Segmenting Retinal Blood Vessels with Deep Neural Networks

Paweł Liskowski, Krzysztof Krawiec, *Member, IEEE*

***Abstract*—The condition of the vascular network of human eye is an important diagnostic factor in ophthalmology. Its segmentation in fundus imaging is a nontrivial task due to variable size of vessels, relatively low contrast, and potential presence of pathologies like microaneurysms and hemorrhages. Many algorithms, both unsupervised and supervised, have been proposed for this purpose in the past. We propose a supervised segmentation technique that uses a deep neural network trained on a large (up to $400,000$) sample of examples preprocessed with global contrast normalization, zero-phase whitening, and augmented using geometric transformations and gamma corrections. Several variants of the method are considered, including structured prediction, where a network classifies multiple pixels simultaneously. When applied to standard benchmarks of fundus imaging, the DRIVE, STARE, and CHASE databases, the networks significantly outperform the previous algorithms on the area under ROC curve measure (up to > 0.99) and accuracy of classification (up to > 0.97). The method is also resistant to the phenomenon of central vessel reflex, sensitive in detection of fine vessels (sensitivity > 0.87), and fares well on pathological cases.**

***Index Terms*—Classification, deep learning, neural networks, structured prediction, feature learning, fundus, retina, vessel segmentation, retinopathy.**

I. Introduction

ARTIFICIAL Neural Networks (NNs) have an impressive record of applications in image analysis and interpretation, including medical imaging. Network architectures designed to work with image data, in particular convolutional networks (CNNs), were routinely built already in 1970's [1] and managed to find commercial applications [2] and surpass other approaches on challenging tasks like handwritten character recognition [3]. However, the break of millennia witnessed noticeable stagnation – despite intense research efforts, NNs failed to scale well with task complexity. The dawn of alternative ML techniques, like support vector machines and Bayesian networks, temporarily demoted the NNs, and it was only the relatively recent arrival of *deep learning* (DL) that brought them back into the spotlight. Today, large-scale DL-trained NNs successfully tackle generic object recognition tasks with thousands of object classes [4], a feat that many considered unthinkable just ten years ago.

The capabilities of deep NNs stem from several developments. Transfer functions of units in conventional NNs are typically squeezing functions (sigmoid or hyperbolic tangent), with derivatives close to zero almost everywhere. This leads to *diminishing gradient* in training – the backpropagated errors quickly decrease with each network layer, rendering training ineffective or painfully slow. In contrast, the transfer functions used in deep CNNs, most notably the rectifying linear units [5], do not vanish in extremes and so allow effective training of networks with dozens of layers [4]. Secondly, DL brought with it new techniques for boosting network robustness like *dropout* [6], where randomly selected units are temporarily disabled (Section IV-B). This forces a network to form weight configurations that provide correct outputs even if some image features are missing, and so improves generalization. Thirdly, more sophisticated DL methods train networks in a stage-wise (layered) fashion, promoting emergence of generic visual features [7]. Last but not least, these developments fortuitously coincided with the dawn of general-purpose computing on graphical processing units (GPUs), and NNs, by being an inherently distributed model of computing, excellently lend themselves to parallelization with GPUs.

In this paper, we propose a DL-based method for the problem of detecting blood vessels in fundus imagery, a medical imaging task that has significant diagnostic relevance and was subject to many studies in the past [8]. To our best knowledge, this is the first published result of this kind. The proposed approach outperforms previous methods on two major performance indicators, i.e., accuracy of classification and area under the ROC curve. We consider several network architectures and image preprocessing methods, verify the results on three image databases, perform cross-check between the databases, compare the results against the previous methods, and scrutinize the segmentations visually. Experimental results are presented in Sections VI and VII, preceded by discussion of medical problem and data (Section II), review of past work (Section III), and description of the proposed method (Section IV). An important methodological contribution is the involvement of the relatively novel approach of *structural prediction* (Section VII), which leads to the best overall results. In Section VIII, we perform robustness analysis by factoring the results into healthy and pathological cases, testing method's resistance to the phenomenon of central vessel reflex, and performing separate quantitative analysis for fine vessels.

II. Diagnostic problem and data

The condition of the human vascular system is an important diagnostic factor in a large number of medical conditions like atherosclerosis or diabetes. An organ that is particularly sensitive to vascular system pathologies is the eye. A malfunction of blood vessels in the retina has severe impact on the quality

The authors are with the Institute of Computing Science, Poznan University of Technology, Poland, e-mails: {pliskowski, kkrawiec}@cs.put.poznan.pl.

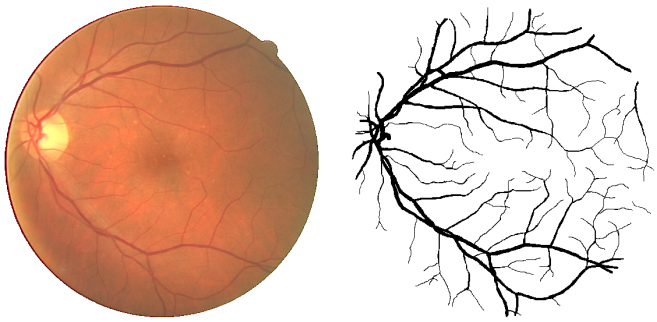


Figure 1: A training image from the DRIVE database (left) and the corresponding manual segmentation (right).

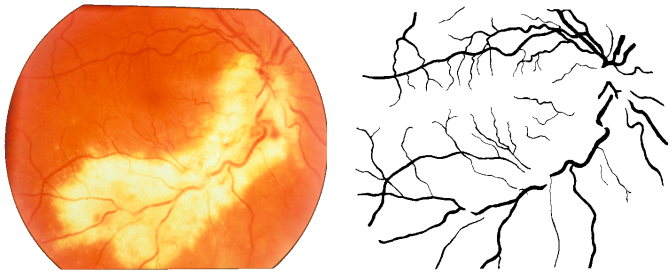


Figure 2: A pathological image from the STARE database (left) and the corresponding manual segmentation (right).

of vision. Contemporary, the most common cause of such anomalies is diabetes, which according to American Diabetes Association had 9.3 percent incidence in the US in 2012[1] and continues to rise. As a result, diabetic retinopathy affects over a quarter of adults with diabetes, and is currently the most common cause of blindness in the Western world.

There are several medical imaging techniques for assessing the state of retinal vascular system, including fundus imaging, fluoresceine angiography, and OCT (optical coherence tomography) angiography. In this study, we consider fundus images, i.e. pictures of the back of the eye taken in the visible band. Segmentation of blood vessels in this modality was subject to many past studies reviewed in Section III.

In pattern recognition terms, detection of blood vessels is a *segmentation task*, where the objective is to separate the structure of interest from the background. From the perspective of machine learning, it is a problem of binary classification: assign every image pixel to the positive (vessel) or negative (non-vessel, background) decision class. Following other studies, we assume that the decision on pixel's class can be made based on its neighborhood, defined as a square window (*patch*) centered on the pixel to be classified. The content of a patch is fed into a classifier, which is to return the decision on the central pixel. In experimental part, we rely mostly on two publicly available datasets, DRIVE and STARE, which according to review in [8] are the most popular in this domain. To further scrutinize robustness of the proposed method, we perform additional experiments with the CHASE database.

The **DRIVE database** [9][2] is the result of a screening study conducted in the Netherlands on 400 subjects aged 25–90. The database contains a sample of 40 subjects from that group, 7 of which show mild signs of early diabetic retinopathy, and the remaining ones represent the clinical norm. Each image is taken with a 3CCD camera, and the anatomical structures occupy the central circular area with a radius of approximately 540 pixels. The database is divided into the training set and the test set, both composed of 20 images taken from different patients. For every image, manual segmentation of retinal vessels is provided (we use the 'gold standard' one for the both sets). Figure 1 presents an exemplary training image and the corresponding manual segmentation.

The **STARE database** [10][3] consists of 20 retinal fundus slides captured by a TopCon TRV-50 fundus camera. The slides were then digitized to produce 605×700 images. First half of the dataset comprises images of healthy subjects, while the other half pathological cases with abnormalities that overlap with blood vessels, in some cases obscuring them completely. The presence of pathologies makes the segmentation more challenging and allows performance evaluation in more realistic conditions. The database contains two sets of manual segmentations prepared by two observers, and the former one is considered as the ground truth. Figure 2 presents a pathological image and the corresponding manual segmentation.

The **CHASE** database (CHASE_DB1 in [11][4]) is a subset of retinal images of multiethnic children from the Child Heart and Health Study in England and comprises 28 images with a resolution of 1280×960 pixels. Compared to DRIVE and STARE, CHASE images have non-uniform background illumination, poor contrast of blood vessels and wider arteriolars that have a bright strip running down the centre, known as the *central vessel reflex*.

Tables IV and IX summarize best-to-date results achieved on all three considered databases.

III. Related work

Blood vessel segmentation in fundus imaging was subject to numerous studies in the past – the extensive review published in 2012 cites 73 works that used machine learning, matched filtering, morphological processing, vessel tracking, multiscale and model-based approaches [8]. Here, we briefly review the most relevant and best-performing approaches from that study and the methods published later. In Table IV, we provide their accuracy of classification (Acc) and area under ROC curve (AUC), the two most common performance measures, reported in almost all studies.

Segmentation methods can be divided into *unsupervised* and *supervised*. In the former ones, the properties of structures to be detected are manually hard-coded into algorithm's structure, and learning is limited or altogether absent. A representative of this group is the first of two methods proposed in [12], which uses line detectors to accumulate the evidence for the presence

[1] http://www.diabetes.org/diabetes-basics/statistics/
[2] http://www.isi.uu.nl/Research/Databases/DRIVE/
[3] http://www.ces.clemson.edu/~ahoover/stare/
[4] https://blogs.kingston.ac.uk/retinal/chasedb1/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMI.2016.2546227, IEEE Transactions on Medical Imaging

of blood vessels at individual image pixels. Another example is [13], where a co-occurrence matrix is calculated from an image patch, and decision is made by thresholding a feature calculated from that matrix. In [14], vessels are detected using a combination of co-linearly aligned difference-of-Gaussian filters, configured according to prototype patterns (e.g., straight vessel, bifurcation, crossover point) provided by a user. Zhao *et al*. in [15] employ a sophisticated active contour model that involves not only pixel brightness, but also other features derived from an image.

In supervised methods, segmentation algorithms acquire the necessary knowledge by learning from image patches annotated by ground truth. For instance in [9], ridge features (zero-crossings of brightness derivative) are detected, grouped, and fed into a nearest-neighbor classifier. In [16], a multiscale Gabor transform (wavelets) is used to extract features from a patch, and train Bayesian classifiers on them. In [17], a morphology-based method detects the evident vessels and background areas, while a Gaussian mixture model classifies the more difficult pixels.

Supervised methods are usually less labor-intensive and often perform better than the unsupervised ones. However, the boundary between these two categories is sometimes hard to determine. For instance, though the method described in [9] ultimately engages learning, it relies on multiple (a dozen or more) features based on domain knowledge. Also, almost every unsupervised method can be extended with learning. For instance, the second method presented in [12] augments the unsupervised line detection technique by training an SVM classifier on manually designed features collected from the vicinity of detected line. The method attains noticeably better performance indicators than the original, non-adaptive technique.

Several past studies engaged neural networks (see [8] for the complete list). Among them, [18] attained particularly good AUC. However, the method proposed there is not purely supervised in the strict meaning of this term, as the network learns from features defined by a human expert. According to [8], the only purely neural and thus fully supervised approach (i.e., with a network directly applied to image patches) was presented in [19], which however contained only visual inspection of the resulting segmentations and did not report objective quality measures.

In the context of the past work, the method proposed in this paper should be classified as supervised: no prior domain knowledge on vessel structure was used for its design.

IV. The method

In this section, we describe the core components of the proposed approach: convolutional neural networks and the deep learning algorithms.

A. Deep neural networks

A convolutional neural network (CNNs) is a composite of multiple elementary processing units, each featuring several weighted inputs and one output, performing convolution of input signals with weights and transforming the outcome with some form of nonlinearity. The units are arranged in rectangular *layers* (grids), and their locations in a layer correspond to pixels in an input image (Fig. 3). The spatial arrangement of units is the primary characteristics that makes CNNs suitable for processing visual information; the other features are *local connectivity*, *parameter sharing* and *pooling* of hidden units.

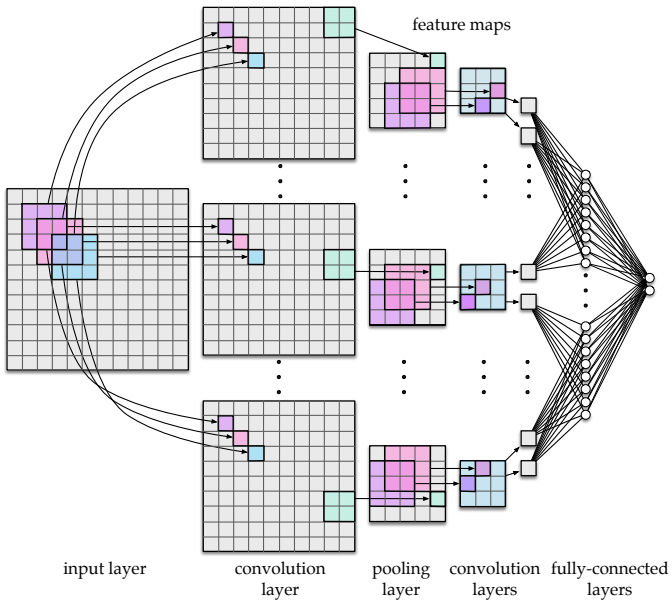


Figure 3: Architecture of a convolutional neural network with three convolutional layers, one pooling layer, and two fully-connected layers. The network uses 3×3 convolution units with stride 1 and 2×2 pooling units with stride 2.

Local connectivity means that a given unit receives data only from its *receptive field* (RF), a small rectangle of image pixels (for the units in the first layer) or units in the previous layer (for the subsequent layers) The RFs of neighboring units in a layer typically offset by *stride*. Image size, RF size and stride together determine the dimensions of a layer. For instance, a layer of units with 3×3 RFs with one pixel stride needs only 9 units when applied to a 5×5 monochromatic (single-channel) image, because three RFs of width 3 each, overlapping by two pixels, span the entire image. Larger stride and larger RFs lead to smaller layers (Fig. 3). Local connectivity substantially reduces the number of weights in comparison to the fully-connected conventional networks. It is also consistent with the spatial nature of visual information and mimics certain aspects of natural visual systems [1].

Parameter sharing consists in sharing weights across units in the same layer. When the units in a given layer share the same vector of weights, they form a *feature map*, with each of them calculating the same local feature, albeit from a different part of the image. This reduces the number of parameters even further and makes the extracted features equivariant. For instance, a layer of units with 3×3 RFs connected to a single-channel image has only 10 parameters (nine per channel for the pixels in the RF and one for neuron threshold), *regardless the number of units*.

Pooling (*subsampling*) consists in aggregating outputs of multiple units by other means than convolution. In the most

popular aggregation scheme, *max-pooling,* each aggregating unit returns the maximum value in its RF. Like local connectivity, pooling reduces the resolution w.r.t. previous layer and provides for translational invariance.

A typical CNN architecture consists of several convolutional feature maps intertwined with max-pooling layers, finalized with at least one fully-connected layer that 'funnels' the excitations into output neurons, each corresponding to one decision class (Fig. 3). Sliding RFs by the number of pixels defined in stride across the input image causes consecutive layers to be smaller, so that the final grid fed into the fully-connected is usually much smaller than the input image. There are often several feature maps working in parallel, each of them responsible for extracting one feature (see convolution and pooling layers in Fig. 3). Large networks may involve even several dozens of feature maps [4]. In case of multi-channel (e.g., RGB) images, separate feature maps are connected to all channels. The subsequent layers fetch data from multiple maps in the previous layer and so combine the information from particular channels. In such a case, each unit has multiple RFs with separate weight vectors, and the weighted signals coming from all input maps together form its excitation.

B. Network training

Training consists in an iterative propagation of *examples* (images) through a network and modification of its weights, which are initialized with small signed random values. The cycle of presenting all training examples, an *epoch*, is split into smaller units called *batches* (256 image patches in our approach). The backpropagated errors committed by particular units are accumulated and translated into weight updates when a batch is complete. DL offers a broad spectrum of training methods, of which we employ here the error backpropagation algorithm extended with *dropout* [6]. While the algorithm is described in every NN textbook, dropout is relatively new and less common so we describe it in the following.

With dropout, a subset of network units (typically 50 percent) is drawn at random and temporally 'switched off' during training. Those units do not propagate signals when an example is presented (their outputs are forced to zero), nor participate in the process of error backpropagation. The subset of disabled units is drawn independently for each batch, resulting in effectively a different network architecture. A network of m units can be 'trimmed down' in this way to any of its 2^m subnetworks. This creates an additional challenge for the training process, which strives to achieve the desired functionality with each such 'handicapped' subnetwork. To that aim, a network must maintain multiple alternative dataflows that support decisions on particular examples. This makes it more robust and increases generalization ability [20].

For a more comprehensible description of deep learning and deep CNNs, we recommend [21], [22].

V. Data preparation

The raw fundus images are subject to preprocessing comprising extraction of examples (Section V-A), image processing (Section V-B), and optionally augmentation (Section V-C).

Table I: Database statistics.

	DRIVE		STARE	
Validation scheme	One-off train + test		Leave-one-out	
Image dimensions	565×584		605×700	
	Training	Testing	Training	Testing
Fundus images	20	20	19	1
Total patches	3,857,818	3,993,351	5,326,283	280,330
- positive (vessel)	516,152	538,475	558,261	29,382
- negative	3,341,666	3,454,876	4,768,022	250,948

A. Extraction of patches

The ground-truth data provided in the manual segmentations (Figs. 1 and 2) frames the blood vessel detection as a binary classification problem. As in many other studies, in our approach the decision on the class of a particular pixel is based on an $m \times m$ *patch* centered at that pixel. A triple of such patches, each reflecting image content at the same location for the RGB channels, forms the input fed into a neural network. Together with the associated class label of the central pixel, it forms an example. In this study, we use $m = 27$, so an example is a vector of length $3 \times 27 \times 27 = 2187$. Figure 4 shows examples of positive and negative patches.

For the DRIVE database, we conform to the division of raw fundus images into training and test part (Section II). No analogous division was originally provided for the STARE database, so we perform there the *leave-one-out* validation: the training and testing cycle is repeated 20 times, and in each iteration 19 of the 20 fundus images form the source of training examples, and the remaining image is the source of test patches. From each training fundus image, we sample 20,000 training patches at random, so that the training set always comprises 400,000 patches for DRIVE and 380,000 patches for STARE. For testing, we extract all patches from each test image, in order to make performance assessment as accurate as possible. The resulting sizes of training and test sets, together with the proportions of decision classes, are summarized in Table I.

We consider only patches that completely fit in the circular active area in fundus images, called *field of view* (FOV) in the following (Fig. 1). For DRIVE, we use the original FOVs provided in the database. In STARE, FOVs are not provided, so we obtain them by thresholding fundus images on the green channel, followed by morphological opening with a 3×3 structural element.

B. Image preprocessing

Deep learning architectures can effectively learn from raw image data. However, they tend to perform better on appropriately preprocessed images. In the following, we describe the preprocessing applied in this study.

1) Global Contrast Normalization (GCN): The images in Figs. 1 and 2 clearly indicate that brightness may vary across the FOV. In order to help the learning process to abstract from these fluctuations and focus on vessel detection, we perform local (per-patch) brightness and contrast normalization. Every patch is standardized by subtracting the mean and dividing by the standard deviation of its elements (independently in

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMI.2016.2546227, IEEE Transactions on Medical Imaging

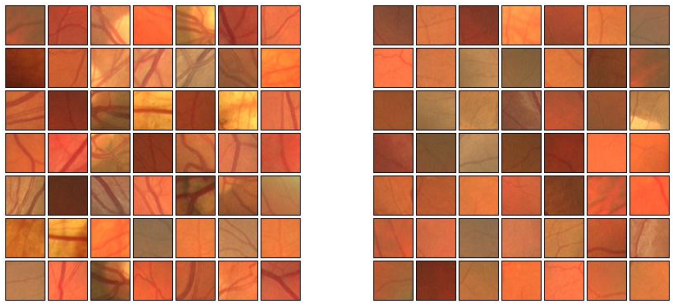


Figure 4: Examples of positive (left) and negative (right) 27×27 training patches extracted from the DRIVE images.

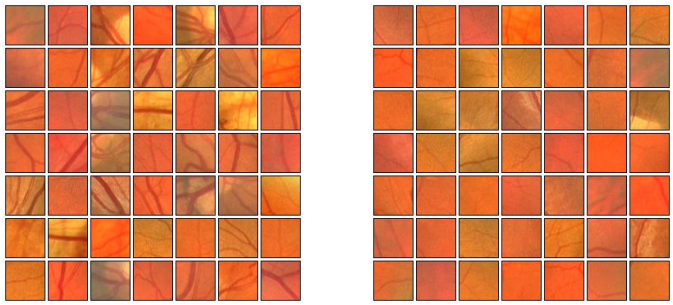


Figure 5: Examples of positive (left) and negative (right) training patches after applying GCN transformation.

the R, G, and B channel), which also maps the originally byte-encoded brightness to signed reals. Figure 5 presents the outcome of this processing for the patches from Fig. 4.

2) Zero-phase Component Analysis (ZCA Whitening): In natural images, neighboring pixels are strongly correlated as they are likely to represent the same structure in a scene. When learning a statistical model of images (which is what a neural network is essentially doing), it is thus desirable to abstract from these universal characteristics and focus on the higher-order correlations, i.e., capture the relevant regularities rather than the quite obvious fact that nearby pixels are likely to be similar [23]. Removal of universal correlations can be achieved by multiplying the data matrix by a *whitening matrix* [24]. After this common in DL transformation, it becomes impossible to predict the value of one pixel given the value of only one other pixel.

Formally, let the centered (zero mean) data be stored in matrix X with features in columns and data points in rows. In our case, X has $3 \times 27 \times 27 = 2187$ columns and the number of rows equal to the number of examples (patches). Let the covariance matrix $\Sigma = \frac{1}{m} \sum_{i=1}^{m} XX^T$ have eigenvectors in columns of U and eigenvalues on the diagonal of Λ, so that $\Sigma = U \Lambda U^T$. Then, U is an orthogonal rotation matrix, and U^T gives a rotation needed to decorrelate the data (i.e., it maps the original features onto the principal components). The classical PCA whitening transformation is given by:

$$W_{PCA} = \Lambda^{-1/2} U^T.$$

Each component of W_{PCA} has variance given by the corresponding eigenvalue.

However, whitening is not unique:, the whitened data remains whitened under rotation, which means that any $W = RW_{PCA}$ with an orthogonal matrix R will also be a whitening transformation. In what is called *zero-phase component analysis* (ZCA) whitening [25], we take U as this orthogonal matrix, i.e.:

$$W_{ZCA} = U \Lambda^{-1/2} U^T = \Sigma^{-1/2}.$$

The defining property of ZCA transformation (also called *Mahalanobis transformation*) is that it results in whitened data that is as close as possible to the original data (in the least squares sense). In other words, if $||X - XY^T||^2$ is to be minimized subject to XY^T being whitened, then the solution is to set $Y = W_{ZCA}$.

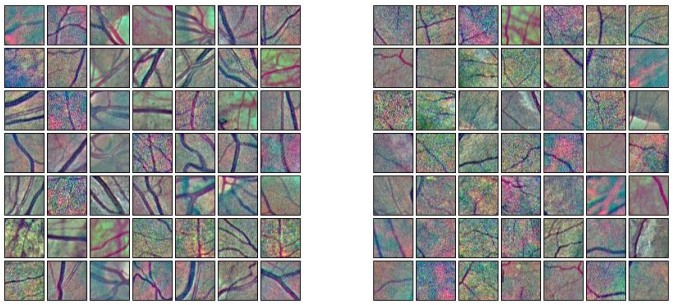


Figure 6: Positive (left) and negative (right) training patches after applying ZCA whitening transformation.

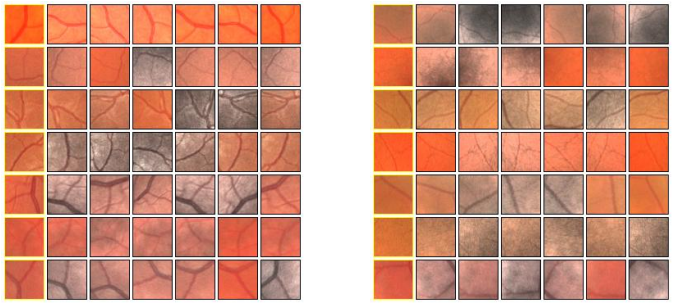


Figure 7: Augmentations of positive (left) and negative (right) training patches. Each row shows 6 random augmentations of the leftmost patch.

Figure 6 presents the patches from Fig. 5 whitened by the ZCA transform. In experiments, we first apply GCN to individual patches in the training set, then learn the ZCA transformation from these patches, and finally apply it to both training and test sets.

C. Augmentations

Despite using large numbers (Table I) of relatively small (27×27) patches for training, the CNNs may overfit. Other works on deep learning [26] show that the reason for that is the still relatively small number of examples compared to the number of model (CNN's) parameters, which for the network in Section VI is in the order of 48 millions. The remedy commonly used in deep learning is augmentation, meant as generation of additional examples by (typically randomized) transformations of existing training examples.

The data was augmented offline, prior to learning. Each patch, normalized and whitened as per Sections V-B and V-B2,

Table II: Architecture of the evaluated CNNs. Layer names are followed by numbers of feature maps. Square brackets specify RF size, stride and padding.

PLAIN	$conv64_{[4\times4\times1\times0]} \rightarrow conv64_{[3\times3\times1\times1]} \rightarrow maxpool_{[2\times2\times2\times0]} \rightarrow conv128_{[3\times3\times1\times1]} \rightarrow conv128_{[3\times3\times1\times1]} \rightarrow maxpool_{[2\times2\times2\times0]} \rightarrow fc512 \rightarrow fc512 \rightarrow fc2$
NO-POOL	$conv64_{[3\times3\times1\times1]} \rightarrow conv64_{[3\times3\times1\times1]} \rightarrow conv128_{[3\times3\times1\times1]} \rightarrow conv128_{[3\times3\times1\times1]} \rightarrow fc512 \rightarrow fc512 \rightarrow fc2$

was subject to 10 independent transformations, each composed of four randomized actions from the following list:

- Scaling by a factor between 0.7 and 1.2,
- Rotation by an angle from $[-90, 90]$,
- Flipping horizontally or vertically,
- Gamma correction of Saturation and Value (of the HSV colorspace) by raising pixels to a power in $[0.25, 4]$.

We tuned the ranges of parameters in a preliminary experiment. The parameters are drawn from them uniformly. We apply augmentations to image fragments of size $2m \times 2m = 49\times49$ to ensure scaling and rotation can be performed without artifacts. Once the above transformations have been applied, any redundant pixels are discarded so that the resulting patches are of size $m \times m = 27 \times 27$, i.e., same as the original ones.

Each resulting patch receives the same class label as the original patch. In this way, we increase the number of training patches (Table I) by the factor of 11. This preprocessing does not affect the testing patches.

VI. THE EXPERIMENT

A. Network architectures and training parameters

We considered a range of network architectures, of which only two shown in Table II are used in the following. In both architectures, the input image is passed through a stack of convolutional layers with units equipped with 3×3 RFs [29], with stride set to 1. Whenever there is need to preserve spatial resolution after convolution, we pad the input with zeros on grid border (so that the RF can virtually reach beyond the input grid). The architectures differ only in spatial pooling, which is present only in PLAIN in the form of three max-pooling layers (not all the convolutional layers are followed by max-pooling). Max-pooling is performed over a 2×2 pixel window, with stride 2. NO-POOL abandons max-pooling, as it has been recently shown that networks with convolutional layers only might perform better when applied to small images [30].

Apart from these two basic configurations, we consider four others, all using the architecture of the PLAIN network: GCN, ZCA AUGMENT, and BALANCED (Table III). The former three train PLAIN architecture on the data preprocessed by the methods described in Section V-B. In BALANCED, the PLAIN architecture is trained on balanced data, i.e., with equal proportions of decision classes. All architectures except for BALANCED learn from the same training sets specified in Table I. In preliminary experiments, we combined the configurations (e.g., using GCN with decision class balancing); that however did not lead to further performance improvements.

In all architectures, a stack of convolutional layers is followed by three fully-connected (FC) layers: the first two have 512 units each, the third performs 2-way classification and thus contains 2 units (one for each class). The configuration of the fully connected layers is the same in both architectures. The output units are sigmoidal; all hidden layers are equipped with the rectification non-linearity (ReLU, [31]), i.e., unit's output is $\max(0, e)$, where e is the outcome of convolution. Thanks to their non-saturating form, ReLUs greatly accelerate (e.g., six-fold in [4]) the convergence of stochastic gradient descent compared to the squeezing nonlinearities (sigmoid and $\tanh$). Also, ReLUs can be implemented by simply thresholding a matrix of activations at zero and so execute much faster. Weights in convolutional layers are set using Xavier initialization [32], while those in the FC layers with values sampled from the $\mathcal{N}(0, 0.01)$ Gaussian distribution. Units' biases (offsets) were initially set to zero.

Training is carried out by stochastic gradient descent with batch updates and momentum, which is equivalent to optimizing the multinomial logistic regression objective. Single iteration of learning involves presentation of an image patch to the network, propagating excitations through the network up to the output layer, calculating errors committed by the neurons in the output layer, propagating those errors backward through the network (resulting in an error value for every neuron in all hidden layers), calculating weight corrections from those errors, and accumulating them in for each weight individually. Weight update proceeds in batches, i.e., every 256 iterations the accumulated weight corrections are applied to the weights. That update involves also the momentum term of 0.9, i.e., the applied update is the sum of the current accumulated correction and 90 percent of the previous correction. The training is regularized by weight decay (L_2 penalty multiplier set to $5 \cdot 10^{-4}$) and dropout regularization [6] for the first two FC layers (dropout ratio set to 0.5). The learning rate was initially set to 10^{-3}, and then decreased by a factor of 10 every 10,000 iterations (batches). In total, the learning rate was decreased 3 times, and the learning was stopped after 30,000 iterations (19 epochs). The implementation was based on the Caffe framework [33], which performs all computation on GPUs in single-precision arithmetic. The experiments were conducted on two hardware configurations: Intel Core i7-4770K CPU with four NVIDIA GTX Titan graphics cards, and Intel Core i7-3770K CPU with NVIDIA Tesla K20c card.

B. Results

In Table III, we present the performance of networks tested on the benchmark-specific test sets in terms of area under ROC curve AUC, accuracy Acc, sensitivity $Sens$, specificity $Spec$, Cohen's Kappa, defined as

$$Acc = \frac{TP+TN}{TP+TN+FP+FN}, \; Sens = \frac{TP}{TP+FN},$$
$$Spec = \frac{TN}{TN+FP}, \; Kappa = \frac{Acc-Acc_r}{1-Acc_r},$$

where TP, TN, FP, and FN are respectively the numbers of true positive, true negative, false positive, and false negative

Table III: Performance of models (point estimates for DRIVE, averages with .95 confidence intervals for STARE).

	DRIVE						STARE					
	AUC	Acc	Acc*	Kappa	Sens	Spec	AUC	Acc	Acc*	Kappa	Sens	Spec
PLAIN	.9683	.9479	.9473	.7653	.7417	.9804	.9767±.0053	.9559±.0071	.9551±.0072	.7477±.0451	.7495±.0721	.9788±.0081
GCN	.9708	.9487	.9475	.7708	.7550	.9792	.9787±.0049	.9571±.0064	.9572±.0064	.7573±.0394	.7620±.0656	.9789±.0072
ZCA	.9719	.9485	.9472	.7756	.7819	.9748	.9783±.0062	.9563±.0064	.9562±.0066	.7598±.0317	.7718±.0490	.9783±.0055
AUGMENT	.9663	.9466	.9453	.7610	.7447	.9784	.9744±.0048	.9527±.0068	.9512±.0069	.7306±.0431	.7376±.0720	.9769±.0086
BALANCED	.9738	.9230	.9251	.7193	.9160	.9241	.9820±.0045	.9309±.0107	.9620±.0051	.7021±.0305	.9307±.0274	.9304±.0133
NO-POOL	.9720	.9495	.9486	.7781	.7763	.9768	.9785±.0066	.9566±.0082	.9568±.0081	.7622±.0415	.7867±.0698	.9754±.0099

decisions. Acc_r is the accuracy of classification of a classifier that makes random decisions according to the distribution of decision classes observed in the training data. By relying on it, Kappa takes into account the proportions of decision classes, which are particularly imbalanced here (Table I). AUC is calculated using the implementation provided in the *scikit-learn* Python library and modeled using a series of trapezoids built on all points that form the ROC curve.

The four performance indicators are based on the default interpretation of network decisions: positive decision (vessel) is made when the output of the (sigmoidal) unit associated with the positive class in the FC output layer is greater than the 0.5 threshold; otherwise, negative decision is made (non-vessel). However, this arbitrary threshold does not necessarily maximize the accuracy of classification. Thus, following the common practice[5], we introduce an alternative decision rule where the threshold is selected from the interval $[0, 1]$ so as to maximize the accuracy on the training data; this threshold is then used when classifying the test examples. The respective accuracy is marked as Acc^*. For the training data it always holds that $Acc^* \geq Acc$; however, this is not guaranteed for the test set, where other threshold may be optimal.

The left part of Table III presents only point estimates, as DRIVE has a fixed division into training and test set, and therefore only one network was trained for each configuration. For STARE, we report both averages and 0.95 confidence intervals over 20 folds of leave-one-out.

For reference, in Table IV we provide Acc and AUC values for the best-to-date methods reviewed in Section III. The trained networks consistently achieve very high values of all indicators, in some configurations outperforming all previous algorithms . The scores are particularly high on AUC, the arguably most important indicator that abstracts from the proportions of decision classes and mitigates the sensitivity-selectivity trade-off by characterizing the entire ROC curve with a single number. The most promising configuration is BALANCED: it does not only achieve the best-to-date AUC on both DRIVE (.9738) and STARE ($.9820 \pm .0045$), but also the best-to-date Acc^* ($.9620 \pm .0051$) on the latter dataset. The differences on STARE are statistically significant – the best-to-date values are outside the confidence intervals of the results of BALANCED. Though BALANCED underperforms on DRIVE with respect to both Acc and Acc^*, we attribute that to the possible bias in the one-off division of DRIVE data into training and test set. The multiple leave-one-out evaluation scheme used in STARE reduces that bias, so the results obtained there should be considered as more reliable.

The other well-performing architecture is NO-POOL, which we attribute to low dimensionality of input images (compared to much larger images often used in DL). In the remaining configurations, pooling in consecutive layers quickly reduces the already quite scarce information contained in the arguably small 27×27 patch, making classification more difficult. By discarding pooling layers, we maintain the dimensionality of patches across the entire network architecture, allowing convolutional filters in successive layers to be applied at the same resolution.

Apart from AUC and Acc, the values of the remaining indicators are also encouraging; however, sensitivity and specificity are tied with an inherent trade-off controlled by the threshold mentioned earlier, and therefore are hard to compare individually.

Generalization performance of the proposed approach was also verified in a more demanding scenario of training a network on one dataset and testing it on another. While many machine learning methods work well only assuming that the training and test data are drawn from the same distribution, our networks prove to be capable of *knowledge transfer*, i.e., the networks trained on DRIVE can act as effective classifiers on STARE and vice-versa. Table V shows that the NO-POOL configuration in such scenarios diverges by no more than 1–2 percent compared to the intra-dataset performances in Table III. Also the network trained on the combined DRIVE+STARE

[5]http://www.isi.uu.nl/Research/Databases/DRIVE/results.php

Table IV: Best-to-date results on DRIVE and STARE database. Bold font marks the best overall result according to [8].

	DRIVE		STARE	
Method	AUC	Acc	AUC	Acc
2nd. observer [10]	–	.9473	–	.9349
Jiang et al. [27]	.9327	.8911	.9298	.9009
Staal et al. [9]	.9520	.9441	.9614	.9516
Soares et al. [16]	.9614	.9466	.9671	.9480
Ricci et al. [12]	.9558	.9563	.9602	**.9584**
Villalobos-Castaldi et al. [13]*	–	**.9759**	–	–
Osareh et al. [28]	**.9650**	–	–	–
Marin et al. [18]	.9588	.9452	**.9769**	.9526
Roychowdhury et al. [17]	.9620	.9519	.9688	.9515
Azzopardi et al. [14]	.9614	.9442	.9563	.9497
Zhao et al. [15]	.8620	.9540	.8740	.9560

*Result obtained with a different definition of FOV.

Table V: Performance of the NO-POOL architecture in knowledge transfer mode.

	DRIVE		STARE	
Training set	AUC	Acc	AUC	Acc
DRIVE	.9720	.9495	.9595±.0222	.9505±.0093
STARE	.9605	.9416	.9785±.0066	.9566±.0082
DRIVE+STARE	.9700	.9491	.9776±.0068	.9566±.0067

database performs on par with them. This is a nontrivial result that corroborates the robustness of neural networks, given that these datasets were acquired using two devices with noticeably different physical resolutions (Table I).

Training of a network can take up to 8 hours on a single GPU. Compared to that, preprocessing is relatively cheap: it takes only 6 seconds on average to apply the GCN transformation and 124 seconds to perform the ZCA whitening of the dataset containing 400,000 images on a multi-core CPU with matrix operations executed on GPU. These figures are however irrelevant from the viewpoint of a prospective user of a trained network, who would only query it on a new image. Fortunately, that is quite fast as well: it takes only 246 milliseconds (NVIDIA GTX Titan GPU) on average to classify a batch of 100 patches and 92 seconds to segment an entire image (containing usually a few hundred of thousands patches), a timing we find acceptable in clinical use scenario. This time could be further reduced by partitioning the image into several windows and delegating separate GPUs to each of them. As decisions for individual pixels (or output windows in the structured case) are made independently, one may expect close-to-linear acceleration with the number of engaged GPUs. Also, the trained network may be redundant in featuring neurons (or possibly entire feature maps) that have marginal impact on network output. Such network components could be identified and removed, leading to further speedups of querying time.

VII. Structured prediction

In Section II, we posed the segmentation problem as a conventional machine learning task, with individual pixels classified separately. Such an approach ignores the fact that spatial relations between pixels influence their odds for representing particular structures. A pixel representing background is more likely to be neighbored by another background pixel than by a vessel pixel. The information on the actual class labels of neighboring pixels could help making better decisions – if it only was available at the time of querying.

Nevertheless, we can make use of this observation during training, because at that time pixel labels are available for all pixels in the image. To that aim, we resort to a relatively recent machine learning formalism of *structured prediction* (SP) [34]. More specifically, we pose the segmentation task as a *multi-label* inference problem on a set of binary predictions subject to a joint loss. This special case of SP is particularly appealing, as it currently remains unclear how to render an arbitrarily structured prediction problem onto a structure which is amenable to stochastic gradient descent optimization. Fortunately, extending existing deep architectures to multi-class setting requires only slight modifications.

Multi-label learning is the problem of finding a model that maps inputs to binary vectors $\mathbf{y}$ rather than to scalars. A natural neural approach to this problem is Backpropagation for Multi-label Learning (BP-MLL) [35], where a neural network has multiple output nodes, one for each label. The recent extension of this approach [36] shows that more efficient and more effective training can be attained by replacing BP-MLL's pairwise ranking loss with cross entropy (CE) loss defined as:

$$J_{CE}(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_i (y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i))$$

where $\hat{y}_i$ and y_i are the prediction and the target for the ith output node, respectively. CE is commonly used to train neural networks for classification together with sigmoid transfer function in the output layer. It can be shown that minimizing log loss on y_is for individual output nodes in a network is equivalent to minimizing CE loss on $\mathbf{y}$. CE is also computationally efficient and scales well with the number of labels.

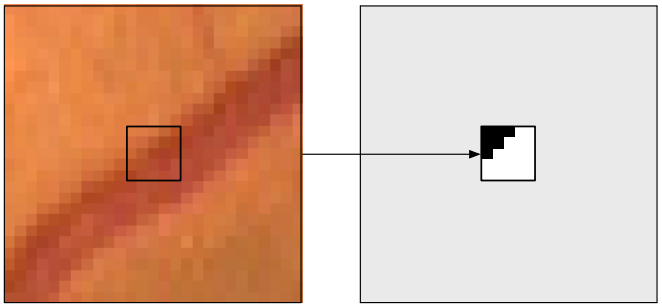


Figure 8: A training example for the SP approach: the 27×27 patch with the $s \times s = 5 \times 5$ output window (left) and the corresponding desired output (right).

In the following SP experiment, we predict the class assignment for all pixels in an $s \times s$ window centered in the 27×27 image patch, and thus solve an s^2-label classification problem. An example is now a pair composed of 27×27 RGB image patch that forms input to the network and s^2 desired outputs (target labels) associated with the pixels in the $s \times s$ window centered at the input patch (Fig. 8). We apply only necessary changes to the architecture of the two-output networks used so far, i.e., we replace the two output neurons (*fc*2 layer in Table II) with s^2 output neurons, which are fully connected to the second last layer. Other settings remain unchanged.

In testing, every querying of a network on a patch produces s^2 decisions for the pixels in the output window. As the input patches on which a network is queried overlap, so do the output windows. We obtain thus s^2 network's output values for each pixel, which are then summed and thresholded, analogously to the conventional approach.

We consider only BALANCED and NOPOOL configurations, as they fared the best in the non-SP setting. In Table VI we present the results obtained for $s = 3, 5, 7$, i.e., with networks classifying 9, 25 and 49 simultaneously. Confrontation of these results with those in Table III clearly shows that SP brings material advantage. Both architectures perform better, particularly for $s = 5$. On the DRIVE dataset, SP allows us to improve AUC by 0.5 percentage point (pp), while on STARE by 1.1pp. In terms of Acc, we gain respectively 0.4pp and 1.09pp. With these results, we improve upon the best-to-date results by Osareh et al. [28] by 1.4pp (DRIVE) and Marin et al. [18] by 1.61pp (STARE; see Table IV).

Importantly, this does not incur significant overhead in training nor testing: the SP architectures involve only $512(s^2 - 2)$

Table VI: Performance of SP models (point estimates for DRIVE, averages with .95 confidence intervals for STARE).

		DRIVE					STARE				
	s	AUC	Acc	Kappa	Sens	Spec	AUC	Acc	Kappa	Sens	Spec
BALANCED-SP	3	.9787	.9507	.7925	.8460	.9673	.9930±.0016	.9667±.0038	.8338±.0163	.9289±.0124	.9710±.0039
NO-POOL-SP	3	.9771	.9519	.7891	.8020	.9757	.9917±.0018	.9713±.0031	.8426±.0173	.8509±.0303	.9850±.0020
BALANCED-SP	5	.9788	.9530	.7953	.8149	.9749	.9928±.0016	.9700±.0031	.8453±.0137	.9075±.0165	.9771±.0029
NO-POOL-SP	5	.9790	.9535	.7910	.7811	.9807	.9928±.0014	.9729±.0027	.8507±.0155	.8554±.0286	.9862±.0018
BALANCED-SP	7	.9729	.9503	.7821	.7996	.9740	.9857±.0032	.9623±.0044	.8021±.0204	.8506±.0281	.9752±.0042
NO-POOL-SP	7	.9747	.9518	.7828	.7750	.9795	.9868±.0025	.9638±.0043	.7933±.0282	.7766±.0495	.9854±.0028

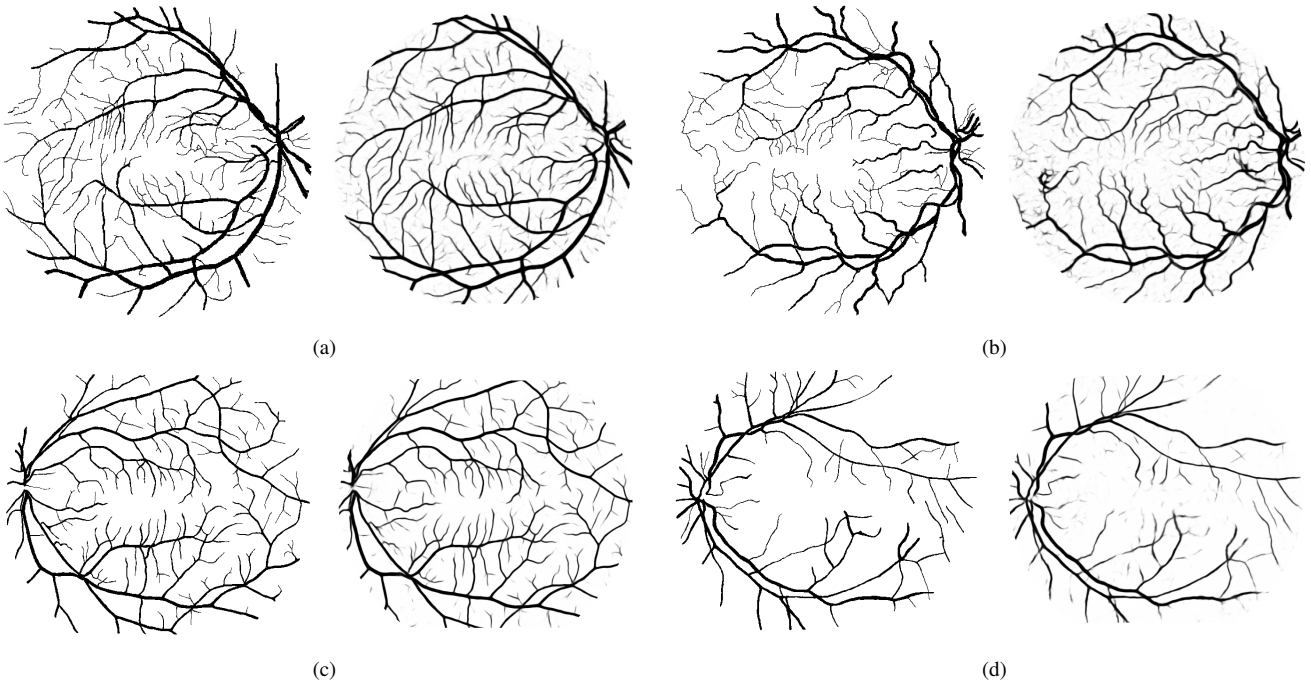


Figure 9: Ground truth (left) and segmentation result (right) for two healthy subjects: (a) DRIVE image #13, (c) STARE image #0255, and two subjects with pathologies: (b) DRIVE image #8, (d) STARE image #0002.

more weights than the ones considered earlier, which is negligible given the overall millions of weights.

We present exemplary segmentations produced by networks trained in the NO-POOL-SP configuration for selected healthy and pathological subjects from both databases in Fig. 9. The agreement with the ground truth is very good, with few spatially isolated false positives. Errors occur mainly when local contrast is low and for the thinnest vessels. Segmentations for all images and other results are available online.[6]

To identify the essential differences between the operational characteristics of non-SP and SP networks, in Fig. 10 we juxtapose the segmentation outcomes of a non-SP NOPOOL network and a NOPOOL-SP network for $s = 5$, as these configurations achieved overall best accuracy in Tables III and VI respectively. The SP networks reconstruct the boundaries of vessels more precisely and are clearly more conservative in committing relatively few FP errors (in red). On the other hand, they fail to detect some fine vessels, leading to FN errors (in blue). The conventional, non-SP networks commit more FP errors on the edges of vessels and are more eager to respond positively for the finest vessels , including – interestingly – those that have not been manually segmented but are clearly present in the image. Therefore, although the non-SP networks are inferior in terms of performance indicators presented in tables, they may be still useful for detecting the fine vessels and reliable reconstruction of entire vessel network. We corroborate this observation experimentally at the end of Section VIII.

[6]http://www.cs.put.poznan.pl/dltmi

VIII. ROBUSTNESS

Segmentation of blood vessels becomes harder in presence of pathologies like microaneurysms and hemorrhages. To assess the robustness of networks in such conditions, we factor the performance indicators from Table VI into healthy and pathological cases for the STARE database and present them in Table VII (DRIVE does not contain diagnosis). The top part of the table presents the averages for the 10 leave-one-out folds where networks were tested on a healthy subject, and analogously for the 10 pathological subjects in the bottom part of the table. The networks are largely robust: compared to Table VI, AUC and Acc for the pathological cases drop by

Table VII: Performance of SP models on pathological and healthy subjects from STARE.

s	AUC	Acc	Kappa	Sens	Spec
STARE-healthy (BALANCED-SP)					
3	.9947±.0012	.9693±.0029	.8585±.0099	.9419±.0137	.9731±.0032
5	.9945±.0010	.9715±.0024	.8660±.0076	.9276±.0156	.9774±.0024
7	.9899±.0033	.9662±.0049	.8389±.0166	.8821±.0318	.9777±.0026
STARE-healthy (NO-POOL-SP)					
3	.9939±.0014	.9729±.0033	.8687±.0098	.8941±.0240	.9836±.0018
5	.9944±.0011	.9740±.0029	.8737±.0086	.8966±.0219	.8450±.0016
7	.9897±.0024	.9656±.0049	.8301±.0183	.8387±.0418	.9830±.0028
STARE-pathological (BALANCED-SP)					
3	.9912±.0028	.9641±.0075	.8090±.0227	.9159±.0194	.9690±.0079
5	.9911±.0030	.9685±.0063	.8247±.0196	.8874±.0254	.9768±.0060
7	.9814±.0044	.9585±.0073	.7652±.0154	.8192±.0420	.9726±.0085
STARE-pathological (NO-POOL-SP)					
3	.9895±.0029	.9697±.0057	.8165±.0246	.8077±.0427	.9864±.0038
5	.9912±.0023	.9718±.0051	.8277±.0227	.8142±.0403	.9879±.0033
7	.9839±.0039	.9620±.0080	.7564±.0451	.7144±.0773	.9877±.0049

Table VIII: Performance of SP models on CHASE database.

s	AUC	Acc	Kappa	Sens	Spec
BALANCED-SP					
3	.9824±.0016	.9436±.0045	.7394±.0172	.9161±.0070	.9468±.0053
5	.9845±.0015	.9577±.0031	.7870±.0132	.8793±.0087	.9668±.0037
7	.9811±.0019	.9553±.0032	.7736±.0141	.8634±.0102	.9659±.0039
NO-POOL-SP					
3	.9815±.0017	.9621±.0020	.7899±.0107	.7913±.0166	.9818±.0024
5	.9823±.0016	.9628±.0020	.7908±.0111	.7816±.0178	.9836±.0022
7	.9801±.0017	.9610±.0019	.7761±.0115	.7544±.0193	.9846±.0021

a fraction of percent only. In return for that, the indicators for the healthy subject are correspondingly better than in Table VI, with AUC getting close to 99.5 percent.

To corroborate these results, we perform additional experiments on the CHASE database of fundus images acquired from children (cf. Section II), abounding in *vessel reflex artifact*, i.e., a bright strip running down the length of wider vessels as well as in the arteries. Figure 11a presents magnified fragments of two CHASE images with a particularly prominent vessel reflex.

We prepare the training and test set in the same way as for the other databases, perform the same train-and-test cycle with SP networks, and summarize the results in Table VIII. Despite the more demanding character of these data, deep learning manages to maintain high values of most performance indicators, typically in between those for DRIVE and STARE (Table VI). Crucially, these numbers are also higher than those reported in all known to us studies on CHASE (Table IX). In particular, our best network (BALANCED-SP-5) achieves AUC score that is 0.85 percentage point (pp) higher than the best-to-date result by Fraz et al. [39]. In terms of Acc, the

Table IX: Best-to-date results on CHASE database.

Method	AUC	Acc
2nd. observer [37]	-	.9538
Roychowdhury et al. [38]	.9623	.9494
Azzopardi et al. [14]	.9487	.9387
Roychowdhury et al. [17]	.9532	.9530
Fraz et al. [39]	.9760	.9524
Fraz et al. [37]	.9712	.9469

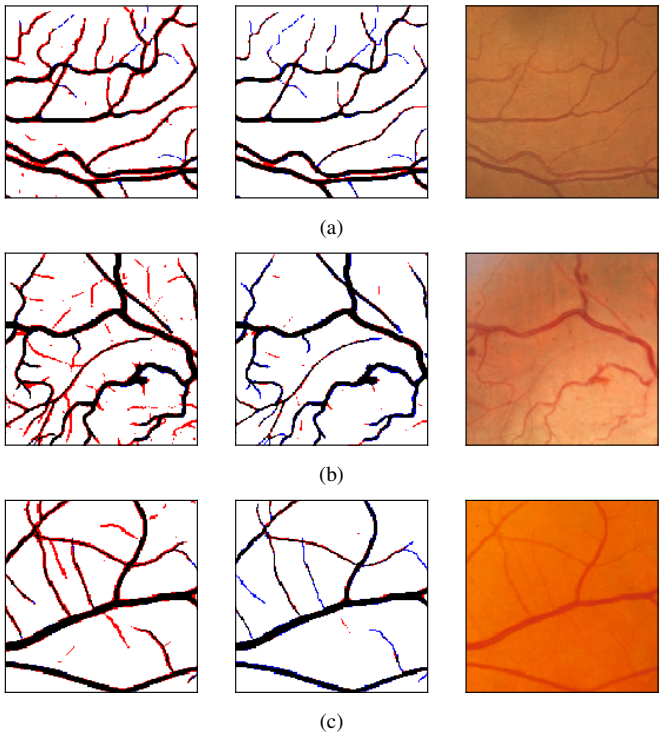
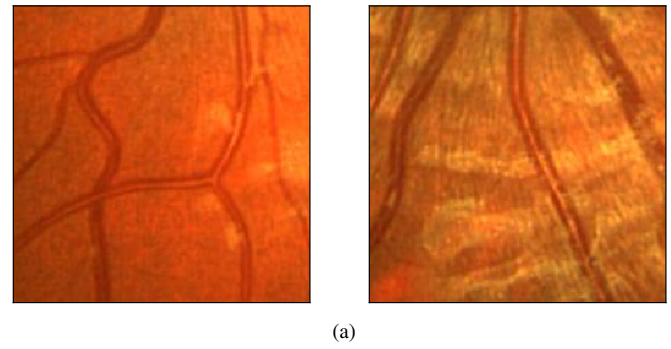


Figure 10: Examples of segmentation by non-SP (left) and SP (middle) models for images (right) of DRIVE (a) and STARE (b, c) database. Red pixels: FP errors, blue: FN errors, black: TP, white: TN.

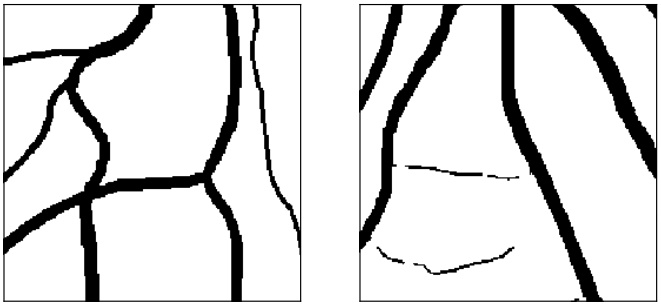
NO-POOL-SP-5 fares the best and improves on the result by Roychowdhury et al. [17] by nearly 1pp. This network also scores the overall highest Kappa, suggesting that the highest results in terms of Acc stems from striking the best balance in sensitivity and specificity. In Fig. 11b, we present magnified segmentation results for the regions with central vessel reflex: the networks are not misled by the reflex and classify most vessel pixels correctly. The accompanying online material[6] provides visualizations for these and other images from the CHASE database.

An important characteristics of a vessel segmentation algorithm is its capability to detect capillaries. In order to examine this characteristics of DL networks, we modify the manual segmentations in the DRIVE database to automatically remove the thick vessels. To this aim, we perform morphological opening, discard the connected components with area ≥ 100 pixels, subtract this image from the original manual segmentation, and remove from the resulting image the connected components that are smaller than 8 pixels (source code of the ImageJ script available online[6]). Figure 12 presents the resulting partitioning of vessels for two training images. Although an expert asked to segment only fine vessels could likely produce a slightly different marking, we claim that the vessels identified in this way form a representative sample of actual fine vessels in the original manual segmentation.

Preliminary experiments showed that the 27×27 patches used so far are too big for detection of thin vessels, so we reduce them to 17×17 pixels, while keeping unchanged the remaining choices about of network architecture (Table



(a)



(b)

Figure 11: Image fragments with central vessel reflex (a) and corresponding segmentation results (b); CHASE database.

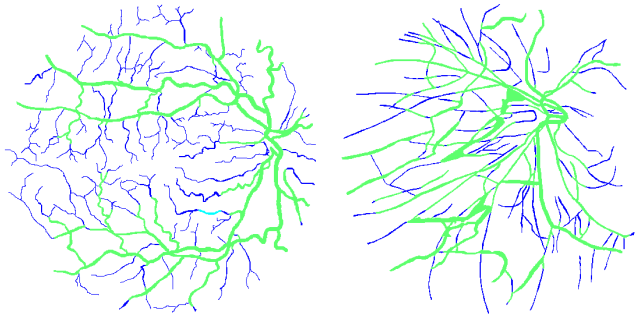


Figure 12: Two manual segmentations from the DRIVE database (images #22 and #34) automatically partitioned into thick vessels (green) and fine vessels (blue).

II). When preparing the training and test sets, every blue pixel in Fig. 12 with the surrounding 17×17 patch forms a positive example; the negative examples are drawn from the background. However, no patch is allowed to overlap with the thick vessels (in green).

Table X presents the performance of networks trained and tested on fine vessels defined in this way, using configurations that proved most successful to this point, i.e., NO-POOL and BALANCED trained to predict one pixel as in Section VI (first two rows in the table) and s^2 pixels in the structural prediction mode (-SP, Section VII). Note that these numbers cannot be directly compared to the previous results, as the problem considered here is different, with mere 4 percent of patches forming the positive class (compared to $14-15$ percent for the original database, cf. Table I). This imbalance, combined with poor visibility of fine vessels, makes this segmentation problem particularly hard.

Table X: Performance of SP on fine vessels (DRIVE).

	s	AUC	Acc	Kappa	Sens	Spec
BALANCED	–	.9470	.8993	.3721	.8703	.9005
NO-POOL	–	.9286	.9668	.4412	.3459	.9929
BALANCED-SP	3	.9493	.9625	.5643	.6577	.9754
NO-POOL-SP	3	.9480	.9695	.5393	.4852	.9899
BALANCED-SP	5	.9515	.9692	.5626	.5466	.9867
NO-POOL-SP	5	.9502	.9687	.5545	.5390	.9864

The results in Table X indicate that particular networks exploit the trade-off between false positives and false negatives in a different way, and none of the configurations is clearly leading on multiple performance indicators. Sensitivity is particularly important in this context, as it reflects detector's capability of noticing the subtle signs of the thinnest and barely visible vessels. The BALANCED, non-SP network fares particularly well on that indicator, which corroborates the observations made earlier in connection to Fig. 10: this network configuration seems to be particularly good at restoring fine vessels, and can even suggest their presence where expert's marking is missing. The spatial arrangement of regions of FPs in Fig. 10 (in red) and their connectivity with the vessels marked by an expert (black and blue) clearly suggest that the positive detections represent largely small blood vessels, and not other anatomical structures.

IX. Conclusions

This study brings evidence that deep neural networks are a viable methodology for medical imaging, even though they solve the task in question in a different way than virtually all well-documented past work. We find this encouraging, in particular given the entirely supervised character of the neural approach, which learns from raw pixel data and does not rely on any prior domain knowledge on vessel structure. While learning, a network autonomously extracts low-level features that are invariant to small geometric variations, then gradually transforms and combines them into higher-order features. In this way, the raw raster image is transformed into a more abstract – and a priori unknown – representation that fosters effective vessel segmentation. The features learned at multiple levels of abstraction are then automatically composed into a complex function that maps an input patch to its label.

The superior performance of SP networks suggests that learning about the dependencies between class labels for neighboring pixels is particularly important in segmentation of continuous anatomical structures. We hypothesize that deep learning may prove effective also when detecting similar structures in other modalities of ophthalmological imaging, and currently work on verifying this claim for OCT.

Medical domain knowledge is notoriously hard to elicit from human experts; on the other hand, gathering large volumes of medical imaging was never easier than today. In this context, reliance on supervised techniques is a necessity. The performance of neural deep learning here and in generic vision tasks suggest that its full potential in medical imaging is yet to be revealed.

Acknowledgment

Research reported in this study was supported by the National Centre for Research and Development, Poland, grant #PBS1/A9/20/2013.

REFERENCES

[1] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.

[2] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[3] B. B. LeCun, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," in *Advances in neural information processing systems*. Citeseer, 1990.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[5] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *International Conference on Artificial Intelligence and Statistics*, 2011, pp. 315–323.

[6] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[7] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle *et al.*, "Greedy layer-wise training of deep networks," *Advances in neural information processing systems*, vol. 19, p. 153, 2007.

[8] M. Fraz, P. Remagnino, A. Hoppe, B. Uyyanonvara, A. Rudnicka, C. Owen, and S. Barman, "Blood vessel segmentation methodologies in retinal images - a survey," *Comput. Methods Prog. Biomed.*, vol. 108, no. 1, pp. 407–433, Oct. 2012. [Online]. Available: http://dx.doi.org/10.1016/j.cmpb.2012.03.009

[9] J. Staal, M. D. Abràmoff, M. Niemeijer, M. A. Viergever, and B. van Ginneken, "Ridge-based vessel segmentation in color images of the retina," *Medical Imaging, IEEE Transactions on*, vol. 23, no. 4, pp. 501–509, 2004.

[10] A. Hoover, V. Kouznetsova, and M. Goldbaum, "Locating blood vessels in retinal images by piecewise threshold probing of a matched filter response," *Medical Imaging, IEEE Transactions on*, vol. 19, no. 3, pp. 203–210, 2000.

[11] C. G. Owen, A. R. Rudnicka, R. Mullen, S. A. Barman, D. Monekosso, P. H. Whincup, J. Ng, and C. Paterson, "Measuring retinal vessel tortuosity in 10-year-old children: validation of the computer-assisted image analysis of the retina (caiar) program," *Investigative ophthalmology & visual science*, vol. 50, no. 5, pp. 2004–2010, 2009.

[12] E. Ricci and R. Perfetti, "Retinal blood vessel segmentation using line operators and support vector classification," *Medical Imaging, IEEE Transactions on*, vol. 26, no. 10, pp. 1357–1365, 2007.

[13] F. M. Villalobos-Castaldi, E. M. Felipe-Riverón, and L. P. Sánchez-Fernández, "A fast, efficient and automated method to extract vessels from fundus images," *J. Vis.*, vol. 13, no. 3, pp. 263–270, Aug. 2010. [Online]. Available: http://dx.doi.org/10.1007/s12650-010-0037-y

[14] G. Azzopardi, N. Strisciuglio, M. Vento, and N. Petkov, "Trainable cosfire filters for vessel delineation with application to retinal images," *Medical image analysis*, vol. 19, no. 1, pp. 46–57, 2015.

[15] Y. Zhao, L. Rada, K. Chen, S. P. Harding, and Y. Zheng, "Automated vessel segmentation using infinite perimeter active contour model with hybrid region information with application to retinal images," *Medical Imaging, IEEE Transactions on*, vol. 34, no. 9, pp. 1797–1807, 2015.

[16] J. V. Soares, J. J. Leandro, R. M. Cesar Jr, H. F. Jelinek, and M. J. Cree, "Retinal vessel segmentation using the 2-d gabor wavelet and supervised classification," *Medical Imaging, IEEE Transactions on*, vol. 25, no. 9, pp. 1214–1222, 2006.

[17] S. Roychowdhury, D. D. Koozekanani, and K. K. Parhi, "Blood vessel segmentation of fundus images by major vessel extraction and subimage classification," *Biomedical and Health Informatics, IEEE Journal of*, vol. 19, no. 3, pp. 1118–1128, 2015.

[18] D. Marín, A. Aquino, M. E. Gegúndez-Arias, and J. M. Bravo, "A new supervised method for blood vessel segmentation in retinal images by using gray-level and moment invariants-based features," *Medical Imaging, IEEE Transactions on*, vol. 30, no. 1, pp. 146–158, 2011.

[19] R. Nekovei and Y. Sun, "Back-propagation network and its configuration for blood vessel detection in angiograms." *IEEE Transactions on Neural Networks*, vol. 6, no. 1, pp. 64–72, 1995. [Online]. Available: http://dblp.uni-trier.de/db/journals/tnn/tnn6.html#NekoveiS95

[20] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[21] Y. Bengio, "Learning deep architectures for ai," *Foundations and trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[22] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[23] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," *Computer Science Department, University of Toronto, Tech. Rep*, vol. 1, no. 4, p. 7, 2009.

[24] A. Hyvärinen and E. Oja, "Independent component analysis: algorithms and applications," *Neural networks*, vol. 13, no. 4, pp. 411–430, 2000.

[25] A. J. Bell and T. J. Sejnowski, "Edges are the" independent components" of natural scenes," in *NIPS*, 1996, pp. 831–837.

[26] A. Dosovitskiy, J. T. Springenberg, M. Riedmiller, and T. Brox, "Discriminative unsupervised feature learning with convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2014, pp. 766–774.

[27] X. Jiang and D. Mojon, "Adaptive local thresholding by verification-based multithreshold probing with application to vessel detection in retinal images," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 25, no. 1, pp. 131–137, 2003.

[28] A. Osareh and B. Shadgar, "Automatic blood vessel segmentation in color images of retina," *Iran. J. Sci. Technol. Trans. B: Engineering*, vol. 33, no. B2, pp. 191–206, 2009.

[29] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[30] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," *arXiv preprint arXiv:1412.6806*, 2014.

[31] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 807–814.

[32] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *International conference on artificial intelligence and statistics*, 2010, pp. 249–256.

[33] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.

[34] G. Bakir, *Predicting structured data*. MIT press, 2007.

[35] M.-L. Zhang and Z.-H. Zhou, "Multilabel neural networks with applications to functional genomics and text categorization," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 18, no. 10, pp. 1338–1351, 2006.

[36] J. Nam, J. Kim, E. L. Mencía, I. Gurevych, and J. Fürnkranz, "Large-scale multi-label text classification revisiting neural networks," in *Machine Learning and Knowledge Discovery in Databases*. Springer, 2014, pp. 437–452.

[37] M. M. Fraz, P. Remagnino, A. Hoppe, B. Uyyanonvara, A. R. Rudnicka, C. G. Owen, S. Barman *et al.*, "An ensemble classification-based approach applied to retinal blood vessel segmentation," *Biomedical Engineering, IEEE Transactions on*, vol. 59, no. 9, pp. 2538–2548, 2012.

[38] S. Roychowdhury, D. D. Koozekanani, and K. K. Parhi, "Iterative vessel segmentation of fundus images," *Biomedical Engineering, IEEE Transactions on*, vol. 62, no. 7, pp. 1738–1749, 2015.

[39] M. M. Fraz, A. R. Rudnicka, C. G. Owen, and S. A. Barman, "Delineation of blood vessels in pediatric retinal images using decision trees-based ensemble classification," *International journal of computer assisted radiology and surgery*, vol. 9, no. 5, pp. 795–811, 2014.