Universidad ORT Uruguay
Facultad de Ingeniería

An approach to explainable deep learning for image-based medical diagnosis based on prototypes

Entregado como requisito para la obtención del título de Master en Ingeniería

Leonardo Cecilia - 150606

Tutor: Sergio Yovine

2023

Declaración de autoría

Yo, Leonardo Cecilia declaro que el trabajo que se presenta en esta obra es de mi propia mano. Puedo asegurar que:

- La obra fue producida en su totalidad mientras realizaba el Proyecto Final del Master en Ingeniería (por Investigación)
- Cuando he consultado el trabajo publicado por otros, lo he atribuido con claridad;
- Cuando he citado obras de otros, he indicado las fuentes. Con excepción de estas citas, la obra es enteramente mía;
- En la obra, he acusado recibo de las ayudas recibidas;
- Cuando la obra se basa en trabajo realizado conjuntamente con otros, he explicado claramente qué fue contribuido por otros, y qué fue contribuido por mí;
- Ninguna parte de este trabajo ha sido publicada previamente a su entrega, excepto dónde se han realizado las aclaraciones correspondientes.

Leonardo Cecilia

01-03-2023

Dedicado a mi familia y amigos que me brindaron su apoyo en los años de trabajo en la tesis.

Agradezco especialmente al Dr. Sergio Yovine por su dedicación y paciencia, sin su ayuda este trabajo no habría sido posible.

Abstract Español

El objetivo de este proyecto es desarrollar un modelo de *deep learning* "explicable" para analizar imágenes médicas, al mismo tiempo que se crean herramientas de visualización fáciles de usar que se pueden aprovechar en entornos clínicos para mejorar la atención al paciente. Al utilizar los últimos avances en técnicas de *deep learning*, este modelo tiene como objetivo proporcionar resultados más precisos e interpretables, lo que permite a los médicos tomar decisiones mejor informadas.

Abstract

The goal of this project is to develop an 'explainable' deep machine learning model for analyzing medical images, while also creating user-friendly visualization tools that can be used in clinical settings to enhance patient care. By leveraging the latest advances in deep learning techniques, this model aims to provide more accurate and interpretable results, enabling clinicians to make better-informed decisions.

Keywords

Explainability; Neural Networks; Medical Images; Deep Learning; Machine Learning;

Contents

1 Introduction

In the last two decades the availability of large collections of health data has dramatically increased [1]. In combination with AI, bigdata, and massively parallel computing, it offers the potential to create a revolutionary way of practicing evidence-based, cost-effective, and personalized medicine [2]. Doctors no longer have to rely on their own experience and education to diagnose and treat patients, but rather can access models trained on thousands of cases to make better decisions.

AI techniques can be used in three main areas of care: early disease prediction and diagnosis, treatment, as well as outcome prediction and prognosis evaluation [3].

Leveraging big data techniques in healthcare enhancements brings not only opportunities, but challenges [4]. Besides issues in annotation or labeling data, transparency also relates to model interpretability or explainability. In other words, humans should be able to understand or interpret how a given technology reached a certain conclusion or prediction [5]. This kind of 'transparent' AI contrasts the 'opaque' or 'black box' approach. It is thought that if the system's reasoning can be explained, then humans can verify whether the reasoning is correct. If the system's reasoning cannot be explained, such evaluation would not be practically feasible. However, there are trade-offs in certain instances, enforcing transparency and interpretability can potentially result in decreased accuracy or predictive performance of a model [5].

Healthcare abounds with possible applications where predictions put patients at risk. For instance: predicting patient risk of sepsis (a potentially life threatening response to infection), predicting tumor malignancy from images, predicting a patient's likelihood of readmission to the hospital, and predicting the need for end of life care, just to name a few [6]. Therefore, not only accuracy should be taken into consideration when using these models, but also the capacity of the model to 'explain' how it reached certain prediction.

This work focuses on implementing an 'explainable' model with deep machine learning techniques to provide accurate predictions. The model is trained and tested using a dataset of medical images to evaluate its accuracy and interpretability. The aim is to demonstrate that explainable models can provide interpretable predictions while maintaining a high level of accuracy, making them a suitable option for use in clinical settings.

Simultaneously, a set of visualization tools are developed as part of this work to gain a deeper understanding of the model's internal workings. These visualization tools enable the identification of critical features used by the model for making predictions and provide insight into the decision-making process. This allows clinicians to gain a better understanding of the reasoning behind the predictions, making it easier for them to interpret and trust the model's results. By developing an explainable model with visualization tools, this work aims to contribute to the field of medical image analysis and improve patient care.

2 Context

2.1 Definitions

Before introducing the context of explainable deep learning models applied to health-related domains, a few concepts will be presented, as they are common terms that have specific meaning in the XAI community. The list is based in a compendium made by Barredo et al [7].

Understandability (or equivalently, intelligibility): denotes the characteristic of a model to make a human understand its output, how the model works, without any need for explaining its internal structure or the algorithmic means by which the model processes data internally and reaches each output [8].

Comprehensibility: when conceived for ML models, comprehensibility refers to the ability of a learning algorithm to represent its learned knowledge in a human understandable fashion. The results of computer model should be symbolic descriptions of given entities, semantically and structurally similar to those a human expert might produce observing the same entities. Components of these descriptions should be comprehensible as single 'chunks' of information, directly interpretable unnatural language, and should relate quantitative and qualitative concepts in an integrated fashion. Given its difficult quantification, comprehensibility is normally tied to the evaluation of the model complexity [9].

Interpretability: it is defined as the ability to explain or to provide the meaning in understandable terms to a human [7].

Explainability: explainability is associated with the notion of explanation as an interface between humans and a decision maker that is, at the same time, both an accurate proxy of the decision maker and comprehensible to humans [10].

Transparency: a model is considered to be transparent if by itself it is un-

derstandable. Since a model can feature different degrees of understandability, transparent models are divided into three categories: simulatable models, decomposable models and algorithmically transparent models [11].

In the context of XAI, understandability is the most important concept. Both transparency and interpretability are closely related to this concept. Transparency refers to a model's ability to be understood by humans without additional explanation. Understandability, on the other hand, is the measure of how well humans can comprehend a model's decision. Comprehensibility is also linked to understandability because it depends on the audience's ability to understand the knowledge encoded in the model.

While explainability and interpretability are related, they are not interchangeable. Explainability is concerned with providing a high-level understanding of a model's behavior, while interpretability is concerned with understanding the specific mechanisms that underlie that behavior. Both are important in different contexts, and deep learning researchers and practitioners need to consider both when building and evaluating models.

2.2 Responsible AI

According to Trocin et al. [12], recent studies have shown that AI used in clinical settings has the potential to create unintended consequences, such as biases, discrimination, errors or unexpected results. Therefore, the need arises to create artificial intelligence models that take these risks into account. This is how the field known as Responsible AI was born.

Responsible AI is a governance framework that uses to harness, deploy, evaluate, and monitor AI machines to create new opportunities for better service provision [13]. Its main focus is designing and implementing ethical, transparent, and accountable AI solutions that help maintain individual trust and minimize privacy invasion. Responsible AI understands the implications and risks these techniques carry with them and places humans (e.g., end-users) at the center, without forgetting stakeholder expectations and applicable regulations and laws. Prior to designing and implementing responsible AI, organizations need to understand the practices that will help them drive ethics and trust of AI use.

Mittelstadt [14] proposes a set of nine ethical principles for designing responsible models and devices: 1. Facilitate public health actions and user engagement with research via the model; 2. Non-maleficence and beneficence; 3. Respect

autonomy and avoid subtle nudging of user behaviour; 4. Respect individual privacy; 5. Respect group privacy; 6. Embed inclusiveness and diversity in design; 7. Collect the minimal data required; 8. Establish and maintain trust and confidentiality between model users and providers; 9. Ensure data processing protocols are transparent and accountable.

As stated in the previous definitions and as expressed by Kumar [15], core to Responsible AI is Explainability. Models need to be constructed with not only accuracy as a target, but also the capacity to explain their results.

2.3 Explainabililty in Healthcare

2.3.1 Model-specific vs. Model-agnostic

Chen et al. [16] propose a classification of interpretable models in two categories: model-specific and model-agnostic.

Model-specific or intrinsic interpretability consists in the adoption of models that are explainable by design. The architecture is created in such a fashion that following the internals from input to output is straightforward.

In opposition, model-agnostic methods are created by using a separate model to provide explanations for an existing model. These techniques have the advantage of allowing developers to use any machine learning model they prefer, without having to sacrifice prediction performance in order to provide explanations.

2.3.2 User-Friendly Explanations

User-friendly techniques for explainability in deep learning focus on presenting the output of a model in a way that is easily understandable and usable for non-experts. These techniques can help users gain insights into how a model works and why it makes certain predictions.

Some of the most common user-friendly techniques for explainability include:

1. Visualization [17]: This involves presenting the output of the model in a visual format, such as a heatmap or a graph, to help users understand how the model is processing the input data.

2. Feature Importance [18]: This technique involves identifying which features of the input data are most important for the model's output. This can help users understand which aspects of the input data are driving the model's predictions.

3. Counterfactual Explanations [19]: This technique involves presenting alternative scenarios to the user, showing what changes would need to be made to the input data to achieve a different model output. This can help users understand how the model is making its predictions and why it may have made a particular decision.

4. Natural Language Explanations [20]: This technique involves presenting the output of the model in natural language, allowing users to understand the model's predictions without requiring technical knowledge.

2.4 Positioning of this work

In the next chapter, a novel deep learning architecture will be presented which has been specifically designed to provide explainability for the classification of medical images. This architecture has been carefully chosen to provide explainability by design, as described in 2.3.1.

To achieve this level of explainability, this work will focus on several key explanations techniques defined in 2.3.2. Firstly, visualization methods will be used to show the prototypes learned by the model and how these evolve over the training epochs. By comparing test images to these prototypes, users can gain a better understanding of how the model is making its classifications and the specific features that are being used to do so.

Secondly, the prototypes learned by the model will be treated as features in the last two layers of the architecture. This approach will allow for a detailed analysis of which features are more important than others in order to classify each image.

Finally, the model will be able to provide counterfactual explanations. This means that the model will not only compare test images to the prototypes of the correct class, but also to all prototypes in the incorrect classes. This information can be interpreted as an explanation of why the other classes did not manage to win in the classification poll.

Overall, this deep learning architecture has been designed with a strong focus

on explainability. By incorporating a range of visualization techniques, feature importance analysis, and counterfactual explanations, users can gain a deeper understanding of the model's decision-making process and the specific features that are being used to classify medical images.

3 Prototype-based Deep Learning

3.1 Origins

One post-hoc mechanism for explaining the decisions made by a model might be to report (in addition to providing predictions) which other examples the model considers to be most similar [11]. After training a deep neural network or latent variable model for a discriminative task, we then have access not only to predictions but also to the learned representations. Then, for any example, in addition to generating a prediction, we can use the activations of the hidden layers to identify the nearest neighbors based on the proximity in the space learned by the model. This type of explanation by example has similarities to how humans use analogy to justify actions, such as when doctors refer to case studies to support a treatment protocol.

Based on the previously presented ideas, a group of researchers from Duke University created a deep network architecture that takes images as input and outputs predictions and explanations based on prototypes. They called this network Prototypical Part Network or ProtoPNet [21]. They attempted to replicate in the architecture the reasoning process some professionals, such as physicians, use to classify images. This network dissects images to find similar patches in known images from each category, and finally makes the classification based on which category had the most similarities with the input image. It is important to note that in the input data there is no reference to which parts should be compared, in other words, the model is responsible for finding the regions of interest in each image; the only input besides the image is its category label.

In the mentioned paper, researchers tested the model with two datasets: one comprising bird images to be classified in species [22], and the other car images to determine their model [23]. As part of this work, the ProtoPNet architecture will

be applied to medical images, as it seems reasonable that the same approach can be used to assist health care professionals to give a diagnostic from an image and provide an adequate explanation for the decision.

3.2 ProtoPNet Architecture

As explained by Chen et. al.[21], the ProtoPNet architecture comprises three modules: a convolutional neural network f, a prototype layer g_P and a fully connected layer h. A scheme of the architecture can be observed in 3.1.

For the convolutional layer f, VGG-19 pre-trained on ImageNet was used in this thesis; similarly to what the Duke team found, that is the model that yielded the best results. On top of VGG-19 two additional 1×1 convolutional layers are used.

After an image x passes through the f layer, it enters the prototype layer g_P. In it, the model will learn m prototypes. Each prototype has the same depth as the convolutional output but smaller height and width. These prototypes represent a specific pattern of activation in a particular patch of the convolutional output. This pattern corresponds to a particular image patch in the original pixel space. Therefore, each prototype represents a latent representation of a specific part of x.

After the prototype layer generates similarity scores, these scores are multiplied by the weight matrix in the fully connected layer h. The resulting output logits are then normalized using softmax to obtain predicted probabilities indicating the likelihood of a given image belonging to different classes.

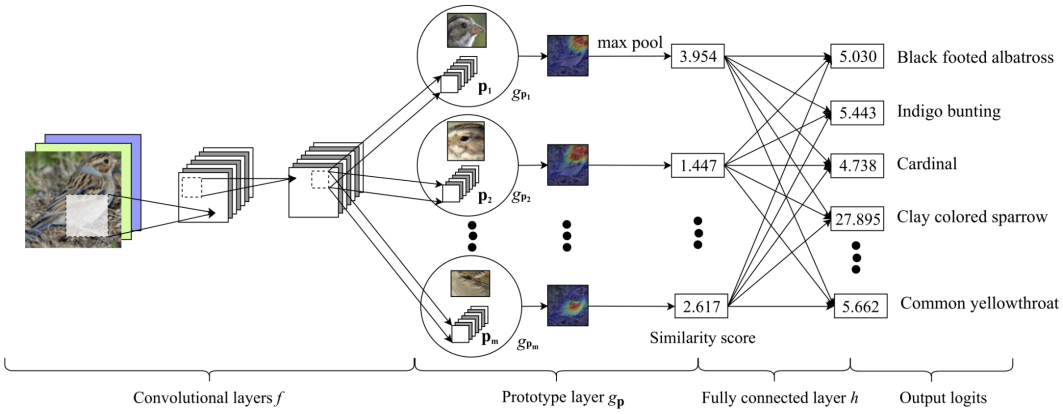


Figure 3.1: ProtoPNet architecture [21]

3.3 Training Algorithm

As explained by Chen et. al. [21], the training process of ProtoPNet is divided into: (1) stochastic gradient descent (SGD) of layers before the last layer; (2) projection of prototypes; (3) convex optimization of last layer.

3.3.1 SGD of layers before last layer

During the initial training phase, the objective is to create a meaningful latent space where crucial image patches for classification are grouped together (using $\|\cdot\|_2$-distance) around prototypes that represent semantically similar classes. Additionally, the model tries to ensure that the clusters centered on prototypes from different classes are distinct and well-separated. To accomplish this, stochastic gradient descent (SGD) is used to optimize both the convolutional layers' parameters and the prototypes in the prototype layer simultaneously, while keeping the last layer's weight matrix unchanged.

Given:

- Convolutional layers' parameters: w_{conv}
- Number or prototypes: m
- Prototypes in prototype layer: $\mathrm{P} = \{\mathrm{p}_j\}_{j=1}^{m}$

The optimization problem to solve is:

$$\min_{\mathrm{P},\omega_{\mathrm{conv}}} \frac{1}{n}\sum_{i=1}^{n} \mathrm{CrsEnt}(h \circ g_{\mathrm{P}} \circ f(x_i), y_i) + \lambda_1 \mathrm{Clst} + \lambda_2 \mathrm{Sep} \tag{3.1}$$

where CrsEnt is the cross-entropy loss, and Clst and Sep are defined as:

$$\mathrm{Clst} = \frac{1}{n}\sum_{i=1}^{n} \min_{j:\mathrm{p}_j \in \mathrm{P}_{y_i}} \min_{\mathrm{z} \in \mathrm{patches}(f(\mathrm{x}_i))} \|\mathrm{z} - \mathrm{p}_j\|_2^2 \tag{3.2}$$

$$\mathrm{Sep} = -\frac{1}{n}\sum_{i=1}^{n} \min_{j:\mathrm{p}_j \notin \mathrm{P}_{y_i}} \min_{\mathrm{z} \in \mathrm{patches}(f(\mathrm{x}_i))} \|\mathrm{z} - \mathrm{p}_j\|_2^2 \tag{3.3}$$

and λ_1 and λ_2 are tuning parameters.

3.3.2 Projection of prototypes

In order to visualize the prototypes as training image patches, the training process includes a technique that involves projecting ("pushing") each prototype onto the nearest latent patch from a training image that belongs to the same class as the prototype. By doing so, each prototype can be essentially considered to be equivalent to a training image patch.

Mathematically, for prototype p_j of class k, i.e., $\mathrm{p}_j \in \mathrm{P}_k$, the update is performed as follows:

$$\mathrm{p}_j \leftarrow \arg\min_{\mathrm{z} \in \mathcal{Z}} \|\mathrm{z} - \mathrm{p}_j\|_2 \tag{3.4}$$

where:

$$\mathcal{Z} = \{\tilde{z} : \tilde{z} \in \mathrm{patches}(f(\mathrm{x}_i)) \; \forall i \; s.t. \; y_i = k\} \tag{3.5}$$

To expedite the training process, we perform the projection of prototypes once every ten epochs instead of every iteration. As explained later in this document, a visualization mechanism has been developed as part of this work to monitor the evolution of each prototype. This mechanism confirms that prototypes do not change frequently enough to necessitate projecting them in every epoch.

3.3.3 Convex optimization of last layer

In the last training phase, a convex optimization is performed on the weight matrix w_h of the last layer h. The goal here is to adjust the last layers weights $w_h^{(k,j)}$ for k and j with $\mathrm{p}_j \notin \mathrm{P}_k$, so the final model has the sparsity property $w_h^{(k,j)} \approx 0$. In other words, it is desired that the model relies less on a negative reasoning process such as "this image is of class l because it is not of class k".

In other words, a Lasso Regularization is applied to the optimization algorithm (Stochastic Gradient Descent). The network minimizes the loss function plus the regularization penalty, leading to a balance between fitting the training data and maintaining a simpler model. In practice, the lasso regularization penalty has the effect of setting some of the weights in the network to zero, effectively eliminating some of the features in the input data, which in this case turns to be some prototype p_j of class k.

Mathematically, the optimization problem to solve is:

$$\min_{w_h} \frac{1}{n} \sum_{i=1}^{n} \mathrm{CrsEnt}(h \circ g_{\mathrm{P}} \circ f(x_i), y_i) + \lambda \sum_{k=1}^{K} \sum_{j:\mathrm{p}_j \notin \mathrm{P}_k} |w_h^{(k,j)}| \tag{3.6}$$

In this stage, all parameters in the convolutional and prototype layers are held constant. Consequently, the optimization process is convex and enhances accuracy without modifying the learned latent space of prototypes.

3.4 Datasets

3.4.1 Initial tests

In order to ensure the replicability of the results presented in the ProtoPNet paper, the first experiment was conducted using the original dataset of bird images [22]. However, this process was not without its challenges. Technical difficulties such as hardware differences, specifically GPUs used for training the model, and PyTorch version had to be overcome. Despite these challenges, the first experiment was successful in reproducing the published results.

To gain a deeper understanding of how the model generates prototypes and learns, the next experiment utilized a dataset of clothing images [24]. This approach was chosen due to the relative ease of identifying different parts of cloth-

ing items compared to medical images, such as tumors, which require specialized trained medical professionals to verify the model's performance. To further evaluate the explainability power of ProtoPNet, a bee images dataset [25] was employed as an intermediate step before moving on to medical images.

3.4.2 Skin Lesion Images Dataset

The primary dataset utilized in this study is sourced from the ISIC Challenge 2019 [26] [27] [28]. The International Skin Imaging Collaboration (ISIC) is an international organization that aims to improve the diagnosis of melanoma and other skin conditions through the use of artificial intelligence and machine learning [29]. The organization hosts an annual challenge known as the ISIC Challenge, which is a widely recognized benchmark for the evaluation of automated melanoma detection systems.

The ISIC Challenge provides participants with a dataset of skin images, including both benign and malignant lesions, which are annotated by a team of dermatologists. Participants are required to develop and submit a computer algorithm that can accurately classify skin lesions. The algorithms are then evaluated based on their accuracy, sensitivity, specificity, and area under the receiver operating characteristic (ROC) curve. The ISIC Challenge also includes a sub-challenge focused on lesion segmentation, which involves accurately identifying the boundaries of skin lesions within an image.

The dataset sourced from the 2019 challenge includes 25,331 images of skin lesions that have been classified into eight distinct disease categories: Melanoma (MEL), Melanocytic nevus (NV), Basal cell carcinoma (BCC), Actinic keratosis intraepithelial carcinoma (AK), Benign keratosis (BKL), Dermatofibroma (DF), Squamous cell carcinoma (SCC) and Vascular lesion (VASC). Figure 3.2 shows the distribution of images per class available in the training dataset.

The results obtained using this dataset will be detailed later in this document.

3.4.3 Other Medical Datasets Explored

A couple of additional datasets were used for both training and testing, but did not yield satisfactory results. These datasets, which will be listed below, were likely unsuitable for ProtoPNet due to certain characteristics of the images. Potential

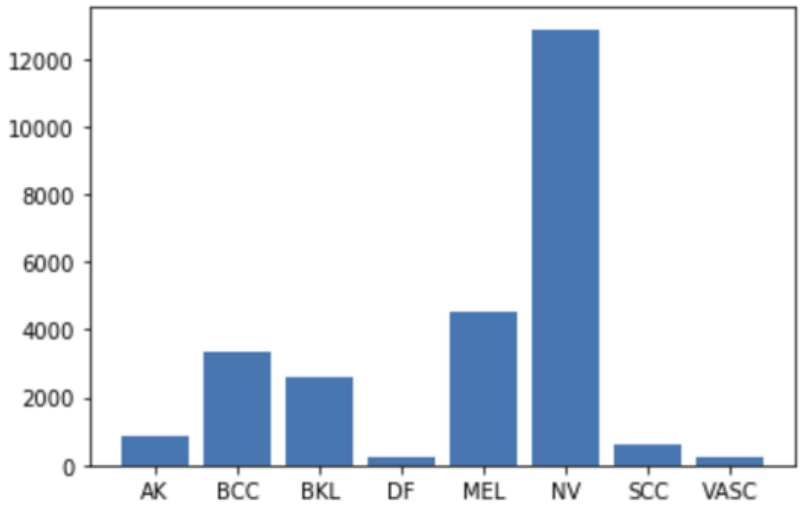


Figure 3.2: Number of images per class in the ISIC Challenge 2019 dataset

hypotheses for why ProtoPNet may have underperformed on these datasets will be presented. However, a comprehensive analysis of why ProtoPNet was unable to achieve satisfactory levels of accuracy and explainability on these datasets will be left for future work, as it falls beyond the scope of the current study.

3.4.3.1 Brain MRI Images Dataset

The TCGA-LGG dataset [30] is a commonly used dataset in the field of medical imaging, particularly in the analysis of brain tumors. The dataset includes MRI images of both healthy brains and brains with various types of tumors. ProtoPNet was evaluated on this dataset to assess its ability to accurately classify the images and provide meaningful insights into the underlying features that distinguish tumors from healthy brain tissue.

However, the results obtained from ProtoPNet on the dataset were not particularly informative in terms of explainability. One possible explanation for this could be the lack of identifiable characteristics in the tumor images. Unlike skin lesions, which often exhibit visible features such as color, texture, and shape, the tumors in the TCGA-LGG dataset lack clear distinguishing features that could help the model identify ten different prototypical parts, as can be seen in Figure 3.3.

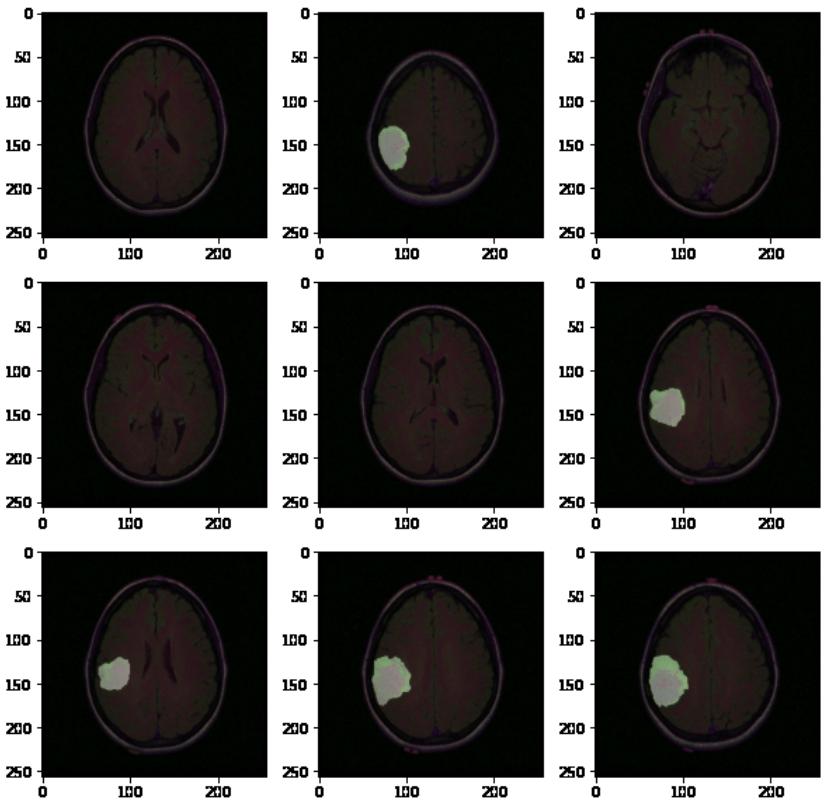


Figure 3.3: Sample MRI images from the TCGA-LGG dataset

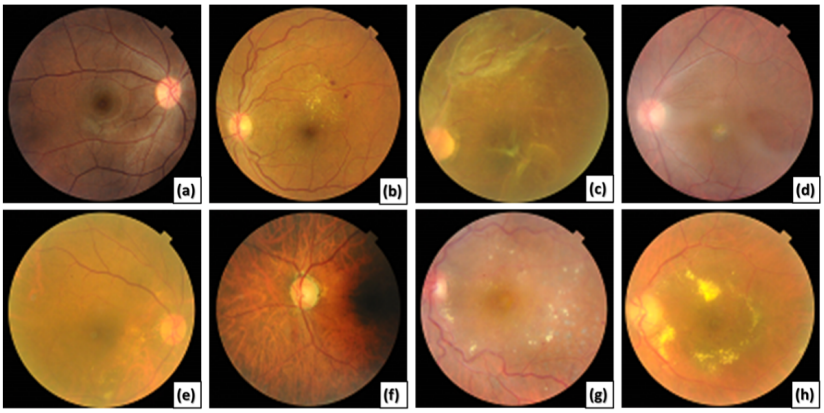


Figure 3.4: Sample Retinal Fundus from the RFMiD

3.4.3.2 Retinal Fundus Images Dataset

The Retinal Fundus Multi-Disease Image Dataset (RFMiD) [31] is a comprehensive dataset that contains 3,200 fundus images, each of which has been annotated with one of 46 retinal conditions through the adjudicated consensus of two senior retinal experts. ProtoPNet was evaluated on this dataset to assess its ability to classify these conditions accurately and provide insights into the relevant features that distinguish them from one another.

However, ProtoPNet failed to create good prototype images on the RFMiD dataset. One possible explanation for this is that retinal conditions often manifest as subtle changes in the overall texture of the image, rather than appearing as distinct regions or shapes, as is often the case with skin lesions. In figure 3.4 there are eight retina images with different conditions, it can be observed how subtle the chages are. This makes it challenging for a model like ProtoPNet, which is designed to identify prototypes in specific regions of an image, to effectively capture the relevant features that distinguish between the different conditions.

Additionally, the high dimensionality of the fundus images presents a significant challenge for training models, as it requires a correspondingly large amount of memory to process. In order to make the RFMiD dataset more manageable for ProtoPNet, the images were down-sampled prior to testing. However, this approach may have impacted the quality of the results obtained, as some information may have been lost in the re-scaling process.

3.5 Metrics

When developing and evaluating a deep learning model, it is essential to use appropriate performance metrics to assess its effectiveness. In this study, two primary metrics were used: accuracy and recall.

3.5.1 Accuracy

Accuracy is a performance metric that measures the percentage of correctly predicted examples in a dataset. It is one of the most commonly used metrics in machine learning, particularly for classification problems where the goal is to assign examples to one of two or more classes.

Accuracy is calculated as the number of correct predictions divided by the total number of predictions made by the model. It provides a measure of the model's overall performance in terms of its ability to correctly classify examples. A high accuracy score indicates that the model is effective at making correct predictions, while a low accuracy score suggests that the model may be making more incorrect predictions than correct ones.

Accuracy is an important metric for deep learning models because it is easy to understand and interpret. It provides a general measure of the model's prediction performance, and it can be used to compare the performance of different models or different versions of the same model. However, accuracy has some limitations. For example, it may not be appropriate when the distribution of classes in the dataset is imbalanced, where the majority of examples belong to one class. In such cases, the model may achieve a high accuracy simply by predicting the majority class for all examples, which may not be useful in practice.

3.5.2 Per-class Recall

In deep learning, recall is a performance metric that measures the ability of a model to correctly identify all positive examples in a dataset. Recall is calculated as the ratio of true positives to the sum of true positives and false negatives. In other words, it measures the proportion of actual positive examples that are correctly identified as positive by the model. A high recall indicates that the model is effective at identifying positive examples, while a low recall suggests that the model may be missing some positive examples.

Recall is an important metric for deep learning models because it reflects the model's ability to correctly identify instances of the target class, which is often the main objective in many real-world applications. For example, in medical diagnosis as is the case in this work, it is crucial to identify all cases of a particular disease, even if it means potentially identifying some healthy patients as positive (i.e., a false positive).

Per-class recall, also known as class-specific recall, is a variant of recall that is calculated separately for each class in a multi-class classification problem. It was the appropriate variant to use here, as the experiments conducted were multi class problems. Per-class recall measures the proportion of positive instances in each class that are correctly identified by the model, and is calculated using the same formula as recall but applied to each class separately.

By using both accuracy and per-class recall as performance metrics, this study aimed to evaluate the overall effectiveness of the deep learning model. While accuracy provides a general indication of the model's prediction performance, recall provides a more specific insight into the model's ability to identify the target class. Together, these metrics provide a more complete picture of the model's strengths and weaknesses, enabling researchers to fine-tune the model and improve its performance.

4 Contributions

This chapter presents the main contributions of the work, starting with an overview of the visualization tools developed to facilitate the interpretation of the results. The mechanisms used to improve both accuracy and explainability are then explained, including techniques such as pruning duplicate prototypes, modifying the loss function, and analyzing the top-k predictions. By leveraging these methods, a significant improvement has been achieved in both the performance and interpretability of the deep learning model. Overall, this chapter provides a comprehensive summary of the innovative approaches taken to address the challenges of medical image analysis.

4.1 Visualization

Visualization is crucial for explainability in deep learning models because it helps humans understand how the model makes its predictions. Event though ProtoPNet is an architecture designed to be interpretable, visualization can help make the inner workings of this models more transparent.

Visualization can also be used to identify and diagnose problems with a deep learning model. For example, visualization tools developed in this work were used to find problems with prototype images being repeated.

Furthermore, visualization can help build trust in deep learning models, which is especially important in contexts where patients' health is at stake.

4.1.1 Learned Prototypes

Before proceeding to create new visualization tools, an existing tool provided by the original authors of ProtoPNet [21] was adapted to display all the learned proto-

types. The adapted tool allowed for a visual inspection of the learned prototypes, which facilitated a more thorough analysis of the model's performance. By examining the prototypes, it was possible to assess the quality and relevance of the learned features, as well as the effectiveness of the model in capturing the key characteristics of the input data.

Moreover, this tool provided insights into the overall architecture of the model, which was instrumental in identifying potential sources of error and identifying areas for improvement. This analysis ultimately helped to refine the model and optimize its performance.

In image 4.1, the prototypes learned by the model for each class are visible. The prototypes are labeled with 'C' for class and 'P' for prototype. For example, 'C:2 P:8' indicates the eighth prototype for class 2. Each row in the image corresponds to the ten prototypes for a specific class.

As explained in 3.3.2, visualizing the prototypes, which are tensors in the model, requires a comparison of each prototype with all possible patches of the training images belonging to the same class as the prototype. The patch in the training image that reduce the distance to the prototype is selected as the closest match. Image 4.2 displays the training image that corresponds to each prototype. Additionally, image 4.3 presents a heatmap over the same training image, highlighting the exact location of the chosen patch in the corresponding training image.

4.1.2 Prototypes Evolution

The first visualization tool developed from scratch intended to demonstrate how learned prototypes evolve in the training process. As explained in Section 3.3.2, the visual representation of each prototype is updated every ten epochs. That means that after weights are updated for the tenth consecutive time, prototype images will be updated finding training images with features as similar as possible to the prototypes themselves.

The tool developed here shows each different prototype with a different color. As depicted in image 4.4, each row represents all prototypes for the same class [0, 1 ... 9] at a given training epoch, starting with epoch ten, second row corresponds with epoch twenty and so on. Given a chart, each color represents a different visual prototype, meaning that having the same color multiple times, in the same epoch or not, is equivalent to having a prototype image that is repeated.

Figure 4.1: Learned prototypes

Figure 4.2: Training images

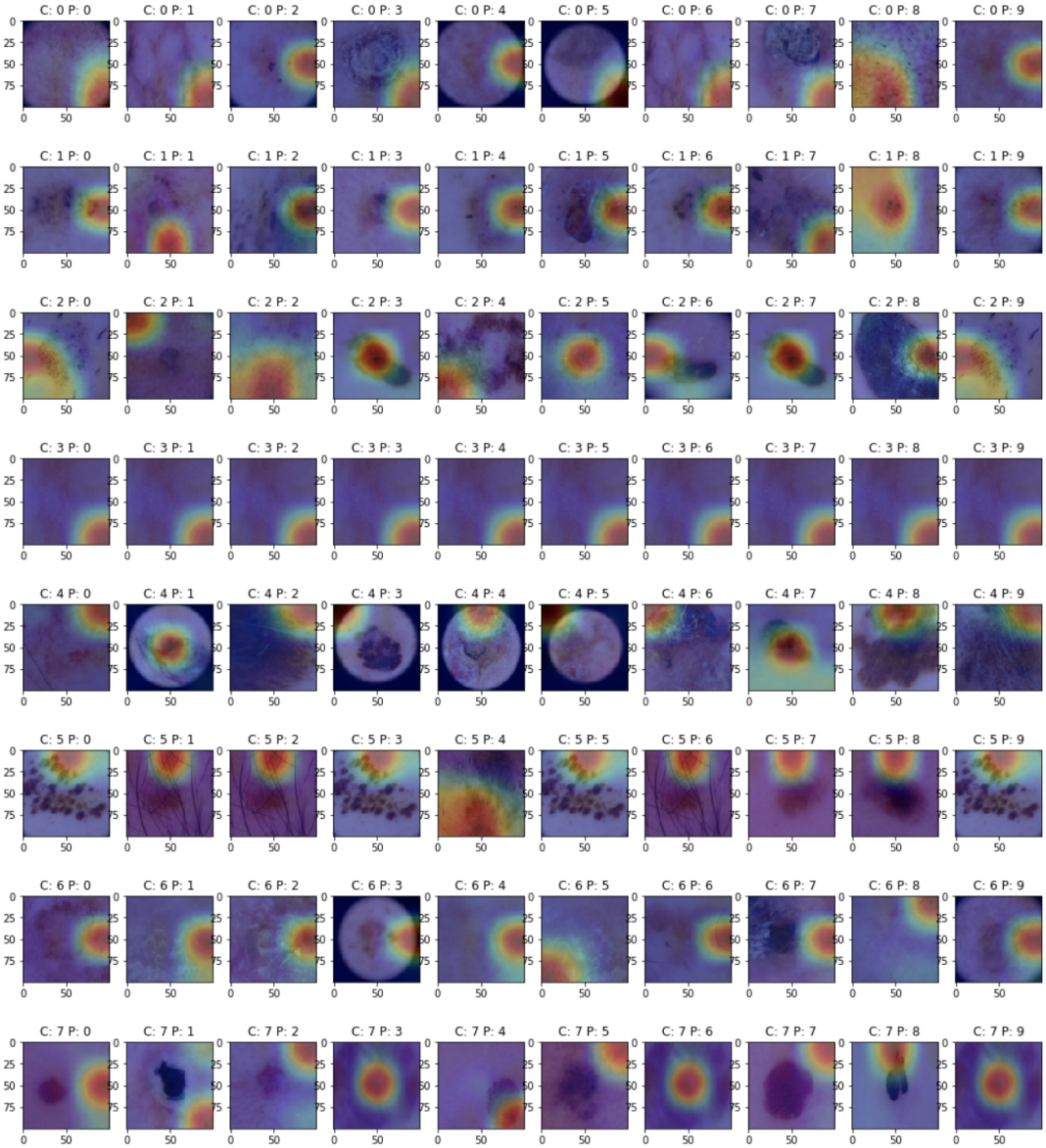


Figure 4.3: Prototypes heatmaps

For a more comprehensive visualization, refer to image 4.5, which displays the charts. The charts illustrate the number of unique prototypes per class and how this number evolves every 10 training epochs.

4.1.3 Test Image Visualization

Another visualization tool developed in this work is aimed at evaluating how the model analyzes a test image and providing a good explanation for its output. While the tool was based on existing code from ProtoPNet, the overall structure and presentation are novel to this work.

The tool includes an analysis for each possible class for the test image. For each class, the tool presents all distinct prototypes and the region from the test image where the prototypes activate the most. Additionally, the tool presents the similarity score of the test image patch to the prototype along with the class connection.

Finally, the tool presents the overall score for each class, which is essentially the sum of the product of the similarity score by the class connection. The final prediction is made by selecting the class with the highest score. The tool's ability to present a detailed analysis of each class and prototype significantly improves the model's interpretability and provides insights into how the model makes its predictions.

In this section, the test image 4.6 and its analysis for the predicted class 4.7 are presented. For a comprehensive view of all classes, the appendix 7.1 should be consulted.

As an extension of the previous tool, the idea of rendering prototypes from training images based on the closest patch was explored further to enhance explainability. To accomplish this, an alternative method was developed wherein not just the closest patch but the three closest patches were chosen as prototypes. This approach aimed to provide users with greater variations and, consequently, a higher level of explainability. The rationale behind this approach was that the closest patch may not always be visually interesting to the human eye, but its variation might be.

To demonstrate the effectiveness of this method, image X 4.8 depicts the results of the approach for the correct class of the test image. By providing users with

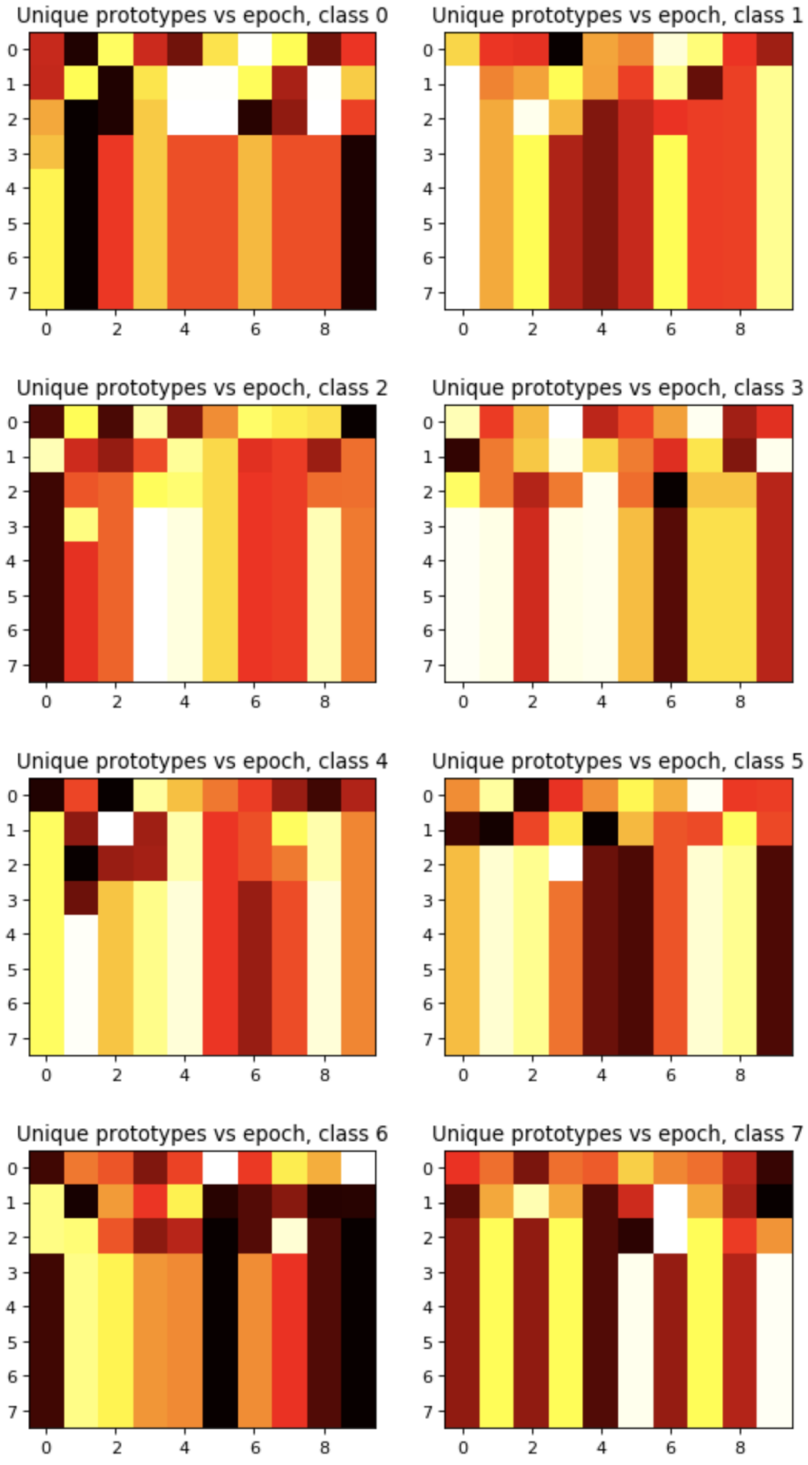


Figure 4.4: Prototypes evolution

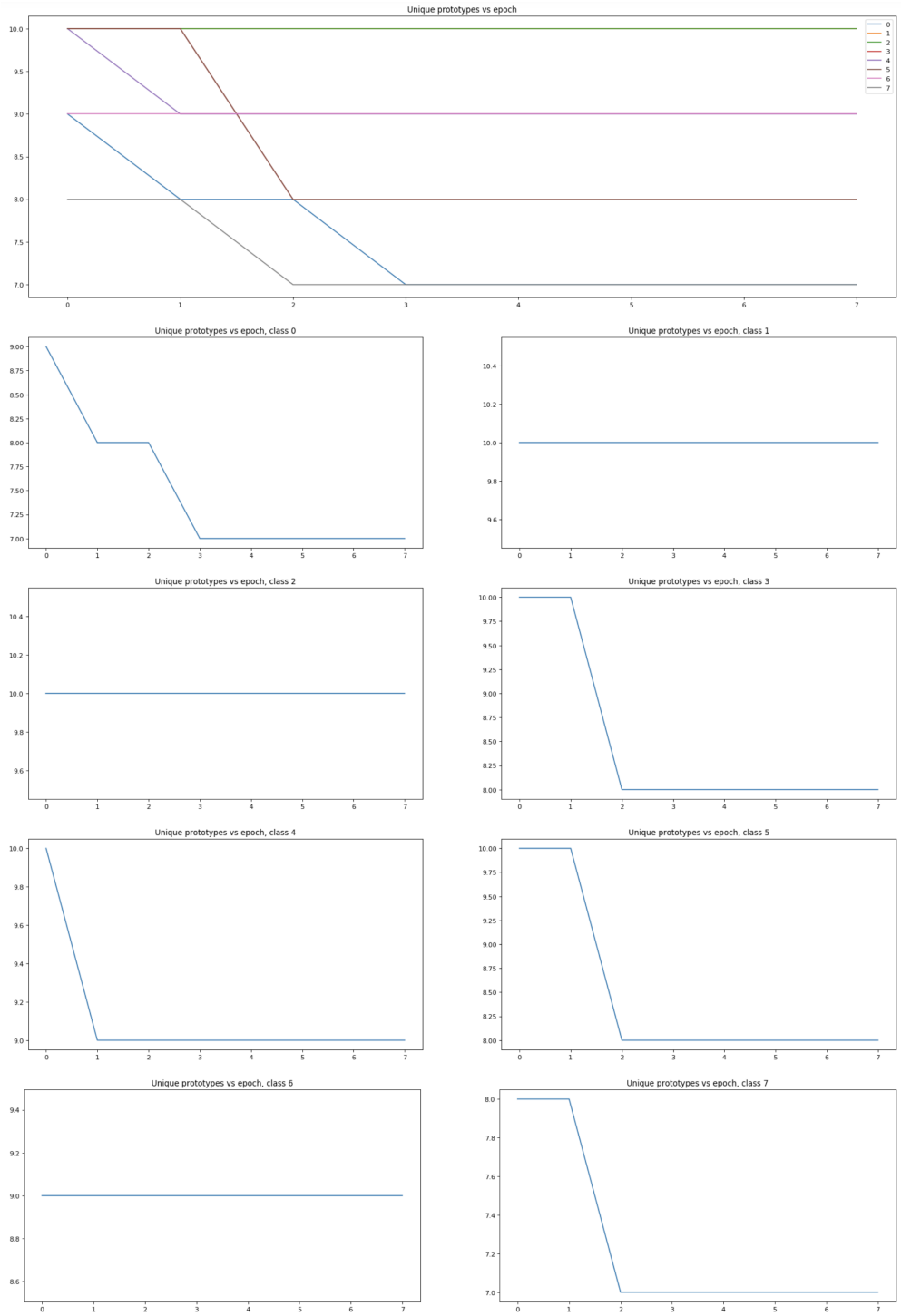


Figure 4.5: Prototypes evolution

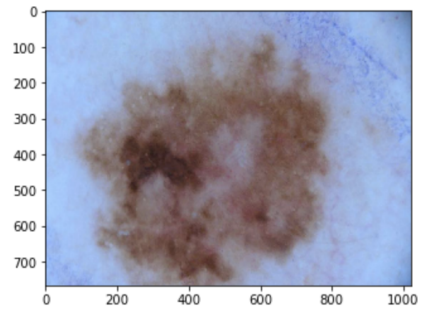
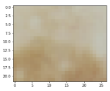
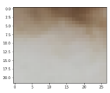
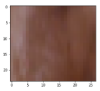
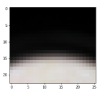
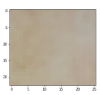
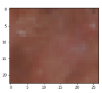


Figure 4.6: Test image

a more diverse set of patches to examine, this approach can lead to a better understanding of the model's decision-making process, and ultimately, enhance its interpretability.

Class 4

Original image (box showing part that looooks like prototype)	Prototype	Training image where prototype comes from	Activation map	Similarity Score	Class connection	Point contributed
				2.2477357387542725	0.7536998987197876	1.6941181421279907
				3.531721591949463	0.3618588447570801	1.2779847383499146
				2.5548222064971924	0.9023345708847046	2.3053042888641357
				2.1577799320220947	0.6141229867935181	1.32514226436615
				2.2458603382110596	0.7498186826705933	1.683988094329834
				2.9767518043518066	0.9449814558029175	2.8129751682281494
				2.255777359008789	0.7116814851760864	1.605394959449768
				2.490800380706787	0.5712887644767761	1.4229662418365479
				2.34517240524292	0.8628702163696289	2.0235793590545654

Total points to class 4: 16.151453018188477

Figure 4.7: Test image comparison to prototypes from the correct class

Class 7

Original image (box showing part that looooks like prototype)	P. variant: 0	Training image where prototype variant 0 comes from	P. variant: 1	Training image where prototype variant 1 comes from	P. variant: 2	Training image where prototype variant 2 comes from	Similarity Score	Class connection	Point contributed
							2.054E+00	4.295E-01	8.823E-01
							2.539E+00	5.758E-01	1.462E+00
							1.888E+00	3.317E-01	6.263E-01
							1.001E+00	9.565E-01	9.575E-01
							2.047E-01	7.576E-01	1.551E-01
							4.692E+00	1.168E+00	5.481E+00
							1.001E+00	9.565E-01	9.575E-01
							1.861E+00	5.214E-01	9.704E-01
							1.095E+00	1.358E+00	1.487E+00
							1.001E+00	9.565E-01	9.575E-01

Total points to class 7: 13.937006950378418

Figure 4.8: Test image comparison to prototypes variations from the correct class

4.2 Prunning

Improving the explainability of the model involves analyzing the prototypes learned during the training process, to assess how meaningful or not they are to users. However, after the initial tests, it was discovered that some prototypes were less valuable than others. In some cases, the model learned prototypes that were repeated or prototypes that did not include part of the lesions in them, instead they matched the background skin or image borders. Therefore, it became necessary to prune or remove these prototypes. Two approaches were tested: the pruning mechanism used by the authors of the original paper and a custom prune based on removing duplicated prototypes. This enabled us to identify the most informative and representative prototypes, which in turn improved the interpretability of the model.

In the original work, authors proceeded to prune prototypes after training the model [32]. The pruning process begins with identifying the k-nearest latent patches of training images to each prototype p_j. Since the labels of all training images are known, it is possible to determine the class label of the k training images that have the k-nearest latent patches to p_j. If the number of training images from the designated class of prototype p is below a specific threshold, the prototype p_j is considered to correspond to a background patch, and it is removed from the model.

The effectiveness of this approach was evaluated using skin lesion images, but the results were not satisfactory. It is presumed that this lack of success can be attributed to the fact that, unlike bird images, skin lesions exhibit a largely homogeneous background (skin). In a test with a binary classification of skin lesions (malign vs. benign), only one prototype was removed, out of a total of twenty (ten per class). In addition, it should be noted that the aforementioned pruning mechanism did not effectively address the issue of repeated prototypes. As a result, an alternative pruning technique was employed.

Since the main issue to be addressed was the existence of multiple repeated prototypes within certain classes, the pruning algorithm was modified to eliminate prototypes from the same class that originated from the same patch within the same training images. Such prototypes would appear visually identical, and removing them would enhance the model's interpretability. Although this change did not significantly impact the model's accuracy, it did improve its explainability, as multiple prototypes depicting the same image could be confusing for the end user.

4.3 Loss Function Modification

The original loss function defined by ProtoPNet presented in Section 3.3.1 had a Clst component. The minimization of the cluster cost (Clst) encourages each training image to have some latent patch that is close to at least one prototype of its own class. After examining the training process, it became clear that implementing the Clst function in the manner suggested by the authors resulted in repeated prototypes, particularly in the skin lesion dataset. The issue arose because asking for only one prototype to be similar to at least one patch of an image could lead to the identification of a generic prototype that is similar enough to several training images, resulting in its repetition across the dataset.

As a result, the proposed modification to the Clst function involves utilizing the mean distance from prototypes to latent patches in the training images. This approach mandates that all prototypes in the image's class be close to at least one patch to avoid excessive penalization. That change substitutes Clst from the original implementation in 3.1 with the following:

$$\text{Clst} = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{1}{|\mathrm{P}_{y_i}|}\sum_{\mathrm{p}_j \in \mathrm{P}_{y_i}} \min_{\mathrm{z}\in \text{patches}(f(\mathrm{x}_i))} \|\mathrm{z}-\mathrm{p}_j\|_2^2\right] \tag{4.1}$$

As depicted in the image 4.9, prior to the update of the loss function, the number of repeated prototypes was significant. However, after the loss function update, there was a significant increase in the number of unique prototypes. For example, class 3 initially had only one unique prototype, but after the update, it had eight different prototypes. This increase in the number of unique prototypes was achieved by ensuring that each prototype in a class has at least one closely related patch in the training images, as explained earlier. The improved distribution of unique prototypes in the classes can lead to a better interpretability of the model. For a view of the prototype images, before and after the loss function update, refer to appendix 7.2.

The change in the loss function not only resulted in improved prototypes, but also led to an enhancement in overall performance. Table 4.1 demonstrates that the accuracy increased from 66.62 to 73.89, and the recall for all classes also saw an improvement.

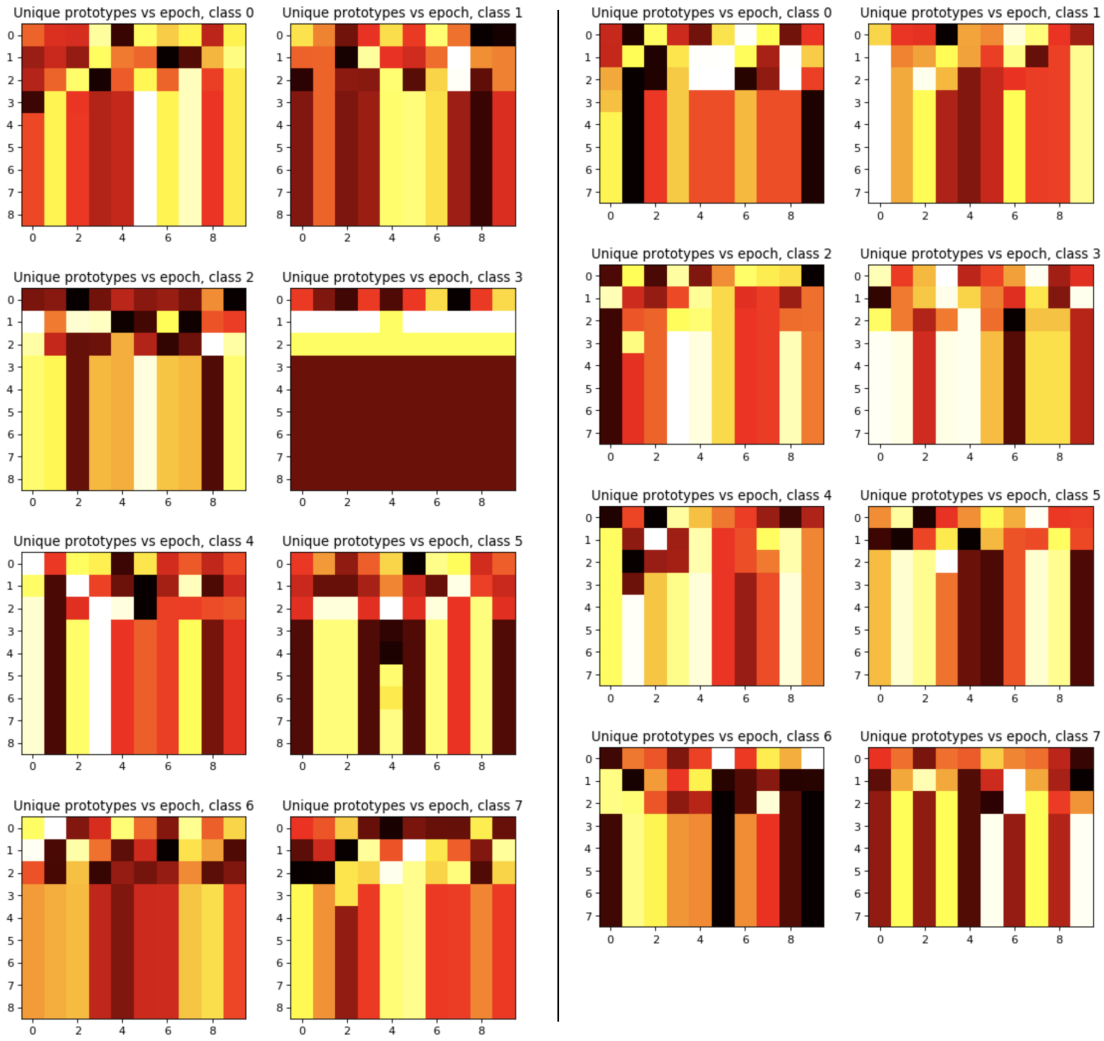


Figure 4.9: Prototypes evolution comparison. Left: pre-changing loss function. Right: post-changing loss function

	Original loss function	New loss function
Accuracy	66.62	73.89
Recall Class 0	0.5346	0.8338
Recall Class 1	0.6959	0.8487
Recall Class 2	0.6022	0.6700
Recall Class 3	0.8420	0.9141
Recall Class 4	0.5214	0.6815
Recall Class 5	0.7257	0.7268
Recall Class 6	0.5934	0.8078
Recall Class 7	0.9522	0.9742

Table 4.1: Accuracy and per-class recall results, before and after updating the loss function

4.4 Top-K

In deep learning, "top-k" is a method used to evaluate the performance of a classification model. It measures the accuracy of the model by looking at the percentage of times the correct label appears in the top k predictions made by the model. The top-k evaluation metric measures the percentage of cases where the correct diagnosis is within the top k predicted diagnoses. For example, if k=1, this is equivalent to the standard accuracy metric, where the model is evaluated on whether it predicts the correct diagnosis as the most likely one. If k=5, the metric measures whether the correct diagnosis is among the top 5 diagnoses predicted by the model.

As proven by Dorr et al. [33], physicians can benefit from the aid of AI models. Providing physicians with a shortlist of options in a multi-class problem can assist them in making the correct diagnosis. Therefore, even if the correct class is not the top-ranked option, if it appears in the top-2 or top-3 choices, the model remains relevant.

Top-k evaluation is particularly relevant in skin lesion diagnosis, where the correct diagnosis may not always be the most probable one. In some cases, the model may predict a different diagnosis with a slightly higher probability, but the correct diagnosis may still be within the top k predicted diagnoses. As shown in table 4.2, considering just the top-2 most probable cases significantly improves accuracy. This is especially important in clinical practice, where physicians typically consider a short list of possible diagnoses based on their experience and the patient's clinical presentation.

Overall, top-k evaluation is a useful metric for assessing the performance of

	Original loss function			New loss function		
	Top-1	Top-2	Top-3	Top-1	Top-2	Top-3
Accuracy	66.62	84.13	93.29	73.89	91.29	96.97
Recall Class 0	0.5346	0.8038	0.9154	0.8338	0.9324	0.9616
Recall Class 1	0.6959	0.8081	0.9169	0.8487	0.9441	0.9747
Recall Class 2	0.6022	0.7992	0.9275	0.6700	0.8644	0.9710
Recall Class 3	0.8420	0.8975	0.9143	0.9141	0.9399	0.9519
Recall Class 4	0.5214	0.8144	0.9093	0.6815	0.9119	0.9759
Recall Class 5	0.7257	0.8658	0.9470	0.7268	0.9126	0.9685
Recall Class 6	0.5934	0.8549	0.9297	0.8078	0.8963	0.9304
Recall Class 7	0.9522	0.9753	0.9815	0.9742	0.9833	0.9878

Table 4.2: Top 1, 2 and 3 accuracy and per-class recall results

deep learning models in medical predictions with skin lesions, as it provides a more nuanced evaluation of the model's ability to predict the correct diagnosis among a set of likely possibilities.

5 Conclusions

This work has evaluated and improved ProtoPNet as an explainable deep learning model and demonstrated its application to the analysis of medical images. In the process of applying ProtoPNet to this novel field, several technical and semantic challenges were discovered, which were identified and addressed in order to improve the performance and interpretability of the model.

The first important point to note is that ProtoPNet is not universally applicable to all datasets. While it performed well on skin lesion images, it was not as effective on brain MRI and retinal images. Our analysis revealed that in order to identify meaningful prototypes, the region of interest must be localized in one part of the image and possess distinguishing characteristics such as color, shape, and texture. This was a significant factor in the model's lack of success in the analysis of brain MRI and retinal images, which tend to have more diffuse and less distinctive features that are difficult to localize.

The pruning technique introduced, which involves removing repeated prototypes, was found to be effective in improving explainability. This technique reduces the number of prototypes that the user needs to focus on, making it easier to interpret the model's predictions. While this technique does not improve the model's overall performance, it does enhance the usability of the model, especially in cases where the number of prototypes can be overwhelming.

The visualization tools developed for this work were highly effective in enhancing the interpretability of ProtoPNet, providing insight into its internal learning process. These tools enabled the user to better understand the model's decision-making process and identify which features and prototypes were most important in making predictions. Furthermore, the visualization tools helped to identify potential areas for improvement in the model, such as cases where the model's predictions were unclear or where certain prototypes were over-represented. This information was be used to fine-tune the model and improve its overall performance.

The most significant impact on both the interpretability and performance of ProtoPNet was the change in the loss function. Specifically, moving from requiring at least one prototype to be close enough to a patch of each training image to requiring all prototypes of a given class to find patches on each input image from the same class made the prototypes more distinct and improved the overall performance of the model.

In cases where the model is unable to accurately predict the correct class, using top-k predictions can make a significant difference. This approach is in line with how physicians make their diagnoses, which typically involves considering a few potential options and selecting one among them, often requiring further tests or examinations. Through experimentation with ProtoPNet, it was observed that using top-k predictions (e.g., top-2 or top-3) instead of top-1 resulted in a significant improvement in the model's performance. This suggests that the model's predictions can benefit from considering multiple possible classes, rather than relying solely on the most likely option. In a clinical setting it would mean a health care professional would receive two or three probable diagnosis, giving them the possibility to conduct further studies.

6 Bibliographic references

[1] V. L. Patel, E. H. Shortliffe, M. Stefanelli, P. Szolovits, M. R. Berthold, R. Bellazzi, and A. Abu-Hanna, "The Coming of Age of Artificial Intelligence in Medicine," *Artificial intelligence in medicine*, vol. 46, no. 1, p. 11, May 2009. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2752210/

[2] S. E. Dilsizian and E. L. Siegel, "Artificial intelligence in medicine and cardiac imaging: Harnessing big data and advanced computing to provide personalized medical diagnosis and treatment," *Current Cardiology Reports*, vol. 16, no. 1, p. 7, Jan 2014.

[3] F. Jiang, Y. Jiang, H. Zhi, Y. Dong, H. Li, S. Ma, Y. Wang, Q. Dong, H. Shen, and Y. Wang, "Artificial intelligence in healthcare: Past, present and future," *Stroke and Vascular Neurology*, vol. 2, no. 4, p. 239, Dec 2017.

[4] A. Rehman, S. Naz, and I. Razzak, "Leveraging big data analytics in healthcare enhancement: trends, challenges and opportunities," *Multimedia Systems 2021 28:4*, vol. 28, no. 4, p. 1362, Jan 2021. [Online]. Available: https://link.springer.com/article/10.1007/s00530-020-00736-8

[5] J. He, S. L. Baxter, J. Xu, J. Xu, X. Zhou, and K. Zhang, "The practical implementation of artificial intelligence technologies in medicine HHS Public Access," *Natural Medicine*, no. January, pp. 4–5, 2019. [Online]. Available: www.nature.com/reprints.

[6] M. A. Ahmad, C. Eckert, A. Teredesai, and G. Mckelvey, "Interpretable Machine Learning in Healthcare," *IEEE Intelligent Informatics Bulletin*, vol. 19, pp. 1–3, Aug 2018.

[7] A. Barredo Arrieta, N. Díaz-Rodríguez, J. D. Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, and F. Herrera, "Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI," Tech. Rep., 2019.

[8] G. Montavon, W. Samek, and K.-R. Müller, "Methods for interpreting and understanding deep neural networks," *Digital Signal Processing*, vol. 73, p. 2, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1051200417302385

[9] A. Fernandez, F. Herrera, O. Cordon, M. Jose Del Jesus, and F. Marcelloni, "Michael Margaliot Evolutionary Fuzzy Systems for Explainable Artificial Intelligence: Why, When, What for, and Where to?" *IEEE Computational Intelligence Magazine*, p. 71, Jan 2019.

[10] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, "A Survey of Methods for Explaining Black Box Models," *ACM Comput. Surv*, vol. 51, no. 93, p. 5, 2018. [Online]. Available: https://doi.org/10.1145/3236009

[11] Z. C. Lipton, "The Mythos of Model Interpretability," Tech. Rep., 2017.

[12] C. Trocin, P. Mikalef, Z. Papamitsiou, and K. Conboy, "Responsible AI for Digital Health: a Synthesis and a Research Agenda," *Information Systems Frontiers*, p. 1, 2021. [Online]. Available: https://doi.org/10.1007/s10796-021-10146-4

[13] Y. Wang, M. Xiong, and H. Olya, "Toward an understanding of responsible artificial intelligence practices," in *Proceedings of the 53rd hawaii international conference on system sciences*. Hawaii International Conference on System Sciences (HICSS), 2020, pp. 4964–4965. [Online]. Available: http://hdl.handle.net/10125/64352

[14] B. Mittelstadt, "Designing the Health-related Internet of Things: Ethical Principles and Guidelines," *Information 2017, Vol. 8*, vol. 8, no. 3, p. 2, Jul 2017. [Online]. Available: https://www.mdpi.com/2078-2489/8/3/77/htmhttps://www.mdpi.com/2078-2489/8/3/77

[15] P. Kumar, Y. K. Dwivedi, and A. Anand, "Responsible Artificial Intelligence (AI) for Value Formation and Market Performance in Healthcare: the Mediating Role of Patient's Cognitive Engagement," *Information Systems Frontiers*, p. 4, 2021. [Online]. Available: https://doi.org/10.1007/s10796-021-10136-6

[16] H. Chen, G. Michalopoulos, S. Subendran, R. Yang, R. Quinn, M. Oliver, Z. Butt, and A. Wong, "Interpretability of ML Models for Health Data -A Case Study," Jan 2019, p. 2. [Online]. Available: https://www.researchgate.net/publication/338487561_Interpretability_of_ML_Models_for_Health_Data_-A_Case_Study

[17] G. Ras, N. Xie, M. van Gerven, and D. Doran, "Explainable Deep Learning: A Field Guide for the Uninitiated," *Journal of Artificial Intelligence Research*, vol. 73, pp. 329–396, Jan 2022. [Online]. Available: https://www.jair.org/index.php/jair/article/view/13200

[18] S. Tonekaboni, S. Joshi, M. D. Mccradden, A. Goldenberg, and A. G. Ai, "What Clinicians Want: Contextualizing Explainable Machine Learning for Clinical End Use," pp. 359–380, Oct 2019. [Online]. Available: https://proceedings.mlr.press/v106/tonekaboni19a.html

[19] S. Verma, V. Boonsanong, M. Hoang, K. E. Hines, K. Ai, A. Ai, W. D. C. Usa, J. P. Dickerson, J. Ai, and C. Shah, "Counterfactual Explanations for Machine Learning: A Review Treatment of DNA damaged by gamma radiation by synthetic biology method. View project Counterfactual Explanations and Algorithmic Recourses for Machine Learning: A Review," 2022. [Online]. Available: https://www.researchgate.net/publication/344802765

[20] E. Tjoa and C. Guan, "A Survey on Explainable Artificial Intelligence (XAI): Toward Medical XAI," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 11, pp. 4793–4813, Nov 2021.

[21] C. Chen, O. Li, C. Tao, A. J. Barnett, J. Su, and C. Rudin, "This Looks Like That: Deep Learning for Interpretable Image Recognition," Tech. Rep., 2019.

[22] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The Caltech-UCSD Birds-200-2011 Dataset," 2011. [Online]. Available: https://resolver.caltech.edu/CaltechAUTHORS:20111026-120541847

[23] J. Krause, M. Stark, J. Deng, and L. Fei-Fei, "3d object representations for fine-grained categorization," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 554–561, 2013.

[24] P. Aggarwal, "Fashion Product Images Dataset — Kaggle." [Online]. Available: https://www.kaggle.com/datasets/paramaggarwal/fashion-product-images-dataset

[25] "The BeeImage Dataset: Annotated Honey Bee Images — Kaggle." [Online]. Available: https://www.kaggle.com/datasets/jenny18/honey-bee-annotated-images

[26] M. Combalia, N. C. F. Codella, V. Rotemberg, B. Helba, V. Vilaplana, O. Reiter, C. Carrera, A. Barreiro, A. C. Halpern, S. Puig, and J. Malvehy, "BCN20000: Dermoscopic Lesions in the Wild," Aug 2019. [Online]. Available: http://arxiv.org/abs/1908.02288

[27] V. Rotemberg, N. Kurtansky, B. Betz-Stablein, L. Caffery, E. Chousakos, N. Codella, M. Combalia, S. Dusza, P. Guitera, D. Gutman, A. Halpern, B. Helba, H. Kittler, K. Kose, S. Langer, K. Lioprys, J. Malvehy, S. Musthaq, J. Nanda, O. Reiter, G. Shih, A. Stratigos, P. Tschandl, J. Weber, and H. P. Soyer, "A patient-centric dataset of images and metadata for identifying melanomas using clinical context," *Scientific Data*, vol. 8, no. 1, Dec 2021.

[28] P. Tschandl, C. Rosendahl, and H. Kittler, "Data descriptor: The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions," *Scientific Data*, vol. 5, Aug 2018.

[29] International Skin Imaging Collaboration (ISIC), "ISIC Challenge." [Online]. Available: https://challenge.isic-archive.com/

[30] N. Pedano, A. E. Flanders, L. Scarpace, T. Mikkelsen, J. M. Eschbacher, B. Hermes, V. Sisneros, J. Barnholtz-Sloan, and Ostrom, "The Cancer Genome Atlas Low Grade Glioma Collection (TCGA-LGG) (Version 3) [Data set]," 2016. [Online]. Available: https://doi.org/10.7937/K9/TCIA.2016.L4LTD3TK

[31] S. Pachade, P. Porwal, D. Thulkar, M. Kokare, G. Deshmukh, V. Sahasrabuddhe, L. Giancardo, G. Quellec, and F. Mériaudeau, "Retinal Fundus Multi-Disease Image Dataset (RFMiD): A Dataset for Multi-Disease Detection Research," 2021. [Online]. Available: https://doi.org/10.3390/data6020014

[32] C. Chen, O. Li, C. Tao, A. J. Barnett, J. Su, and C. Rudin, "This Looks Like That: Deep Learning for Interpretable Image Recognition (Supplementary Material)," 2019. [Online]. Available: https://github.com/cfchen-duke/ProtoPNet/blob/master/supp_this_looks_like_that.pdf

[33] F. Dorr, H. Chaves, M. M. Serra, A. Ramirez, M. E. Costa, J. Seia, C. Cejas, M. Castro, E. Eyheremendy, D. Fernández Slezak, M. F. Farez, M. Villalobos Olave, D. Herquiñigo Reckmann, C. Pérez, J. Hernández Pinzon, O. García Almendro, D. Valdez, R. J. Montoya, E. Osa Sanz, N. I. Stefanoff, A. Hualpa, M. Di Cecco, H. Sotelo, F. Ferreyra Luaces, F. Larzabal, J. Ramirez Acosta, R. J. Mosquera Luna, V. Castro, F. Avallay, S. Vargas, S. Villena, R. Forlenza, J. Martinez Pereira, M. Aloisi, M. Conde Blanco, F. Diaz Telli, M. S. Toronchik, C. Gutierrez Occhiuzzi, G. Fourzans, P. Kuschner, R. Castagna, B. Abaz, D. Casero, M. Saborido, M. Escolar, C. Lineros, S. De Luca, G. Doctorovich, L. Dragonetti, C. Carrera, J. Costa Cañizares, L. Minuet, V. Charcopa, C. Mamani, A. Toledo, M. J.

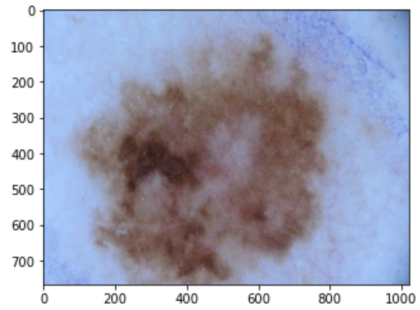
Vargas, A. Quiroz, E. Angeletti, J. Goyo Pinto, C. Correa, J. Pizzorno, R. De Luca, J. Rivas, M. Concheso, A. Villareal, M. Zuleta, and G. Barmaimon, "COVID-19 pneumonia accurately detected on chest radiographs with artificial intelligence," *Intelligence-Based Medicine*, vol. 3-4, Dec 2020.

7 Appendices

7.1 Complete Prototypes Visualization and Analysis for a Test Image

As explained in 4.1.3, the tool developed examines every possible category for the test image and produces an analysis for each one. For every category, the tool displays all prototypes and the specific part of the test image where prototypes activate the most. Furthermore, the tool reveals the similarity score between the test image and the prototype patch, as well as the class connection.

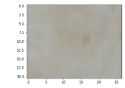
In conclusion, the tool displays an overall score for each category, which is the total of the product of the similarity score and class connection. The class with the highest score is chosen as the final prediction.




```
0 (4, 4)
Predicted: 4
Actual: 4
image index 0 in batch
```

Figure 7.1: Test image

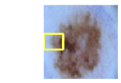
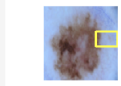
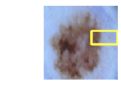
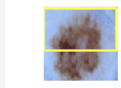
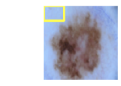
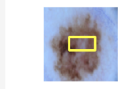
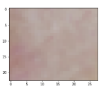
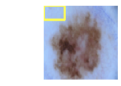
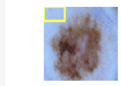
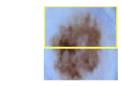
Class 0

Original image (box showing part that looooks like prototype)	Prototype	Training image where prototype comes from	Activation map	Similarity Score	Class connection	Point contributed
				2.407437562942505	-0.4145619869232178	-0.9980320930480957
				5.613049030303955	-0.2788541615009308	-1.5652220249176025
				3.535785675048828	-0.29678431153297424	-1.049365758895874
				1.7854975461959839	-0.8234895467758179	-1.4703385829925537
				4.506862640380859	-0.06394600123167038	-0.2881958484649658
				4.071262359619141	-0.29482367634773254	-1.2003045082092285
				3.291476011276245	-0.588026762008667	-1.9354759454727173

Total points to class 0: -8.50693416595459

Figure 7.2: Class 0 analysis

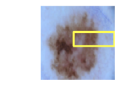
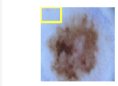
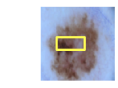
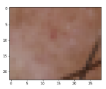
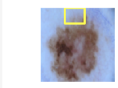
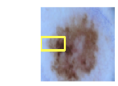
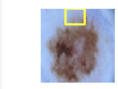
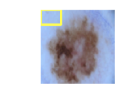
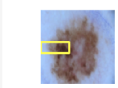
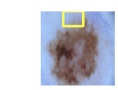
Class 1

Original image (box showing part that looooks like prototype)	Prototype	Training image where prototype comes from	Activation map	Similarity Score	Class connection	Point contributed
				3.736586570739746	-0.08288493007421494	-0.3097067177295685
				5.688477516174316	-0.35402458906173706	-2.0138609409332275
				2.3065807819366455	-0.28362616896629333	-0.6542066931724548
				2.2801513671875	-0.16570043563842773	-0.37782207131385803
				4.656774044036865	-0.0874323919415474	-0.4071528911590576
				2.9304237365722656	-0.3906087875366211	-1.1446492671966553
				2.5209243297576904	-0.42391687631607056	-1.068662405014038
				2.6296989917755127	-0.27564486861228943	-0.7248630523681641
				2.5962321758270264	-0.23267747461795807	-0.6040847301483154
				3.3659698963165283	-0.5884462594985962	-1.9806923866271973

Total points to class 1: -9.285701751708984

Figure 7.3: Class 1 analysis

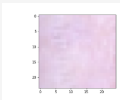
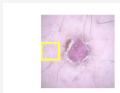
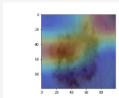
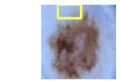
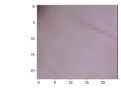
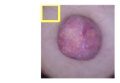
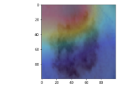
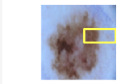
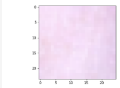
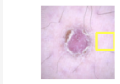
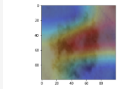
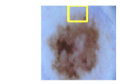
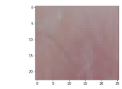
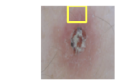
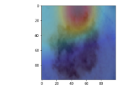
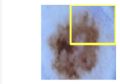
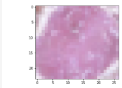
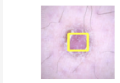
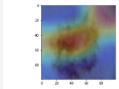
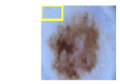
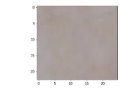
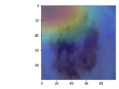
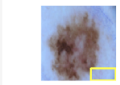
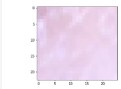
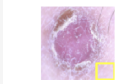
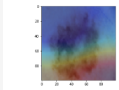
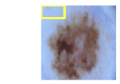
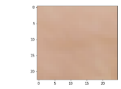
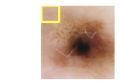
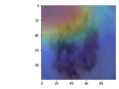
Class 2

Original image (box showing part that looooks like prototype)	Prototype	Training image where prototype comes from	Activation map	Similarity Score	Class connection	Point contributed
				2.2817463874816895	-0.44327542185783386	-1.0114420652389526
				1.9095516204833984	-0.3360713720321655	-0.6417456269264221
				2.6369731426239014	-0.5852484107017517	-1.543284296989441
				2.100165605545044	-0.6509957909584045	-1.3671989440917969
				2.5678250789642334	-0.3661302328109741	-0.9401583671569824
				3.1870288848876953	-0.5175484418869019	-1.6494418382644653
				2.5699477195739746	-0.1687006652355194	-0.4335518777370453
				2.591468334197998	-0.2167089283466339	-0.5615943074226379
				2.732438564300537	-0.303755521774292	-0.8299933075904846
				2.6940298080444336	-0.1620667427778244	-0.43661263585090637

Total points to class 2: -9.415023803710938

Figure 7.4: Class 2 analysis

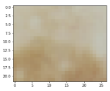
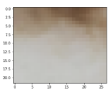
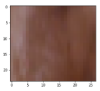
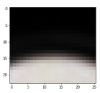
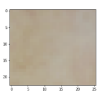
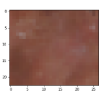
Class 3

Original image (box showing part that looooks like prototype)	Prototype	Training image where prototype comes from	Activation map	Similarity Score	Class connection	Point contributed
				3.642277717590332	-0.5800615549087524	-2.1127452850341797
				2.1335997581481934	-0.6071507334709167	-1.2954167127609253
				2.447902202606201	-0.36715060472488403	-0.8987487554550171
				3.8699307441711426	-0.5304484963417053	-2.0527989864349365
				3.4753165245056152	-0.6672994494438171	-2.3190767765045166
				3.2312073707580566	-0.43635767698287964	-1.4099621772766113
				1.599468469619751	-0.5719116926193237	-0.9147547483444214
				2.317270278930664	-0.5281995534896851	-1.2239811420440674

Total points to class 3: -12.227484703063965

Figure 7.5: Class 3 analysis

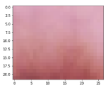
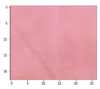
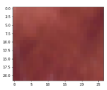
Class 4

Original image (box showing part that looooks like prototype)	Prototype	Training image where prototype comes from	Activation map	Similarity Score	Class connection	Point contributed
				2.2477357387542725	0.7536998987197876	1.6941181421279907
				3.531721591949463	0.3618588447570801	1.2779847383499146
				2.5548222064971924	0.9023345708847046	2.3053042888641357
				2.1577799320220947	0.6141229867935181	1.32514226436615
				2.2458603382110596	0.7498186826705933	1.683988094329834
				2.9767518043518066	0.9449814558029175	2.8129751682281494
				2.255777359008789	0.7116814851760864	1.605394959449768
				2.490800380706787	0.5712887644767761	1.4229662418365479
				2.34517240524292	0.8628702163696289	2.0235793590545654

Total points to class 4: 16.151453018188477

Figure 7.6: Class 4 analysis

Class 5

Original image (box showing part that looooks like prototype)	Prototype	Training image where prototype comes from	Activation map	Similarity Score	Class connection	Point contributed
				2.057603597640991	-0.5025848150253296	-1.0341203212738037
				2.101943254470825	-0.4178091287612915	-0.8782110810279846
				1.827735185623169	-0.4262702465057373	-0.7791091203689575
				2.042240619659424	-0.5617946982383728	-1.1473199129104614
				2.6842246055603027	-0.4980124831199646	-1.3367773294448853
				1.9356812238693237	-0.40085092186927795	-0.7759196162223816
				2.6662683486938477	-0.4837680459022522	-1.2898554801940918
				2.0566046237945557	-0.4600924551486969	-0.9462282657623291

Total points to class 5: -8.187541007995605

Figure 7.7: Class 5 analysis

Class 6

Original image (box showing part that looooks like prototype)	Prototype	Training image where prototype comes from	Activation map	Similarity Score	Class connection	Point contributed
				4.692083835601807	-0.06424683332443237	-0.30145153403282166
				3.5557429790496826	-0.22742408514022827	-0.8086615800857544
				3.904927968978882	-0.07793357968330383	-0.30432501435279846
				4.592701435089111	-0.29110753536224365	-1.3369699716567993
				2.68636155128479	-0.4402626156806946	-1.1827045679092407
				4.29979944229126	-0.30458274483680725	-1.3096446990966797
				4.2680463790893555	-0.7117022275924683	-3.0375781059265137
				5.178215980529785	-0.34753379225730896	-1.7996050119400024
				5.394185543060303	-0.08010892570018768	-0.4321224093437195

Total points to class 6: -10.513063430786133

Figure 7.8: Class 6 analysis

Class 7

Original image (box showing part that looooks like prototype)	Prototype	Training image where prototype comes from	Activation map	Similarity Score	Class connection	Point contributed
				2.012319803237915	-0.43429046869277954	-0.8739312887191772
				2.583183765411377	-0.3196214437484741	-0.8256409168243408
				2.037024736404419	-0.4586748480796814	-0.934332013130188
				3.621241569519043	-0.23788276314735413	-0.8614309430122375
				2.1884427070617676	-0.33661869168281555	-0.736670732498169
				2.995006561279297	-0.36499446630477905	-1.09316086769104
				3.9545602798461914	-0.3846272826194763	-1.5210317373275757

Total points to class 7: -6.846198558807373

`Prediction is correct.`

Figure 7.9: Class 7 analysis

7.2 Prototypes comparison: before and after changing loss function

As depicted in the image 7.10, prior to the update of the loss function, the number of repeated prototypes was significant. However, after the loss function update, there was a significant increase in the number of unique prototypes, as shown in image 7.11.

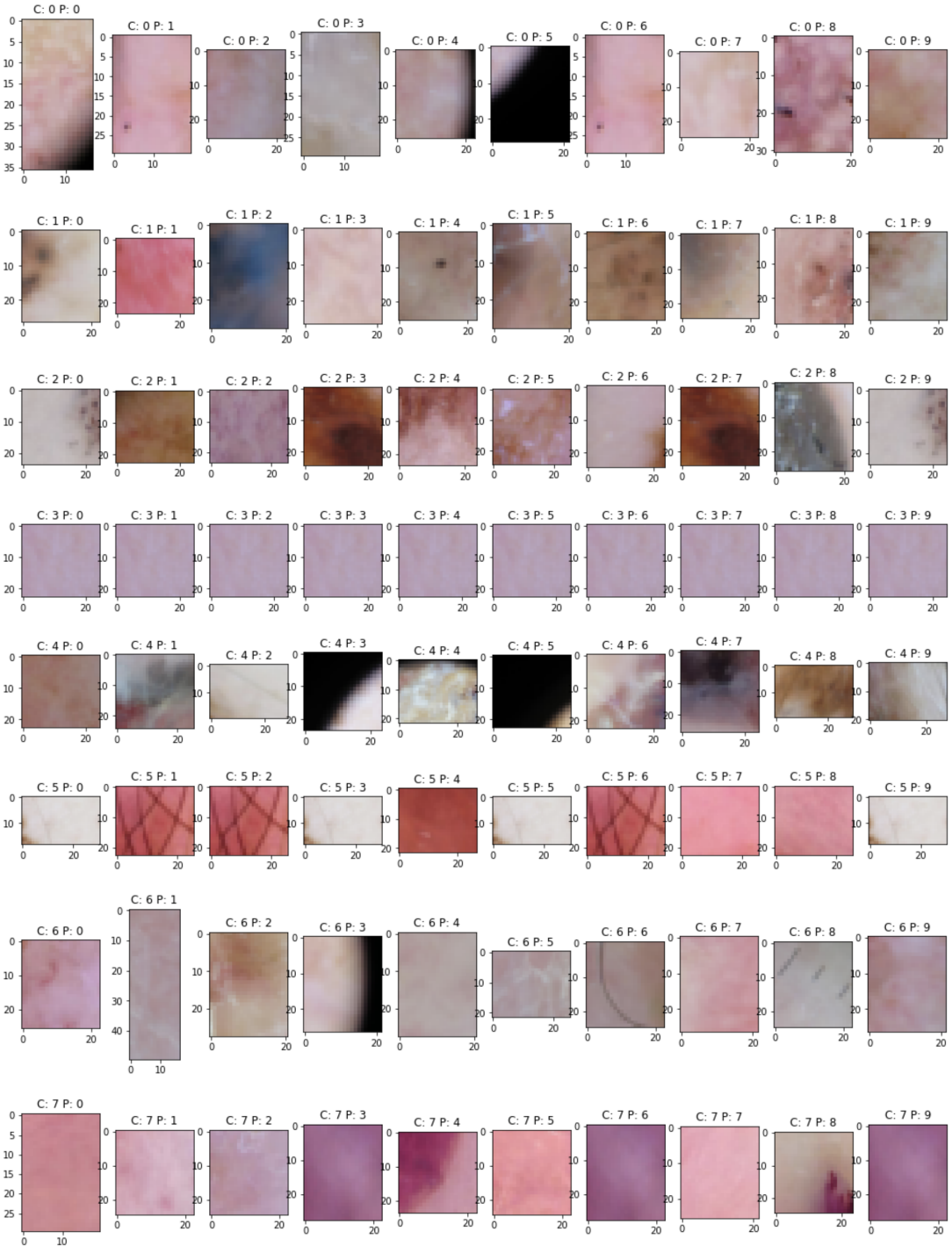


Figure 7.10: Learned prototypes with original loss function

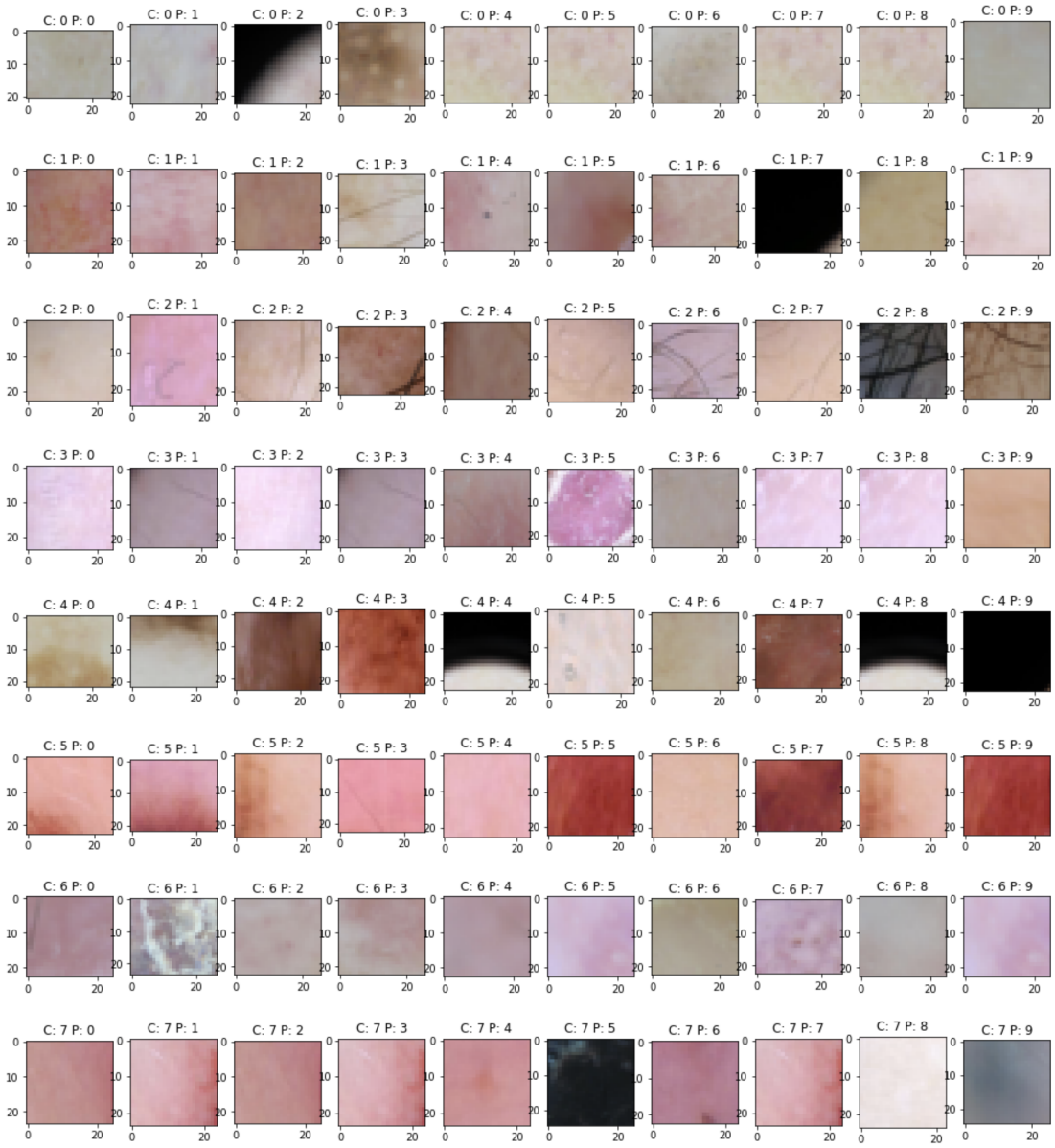


Figure 7.11: Learned prototypes with new loss function