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

In this paper we present a diabetic retinopathy deep learning interpretable classifier. On one hand, it classifies retina images into different levels of severity with good performance. On the other hand, this classifier is able of explaining the classification results by assigning a score for each point in the hidden and input spaces. These scores indicate the pixel contribution to the final classification. To obtain these scores, we propose a new pixel-wise score propagation model that for every neuron, divides the observed output score into two components. With this method, the generated visual maps can be easily interpreted by an ophthalmologist in order to find the underlying statistical regularities that help to the diagnosis of this eye disease.

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

2010 MSC: 68T10

1. Introduction

Deep Learning (DL) methods [1], [2] have been extensively used in the last years for many automatic classification tasks. For the case of image classifica-

*Corresponding author

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

tion, the usual procedure consists on extracting the important features using a set of convolutional layers and, after that, make a final classification with these features using a set of fully connected layers. At the end, a soft-max output layer gives the predicted output probabilities of belonging to the classes predefined in the model. During training, model parameters are changed using a gradient-based optimization algorithm, which minimizes a predefined loss function.[3]

Once the classifier has been trained (i.e. model layer parameters have been fixed), the quality of the classification output 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 committee of human experts.

This mapping allows the classification of multi-dimensional 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. In addition, every connection has a parameter that corresponds to the weight of the relation.

The function of every neuron is to make a transformation of the received inputs into a calculated output value. For each incoming connection, a weight is multiplied by the input value and the aggregation over all inputs is fed to an activation function that calculates the neuron output. Parameters are usually optimized using a stochastic gradient descent algorithm that minimizes a predefined loss function. 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 [4], [5].

The first successful convolutional neural network (CNN) published [6] was designed for hand-written digit recognition. This early CNN implementation used a combination of convolution, pooling and non-linearity that has been the key feature of DL until now. It used convolutions for extracting spatial fea-

tures, sub-sampling for reducing maps size, a non-linearity in the form of tanh or sigmoids, and a fully connected multi-layer neural network as final classifier. Network parameters were all learned using an end-to-end optimization algorithm using stochastic gradient descent. One of the first successful implementations using GPUs was published in [7] where they successfully trained a neural network with 9 layers, also for handwritten digit recognition. The DL breakthrough took place with the publication of [8] where, for the first time, a CNN won the Imagenet[9] classification competition by a large margin. The network, named AlexNet, introduced a set of innovative techniques like data augmentation, the use of rectified linear units (ReLUs) as non-linearities, the use of dropout for avoiding overfitting, overlapping max-pooling avoiding the averaging effects of avg-pooling and the use of GPUs for speeding up the training time. This paper proved experimentally that CNNs were able to solve complex tasks. From the publication of this paper many improvements where published, like [10] with the introduction of Overfeat, a network derivation of AlexNet where they proposed learning bounding boxes, which later gave rise to many other papers on the same topic. In [11], VGG networks where presented. It was the first time that small 3x3 convolution filters where used and combined as a sequence of convolutions. The previous big filters of 9x9 and 11x11 present in AlexNet started to become smaller. The great advantage of VGG network was the insight of stacking 3x3 convolutions as a substitution of large filters. These ideas were used in the design of posterior networks like ResNet and Inception. GoogleNet[12] and Inception [13] were the first attempts to reduce the size of big networks. DL was becoming very useful for categorization of images and video frames, but its main concern was its low efficiency in size and computation. Inception module used a parallel calculation of 1x1, 3x3 and 5x5 convolutions significantly reducing the number of operations required by networks like AlexNet and VGG. Bottleneck layers (1x1 convolutions) where used before the calculation of bigger size convolutions for reducing the number of input filters, reducing in this way the computational costs without losing generality. So, 1x1 convolutions have been proven to achieve state-of-the-art results in Imagenet classification tasks.

The reason of this success is that input features are correlated, being removed by combining them appropriately with the 1x1 convolutions. Then, after convolution with a smaller number of features, they can be expanded again into a meaningful combination for the next layer. Another revolution came with the introduction of residual networks in [14]. These networks used a combination of 3x3 convolutional layers with a by-pass of the input every two layers that was summed up to the output. The introduction of such bypass improved the gradient propagation through the network allowing the use of deeper networks and improving the classification capabilities. In [15], a modified version of Inception networks called InceptionResNet was published to introduce this idea to such networks. Residual blocks allowed the reduction of training time, not improving significantly the classification performance.

DL models have been also successfully applied in many medical classification tasks. In [16] a DL classifier was designed achieving dermatologist-level accuracy for skin cancer detection. In [17] a 3D CNN for automated pulmonary nodule detection and classification was built. In [18] a CNN Alzheimer's disease classifier with high performance was also described. In [19] the authors presented a diabetic retinopathy classifier with better performance than human experts in the detection of the most severe cases of the disease.

In medical diagnosis tasks it is important not only the accuracy of predictions but also the reasons behind decisions. 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 [20] a network propagation technique is used for input-space feature visualization. After this work, [21] used a pixel-wise decomposition for classification problems. 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 [22] 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 an approach similar to pixel-wise decomposition taking into account the compositional nature of the topology, as in [20] and [21]. The concept of *score* in our paper is similar to the concept of *relevance* used in the layer-wise relevance propagation method. The novelty of our approach comes from the fact that we assume that there are two factors that contribute to output score: one is the input-space contribution, while the other is a contribution coming from the receptive fields of each layer. This last contribution depends solely on the parameters of each layer, thus being independent from the input-space. The second factor 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. Therefore, we only propagate back the score part that depends on the precedent input in each layer. In the proposed explanation model, we consider the constant part as a property of the RF of each layer. This approach allows us to do an exact propagation of the scores using a de-convolutional approach. Differing also from [20], 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 DL functional constituents is derived in the paper, in order to facilitate the portability of this new explanatory method 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 [23] 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 report the predicted class and also to score the importance of each 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, first, the current work on DL applied to DR is briefly introduced, then, the main works on interpretation of DL are presented. Section 3 presents the complete mathematical formulation of the proposed interpretable model, describing the score propagation model in detail. Section 4 explains the DL classification model for diabetic retinopathy. Section 5 presents the results showing a set of examples of the type of visual interpretations that are obtained with this new method. Finally, Section 6 gives the final conclusions of our work.

2. Related Work

Many deep learning classifiers for diabetic retinopathy have been published in the last years. In [24] a DL 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 [25] was close to the reported by human experts in the test set using a unique DL model without

ensembling.

In [19] a binary DL 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 ophthalmologists. This model surpassed the human expert capabilities, reaching approximately a performance of 97% in sensitivity and 93.5% in specificity in test sets of about 10,000 images. 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 DL 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 rationale behind the decisions, making difficult to the experts to have clues to improve the diagnostics.

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

2.1. Sensitivity maps

Sensitivity maps [26] 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 DL 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, which should not necessarily be in the same direction

that the global minimum [27].

2.2. Layer-wise relevance propagation

In [21] the authors split the 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, which has to be decomposable into several layers of computation (like a deep neural network), the second is about the fact that the total relevance must be preserved from one layer to another, which means that the relevance of one layer is equal to the ones of all other layers (eq. 1) and, finally, the relevance of each 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 $R_d^{(l)}$ is 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 $R_{i \leftarrow k}^{(l,l+1)}$ is the relevance message passing from node k located in layer $l+1$ to node i, located in layer l; a_i is the activation of node i and ω_{ij} is the weight connecting node i and j.

2.3. Taylor-type decomposition

Another way for solving the interpretability problem is using the classification function gradient to calculate the next Taylor approximation [21]:

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

Being I_0 a free parameter that should be chosen in a way that $f(I_0) = 0$ in the case that $f(I)$ is 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:

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

Equation 4 is per se an explanation of $f(I)$ dependent only of the derivative function and of I_0. The main concern of this approach is finding a valid root that is close to the analyzed image I using the Euclidean norm. In that way, 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 [22] uses an approximation that combines the layer-wise and Taylor type models. Taking advantage of the compositional nature of DL models, this approach assumes also the decomposability of the relevance function, considering 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 relevance scores until eventually reach the input-space. Then, it generates a heat-map of the total relevance of the prediction in the input image.

3. Receptive Field Score Distribution Model

In this section we describe an alternative method to distribute the output score into the previous layers, in DL models. We hypothesize that the output score of a particular classification model is not only due to pixel score contributions but also due to all the contributions of the receptive fields analyzed by the network. In next sections we will prove that, for all typical neural network types of layers, the output score 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 backwards the scores, but with an important difference: pixel-wise [21] propagates back not only the input layer contribution, but also constants,such as the bias, etc. Such a way of propagating backwards the scores breaks the linear transformation. A way of maintaining the linear transformation is propagating backwards only 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. Then, the 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 the contribution of each node to every following layer. Such node constant scores depend on layer parameters and, in some way, on the output activations of the 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 in the input-space, but also and, most 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.

Formally, 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 for $l \in \{1, ..., L\}$, being L the total number of layers.

Proposition 2. The score observed as output in one layer can be decomposed into two parts: one dependent on the score of the previous layer 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 all the different usual block elements of DL networks.

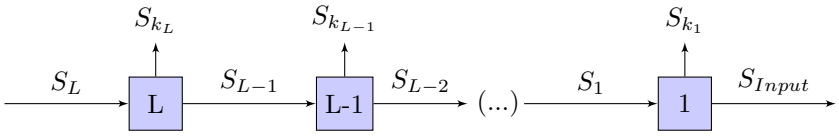


Figure 1: Score distribution through layers

The output score is expressed 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 the parts of the image 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 into 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:

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

Now, the output score 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, they are $S_o = S_i$ and $S_k = 0$.

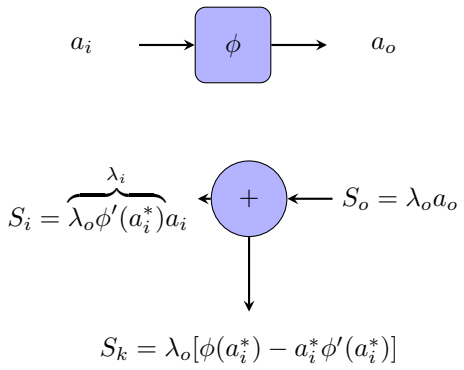


Figure 2: Score propagation through an activation function node

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 and into $S_i = (\lambda_o\frac{\gamma}{\sigma})a_i = \lambda_i S_i$ following Proposition 2, being $\lambda_i = \lambda_o\frac{\gamma}{\sigma}$ (see fig. 3)

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

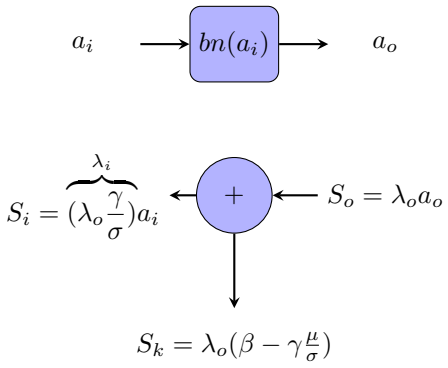


Figure 3: Score propagation through an batch normalization node

calculation of different outputs. This is the reason why every input receives a contribution of the score of the different outputs that are, then, summed up.

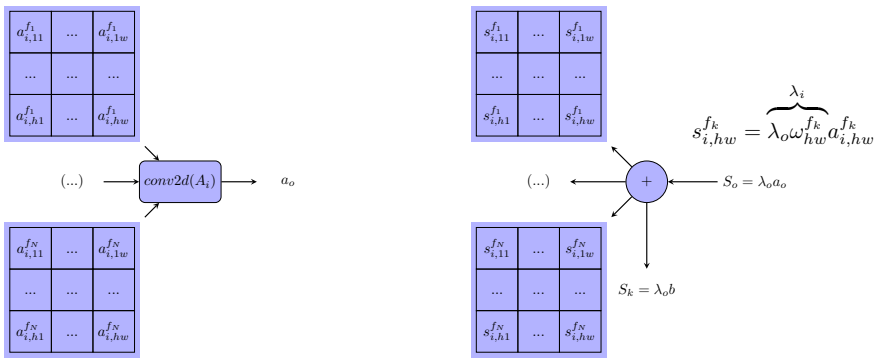
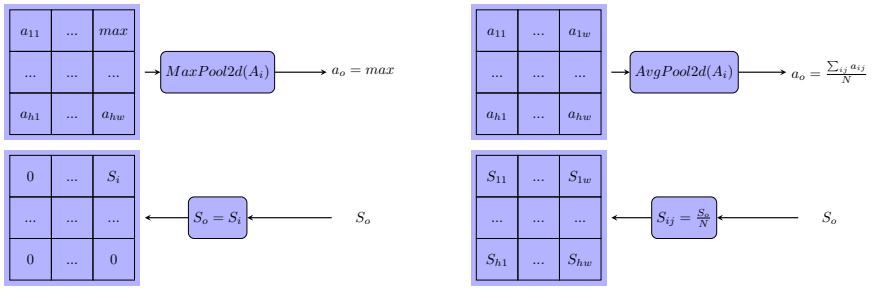


Figure 4: Convolution score calculation. Score spreads into the different inputs. The bias related part of the score is not backpropagated.

3.4. Score propagation through pooling layers

The score propagation through a max-pooling layer is straightforward. The score of output is copied into the input that was selected in the forward pass (see fig. 5). For average pooling, it 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 activations and their weights. The 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).

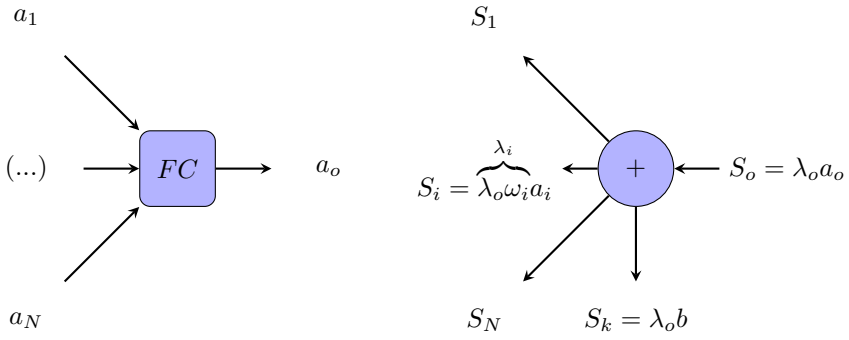


Figure 6: Score propagation through a fully connected node

3.6. Score propagation through a dropout layer

Dropout in evaluation time weights 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:

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

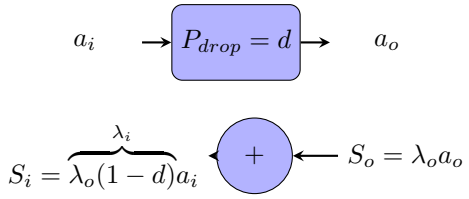


Figure 7: Score propagation through a dropout node

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 layer-dependent. At this point, we are going to describe a method to transport backwards also such last values into the input-space. From [28], 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, it is possible to make an approximate conversion of the full and constant scores in the hidden-space 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), it is possible to go back. Summing up 20x20 gaussian distributions of mean equal to the hidden-space values and summing up the coincident points, it is possible to map the distribution into input-space. We fixed $RF = 2\sigma$ as an approximate distribution of the scores that seems acceptable [28], since 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 [23]. The images are taken in variable conditions: by different cameras, illumination conditions and resolutions.

The dataset is split into 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 [19] but allows to train models near human expertise that are useful to show the purposes of our work, which is not only a good performance of the classification results but mainly to provide tools for interpretability of the final classification of each patient.

4.2. Construction of the classifier

The model calculates the probability $P(\mathcal{C}|\mathcal{I})$, being $\mathcal{C}$ one of the possible output classes 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 $argmax(S_i) = argmax(Softmax(S_i))$. Thus, we skip $Softmax$ from the interpretation analysis.

4.2.1. Design guidelines for DR classification

Up to now, the design of deep neural network models is mainly driven 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 [29] and [24]), we have tested different kinds of architectures for DR classification. Using the previous experience in such works we report here a set of guidelines that have been used to build the proposed classification model for diabetic retinopathy. These design principles for DR classification can be summarized into: use an optimal image resolution, use all the image information available, use a fully CNN, 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 the one hand, image input size has a great importance in the classification results. For this problem, other papers like [29] have shown that better results can be achieved with retina diameters of 512 pixels than with 384, 256 or 128 pixels. Some tests done using densities larger 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 is a retina diameter equal to 640 pixels. This is the one used for the results shown in this paper.

Use all the available image information. In previous studies published in [29], 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.

Use a fully convolutional neural network. CNNs 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. Spatial convolutions are very expensive in memory and time requirements, growing both quadratically with kernel size. Papers [11], [14], [13] proved experimentally that stacking small convolution layers and increasing depth is possible to obtain better results than using convolutions of bigger size with less depth. In [30] and [31] theoretical studies proved also that model expressiveness grows exponentially with depth. In our paper, we follow a similar approach to the one used in [11] using exclusively 3x3 convolutions in every feature layer. Even convolution sizes are discarded due to its asymmetry when used with zero padding.

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, for this reason sometimes can be necessary to slightly modify the

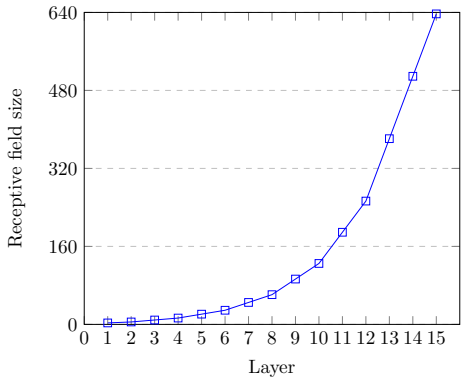


Figure 8: Model RF growth

convolution sizes of some layer to get the desired one. Figure 8 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 CNNs[32]. 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 [33] 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 [3]. In [24] it 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 [24]. 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.

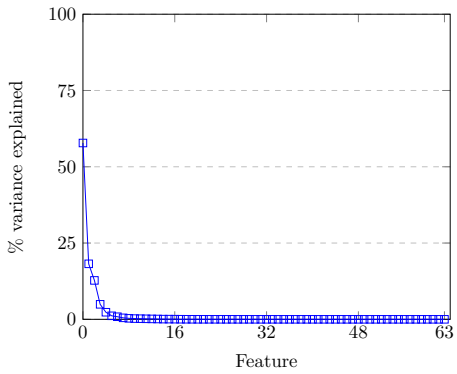


Figure 9: Feature space cummulative PCA variance of training set

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. In addition, we would like to be able to implement the result in devices with low resources too. Having this in mind, we tested networks of different sizes. In 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 using 64 features. Higher number of features did not improve the results. In figure 9 we show the variance explained by the 64 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. Figure 10

shows a block diagram of the model. It is a CNN with 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 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, the training set is artificially equalized using data augmentation techniques [34] based on $0 - 180°$ random rotation, X and Y mirroring and contrast&brightness random sampling.

A random initialization based in the Kaiming&He approach [35] is used. All models are optimized using a batch based first order optimization algorithm called Adam [36]. The loss function used for optimizing the model is the QWK-loss function, with a batch size of 15 and a learning rate of $3x10^{-4}$ [24]. 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.

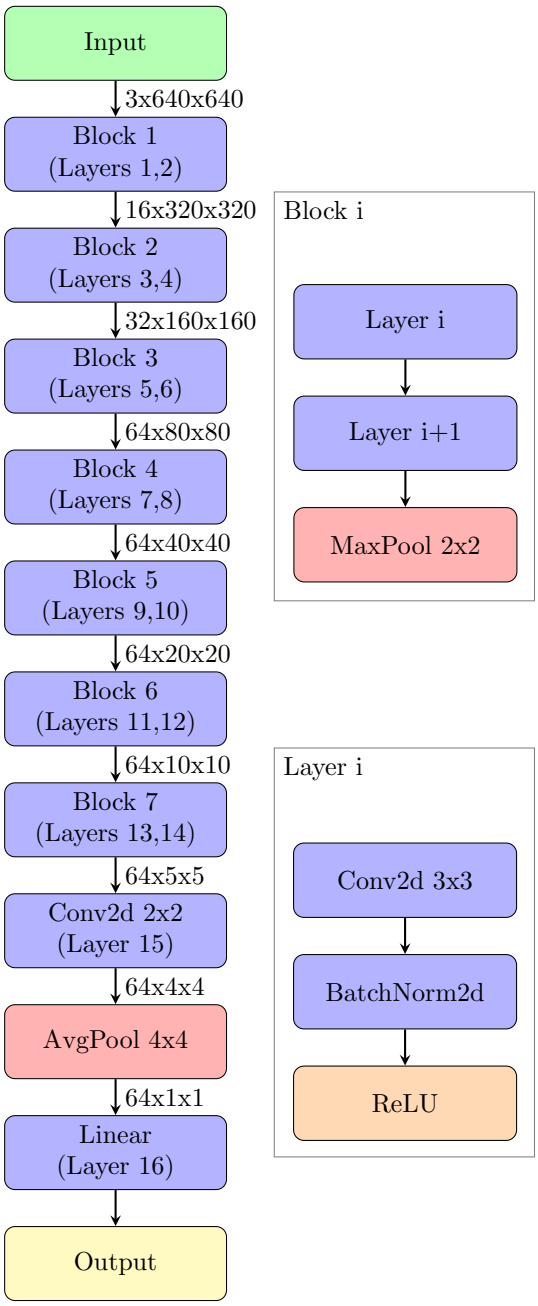


Figure 10: Prediction model of 391,325 trainable parameters located in blue blocks.

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 testing set (with images not used in the training) of 10,000 images is of 0.801. Using a linear classifier for combining the features of both eyes QWK_{test}, we can reach a QWK value of 0.844. Expert ophthalmologist report QWK inter-rating agreement values close to 0.80.

If we consider a binary classification with two categories, one for severe cases of DR (grouping classes 2, 3 and 4) and another for non-sever cases (classes 0 and 1), we can calculate the usual classification evaluation metrics. Over EyePACS test set we obtain the following values for different measures[1]: N=10,000, TP=1,727, TN=6,859, FP=1,235, FN=179, Sensitivity=0.906 (95% CI: 0.893 to 0.919), Specificity=0.847 (95% CI: 0.840-0.855), PPV=0.583, NPV=0.975, Accuracy=0.857, F_1=0.710 and MCC=0.648.

For comparison purposes with other works, we tested also our model against the standardised Messidor-2 dataset [37]. We achieve a QWK of 0.832 for this dataset. Binary classification statistics for prediction of the most severe cases of DR (grouping classes 2, 3 and macular edema) are[1]: N=1200, TP=465, TN=627, FP=61, FN=47, Sensitivity=0.908 (95% CI: 0.883 to 0.933), Specificity=0.911 (95% CI: 0.890-0.933), PPV=0.884, NPV=0.930, Accuracy=0.910, F_1=0.896 and MCC=0.817.

We can see that the results obtained with the Messidor-2 dataset are better because of the improved quality of images in the dataset and a also a better labelling of the examples.

In order to compare our model with state-of-the-art [19], table 1 show statistics for prediction of the most severe cases of DR (classes 2, 3 or macular edema) for Messidor-2 Dataset. We see that with a two orders of magnitude smaller

[1] Notation: N: sample size, TP: true positives, TN: true negatives, FP: false positives, FN: false negatives, CI: confidence interval, PPV: positive predictive value, NPV: negative predictive value, F_1: F1 score and MCC: Matthews correlation coefficient

model, it is possible to obtain good enough values of sensitivity and specificity. Our model is designed to be simple enough to be run is low resources devices, like mobile phones, with good performance. Furthermore, model simplicity eases the score calculations required for generating explanations.

Reference	Model Parameters	Model Depth	Sensitivity	Specificity
[19]	23,851,784	159	96.1 %	93.9 %
Our work	391,325	17	91.1 %	90.8 %

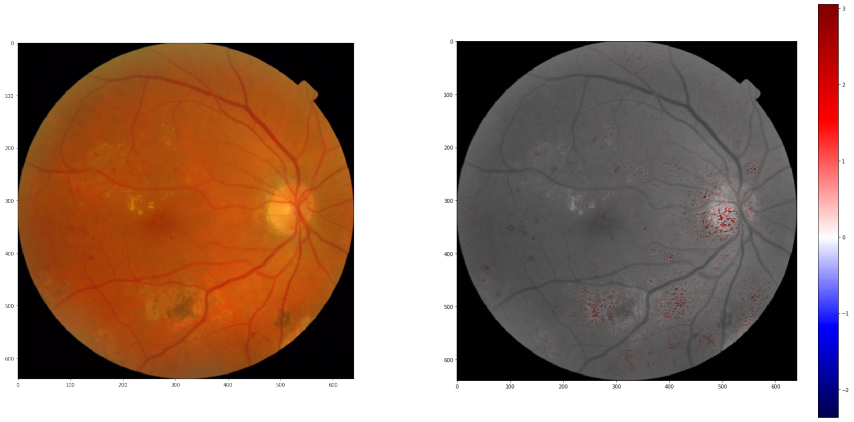
Table 1: Prediction performance & model complexity comparison between of our proposal and the state-of-the-art model (Messidor-2 data set)

Building a multi-class classification model needs to account for the encoding of required features for distinguishing between the different disease severity levels. This is is more difficult than a binary classifier, since it has to differentiate between more levels of DR severity. Thus, training the model for an aggregated detection (grouping the positive classes) increases the accuracy of the classifier, at the prize of missing the coding of important features that separate positive classes (1 to 4).

In our case, opthalmologists want to distinguish all levels of severity since the treatment can be then personalized to each patient with more detail. Therefore, our goal is to make the model learn such differences in order to visualize them in the explanation model. For this reason, it 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 that allow to separate the intermediate classes, even at the prize 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 get more knowledge about the image detection of DR signs.

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. 11 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. 11b 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.



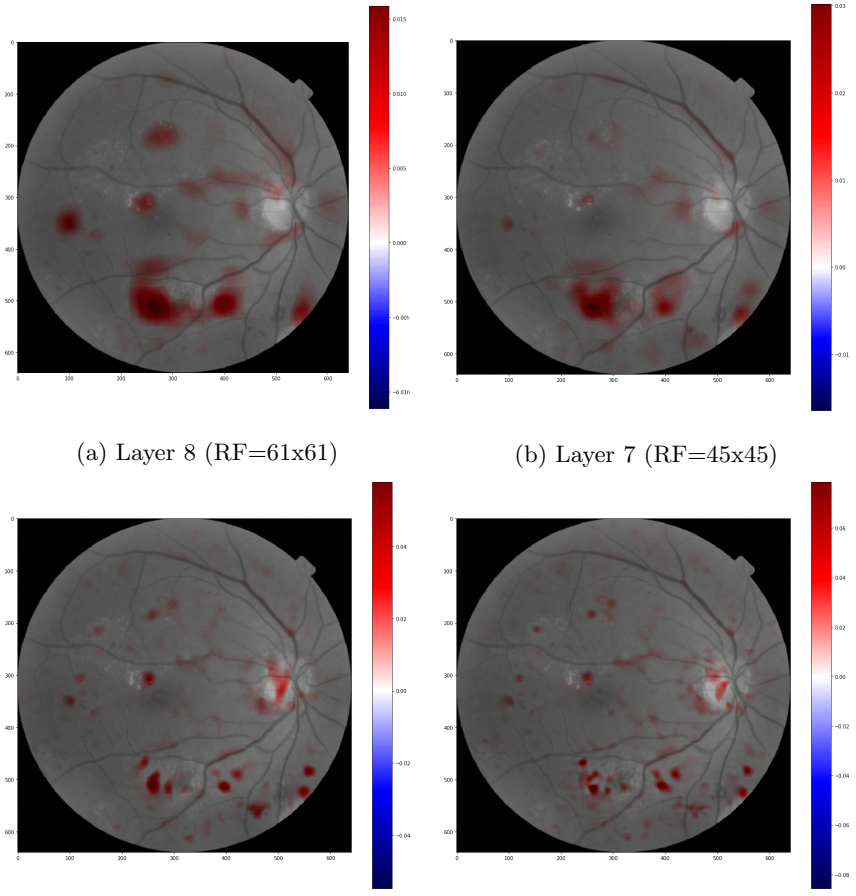
(a) Network input image

(b) Class 4 positive score map

Figure 11: Class 4 sample image

Figure 12 show the aggregated scores for different intermediate layers. 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 $O = max(0, \beta + \gamma(\frac{WI+b-\mu}{sigma})$, 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

$S_k = \lambda(\beta + \gamma\frac{b-\mu}{\sigma})$.



(a) Layer 8 (RF=61x61)

(b) Layer 7 (RF=45x45)

(c) Layer 6 (RF=29x29)

(d) Layer 5 (RF=21x21)

Figure 12: Some of the class 4 intermediate score maps generated for the sample image.

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.

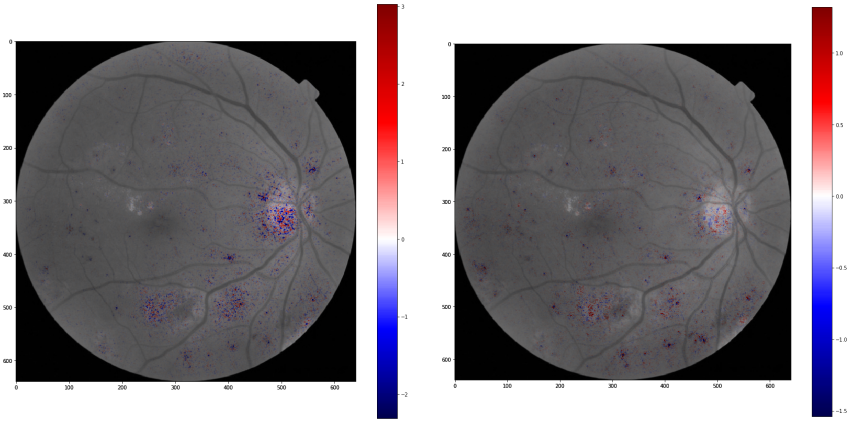
In red we have the zones that are contributing positively to classify the image as class 4. It is possible to plot also the zones contributing negatively to the classification. In case of being analyzing a class 4, negative score zones(not shown) would be zones without lesions. For clarity purposes only 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.

Intermediate scores are also very useful for identifying clusters of micro-lesions 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.

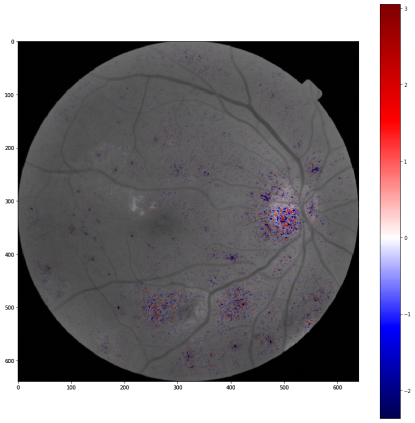
Map	μ	σ	Max	Min	Sum	Non-null
S_I	0.0	0.17	+19.6	-9.1	+6.7	99.97%
$\sum S_k$	0.0	0.06	+2.7	-3.7	+66.5	99.90%
S_T	0.0	0.19	+20.2	-9.6	+73.2	99.97%

Table 2: Class 4 score map statistics for the analyzed image

Finally, in figure 13 we present the total score map for the analyzed image,



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



(c) S_{total}

Figure 13: Total score maps: $S_{total} = S_I + \sum_{l=1}^{L} S_{k_l}$

plotting not only pixels with positive contribution (red) but also the negative ones (blue). The score input S_I (fig. 13a) represents the map obtained from back-propagating the part that linearly depends on inputs. The score constant $\sum S_k$ (fig. 13b) is the map obtained from summing up S_k for all layers mapped into input space. Score total S_T (fig. 13c) is the sum of the two previous maps and represent the total class 4 score distribution. The sum of all the scores of the pixels is equal to the output score. Table 2 shows some statistics of the three score maps. We can see that $\sum S_k$ 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. It seems that $\sum S_k$ 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.

6. Conclusions and future work

We presented a new method for the explanation of DL classification models based on the distribution of scores obtained in the last layer among the input pixels of the analyzed image. We formalized a general theoretical derivation of the score calculation for the more typical DL building blocks to make possible the generation of score propagation networks for any type of applications based on DL models.

Additionally, we applied the model to design a DR interpretable classifier, reaching more than 90% of sensitivity and specificity for the detection of the more severe cases of DR, not far from the state-of-the-art solution with two orders of magnitude less parameters. The designed model in not only able to classify retina images into the five standardized levels of disease severity, but also to report, for every class, score importance pixel maps, providing ophtalmologists the possibility of both inference and interpretation. The reduced number of model parameters enables the use of these algorithms even in low resources devices, like mobile phones. 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. In this way, this method is are able to generate scores in a unique and exact way.

Additionally, we have developed a technique, consisting on applying a 2D-gaussian prior over the RFs, for mapping the constant hidden-space scores to the input. With this technique we can generate 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. The combination of the micro-information given by the input space maps with the macro-information obtained from intermediate scores is the key for understanding the results and to help the medical personnel to improve the diagnosis processes.

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

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

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

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

[7] D. C. Ciresan, U. Meier, L. M. Gambardella, J. Schmidhuber, Deep big simple neural nets excel on handwritten digit recognition, 2010, Cited on 80.

[8] 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.

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, ImageNet: A Large-Scale Hierarchical Image Database, in: CVPR09, 2009.

[10] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, Y. Lecun, Overfeat: Integrated recognition, localization and detection using convolutional networks, in: International Conference on Learning Representations (ICLR2014), CBLS, April 2014, 2014.

[11] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, International Conference on Learning Representations, ICLR 2015.

[12] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Computer Vision and Pattern Recognition (CVPR), 2015.

[13] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2818–2826.

[14] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[15] C. Szegedy, S. Ioffe, V. Vanhoucke, A. A. Alemi, Inception-v4, inception-resnet and the impact of residual connections on learning., in: AAAI, Vol. 4, 2017, p. 12.

[16] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, S. Thrun, Dermatologist-level classification of skin cancer with deep neural networks, Nature 542 (7639) (2017) 115.

[17] W. Zhu, C. Liu, W. Fan, X. Xie, Deeplung: Deep 3d dual path nets for automated pulmonary nodule detection and classification, in: IEEE Winter Conference on Applications of Computer Vision, 2018.

[18] S.-H. Wang, P. Phillips, Y. Sui, B. Liu, M. Yang, H. Cheng, Classification of alzheimers disease based on eight-layer convolutional neural network with leaky rectified linear unit and max pooling, Journal of medical systems 42 (5) (2018) 85.

[19] V. Gulshan, L. Peng, M. Coram, Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs, Journal of the American Medical Association 316 (22) (2016) 2402–2410.

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

[21] 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.

[22] 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.

[23] 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.

[24] 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.

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

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

[27] 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.

[28] 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.

[29] 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.

[30] R. Eldan, O. Shamir, The power of depth for feedforward neural networks, in: Conference on Learning Theory, 2016, pp. 907–940.

[31] N. Cohen, O. Sharir, A. Shashua, On the expressive power of deep learning: A tensor analysis, in: Conference on Learning Theory, 2016, pp. 698–728.

[32] 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.

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

[34] 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.

[35] 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.

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

[37] E. Decencire, X. Zhang, G. Cazuguel, B. Lay, B. Cochener, C. Trone, P. Gain, R. Ordonez, P. Massin, A. Erginay, B. Charton, J.-C. Klein, Feedback on a publicly distributed database: the messidor database, Image Analysis & Stereology 33 (3) (2014) 231–234. doi:10.5566/ias.1155.
URL `http://www.ias-iss.org/ojs/IAS/article/view/1155`