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. The main weak point is its lack of interpretable explanations about the reported results, although they are able to give results with high statistical confidence. The vast amount of parameters of these models make difficult to infer a rationale interpretation from them. In this paper we present an interpretable classifier able to classify retina images into the different levels of diabetic retinopathy severity with good performance, as well as of explaining its results by assigning a score for every point in the hidden and input spaces, evaluating its contribution to the final classification in a linear way. The generated visual maps can be easily interpreted by an ophthalmologist in order to find the lesions present in the retina that are causing the disease.

Keywords: deep learning, classification, explanations, diabetic retinopathy, model interpretation

2010 MSC: 68T10

*Corresponding author

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

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 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, a weight is multiplied by the input value and the aggregation over all inputs is feeded to an activation function that calculates the neuron output. Parameters are usually optimized using a stochastic gradient descent based algorithm that minimizes a predefined loss function. The network parameters are updated after back-propagating 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 labeled data (order of magnitude of tens of thousands of examples per class) but their main concern is its *lack of interpretability*. Every successful model tends 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 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 probability of a good diagnostic, which may have a significant influence in patient's treatment.

Different attempts have been done for interpreting the results reported by neural networks. In [5] a network propagation technique is used for input-space feature visualization. After this work, [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 the inherent function topology for applying a message passing technique, propagating back into the pixel space the last layer output values. After this work, 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 this paper we follow a similar approach to the one used in the pixel-wise decomposition, taking into account the compositional nature of the topology as in [5] and [6]. The concept of *score* in our paper is similar to the concept of *relevance* used in layer-wise relevance propagation method. The novelty of our approach comes from the fact that we hypothesize there are two sources that contribute to output score: one is the input-space contribution, and the other is a contribution coming from the receptive fields of each layer. This last contribution depends solely on the parameters of each layer ie., it is independent from the input-space. Therefore, it is not an attribute of the individual pixels that has to be propagated back but a contribution of the receptive field (RF)

that represents the layer as an individual entity. Thus, we only propagate back the score part that depends on the precedent input for every layer. In our explanation model 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 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 our 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 retina: 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.

The paper presents 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 it 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 presents the complete mathematical formulation of our interpretable model, describing the score propagation model. Section 4 describes the DR deep learning classification model. Section 5 presents the results showing a set of examples of the type of visual interpretations that are obtained with our method. Finally, Section 6 gives the final conclusions of our work.

2. Related Work

Many deep learning based DR classifiers have 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 binary deep learning classifier was published for the detection of the most severe cases of DR (grouping classes 0 and 1 of DR on one side, and classes 2, 3 and 4 on another). 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 final decisions

making difficult to the experts to balance and compare its prior knowledge with the reasons behind final conclusions 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 $\frac{\partial f(I)}{\partial I_{c,i,j}} \quad \forall c, i, j$, where I is the input image and $f(I)$ is the deep learning function. These matrices are easy to calculate for deep neural networks because 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 these models is that being the objective to explain $f(x)$, $\frac{\partial f(I)}{\partial I_{c,i,j}}$ is only giving 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 total score of a classification into individual *relevance scores* R_d that act as a positive or negative contributions to the final result.

The method has the next general assumptions: 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 incoming

relevance messages of such node and also equal to the sum of all the outgoing relevance messages from the same node (eq. 2).

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

where: $R_d^{(l)}$ the relevance of node d in layer l

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

where: $R_{i \leftarrow k}^{(l,l+1)}$ relevance message passing from node k located in layer $l+1$ to node i, located in layer l; a_i activation of node i and ω_{ij} weight value connecting node i and j.

2.3. Taylor-type decomposition

Another way for solving the interpretability problem is using the classification function gradient for calculating 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 concern 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 an approximation that combines the layer-wise and Taylor type models. Taking advantage of the compositional nature of deep learning models, this approach supposes also the decomposability of 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 intermediate relevances to eventually reach the input-space and come to a heatmap of the total relevance of the prediction.

3. Receptive Field Score Distribution Model

In this section we describe an alternative method for output score distribution in deep learning models. We hypothesize that the output score of a particular classification is not only due to pixel score contributions but also due to all the contribution of the receptive fields analyzed by the network. In following sections we will prove that for all typical neural network types of layers the score output can be expressed as the sum of layer's constants plus a value proportional to the layer's input. Pixel-wise explanation model uses a similar approach for propagating back the scores but with an important difference: pixel-wise [6] propagates back not only the input layer contribution but also constants, ie. bias, etc. Such a way of propagating back the score, breaks the linear transformation. A way of maintaining the linear transformation is propagating back solely the part that depends on the input, leaving the other part

as a contribution of the layer, ie. of the receptive field. As the biases are not propagated, the normalization term used in pixel-wise method is not required anymore. Although neural network design is non-linear, score back-propagation is linear over the activated nodes. Nonlinear phase takes place in the forward pass, but the backward part only takes place over the activated nodes. In our formulation, we consider the score entering into a node as the combination of two parts: one that can be transformed into a linear function dependent on the activated inputs and another one that is constant for the considered node. Final score is expressed as the sum of contributions of all nodes belonging to the studied feature space (or pixel space) plus a constant score for each node contribution to every following layer. Such node constant scores depend on layer parameters and, in some way, on the output activations of previous layer. Thus, the propagated score depends solely on the individual activation inputs of the layer. In such a way, we are able not only to find a unique way for mapping the score of every output onto the network input-space but also and more importantly to find a linear relationship between output scores and image scores. This last property is due to the fact of using exclusively linear transformations in all layers.

In this work, the following propositions are assumed:

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

$$S_l = \lambda_l a_l \tag{5}$$

being S_l the tensor representative of all the scores of layer l, a_l the activations tensor and λ_l another tensor establishing the required relationship between S_l and a_l.

Proposition 2. The score observed as output in one layer can be decomposed in two parts, one dependent on the previous layer score S_{l-1} and another one S_{k_l} that is constant because it does not depend on the input activations but only on the model parameters that are used in that layer:

$$S_{l+1} = S_l + S_{k_l} \tag{6}$$

The propagation model proposed (see fig. 1) makes a different treatment of the components S_l and S_{k_l}.

First, S_l is a linear transformation of the layer activation and S_{k_l} is a tensor of the same dimensions independent from the activation that acts shifting the score coming from the previous layer S_{l-1}. In the following subsections we explain how to obtain S_l and S_{k_l} for different typical block constituents of deep learning networks.

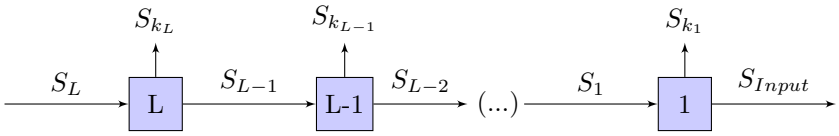


Figure 1: Score distribution through layers

Output score is expressed then as:

$$S_L = \sum_{l=1}^{L} \left(\sum S_{k_l} \right) + \left(\sum S_{Input} \right) \tag{7}$$

being S_L last layer feature score, S_{k_l} the constant tensors of each layer, $\sum S_{k_l}$ the element-wise sum of scores and $\sum S_{k_l}$ the pixel-wise sum of scores.

Second, if required, S_{k_l} values obtained from each of the layers can also be mapped into the input-space using an additional final procedure described in section 3.7.

Receptive field pixel maps allow the identification of image parts that the network is considering important. In our case study, diabetic retinopathy classification, it is not only important to detect the pixel-size micro lesions but also the clusters of lesions localized in different parts of the image. Taking into account each receptive field contributions apart from the input, facilitates the detection of such clusters and not only the individual contributions of each pixel.

3.1. Score propagation through an activation function node

In fig. 2 we show the activation function node. A input activation a_i is transformed onto the output activation as $a_o = \phi(a_i)$. Taking $S_o = \lambda_o a_o$, and

substituting a_o we get $S_o = \lambda_o \phi(a_i)$. According to proposition 1, we have 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 parameter-wise 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:

$$S_o = \lambda_o[\phi(a_i^*) - \phi'(a_i^*)a_i^*] + \lambda_o \phi'(a_i^*)a_i \tag{8}$$

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

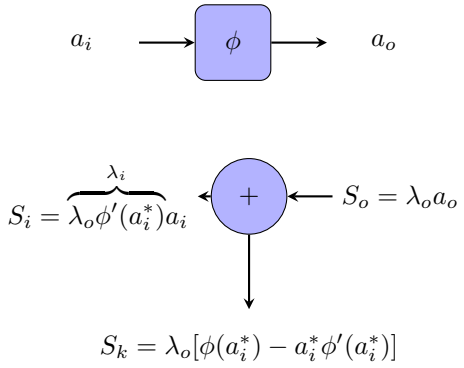


Figure 2: Score propagation through an activation function node

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:

$$S_o = \lambda_o(\beta - \gamma\frac{\mu}{\sigma}) + \lambda_o\frac{\gamma}{\sigma}a_i \tag{9}$$

As we see, the output score can be exactly split 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. 3)

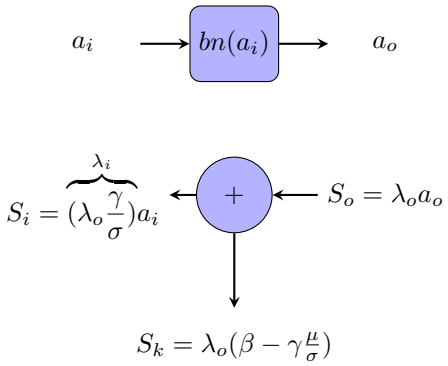


Figure 3: Score propagation through an batch normalization node

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. 4). 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.

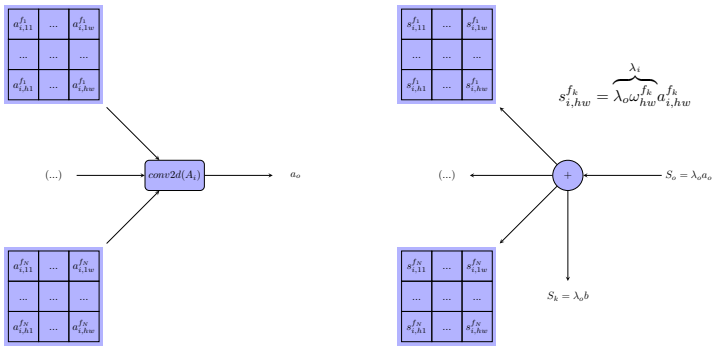
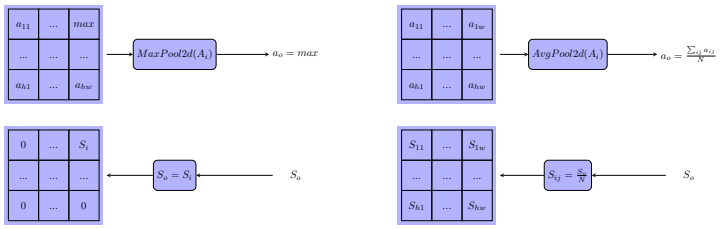


Figure 4: Convolution score calculation. Score spreads into the different inputs. The bias related part of the score is not backpropagated.

3.4. Score propagation through pooling layers

The score propagation through a max-pooling layer is straightforward. The output score value is copied into the input that was selected in the forward pass (see fig. 5). For average pooling is also straightforward. The score value is split into N equal parts, being N the number of inputs (see fig. 5).



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

Figure 5: Score propagation through different pooling layers

3.5. Score propagation through a fully connected layer

A fully connected layer is a linear combination of the input activities and the weights. Final score is split into the individual elements leaving apart the bias that becomes the constant score contribution of the own layer (see fig. 6).

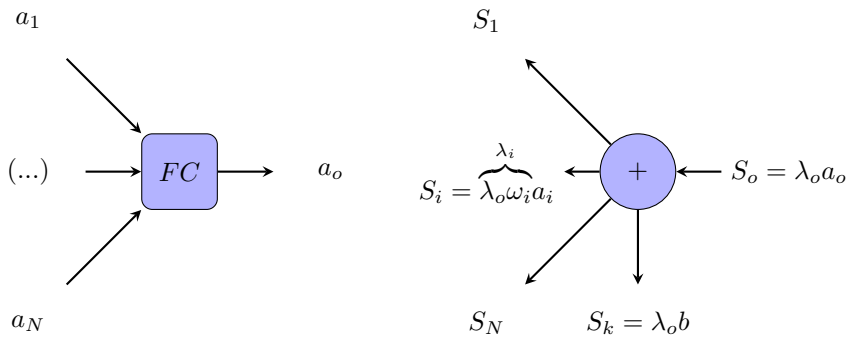


Figure 6: Score propagation through a fully connected node

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 score conservation through the node ($S_o = S_i$ in this case, due to the absence of constant score) we get that the final equation:

$$\lambda_i = \lambda_o(1-d) \tag{10}$$

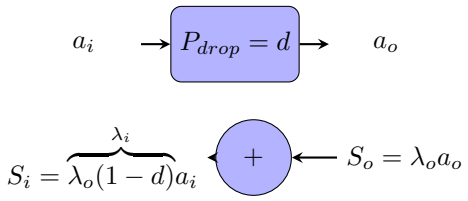


Figure 7: Score propagation through a dropout node

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

We know that every node has two score constituents: one input-dependent, that can be easily forwarded, and another one RF-dependent, i.e the layer-dependent. At this point we are going to describe a method to transport back also such last values onto 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 prior information 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 each one of such points is a representation value of a 189x189 RF 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 hidden-space values and summing up the coincident points is possible to map the distribution onto 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 for Diabetic Retinopathy

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 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 dataset is split in two disjoint sets containing eye images of different patients, one for training and the other for testing.

The training set contains a total of $75,650$ images; $55,796$ of class 0, $5,259$ of class 1, $11,192$ of class 2, $1,805$ 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 contains a total of $10,000$ images of patients not present in training set; $7,363$ 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. Construction of the classifier

The model calculates $P(\mathcal{C}|\mathcal{I})$, being $\mathcal{C}$ the class to predict and $\mathcal{I}$ the retina image. Using as a last layer a $SoftMax$ function over the values after the last linear combination of features. This probability is calculated as $P(\mathcal{C}|\mathcal{I}) = \frac{e^{S_i}}{\sum_{j=1}^{C} e^{S_j}}$. Let us call S_C the score of 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 do not need evaluate $Softmax$ because $argmax(S_i) = argmax(softmax(S_i))$.

4.2.1. Design guidelines for DR classification

Deep neural network model design up to know is driven mainly by experience. Nowadays it 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 [9]) 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.

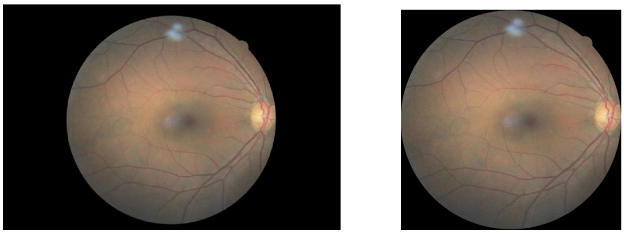
Using the previous experience in such works we summarize a set of guidelines that ruled the final model selection. These design principles applicable to this the DR particular application, and that are explained below, are: use an optimal image resolution, use all the image information available, use 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 image input size has a great importance in the classification results. For this problem in other papers like [15] is shown that better results can be achieved with retina 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 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. For this present work we tested models of 128, 256, 384, 512, 640, 724, 768 and 892 pixels of retina 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 retina 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 retina. A 512x512 input image model was used with a random selection of a rotated square (diagonal equal to the retina diameter). In this way only a 64% of the retina 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 8).

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.



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

Figure 8: A training sample showing the preprocessing treatment

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 idea only 3x3 convolutions have been used in feature layers.

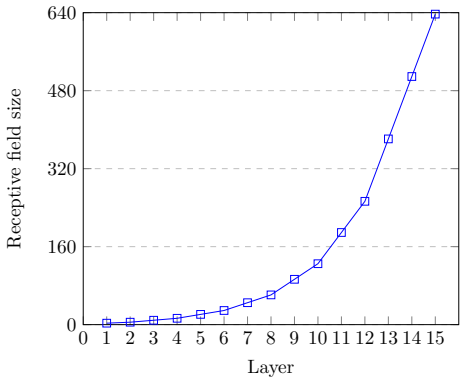


Figure 9: Model RF growth

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 9 shows the RF growth of our model.

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[16]. We have tested other activation functions such as LeakyReLU, ELU and SeLU reporting similar and even worse results, introducing complexity to the model without adding any significant advantage to the final result.

Use batch normalization in every layer. Batch normalization [17] stabilize the training and accelerates convergence. In this application 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 [18]. In [9] is shown that for ordinal regression problems, where not only a multi-class classification is taking place but also it 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 [9]. The difference in performance of the final results is very high. Optimizing directly QWK allows achieving better classification results.

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

Use a efficient number of features. With infinitely number of resources we can use a big network. In our project we have limited device resources and additionally we would like also to be able to implement the result in devices with low resources. Having this in mind we tested networks of different sizes and in

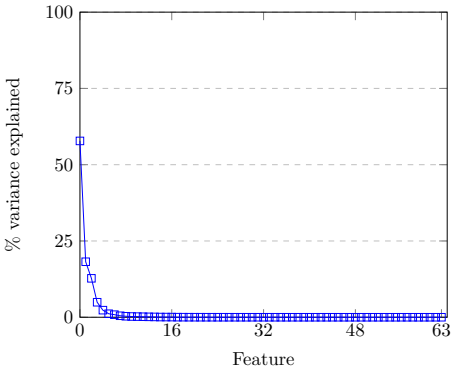


Figure 10: Feature space cummulative PCA variance of training set

order to check the redundancy of 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 10 we show the variance explained by the final feature vector space.

4.2.2. Classification model description

Following the given guidelines, our model uses 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. 8). Figure 11 shows a block diagram of the model. It is a CNN of 391,325 parameters, divided in 17 layers. Layers are divided in 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 9 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 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

As presented in section 4.1, the training set 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 to facilitate the learning, training set is artificially equalized using data augmentation techniques [19] based on $0 - 180°$ random rotation, X and Y mirroring and contrast&brightness random sampling.

A random initialization based in the Kaiming&He approach [20] is used. All models are optimized using a batch based first order optimization algorithm called Adam [21]. 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.

After network training, 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 value of 0.814 on the validation set. The value achieved in the never seen before test set of 10.000 images is of 0.801. Using a linear classifier for combining the features

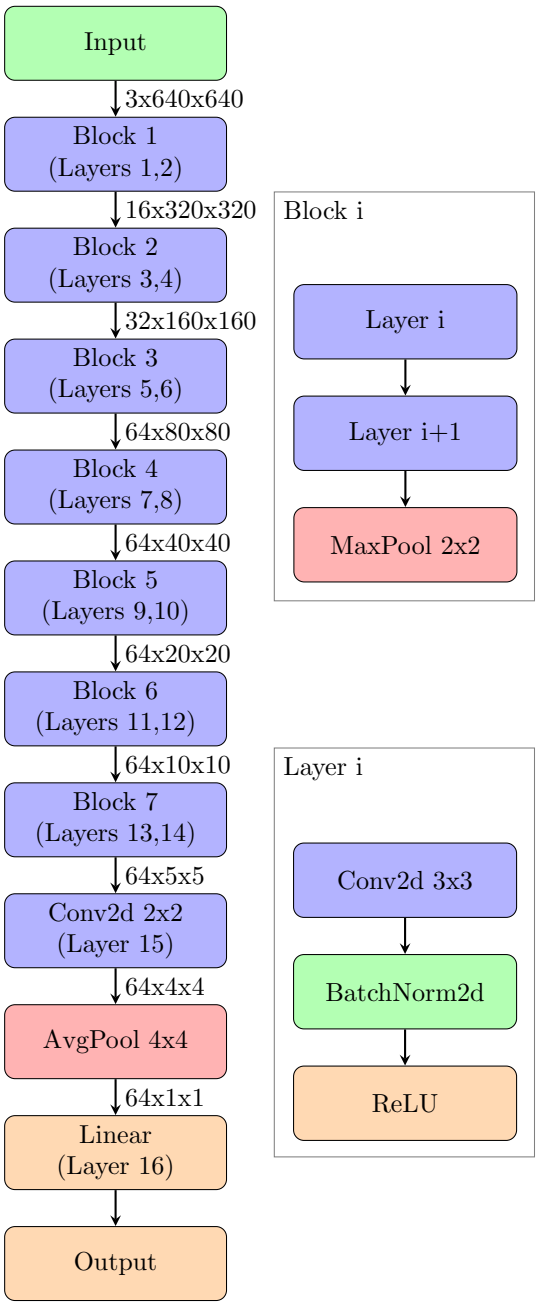


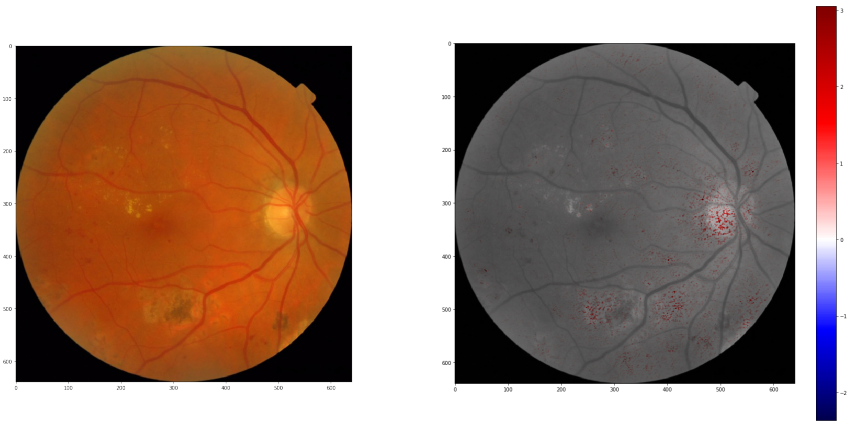
Figure 11: Prediction model

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 disease severity levels. 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 differences to visualize them in the explanation model, that is 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 report 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. Pixel and Receptive Field Map Generation

In this subsection we describe the steps followed in the score calculation of a test set sample. Fig. 12 is tagged in the test set as class 4. For this image the model reports the next classification scores (previous to soft-max): $C_0 = -451.2$, $C_1 = -229.0$, $C_2 = -53.4$, $C_3 = +37.4$ and $C_4 = +73.2$. Being C_4 the highest value, the image is correctly classified as class 4. In fig. 12b we show a black and white version of the image with the distribution of the positive contributions to the class 4 total score in input space. Having the total score map is possible to visualize different versions of it, showing the negative contributions, or the most positive ones, or defining a positive threshold for displaying only the higher scores. In this way is possible to identify the points that contribute the most to a particular classification. There should be a correlation between such points and lesions in the eye if we are studying, as in this sample, a severe disease class.

Figures 13, 14 and 15 show the aggregated scores for different intermediate layers. For visualization purposes, layer scores are presented considering the



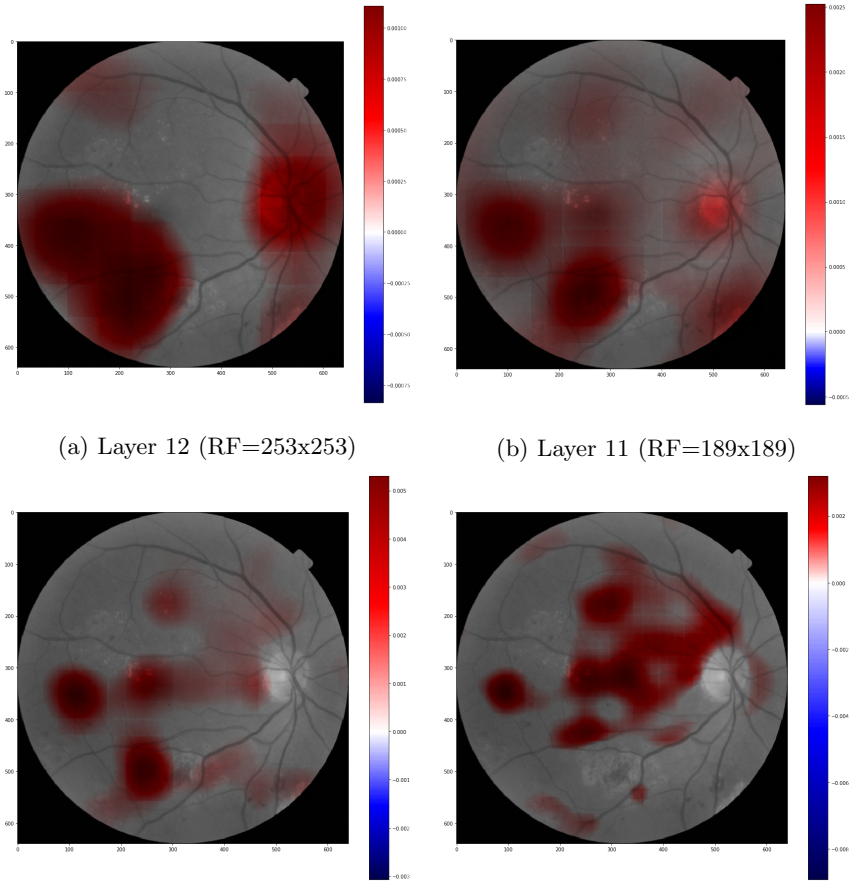
(a) Network input image

(b) Class 4 positive score map

Figure 12: Class 4 sample image

layer as a unique block combination of *convolution - batch normalization - ReLU*. The output of this function block can be mathematically expressed as $O = max(0, \beta + \gamma(\frac{WI+b-\mu}{sigma})$, being $S_O = \lambda(\beta + \gamma\frac{b-\mu}{\sigma}) + \lambda\frac{\gamma}{\sigma}WI$. In this way $S_I = \lambda\frac{\gamma}{\sigma}WI$ and $S_k = \lambda(\beta + \gamma\frac{b-\mu}{\sigma})$.

Individual feature scores are first calculated, *receptive field-wise* summed up and mapped into input-space (section 3.7). The same is done for $S_k^{(l)} \quad \forall l \in L$. Score inputs can be combined with constant scores to define a unique input score map. 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, as noted before, it 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 $n\sigma$. These score maps are useful for interpreting the reasons behind a classification, 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.



(a) Layer 12 (RF=253x253)

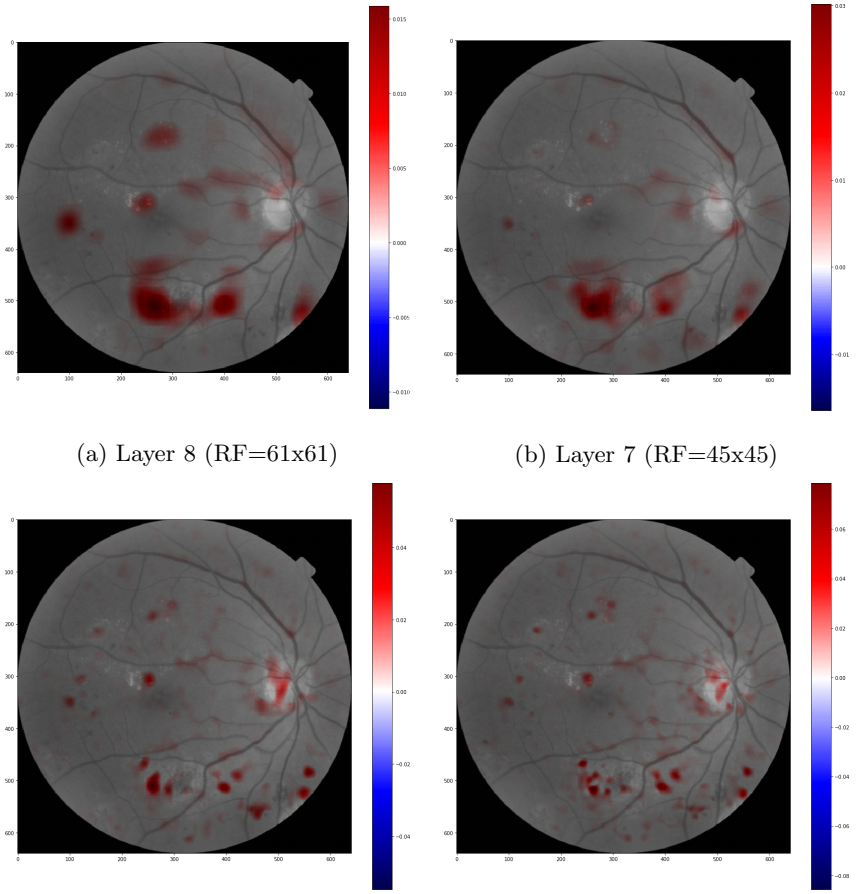
(b) Layer 11 (RF=189x189)

(c) Layer 10 (RF=125x125)

(d) Layer 9 (RF=93x93)

Figure 13: Some of the class 4 intermediate score maps generated for the sample image.

In red we have the zones that are contributing positively to classify the image as class 4. It is possible to plot also the zones contributing negatively to the classification. In case of being analyzing a class 4, negative score zones(not shown) would be zones without lesions. For clarity purposes only positive values are shown. Regions of red and blue pixels are sequentially fine-grained as the size of the receptive field decreases. Having the possibility to choose the RF size, permits to analyze the areas contributing to image classification at different levels of precision.



(a) Layer 8 (RF=61x61)

(b) Layer 7 (RF=45x45)

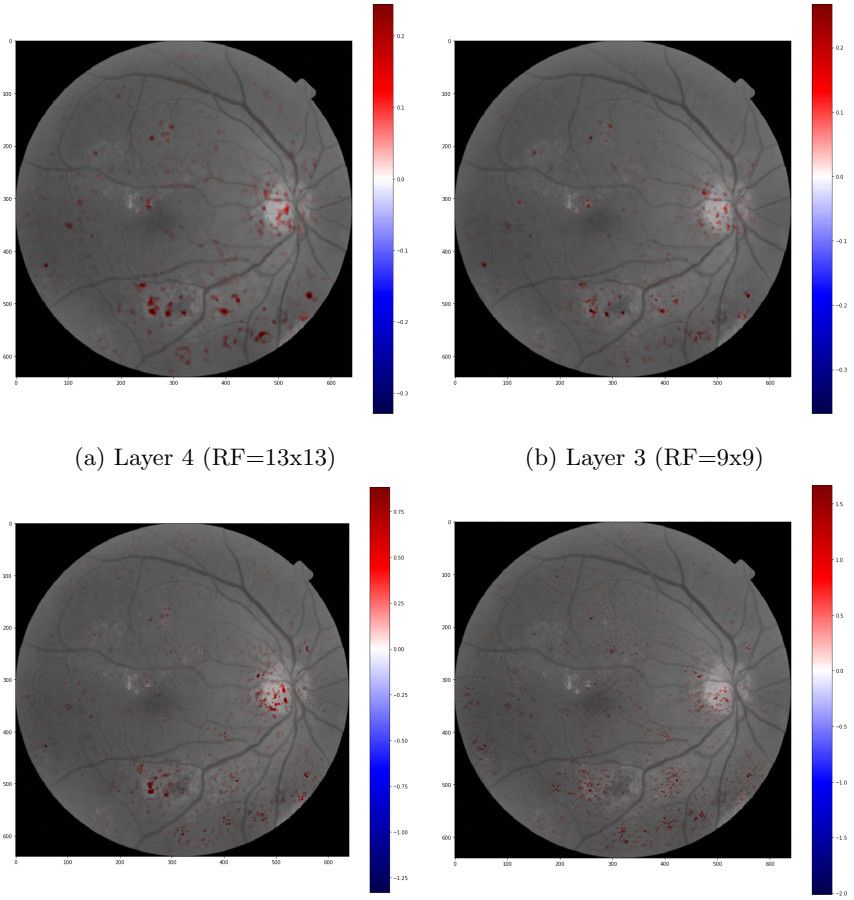
(c) Layer 6 (RF=29x29)

(d) Layer 5 (RF=21x21)

Figure 14: Some of the class 4 intermediate score maps generated for the sample image.

Intermediate scores are also very useful for identifying clusters of microlesions and evaluate its combined weight in the final score. They enable the location of lesion clusters and also zones not affected by the disease. For example, in layer 12, with a receptive field of 253x253 the network identifies two zones with high contribution to class 4 score. In layer 11, where the receptive field is smaller (189x189) we see how the detail is increasing, showing more localized zones. In layer 8, 7, 6, 5 and finally input show increasing level of detail dividing big clusters into smaller ones until reaching the input space where individual

pixels are identified.



(a) Layer 4 (RF=13x13)

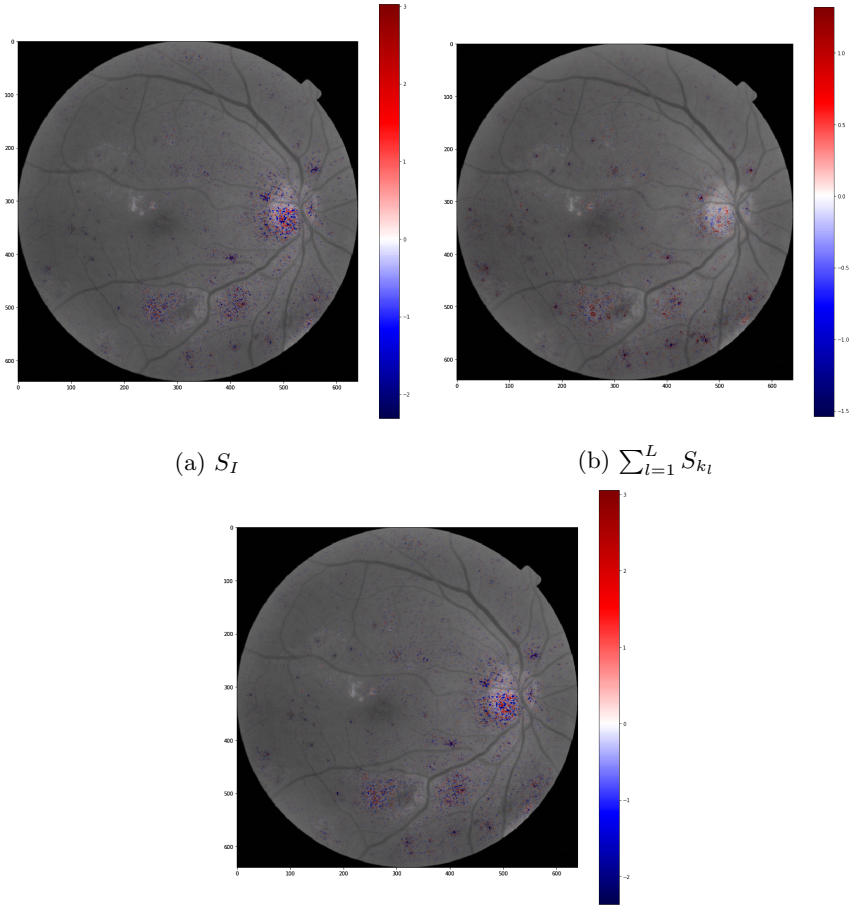
(b) Layer 3 (RF=9x9)

(c) Layer 2 (RF=5x5)

(d) Layer 1 (RF=3x3)

Figure 15: Some of the class 4 intermediate score maps generated for the sample image.

Finally, in figure 16 we present the total score map for the analyzed image plotting not only pixels with positive contribution (red) but also the negative ones (blue). Score input S_I (fig. 16a) represent the map obtained from back-propagating the part that linearly depends on inputs. Score constant $\sum S_k$ (fig. 16b) is the map obtained from summing up S_k for all layers mapped into input space. Score total S_T (fig. 16c) is the sum of the two previous maps and represent the total class 4 score distribution. The sum of all score pixels



(a) S_I (b) $\sum_{l=1}^{L} S_{k_l}$

(c) S_{total}

Figure 16: Total score maps: $S_{total} = S_I + \sum_{l=1}^{L} S_{k_l}$

is equal to the output score. Table 1 shows some statistics of the three score maps. $\sum S_k$ is the score map contributing the most to final total score. This map is flatter than S_I, so qualitatively S_I and S_T are very similar. $\sum S_k$ acts more as a mild shifting of each receptive field considered by the network and S_I acts reinforcing the importance of representative individual pixels.

Map	μ	σ	Max	Min	Sum	Non-null
S_I	0.0	0.17	+19.6	-9.1	+6.7	99.97%
$\sum S_k$	0.0	0.06	+2.7	-3.7	+66.5	99.90%
S_T	0.0	0.19	+20.2	-9.6	+73.2	99.97%

Table 1: Class 4 score map statistics for the analyzed image

6. Conclusions

We presented a new method for the explanation of deep learning classification models based on the distribution of the scores obtained in the last layer among the input pixels of the analyzed 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 DR interpretable classifier, able to classify retina images into the five standardized levels of disease severity and able also to report, for every class, score importance pixel maps, providing to ophtalmologists the possibility of inference and interpretation. The score generation is done back-propagating layer-wise only the score part that depends on the inputs and leaving the constant part as a contribution to the score of the considered layer. 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 output layer.

We concluded also that for a good understanding of the causes behind a classification not only the pixel-space maps should be considered but also the intermediate ones. Combination of the micro-information given by input space maps with the macro-information obtained from intermediate scores is the key for understanding network results and to aid improving diagnosis processes.

7. Future Work

Due to the lack of RD images with manually marked lesions, we are now working with ophtalmologists to create such kind of knowledge in order to later on validate empirically the output of the proposed method.

Acknowledgements

This work is supported by the URV grants 2017PFR-URV-B2-60, as well as, for the Spanish research projects PI15/01150, PI12/01535 (Instituto de Salud Carlos III) and Fondos Feder from EU. The authors would like to thank to the Kaggle and EyePACS for providing the data used in this paper.

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 Lettersdoi:https://doi.org/10.1016/j.patrec.2017.05.018.

[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.

[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.

[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.

[16] 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.

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

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

[19] 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.

[20] 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.

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