A Deep Learning Interpretable Classifier for Diabetic Retinopathy Disease Grading

Jordi de la Torre[a,*], Aida Valls[a], Domenec Puig[a]

[a] *Departament d'Enginyeria Informàtica i Matemàtiques.*
Escola Tècnica Superior d'Enginyeria.
Universitat Rovira i Virgili
Avinguda Paisos Catalans, 26. E-43007
Tarragona, Spain

Abstract

Deep neural network models have been proven to be very successful in image classification tasks, also for medical diagnosis, but their main concern is its lack of interpretability. They use to work as intuition machines with high statistical confidence but unable to give interpretable explanations about the reported results. The vast amount of parameters of these models make difficult to infer a rationale interpretation from them. In this paper we present a diabetic retinopathy interpretable classifier able to classify retine images into the different levels of disease severity and of explaining its results by assigning a score for every point in the hidden and input space, evaluating its contribution to the final classification in a linear way. The generated visual maps can be interpreted by an expert in order to compare its own knowledge with the interpretation given by the model.

Keywords: deep learning, classification, explanations, diabetic retinopathy, model interpretation
2010 MSC: 68T10

1. Introduction

Deep Learning methods have been used extensively in the last years for many automatic classification tasks. For the case of image analysis, the usual procedure consists on extracting the important features with a set of convolutional layers and, after that, make a final classification with these features using a set of fully connected layers. Finally, a soft-max output layer gives as a result the predicted output probabilities of the set of classes predefined in the model. During training, model parameters are changed using a gradient-based optimization

*Corresponding author
Email addresses: jordi.delatorre@gmail.com (Jordi de la Torre), aida.valls@urv.cat (Aida Valls), domenec.puig@urv.cat (Domenec Puig)

algorithm, which minimizes a predefined loss function.

Once the classifier has been trained (i.e. the parameters of the different layers of the model have been fixed), the quality of the classification outputs predicted is compared against the correct "true" values stored on a labeled dataset. This data is considered as the gold standard, ideally coming from the consensus of the knowledge of a human experts committee.

This mapping allows the classification of multidimensional objects into a small number of categories. The model is composed by many neurons that are organized in layers and blocks of layers, piled together in a hierarchical way. Every neuron receives the input from a predefined set of neurons. Every connection has a parameter that corresponds to the weight of the connection.

The function of every neuron is to make a transformation of the received inputs into a calculated output value. For every incoming connection, the weight is multiplied by the input value received by the neuron and the aggregated value is feeded to an activation function that calculates the output of the neuron. The parameters are usually optimized using a stochastic gradient descent algorithm that minimizes a predefined loss function. The parameters of the network are updated after backpropagating the loss function gradients through the network. These hierarchical models are able to learn multiple levels of representation that correspond to different levels of abstraction, which enables the representation of complex concepts in a compressed way [1], [2], [3], [4].

Deep Learning based models have been proven to be very effective when trained with enough labelled data (order of magnitude of tens of thousands of examples per class) but their main concern is its *lack of interpretability*. Every successful model tend to have millions of parameters, making difficult to get from them a rationale interpretation.

In medical diagnosis tasks is important not only the accuracy of the predictions but also the reasons behind a decision. Self-explainable models enable the physicians to contrast the information reported by the model with their own knowledge, increasing the information and the probability of a good diagnostic.

Different attempts have been done in order to interpret the results reported by neural networks. In [5] a network propagation technique is used for the visualization of the features in the input space. After this [6] used a pixel-wise decomposition for classification decision. This decomposition could be done in two ways: considering the network as a global function, disregarding its topology (functional approach) or using the natural properties of decomposition of the inherent topology of the function to use a message passing technique for propagating back into the pixel space the decomposition. After this, in [7] they used a so named Deep Taylor decomposition technique to replace the inherently intractable standard Taylor decomposition using a multitude of simpler analytically tractable Taylor decompositions.

In our work we use a similar approach to the used in the pixel-wise decomposition, taking into account the compositional nature of the topology as in [5] and [6]. The novel approach comes from the fact that being the score globally conservative, the conservation does not hold between layers. The concept of *score* in our paper is similar to the concept of *relevance* used in layer-wise

relevance propagation. Apart from the input-space contribution, there is also another one coming from every layer that is independent from the input-space and that depends only on the parameters of each layer. It is not an attribute of the individual pixels that has to be back-propagated but a contribution of the receptive field (RF) that represents the layer as an individual entity. We only propagate back the part of the score that depends on the precedent input for every layer. In our model explanation we consider the constant part as a property of the RF of every layer. This approach, allows us to do an exact propagation of the scores using a deconvolutional approach. Differing also from [5], our method allows the integration of the batch normalization and of other typical neural network block constituents into the score propagation. A full set of score propagation blocks with the more typical deep learning functional constituents is derived in order to make as easy as possible the porting of the paper results to other networks and applications.

This interpretation model is tested in our application research area: diabetic retinopathy (DR). DR is a leading disabling chronic disease and one of the main causes of blindness and visual impairment in developed countries for diabetic patients. Studies reported that 90% of the cases can be prevented through early detection and treatment. Eye screening through retinal images is used by physicians to detect the lesions related with this disease. Due to the increasing number of diabetic people, the amount of images to be manually analyzed is becoming unaffordable. Moreover, training new personnel for this type of image-based diagnosis is long, because it requires to acquire expertise by daily practice. Medical community establishes a standardized classification based on four severity stages [8] determined by the type and number of lesions (as micro-aneurysms, hemorrhages and exudates) present in the retine: class 0 referring to no apparent retinopathy, class 1 as a Mild Non-Proliferative Diabetic Retinopathy (NPDR), class 2 as Moderate NPDR, class 3 as a Severe NPDR and class 4 as a Proliferative DR.

We design a *DR interpretable image classification model* for grading the level of disease. This model is able to not only report the predicted class but also to score the importance of every pixel of the input image in the final classification decision. In such a way is possible to determine which pixels in the input image are more important in the final decision and facilitate the human experts an explanation to verify the results reported by the model.

The paper is structured as follows: in Section 2 the current work on deep learning applied to DR is briefly introduced, then, the main works on interpretation of DL are presented. Section 3 we present the complete mathematical formulation of our interpretable model describing the score propagation model, Section 4 describes the DR DL classification model, Section 5 present the results showing a set of samples of the type of visual interpretations and finally Section 6 present the final conclusions of our work.

2. Related Work

Many deep learning based DR classifiers has been published in the last years. In [9] a deep learning classifier was published for the prediction of the different disease grades. This model was trained using the public available EyePACS dataset. The training set had 35,126 images and the test set 53,576. The quadratic weighted kappa (qwk) evaluation metric [10] over the test set using a unique deep learning model without ensembling was close to the reported by human experts.

In [11] a deep learning classifier was published for the detection of the most severe cases of DR (grouping for the joined detection of the classes of referable DR, defined as moderate or worse DR or referable macular edema). This model was trained using an extended version of the EyePACS dataset mentioned before with a total of 128,175 images and improving the proper tagging of the images using a set of 3 to 7 experts chosen from a panel of 54 US expert Ophtalmologists. This model surpassed the human expert capabilities, reaching at the final operating point approximately 97% sensitivity and 93.5% specificity in the test sets of about 10,000 images for detecting the worse cases of DR. The strength of this model was its ability to predict the more severe cases with a sensitivity and specificity greater than human experts. The drawback, as many deep learning based models, is its lack of interpretability. The model acts like a *intuition machine* with a highly statistical confidence but lacking an interpretation of the foundations of the final decisions making difficult to the experts to balance and compare its prior knowledge with the reasons behind the final conclusion to get even better diagnostics.

In last years different approximations have been derived to convert the initial deep learning black box classifiers into *interpretable classifiers*. In the next sections we introduce the more successful interpretation models existing today: sensitivity maps, layer-wise relevance propagation and Taylor type decomposition models.

2.1. Sensitivity maps

Sensitivity maps [12] are pixel-space matrices obtained from the calculation of $\frac{\partial f(I)}{\partial I_{c,i,j}} \quad \forall c, i, j$. This matrices are easy to calculate for deep neural networks. They use the same backpropagation rules that are used during training, requiring only one more backpropagation step for reaching the input space. The problem with this approach is that there is no direct relationship between $f(I)$ and $\nabla f(I)$. The main concern of this models is that being the objective to explain $f(x)$, $\frac{\partial f(I)}{\partial I_{c,i,j}}$ is only giving us information about the local change of the function. For high non-linear functions like deep neural networks the local variation is pointing to the nearest local optimum that not necessarily should be in the same direction that the global minimum [13].

2.2. Layer-wise relevance propagation

In [6] the authors split the total score of a classification into individual *relevance scores* that act as a positive or negative contributions to the final

result.

The method has the next general constraints: the first one is the nature of the classification function that has to be decomposable into several layers of computation (like a deep neural network), the second one that the total relevance must be preserved from one layer to another, that is to say that the relevance of one layer equals the ones of all other layers (eq. 1) and finally that the relevance of every node must be equal to the sum of all the relevance messages incoming to such a node and also equal to the sum of all relevance messages outgoing from the same node (eq. 2).

$$f(x) = \sum_{d \in l+1} R_d^{(l+1)} = \sum_{d \in l} R_d^{(l)} = ... = \sum_d R_d^{(1)} \tag{1}$$

$$R_{i \leftarrow k}^{(l,l+1)} = R_k^{(l+1)} \frac{a_i \omega_{ik}}{\sum_h a_h \omega_{hk}} \tag{2}$$

As the authors explain in [6], these constraints does not assure a unique way of splitting the score into the different nodes and does not guarantee the final score distribution to have a meaningful interpretation of the classifier prediction.

2.3. Taylor-type decomposition

Another way for solving the interpretability problem is using the gradient of the classification function for the calculation of the next Taylor approximation [6]:

$$f(I) \approx f(I_0) + \nabla(I_0)[I - I_0] = f(I_0) + \sum_{c=1}^{C} \sum_{i=1}^{H} \sum_{j=1}^{W} \frac{\partial f}{\partial I_{c,i,j}} (I_{c,i,j} - I_{0c,i,j}) \tag{3}$$

Being I_0 a free parameter that should be chosen in a way that $f(I_0) = 0$ in the case of f(I) defined as a function that reports a value greater than one when belongs to the class and lower than 0 otherwise. Defined in such a way, $f(I) = 0$ express the case of maximum uncertainty about the image. Finding I_0 allows us to express $f(I)$ as:

$$f(I) \approx \nabla(I_0)[I - I_0] = \sum_{c=1}^{C} \sum_{i=1}^{H} \sum_{j=1}^{W} \frac{\partial f}{\partial I_{c,i,j}} (I_{c,i,j} - I_{0c,i,j}) \quad being \quad f(I_0) = 0 \tag{4}$$

Equation 4 is per se an explanation of $f(I)$ dependent only of the derivative and of I_0. The main problem of this approach is finding a valid root that is close under the euclidean norm to the analyzed image I. We are approximating the function with a order 1 Taylor expansion and the residuum is proportional to the euclidean distance between both points. Different ways for finding I_0 have been proposed. For example, doing a unsupervised search of $f(I)$ over the training set looking for those images reporting $f(I)$ near 0 and averaging them for finding I_0.

2.4. Deep Taylor decomposition

Deep Taylor decomposition [7] uses approximation that combines the layer-wise and the Taylor type models. Being compositional the nature of deep learning models, this approach supposes also the decomposability of the relevance function, presuming the existence for every node of a partial relevance function $R_i(a_i)$ that depends on the activation. It considers this function unknown and applies a Taylor decomposition through a root point. Summing up all the individual contributions using the relevance conservation property defined in the previous models, makes possible the propagation of the intermediate relevance to eventually reach the input space and come to a heatmap of the total relevance of the prediction.

3. Receptive field and Pixel-wise Explanation Model

In this section we describe our contribution to the explanation models. The model is based on the layer-wise relevance propagation model described above. We reformulate one of the properties of the relevance propagation. All the models of the previous section are based on the fact that relevance should be conservative between layers. In our formulation, we consider the score (we rename relevance to score) entering to a node as the combination of two parts: one that can be transformed into a function dependent on the inputs and another one that is constant and that belongs to the own node. The final score continues to be conservative but not through layers. The final score is the sum of the contribution of the studied feature-space (that can be also the pixel space) plus the score contributions of every following layer. The contribution of every following layer depends of the parameters of the layer and in some way of the output activations. The propagated score depends solely on the individual activation inputs of the layer. In such a way, we are able to find a *unique way* for mapping the score of every output to the input space for the network.

The following propositions are assumed:

Proposition 1. The score of every activation in the network is proportional to the activation value:

$$S_k = \lambda_k a_k \tag{5}$$

Proposition 2. The score observed as output one layer can be decomposed in two parts, one dependent on the inputs and another one independent from them that is constant:

$$S_o = S_i + S_k \tag{6}$$

where: S_i depends on the input activation of that layer and S_k does not depend on the input activation but only on the parameters of the model that is executed in that layer.

The propagation model proposed makes a different treatment of the components S_i and S_k. On one hand, S_i depends on the input activation arriving from the original image, so during the propagation backwards we separate the

second component S_k and we take $S_o^{(l-1)} = S_i^{(l)}$. In the following subsections we explain how to obtain S_i and S_k for different typical block constituents of deep learning networks.

On the other hand, the S_k values obtained from each of the layers are mapped to the input space in a final procedure by means of the corresponding RFs.

3.1. Score propagation through an activation function node

In fig. 1 we show the activation function node. A input activation a_i is transformed into $a_o = \phi(a_i)$. We know that $S_o = \lambda_o a_o$, substituting a_o we get $S_o = \lambda_o \phi(a_i)$. In order the proposition to be true, we require also that $S_i = \lambda_i a_i$. For ReLU family functions ($\phi(x) = max(0, kx)$), S_i continues verifying the proposition. For other type of activation functions, as we are calculating the score of a particular image, we can consider the network to have parameterizable activation functions. For a particular image we can consider the first order Taylor expansion and see the activation function as a linear function of the form $\phi(a_i) = [\phi(a_i^*) + \phi'(a_i^*)(a_i - a_i^*)]$, where a_i^* is a value close enough to a_i to have a good approximation of ϕ. After this transformation, the proposition holds for every type of activation function. Substituting and reordering the expression of S_o we obtain that:

$$S_o = \lambda_o[\phi(a_i^*) - \phi'(a_i^*)a_i^*] + \lambda_o \phi'(a_i^*)a_i \tag{7}$$

The score of the output can be splitted in two parts: a constant one that is independent of the activation and belongs to the layer, and another one dependent on the activation. For ReLU, $S_o = S_i$ and $S_k = 0$.

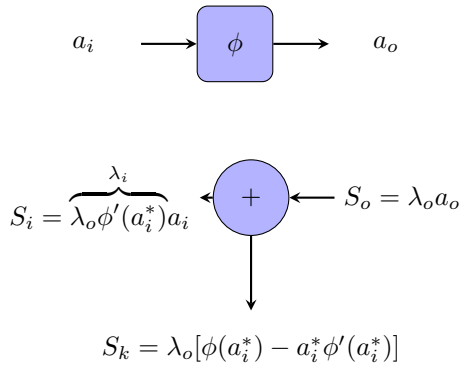


Figure 1: Score propagation through an activation function node

3.2. Score propagation through a batch normalization node

The function implemented in a batch normalization node is $a_o = \beta + \gamma(\frac{a_i - \mu}{\sigma})$. Having $S_o = \lambda_o a_o$, S_o is also $S_o = \lambda_o(\beta + \gamma(\frac{a_i - \mu}{\sigma}))$. Reordering the expression, we can separate the input independent constants:

$$S_o = \lambda_o(\beta - \gamma\frac{\mu}{\sigma}) + \lambda_o\frac{\gamma}{\sigma}a_i \tag{8}$$

As we see, the output score can be exactly splitted into a constant value $S_k = \lambda_o(\beta - \gamma\frac{\mu}{\sigma})$ that is a inherent property of the node and is completely independent of a_i plus $S_i = (\lambda_o\frac{\gamma}{\sigma})a_i = \lambda_i S_i$ that continues to be consistent with the score property proposition, being $\lambda_i = \lambda_o\frac{\gamma}{\sigma}$ (see fig. 2)

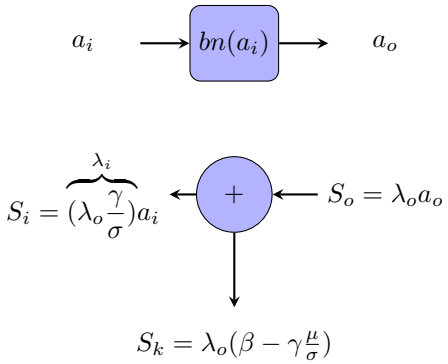


Figure 2: Score propagation through an batch normalization node

3.3. Score propagation through a convolutional layer

In the forward propagation of a two dimensional convolution of an image, the set of all the different feature activations of a predefined locality are linearly combined to get the output a_o (see fig. 3). Backpropagating a score in a convolutional layer requires to divide it into all its individual components. Every component can be either positive or negative. There is also a bias part, that comes from the inherent nature of the layer and that is not attributable to any of the inputs and that must be treated also as a property of the layer. Due to the nature of the convolution operator, every input node contributes to the calculation of different outputs, that's why every input receives a contribution of the score of different outputs that are summed up.

3.4. Score propagation through pooling layers

The score propagation through a max-pooling layer is straightforward. For score propagation the value of the score of the output is copied into the input that was selected in the forward pass (see fig. 4). For average pooling is also straightforward. For score propagation the value of the score is splitted into N equal parts, being N the number of inputs (see fig. 4).

3.5. Score propagation through a fully connected layer

A fully connected layer is a linear combination of the input activities and the weights. The final score is splitted into the individual elements leaving apart the bias that becomes the score contribution of the own layer (see fig. 5).

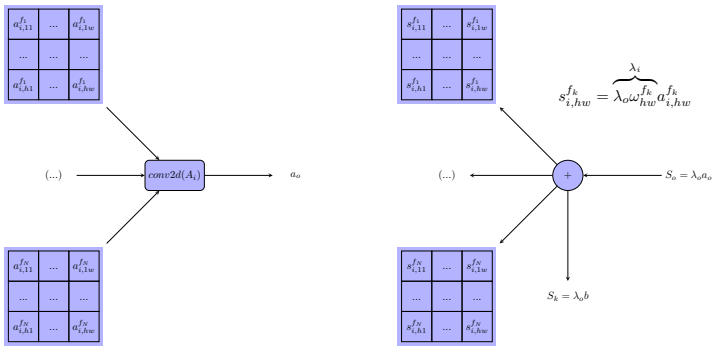
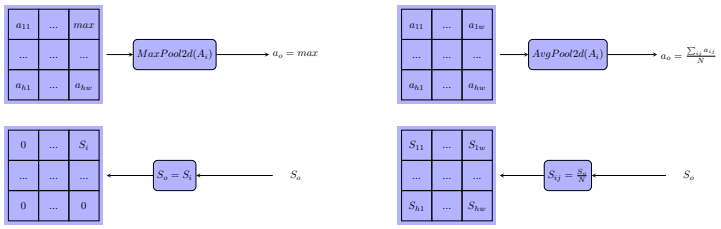


Figure 3: Convolution score calculation. Score spreads into the different inputs. The bias related part of the score is not backpropagated.



(a) Max-Pooling (b) Avg-Pooling

Figure 4: Score propagation through different pooling layers

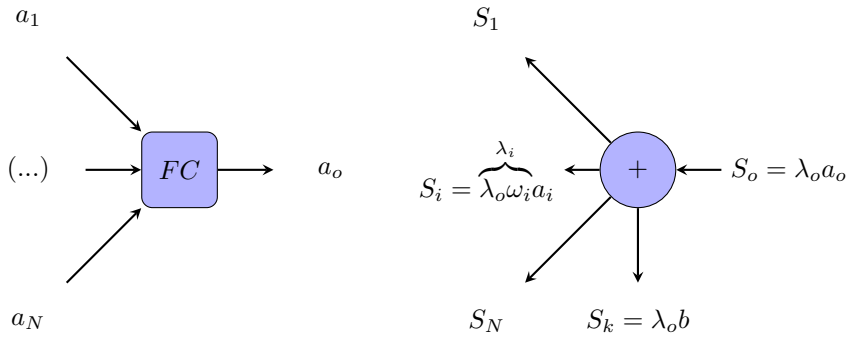


Figure 5: Score propagation through a fully connected node

3.6. Score propagation through a dropout layer

Dropout in evaluation time acts weighting the output to a value proportional to the dropout probability $a_o = (1-d)a_i$. Inserting this equation into $S_o = \lambda_o a_o$ and applying the conservation of the score through the node ($S_o = S_i$ in this

case, due to the absence of constant score) we get that the final equation:

$$\lambda_i = \lambda_o(1-d) \tag{9}$$

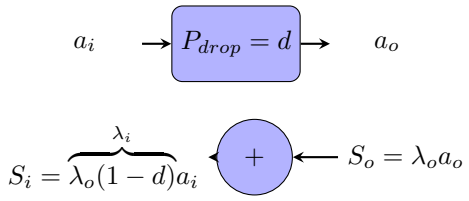


Figure 6: Score propagation through a dropout node

3.7. Mapping the score of hidden layers and S_k to input-space

We have seen that every block has two score constituents: one that is dependent on the inputs and that can be easily forwarded, and another one that depends on the RF, i.e the layer. At this point we are going to transport back also such values to the input-space. From [14] we know that the effective RF is not equal to the theoretical RF. The effective one acts more like a 2D gaussian function where the points located in the borders contribute less than the center ones. Using such property is possible to make an approximate conversion of the hidden-space full and constant scores to the input space using a 2D gaussian prior. For example, for a 20x20 hidden layer with a RF of 189x189 pixels, we know that every of such points is a representation value of a RF of 189x189 in the input space. Having a prior information about the statistical distribution of the input space pixels (in this case gaussian) is possible to go back. Summing up 20x20 gaussian distributions of mean equal to the values of the hidden space and summing up the coincident points is possible to map the distribution in the input space. We fixed $RF = 2\sigma$ as an approximate distribution of the scores, that seems acceptable [14], 98% of the information of the gaussian is inside the RF. We normalize the function to fit 100% of the information inside the RF.

4. Classification Model

4.1. Data

In this study we use the EyePACS dataset of the Diabetic Retinopathy Detection competition hosted on the internet Kaggle Platform. For every patient right and left eye images are reported. All the images are classified by ophthalmologists according to the standard severity scale presented before in [8]. The images are taken in variable conditions: by different cameras, illumination conditions and resolutions.

The training set contains a total of $75,650$ images; $55,796$ of class 0, $5,259$ of class 1, $11,192$ of class 3, $1,805$ of class 3 and $1,598$ of class 4. The validation

set used for hyper-parameter optimization has 3,000 images; 2,150 of class 0, 209 of class 1, 490 of class 2, 61 of class 3 and 90 of class 4. The test set, used only one time for generalization evaluation, contains a total of 10,000 images; 7,363 of class 0, 731 of class 1, 1,461 of class 2, 220 of class 3 and 225 of class 4.

This dataset is not so rich and well tagged as the used in [11] but allows to train models near human expertise that are useful to show the purposes of our work, that is not only a good performance of the results but mainly study the pixel interpretability of the conclusions (final classification) given by the model.

4.2. Prediction model

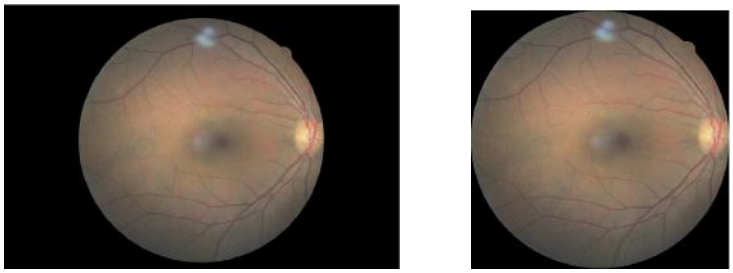
The model calculates $P(\mathcal{C}|\mathcal{I})$ using as a last layer a $SoftMax$ function over the values after the last linear combinations of the features. This probability is calculated as $P(\mathcal{C}|\mathcal{I}) = \frac{e^{S_i}}{\sum_{j=1}^{C} e^{S_j}}$. Let's call S_C the score of the class C, being S_C the final value of each output neuron before applying the $Softmax$. $SoftMax$ function is required for calculating the probability of every class, but in case of being interested only on $argmax(Softmax)$, we needn't evaluate $Softmax$ because $argmax(S_i) = argmax(softmax(S_i))$.

Deep neural network model design up to know is driven mainly by experience. Nowadays there is still more an art than a science and lacks a systematic way for designing the best architecture for solving a problem. In previous works (see [15] and [16]) we have tested different kinds of architectures that allow us to have a previous knowledge of which kind of models work better for solving this particular classification task.

Using the previous experience in such works we summarize a set of guidelines that ruled the final model selection. These design principles applicable to this the DR particular application, and that are explained below, are: use an optimal image resolution, use all the image information available, use a fully convolutional neural network, use small convolutions, adapt the combination of convolution sizes and number of layers to have a final RF as similar as possible to the image size, use ReLU as activation function, use batch normalization in every layer, use QWK as a loss function, use a efficient number of features and use a linear classifier as the last layer.

Use an optimal image resolution. On one hand the size of the input image has a great importance in the classification results. In this problem in other papers like [15] is shown that better results can be achieved with retine diameters of 512 pixels than the ones obtained with 384, 256 or 128 pixels. Some tests done using greater densities than 512 pixel/diameter seem to not improve significantly the classification rates. On the other hand, the hardware of the calculation devices fix a limitation on the available resources. Input image size has a great impact on the memory and calculation time required for the training and test of the deep neural network models. In this work we tested models of 128, 256, 384, 512, 640, 724, 768 and 892 pixels of retine diameter. With this dataset, diameters greater than 640 does not seem to report better results. The optimal size and the used in this study is a retine diameter equal to 640 pixels.

Use all the available image information. In previous studies published in [15] due to hardware limitations the classification models were designed using limited input information, using only part of the available input, requiring ensembling solutions to combine the results from evaluating different parts of the same retine. A 512x512 input image model was used with a random selection of a rotated square (diagonal equal to the retine diameter). In this way only a 64% of the retine information available was used in the classification prediction. On test time five rotated versions of the input where averaged in order to get a better evaluation result. In this paper we use a network that receives all the input information available not requiring ensembling on test time. Only background located further from the diameter is removed (see fig 7).



(a) Original 4752x3168 pixels (b) Trim&resize 640x640 pixels

Figure 7: A training sample showing the preprocessing treatment

Use a fully convolutional neural network. Convolutional neural networks (CNN) are computationally more efficient than fully connected ones. CNNs are ideal for exploiting the typical high local pixel correlations present in images.

Use small size convolutions. The stacking of small size convolutions is more efficient than the usage of big size convolutions. With a lower number of parameters is possible to generate more nonlinear relationships between the pixels that only using a unique convolution of higher size. Following this philosophy only 3x3 convolutions have been used in the feature layers.

Adapt convolution sizes and number of layers to get a RF as similar as possible to the image size. One important aspect of CNNs is the RF size. RF defines the theoretical space covered by a convolution in the input space. The ideal case is having a RF in the last layer equal to the image size, because in such a way we are sure that all the information available is used. RFs greater than image size are inefficient, that's why sometimes can be necessary to slightly modify the convolution sizes of some layer to get the desired one. Figure 8 shows the RF growth of our model.

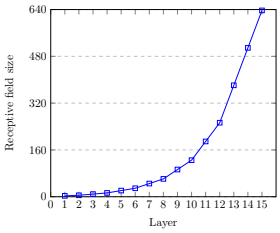


Figure 8: Model RF size growth

Use rectified linear unit (ReLU) as activation function. ReLU is a computationally efficient activation function that is very suitable to be used with very deep convolutional neural networks[17]. Derivatives and Scores as we will see are easily calculated. We have tested other activation functions such as LeakyReLU, ELU and SeLU reporting similar and even worse results, introducing complexity to the model without a clear advantage in the final result.

Use batch normalization in every layer. Batch normalization [18] stabilize the training and accelerates convergence. In this problem there is a great difference between using batch normalization or not, to the point that not using it makes very difficult or even impossible the training.

Use QWK as a loss function. For multi-class classification the standardized loss function to use is the logarithmic loss [19]. In [9] is shown that for ordinal regression problems, where not only a multi-class classification is taking place but also there is possible to establish a sorting of the classes based on some hidden underlying causes, QWK-loss can also be used with better results. The properties of this function as a loss function have been widely studied in [16]. The difference in the performance of the final results is very high. Optimizing directly QWK allows getting better classification results.

Use a linear classifier as a last layer. For simplicity and interpretability of the model we expect the model to disentangle completely the features required for the classification. The final classification is required to be a linear combination of the features of the last layer.

Use a efficient number of features. With infinitely number of resources we can use a big network. In our case we have limited resources and not only this but we would like also to be able to implement the result in devices with low resources. In this way we have tested networks of different sizes and in order to check the redundancy of the information, we made a principal component analysis (PCA) in the feature space of the last layer, arriving to the conclusion that about 32 of the features explain 98.3 % and 48 features, 99.997% of the total variance. We studied different configurations using different number of features from 512 to 32. Values of 32 showed a reduction in performance that increased when increasing the features to 64. Higher number of features did not improve the results. In figure 9 we show the variance explained by final feature vector space.

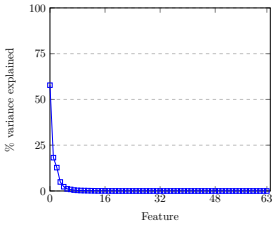


Figure 9: Feature space cummulative PCA variance computed over the training set

Model description. The model use a 3x640x640 input image obtained from a minimal preprocessing step where only the external background borders are trimmed and later resized to the required input size (see fig. 7). Figure 10

shows a block diagram of the model. It is a CNN of 391,325 parameters, divided in 17 layers. Layers are divided into two groups: the feature extractor and the classifier. The feature extraction has 7 blocks of 2 layers. Every layer is a stack of a 3x3 convolution with stride 1x1 and padding 1x1 followed by a batch normalization and a ReLU activation function. Between every block a 2x2 max-pooling operation of stride 2x2 is applied. After the 7 blocks of feature extraction, the RF of the network has grown till reaching 637x637, that is approximately equal to the input size 640x640 (see fig 8 to see the RF of every layer). Afterwards, the classification phase takes place using a 2x2 convolution. A 4x4 average-pooling reduces the dimensionality to get a final 64 feature vector that are linearly combined to obtain the output scores of every class. A soft-max function allows the conversion of the scores to probabilities to feed the values to the proper cost function during the optimization process. The feature extractor has 16 filters in the first block, 32 in the second and 64 in all the other.

4.3. Training Procedure

The training set has 75,650 images and the validation set used for hyper-parameter selection 3,000. Notice that the image set is highly imbalanced. In order facilitate the learning, the training set is artificially equalized using data augmentation techniques [20] based on $0 - 180°$ random rotation, X and Y mirroring and contrast and brightness random sampling.

A random initialization based in the Kaiming&He approach [21] is used for all the networks. All models are optimized using a batch based first order optimization algorithm called Adam [22]. The loss function used for optimizing the model is the qwk-loss, with a batch size of 15 and a learning rate of $3x10^{-4}$ [9].

For every batch, the images are chosen randomly from the training set, with repetition. Data augmentation techniques are applied to increase the diversity of the classes (random rotations and brightness and contrast modifications).

After training the network, a linear classifier formed by the combination of the 128 features of the two eyes of the patient is trained. In this way is possible to increase further the prediction performance of the model using all the information available of the patient.

5. Results

5.1. Classification

The model is trained for 300 epochs,reaching a QWK evaluation metric over the validation set of 0.814. The value achieved in the never seen before test set is of 0.801. Using a linear classifier for combining the features of both eyes QWK_{test} reaches 0.844. Expert ophthalmologist report QWK inter-rating agreement values in the $0.80s$. Training the model as a multi-class classification model facilitates the encoding of the required features for distinguishing between the different severity levels of the disease. Training the model for an aggregated detection (grouping the positive classes) will for sure increment the accuracy,

at the prize of missing the coding of important features that separate positive classes (1 to 4). In our case we want the model to learn such a differences to visualize them in the explanation model, that's why in our case is better to use all the information available about the gradation of disease (intermediate classes) in order to force the model to encode the required features for separating between the intermediate classes at the prize obviously of reducing accuracy in the correct predictions. In this way after back-propagating the explanations we could get the scores that the model gives in the evaluation of the different classes for the same image, allowing the expert to include its own expertise in the final decision.

5.2. Explanations

In this subsection we describe the steps followed in the score calculation of a test set sample. After that we present also a set of test images with its calculated score maps for the predicted class.

The image shown in fig. 11, is tagged in the test set as class 4. After feeding it into the model we get the next classification scores (previous to soft-max): $C_0 = -638.9$, $C_1 = -379.7$, $C_2 = -114.6$, $C_3 = +62.8$ and $C_4 = +167.1$. Being C_4 the highest value, the image is correctly classified as class 4.

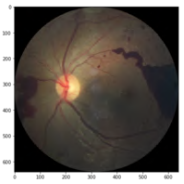


Figure 11: Retine test sample

Fig. 12 shows layer 16 individual feature scores. It can be observed in the bar plot how the same features score different for the prediction of the different classes.

For visualization purposes, layer scores are presented considering the layer as a unique block combination of *convolution - batch normalization - ReLU*. The output of this function block can be mathematically expressed as:

$$O = max(0, \beta + \gamma(\frac{WI + b - \mu}{sigma}) \tag{10}$$

being the output score:

$$S_O = \lambda(\beta + \gamma\frac{b - \mu}{\sigma}) + \lambda\frac{\gamma}{\sigma}WI \tag{11}$$

In this way the output score can be splitted in the two parts: the score input, $S_I = \lambda\frac{\gamma}{\sigma}WI$ and the constant score of the layer, $S_k = \lambda(\beta + \gamma\frac{b-\mu}{\sigma})$.

Figures 13, 14, and 15 show the aggregated scores of every hidden layer and of the final output layer. Individual feature scores are first calculated, *receptive field-wise* summed up and mapped into input-space (section 3.7). The same is done for $S_k^{(l)} \quad \forall l \in L$. Figures 16, 17 and 18 show the S_k of all the hidden layers. Fig. 15d show the part of score that depend exclusively of the input.

Score inputs can be combined with constant scores to define a unique input score map (see fig. 19). The sum of these scores is equal to the last layer inference score and determines the relative importance of every pixel in the final

decision. A density plot and a standard deviation can also be calculated. In order to determine the importance of pixels is possible to restrict the visualization to positive scores or also be even more restrictive and visualize only pixels with a score greater than a predefined threshold, for example $n\sigma$ (see fig. 19f). These score maps are useful for building explanations, for detecting the cause of non-expected classifications, for example pixels with excessive importance in the final decision, conclusions based only on partial or incorrect information, etc.

Fig. 20 shows three different score maps generated for images belonging to diverse classes. For an appropriate analysis the score maps of every class should be considered and different threshold maps should be analyzed. In this figure due to space limitations only the predicted class map is shown. In future publications we will study the best method to extract conclusions of the generated maps.

6. Conclusions

In this paper we presented a new model for the explanation of deep learning classification models based on the distribution of the last layer scores between the input pixels of the image. We presented a general theoretical derivation of the score calculation for the more typical deep learning building blocks to make possible the generation of score propagation networks for any type of applications based on deep learning models. Additionally, we applied the model to design *a human expert performance level DR interpretable classifier*. A model able to classify retine images into the five standardized levels of disease severity and able also to report, for every class, score importance pixel maps, providing the human expert the possibility of inference and interpretation. The score generation is done using a modified version of the pixel-wise relevance propagation algorithm, with the key difference of back-propagating only the part of the score that depends on the inputs and leaving the constant part as a contribution to the score of the considered layer. In this way, we are able to generate scores in a unique and exact way. Additionally, we developed a technique, consisting on applying a 2d-gaussian prior over the RFs, for mapping the constant hidden-space scores to the input, for generating a unique score map representative of the class, making possible to distribute the 100% score class information of the last layer.

Acknowledgements

This work is supported by the URV grants 2014PFR-URV-B2-60 and 2015PFR-URV-B2-60, as well as, for the Spanish research projects PI15/01150 and PI12/01535 (Instituto de Salud Carlos III). The authors would like to thank to the Kaggle and EyePACS for providing the data used in this paper.

References

References

[1] Y. Lecun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.

[2] J. Schmidhuber, Deep learning in neural networks: An overview, Neural Networks 61 (2015) 85–117.

[3] Y. Bengio, A. Courville, P. Vincent, Representation learning: A review and new perspectives, IEEE Trans. Pattern Anal. Mach. Intell. 35 (8) (2013) 1798–1828.

[4] Y. Bengio, Learning deep architectures for AI, Foundations and Trends in Machine Learning 2 (1) (2009) 1–127.

[5] M. D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, in: European conference on computer vision, Springer, 2014, pp. 818–833.

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

[7] G. Montavon, S. Lapuschkin, A. Binder, W. Samek, K.-R. Müller, Explaining nonlinear classification decisions with deep taylor decomposition, Pattern Recognition 65 (2017) 211–222.

[8] C. Wilkinson, F. Ferris 3rd, R. E. Klein, P. P. Lee, C. D. Agardh, M. Davis, D. Dills, A. Kampik, R. Pararajasegaram, J. T. Verdaguer, Global diabetic retinopathy project group. proposed international clinical diabetic retinopathy and diabetic macular edema disease severity scales, Ophthalmology 110 (9) (2003) 1677–1682.

[9] J. de la Torre, D. Puig, A. Valls, Weighted kappa loss function for multi-class classification of ordinal data in deep learning, Pattern Recognition Letters`doi:https://doi.org/10.1016/j.patrec.2017.05.018`.
URL `http://www.sciencedirect.com/science/article/pii/S0167865517301666`

[10] J. Cohen, Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit., Psychological bulletin 70 (4) (1968) 213.

[11] G. V, P. L, C. M, et al, Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs, JAMA 316 (22) (2016) 2402–2410. `arXiv:/data/journals/jama/935924/joi160132.pdf`, `doi:10.1001/jama.2016.17216`.
URL `+http://dx.doi.org/10.1001/jama.2016.17216`

[12] K. Simonyan, A. Vedaldi, A. Zisserman, Deep inside convolutional networks: Visualising image classification models and saliency maps, CoRR abs/1312.6034. arXiv:1312.6034.
URL http://arxiv.org/abs/1312.6034

[13] D. Baehrens, T. Schroeter, S. Harmeling, M. Kawanabe, K. Hansen, K.-R. MÃžller, How to explain individual classification decisions, Journal of Machine Learning Research 11 (Jun) (2010) 1803–1831.

[14] W. Luo, Y. Li, R. Urtasun, R. Zemel, Understanding the effective receptive field in deep convolutional neural networks, in: Advances in Neural Information Processing Systems, 2016, pp. 4898–4906.

[15] J. de la Torre, A. Valls, D. Puig, Diabetic retinopathy detection through image analysis using deep convolutional neural networks, in: À. Nebot, X. Binefa, R. L. de Mántaras (Eds.), Artificial Intelligence Research and Development - Proceedings of the 19th International Conference of the Catalan Association for Artificial Intelligence, Barcelona, Catalonia, Spain, October 19-21, 2016, Vol. 288 of Frontiers in Artificial Intelligence and Applications, IOS Press, 2016, pp. 58–63. doi:10.3233/978-1-61499-696-5-58.
URL https://doi.org/10.3233/978-1-61499-696-5-58

[16] J. de la Torre, D. Puig, A. Valls, Weighted kappa loss function for multi-class classification of ordinal data in deep learning, Pattern Recognition Letters.
URL https://doi.org/10.1016/j.patrec.2017.05.018

[17] G. E. Dahl, T. N. Sainath, G. E. Hinton, Improving deep neural networks for LVCSR using rectified linear units and dropout, in: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2013, pp. 8609–8613.

[18] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, CoRR abs/1502.03167.

[19] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016.

[20] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: F. Pereira, C. J. C. Burges, L. Bottou, K. Q. Weinberger (Eds.), Advances in Neural Information Processing Systems 25, Curran Associates, Inc., 2012, pp. 1097–1105.

[21] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: 2015 IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1026–1034.

[22] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, CoRR abs/1412.6980.

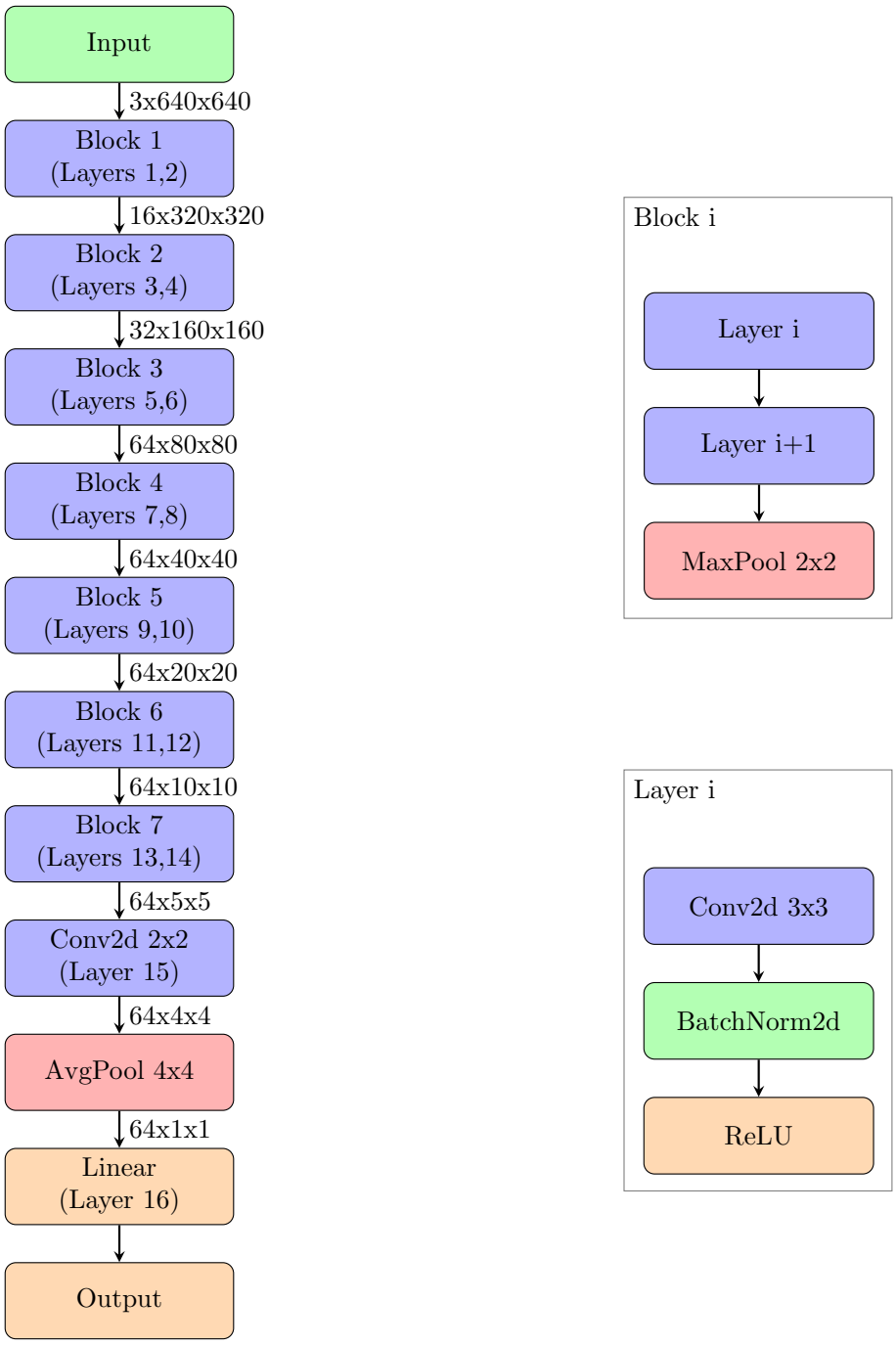


Figure 10: Classification model for DR prediction

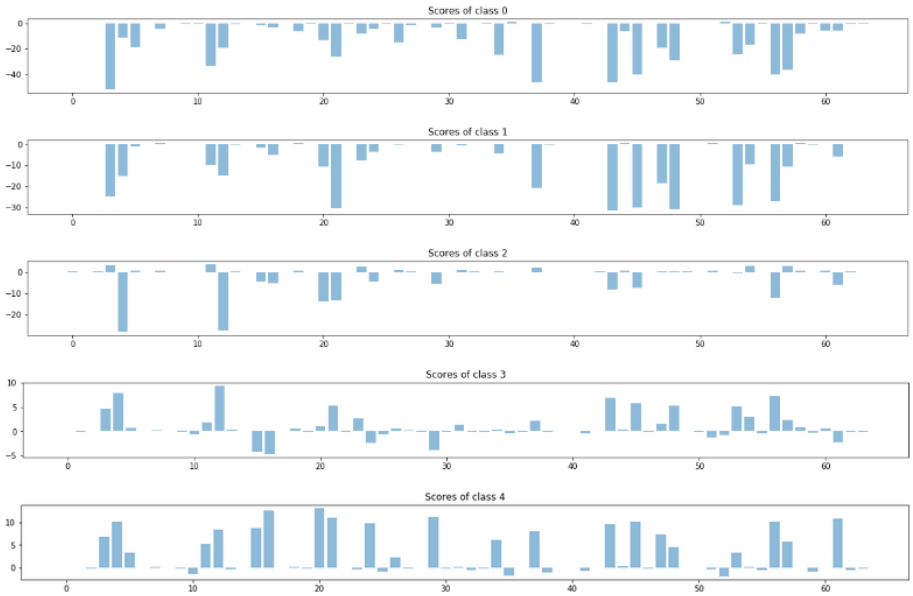
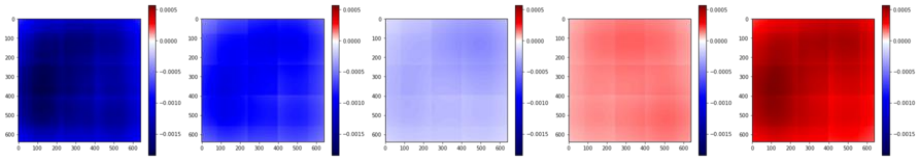
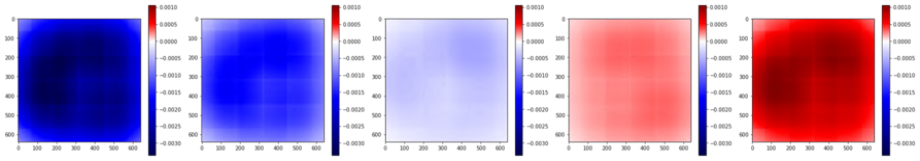


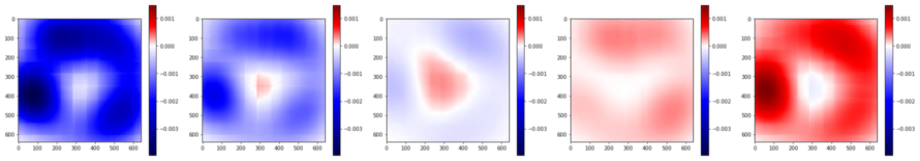
Figure 12: Layer 16 feature scores for considered test sample



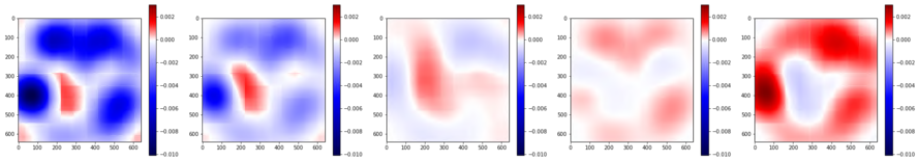
(a) $S^{(15)}$, RF=637x637



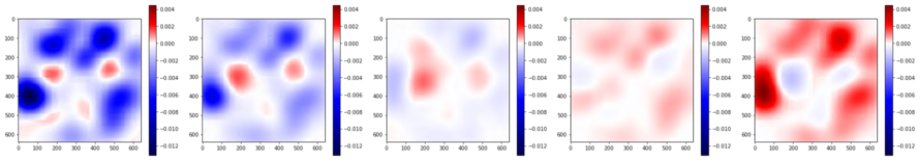
(b) $S^{(14)}$, RF=509x509



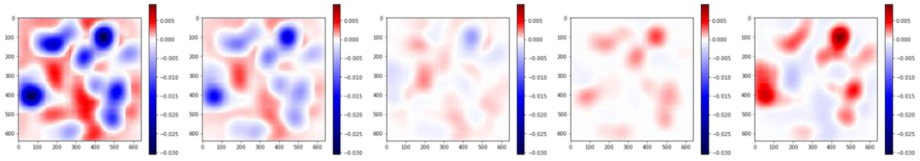
(c) $S^{(13)}$, RF=381x381



(d) $S^{(12)}$, RF=253x253

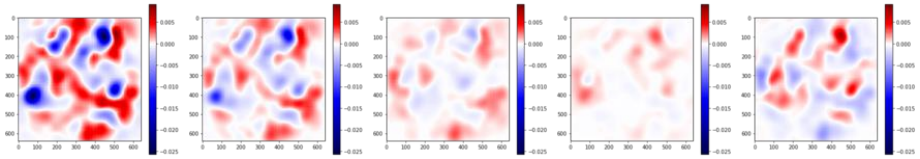


(e) $S^{(11)}$, RF=189x189

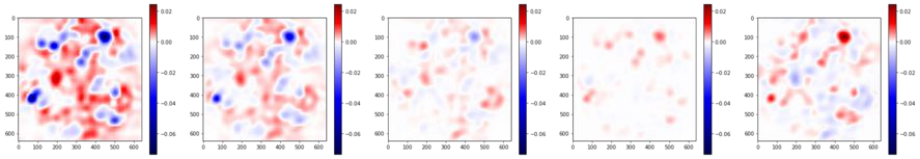


(f) $S^{(10)}$, RF=125x125

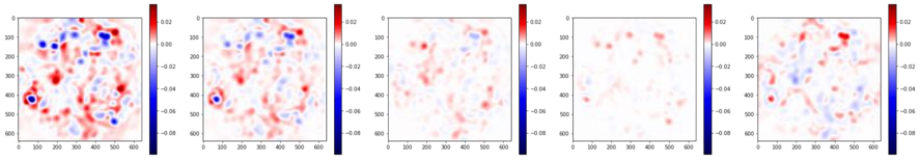
Figure 13: Full explanation of the classification of test retine image (layers 15-10). From left to right aggregated score maps for class 0 to class 4 of every referred layer



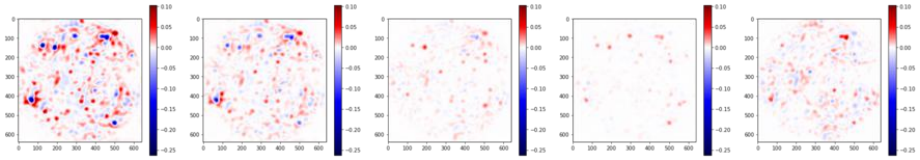
(a) $S^{(9)}$, RF=93x93



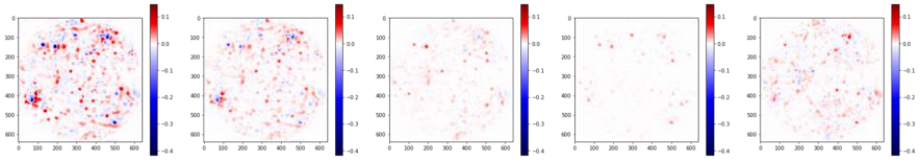
(b) $S^{(8)}$, RF=61x61



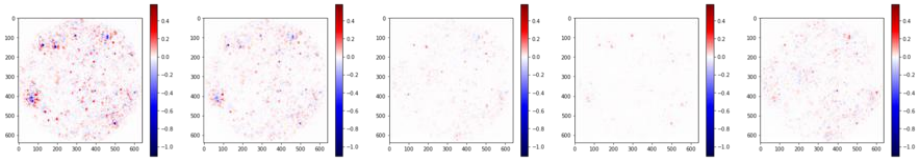
(c) $S^{(7)}$, RF=45x45



(d) $S^{(6)}$, RF=29x29

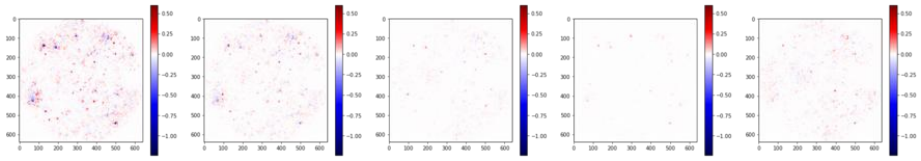


(e) $S^{(5)}$, RF=21x21

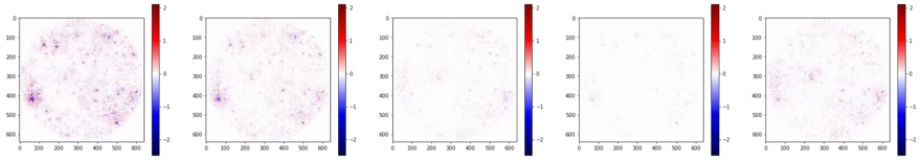


(f) $S^{(4)}$, RF=13x13

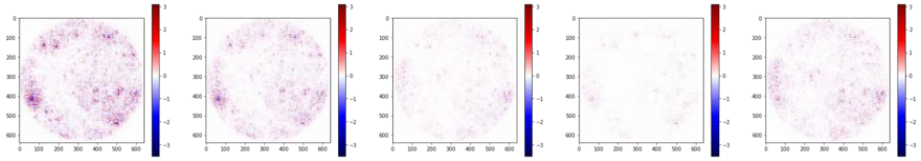
Figure 14: Full explanation of the classification of test retine image (layers 9-4). From left to right aggregated score maps for class 0 to class 4 of every referred layer



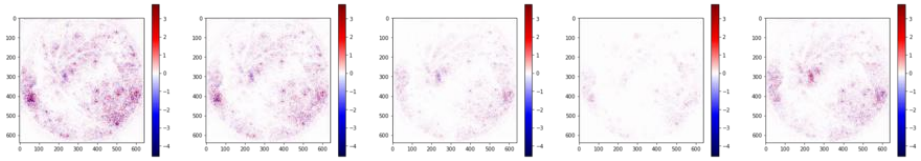
(a) $S^{(3)}$, RF=9x9



(b) $S^{(2)}$, RF=5x5

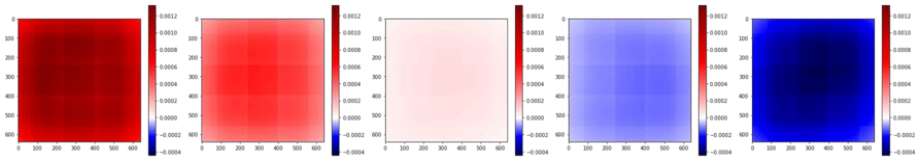


(c) $S^{(1)}$, RF=3x3

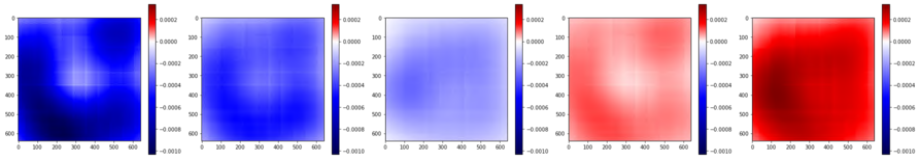


(d) $S^{(input)}$

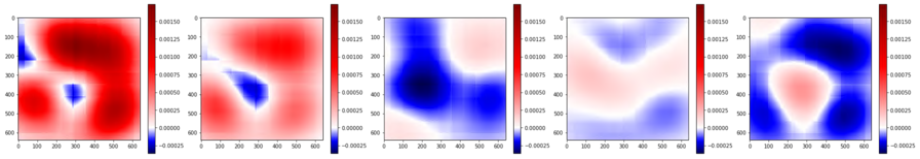
Figure 15: Full explanation of the classification of test retine image (layer 3-0). From left to right aggregated score maps for class 0 to class 4 of every referred layer



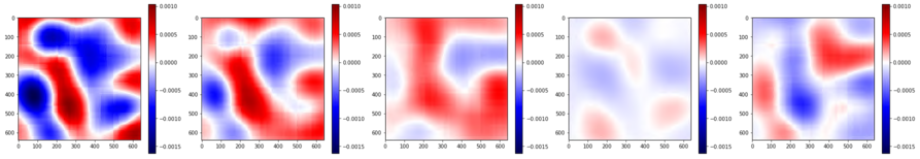
(a) $S_k^{(15)}$, RF=637x637



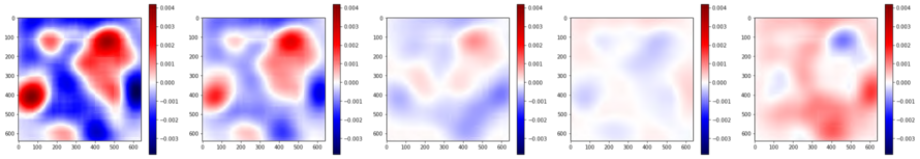
(b) $S_k^{(14)}$, RF=509x509



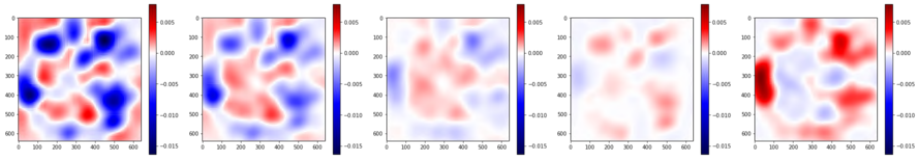
(c) $S_k^{(13)}$, RF=381x381



(d) $S_k^{(12)}$, RF=253x253

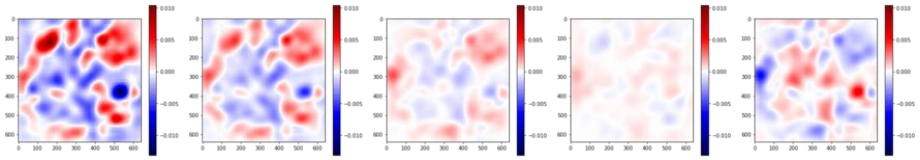


(e) $S_k^{(11)}$, RF=189x189

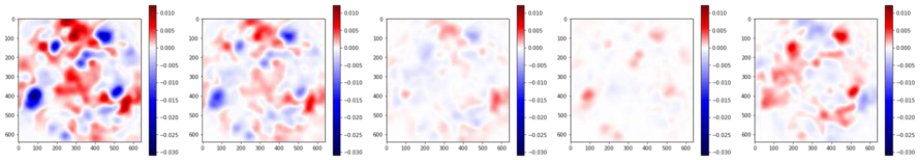


(f) $S_k^{(10)}$, RF=125x125

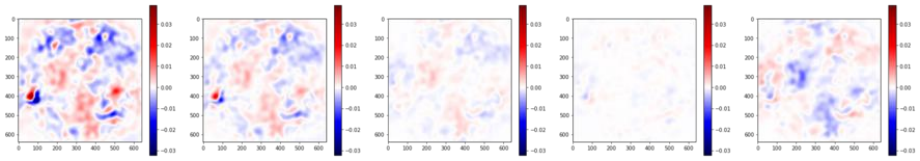
Figure 16: RF dependent constant scores for test sample (layers 15-10). From left to right aggregated score maps for class 0 to class 4 of every referred layer



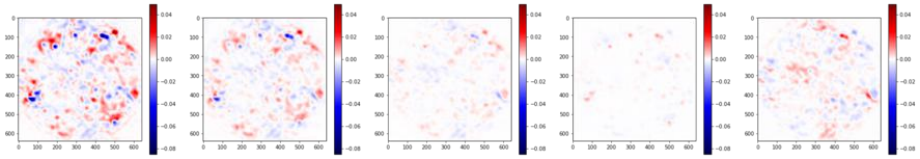
(a) $S_k^{(9)}$, RF=93x93



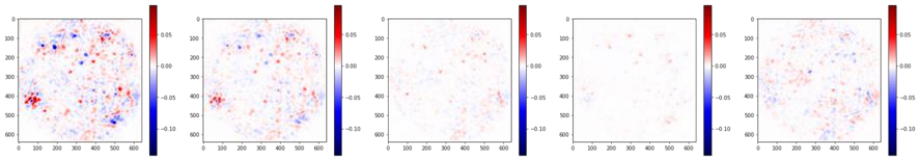
(b) $S_k^{(8)}$, RF=61x61



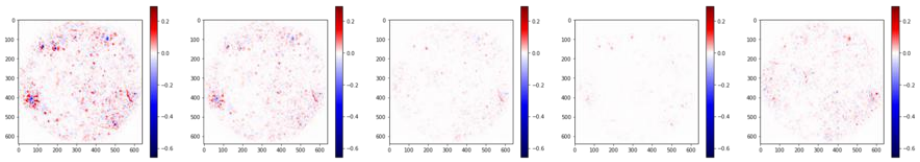
(c) $S_k^{(7)}$, RF=45x45



(d) $S_k^{(6)}$, RF=29x29

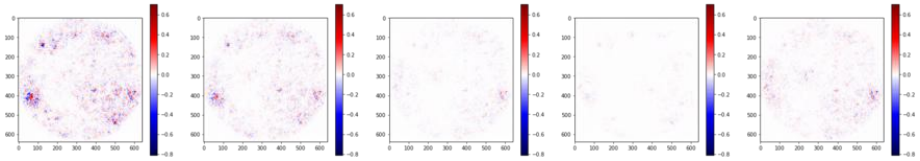


(e) $S_k^{(5)}$, RF=21x21

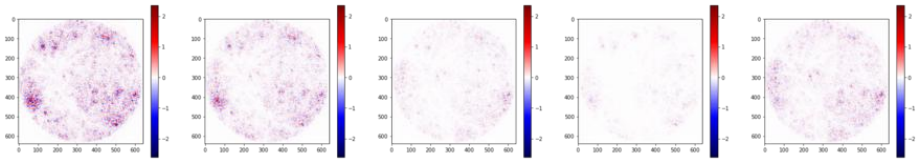


(f) $S_k^{(4)}$, RF=13x13

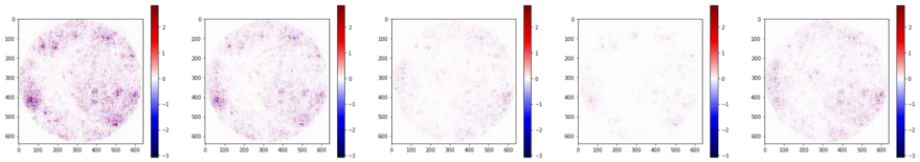
Figure 17: RF dependent constant scores for test sample (layers 9-4). From left to right aggregated score maps for class 0 to class 4 of every referred layer



(a) $S_k^{(3)}$, RF=9x9

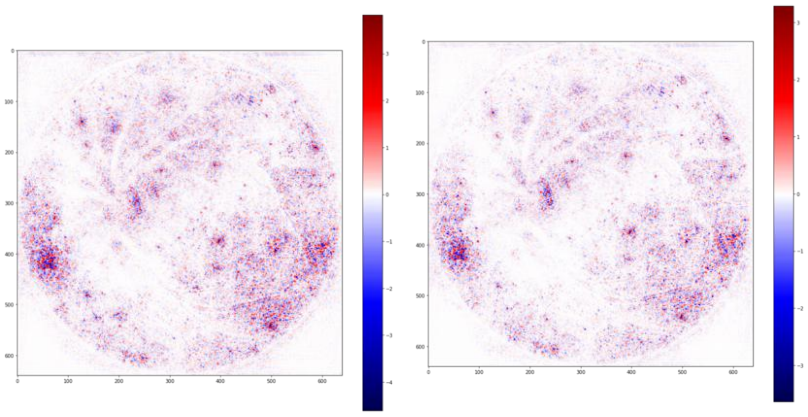


(b) $S_k^{(2)}$, RF=5x5



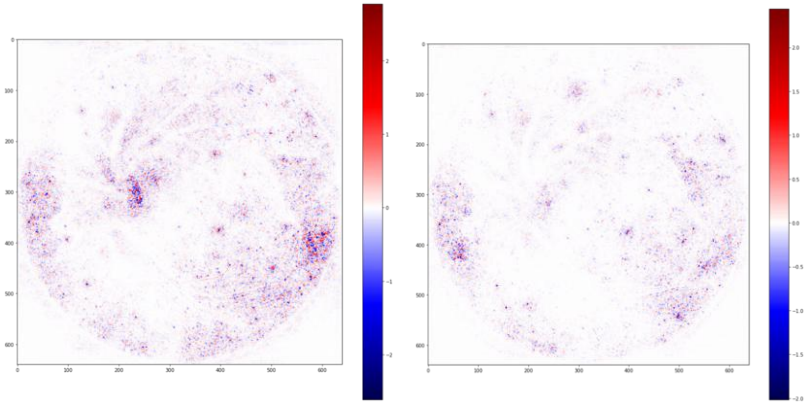
(c) $S_k^{(1)}$, RF=3x3

Figure 18: RF dependent constant scores for test sample (layers 3-1). From left to right aggregated score maps for class 0 to class 4 of every referred layer



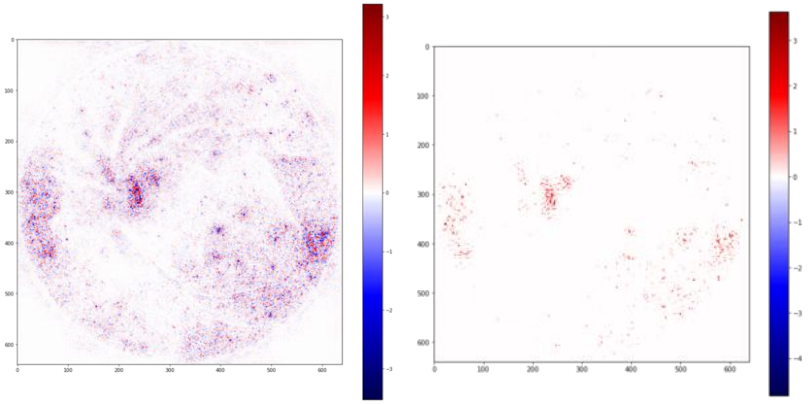
(a) Total score for C_0

(b) Total score for C_1



(c) Total score for C_2

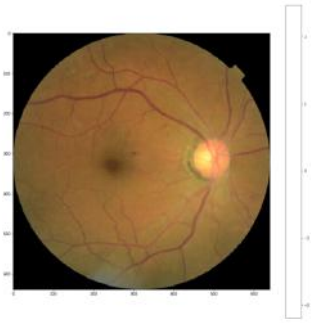
(d) Total score for C_3



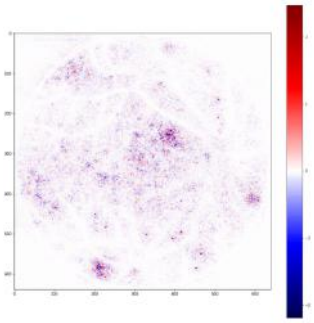
(e) Total score for C_4

(f) Total score for C_4 ($S^{(in)} > 2\sigma$)

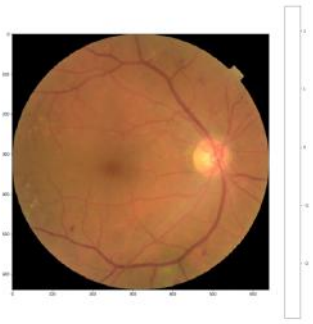
Figure 19: Total score input-space distribution for classes 0, 1, 2, 3, 4 for test sample



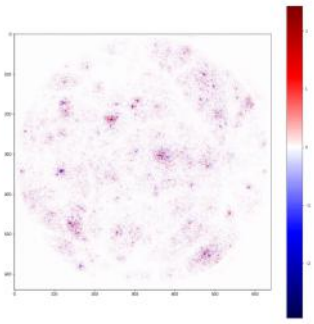
(a) A C_2 sample



(b) Score map generated for C_2 sample

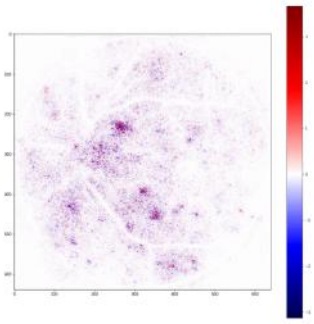


(c) A C_3 sample



(d) Score map generated for C_3 sample

(e) A C_4 sample



(f) Score map generated for C_4 sample

Figure 20: Total score maps generated for different input images for the predicted class (class of maximum score)