TECHNISCHE UNIVERSITÄT ILMENAU
Faculty of Computer Science and Automation
Software Engineering for Safety-Critical Systems

Master Thesis

Analyzing Possible Influences on Class Activation Based Explainability Methods for Image Classifiers

submitted by

Roman Mysianov
Matrikel 63450

Supervisor:

Prof. Patrick Mäder
M. Sc. Daniel Scheliga

Ilmenau, December 1, 2021

Acknowledgments

I would like to express my deep and sincere gratitude to my research supervisor, M. Sc. Daniel Scheliga, for giving me the opportunity to do research and providing invaluable guidance throughout this research. He has taught me the methodology to carry out the research and to present the research works as clearly as possible. It was a great privilege and honor to work and study under his guidance. I am extremely grateful for what he has offered me.

I would like to express my special thanks of gratitude to Dr. rer. nat. Marco Seeland, who gave me the opportunity to do this wonderful project on the topic Analyzing Possible Influences on Class Activation Based Explainability Methods for Image Classifiers, which also helped me in doing a lot of research and I came to know about so many new things I am really thankful to them.

Abstract

Deep Learning models have been shown to be very effective on a variety of tasks, and have even beaten human experts. The lack of transparency of neural networks is a significant disadvantage, limiting their interpretability and practical applications. Recent studies have proposed methods based on the Class Activation Mapping approach, aiming to highlight the features that most influence a model's decision. However, there is no information on what factors influence the performance of explainability methods, how different regularization techniques affect the explainability performance and how the explainability performance correlates with classification accuracy.

This paper provides the results of a series of experiments using the state-of-the-art classification model with different Class Activation Mapping approaches to find out which of them performs best and how different training hyperparameters and regularization techniques affect explainability performance.

Based on the findings, Score-CAM outperforms CAM, Grad-CAM and Grad-CAM++ by up to 7% on IOU. Increasing the depth, epochs, and batch size leads to an increase in the explainability performance. Using regularization techniques like, Batch Normalization, Group Normalization and PRECODE techniques have a positive impact on the explainability performance compared to the baseline model, while models with Weight Decay, Dropout have a negative impact on the explainability. It was found that there is positive correlation between the classification and the explainability performance of the model in the experiments regarding depth and epochs. In the experiments with batch sizes, regularization techniques, and PRECODE, a trade-off between classification and explainability performance was observed. Therefore it should be considered to evaluate models in practice not only by accuracy metrics but also by explainability metrics.

Zusammenfassung

Deep-Learning-Modelle haben sich bei einer Vielzahl von Aufgaben als sehr effektiv erwiesen und sogar menschliche Experten übertroffen. Die mangelnde Transparenz neuronaler Netze ist ein erheblicher Nachteil, der ihre Interpretierbarkeit und praktischen Anwendbarkeit einschränkt. Jüngste Studien haben Methoden vorgeschlagen, die auf dem Class Activation Mapping-Ansatz basieren und darauf abzielen, die Merkmale, welche die Entscheidung eines Modells am meisten beeinflussen. Es gibt jedoch keine Informationen darüber, welche Faktoren die Leistung von Explainability-methode beeinflussen, wie verschiedene Regularisierungstechniken die Explainability-leistung beeinflussen und wie die Explainability-leistung mit der Klassifikationsgenauigkeit korreliert.

In diesem Arbeit werden die Ergebnisse einer Reihe von Experimenten vorgestellt, bei denen ein State-of-the-Art-Klassifikationsmodell mit verschiedenen Class-Activation-Mapping-Ansätzen verwendet wurde, um herauszufinden, welcher am besten abschneidet und wie sich verschiedene Trainingshyperparameter und Regularisierungstechniken auf die Explainability auswirken.

Die Ergebnisse zeigen, dass Score-CAM CAM, Grad-CAM und Grad-CAM++ um bis zu 7% bezüglich IOU übertrifft. Eine Erhöhung der Netzwerk-Tiefe, der Epochen und der Batch Größe führt zu einer Steigerung der Explainability-leistung. Die Verwendung von Regularisierungstechniken wie Batch-Normalisierung, Gruppe-Normalisierung und PRECODE haben im Vergleich zum Basismodell, während Modelle mit Weight Decay und Dropout einen negativen Einfluss auf die Explainability-leistung haben einen positiven Einfluss auf die Explainability-leistung. Es wurde festgestellt, dass es eine positive Korrelation zwischen der Klassifizierungs- und der Explainability-leistung des Modells in den Experimenten bezüglich Netzwerk-Tiefe und Epochenanzahl gibt. In den Experimenten mit Batch-größen, Regularisierungstechniken und PRECODE wurde ein Trade-off zwischen Klassifizierungs- und Explainability-leistung beobachtet. Daher sollte zukunftig in Betracht gezogen werden, Modelle in der Praxis nicht nur anhand von Genauigkeitsmetriken, sondern auch anhand von Explainability-metriken zu bewerten.

Declaration of Independent Creation

I assure that I have prepared this thesis without any unauthorized help from third parties and without using any other resources than those indicated. Data and concepts taken directly or indirectly from other sources are appropriately marked. The work has not been submitted to an examination authority in the same or similar form or published in any other way, either in Germany or abroad.

Russia, 23.10.2021 Roman Mysianov

Contents

1 Introduction

Deep Neural Networks are widely used in modern machine learning and artificial intelligence because of their high performance. Unfortunately, the popularity of neural networks for applications is not matched by a clear understanding of how they work. The field of neural networks will advance, as the trust to the neural networks will increase, if there will be a more comprehensive theory about how they work, and what parameters or techniques affect interpretability of the neural networks. In order to understand neural networks better, a number of researchers have proposed appropriate approaches and principled tools, but there are no studies about factors, that may have an impact on the explainability performance of the model.

1.1 Motivation

Convolutional Neural Networks (CNN) have shown their progress in extracting features from images and performing tasks such as image classification, image segmentation, object recognition, image captioning [STE13]. On one hand, these models have good performance, on the other hand, they have difficulties to be interpreted. Classical models based on rules such as Linear Regression, Random Forest, SVM and etc. have good interpretability but are not very accurate. The use of deep learning models achieves higher performance through greater abstraction, but sacrifices the interpretability of the models. Interpretability becomes even more important when deep learning models are used to make critical decisions. In the context of medical image diagnosis or insurance claim assessment, when the patient or customer needs an explanation of the decisions, that neural network models provide. Recent research has proposed some methods such as Class Activation Mapping (CAM) [MHLW19] that are suitable for the problem of explainability, but there is no information about what influences the performance of explainability, how different factors make the model more or less explainable. In this paper, we analyze possible influences on class activation-based explainability methods for image classifiers. Images are a domain where neural networks have become commonplace. Images are complex and high-dimensional data sources. Therefore, the learning process can be better understood when realistic images are used as the data source. In this paper, the analysis of class activation mapping methods is applied to CNN models trained on medical image data.

1.2 Task Description

This paper is dedicated to the investigation of which factors influence the explainability of ConvNet-based image classifiers. Methods based on the activation mapping approach will be described and evaluated by various explainability metrics. These methods produce visualizations that shall show area of pixels from the input image, that convolutional neural network uses to make a given prediction. A ResNet [KXSJ15] shall be used as a base model for binary classification. The goal of the work is to see how different hyperparameters such as network depth, epochs, batch size and techniques like Weight Decay, Dropout, Batch Normalization, Group Normalization, PRECODE impact on explainability performance of the neural network model. To evaluate the methods a practical dataset is used, which contains 7000 colored images of healthy and glaucoma eyes, and expert masks for every image. Most of the previous researches were limited to fully connected networks, toy problems or toy datasets and hence will be extended to a real medical problem using a state of the art CNN.

The goal of this thesis is to answer to the following questions:

1. How does network depth affect explainability performance?
2. How do hyperparameters like batch size, epochs affect explainability performance?
3. Which of the considered state of the art class activation mapping approaches performs best?
4. How do regularization techniques influence explainability performance?
5. How does PRECODE (sampled features from a constrained latent space) influence explainability performance?
6. How does CNN performance correlate with explainability results?

1.3 Thesis structure

A technical background to deep neural networks and regularization techniques is given in Chapter 2. The current state of the art explainability methods are described in Chapter 3. Chapter 4 explains the design of the experiments in this research. Chapters 5 provides and discusses the results of experiments with different hyperparameters such as network depth, epochs, batch size and techniques like Weight Decay, Dropout, Batch Normalization, Group Normalization, PRECODE. Chapter 6 concludes this work and offers directions for future research.

2 Technical Background

Deep neural networks are widely used parametric models for machine learning. The focus of this paper is on neural networks for images. Convolutional models usually improve the performance of neural networks for image related problems, since the image has a high dimensionality where each pixel represents a feature. CNN are very effective in reducing the number of parameters through filters without compromising the quality of the models. Realistic images provide a suitable problem domain for information theoretic analysis because they represent a complex and high-dimensional data source.

2.1 Deep Neural Networks

Neural networks are stacked multi layer processing machines. Each successive layer performs a linear mapping to its input, followed by a nonlinear activation function.

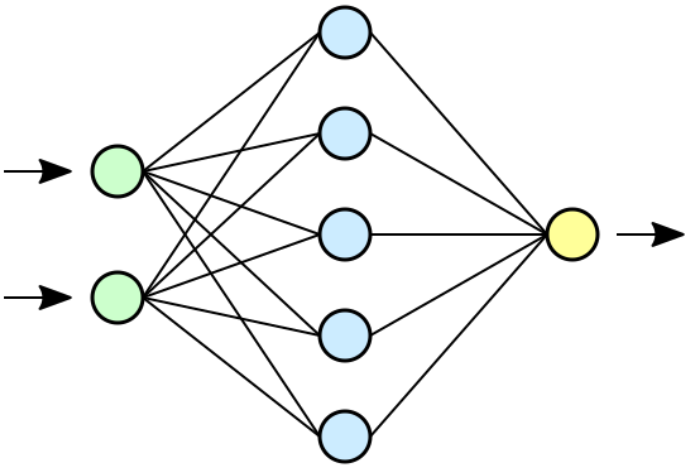


Figure 2.1: A simple neural network. The input of the green layer is processed through the hidden layer and then to the output layer from left to right by a weighted sum and activation functions [MHLW19].

Figure 2.1 illustrates the processing for a two-layer, fully connected neural network. One can imagine that each computation layer performs feature engineering on its inputs, re-

sulting in an internal hidden representation. Neural networks are optimized end-to-end; the resulting hidden representations are learned from the data instead of being created by hand. Stochastic gradient descent (SGD) through backpropagation is the standard neural network optimization technique [Bot10]. The gradient of the loss function is computed in order to update parameters. These updates - effectively the learning signal - are back-propagated through a neural network. The current state and relative performance of each layer affects the quality of the updates. Deeper models suffer more from gradient degradation because the learning signal to earlier layers is getting smaller due to its backpropagation through many layers. Some modern techniques alleviate the challenges associated with training very deep neural networks. Modern neural networks often have millions of parameters. Regularization combats overfitting in complex models, but does not fully contribute to generalization ability. Overfitting is the property of a model that the model predicts very well labels of examples used during training, but often makes errors when applied to examples that the learning algorithm did not see during training [Haw19]. The generalization ability of the network is mainly determined by the system complexity and the training of the network [NMZ+09].

2.1.1 Convolutional neural networks

Convolutional neural networks (CNN) have become popular in different computer vision tasks. CNNs use convolutional layers, pooling layers, and fully connected layers to learn spatial hierarchies of features [Mis19]. The preprocessing required in a ConvNet is much less compared to other classification algorithms. While in primitive methods the filters are created by hand, CNNs are capable of learning these filters/characteristics with sufficient training.

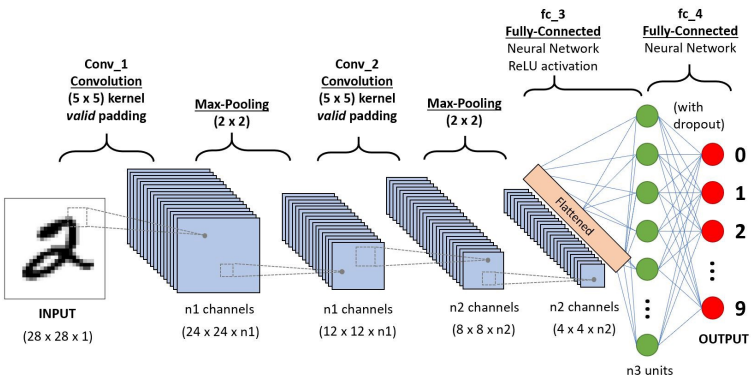


Figure 2.2: The architecture of a CNN [Mis19].

As can be seen from the figure 2.2, CNN uses filters to capture spatial and temporal

dependencies in an image. The architecture performs better adaptation to the image dataset by reducing the number of parameters involved and reusing the weights. In other words, the network can be trained to capture the complexity of the image [Ele20, Che19].

Convolution is repeated using the same kernel parameters for each unique location in the input. A "strided" convolution reduces the spatial size of the feature map by applying the convolution in steps larger than one. In this way, subsequent convolutions span a larger receptive field. Each resulting value passes through a non-linear function to construct the feature map.

2.1.2 Hidden Representations

Hidden Representations are part of feature learning and represent the machine-readable data representations learned by the hidden layers of a neural network. The output of an activated hidden node or neuron is used for classification or regression in the output layer. The representation of the input data, independent of the subsequent analysis, is called the hidden representation [MHN18]. The convolutional process if depicted on the figure 2.3:

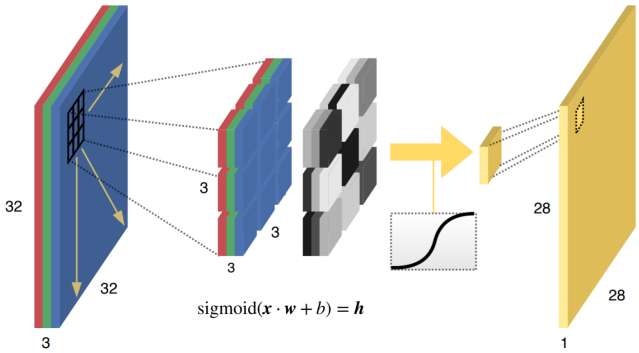


Figure 2.3: Convolution process [Mis19].

The performance of any Deep Learning method depends on the choice of data feature representation it works with. Much of the effort in deploying machine learning algorithms therefore goes into developing preprocessing pipelines and other data transformations, just to represent the data in the most efficient way for a machine to learn from. This feature engineering is critical, but labour intensive [LLL+19].

The need for human intervention in the representation framework highlights one of the major weaknesses of current learning algorithms: their inability to extract and organize the discriminative information from the data without outside help. But the use of CNN's eliminates the need for human feature engineering as it learn feature by themselves.

2.2 ResNet

As the number of layers in a Convolutional Neural Network increases, so does the model's ability to incorporate more complex functions. As a result, having more layers is always preferable. The backpropagation algorithm is used to update the weights of a neural network. The backpropagation algorithm makes a tiny change to each weight to reduce the model's loss. It updates each weight so that it moves in the direction of the decreasing loss. As a result, the gradient becomes lower and smaller, resulting in very little updates to the initial layers, which greatly increases training time [Sar19]. The ResNet architecture is depicted on the figure 2.4:

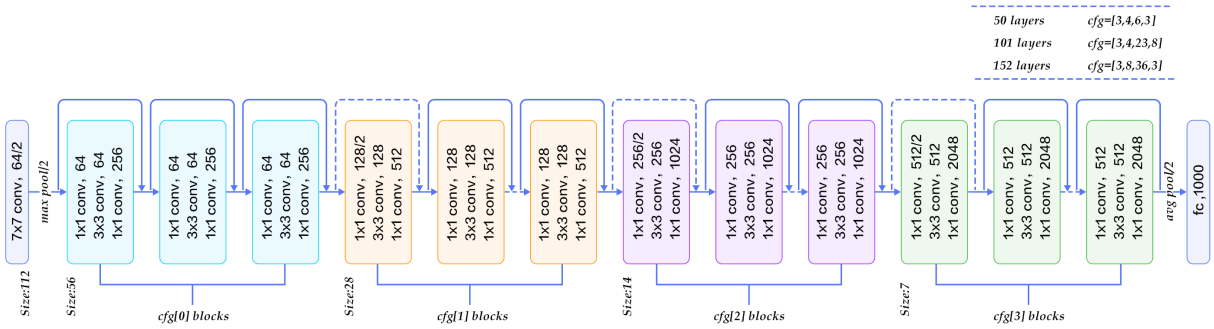


Figure 2.4: ResNet architecture [KXSJ15].

Typical ResNet models are built with double or triple layer skips that contain intermediate nonlinearities (ReLU) and stack normalization. To prevent vanishing of the gradient and accuracy saturation problem was presented a method of skipping connections. During training, the weights adjust to muffle the upstream layer and enhance the previously skipped layer [KXSJ15, HZRS16]. The illustration of the skipping layer is depicted in the figure 2.5.

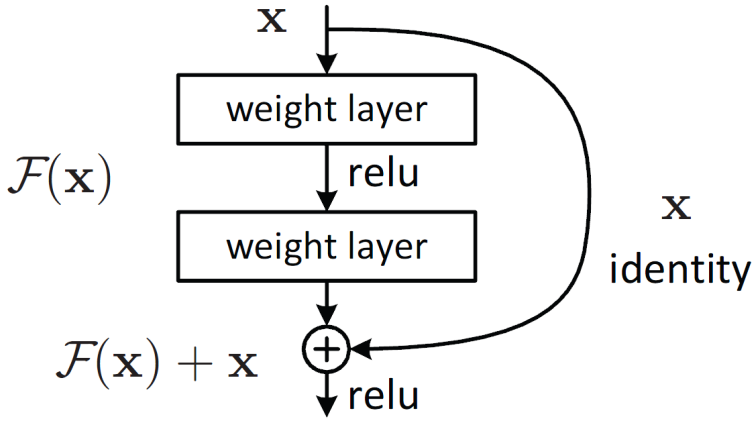


Figure 2.5: Illustration for one ResNet skip connection [HZRS16].

By utilizing fewer layers in the early training rounds, skipping effectively simplifies the network. Because there are fewer layers to travel, this speeds up learning by minimizing the influence of disappearing gradients. As it learns the feature space, the network eventually recovers the skipped levels [KXSJ15].

2.3 Regularization Techniques

Regularization used to address the issue of overfitting by reducing the generalization error without affecting the training error much. Regularization adds information to a model to prevent the occurrence of overfitting by reducing the variance of the model under consideration. The overfitting happens when model learns signal as well as noise in the training data and wouldn't perform well on new data on which model wasn't trained on.

Earlier research focused on finding means to make CNN's deeper since an increase in depth typically improves convolutional model performance. Other generalization advantages, such as speedier compression, make depth a favorable characteristic [KGC17].

Unfortunately, it is not possible to naively stack convolutional layers without greatly increasing the computational cost or making the network untrainable due to gradient backpropagation issues. Weight Decay, Dropout, Batch Normalization and Group Normalization allow training of deeper neural networks.

2.3.1 Weight Decay

Weight Decay is a regularization technique applied to the weights of a neural network. It minimizes a loss function that includes both the original loss function and a penalty. The L1 norm combats overfitting by shrinking the parameters towards 0. This makes

some features obsolete. L1 norm is represented in the formula 2.1, where w and $L_{original}$ denotes the weight matrix and original loss function, respectively:

$$L1_{new}(w) = L_{original}(w) + \lambda * |w| \tag{2.1}$$

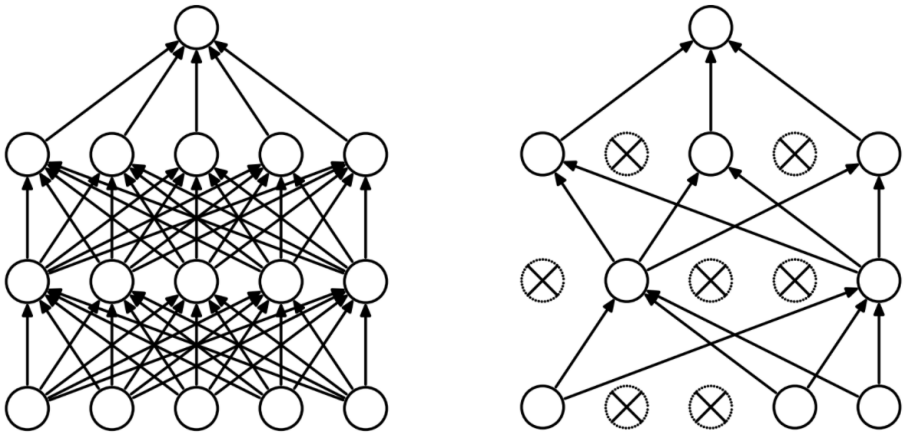
The L2 norm combats overfitting by making the weights small but not exactly 0. L2 norm is represented in the formula 2.2:

$$L2_{new}(w) = L_{original}(w) + \lambda * w^2 \tag{2.2}$$

Where λ is a value that determines the strength of the penalty (favoring smaller weights). Weight decay can be incorporated directly into the weight updating rule, rather than implicitly by defining it via the objective function [ZWXG18].

2.3.2 Dropout

Dropout is a powerful neural network training algorithm that reduces overfitting by disabling neurons with given probability during training process to avoid co-adaptation of feature detectors. The dropped units have no contribution of both passes through the network, forward and backward, respectively. It prevents overfitting and provides a way to efficiently combine approximately exponentially many different neural network architectures [SHK+14]. The main idea behind the Dropout is to train multiple neural networks and then average the results. The Dropout is used only during the training phase. The choice of which units are dropped out is random. The probability of removing units from neural network is fixed [LSV19]. The side effect of the dropout in CNN is, that it can increase the correlation between activations, that use the spatial relationships encoded in feature maps.



(a) Neural Network without Dropout (b) Neural Network with Dropout

Figure 2.6: Dropout illustration [Sri17].

The figure 2.6 demonstrates a neural network produced after applying dropout to the fully connected neural network. As can be seen, disabled neurons do not have any connections, since they do not ”exist” in the context of the training phase because of Dropout.

2.3.3 Batch Normalization

The Batch Normalization is a tool used to stabilize the neural network. It normalizes the output of the previous layer before it passes through the next layer. Without normalization, the model trains more slowly as the distribution of the data changes [BGS18]. To increase the stability of a neural network, batch normalization normalizes the output of a previous activation layer by subtracting the batch mean and dividing by the batch standard deviation. These regularizations improve the loss function and make the gradient more reliable [STIM19]. The loss of the non Batch Normalization network has a very wide range of values along the direction of the gradient, especially in the initial phases of training, when Batch Normalization network changes loss at a smaller rate and the magnitudes of the gradients are smaller too.

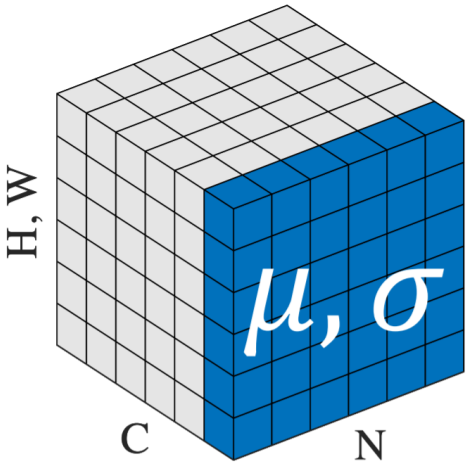


Figure 2.7: Batch Normalization illustration [IS15].

The figure 2.7 above illustrates Batch Normalization technique, where N is a batch size, C is a channel axis with width W and height H. The value of pixels in the batch are normalized using mean (μ) and variance (σ) computed across the batch [WH18].

2.3.4 Group Normalization

Group normalization is a normalization method that avoids stack dimension exploitation and is thus independent of stack size. Group normalization was proposed in March 2018 [WH18]. Group normalization optimizes the disadvantage that Batch Normalization does not work well for a small mini-batch. Group normalization is very similar to Batch Normalization, except that it divides the channels into groups and calculates the mean within each group for normalization. The group normalization calculation has nothing to do with batch size, so it is very stable for high precision images with small batch size [WH18].

For a regular stack size, group normalization performs comparably well to Batch Normalization and outperforms other normalization variants [WH18].

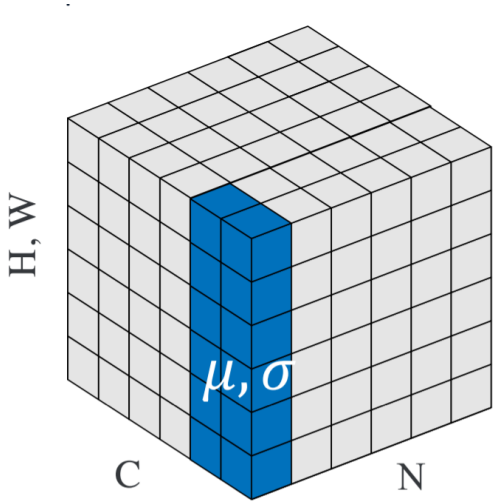


Figure 2.8: Group Normalization illustration [IS15].

The figure 2.8 illustrates Batch Normalization technique, where N is a batch size, C is a channel axis with width W and height H. The value of pixels in the groups are normalized using mean (μ) and variance (σ) computed across values in the groups [WH18].

2.4 PRivacy EnhanCing mODulE (PRECODE)

PRECODE is a privacy enhancing module that is realized with a Variational Bottleneck (VB). The VB principle defines an optimal representation and contains the most distinct information about input and output. Minimizing the term VB means that the information is restricted from the input data to the intermediate representation as well as to the output variable. Maximizing the cross entropy between the intermediate representation and the

output forces the information flowing from the input to predict the output. When these two processes occur simultaneously, the overall loss causes the model to focus only on the information related to the output in the input data [LXY21]. This may improve the generalization of the neural network and have a positive effect on the explainability performance.

VB provides a new representation from an approximated prior distribution using stochastic sampling. VB consists of a sequential encoder and decoder. The hidden representation generated by all layers of the CNN before the classifier serves as input to the encoder. The encoder then encodes the representation into a latent distribution. The number of neurons in the decoder corresponds to the number of features in the hidden representation. A new representation from the decoder is used to compute the model prediction [SMS21]. As the new representation, generated from VB, contains relevant information for model interest, it may have an impact on explainability. The PRECODE realization is depicted in the figure 2.9:

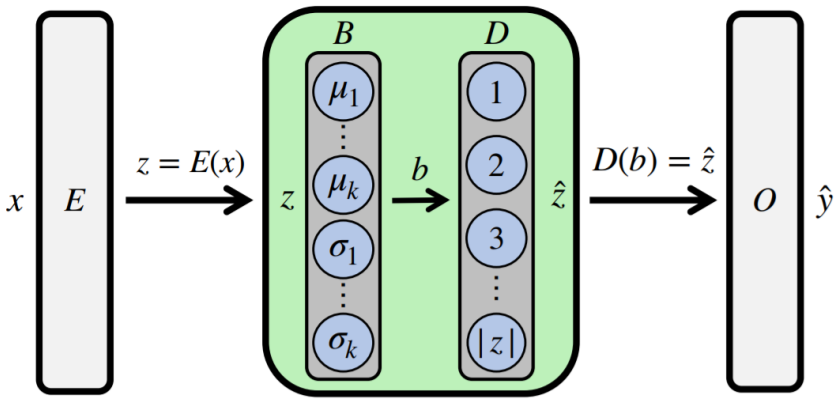


Figure 2.9: Realization of the PRECODE extension as variational bottleneck [SMS21].

It was found that PRECODE may have a regularization effect [SMS21], which will be examined in this work.

2.5 Metrics

The accuracy score, Confusion Matrix and Receiver Operating Characteristic are widely used metrics to evaluate classification performance of the model. The most suitable metrics to estimate explainability are the Intersection over the Union, the Average Drop and the Increase in Confidence.

2.5.1 Classification performance metrics

The confusion matrix is a table that summarizes how successful the classification model is in predicting examples that belong to different classes. In a binary classification problem, there are two classes. The notation that is used by the Confusion Matrix:

- True Positive (TP): A correct detection.
- False Positive (FP): A wrong detection.
- False Negative (FN): A ground truth not detected.
- True Negative (TN): A correct detection of the negative class.

You can see the example of Confusion Matrix for binary classification in figure 2.10:

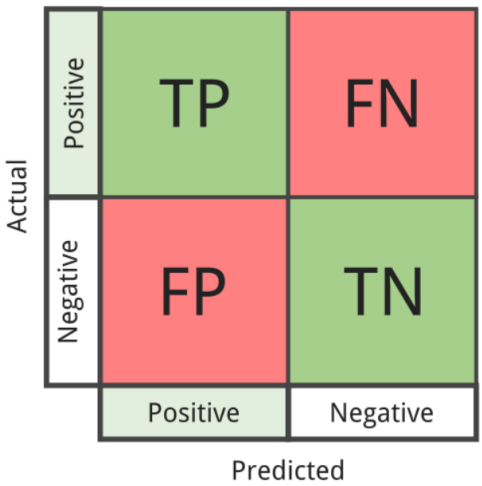


Figure 2.10: Confusion matrix [VRBPY19].

Accuracy is a metric for evaluating classification models. Informally, accuracy is the proportion of predictions that our model got right. Accuracy is given by the number of correctly classified examples divided by the total number of classified examples. Formally, accuracy has the following definition:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} \tag{2.3}$$

For binary classification, accuracy can also be calculated in terms of positives and negatives as follows:

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}} \tag{2.4}$$

Receiver Operating Characteristic (ROC) is one of the most commonly used metrics for classifier evaluation. ROC depicts how classifiers predict positive examples over negative examples [Adi18]. It shows the relationship between True Positive Rate and False Positive Rate for a given classifier.

- True Positive Rate (Recall or Sensitivity) is defined as

$$Recall = \frac{TP}{TP + FN} \tag{2.5}$$

- False Positive Rate is defined as

$$FalsePositiveRate = \frac{FP}{FP + TN} \tag{2.6}$$

2.5.2 Explainability performance metrics

Intersection Over Union (IOU) is used as a metric for explainability methods [PPD+21]. IOU is a measure based on Jaccard Index that evaluates the overlap between two bounding boxes. It requires a ground truth bounding box B_{gt} and a predicted bounding box B_p. A perfect match exists if $IOU = 1$. If the two bounding boxes do not intersect, $IOU = 0$. The general formula of IOU is described below:

$$IOU = \frac{B_p \bigcap B_{gt}}{B_p \bigcup B_{gt}} \tag{2.7}$$

The figure 2.11 demonstrates the calculation of the IOU:

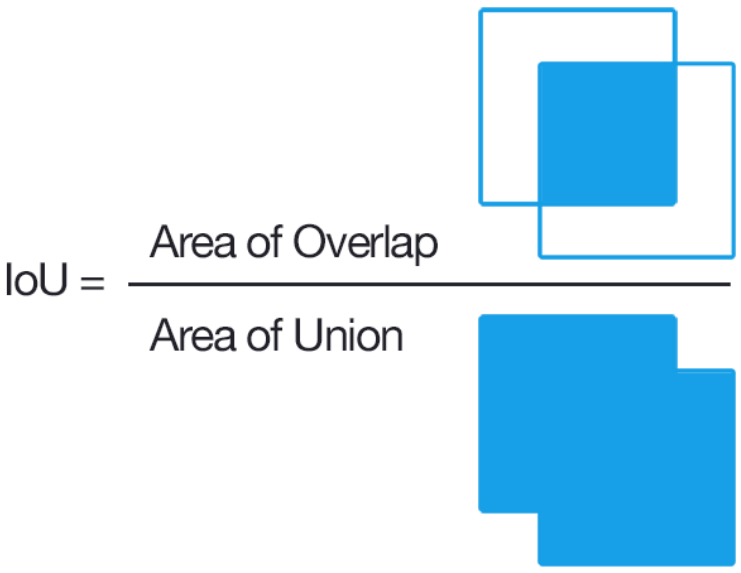


Figure 2.11: Computing the Intersection over Union [Cza21]

In this work we extend IOU metric by calculating not only intersection over the union between bounding boxes, but between ground truth area that is described through a mask and the the heatmap provided by explainability method.

$$IOU = \frac{Mask \bigcap Heatmap}{Mask \bigcup Heatmap} \tag{2.8}$$

As IOU metric requires binarized input, and heatmaps, provided by explainability methods, and masks provided by the dataset are not binary initially, binarization of the heatmaps and masks are described in chapter 4.1.

Faithfulness evaluation metric measures the feature importance provided by an explanatory method. The faithfulness evaluation consists of two metrics, which are denoted as Average Drop and the Increase in Confidence [CSHB17, WDYZ19].

To measure the Average Drop and the Increase in Confidence, the heatmap, provided by the explainability method, is used as an input image to the neural network. The Average Drop notes the change in confidence between the setting when the full image is provided as input, and the setting when only the explanation maps are provided as input. The Increase in Confidence is complementary to the Average Drop. It would be expected that there must be scenarios where providing only the explanation map region as input (instead of the full image) rather increases the confidence in the prediction (especially when the context is distracting).

The Average Drop is expressed as

$$\sum_{i=1}^{N} \frac{max(0, Y_i^c - O_i^c)}{Y_i^c} * 100 \tag{2.9}$$

The Increase in Confidence (also denoted as Average Increase) is expressed as

$$\sum_{i=1}^{N} \frac{Sign(Y_i^c < O_i^c)}{N} \tag{2.10}$$

where Y_i^c model's output score (confidence) for class c on image i and O_i^c is the same model's confidence in class c with only the explanation map region as input. $Sign$ presents an indicator function that returns 1 if input is True and 0 if False.

Summary

This chapter gave a technical background on deep neural networks and explained the widely used convolutional architecture and modern techniques typically used to enable better, faster, more stable training and achieve appropriate convergence in deep neural networks. The main idea behind ResNet architecture and residual connections was provided and explained. The ResNet uses skip connections to avoid vanishing gradient

problems, that can occur with an increase of neural network depth. Regularization techniques such as Weight Decay, Dropout, Batch Normalization and Group Normalization are studied in this work to obtain influences of explainability methods were described. A brief description of the metrics used to quantify the performance of explainability methods and neural networks was outlined. The accuracy score, Confusion Matrix and ROC metrics will be used to evaluate classification performance of the neural network. IOU, The Average Drop and the Increase in Confidence will be used as a metrics for explainability performance. In this work we extend IOU metric by calculating not only intersection over the union between bounding boxes, but between ground truth area that is described through a mask and the the heatmap provided by the explainability method. The following chapter describes state of the art explainability techniques: the Mutual Information approach, which measures the interdependence of two random variables, the Counterfactual Explanation method explains predictions of individual instances, the methods based on Class Activation Mapping, which provide an area of model interest by using feature maps from the last convolutional layer.

3 Class Activation Mapping In Neural Networks

In this work, we want to train a binary image classifier on an eye disease dataset and examine how different factors affect the explainability performance of the neural network. Multiple explanation methods for image classification have been proposed in the literature. For example, Class Activation Mapping, Counterfactual Explanation [VM20] and techniques based on Mutual Information (MI) [BGS18].

3.1 Non Class Activation Mapping Based Methods

3.1.1 Mutual Information

Mutual Information (MI) measures the interdependence of two random variables. It also quantifies the amount of information obtained about one random variable by observing the other. The following formula 3.1 shows the calculation of mutual information for two discrete random variables.

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p_{x,y} \log \frac{p(x,y)}{p(x)p(y)} \tag{3.1}$$

Here p_x and p_y are the marginal probability density functions and $p_{x,y}$ is the joint probability density function. Accurate and tractable estimation of information sets are an open question when applying information theoretic concepts to convolutional neural networks [YJP18]. Therefore, the quantity we really need to measure is the multivariate mutual information (MMI) between a single variable and a group of variables (e.g., different feature maps). This makes the calculation even more difficult and thus impractical for use in practical scenarios [FW18].

3.1.2 Counterfactual Explanation

Counterfactual explanations can be used to explain predictions of specific occurrences in interpretable machine learning. The ”event” is the anticipated result of an instance, and the ”causes” are the specific feature values of that instance that were fed into the model and ”caused” a certain prediction. The link between the inputs and the prediction

is extremely straightforward when depicted as a diagram: the feature values cause the prediction.

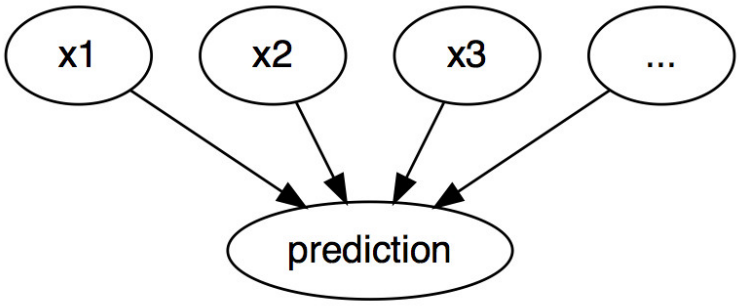


Figure 3.1: The inputs cause the prediction.

On the figure 3.1, [x1, x2, x3, ...] represent a series of the events, that cause the prediction. As observed in the literature, methods based on counterfactual explanations have a problem in interpreting image classifiers [VM20]. For each instance, there can be multiple counterfactual explanations (Rashomon effect), causing the same event to be interpreted in very different ways by different people [Ver20]. That makes the explainability tasks complicated and requires additional supervision by a human expert.

3.2 Class Activation Mapping

Class Activation Mapping (CAM) is a technique, that uses activation maps from the last convolutional layer and global average pooling (GAP) layer to obtain an area, that corresponds to the model interest. That requires a specific architecture to the model, especially a global average pooled layer. The final classification score y^c for a particular class c can be written as a linear combination of its global average pooled last convolutional layer feature maps A_k .

$$y^c = \sum_k w_k^c \sum_i \sum_j A_{ij}^k \tag{3.2}$$

Each spatial location (i, j) in the class-specific saliency map L^c is then calculated as:

$$L_{i,j}^c = \sum_k w_k^c A_{ij}^k \tag{3.3}$$

CAM has two distinct disadvantages: First, the network architecture is restricted to have a global average pooling layer after the final convolutional layer, and then a linear layer.

For all other networks, the structure must be changed and the network must be re-trained under the new architecture. Second, the method, which is limited to visualising only the last convolutional layers of a CNN, is only useful in interpreting the very last stages of image classification of the network, and it is not able to provide insight into the previous stages [SDV+16].

3.3 Gradient Class Activation Mapping

To prevent those disadvantages, Grad-CAM was introduced [CSHB17]. Grad-CAM can be applied to any architecture [Che19].

To construct a heatmap, Grad-CAM uses the gradient of the last convolutional layer before the classifier. A Rectified Linear Unit ($ReLU$) is a function, that get rids of negative values of feature maps, because there is interest in positive effects of maps. To obtain the importance of neuron weights w_k^c, Grad-CAM computes gradients of the feature maps for the score class [SDV+16] as depicted in the formula 3.4.

$$w_k^c = \frac{1}{Z} \sum_i \sum_j \frac{\delta y^c}{\delta A_{i,j}^k} \tag{3.4}$$

The number of pixels in activation map is denoted as Z. The weights w_k^c represent a partial linearization of the deep network downstream from A, and capture the 'importance' of feature map k for a target class c [ZWXG18].

$$L_{i,j}^c = ReLU(\sum_k w_k^c * A_{i,j}^k) \tag{3.5}$$

A $ReLU$ is very important operation, since negative pixels belongs to other classes. As expected, without this $ReLU$, localization maps sometimes highlight more than the class of interest and perform worse in localization [SDV+16, WDYZ19].

3.4 Gradient Class Activation Mapping++

For single object pictures, Grad-CAM heatmaps frequently do not capture the full item in its entirety, which is required for improved performance on the related identification task. Grad-CAM cannot locate objects in an image if there are numerous occurrences of the same class [WDYZ19, CSHB17]. Grad-CAM heavily relies on an object's spatial footprint in an image. As a result, if there are numerous occurrences of an item with slightly varied orientations or perspectives, various feature maps with varying spatial footprints may be triggered, and feature maps with smaller footprints may fade in the final saliency map. This issue can be resolved by calculating a weighted average of the pixel-wise gradients [CSHB17]. In particular, one can reformulate 3.4 by explicitly

encoding the structure of the weights w_k^c as:

$$w_k^c = \sum_i \sum_j \alpha_{ij}^{kc} ReLU \frac{\delta y^c}{\delta A_{i,j}^k} \tag{3.6}$$

The α_{ij}^{kc} is weighting co-efficient for the pixel-wise gradients for class c and convolutional feature map A_k [LCY13].

Grad-CAM++ derives a method for obtaining the gradient weights α_{ij}^{kc} for a particular class c and activation map k. Let y^c be the predicted score of the model for given input of a particular class c. Combining 3.2 and 3.6, we get:

$$y^c = \sum_k \sum_a \sum_b \alpha_{ab}^{kc} ReLU \frac{\delta y^c}{\delta A_{ab}^k} (\sum_i \sum_j A_{ij}^k) \tag{3.7}$$

Here, (i, j) and (a, b) are iterators over the same activation map A_k. Taking partial derivative w.r.t. A_{ij}^k on both sides:

$$\frac{\delta y^c}{\delta A_{ij}^k} = \sum_a \sum_b \alpha_{ab}^{kc} \frac{\delta y^c}{\delta A_{ab}^k} + \sum_a \sum_b A_{ab}^k (\alpha_{ij}^{kc} \frac{\delta^2 y^c}{(\delta A_{ij}^k)^2}) \tag{3.8}$$

Taking a further partial derivative w.r.t. A_{ij}^k :

$$\frac{\delta^2 y^c}{(\delta A_{ij}^k)^2} = 2\alpha_{ij}^{kc} \frac{\delta^2 y^c}{(\delta A_{ij}^k)^2} + \sum_a \sum_b A_{ab}^k (\alpha_{ij}^{kc} \frac{\delta^3 y^c}{(\delta A_{ij}^k)^3}) \tag{3.9}$$

Rearranging terms, we get:

$$\alpha_{ij}^{kc} = \frac{\frac{\delta^2 y^c}{(\delta A_{ij}^k)^2}}{2\frac{\delta^2 y^c}{(\delta A_{ij}^k)^2} + \sum_a \sum_b A_{ab}^k (\frac{\delta^3 y^c}{(\delta A_{ij}^k)^3})} \tag{3.10}$$

Substituting 3.10 in 3.6, we get the following GradCAM++ weights:

$$w_k^c = \sum_i \sum_j (\frac{\frac{\delta^2 y^c}{(\delta A_{ij}^k)^2}}{2\frac{\delta^2 y^c}{(\delta A_{ij}^k)^2} + \sum_a \sum_b A_{ab}^k (\frac{\delta^3 y^c}{(\delta A_{ij}^k)^3})}) ReLU \frac{\delta y^c}{\delta A_{i,j}^k} \tag{3.11}$$

Figure 3.2 shows the intuition behind the Grad-CAM++. Clearly taking a weighted combination of gradients Grad-CAM++ provides better salient features, than its unweighted counterpart Grad-CAM.

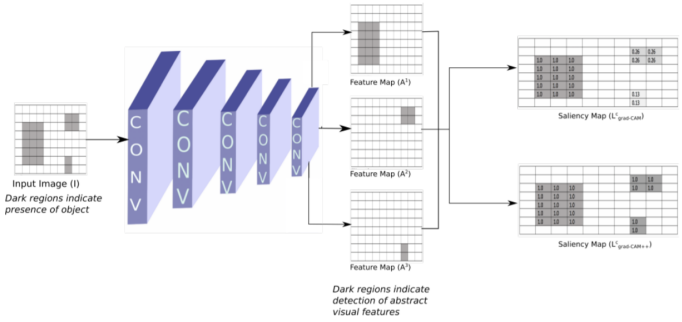


Figure 3.2: Grad-CAM++ [CSHB17].

The figure 3.3 below demonstrates the difference between three CAM methods (CAM, Grad-CAM, Grad-CAM++) [CSHB17]. Overview of three CAM methods, as can be seen base CAM requires a GAP layer, to provide heatmap of model interest, while Grad-CAM and Grad-CAM++ uses gradients from last convolutional layer.

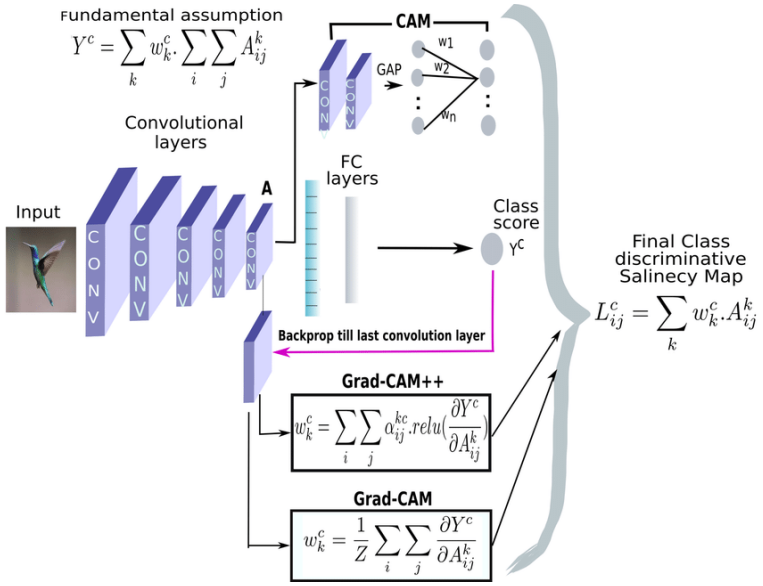


Figure 3.3: Difference between CAM, Grad-CAM, GradCAM++ [CSHB17].

3.5 Score Class Activation Mapping

Since Grad-CAM and Grad-CAM++ use gradients to produce heatmap, the gradients in deep neural networks may be noisy and also tend to vanish due to the saturation problem [WDYZ19].

Score Class Activation Mapping (Score-CAM) is a gradient-free visual explanation method based on class activation mapping. Score-CAM works by Channel-wise Increase in Confidence (CIC) to measure the importance of each activation map [WWD+20]. Score-CAM picks an internal convolutional layer l in model f and the corresponding activation as A. The k is a channel of A_l and denoted as A_l^k For a known baseline input X_b, the contribution A_l^k towards Y is defined as

$$C(A_l^k) = f(X \circ H_l^k) - f(X_b) \tag{3.12}$$

where

$$H_l^k = norm(Up(A_l^k)) \tag{3.13}$$

$Up(\cdot)$ denotes the operation that upsamples A_l^k into the input size. In order to a generate smoother mask H_l^k for an activation map, Score-CAM is using normalization function $norm(\cdot)$ described in equation 3.14, to normalizes the raw activation values in each activation map into [0, 1].

$$norm(A_l^k) = \frac{A_l^k - minA_l^k}{maxA_l^k - minA_l^k} \tag{3.14}$$

CIC first upsamples an activation map corresponding to a particular region in the original input space, and then perturbs the input with the upsampled activation map. The importance of this activation map is determined by the target value of the masked input. Each upsampled activation map not only represents the spatial locations most relevant to an internal activation map, but can also directly serves as a mask to perturb the input image. L^c for the Score-CAM can be defined as [WDYZ19].

$$L^c = ReLU(\sum_k \alpha_k^c A_l^k) \tag{3.15}$$

where

$$\alpha_k^c = C(A_l^k) \tag{3.16}$$

where $C(\cdot)$ denotes the CIC score of activation map A_l^k [Che19, WDYZ19]. In the figure 3.4 can be see the pipeline of the Score-CAM approach.

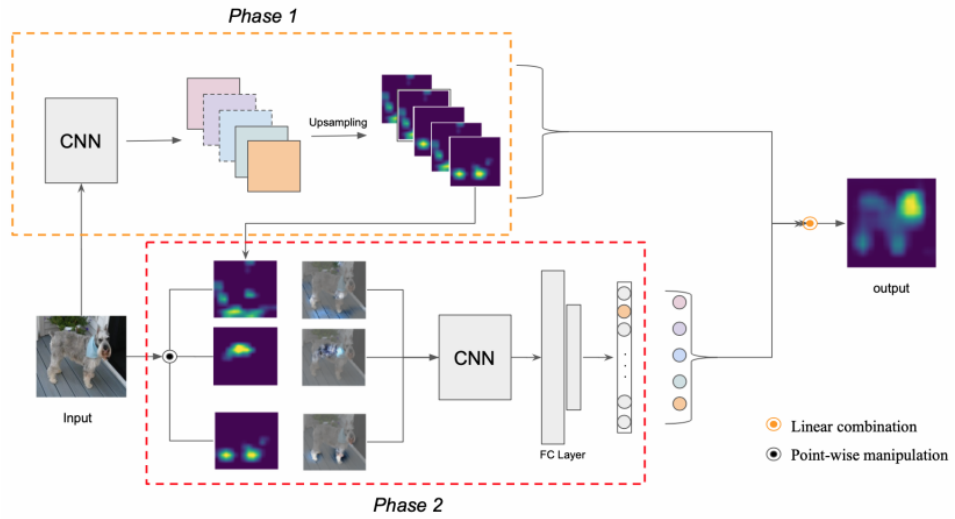


Figure 3.4: Pipeline of Score-CAM [WDYZ19].

An image is used as an input to the neural network producing feature maps. Each activation map will be resized to the original image size. After that Score-CAM calculates the score for each activation map. The heatmap will be obtained by linear combination of score-based weights and activation maps.

3.6 Comparison of the methods

The computation of mutual information for high-dimensional variables is a notoriously hard problem. The multivariate mutual information (MMI) between a single variable and a group of variables (e.g., different feature maps) is widely acknowledged as an intractable or infeasible task in machine learning and information theory communities, especially when each variable is in a high-dimensional space [YJP18]. Pioneering works in this direction focus either on small network models with discrete (continuous, eventually binned) activations [YJP18], or on linear networks [THKT20]. Therefore the use of Mutual Information explainability methods is infeasible for practical scenarios.

Another explainability method is Counterfactual Explanation. Counterfactual explanations can be used to explain predictions of individual instances. For each instance multiple counterfactual explanations might be found (Rashomon effect). This leads to an increase in complexity of the model explainability. After counterfactual explanations are generated for one instance, each explanation should be analyzed in order to get the relevant explanation for the model prediction [VM20]. This is also a practical challenge, that makes the explainability tasks complicated and requires additional supervision by a human expert. Another distinct drawback is that some of the methods based on the

counterfactual explanation require a specific model architectures [VDH20].

Compared to other state of the art explainability methods, CAM methods get rid of disadvantages such as computation time, specific model architecture, the complexity of explanation . In the papers Grad-CAM ++ [CSHB17] and Score-CAM [WDYZ19] were conducted a experiments with VGG16 model and ILSVRC2012 dataset to evaluate Grad-CAM, Grad-CAM++ and Score-CAM methods. A good explanation map for a class should highlight the regions that are most important for decision making. The Average Drop and The Increase In Confidence were used in the experiments [CSHB17, WDYZ19] as the metrics to measure the quality of the explainability methods. Evaluation results measured by explainability metrics for Grad-CAM, Grad-CAM++ and Score-CAM methods are depicted in the table 3.1. As can be seen Score-CAM achieves an average drop of 31.5% and an average increase of 30.6%, respectively, and far outperforms GradCAM and GradCAM++ methods [WDYZ19].

Method	GradCAM	GradCAM++	ScoreCAM
Average Drop	47.8%	45.5%	**31.5%**
Average Increase	19.6%	18.9%	**30.6%**

Table 3.1: Evaluation results (lower is better in Average Drop, higher is better in Average Increase).

It can be seen that the Score-CAM can distinguish different classes as shown in figure 3.5. The VGG-16 model classifies the input as ‘bull mastiff’ with 49.6% confidence and ‘tiger cat’ with 50.2% confidence.

Figure 3.5: Class discriminative result. The middle plot is generated w.r.t ‘bull mastiff’, and the right plot is generated w.r.t ‘tiger cat’ [WDYZ19].

As shown in this example below in figure 3.6, Grad-CAM only tends to focus on one object, while Grad-CAM++ can highlight all objects. Score-CAM further improves the quality of finding all evidences of target class. Compared to previous approaches, Score-CAM can not only accurately locate a single object, but also shows better performance in locating multiple objects of the same class than previous works. The result is shown in the figure 3.6, Grad-CAM [PVS20] tends to detect only one object in the image, Grad-CAM ++ [LSV19] and Score-CAM both show the ability to locate multiple objects, but the saliency maps of Score-CAM are more concentrated than Grad-CAM ++.

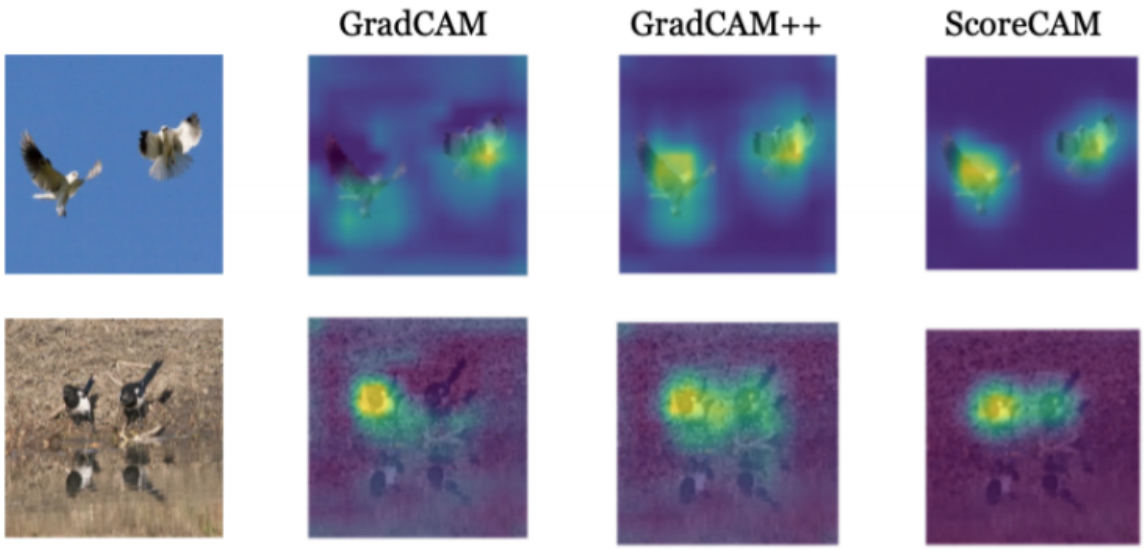


Figure 3.6: Results on multiple objects[WDYZ19].

To construct a heatmap of the neural network interest, Score-CAM calculates the score for all upsampled feature maps, provided by an input image. This can increase the computational cost of the method.

Summary

This chapter provides an overview over state of the are explainability methods. Compared to other state of the art explainability methods, CAM methods get rid of disadvantages such as computation time, specific model architecture, the complexity of explanation. When analyzing the methods based on CAM, Score-CAM was deemed to outperform all previous CAM-based methods and other state-of-the-art methods in detection and localization evaluation metrics [WDYZ19].

4 Design of the Experiments

This chapter introduces the design of the experiments to investigate possible influences on class activation based explainability methods for image classifiers in the context of realistic image data. For all the experiments will be used three different parameterization seeds and the mean metric values with variance will be reported. The accuracy score, confusion matrix, and AUC will be used to evaluate the classification performance of the model, and IOU by area and bounding boxes, the Average Drop and Increase in Confidence to evaluate explainability.

4.1 Dataset Preprocessing

The LAG dataset [LXW+19] contains fundus images corresponding to suspicious and negative glaucoma samples and masks of expert, as ground truth, which will be used for calculating IOU metric. All the samples are labeled with the diagnosis results (0 refers to negative glaucoma (healthy) and 1 refers to suspicious glaucoma). As can be seen in the table 4.1, dataset is split into train, test and validation sets:

Type	Train	Test	Validation
Glaucoma	2372	661	66
Healthy	2421	754	93

Table 4.1: Number of images from dataset used in train, test, and validation phases.

IOU metric requires binarized input. Heatmaps and masks are not binary initially. The heatmap, provided by the explainability method, and masks of experts, provided by the dataset, resized to an input image size, then binarized using Otsu Thresholding Algorithm [You15]. Otsu's thresholding method involves iterating through all the possible threshold values and calculating a measure of spread for the pixel levels each side of the threshold, i.e. the pixels that either fall in foreground or background. The aim is to find the threshold value where the sum of foreground and background spreads is at its minimum. The image examples of heatmap and mask before and after binarization provided in the figure 4.1

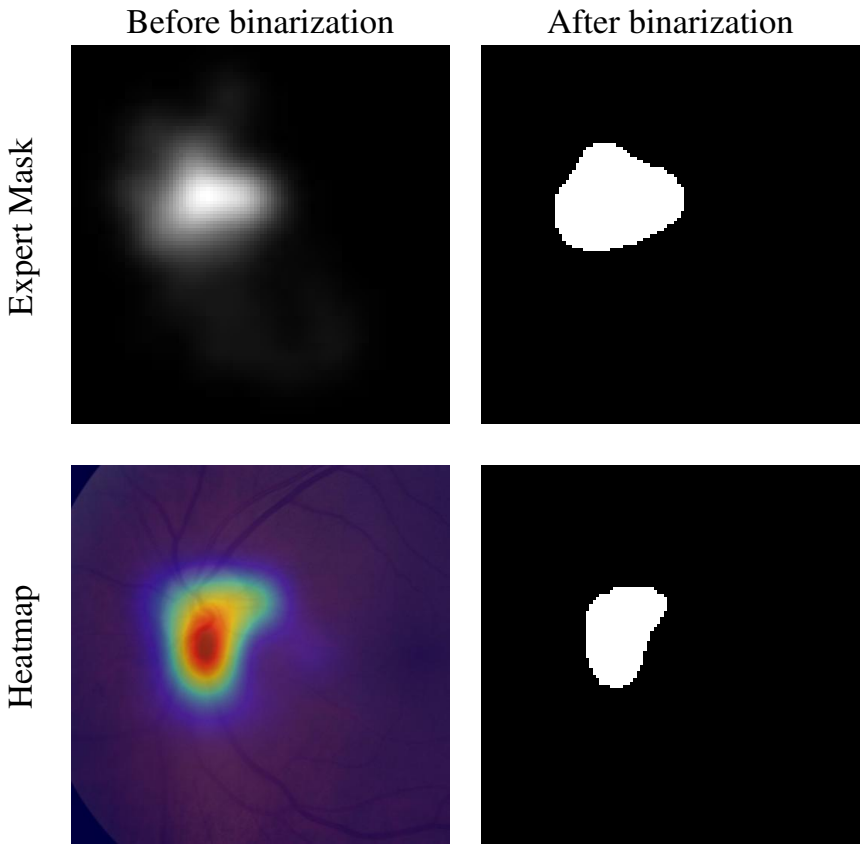
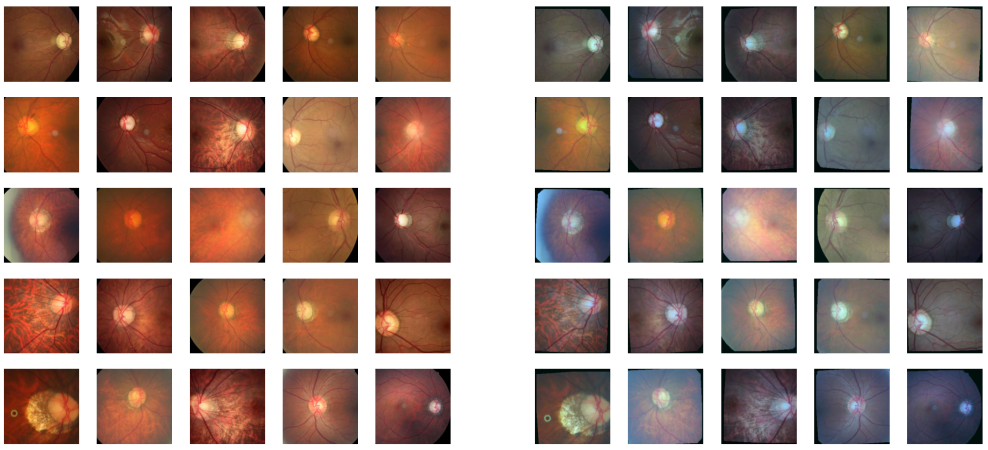


Figure 4.1: The image examples of heatmap and mask before and after binarization

4.2 Augmentation

Analyzing the medical imaging models it was found, that the next augmentation techniques have the most impact on accuracy of the model. According to the nature of the data, random crop, random rotation, flipping (horizontal) and normalization were the most suitable transformations for data augmentation. The choice of transformation techniques for augmentation that is used in this work is inspired by [SHK+]. The input images were resized to (224 x 224 x3). This paper worked with the medical images dataset and found that these transformations on this data resulted in good accuracy. The transforms operations are applied to original images at every batch generation. So dataset is left unchanged, only the batch images are copied and transformed every iteration.

Each input sample is a coloured image depicting an eye scan. On the figure 4.2 can be seen image examples before and after augmenting.



(a) Before augmentation (b) After augmentation

Figure 4.2: Random image examples from the dataset.

4.3 Neural Network Design

ResNet architecture was chosen as the baseline, but it has many variations of depth as can be seen in the figure 4.3. The ResNet architecture prevent vanishing of the gradient and accuracy saturation problem by using method of skipping connections described in section 2.2.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 4.3: ResNet Architectures

The increasing depth of the Neural Network model can capture more complex representations and provide more detailed activation maps, but it also leads to an increase

in computational cost. To answer research question 1 experiments with 18, 34, and 50 layer architectures were designed, and compared by explainability metrics and overall accuracy. We are not looking at 101 and 152 layer architectures, because they are to deep for given binary classification task [SHK+].

4.4 Optimization process

This section describes optimization algorithms and hyperparameters of the neural network. Learning rate, optimizer, and loss function are fixed for all experiments. To answer the research question 2 experiments where different hyperparameters like depth, epochs, and batch size were designed and investigated on the impact of the explainability method.

4.4.1 Learning rate

The appropriate choice of learning rate was critical for achieving rapid network convergence during training. The training took too long to converge if the learning rate was set too low, or the optimizer became stuck in local minima, preventing the loss function from being modified to generalize the network. The network did not always converge if the learning rate was set too high. To reduce the number of parameters to optimize for in all the experiments we are looking at, were conducted preliminary experiments with learning rate finder [Bev18] in which was found, that 10^{-3} is an optimal learning rate.

4.4.2 Optimizer

There are multiple optimization algorithms for neural networks, such as Root Mean Square Propagation (RMSprop), Adaptive Gradient Algorithm (Adagrad), ADAptive with Momentum (ADAM) and Stochastic Gradient Descent (SGD). Based on past research papers [CSHB17, WDYZ19], ADAM was chosen as an optimization algorithm, as it achieves better loss and leads to better explainability, than SGD and RMSprop [Fir].

ADAM, which stands for ADAptive with Momentum was chosen as an optimization algorithm [Bus18]. It calculates the individual adaptive learning rate for each parameter from estimates of first and second moments of the gradients. ADAM can be viewed as a combination of Adagrad, which works well on sparse gradients and RMSprop which works well in online and non stationary settings [Rud16]. ADAM implements the exponential moving average of the gradients to scale the learning rate instead of a simple average as in Adagrad. It keeps an exponentially decaying average of past gradients ADAM is computationally efficient and has very little memory requirement ADAM is used to change the attributes of neural network such as weights and learning rate in order to reduce the losses [Bus18, Rud16].

4.4.3 Loss function

Cross-entropy is commonly used in machine learning classification tasks. Since we consider, that task in this work is classification, we are using Cross-entropy which measures the difference between two probability distributions for a given random variable or set of events. Cross-entropy loss is used when adjusting model weights during training. The aim is to minimize the loss, i.e, the smaller the loss the better the model. A perfect model has a cross-entropy (ce) loss of 0 [Bus]. The loss can be defined as:

$$ce(y, class) = -log(\frac{e(y[class])}{\sum_j e(y[j])}) = -y[class] + log(\sum_j e(y[j])) \tag{4.1}$$

4.4.4 Epoch

An epoch means one complete pass of the training dataset through the deep learning model, each sample in the training dataset had the opportunity to update the internal model parameters. An epoch consists of one or more batches.

In the experiments we will investigate training progress for 100 epochs. An increase in the epochs may have a positive impact on the explainability of the model since a neural network has the ability to generalize better with more different examples are passed through the network since augmentation is used.

4.4.5 Batch Size

The number of samples from the dataset that will be propagated through the network is called batch size. The error gradient is a statistical estimate for a given batch. The more training examples used to estimate it, the more accurate the estimate and the more likely it is that the weights of the network will be adjusted to improve the performance of the model [Ben12, ML18]. The improved estimate of the error gradient comes at the cost of requiring many more predictions to be made with the model before the estimate can be computed and thus the weights can be updated [Bro19]. Alternatively, using fewer examples leads to a less accurate estimate of the error gradient, which is highly dependent on the specific training examples used [Bro19]. The small batch size may not represent the whole dataset, thus gradient estimation will be not correct.

By using a batch size lesser than number of all samples, it requires less memory, Since network is trained using fewer samples, the overall training procedure requires less memory [BGU21]. Typically networks train faster with mini-batches. That's because weights updated after each propagation [BGU21]. The smaller the batch the less accurate the estimate of the gradient will be [SKL17]. In the experiments we will investigate 16, 32, 64, 128, 256 batch sizes. Increase in the batch size may have a positive impact on the explainability of the model, since neural network use more examples from dataset in gradient estimation.

4.5 Comparison of the CAM methods

To answer research question 3 experiments with different CAM approaches, such as CAM, Grad-CAM, Grad-CAM++ and Score-CAM were designed. The baseline models with optimal hyperparameters will be provided for each CAM method in order to see which of the methods achieves better explainability performance.

4.6 Regularization techniques

After defining hyperparameters, the effect of regularization techniques on explainability of the model will be studied. The description of the investigated techniques can be found in section 2.3. The structure of experiments with regularization techniques is described below:

- Weight Decay with L1 & L2 norms, and different λ in range $[\ 10^{-3}, 10^{-4}, 10^{-5}\ ]$
- Dropout with dropout rates of 0.10, 0.33, 0.50 and 0.75.
- Batch Normalization.
- Group Norm with group sizes 16, 32 and 64.

4.7 PRECODE

In order to see impact on the explainability and answer research question 5 experiments with PRECODE technique with different sizes of bottleneck representation [128, 256, 512] and β in range $[\ 10^{-3}, 10^{-4}\ ]$ were designed.

For the PRECODE technique, the experiments not only on the impact of the explainability method but on the regularization effect will be conducted. The baseline model and model with PRECODE will be trained for 300 epochs in order to observe regularization effects.

Summary

This chapter provides the design of the experiments to investigate possible influences on class activation based explainability methods for image classifiers in the context of realistic image data. Neural network design and the description of the optimization process, such as learning rate choice, optimization algorithm, and loss function was provided. The description of the regularization techniques and hyperparameters, that will be used in the experiments, is introduced. Initial hyperparameters for the baseline model were chosen regarding similar state-of-the-art models [SHK+]. Learning rate 10^{-3}, ADAM

optimizer, Cross-entropy loss function and parameterization seeds are fixed for all experiments.

5 Experiments

This chapter provides the results of the experiments with hyperparameters and regularization techniques in order to observe possible influences on class activation based methods and answer the research question described in chapter 1. The data relating to the classification and explainability performance was obtained by evaluating the models on the validation dataset. In all the experiments the models are chosen by the lowest validation loss during the training phase.

The notation used in the table is described in table 5.1.

Table 5.1: Notation for the experiment tables.

Metric:	
Accuracy score	Acc
IOU (Area)	IOU (A)
IOU (Bounding Boxes)	IOU (BB)
Average Drop	AD
Increase in Confidence	IiC
Signs:	
Higher is better	$\uparrow$
Lower is better	$\downarrow$

5.1 Neural Network Design

In this section, we perform experiments described in 4.3, to observe the influence of neural network depth to explainability performance. For the experiments in this section, 16 batch sizes and 25 epochs were used.

An increase in depth on the neural networks gives more power to express more and more complicated functions. Three experiments with different depth were investigated. It was found that the architecture with 50 layers depth has the best classification accuracy and Score-CAM achieves better explainability performance compared to CAM, Grad-CAM and Grad-CAM++ as can be seen in table 6.1:

Depth		18	34	50
↑ Acc		82.20% ± 1.24%	83.50% ± 1.54%	**85.30% ± 1.20%**
↑ ROC AUC		78.00% ± 0.33%	80.00% ± 1.00%	**81.00% ± 1.00%**
↑ F1 score		83.00% ± 1.43%	85.10% ± 2.20%	**85.30% ± 1.54%**
↑ *IOU* (A)	CAM	3.50% ± 0.51%	3.70% ± 0.23%	**5.00% ± 1.10%**
	Grad-CAM	11.60% ± 0.21%	11.50% ± 0.62%	**16.30% ± 0.89%**
	Grad-CAM++	12.20% ± 1.01%	12.30% ± 0.42%	**17.50% ± 0.73%**
	Score-CAM	16.20% ± 1.10%	15.50% ± 0.92%	**21.00% ± 0.33%**
↑ *IOU* (BB)	CAM	3.90% ± 0.54%	3.80% ± 0.22%	**4.80% ± 0.92%**
	Grad-CAM	11.90% ± 0.30%	12.10% ± 0.57%	**15.90% ± 0.93%**
	Grad-CAM++	12.20% ± 1.01%	12.30% ± 0.42%	**15.50% ± 0.73%**
	Score-CAM	14.70% ± 0.94%	13.90% ± 1.03%	**16.70% ± 0.87%**
↓ AD	CAM	92.00% ± 1.37%	88.10% ± 0.86%	**87.40% ± 0.90%**
	Grad-CAM	67.10% ± 1.72%	67.30% ± 1.34%	**65.50% ± 0.96%**
	Grad-CAM++	66.70% ± 1.83%	66.50% ± 1.61%	**66.00% ± 1.66%**
	Score-CAM	56.00% ± 0.57%	60.30% ± 1.02%	**52.20% ± 0.95%**
↑ IiC	CAM	0.00% ± 0.00%	0.00% ± 0.00%	**0.10% ± 0.01%**
	Grad-CAM	2.20% ± 0.22%	2.50% ± 0.41%	**2.60% ± 0.34%**
	Grad-CAM++	2.70% ± 0.20%	2.60% ± 0.31%	**2.80% ± 0.40%**
	Score-CAM	4.80% ± 0.44%	3.70% ± 0.51%	**5.60% ± 0.49%**

Table 5.2: Layer depth evaluation.

As can be seen from the table, 50 depth has higher overall accuracy by accuracy score, AUC and F1 metrics. It also outperforms other depths by both *IOU* metrics, bounding boxes, and area respectively, and has better score on Average Drop and Increase in Confidence. With more layers model can capture more complex representation. With more complex representations explainability of the models is also increasing. It can be explained by the fact, that activation maps produced by the model with higher depth, are less noisy and represents model interest better. For the next experiments model with 50 depth will be used as the baseline model, since it achieves better explainability performance.

5.2 Optimization process

In this section, we perform experiments described in 4.4, to chose what hyperparameters combination will be used for all the further experiments.

5.2.1 Epoch

During the training phase, the aim is to minimize the error rate as well as to make sure that the model generalizes well on new test data. A good model is expected to capture the underlying structure of the data. In other words, it should not overfit or underfit. The

experiment was executed with 16 batch size, in order to get the influence of epochs on explainability. The results of training for 100 epochs can be seen in the table 5.3:

↑ Acc		88.50% ± 1.36%
↑ ROC AUC		88.00% ± 1.00%
↑ F1 score		92.30% ± 1.28%
↑ *IOU* (A)	CAM	15.80% ± 0.52%
	Grad-CAM	18.30% ± 0.69%
	Grad-CAM++	19.50% ± 0.44%
	Score-CAM	25.10% ± 0.62%
↑ *IOU* (BB)	CAM	14.20% ± 0.86%
	Grad-CAM	15.90% ± 0.93%
	Grad-CAM++	18.90% ± 0.53%
	Score-CAM	22.10% ± 0.87%
↓ AD	CAM	66.40% ± 0.65%
	Grad-CAM	62.50% ± 0.84%
	Grad-CAM++	63.00% ± 1.02%
	Score-CAM	48.90% ± 0.95%
↑ IiC	CAM	4.30% ± 0.98%
	Grad-CAM	4.70% ± 1.00%
	Grad-CAM++	5.00% ± 0.86%
	Score-CAM	6.90% ± 0.49%

Table 5.3: Results of the training for 100 epochs.

On the figure 5.1 can be observed the progress of the model during the training by classification and explainability metrics.

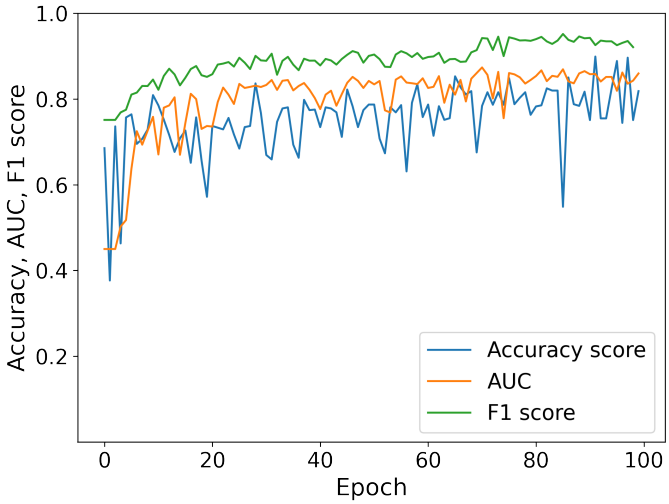
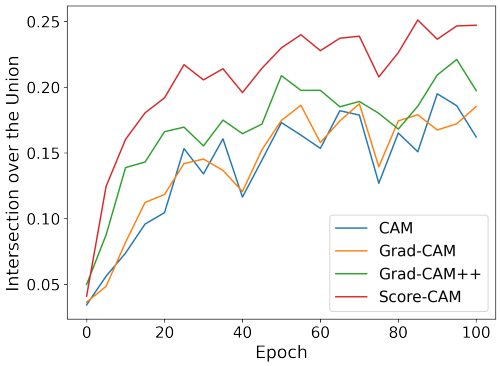


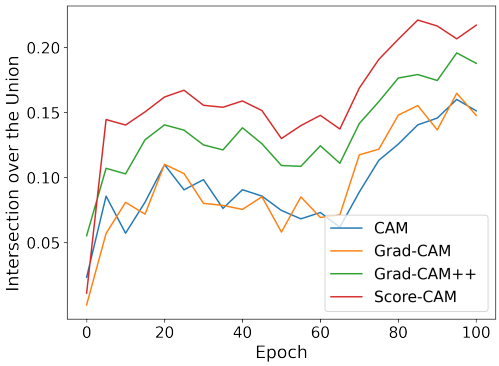
Figure 5.1: Progress of the model during the training by classification metrics.

As can be seen in the graph 5.1, there is an increase in classification performance after training for 70 epochs. The training for more than 100 epochs is not necessary, as classification performance does not change.

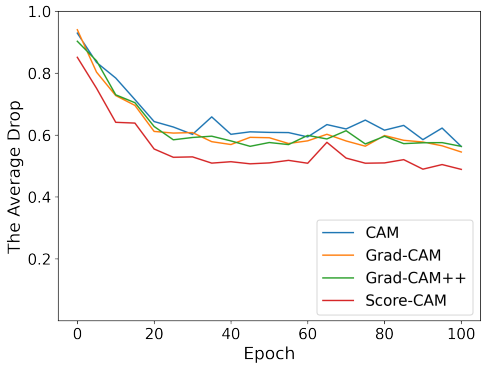
On the figure 5.2 can be observed the progress of the model during the training by explainability metrics.



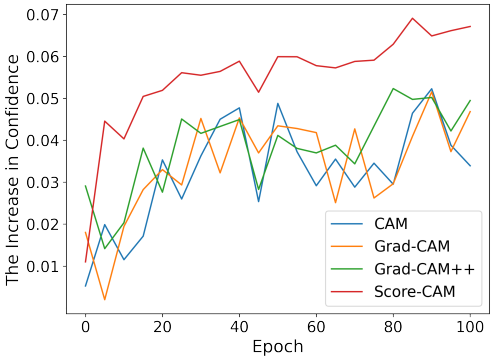
(a) Intersection over the Union (Area)



(b) Intersection over the Union (Bounding Boxes)



(c) The Average Drop



(d) The Increase in Confidence

Figure 5.2: Progress of the model during the training by explainability metrics.

As can be seen in the graph 5.2, there was an increase in the explainability performance after training for 70 epochs, especially in IOU (Bounding Boxes) metrics. The training for more than 100 epochs is not necessary, as the explainability performance does not increase.

As can be seen from the table, training for 100 epochs have increased overall classification performance by accuracy score, AUC, F1 and have better explainability performance by both IOU metrics and have better scores on Average Drop and Increase in Confidence for all explainability methods. Increasing the epochs number gives the model the ability to have better generalization, as it trains the model longer, since Gradient Descent is an iterative process, with more time for find an optimization minimum, model performs better, but if neural network trained too long it may overfit, which leads to worse generalization. That effect did not appear in our experiments. As was noticed in the experiments, an increase of epochs leads to an increase in the explainability of the model. But at some point, the further increase in epochs not necessary, as the classification and explainability performance does not change, 100 epochs will be used for

the next experiments, since it achieves better explainability performance and training for even longer has no noticeable impact.

5.2.2 Batch Size

The batch size defines the number of samples that will be propagated through the network. The more training examples used in the estimate, the more accurate this estimate will be and the more likely that the weights of the network will be adjusted in a way that will improve the performance of the model. Batch size with level values 16, 32, 64, 128, 256 were examined. The results of the experiments can be seen in the table 5.4:

	Batch size		16	32	64	128	256
↑	Acc		88.50% ± 1.36%	87.80% ± 0.75%	90.10% ± 1.23%	89.40% ± 1.01%	**91.10% ± 0.83%**
↑	ROC AUC		88.00% ± 0.33%	89.00% ± 1.00%	85.00% ± 0.33%	90.00% ± 0.33%	**91.00% ± 1.00%**
↑	F1 score		92.30% ± 1.28%	91.10% ± 0.54%	93.60% ± 1.79%	91.70% ± 1.89%	**94.00% ± 1.09%**
↑	*IOU* (A)	CAM	15.80% ± 0.52%	15.70% ± 0.43%	14.10% ± 1.10%	16.20% ± 0.86%	**17.20% ± 1.12%**
		Grad-CAM	18.30% ± 0.69%	12.50% ± 0.66%	16.30% ± 1.19%	20.4% ± 0.98%	**20.30% ± 0.58%**
		Grad-CAM++	18.90% ± 0.53%	13.10% ± 0.42%	18.20% ± 0.53%	21.20% ± 1.09%	**20.90% ± 0.73%**
		Score-CAM	25.10% ± 1.10%	19.10% ± 0.33%	22.10% ± 1.27%	17.50% ± 0.92%	**27.80% ± 0.33%**
↑	*IOU* (BB)	CAM	14.20% ± 0.86%	16.80% ± 0.97%	15.10% ± 0.75%	15.90% ± 1.15%	**18.50% ± 0.75%**
		Grad-CAM	18.30% ± 0.69%	14.10% ± 0.57%	17.90% ± 0.93%	22.20% ± 0.84%	**22.10% ± 0.65%**
		Grad-CAM++	19.50% ± 0.44%	14.80% ± 0.95%	15.50% ± 0.86%	**22.50% ± 1.13%**	20.90% ± 0.81%
		Score-CAM	22.10% ± 0.87%	14.70% ± 1.03%	15.80% ± 0.78%	15.80% ± 1.22%	**39.20% ± 0.69%**
↓	AD	CAM	66.40% ± 0.65%	70.10% ± 0.86%	65.40% ± 0.90%	66.20% ± 0.82%	**62.10% ± 1.01%**
		Grad-CAM	62.50% ± 0.84%	67.30% ± 1.34%	65.80% ± 0.91%	**58.50% ± 1.16%**	59.40% ± 0.89%
		Grad-CAM++	63.00% ± 1.02%	67.50% ± 1.21%	69.00% ± 1.62%	**58.40% ± 1.31%**	59.00% ± 1.06%
		Score-CAM	48.90% ± 0.95%	60.20% ± 1.02%	52.10% ± 1.02%	54.80% ± 1.02%	**44.50% ± 0.95%**
↑	IiC	CAM	4.30% ± 0.98%	4.70% ± 0.65%	5.10% ± 1.08%	5.10% ± 0.82%	**5.40% ± 0.61%**
		Grad-CAM	4.70% ± 1.00%	4.20% ± 1.17%	4.40% ± 1.29%	6.10% ± 1.06%	**5.70% ± 0.34%**
		Grad-CAM++	5.00% ± 0.47%	4.40% ± 1.31%	4.70% ± 1.22%	**6.20% ± 0.31%**	5.80% ± 0.40%
		Score-CAM	6.90% ± 0.49%	5.00% ± 1.57%	5.60% ± 1.01%	4.90% ± 0.99%	**8.20% ± 0.49%**

Table 5.4: Images examples for different batch sizes of the model

As can be seen from and table above, batch size 256 has 91.1% accuracy score, 91% AUC, 94% F1 score and outperforms other batch sizes by CAM and Score-CAM by explainability metrics (area and bounding boxes intersection). Score-CAM has better explainability score over all other methods by explainability metrics(area and bounding boxes intersection) with 27.8% and 39.2% respectively, also has an better score by Average Drop and Increase in the Confidence metrics. But batch size 128 has slightly better explainability performance by Grad-CAM and Grad-CAM++ methods.

5.3 Comparison of the CAM methods

As can be seen from the experiments above Score-CAM has better performance by all explainability metrics. Although Score-CAM takes significantly more evaluation time, because of forward passing each activation map to model, Score-CAM localizes glaucoma better than CAM, Grad-CAM++ and Grad-CAM. It was confirmed that Score-CAM performs better, as other research before has already observed [WDYZ19]. Based

on findings, Score-CAM has a better explainability score, lower Average Drop, and higher Increase in Confidence, compared to other CAM approaches. This can be explained by the fact, that gradient in deep models might be noisy. For all upcoming experiments Score-CAM will be used as the investigated explainability method.

5.4 Regularization techniques

Summarizing the finding from the above experiments the baseline model and training hyperparameters for all the following experiments were chosen as can be seen in the table 5.5.

Epochs	Depth	Batch Size	Optimizer	Loss function	Method
100	50	256	ADAM	Cross Entropy Loss	Score-CAM

Table 5.5: Experimental setup.

5.4.1 Weight Decay

The results of the experiments with baseline model with Weight Decay over different λ and norms can be seen in the table 5.6:

	Model	Baseline	Weight Decay					
	Norm		L1			L2		
	λ		10^{-5}	10^{-4}	10^{-3}	10^{-5}	10^{-4}	10^{-3}
↑	Acc	**91.10% ± 0.83%**	87.30% ± 2.38%	86.60% ± 1.21%	91.60% ± 0.98%	88.50% ± 0.79%	91.40% ± 1.08%	89.50% ± 1.14%
↑	AUC	**91.00% ± 1.00%**	85.00% ± 1.00%	84.66% ± 2.33%	90.00% ± 1.33%	88.00% ± 1.00%	89.00% ± 4.00%	88.00% ± 4.00%
↑	F1 score	**94.00% ± 1.09%**	89.80% ± 1.30%	89.40% ± 1.78%	92.80% ± 2.31%	91.60% ± 2.11%	93.60% ± 3.08%	91.30% ± 2.02%
↑	IOU (A)	**27.80% ± 0.33%**	17.40% ± 0.34%	26.10% ± 0.51%	19.40% ± 0.74%	19.10% ± 0.45%	22.20% ± 0.92%	15.20% ± 0.26%
↑	IOU (BB)	**39.20% ± 0.69%**	17.60% ± 1.27%	26.60% ± 1.03%	19.40% ± 1.64%	15.40% ± 0.71%	19.10% ± 1.33%	12.10% ± 0.76%
↓	AD	**44.50% ± 0.95%**	61.30% ± 2.38%	45.40% ± 2.22%	59.30% ± 1.90%	61.20% ± 0.48%	59.20% ± 1.25%	61.20% ± 0.67%
↑	IiC	**8.20% ± 0.49%**	5.00% ± 0.98%	7.60% ± 0.22%	5.20% ± 1.12%	4.40% ± 0.94%	5.40% ± 0.74%	4.40% ± 0.66%

Table 5.6: Weight Decay experiment evaluation

The model with L1 and $\lambda = 10^{-3}$ has an accuracy of 91.6%, which is better, than the baseline model, but AUC of 90.0% and F1 of 92.8% is lower than the baseline model. However, the model with L1 norm and $\lambda = 10^{-4}$ outperforms the other models with Weight Decay by both IOU metrics (area and bounding boxes intersection) with 26.1% and 26.6%, respectively, and has a better score on the Average Drop and Increase in Confidence metrics. But has lower explainability score than the baseline model with batch size 256 and the model with Batch Normalization techniques. The L1 norm combats

overfitting by shrinking the parameters towards 0. This makes some features obsolete. The L2 norm combats overfitting by making the weights small but not exactly 0. Reducing λ to 10^{-4} increased the explainability, compared to $\lambda = 10^{-4}$. On the one hand, the next reduction of λ to 10^{-5} improved the explainability compared to $\lambda = 10^{-3}$. On the other hand, λ to 10^{-4} still has the better explainability over all Weight Decay models. As observed, reducing the complexity of the model by penalizing the weights has a negative effect on the Score-CAM.

5.4.2 Dropout

The results of the experiments with the baseline model with Dropout technique over different dropout rates can be seen in the table 5.7:

	Dropout Rate	Baseline	0.10	0.33	0.50	0.75
↑	Acc	**91.10% ± 0.83%**	90.10% ± 1.21%	89.70% ± 1.02%	90.40% ± 0.99%	90.10% ± 0.75%
↑	AUC	**91.00% ± 1.00%**	90.00% ± 1.00%	88.00% ± 4.00%	90.00% ± 1.00%	88.00% ± 1.00%
↑	F1 score	**94.00% ± 1.09%**	93.20% ± 1.31%	91.40% ± 0.74%	92.60% ± 1.13%	91.90% ± 1.05%
↑	IOU (A)	**27.80% ± 0.33%**	27.40% ± 0.64%	21.10% ± 1.29%	25.70% ± 1.06%	15.20% ± 0.79%
↑	IOU (BB)	**39.20% ± 0.69%**	33.10% ± 0.82%	21.80% ± 1.16%	26.70% ± 1.82%	6.90% ± 2.12%
↓	AD	**44.50% ± 0.95%**	47.20% ± 1.15%	51.80% ± 1.54%	48.80% ± 1.39%	61.20% ± 0.24%
↑	IiC	**8.20% ± 0.49%**	7.80% ± 0.22%	5.60% ± 0.41%	7.00% ± 0.12%	4.60% ± 0.44%

Table 5.7: Dropout experiment evaluation.

The model with dropout rate 0.5 has 90.4% accuracy score, 90% AUC, but model with rate 0.10 outperforms other dropout rates by both IOU metrics (area and bounding boxes intersection) with 27.4% and 33.1% respectively and has better score by Average Drop and Increase in Confidence metrics. Also the model with dropout rate 0.75 has significantly lower results on explainability, than the previous models. As was noticed, high dropout rate leads to poor explainability of the model. Dropout tends to increase the correlation between the features it is applied to, which reduces the efficiency of representations. This side effect of the Dropout in a CNN was also observed in past research [ZZL18, WZZ+13].

5.4.3 Batch Normalization

The results of the model with Batch Normalization technique can be observed in the table 5.8:

	Baseline	Batch Norm
↑ Acc	**91.10% ± 0.83%**	90.40% ± 0.10%
↑ AUC	**91.00% ± 1.00%**	88.00% ± 1.00%
↑ F1 score	**94.00% ± 1.09%**	92.30% ± 1.02%
↑ *IOU* (A)	27.80% ± 0.33%	**34.80% ± 0.31%**
↑ *IOU* (BB)	39.20% ± 0.69%	**42.40% ± 0.16%**
↓ AD	44.50% ± 0.95%	**40.20% ± 0.39%**
↑ IiC	8.20% ± 0.49%	**10.20% ± 0.28%**

Table 5.8: Batch Normalization experiment evaluation.

On the figure 5.3 clearly can be seen the effect of the Batch normalization to the explainability method.

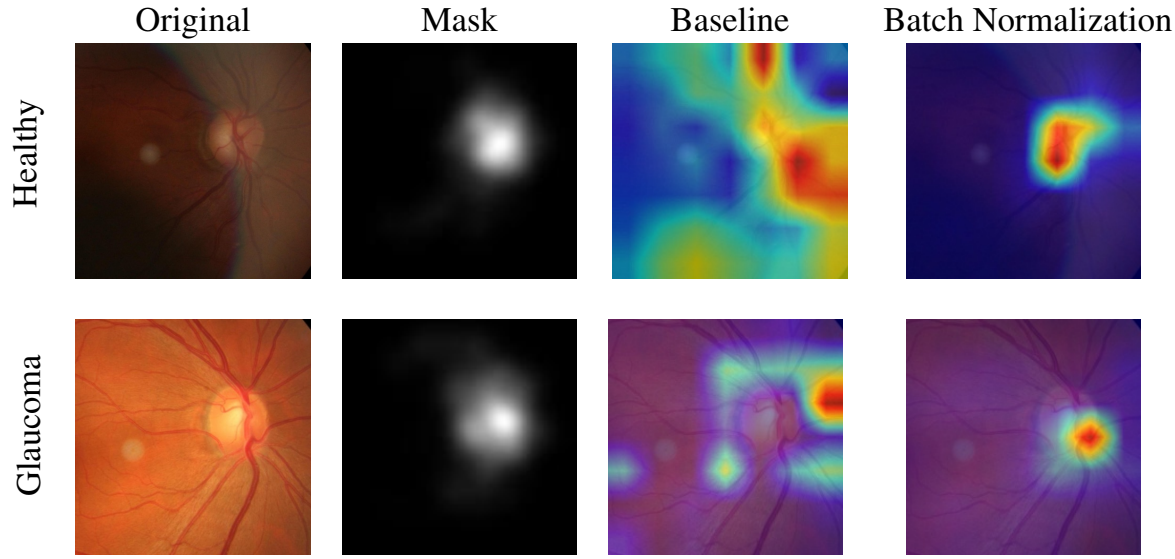


Figure 5.3: Image examples for model with batch normalization

The model with batch normalization has 90.6% accuracy score, 89% AUC and 94.0% F1, which is a little bit lower, than the baseline model, but outperforms baseline model by both explainability metrics (area and bounding boxes intersection) with 34.8% and 42.4% respectively, and has better score by Average Drop and Increase in Confidence metrics. As was observed stabilizing the distributions of layer inputs and normalization of activation maps makes optimization landscape smoothness and leads to increase in explainability [GHK+20]. The model with Batch Normalization can capture shape and edges of the object of interest as can bee seen from the figure 5.3, which makes

activation maps more suitable for the Score-CAM. The trade-off between classification performance and explainability performance is clearly noticeable, as the baseline model achieves better classification performance, while models with Batch Normalization provides better explainability performance. That observation provides an opportunity to evaluate models in practice not only by classification metrics but also by explainability metrics.

5.4.4 Group Normalization

The experiments with Group Size technique over different groups (16, 32 and 64) were executed and results are provided in the table 5.9:

	Group size	Baseline	16	32	64
↑	Acc	**91.10% ± 0.83%**	90.10% ± 0.71%	86.40% ± 1.58%	90.60% ± 0.44%
↑	AUC	**91.00% ± 1.00%**	89.00% ± 1.00%	85.00% ± 4.00%	90.00% ± 1.00%
↑	F1 score	**94.00% ± 1.09%**	93.40% ± 0.30%	89.30% ± 0.58%	93.90% ± 0.52%
↑	IOU (A)	27.80% ± 0.33%	25.70% ± 0.52%	27.70% ± 0.57%	**28.60% ± 0.41%**
↑	IOU (BB)	**39.20% ± 0.69%**	22.80% ± 1.33%	28.90% ± 0.52%	36.20% ± 0.58%
↓	AD	44.50% ± 0.95%	45.50% ± 2.33%	45.00% ± 0.52%	**44.10% ± 0.75%**
↑	IiC	8.20% ± 0.49%	7.60% ± 0.66%	8.10% ± 0.34%	**8.40% ± 0.52%**

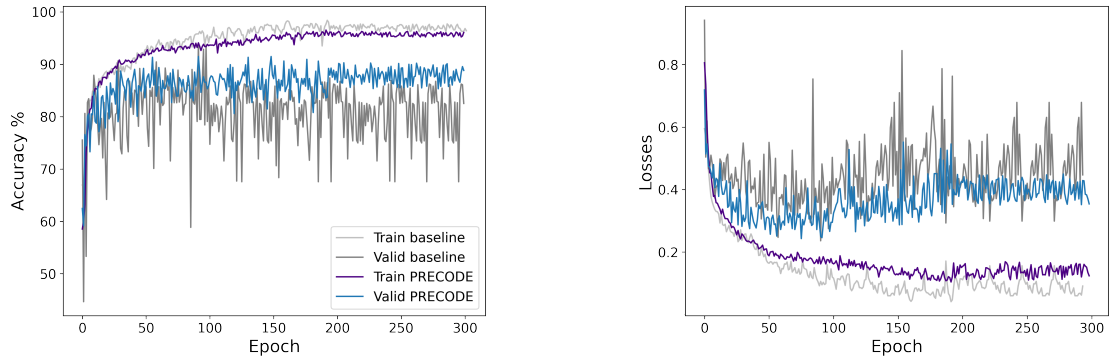
Table 5.9: Group Normalization experiment evaluation

The model with group size 64 has 90.6% accuracy score, 90% AUC and slightly outperforms baseline model by explainability performance. But has lower explainability performance than the model with Batch Normalization technique by 6.2% in both IOU metrics (area and bounding boxes). Normalization over the groups shows better results in explainability, than Weight Decay and Dropout. Group Normalization divides the channels into groups and computes within each group the mean and variance for normalization. Normalization can capture shape and edges of the object of interest, which enables better explainability performance. As in the case of Batch Normalization, there is a trade-off between classification performance and explainability performance, as the baseline model achieves better classification performance, while models with Group Normalization provides better explainability performance.

5.5 PRECODE

First, an experiment was conducted to observe the regularization effects of PRECODE. The model is trained for 300 epochs, to compare the ratio between test accuracy and train accuracy, and test and train losses.

PRECODE has stabilizing effects, as can be observed from the drift between train and val loss, as can be seen on the graphs 5.4:



(a) Train vs Validation Accuracy

(b) Train vs Validation losses

Figure 5.4: The baseline model vs model with PRECODE performance on 300 epochs.

VB is used approximate the distribution of the latent space and obtain a new representation from an approximated prior distribution by stochastic sampling. PRECODE technique has two different hyperparameters: size of the bottleneck in range [128, 256, 512] representation, and β value in range [10^{-3}, 10^{-4}] for the loss function, you can see the results in the table 5.10:

	Model	Baseline	PRECODE					
	β		10^{-3}			10^{-4}		
	k		128	256	512	128	256	512
↑	Acc	91.10% ± 0.83%	89.50% ± 0.52%	91.20% ± 4.1%	87.90% ± 2.50%	89.90% ± 1.40%	**92.30% ± 0.77%**	90.00% ± 0.52%
↑	AUC	**91.00% ± 1.00%**	87.00% ± 1.00%	89.00% ± 1.00%	87.00% ± 1.00%	87.00% ± 1.00%	89.00% ± 4.00%	88.00% ± 1.00%
↑	F1 score	**94.00% ± 1.09%**	92.60% ± 0.40%	92.90% ± 1.33%	91.00% ± 0.52%	92.80% ± 1.57%	93.20% ± 1.22%	92.10% ± 0.89%
↑	IOU (A)	27.80% ± 0.33%	19.00% ± 1.55%	19.50% ± 1.01%	**33.70% ± 0.52%**	17.00% ± 1.09%	22.80% ± 1.69%	18.10% ± 0.55%
↑	IOU (BB)	**39.20 ± 0.69%**	16.00% ± 1.00%	26.70% ± 1.22%	31.70% ± 0.16%	14.70% ± 0.52%	26.30% ± 1.69%	24.20% ± 0.57%
↓	AD	44.50% ± 0.95%	59.00% ± 4.00%	59.30% ± 0.44%	**40.50% ± 1.22%**	54.00% ± 0.66%	52.40% ± 2.11%	62.30% ± 1.20%
↑	IiC	8.20% ± 0.49%	5.10% ± 1.51%	5.10% ± 2.50%	**10.00% ± 1.20%**	5.00% ± 1.04%	5.80% ± 4.00%	4.50% ± 0.16%

Table 5.10: PRECODE experiment evaluation

The model with size of the bottleneck representation of 256 and $\beta = 10^{-4}$ has the better overall accuracy and AUC score, but model with size of the bottleneck representation of 512 and $\beta = 10^{-3}$ achieves significantly better explainability performance by the area and bounding boxes, than other values. It also has better score by Average Drop and Increase in Confidence. That can be explained by the fact, that the model with higher size of the bottleneck can capture more relevant features.

The model with size of the bottleneck representation of 256 and $\beta = 10^{-4}$ has 92.3% accuracy score, 89% AUC, but model with $\beta = 10^{-3}$ and size of the bottleneck representation of 512 outperforms other models with PRECODE technique by both explainability metrics (area and bounding boxes intersection) with 33.7% and 31.7% respectively. It is also shows better results on the explainability, than the baseline model and any other models with regularization techniques, except model for the with Batch Normalization. VB learns a continuous and complete latent feature space distribution. The representation generated from VB positively affects the explainability. The information bottleneck principle defines an optimal representation and contains the most distinct information about input and output. The information is restricted from the input data to the intermediate representation as well as to the output variable. Maximizing the cross entropy between the intermediate representation and the output forces the information flowing from the input to predict the output. When these two processes occur simultaneously, the overall loss causes the model to focus only on the information related to the output in the input data. Research from [LXY20] suggests that the resulting model is less disturbed by information unrelated to the output. That was also observed by the experiments, as in turn VB has positive impact on the explainability, because activation maps produced by VB contain only relevant information to the model interest.

Discussion

This chapter provides the difference between CAM approaches on explainability tasks and how hyperparameter tuning and regularization techniques affect explainability method of image classifiers.

The more detailed representations obtained by increasing the depth of the neural network have a positive impact on explainability performance. It was observed that the ability to see more data with increasing number of epochs in the training steps increased the explainability performance of the neural network. Explainability also increased with the increase in batch size. This can be explained by the fact that the most accurate gradient estimation and confidence in the direction to optimum can be achieved with the increase in the number of training samples used for estimation.

It was observed that Score-CAM has better performance by different explainability metrics. Grad-CAM and Grad-CAM++ use gradient information from the last convolutional layer, which can be noisy and result in noisy activation maps. Score-CAM uses no gradient at all. It forwards activation maps to obtain scores for each activation map, and concludes with a linear combination of weights and activation maps, but also take more time to evaluate. Because of the noisy gradient, that can appear in the deep neural networks, this seems to result in better explainability by Score-CAM.

As experiments with regularization techniques have shown, any change in weights and features, that tends to add noise to the neural network, decreased the explainability. It was found that the models with Weight Decay technique with different norms L1 and L2 with $\lambda = 10^{-3}$ and $\lambda = 10^{-5}$ have significantly decreased explainability performance by 10% and 12% in IOU by area metric, respectively, but $\lambda = 10^{-4}$ with L1 norm has the almost the same explainability performance as the baseline model. Penalizing the weights, which are elements in the convolution filter, in case of CNN, have a bad influence on the explainability, as the less sensitive filters decrease the score for activation maps, that used for combination of resulting activation maps of the Score-CAM. The model with dropout rates of 0.33 and 0.50 has a decrease in explainability of 6% and 2% in IOU by area metric, respectively. The model with a dropout rate of 0.75 has the lowest results with a decrease of 12.6% in IOU by area metric, compared to the baseline model and the models with other regularization techniques. The model with dropout rate of 0.10 have the lowest drop in explainability performance. Dropping out features makes filters more noisy due to high correlation between them compared to the baseline model and results in poor explainability heatmaps.

As the experiments have shown, the model with Batch Normalization increased the explainability of the image classifier by up to 7% on IOU, compared to the baseline model. The model with Group Normalization with group size 64 slightly increased explainability, when channels are divided into groups and the features are normalized within each group. The normalization gives convolutional network a resistance to vanishing gradients during training. The gradients become stronger, the neural network becomes more decorrelated, and the activation maps become smoother as the feature distribution of the

layers is more centered over all training and evaluation time. The model with Batch Normalization, Group Normalization can capture shape and edges of the object of interest, which makes activation maps more suitable for Score-CAM.

The information bottleneck, which is used in PRECODE defines an optimal representation. The information is restricted from the input data to the intermediate representation as well as to the output variable. Maximizing the cross entropy between the intermediate representation and the output forces the information flowing from the input to predict the output. When these two processes occur simultaneously, the overall loss causes the model to focus only on the information related to the output in the input data. In turn that has a positive impact on the explainability because activation maps produced by VB contain only relevant information. The model with PRECODE technique with the size of the bottleneck 512 and $\beta = 10^{-3}$ improves the explainability by 6% in IOU by area metric.

As the experiments showed, there is a correlation between accuracy and explainability in hyperparameters tuning experiments. The classification accuracy increased by 10% when explainability performance also increased by 11%, compared to the initial baseline model. But in the case of the experiments with regularization techniques and PRECODE were a trade-off between classification and explainability performance. The most accurate model was with PRECODE technique with size of bottleneck 256 and $\beta = 10^{-4}$ with an accuracy of 92.3%, but has a lower score on the explainability metric with a value of 22.8%, while the model with Batch Normalization has an overall accuracy of 90.6% and a score of 34.8% on IOU by area.

6 Conclusion

The lack of transparency of neural networks is a significant disadvantage, limiting their interpretability and practical applications. Recent studies have proposed methods based on the Class Activation Mapping approach, aiming to highlight the features that most influence a model's decision. This paper is aimed on the investigation of which factors influence the explainability of CNN-based image classifiers. Methods based on the activation mapping approach were described and evaluated by various explainability metrics. These methods produce visualizations that shall show area of pixels from the input image, that convolutional neural network uses to make a given prediction. The lack of transparency of neural networks is a significant disadvantage, limiting their interpretability and their practical applications. The goal of this work was to observe how different hyperparameters such as network depth, epochs, batch size and techniques like Weight Decay, Dropout, Batch Normalization, Group Normalization, PRECODE have an impact on explainability performance of the neural network model and practical medical dataset.

As was observed by the experiments, higher neural network depth, longer training and diversity of samples used in gradient estimation, provided by increase of the batch size and augmentation, positively affect explainability performance.

After studying the material and evaluating experiments with class activation based approaches, it was investigated that Score-CAM has the best explainability performance compared to Grad-CAM and Grad-CAM++, but takes significantly more time to provide explainability heatmap.

The experiments with the Weight Decay technique have shown, that penalizing the weights leads to less sensitive filters and reduces the explainability of the model. For the Dropout technique it was found, that it tends to increase the correlation between the features to which it is applied, which reduces the efficiency of the representations. The positive impact on explainability was observed by the experiments with Batch Normalization technique, as it provides the ability to capture shape and edges of the object of interest. The same effect was expected for the Group Normalization technique, but the experiments have shown, that it adds noise to the activation maps when channels are divided into groups and the features are normalized within each group. Increase of the model explainability was observed in the experiments with the PRECODE technique, which is implemented with a Variational Bottleneck. As a result, the model is less disturbed by information unrelated to the output, and the activation maps generated by VB contain only relevant information for the model interest.

It was found that there is a correlation between the classification and the explainability performance of the model in the experiments with depth and epochs. In the experiments with batch sizes, regularization techniques, and PRECODE, a trade-off between classification and explainability performance was observed. Therefore it should be considered to evaluate models in practice not only by accuracy metrics but also by explainability metrics.

Due to the limitation of time was unable to examine other regularization techniques as well as all possible hyperparameters for the regularization techniques, which are used in our work. Other binarization algorithms for heatmaps and masks, and different thresholds were not examined. Also other CAM based approaches, such as SmoothScore-CAM, XGrad-CAM, Layer-CAM, were not examined. Based on past research papers [CSHB17, WDYZ19], ADAM was chosen as an optimization algorithm, as it achieves better loss and leads to better explainability, than SGD and RMSprop. But it remains unrevealed how different combination of regularization techniques with other optimization algorithms can perform on the explainability task. Also for the future works it would be useful to investigate how decrease or increase of dataset size affect performance of the explainability methods.

	Depth		18	34	50
↑	Acc		82.20% ± 1.24%	83.50% ± 1.54%	**85.30% ± 1.20%**
↑	ROC AUC		78.00% ± 0.33%	80.00% ± 1.00%	**81.00% ± 1.00%**
↑	F1 score		83.00% ± 1.43%	85.10% ± 2.20%	**85.30% ± 1.54%**
↑	*IOU* (A)	CAM	3.50% ± 0.51%	3.70% ± 0.23%	**5.00% ± 1.10%**
		Grad-CAM	11.60% ± 0.21%	11.50% ± 0.62%	**16.30% ± 0.89%**
		Grad-CAM++	12.20% ± 1.01%	12.30% ± 0.42%	**17.50% ± 0.73%**
		Score-CAM	16.20% ± 1.10%	15.50% ± 0.92%	**21.00% ± 0.33%**
↑	*IOU* (BB)	CAM	3.90% ± 0.54%	3.80% ± 0.22%	**4.80% ± 0.92%**
		Grad-CAM	11.90% ± 0.30%	12.10% ± 0.57%	**15.90% ± 0.93%**
		Grad-CAM++	12.20% ± 1.01%	12.30% ± 0.42%	**15.50% ± 0.73%**
		Score-CAM	14.70% ± 0.94%	13.90% ± 1.03%	**16.70% ± 0.87%**
↓	AD	CAM	92.00% ± 1.37%	88.10% ± 0.86%	**87.40% ± 0.90%**
		Grad-CAM	67.10% ± 1.72%	67.30% ± 1.34%	**65.50% ± 0.96%**
		Grad-CAM++	66.70% ± 1.83%	66.50% ± 1.61%	**66.00% ± 1.66%**
		Score-CAM	56.00% ± 0.57%	60.30% ± 1.02%	**52.20% ± 0.95%**
↑	IiC	CAM	0.00% ± 0.00%	0.00% ± 0.00%	**0.10% ± 0.01%**
		Grad-CAM	2.20% ± 0.22%	2.50% ± 0.41%	**2.60% ± 0.34%**
		Grad-CAM++	2.70% ± 0.20%	2.60% ± 0.31%	**2.80% ± 0.40%**
		Score-CAM	4.80% ± 0.44%	3.70% ± 0.51%	**5.60% ± 0.49%**

Table 6.1: Layer depth evaluation.

Bibliography

[Adi18] D. Aditya. Metrics to evaluate your machine learning algorithm, 2018.

[Ben12] Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. *CoRR*, abs/1206.5533, 2012.

[Bev18] Nikita Bevenden. Learning rate finder. 2018.

[BGS18] Johan Bjorck, Carla P. Gomes, and Bart Selman. Understanding batch normalization. *CoRR*, abs/1806.02375, 2018.

[BGU21] Arwen V. Bradley and Carlos Alberto Gomez-Uribe. How can increased randomness in stochastic gradient descent improve generalization?, 2021.

[Bot10] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In Yves Lechevallier and Gilbert Saporta, editors, *Proceedings of COMPSTAT'2010*, pages 177–186, Heidelberg, 2010. Physica-Verlag HD.

[Bro19] Jason Brownlee. How to control the stability of training neural networks with the batch size. 2019.

[Bus] Nikita Bushaev. Cross-entropy loss function.

[Bus18] Nikita Bushaev. Adam — latest trends in deep learning optimization. 2018.

[Che19] M. Chetoui. Gradient-weighted class activation mapping - grad-cam-, 2019.

[CSHB17] Aditya Chattopadhyay, Anirban Sarkar, Prantik Howlader, and Vineeth N. Balasubramanian. Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. *CoRR*, abs/1710.11063, 2017.

[Cza21] J. Czakon. 24 evaluation metrics for binary classification (and when to use them), 2021.

[Ele20] Ismail Elezi. Exploiting contextual information with deep neural networks. *CoRR*, abs/2006.11706, 2020.

[Fir] Firiuza. Optimizers for training neural networks.

[FW18] H. Fang and V. Wang. Dissecting deep learning networks—visualizing mutual information, 2018.

[GHK+20] Mathilde Guillemot, Catherine Heusele, Rodolphe Korichi, Sylvianne Schnebert, and Liming Chen. Breaking batch normalization for better ex-

plainability of deep neural networks through layer-wise relevance propagation. *CoRR*, abs/2002.11018, 2020.

[Haw19] Douglas M. Hawkins. The problem of overfitting. 2019.

[HZRS16] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. 2016.

[IS15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[KGC17] Jan Kukačka, Vladimir Golkov, and Daniel Cremers. Regularization for deep learning: A taxonomy. 2017.

[KXSJ15] He Kaiming, Zhang Xiangyu, Ren Shaoqing, and Sun Jian. *Deep Residual Learning for Image Recognition*. cornell university, 2015.

[LCY13] M. Lin, Q. Chen, and S. Yan. Network in network. 2013.

[LLL+19] Guanlin Li, Lemao Liu, Xintong Li, Conghui Zhu, Tiejun Zhao, and Shuming Shi. Understanding and Improving Hidden Representations for Neural Machine Translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 466–477, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[LSV19] Alex Labach, Hojjat Salehinejad, and Shahrokh Valaee. Survey of dropout methods for deep neural networks. *CoRR*, abs/1904.13310, 2019.

[LXW+19] Liu Li, Mai Xu, Xiaofei Wang, Lai Jiang, and Hanruo Liu. Attention based glaucoma detection: A large-scale database and cnn model. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[LXY20] Hongpeng Liao, Jianwu Xu, and Zhuliang Yu. Novel convolutional neural network with variational information bottleneck for p300 detection. *Entropy*, 23:39, 12 2020.

[LXY21] Hongpeng Liao, Jianwu Xu, and Zhuliang Yu. Novel convolutional neural network with variational information bottleneck for p300 detection. *Entropy*, 23(1), 2021.

[MHLW19] Fanman Meng, Kaixu Huang, Hongliang Li, and Qingbo Wu. Class activation map generation by representative class selection and multi-layer feature fusion. 2019.

[MHN18] Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. Rectifier nonlinearities improve neural network acoustic models. 2018.

[Mis19] D. Mishra. Demystifying convolutional neural networks using gradcam,

2019.

[ML18] Dominic Masters and Carlo Luschi. Revisiting small batch training for deep neural networks. *CoRR*, abs/1804.07612, 2018.

[NMZ+09] Y. Q. Ni, M.ASCE, H. F. Zhou, J. M. Ko, and F.ASCE. Generalization capability of neural network models for temperature-frequency correlation using monitoring data. 2009.

[PPD+21] Rafael Padilla, Wesley L. Passos, Thadeu L. B. Dias, Sergio L. Netto, and Eduardo A. B. da Silva. A comparative analysis of object detection metrics with a companion open-source toolkit. *Electronics*, 10(3), 2021.

[PVS20] Linardatos Pantelis, Papastefanopoulos Vasilis, and Kotsiantis Sotiris. *Explainable AI: A Review of Machine Learning Interpretability Methods*. MDPI, 2020.

[Rud16] Sebastian Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016.

[Sar19] Sumit Sarin. Vgg vs resnet. 2019.

[SDV+16] Ramprasaath R. Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization. *CoRR*, abs/1610.02391, 2016.

[SHK+] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Diabetic retinopathy detection.

[SHK+14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[SKL17] Samuel L. Smith, Pieter-Jan Kindermans, and Quoc V. Le. Don't decay the learning rate, increase the batch size. *CoRR*, abs/1711.00489, 2017.

[SMS21] Daniel Scheliga, Patrick Mäder, and Marco Seeland. Precode - a generic model extension to prevent deep gradient leakage, 2021.

[Sri17] Nitish Srivastava. Improving neural networks with dropout. 2017.

[STE13] Christian Szegedy, Alexander Toshev, and Dumitru Erhan. Deep neural networks for object detection. 2013.

[STIM19] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization?, 2019.

[THKT20] Søren Taverniers, Eric Joseph Hall, Markos A. Katsoulakis, and Daniel M. Tartakovsky. Mutual information for explainable deep learning of multiscale systems. *CoRR*, abs/2009.04570, 2020.

[VDH20] Sahil Verma, John P. Dickerson, and Keegan Hines. Counterfactual explanations for machine learning: A review. *CoRR*, abs/2010.10596, 2020.

[Ver20] T. Vermeire. *EXPLAINABLE IMAGE CLASSIFICATION WITH EVIDENCE COUNTERFACTUAL.* International Journal of Computer Vision, 2020.

[VM20] Tom Vermeire and David Martens. Explainable image classification with evidence counterfactual. 04 2020.

[VRBPY19] Arya Vijay, K. Rachel, K. Bellamy, and Chen Pin-Yu. *One Explanation Does Not Fit All: A Toolkit and Taxonomy of AI Explainability Techniques*. IBM, 2019.

[WDYZ19] Haofan Wang, Mengnan Du, Fan Yang, and Zijian Zhang. Score-cam: Improved visual explanations via score-weighted class activation mapping. *CoRR*, abs/1910.01279, 2019.

[WH18] Yuxin Wu and Kaiming He. Group normalization. *CoRR*, abs/1803.08494, 2018.

[WWD+20] Haofan Wang, Zifan Wang, Mengnan Du, Fan Yang, Zijian Zhang, Sirui Ding, Piotr Mardziel, and Xia Hu. Score-cam: Score-weighted visual explanations for convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 24–25, 2020.

[WZZ+13] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1058–1066, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[YJP18] Shujian Yu, Robert Jenssen, and José C. Príncipe. Understanding convolutional neural network training with information theory. *CoRR*, abs/1804.06537, 2018.

[You15] Jamileh Yousefi. Image binarization using otsu thresholding algorithm, 05 2015.

[ZWXG18] Guodong Zhang, Chaoqi Wang, Bowen Xu, and Roger B. Grosse. Three mechanisms of weight decay regularization. *CoRR*, abs/1810.12281, 2018.

[ZZL18] Zijun Zhang, Yining Zhang, and Zongpeng Li. Removing the feature correlation effect of multiplicative noise. *CoRR*, abs/1809.07023, 2018.