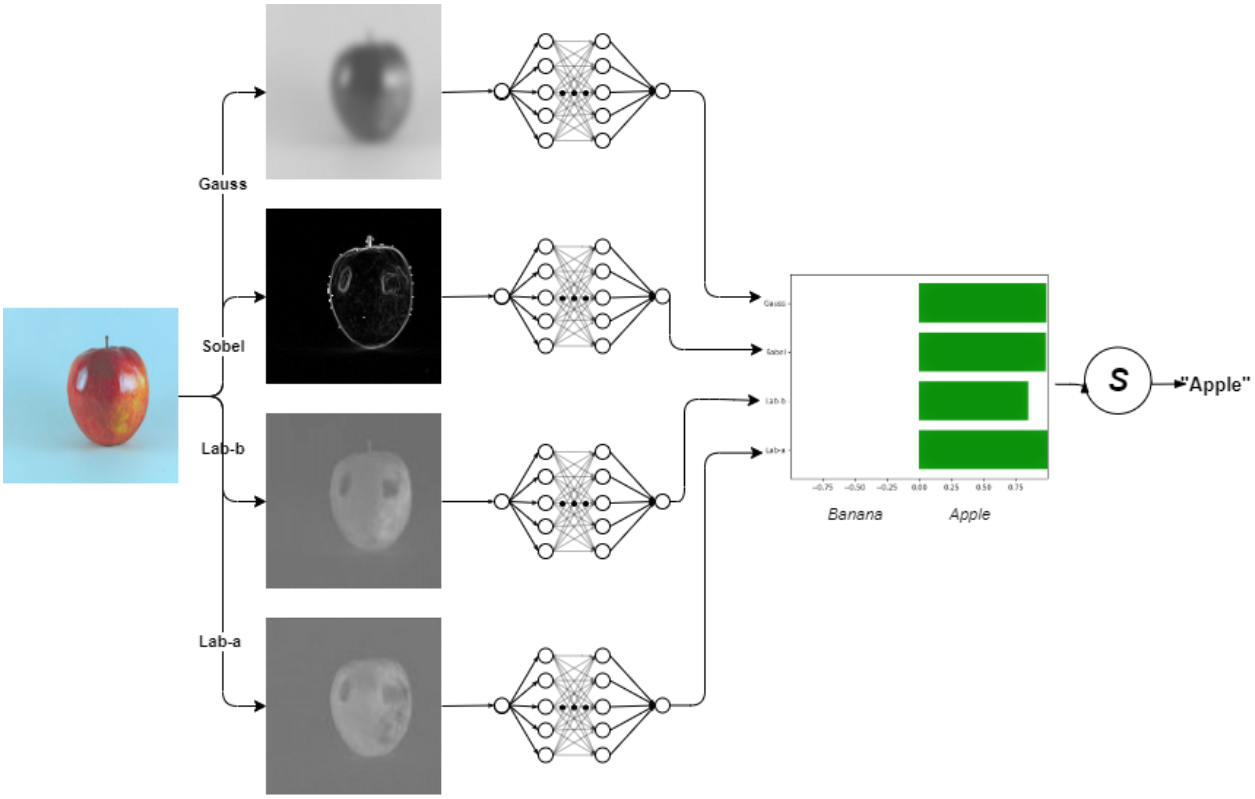


Figure 25**:** *ConGAM Framework. Input image of an apple and its different representations conveyed to the individual subnetworks. 'Gauss' depicts the gaussian filtered image, 'Sobel' the image the Sobel edge detector is applied, 'Lab-b' and 'Lab-a' correspond to the b* and a* channel representations of CIELAB color space. A response within [-1, 1] is obtained from each subnetwork and a sigmoid activation function is applied to bring it to the probability range [0,1] to form the final prediction. The bar-plot brings the verdict to illustrative terms.*

Descriptors:

We define the various representations of the image with perceptually significant information as content Descriptors, they may refer to color, shape, texture, regularity, distribution of lightness etc. Descriptors approximate a basic image representation to the most possible extent with the aim of encoding how humans perceive information in images. Different descriptors are retrieved from the input image and passed to the individual networks, based on their relevance to the considered application. For example, in the task of discerning a pathogenic area in an endoscopic image, its color, irregular texture or shape can be indicative aspects of information towards a diagnosis.

Notably, the effect of descriptors should be restrained in their corresponding network, to prevent implicit interactions among them. This property automatically simplifies the interpretation process since the response of each network is explicitly associated with the corresponding descriptor, thus preventing the costly disentanglement process of overlapping correlations among the networks. The model's transparency is therefore enhanced, as the ineffective descriptors can be evident by their neutral responses and consequently be removed to help the model generalize better.

Primitive Attributes:

In order for the model to capture the essence of descriptors, each of them should be a discrete representation of one or more primitive attributes that are learned from the corresponding subnetwork, for example the attributes of a descriptor of shape can be discontinuities in orientation (curves) and intensity (edges).

Generally, the model can be described as an ensemble of basic content descriptors that encode one or more primitive attributes. Therefore, to interpret the final prediction we consider the separate contribution scores of each descriptor, indicating the degree to which its attributes are present. This task is simplified with the use of the hyperbolic tangent activation function on the final layer of each subnet, by setting its responses in the range of $[-1, 1]$.

5 Results

In this section we give a description of the experiments we conducted to verify the effectiveness of our method on 3 image classification datasets. We validate the proposed ConGAM model in the framework of binary classification and then demonstrate its interpretation.

5.1 Benchmark Datasets

'Bapple':

This dataset was created to demonstrate ConGAM's interpretability, on account of the intuitively straightforward distinction between the apple and banana class. It is comprised of images from flickr.com derived with the use of separate tags: "apple" and "banana" that resulted in 3730 and 4041 images respectively. Irrelevant, non-fruit images were removed, with 2313 remaining from which 1211 belong to the "apple" class and 1102 to "banana" class, as shown in ***Table 3.*** Approximately 10% of the training set was withheld for testing. Variation within the dataset exists in regard to fruit multiplicity, placement (not necessarily centered), size and background.

Table 3: *'Bapple' Dataset overview. Includes number of images in train, augmented and test sets for each class.*

Class	Set		
	Train	Augmented Train	Test
Apple	1096	3288	115
Banana	987	2961	115
Total	2083	6249	230

Endoscopic:

The endoscopic datasets utilized for evaluation in this section share the goal of classifying images between the normal and abnormal class. The 'MICCAI' dataset is derived from a sub-challenge*(https://endovissub-abnormal.grand-challenge.org/)* held in MICCAI 2015 as part of the Endovis challenge(Navab, et al. 2015) and is comprised of gastroscopic images of normal and abnormal findings. The dataset contains 698 images in total, 309 of which belong in the normal class and 389 in the abnormal class. The original resolution of images before downscaling was 489x409 pixels. We report a more detailed view of the dataset in ***Table 4***. 15% of the training set was withheld for testing.

***Table 3**: MICCAI Dataset overview. Includes number of images in train, augmented and test sets for each class..*

Class	Set		
	Train	Augmented Train	Test
Normal	263	789	46
Abnormal	331	993	58
Total	594	1782	104

The KID dataset contains annotated wireless capsule endoscopy (WCE) images and is provided by an open-access database in (Koulaouzidis, et al 2015). For our task, a total of 981 images were used comprised of 530 images in the abnormal class and 451 images in the normal class. Specifically, the abnormal class includes 227 inflammatory and 303 vascular pathogenic findings and the normal class includes 282 esophagus and 169 colon non-pathogenic findings. The original resolution of images before downscaling was 360x360 pixels. We report a more detailed view of the dataset in ***Table 5.*** Approximately 10% of the training set was withheld for testing.

***Table 4**: KID Dataset overview. Includes number of images in train, augmented and test sets for each subgroup of each class.*

Class		Set		
		Train	Augmented Train	Test
Normal				-
	Colon	152	456	17
	Esophagus	253	759	29
Abnormal				-
	Inflammatory	204	612	23
	Vascular	272	816	31
	Total	881	2643	100

In all datasets augmentation was performed by horizontal and vertical flipping.

5.2 Implementation details

The reported results were generated under the following implementation details. To ensure fairness the same data augmentation and scaling procedures are applied to all datasets used for evaluation. The trainable parameters for each subnetwork are updated by the stochastic gradient descent (SGD) optimizer with a batch size of 64 and the learning rate set to the default of 0.01. To prevent overfitting the number of training iterations is determined by the process of early stopping triggered by the loss between true and predicted labels, which is quantified by binary cross entropy (BCE).

To evaluate our method, we perform 5-fold cross validation with 20% split between training and validation sets, while preserving a balanced percentage of samples between classes. The test set is randomly withheld beforehand and remains constant for all folds. The area under the curve (AUC) is set as the measure of performance. We implemented all experiments in the Python environment using the TensorFlow 2.3 platform.

5.3 Quantitative Results

In this section we report the retrieved performance results in terms of the AUC metric, regarding our two proposed architectures, ConGAM-base and ConGAM-Inc, on all datasets utilized for evaluation.

'Bapple' Dataset: Both of the proposed architectures, ConGAM-Inc and ConGAM-base were evaluated on the 'Bapple' dataset via 5-fold cross validation as shown in ***Table 6*** and their obtained results are briefly described as follows: ConGAM-Inc model reaches 92% ± 0.006 AUC and ConGAM-base achieves 91% ± 0.008 AUC.

Table 6*: Individual folds of cross-validation evaluated on the 'Bapple' dataset.*

Folds	Model	
	ConGAM-base	ConGAM-Inc
1	0.92	0.92
2	0.93	0.93
3	0.92	0.91
4	0.91	0.92
5	0.91	0.91

Endoscopic Datasets:

The performance of our proposed architectures in the context of endoscopic datasets will also be reported in terms of the AUC metric. Regarding the KID dataset of wireless capsule endoscopic images we report the performance results as follows: ConGAM-Inc model reaches 94% ± 0.02 AUC and ConGAM-base achieves a 92% ± 0.02 AUC. The obtained results of the MICCAI challenge dataset of gastroscopic images are briefly mentioned as: ConGAM-Inc model reaches 97.3% ± 0.01 AUC and ConGAM-base achieves a 97% ± 0.01 AUC. We report the performance each fold achieved for KID and MICCAI datasets in ***Table 7*** and ***Table 8*** respectively.

***Table 7**: Individual folds of cross-validation on the KID dataset. The ConGAM-Inc* model has does not include the image representations of abstract luminosity, and thus has 3 subnetworks. It was removed after we observed its overall unimpactful role toward a prediction.*

Folds	Model		
	ConGAM-base	ConGAM-Inc	ConGAM-Inc*
mean	0.92	0.92	0.94
1	0.94	0.93	0.9
2	0.89	0.85	0.95
3	0.94	0.93	0.95
4	0.91	0.91	0.97
5	0.91	0.97	0.92

***Table 8**: Individual folds of cross-validation on the MICCAI dataset.*

Folds	Model	
	ConGAM-base	ConGAM-Inc
1	0.97	0.98
2	0.98	0.97
3	0.96	0.98
4	0.95	0.96
5	0.98	0.97

Benchmark datasets results:

In this section we state the performance results in terms of the AUC metric, obtained in the context of various benchmark CNN architectures, on all datasets utilized for evaluation. The network architectures include VGG16, ResNet50 and DenseNet169. Notably, these benchmark models were trained from scratch not utilizing their pretrained weights formulated based on ImageNet, which is a dataset of over 1 million images categorized in 1000 classes and VGG16, ResNet50 and DenseNet169 were originally trained by utilizing this dataset. VGG16 (Simonyan, et al 2014) is a convolutional neural network of layer depth equal to 16 which by default expects images of input size 224x224 pixels. ResNet50 (He, et al 2016) is a convolutional neural network with a depth of 50 layers and its default input images have dimensions of 224x224 pixels. DenseNet169 (Huang, et al 2017) is a convolutional neural network that is 169 layers deep and its default image input size is 224x224 pixels. To ensure fairness all these models were trained with the same input sizes of images as our proposed architectures, which is 128x128 pixels.

The performance results of the mentioned benchmark models trained on the 'Bapple' dataset are briefly mentioned as follows: VGG16 achieves 88% ± 0.003 AUC, ResNet50 reaches 87% ± 0.07 AUC and from DenseNet169 we obtain 89% ± 0.06 AUC.

Regarding the endoscopic datasets, KID and MICCAI, the acquired performance results of the baseline CNN models are shortly reported in the following statements. The metrics for the KID dataset are reported as: VGG16 attained a 90% ± 0.03 AUC, ResNet50 achieved a 92% ± 0.03 AUC and DenseNet169 reached a 94% ± 0.02 AUC. The metrics for the MICCAI challenge dataset shortly are: VGG16 reached a 92% ± 0.01 AUC, ResNet50 achieved a 88% ± 0.1 AUC and DenseNet169 reached a 90% ± 0.12 AUC.

Comparison of benchmark models with ConGAM-Inc:

With all the obtained results as grounding, we proceeded to the quantitative comparison of the proposed ConGAM-Inc model against the benchmark CNN architectures namely of VGG16, ResNet50 and DenseNet169.

From the performance results we gathered regarding the 'Bapple' dataset we can observe that the proposed model is maintaining a superior predictive performance in terms of AUC in comparison with all other models. In a brief statement ConGAM reaches 92% AUC, against DenseNet169(89.3%), VGG16(88.4%) and ResNet50(86.9%) and notably seems more stable by observing the reported measures of standard deviation as 0.006 against DenseNet169(0.06), VGG16(0.034) and ResNet50(0.066) respectively. The evaluation results of ConGAM-Inc and the corresponding baseline models on the 'Bapple' dataset are reported in ***Table 9***.

Table 9*: Performance results on the 'Bapple' dataset based on the 5-fold cross validation. AUC metric ± standard deviation is displayed in each cell. Each row depicts the different model utilized for evaluation.*

Model	Bapple
VGG	0.88±0.003
ResNet50	0.87±0.069
DenseNet169	0.89±0.066
ConGAM-Inc	**0.92±0.006**
ConGAM-base	0.91±0.008

We make the following observations based on the comparison of our model with the collection of the benchmark CNN architectures utilized previously regarding the endoscopic datasets used for evaluation, KID and MICCAI.

We state that our model attains superior performance by at least 5% against all other models in the evaluation of the endoscopic MICCAI challenge dataset. Shortly, ConGAM-Inc achieves a 97.3% AUC compared to VGG16 (92.5%), ResNet50 (87.7%) and DenseNet169 (90%), with a standard deviation of 0.01 which is on par with VGG16 (0.011) and far smaller than the reported in ResNet50 (0.11) and DenseNet169 (0.12), which indicates higher stability over these two networks.

We further report that our model shows advantageous performance against the rest of the CNN architectures utilized for comparison, in the context of the endoscopic KID dataset. We briefly state that ConGAM-Inc reaches a 94% AUC superior over VGG16(90%) and ResNet50(92.3%) and remains on par with DenseNet169(94%). Notably, the reported standard deviations indicate higher stability of our model, reported as ConGAM-Inc (0.02) versus VGG16(0.035), ResNet50(0.027) and DenseNet169(0.05). The evaluation results of ConGAM and the corresponding baseline models on the endoscopic datasets are reported in ***Table 10***. ConGAM-base is also presented in the table, even though not mentioned in the comparative results.

Table 10: *Performance results on the endoscopic datasets, KID and MICCAI, based on the 5- fold cross validation. AUC metric ± standard deviation is displayed in each cell. Each row depicts the different model utilized for evaluation.*

Model	Dataset	
	KID	MICCAI
VGG	0.90±0.03	0.92±0.01
ResNet50	0.92±0.03	0.88±0.11
DenseNet169	0.94±0.05	0.90±0.12
ConGAM-Inc	**0.94±0.02**	**0.973±0.01**
ConGAM-base	0.92±0.02	0.97±0.01

The acquired results indicate that the proposed model maintains superior performance against its non-interpretable comparators, across all datasets used for evaluation. Seemingly, ConGAM-Inc without sacrificing predictive performance, can enhance transparency by providing insights behind the predictions it generates.

5.4 Interpretable representations

The inherently interpretable mechanism of the network is digested down to the form of visual graphs, which carry their own interpretable meanings. The contribution scores produced by each descriptor show how strongly the learned attributes push the prediction to its verdict. Specifically, each figure contains horizontal bars that indicate the impact of each feature on the prediction. The lower the impact the closer to zero the bar appears. Color and direction are indicative of the class. Green bars indicate a feature pushing toward the positive class, whereas red bars indicate a feature pushing toward the negative class, each on opposite sides of the vertical y-axis. In this context both global and local plots can be generated, thus increasing the model's transparency.

Local plots contain the raw scores of each feature for a given image, whereas global plots provide an overview of feature importance for a given class. The individual global bars illustrate the average mean difference between the right and wrong predictions, accompanied with their standard deviation as a black horizontal line. In more intuitive terms, when a feature 'pushes' a prediction to the wrong direction its overall impact is diminished whereas, if the 'push' is in alignment with the true label the feature's impact is enhanced.

The plots we demonstrate in this section all contain the selected descriptors we mentioned in our methodology and their labels are assigned as follows: 'Gauss' refers to the descriptors that encode the information of abstract luminosity, 'Sobel' to the learned patterns that capture shape and 'b-Lab and 'a-Lab' to the representations of images learned with the color ranges of blue-to-yellow and green-to-red respectively, as encoded in the CIELAB color space.

'Bapple':

The global-scale interpretable representations of the 'Bapple' dataset indicate how effectively the selected descriptors of color, shape or abstract luminosity convey their information to nudge the prediction toward the banana or the apple class. The following depictions were generated from the best-performing model in the set of ConGAM-Inc models trained with kfold cross-validation.

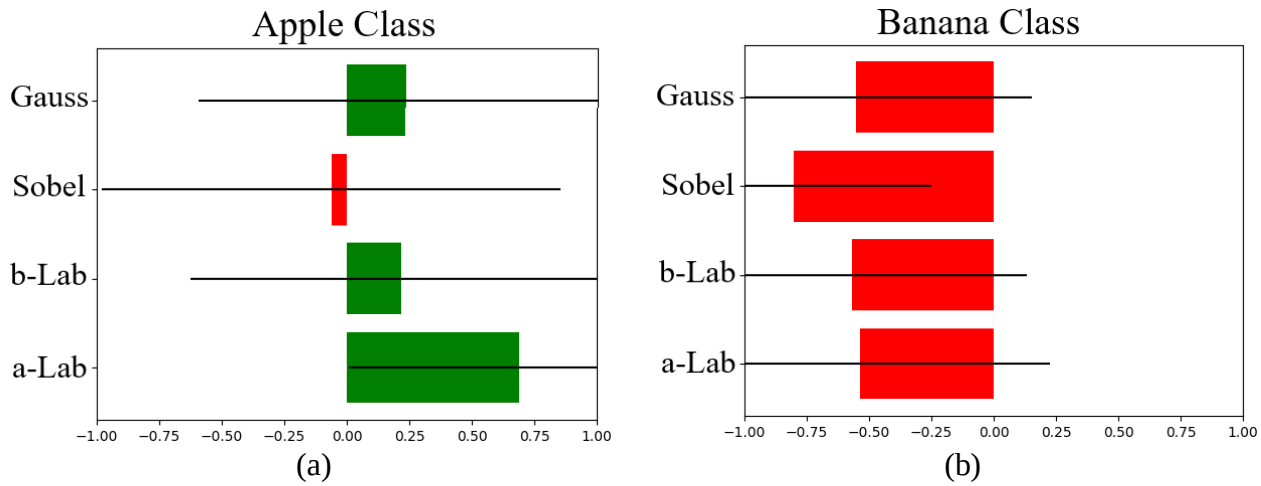


Figure 26: *Global Interpretable Representations of the 'Bapple' dataset from ConGAM-Inc model. (a) depicts global feature importance for the apple class and (b) for the banana class.*

We state our observations ***(Fig.26)*** as follows:

- Regarding the descriptor that encodes the abstract luminosity distribution of images, we can observe that its discriminative power is more prominent in the banana class, possibly because bananas carry more reflective colors in a consistent manner, unlike apples.
- We also observe that the influence of the shape descriptor is primarily determinative in the banana class, whereas it appears counterintuitive in the apple class. This could indicate that the image representations of shape that belong in the apple class have too much variation for the network to extract discriminative motifs from them, in contrast with the images in the banana class. This could be due to the more consistent shape of bananas in the 'Bapple' dataset.
- In both classes a strong indicator toward the true label comes from the color descriptor of the a-channel of CIE-Lab. This indicates that the distribution of color within the range of green to red, as encoded by the a-channel, is highly determinative of the final decision of our model. This is reasonable in the sense that it is easier for the model to classify images based on the presence or absence of red as it primarily portrays apples rather than bananas, at least in a more discriminative manner than other colors do.
- A consequent observation can be made regarding the influence of the descriptor encoding the b-channel of CIE-Lab, as it seems to take on a more impactful role in the guidance of predictions toward the banana class, rather than the apple class where its effect is lessened. This makes sense, as the colors that exist within its range, such as yellow, are more likely to appear in images that contain bananas. Also, note how the overall impact of the 'a-Lab' bar surpasses that of 'b-Lab', which we presume happens because the colors in the blue-to-yellow range appear with greater overlap in both classes, therefore they have less distinctive power.

Here we present the global interpretable bar-plots and our inference of the 2nd proposed subnet architecture, ConGAM-base on the 'Bapple' dataset (***Fig.27***).

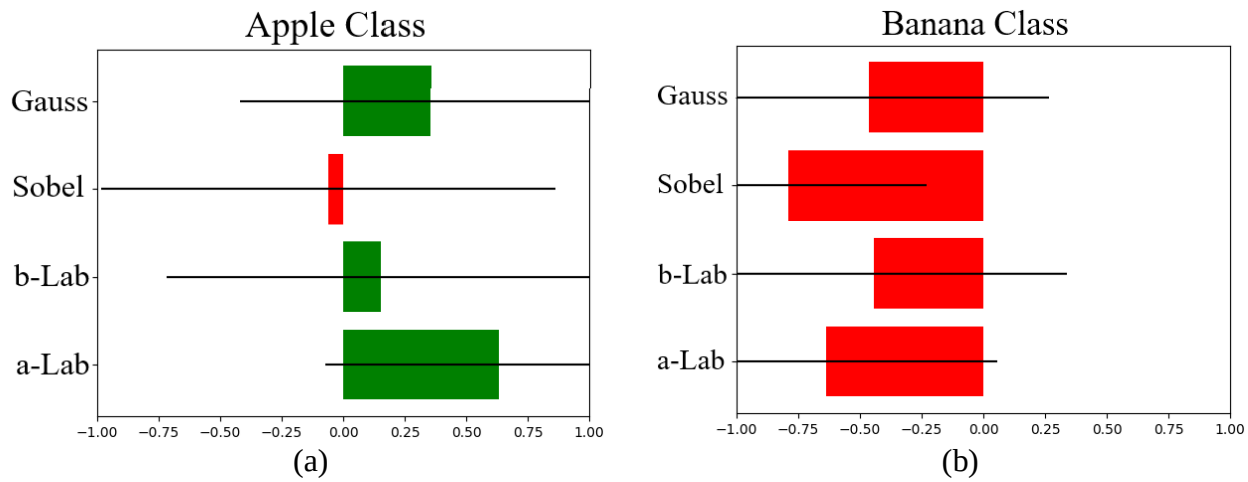
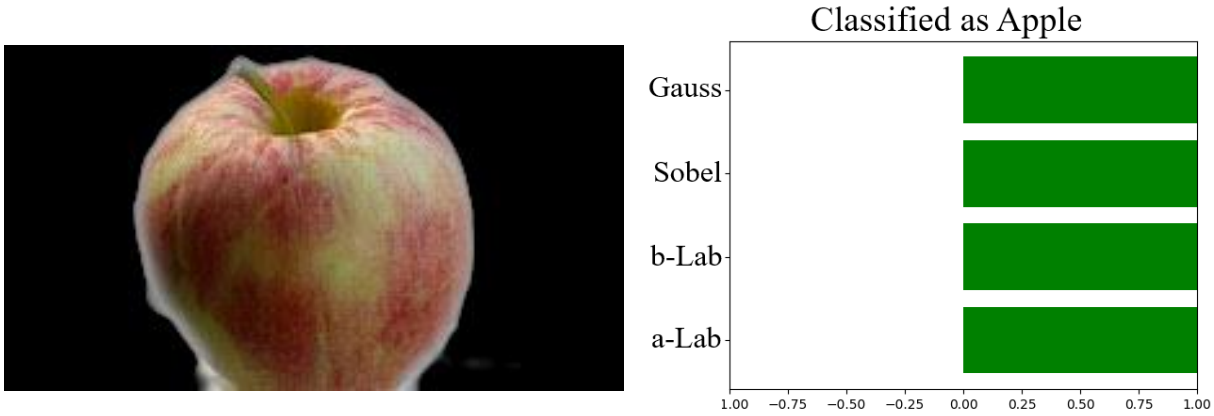


Figure 27: *Global Interpretable Representations of the 'Bapple' dataset from ConGAM-base model. (a) depicts global feature importance for the apple class and (b) for the banana class.*

We can observe that the two proposed architectures are quite similar in terms of feature importance ranking and stability. We note a differentiating point regarding the impact of the feature carrying the 'b-channel' related information, as in ConGAM-base it seemingly assumes a less influential role than it does in ConGAM-Inc. Nevertheless, the overall similarity of how the equivalent descriptors in each model guide the predictions enhances our trust toward the provided interpretations, as it detaches them from the utilized architecture. This counters the limitation inherently interpretable models often face due to their model dependency, although our observation requires further testing with different model architectures and datasets.

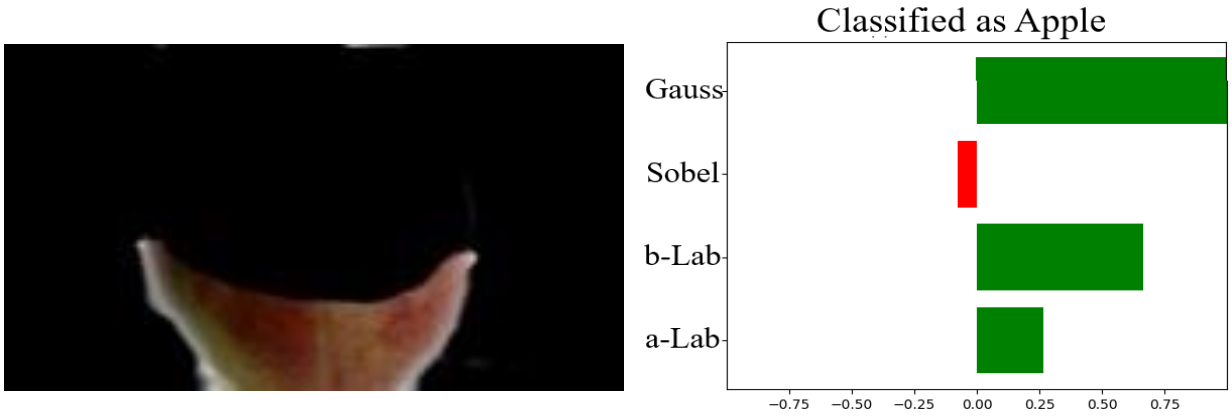
Based on the 'Bapple' dataset, we created a synthetic set of images with the aim of further validating ConGAM's interpretability by emulating the shape and color information of apples and bananas captured by the selected descriptors. To generate the following local interpretable representations we employed the same model in accordance with the global plots illustrated in ***Fig.26.***

In the following example our objective was to alter a unanimous apple classification of the model to that of the opposite class. In this process of 'turning' an apple into a banana, we gradually extracted the parts which from our perspective characterize it as an apple to see if the features' responses would change relatively to our intention.



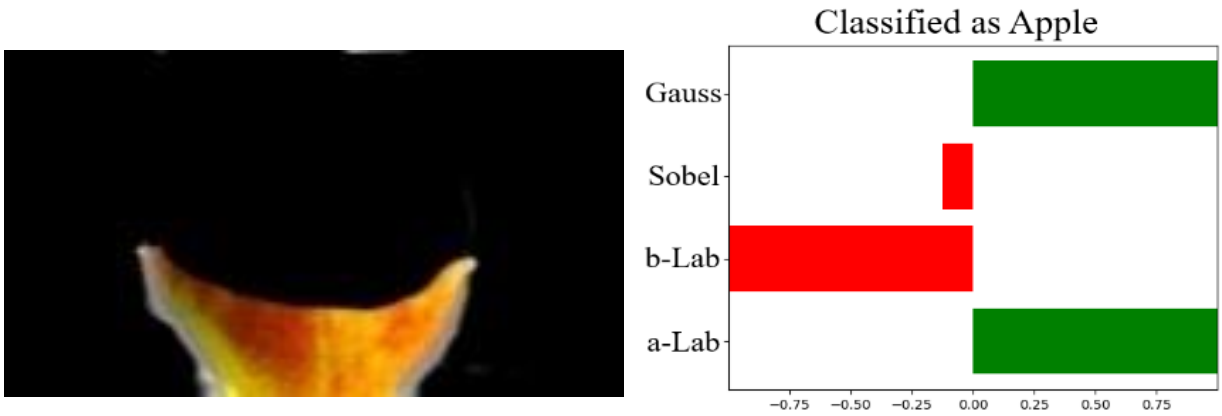
(a) (b)

Figure 28: *(a) shows the original image we used as our baseline and (b) its corresponding bar-plot of feature importance.*



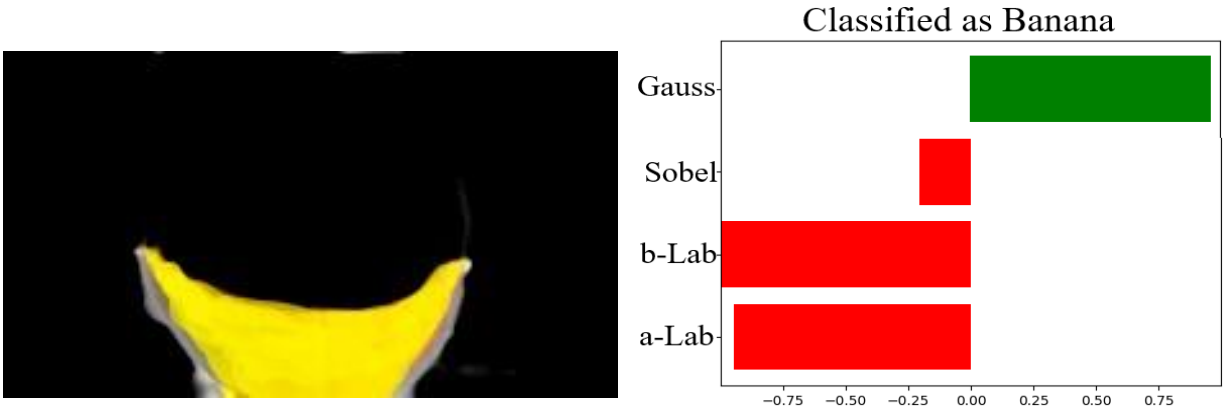
(a) (b)

Figure 29: *(a) depicts how we extracted the top part of the apple from the original image to simulate a 'banana-shaped apple' and (b) shows the obtained visual responses.*



(a) (b)

Figure 30: *(a) shows the enhancement of red and yellow color tones of the object based on the previous image(**Fig.29**) and (b) is the derived visual representation.*



(a) (b)

Figure 31: *Indicates the added distinctive yellow color of a banana to the object and (b) its corresponding feature scores.*

We present the original image in ***Fig.28(a)*** and note how all of its features seem to push the prediction toward the apple class with high confidence. We then observe the effect of the removal of the apple's top part as shown in ***Fig.29(a)*** on the response of the relative shape descriptor*('Sobel')*, which evidently has changed as expected, currently pushing the prediction toward the direction of the banana class.

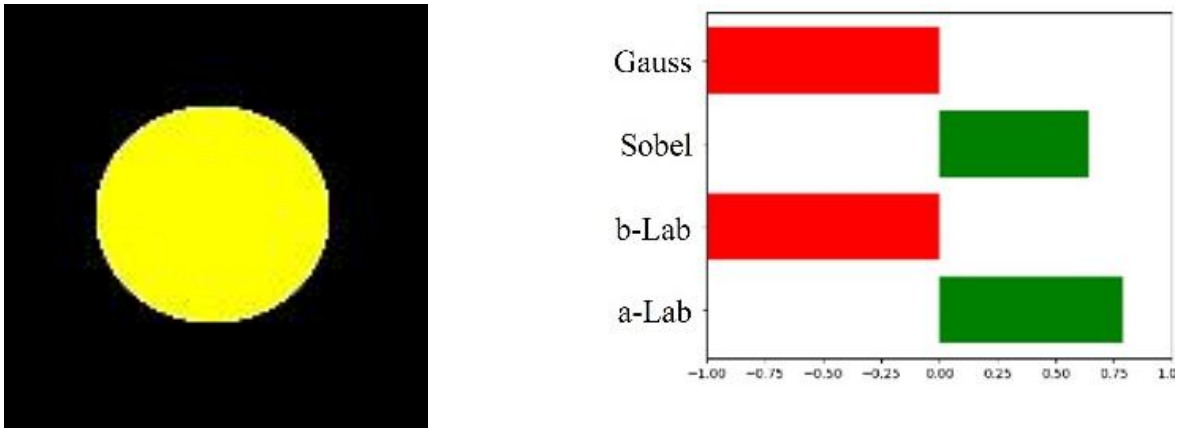
The next step involves the enhancement of red and yellow color tones of the object as depicted in ***Fig.30(a)***. This resulted in the alteration of the relative response of the descriptor conveying the image representation of color within the range of blue to yellow *('b-Lab')*, which is now nudging the prediction toward the banana class. Note how the response of the feature encoding the a-Lab channel image representation is in accordance with our hypothesis that suggests the prominent influence of red color presence.

The last modification includes the addition of the distinctive yellow color of a banana throughout the object to diminish red tones as depicted *in* ***Fig.31(a)***. Seemingly, the descriptor encoding the color information within the green to red range *('a-Lab')* is now guiding the prediction toward the banana class. Ultimately, the predicted label stands as we hoped, classifying this image as a banana, as illustrated by the corresponding bar-plot in ***Fig.31(b).***

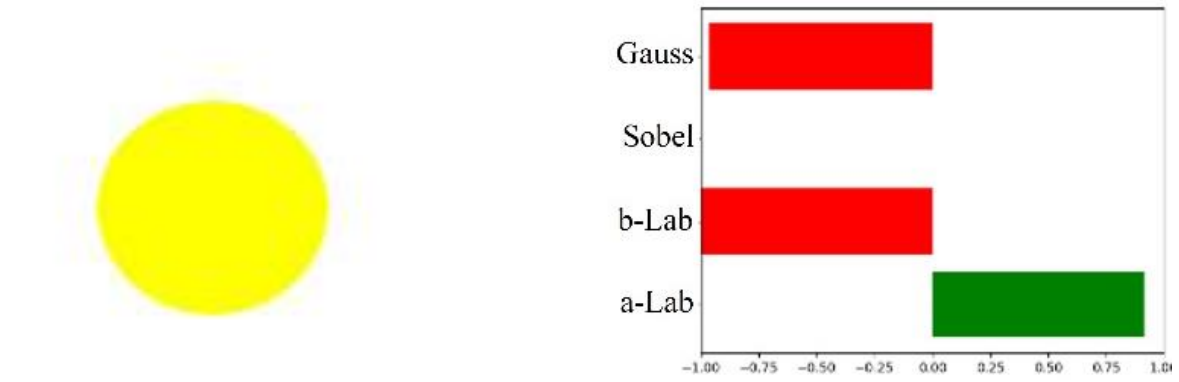
Our purpose of converting the original apple to a banana was achieved successfully but most importantly in a manner that made sense, as the descriptors responses were in alignment with our intuition after each modification of the original image.

We further generated images with primitive attributes to capture the direct responses of the descriptors, in a more controlled manner. We display some of these cases in the following examples and how we interpret them.

(a)



(b)



(c)

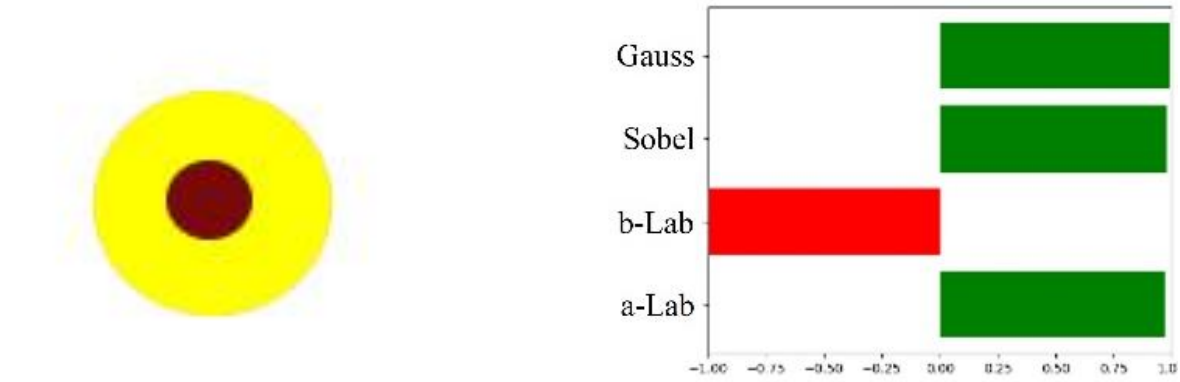
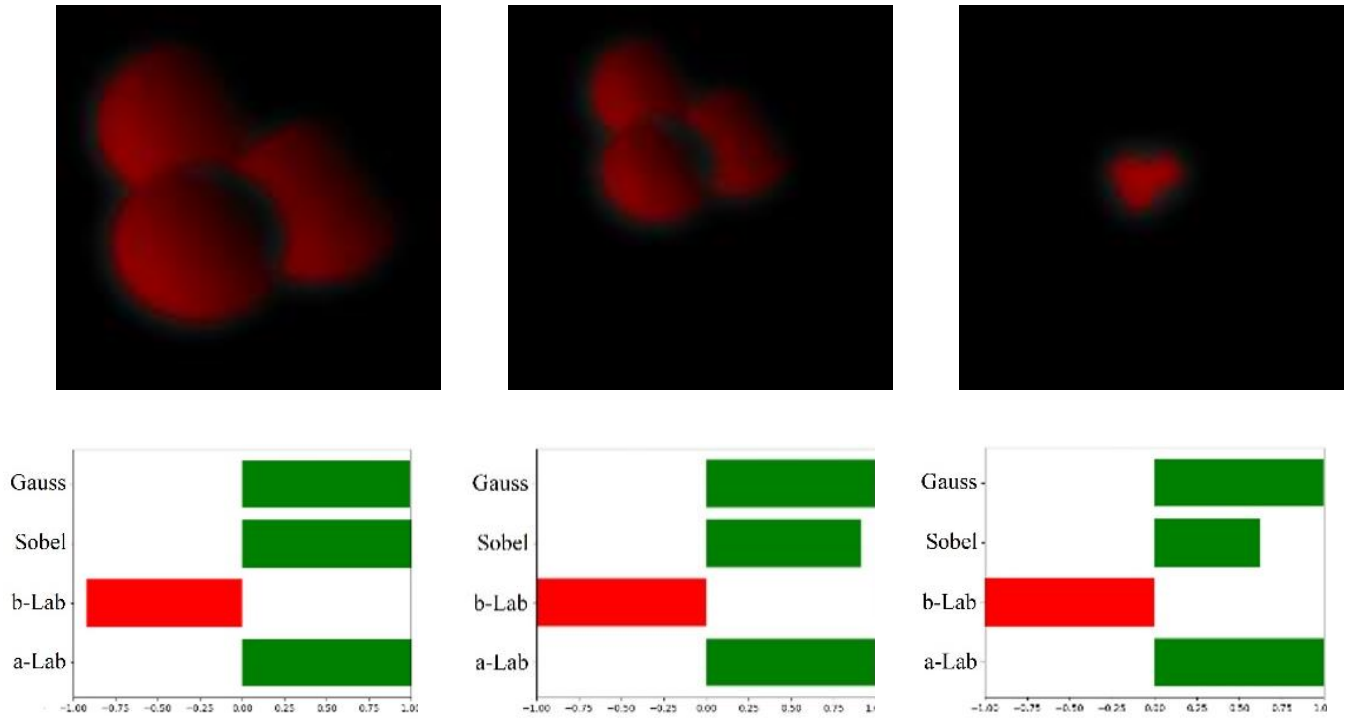


Figure 32: *Synthetic images example. First column depicts the synthetic images and the 2nd column their corresponding bar-plots. We observe the effect of 'Sobel' between (a) and (b) and the effect of 'Gauss' between (b) and (c), regarding shape and abstract luminosity information respectively.*

Between synthetic image ***32.(a)*** and ***32.(b)*** we observe the response of the shape descriptor, which encloses the representation of the image as captured by the Sobel edge detector. As edges depict sharp variations in intensity, the depicted responses are reasonable since in ***32.(b)*** the distinction of the object from the background is harder than ***32.(a).*** With the aim of affecting the luminosity distribution and thus the response of the relative descriptor, we added a smaller brown circle within the yellow one as depicted in synthetic image ***32.(c)***. As shown in the respective bar-plot, the response was faithful to our intention whilst confirming our hypothesis of this descriptor's relation to brightness and how objects that absorb more light are more likely to be considered as apples.

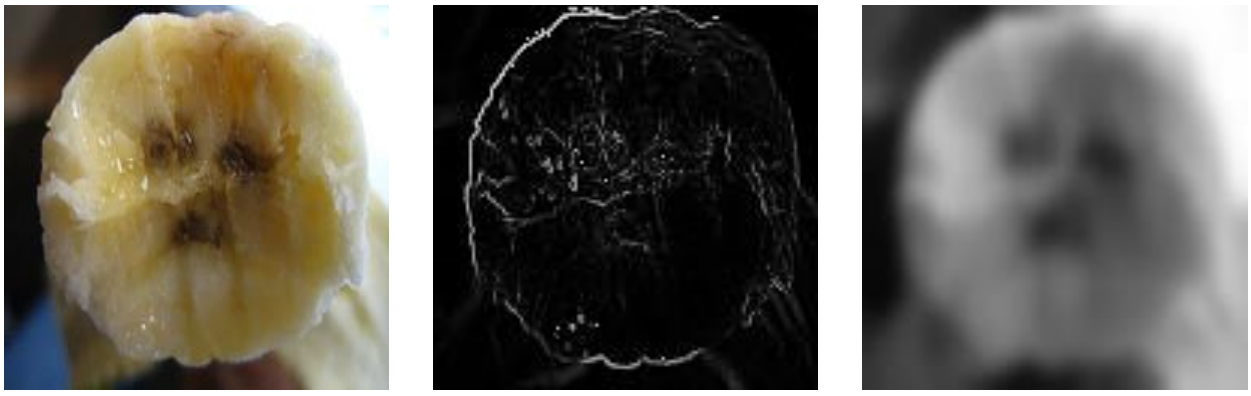


(a) (b) (c)

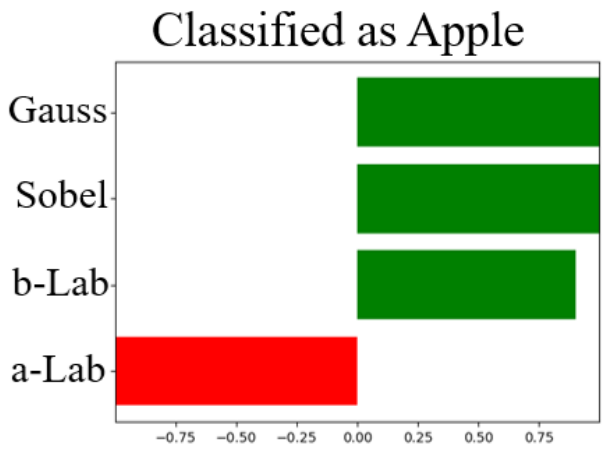
Figure 33: *Synthetic images example. Variation in the size of the object does not seem to majorly impact feature responses.*

In the above example(***Fig.33***) we show how the responses of descriptors are quite similar when changing the size of the objects within the image, further indicating the consistency of our model in this context.

An obvious use of the interpretation mechanism is to understand why the model made a wrong prediction, in this case which descriptors led the final verdict away from the true label. This can be evident by just observing their responses, but can we absolve our model in the cases we find meaning behind its wrong decisions? Below we illustrate such an example.



(a) (b) (c)



(d)

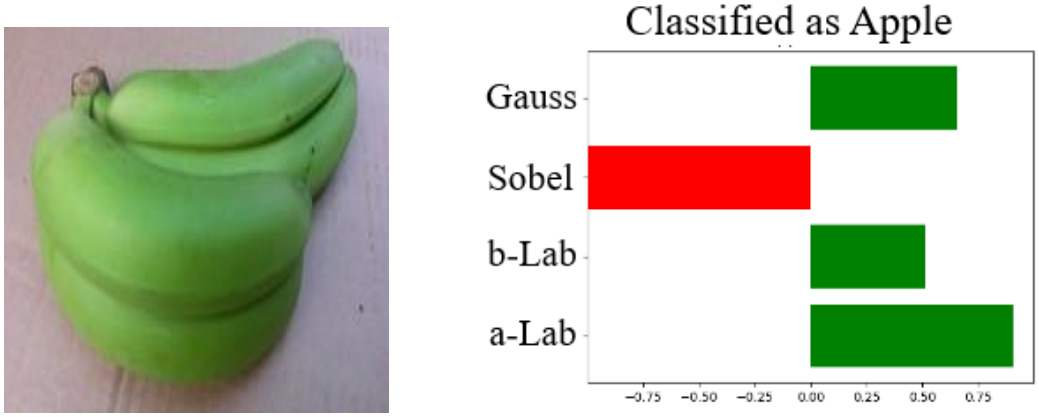
Figure 34: *Wrongly Predicted image. (a) depicts the original image, (b) the filtered image by the Sobel edge detector and (c) the Gaussian filtered image. The individual feature responses are illustrated in (d).*

An initial observation that is in alignment with why we would mistakenly regard this image as an apple concerns its circular shape. Seemingly, the corresponding descriptor of shape information as depicted in ***Fig.34.(b)***, is 'fooled' in a similar manner and therefore nudges the prediction toward the apple class. Subsequently, we observe the response of the 'Gauss' descriptor that encodes the abstract luminosity information as depicted in ***Fig.34.(c)*** and we notice how that the light within the image appears to be absorbed in the center of the object which more likely makes it an apple as we have previously hypothesized*. Lastly, we attribute the wrongful impact of the descriptor that encodes the gamut of color within the b-channel of CIE-Lab, to the more brownish shades of the object. although we make that assumption with less confidence.

*Note how this example is strongly related to our synthetic image example in ***Fig.32.(c)*** where inserting a light-absorbing part within the object pushes the corresponding descriptor's response toward the apple class, proving our model's consistency in this context.

In the following example we demonstrate how we 'turned' a wrong prediction of our model to be correct, by changing the image to look more like a banana, in a way that makes sense to us. Specifically, we altered the color of the object to yellow, to resemble more that of bananas as illustrated in ***Fig.35.*** along with their corresponding bar-plots.

(a)



(b)

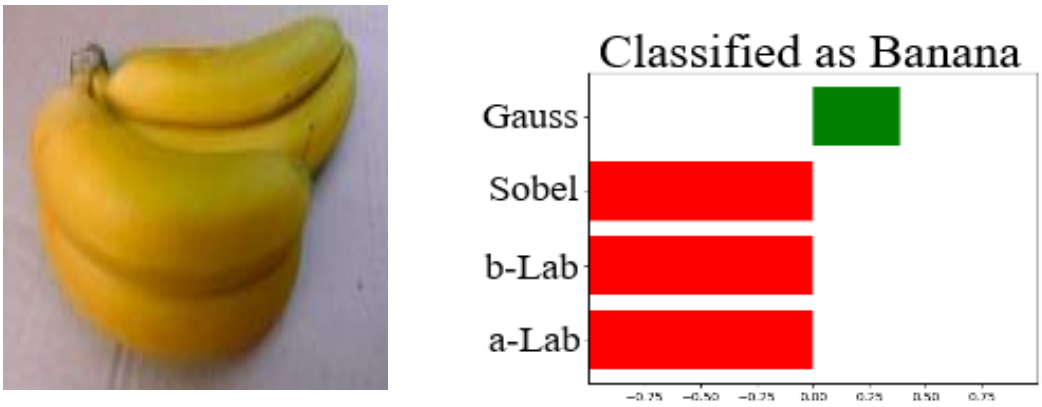


Figure 35: *Turning a wrong prediction to a correct one. (a) depicts the original image and (b) the processed image and their corresponding bar-plots.*

In the original image ***Fig.35(***a) we notice how the shape descriptor pushes the prediction toward the true label of a banana, while the rest of the features seem to nudge the outcome toward the apple class into forming the final verdict of our model. In the processed image we notice how the shape descriptor remains constant, while the color descriptors change their direction and effect toward the banana class as expected. Also the response of the abstract luminosity descriptor indicates that the processed image looks more like a banana than the original one, confirming our hypothesis that when objects are more reflective they are more likely to be considered as bananas rather than apples.

In this section we report the global interpretable representations of both endoscopic datasets we used for the evaluation of our model, KID and MICCAI, and state our observations in terms of overall feature importance.

KID Global Interpretations:

We will first examine the global-scale interpretable representations of the 'KID' dataset***(Fig.36)*** to deduct how effectively the selected descriptors of color, shape or abstract luminosity convey their information to guide the prediction toward the abnormal or normal class. The following illustrations were generated from the best-performing model in the set of ConGAM-Inc models trained with kfold cross-validation.

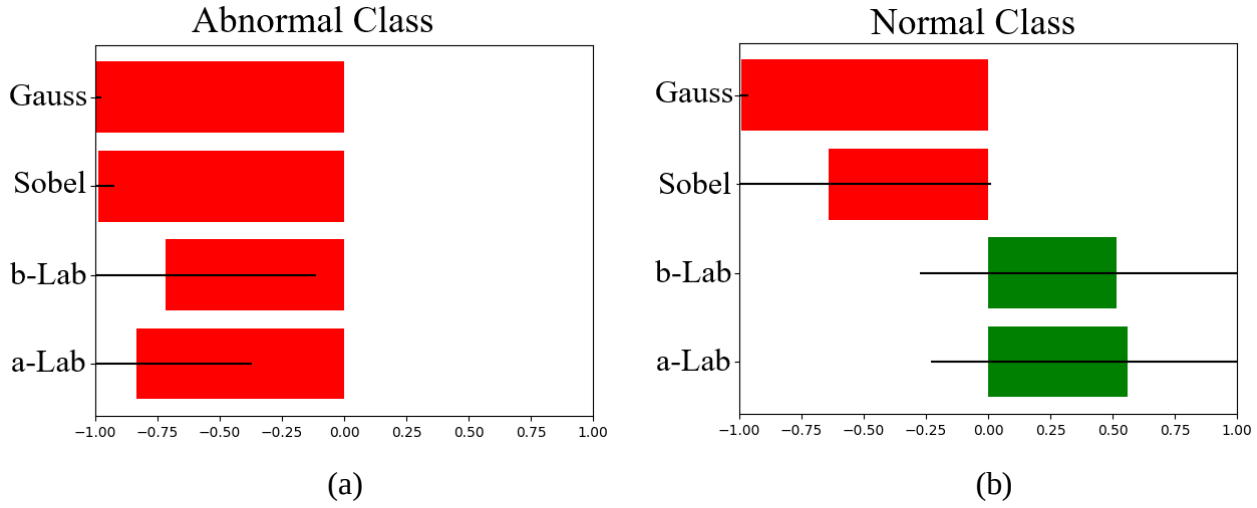


Figure 36: *Global Interpretable Representations of the 'KID' dataset. (a) depicts global feature importance for the abnormal class and (b) for the normal class.*

At a first glance over the global plots, our attention is captured by the trend of both 'Gauss' and 'Sobel' bars, which evidently push the prediction toward the abnormal class regardless of the true label. Hence, we can infer that the motifs extracted by the subnetworks that handle the abstract luminosity and shape information respectively are biased toward the abnormal class.

We then observe the high discriminative power of the feature conveying colors in the green-to-red range encoded as 'a-Lab', which evidently in both classes nudges the prediction with high confidence toward the correct direction. Similarly, the representative descriptor of the gamut of color within the b-channel of CIE-Lab appears to correctly influence the predictions of our model. We can therefore infer that for the task of classifying images between the normal and abnormal class, the color-related image representations convey the most useful information for the model toward the true label.

MICCAI Global Interpretation:

The following observations are based on the depicted global-scale representations of the MICCAI challenge dataset***(Fig.37)*** and describe the impact of the selected descriptors of color, shape or abstract luminosity on the model's prediction toward the abnormal or normal class. The following illustrations were generated from the best-performing model in the set of ConGAM-Inc models trained with kfold cross-validation.

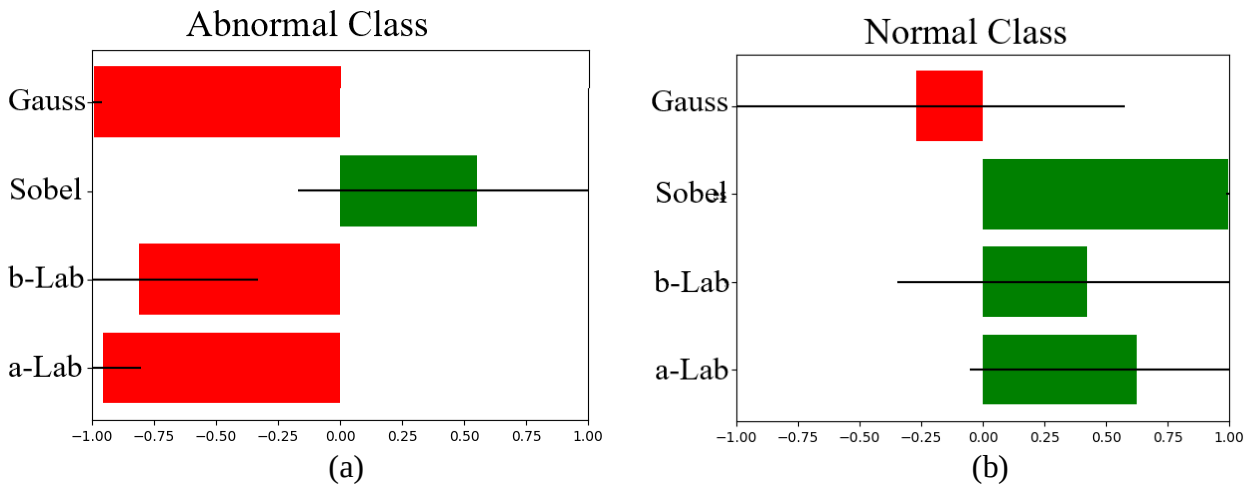


Figure 37: *Global Interpretable Representations of the 'MICCAI' challenge dataset. (a) depicts global feature importance for the abnormal class and (b) for the normal class.*

Similarly with the previous global plots generated from the KID dataset, what captures our attention is the responses of the descriptors that carry the image components of abstract luminosity and shape, namely 'Gauss' and 'Sobel'. Interestingly, their direction of impact remains unchanged regardless of the true label, which indicates that the corresponding subnetworks could not deduct useful information into discriminating both classes.

A subsequent observation is made regarding the color-related image representations the subnetworks learned that seemingly guide the predictions accurately. Specifically, the feature conveying colors in the green-to-red range encoded as 'a-Lab' displays the highest influence into classifying images correctly, whereas 'b-Lab' follows the same trend although with less overall impact.

Notably, the above inferences regarding the endoscopic datasets convey a similar pattern in terms of global feature importance. Specifically, the obtained interpretable representations showcase that the most impactful descriptors of each class toward the true label are color-related, with (Iakovidis, et al. 2014) further confirming the validity of this observation. Moreover, the features that assume an unimpactful and even counterintuitive role in aiding the model's predictions in both datasets, are conveying the abstract luminosity and shape information of images, indicating a biased inclination toward a specific class regardless of the true label.

6 Conclusion and Future Prospects

In this thesis we contradict the apparent conflict between interpretability and performance of deep learning models by presenting a new class of intrinsically interpretable CNN models, referred to as ConGAM. Remaining under the relevant scope of this work, we initially presented a brief overview of the basic principles in the fields of computer vision and deep learning, while highlighting the demand for methods that provide interpretable insights that guide their decisions, leading to the following deductions:

- Data availability and computational advancements facilitate the deployment of deep learning methods in demanding computer vision tasks.
- CNNs ability to capture spatial information to efficiently detect features makes them more suitable for image classification tasks.
- Intelligibility can be enhanced if the network receives information that better approximates human perception.
- Self-explainable models remain highly unexplored compared to methods which depend on external factors to provide interpretations.

With the aim of accommodating the critical need for explanations we discuss the novel work towards devising ConGAM which exploits the intelligibility of GAMs. Hence, it enforces a built-in reasoning process which provides explicit explanations of how the model utilizes its individual features to reach a verdict, for the task of image classification. In correlation with the above inferences we present the following statements which sum up the key points of our method:

- We parametrized the smooth functions of GAMs with CNNs, each of which learns a separate image representation that captures how we intuitively describe objects in images, hence making the subsequent task of interpretation more straightforward.
- The utilized image representations convey information of abstract luminosity, edges and the color distributions that correspond to the a^* and b^* channels of CIELAB.
- We further note the scalability of our framework in terms of the aggregate amount of subnetworks, and thus selected features.
- Designed two CNN architectures which composed the individual subnetworks, with incorporated mechanisms against overfitting.

- Validated our method on 3 binary image classification datasets and reported the experimental results indicating that our framework can maintain advantageous performance against its non-interpretable comparators.
- Without sacrificing predictive power, we showed that our model also offers illustrative interpretable representations, globally and locally, in terms of feature importance.
- Demonstrated how the provided explanations coincided with our intuition on individual predictions and stated our observations regarding the entire datasets. *(e.g* the color descriptors being highly discriminative factors in the endoscopic datasets.*)*

Importantly, our method satisfies the following criteria of interpretability as defined in (Chakraborty, et al 2020) namely, *macro-decomposability* as we can offer intuitive explanations for each subnetwork comprising our model, *visualization* as we illustrate the features' impact and *local explanation* as captured by each single image demonstrated in our examples.

For future work we opt to examine various possible subnetwork architectures and their effect on the provided interpretations, considering how this model's structure is intertwined with its explanations. Also, possibly expand the framework with different types of features and assess how we can model the interactions between them, as well as evaluate it in larger image datasets and further investigate its interpretation capabilities.

Bibliography

Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I., Hardt, M., & Kim, B. (2018). Sanity checks for saliency mapsarXiv preprint arXiv:1810.03292.

Agarwal, R., Frosst, N., Zhang, X., Caruana, R., & Hinton, G. E. (2020). Neural additive models: Interpretable machine learning with neural nets. *arXiv preprint arXiv:2004.13912*.

Alvarez-Melis, D., & Jaakkola, T. S. (2018). Towards robust interpretability with self-explaining neural networks. *arXiv preprint arXiv:1806.07538*.

Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K. R., & Samek, W. (2015). On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, *10*(7), e0130140.

Bansal, S., & Aggarwal, D. (2011). Color image segmentation using CIELab color space using ant colony optimization. *International Journal of Computer Applications*, *29*(9), 28-34.

Bau, D., Zhou, B., Khosla, A., Oliva, A., & Torralba, A. (2017). Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 6541-6549).

Bengio, Y. (2009). *Learning deep architectures for AI*. Now Publishers Inc.

Buja, A., Hastie, T., & Tibshirani, R. (1989). Linear smoothers and additive models. *The Annals of Statistics*, 453-510.

Chakraborty, S., Tomsett, R., Raghavendra, R., Harborne, D., Alzantot, M., Cerutti, F., ... & Gurram, P. (2017, August). Interpretability of deep learning models: a survey of results. In *2017 IEEE smartworld, ubiquitous intelligence & computing, advanced & trusted computed, scalable computing & communications, cloud & big data computing, Internet of people and smart city innovation (smartworld/SCALCOM/UIC/ATC/CBDcom/IOP/SCI)* (pp. 1-6). IEEE.

Chen, J., Vaughan, J., Nair, V., & Sudjianto, A. (2020). Adaptive explainable neural networks (axnns). *Available at SSRN 3569318*.

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., & Abbeel, P. (2016). Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *arXiv preprint arXiv:1606.03657*.

Covert, I., Lundberg, S., & Lee, S. I. (2020). Understanding global feature contributions through additive importance measures. *arXiv preprint arXiv:2004.00668*.

Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009, June). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition* (pp. 248-255). Ieee.

Fong, R. C., & Vedaldi, A. (2017). Interpretable explanations of black boxes by meaningful perturbation. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 3429-3437).

Fong, R., Patrick, M., & Vedaldi, A. (2019). Understanding deep networks via extremal perturbations and smooth masks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 2950-2958).

Friedman, J. H., & Stuetzle, W. (1981). Projection pursuit regression. *Journal of the American statistical Association*, *76*(376), 817-823.

Ganesan, P., Rajini, V., & Rajkumar, R. I. (2010, December). Segmentation and edge detection of color images using CIELAB color space and edge detectors. In *INTERACT-2010* (pp. 393-397). IEEE.

Guo, Y., Su, Y., Yang, Z., & Zhang, A. (2020). Explainable Recommendation Systems by Generalized Additive Models with Manifest and Latent Interactions. *arXiv preprint arXiv:2012.08196*.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., ... & Lerchner, A. (2016). beta-vae: Learning basic visual concepts with a constrained variational framework.

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4700-4708).

Hubel, D. H., & Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, *160*(1), 106-154.

Iakovidis, D. K., & Koulaouzidis, A. (2014, October). Automatic lesion detection in wireless capsule endoscopy—a simple solution for a complex problem. In *2014 IEEE International Conference on Image Processing (ICIP)* (pp. 2236-2240). IEEE.

Ioffe, S., & Szegedy, C. (2015, June). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning* (pp. 448-456). PMLR.

Koulaouzidis, A., & Iakovidis, D. K. (2015). KID: A capsule endoscopy database for medical decision support. *United European Gastroenterology Week (UEGW). Barcelona, Spain*.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, *25*, 1097-1105.

Kuo, C. C. J., Zhang, M., Li, S., Duan, J., & Chen, Y. (2019). Interpretable convolutional neural networks via feedforward design. *Journal of Visual Communication and Image Representation*, *60*, 346-359

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278-2324.

Lundberg, S., & Lee, S. I. (2017). A unified approach to interpreting model predictions. *arXiv preprint arXiv:1705.07874*.

Morgan, N., & Bourlard, H. (1989, January). Generalization and parameter estimation in feedforward nets: Some experiments. In *Proceedings of the 2nd International Conference on Neural Information Processing Systems* (pp. 630-637).

Nair, V., & Hinton, G. E. (2010, January). Rectified linear units improve restricted boltzmann machines. In *Icml*.

Navab, N., Hornegger, J., Wells, W. M., & Frangi, A. (Eds.). (2015). *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III* (Vol. 9351). Springer.

Park, D. H., Hendricks, L. A., Akata, Z., Rohrbach, A., Schiele, B., Darrell, T., & Rohrbach, M. (2018). Multimodal explanations: Justifying decisions and pointing to the evidence. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 8779-8788).

Prechelt, L. (1998). Early stopping-but when?. In *Neural Networks: Tricks of the trade* (pp. 55-69). Springer, Berlin, Heidelberg.

Qin, X., Zhang, Z., Huang, C., Gao, C., Dehghan, M., & Jagersand, M. (2019). Basnet: Boundary-aware salient object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 7479-7489).

Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016, August). " Why should i trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1135-1144).

Robbins, H., & Monro, S. (1951). A stochastic approximation method. *The annals of mathematical statistics*, 400-407.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, *323*(6088), 533-536.

Sabour, S., Frosst, N., & Hinton, G. E. (2017). Dynamic routing between capsules. *arXiv preprint arXiv:1710.09829*.

Scherer, D., Müller, A., & Behnke, S. (2010, September). Evaluation of pooling operations in convolutional architectures for object recognition. In *International conference on artificial neural networks* (pp. 92-101). Springer, Berlin, Heidelberg

Selbst, A. D., & Powles, J. Meaningful Information and the Right to Explanation'(2017). International Data Privacy Law, 7, 233.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision* (pp. 618-626).

Shrikumar, A., Greenside, P., & Kundaje, A. (2017, July). Learning important features through propagating activation differences. In *International Conference on Machine Learning* (pp. 3145-3153). PMLR.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Simonyan, K., Vedaldi, A., & Zisserman, A. (2013). Deep inside convolutional networks: Visualising image classification models and saliency mapsarXiv preprint arXiv:1312.6034.

Smilkov, D., Thorat, N., Kim, B., Viégas, F., & Wattenberg, M. (2017). Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*.

Sobel, I. (1972). Camera models and machine perception (No. CS Technion report CS0016). Computer Science Department, Technion.

Springenberg, J. T., Dosovitskiy, A., Brox, T., & Riedmiller, M. (2014). Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), 1929-1958.

Sundararajan, M., Taly, A., & Yan, Q. (2017, July). Axiomatic attribution for deep networks. In International Conference on Machine Learning (pp. 3319-3328). PMLR.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1-9)

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1-9).

Tsang, M., Liu, H., Purushotham, S., Murali, P., & Liu, Y. (2018, January). Neural Interaction Transparency (NIT): Disentangling Learned Interactions for Improved Interpretability. In *NeurIPS* (pp. 5809-5818).

Vaughan, J., Sudjianto, A., Brahimi, E., Chen, J., & Nair, V. N. (2018). Explainable neural networks based on additive index models. *arXiv preprint arXiv:1806.01933*.

Zebin Yang, Aijun Zhang, & Agus Sudjianto. (2020). GAMI-Net: An Explainable Neural Network based on Generalized Additive Models with Structured Interactions.

Zeiler, M. D., & Fergus, R. (2014, September). Visualizing and understanding convolutional networks. In *European conference on computer vision* (pp. 818-833). Springer, Cham.

Zhang, Q., Wang, X., Wu, Y. N., Zhou, H., & Zhu, S. C. (2019). Interpretable CNNs for Object Classification. *arXiv preprint arXiv:1901.02413*.

Zhang, Q., Yang, Y., Ma, H., & Wu, Y. N. (2019). Interpreting cnns via decision trees. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 6261-6270).

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., & Torralba, A. (2016). Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2921-2929).

Zhou, Y., & Chellappa, R. (1988). Computation of optical flow using a neural network. *IEEE 1988 International Conference on Neural Networks*, 71-78 vol.2.

Zintgraf, L. M., Cohen, T. S., Adel, T., & Welling, M. (2017). Visualizing deep neural network decisions: Prediction difference analysis. *arXiv preprint arXiv:1702.04595*.