Mean Field Networks for Retinal Blood Vessel Segmentation

Master Thesis

Basile Nicolas Rommes

MSc Bioinformatics

Supervisors: Raghavendra Selvan and Jens Petersen

Monday 2nd September, 2019

Contents

Abstract

This report analyses mean field network (MFN) formulations for segmenting blood vessels in eye-fundus images. Blood vessel segmentation is a classic problem in ophthalmology and it involves the prediction of long, elongated structures that thin towards the end into single pixel thickness. Extracting the vessel tree of an eye-fundus image is a well studied problem. MFNs allow a high degree of customizability, because the formulation of the model involves the design of an energy function that determines how the network learns. This allows to use expert knowledge about the data to your advantage for better prediction. Furthermore, this property makes them interesting for analysis, as the choice of energy function may inform about underlying properties of the data. This report investigates the use of different MFN formulations on this particular task via different models of connectedness. First a Multilayer perceptron is applied on the task both to serve as a baseline for comparison and later as an integral part of the MFN, serving as it's unary potential. Then three different MFN formulations are run on the same task. It is shown that increasing connectedness does indeed increase prediction success.

The code is publicly available at `https://github.com/romba050/MFN_RBV_segmentation`.

Acknowledgments

I thank my supervisors, Raghavendra Selvan and Jens Petersen for their continued support and for engaging discussions during our regular meetings as well as for the opportunity to get to work on this fascinating problem. I thank my friends for support and for proof-reading of the final report, especially Flemming Morsch, Lys Sans Moreta, Steen Petersen and Balthasar Schlotmann, as well as Rūta Masiulytė for the inspiring exchange of ideas while working on a similar problem during the past 6 months.

Abbreviations

- Acc = Accuracy
- CLAHE = contrast-limited adaptive histogram equalization
- CRF = conditional random field
- ELBO = evidence lower bound
- FoV = field of view
- KLD = Kullback-Leibler divergence
- MFI = mean field iteration
- MFN = mean field network
- MLP = multi linear perceptron
- ReLU = rectified linear unit
- SLF = standard logistic function
- Se = Sensitivity
- Sp = Specificity
- VI = variational inference

Notation

$$X = \text{pixel features} \tag{1}$$
$$Y = \text{pixel labels} \tag{2}$$
$$x_i = \text{feature vector of pixel i} \tag{3}$$
$$y_i = \text{label of pixel i} \tag{4}$$
$$P(X, Y) = \text{joint distribution} \tag{5}$$
$$Q_i^0(Y) = \text{approximate prob distr. after MLP over pixels } i \in \{1...N\} \tag{6}$$
$$Q_i^k(Y) = \text{approximate prob distr. after k message passing steps of MFN} \tag{7}$$
$$E(X, Y) = \text{Gibb's energy / energy function} \tag{8}$$
$$\psi_u = \text{unary potential} \tag{9}$$
$$\psi_p = \text{pairwise potential} \tag{10}$$
$$Z = \text{partition function / normalizing factor} \tag{11}$$
$$N = \text{number of pixels} \tag{12}$$
$$N_e = \text{number of edges in the conditional random field} \tag{13}$$
$$N_i = \{\forall \text{ pixel } j \colon (i, j) \text{ is edge in CRF}\} \quad \text{(neighbourhood of } i\text{)}$$
$$F = \text{number of features per pixel} \tag{14}$$
$$\mathbb{1}[\text{condition}] = \begin{cases} 1 & \text{, if condition)} \\ 0 & \text{, else} \end{cases} \quad \text{(indicator function)}$$
$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad \text{(standard logistic function)}$$
$$\text{logit}(p) = \log\left(\frac{p}{1 - p}\right) \quad \text{(logit function)}$$
$$\tag{15}$$

1 Introduction

Blood vessel segmentation in eye-fundus images is a tedious and time-consuming task for experts in ophthalmology. It is, however, an important and common procedure that serves to visualize, to diagnose and to plan for treatment and surgery of ocular diseases [1]. It is therefore among the most used modalities to examine human eyes [2]. Perhaps most prominently, it is used in the case of diabetes mellitus patients in order to test for early signs of diabetic retinopathy, a comorbid disease and a leading cause of blindness among working-aged people in the US [3]. In other medical procedures, such as analysis of pigment epithelium or choroid related abnormalities, blood vessels can become an artifact that leads to misdiagnosis and have to be identified in order to not be confused as signals [4]. Properly identifying blood vessels will lead to faster and more accurate diagnostics.

In image segmentation, the task is to assign to each pixel a label corresponding to one out of a defined number of classes. For example, a dataset of photographs from parks could be segmented into the following four classes: people, trees, benches and background. In eye-fundus segmentation, this task is usually defined as a binary segmentation into vessels and background. Since this task is well studied, it is a good target to apply newly designed machine learning models on. In this thesis Mean Field Networks [5] are investigated, since they offer a multitude of design choices and work on a relatively small amount of trainable parameters. This allows to give insight into what the model has learned and in turn, gain knowledge about the particularities of the task and/or Mean Field Networks in general. Mean Field networks are a supervised learning method, which represents datapoints (e.g. the features of pixels) as nodes and allows them to pass information via edges in a graph. This message passing is performed iteratively, such that information is received from neighbours of neighbours etc, which leads to far reaching roads of communications between nodes. Message passing is controlled by parameters trainable via back propagation, which means each iteration of the process can be interpreted as a layer of an artificial neural network [6]. Contrary to neural networks, the weights of the MFN's energy function are scaling user-defined pixel similarities, such that they offer insight into the usefullness of each features for prediction.

A multitude of supervised and unsupervised machine learning methods have been applied to the problem, including: decision trees, support vector machines, different types of neural networks, Gaussian mixture models, random forest and maximum a posteriori estimation [1, 7]. Some of the best performing methods combine classifiers, such as the one by Wang et al. [8] which combines convolutional neural networks and random forests to achieve 97,67% accuracy on the DRIVE and STARE databases. The focus of this thesis is not be beat the state of the art, but rather to investigate how MFNs operate on the problem. An advantage of the low amount of parameters needed for mean field is that it may be better suited for small datasets. Consciously training models on small or sparse datasets is known as "few-shot training" and has several major advantages:

- good performance even when faced with scarcity of labelled data
- decreased risk of overfitting
- intuitive interpretability of the learned weights

Especially in medical imaging, where examples of specific rare diseases could be sparse even in large datasets, being able to develop models that can predict accurately on a few shots is crucial ("shots" being examples of a particular phenomenon in the dataset).

Mean field networks have shown success in the area of graph airways, using a graph network approach [6]. MFNs have also been applied to the vessel segmentation problem before, see Orlando and Blaschko [9]. What distinguishes this thesis from the Orlando paper is that pixels are locally connected instead of fully connected and the energy function is designed from scratch. Using only local connections significantly reduces the computational overhead, so it is worth investigating if connecting pixels

to their immediate neighbourhood is sufficient. Given that message passing is an iterative procedure, the information between pixels is allowed to flow through several connections, reaching pixels that are further away then the immediate neighbourhood, even when using sparsely connected fields.

The thesis shall investigate several angles of MFN:

1. how different pairwise potentials influence performance of MFNs
2. whether sparsely connected CRFs, with only 8 neighbours per pixel, can be successful in the vessel segmentation task, when compared to a baseline
3. how much does each pixel feature contribute to the success of message passing

"For any given model, a variety of algorithms can often be applied. Conversely, any given algorithm can often be applied to a variety of models. This kind of modularity, where we distinguish model from algorithm, is good pedagogy and good engineering."
-Machine Learning, A Probabilistic Approach by Kevin Murphy [10]

In the spirit of this quote, the machine learning method used in this paper shall be explained from the model and from the algorithm perspective. In our case, the MFN is the model. MFN involve a number of assumptions over the underlying distributions of the problem. They require a choice of energy function and design of a conditional random field (CRF)
The algorithm is variational inference (VI) under the aforementioned Mean Field assumption, and in our case involve the learning of model parameters via backward propagation
A Mean field network is an approximation algorithm for problems over which the exact inference is intractable. In each iteration of the mean field, the approximate marginal distributions are updated by getting information from their neighbours a process called message passing.

1.1 Mean Field Theory

To model the segmentation problem, we consider both the pixel features, X, and their labels, Y, to be random variables. Inference tackles the assignment of a label y_i to a given x_i, finding the distribution of Y, which we assume is dependent on the features X: $P(Y|X)$. Calculating $P(Y|X)$ analytically would involve multi-dimensional integration over the features x_i, which in practice is intractable for one of two reasons: there exists no closed-form solution to the integral or it is computational intractible in the sense that the runtime would scale badly in relation to the size of the problem. To circumvent this we need to make use of some stochastic method such as variational inference (VI) [11]. The VI approach is to approximate the posterior $P(Y|X)$ by some distribution $Q(Y)$ over which some simplifying assumptions are made, and to then iteratively optimize this approximation. The measure used to determine whether our approximation approaches the posterior is the Kullback-Leibler divergence D_{KL}.

$$Q^*(Y) = \underset{Q}{\operatorname{argmin}}\ D_{KL}(Q(Y)||P(Y|X))$$

where:

$$D_{KL}(Q(Y)||P(Y|X)) = \sum_Y Q(Y) \log\left[\frac{Q(Y)}{P(Y|X)}\right] = \mathbb{E}_Q\left[\log\frac{Q(Y)}{P(Y|X)}\right] \tag{16}$$

Our optimization algorithm could in theory minimize D_{KL} directly, but the issue is that $D_{KL}(Q(Y)||P(Y|X))$ can't be calculated without knowing $P(Y|X)$. However, the posterior $P(Y|X)$ stands in relation with the model evidence $P(X)$ according to Bayes' theorem:

$$P(Y|X) = \frac{P(X|Y)\cdot P(Y)}{P(X)}$$

$$\text{posterior} = \frac{\text{likelihood} \cdot \text{prior}}{\text{evidence}}$$

This evidence $P(X)$ (or it's logarithm, to be precise) can be bound from below using Jensen's inequality: $\log \mathbb{E}[x] \geq \mathbb{E}[\log(x)]$.

$$\begin{aligned}
\log(P(X)) &= \log \int_Y P(X,Y) && \text{(law of total probability)} \\
&= \log \int_Y P(X,Y) \frac{Q(Y)}{Q(Y)} \\
&= \log \mathbb{E}_{Q(Y)} \left[\frac{P(X,Y)}{Q(Y)} \right] && \text{(def. } \mathbb{E}\text{)} \\
&\geq \mathbb{E}_{Q(Y)} \left[\log \frac{P(X,Y)}{Q(Y)} \right] && \text{(Jensen's inequality)} \\
&= -\mathbb{E}_{Q(Y)} \left[\log \frac{Q(Y)}{P(X,Y)} \right] && \text{(log laws)} \\
&= \mathbb{E}_{Q(Y)} [\log P(X,Y) - \log Q(Y)] && \text{(log laws)} \\
&= \mathbb{E}_{Q(Y)} [\log P(X,Y)] \underbrace{-\mathbb{E}_{Q(Y)} [\log Q(Y)]}_{+H[Y]} \\
& && \text{(by additivity of expectation, H being the shannon entropy)} \\
&:= \text{ELBO} && \text{(definition of the 'evidence lower bound')}
\end{aligned}$$

We have now found a lower bound to $\log P(X)$ based on the joint probability $P(X,Y)$ and Shannon entropy $H(Y)$, which are easier to compute than D_{KL}. This 'variational bound' is called ELBO ('evidence lower bound') for short [12] and is central to VI.

$$\text{ELBO} = \mathbb{E}_Q [\log P(X,Y)] + H[Y] \tag{17}$$

In order to use the ELBO for approximation, we need to reconcile it with our goal to minimize D_{KL}. Let us thus put D_{KL} and ELBO in relation to each other.

$$\begin{aligned}
D_{KL}(Q||P) &= \mathbb{E}_Q \left[\log \frac{Q(Y)}{P(Y|X)} \right] && (18) \\
&= \mathbb{E}_Q \left[\log \frac{Q(Y)}{\frac{P(X,Y)}{P(X)}} \right] && \text{(Bayes theorem)} \\
&= \mathbb{E}_Q \left[\log \frac{Q(Y) \cdot P(X)}{P(X,Y)} \right] && (19) \\
&= \mathbb{E}_Q [\log Q(Y) - \log P(X,Y) + \log P(X)] && (20) \\
&= \mathbb{E}_Q [\log Q(Y) - \log P(X,Y)] + \log P(X) \\
& \qquad \text{(since P(X) is independent of Y, and Q(Y) only depends on Y)} \\
&= -\mathbb{E}_Q [\log P(X,Y) - \log Q(Y)] + \log P(X) && (21) \\
&= -\text{ELBO} + \log P(X) && (22)
\end{aligned}$$

Equation (22) showcases that D_{KL} is the negative ELBO up to an added constant. In order to find the optimal approximation $Q^*(Y)$, minimizing D_{KL} corresponds to maximizing the ELBO.

$$Q^*(Y) = \underset{Q}{\text{argmax}}\ \text{ELBO}(Q(Y)) \tag{23}$$

To calculate ELBO, we need to determine the expected value of the joint log probability as well as the Shannon entropy. The Shannon entropy $H[Y]$ can be interpreted as the amount of information learned by the model. Since $H(Y)$ describes the amount of change in information of the random variable Y it is expected to decrease while the model predictions converge towards a local optimum.

In order two implement this optimization, we need to calculate the joint distribution and derive the update equation for $Q(Y)$. The first is achieved via Gibbs energy function $E(X, Y)$ that we model as the sum over unary potentials ψ_u and pairwise potentials ψ_p. The crucial difference between those potentials being that the unary potential depends on the pixel features whereas the pairwise potential is dependent on other pixels. Thus, each node is associated with a unary potential and each undirected edge with a pairwise potential, see fig. 2.

A conditional random field (CRF) is a Markov network (also sometimes called Markov random field) with a dependency on a subset of variables [13]. Like all Markov networks, CRFs are a set of random variables on an undirected graph fullfilling the markov property of memorylessness.In the case of CRFs however, the random variables are over a conditional distribution $P(Y|X)$. This means that even though the graph over X is undirected, they have a directed dependency over another random variable Y.

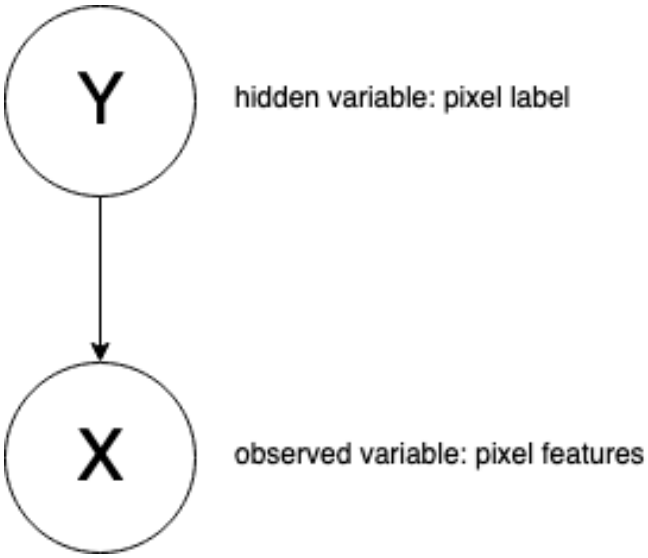


Figure 1: In this probabilistic model, the hidden variable X (defined as a vector of features of the image pixels) underlies the observed variable Y. Edited from Yang et al. [14].

In the rectangular grid representation of an image, each pixel is connected to it's 8 direct neighbours for the pairwise potential. This means that during message passing, the sum of these 8 potentials influences the potential of the central pixel, see fig. 3.

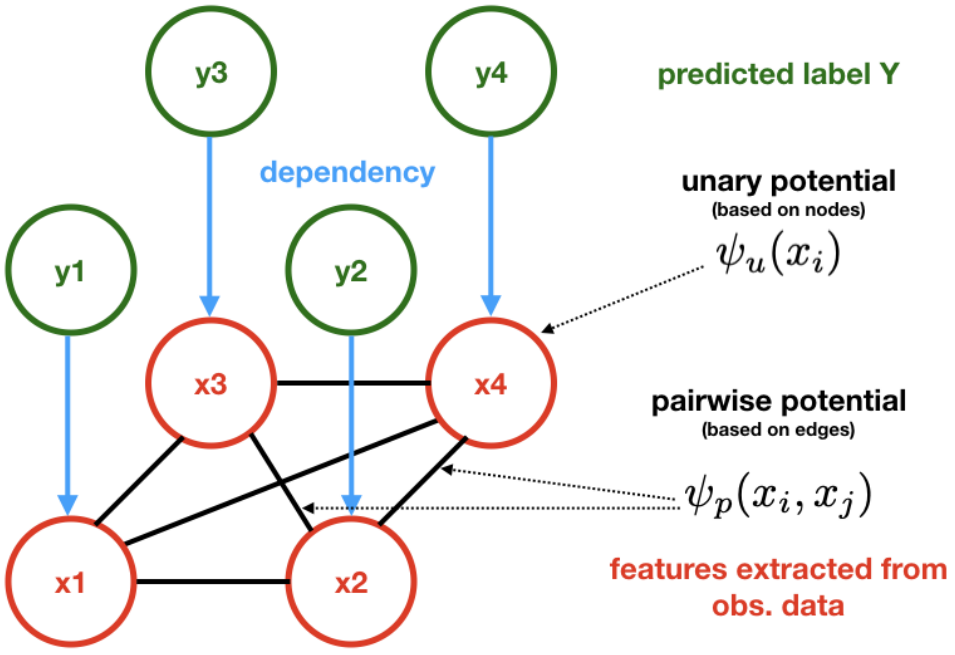


Figure 2: Graphical representation of a conditional random field: for a given pixel i, x_i is it's feature vector and y_i it' predicted label. Neighborhood relations between pixels are expressed by pairwise potential ψ_p, each node x_i has also an associated unary potential ψ_u.

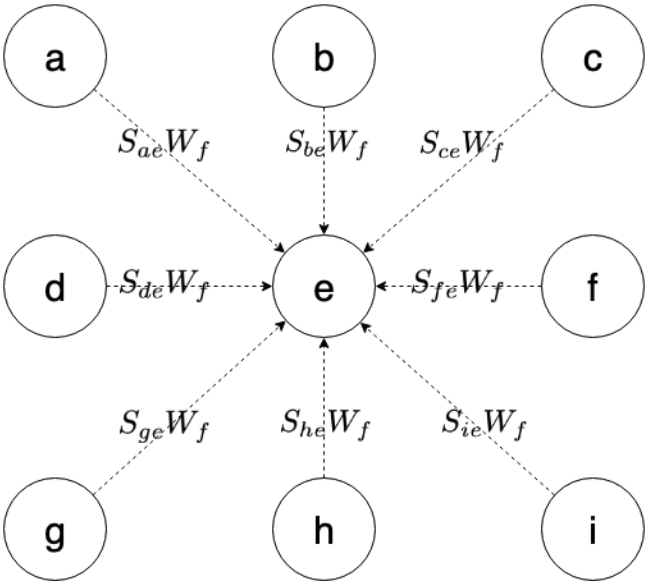


Figure 3: A message passing step for the central pixel e. Information is received from 8 neighbours. The weight vector W_f is the same for each edge, while the similarity S_ij depends on the features of i and j.

1.2 Mean field assumptions

In order to achieve an efficient optimization algorithm, mean fields make a number of simplifying assumptions over the data. Consider the random variable $Y = \{y_i | i \in \{1 \dots N\}$ with $y_i \in \{0, 1\}$. Y represents the labeling over all pixels of an image. The mean field setting assumes then that the approximate distribution $Q(Y)$ is factorizable over all pixel distributions $Q_i(y_i)$.

$$Q(Y) = \prod_i Q_i(y_i) \tag{24}$$

All $Q_i(y_i)$ are Bernoulli distributed over the 2 only possible cases, i.e.

$$Q_i(y_i) = \begin{cases} q_i & \text{, if } y_i = 1 \text{ (blood vessel)} \\ 1 - q_i & \text{, if } y_i = 0 \text{ (background)} \end{cases} \tag{25}$$

Further, mean field defines an energy function over which to optimize $P(X, Y)$. For this thesis, let $P(X, Y) = \exp E(X, Y)$. Then:

$$\begin{aligned} E(X, Y) &= \log P(X, Y) \\ &\approx \log Q(Y) \\ &= \log \prod_i Q_i(y_i) && \text{(see eq. (25))} \\ &= \sum_i \log Q_i(y_i) && \text{(log laws)} \end{aligned}$$

The definition of $E(X, Y)$ allows us to update energies in $\mathbb{R}$ instead of probabilities $\in [0, 1]$. Doing computations in the energy space turns operations that would normally be multiplications into additions. Let us assume that for an effective approximation of $P(y_i|x_i)$, both the features x_i as well as features x_j of neighbours j to i play a role. $E(X, Y)$ becomes then:

$$E(X,Y) = \sum_i \psi_u(x_i, y_i) + \sum_i \sum_{j \in N_i} \psi_p(x_i, x_j, y_i, y_j) \tag{26}$$

Were $\psi_u(x_i, y_i)$ is the unary and $\psi_p(x_i, x_j, y_i, y_j)$ the pairwise potential. In the following we are going to consider the dependency of the potentials on Y implicit and simply write:

$$E(X,Y) = \sum_i \psi_u(x_i) + \sum_i \sum_{j \in N_i} \psi_p(x_i, x_j) \tag{27}$$

1.3 Mean field iterations

Let the probability distribution after the k-th iteration be $Q^k(y)$. Q^0 denotes the distribution after the unary potential and $\mathbb{E}_{Q_{-l}}$ denotes the expectation over all pixels but l. Each mean-field iteration delivers a new probability Q_y by updating the energy function via an update equation derived in section 5.5. It yields the inference algorithm 1.

Algorithm 1 Mean Field inference algorithm

1: $\forall i$ compute $Q_i^0(y_i)$ using the MLP
2: $\forall i$ initialize $\psi_u(x_i) \leftarrow \text{logit}(Q_i^0(y_i))$
3: $\forall i, j$ compute $\psi_p(x_i, x_j)$
4: **for** T iterations **do**
5: $\quad Q_l^k \leftarrow \sigma\left[\mathbb{E}_{Q_{-l}^k}\left[\sum_i \psi_u(x_i) + \sum_i \sum_{j \in N_i} \psi_p(x_i, x_j) - \log Q^k(Y)\right]\right]$

2 HRF dataset

The High-Resolution Fundus (HRF) dataset is provided by Budai et al. [2]. The dataset consists of 45 fundus images and their respective manual segmentation as determined by experts from collaborating ophtalmology clinics. These segmentations will be used as the ground truth to compute the loss on when compared to the model predicitons. Lastly, the dataset contains a mask describing the field of view (FoV) of all fundus images.

To ensure the methods trained on the dataset are robust to pathologies in the training data, the dataset contains 15 images of healthy patients, 15 images of patients with diabetic retinopathy and 15 images of glaucomatous patients.

The original resolution of the images is 2336×3504. To limit computational overhead, both width and height where scaled down to $\frac{1}{8}$, giving dimensions 292×438 and reducing the total number of pixels by a factor of $\frac{1}{64}$. Resizing requires to interpolate pixel values, and so a segmentation of binary values becomes one of probabilities. The HRF dataset contains images from the left as well as from the right eye of the patients. This means that each image of a pair has the fovea in a position that mirrors the one of its counterpart. Therefore, any machine learning approach needs to be able to "learn" this distinction.

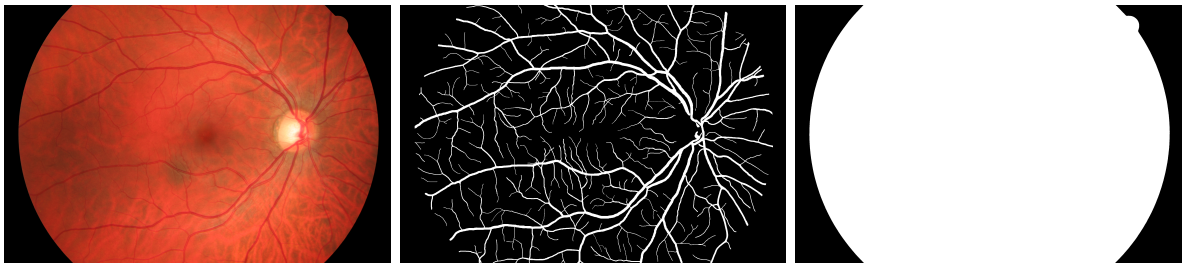


Figure 4: Healthy fundus image, corresponding manual annotation, and mask, taken from the HRF dataset.

All 4 models presented in this work were trained on this dataset, albeit for different amounts of epochs, see table 1.

3 Preprocessing and feature extraction

Classifiers usually do not operate on raw data input, but make use of both preprocessing and feature extraction methods to gather any number of useful features about the data. The feature vector $x_i \in \mathbb{R}$ of pixel i is also called the descriptor or the embedding of i.

Preprocessing of images helps to highlight interesting properties and facilitates learning while simultaneously turning down image noise. The first preprocessing step for vessel analysis is often green channel extraction. It is generally accepted that the green channel of an RGB encoded fundus image is the most expressive when it comes to distinguishing vessels from background [15]. This is due to the fact that the vessels green value is far lower (i.e. darker) than that of the background, whereas this difference is less pronounced in the red and blue channels (see fig. 5).

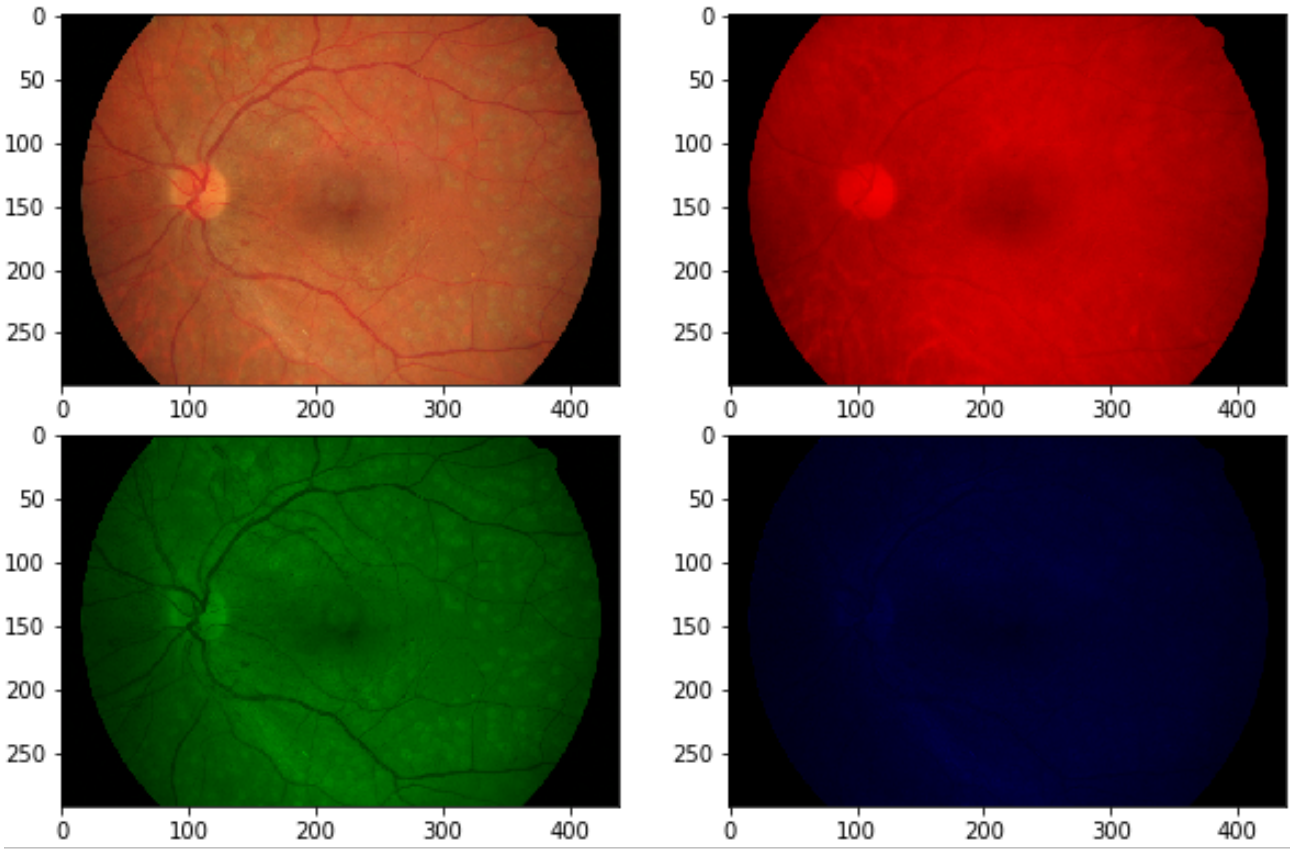


Figure 5: Original image, red channel, green channel and blue channel of fundus image. The green channel exhibits the strongest distinction between vessels and background.

In order to enhance the contrast of pixel intensities of vessels with respect to background pixels, contrast-limited adaptive histogram equalization (CLAHE)[16] was applied to the images. The image histogram (a plot of the image's tonal variations against corresponding the pixel counts) is equalized, which means that the cumulative distribution of the histogram is approximated to a linear function, thereby balancing out the pictures intensity values. By spreading out the most frequent contrasts, this procedure increases local contrasts. The hope is to increase the contrast around areas containing signal (i.e. blood vessels), although since the procedure is not descriminative, AHE may also increase contrast around noise. Especially in homogenous regions of the image the variation in signal intensity may be so close in the variation in noise that AHE smothers the signal. To circumvent that, CLAHE limits the contrast adaptation by clipping the histogram.

Subsequently, gamma correction ($\gamma = 1.2$) was applied for further vessel enhancement [17].

Next, features were extracted, describing pixel properties such as:

- location (i.e. position within the image grid)
- color (color intensities of a given channel)
- shape (shape of nearby pixels of similar brightness intensity)
- texture (spatial variation of the brightness intensity of the pixels)

To that goal, the following features have been extracted:

- green-channel, x-position, y-position (3 features)
- Hessian (3 distinct ones in a 2x2 matrix) (3 features)

- eigenvectors (2 eigenvalues * 2 entries per vector) (4 features)
- eigenvalues (one for each eigenvector) (2 features)

Where the Hessian are the second derivatives with regards to position. Based on the interpretation of the image as a function $I(x, y)$, with x and y being the image coordinates, the hessian is defined as:

$$Hess(I(x,y)) = \begin{bmatrix} \frac{\partial^2 I(x,y)}{\partial x^2} & \frac{\partial I(x,y)}{\partial x \partial y} \\ \frac{\partial F(x,y)}{\partial y \partial x} & \frac{\partial^2 F(x,y)}{\partial y^2} \end{bmatrix}$$

Because certain features appear on different resolutions of the image, gaussian filters with standard deviations $\sigma = 1, 2$ and 4 where applied before calculating the second order derivatives and performing eigenvalue decomposition. Taking the partial derivatives of the second order of the pixel intensities $I(x, y)$ with respect to a combination of x and y coordinates of the pixels, leaves us with 4 combinations of the hessian matrix. After Schwarz' Theorem [18], the second derivatives of $I(x, y)$ are commutative which leave 3 distinct second derivatives. The Hessian, eigenvectors and eigenvalues for 3 different resolutions result in $3 * (3 + 4 + 2) = 27$ features. Together with the green-channel intensity, x-position and y-position this leaves us with a total of 30 features.

Figure 6 shows an example of these 30 features extracted for an arbitrarily chosen image.

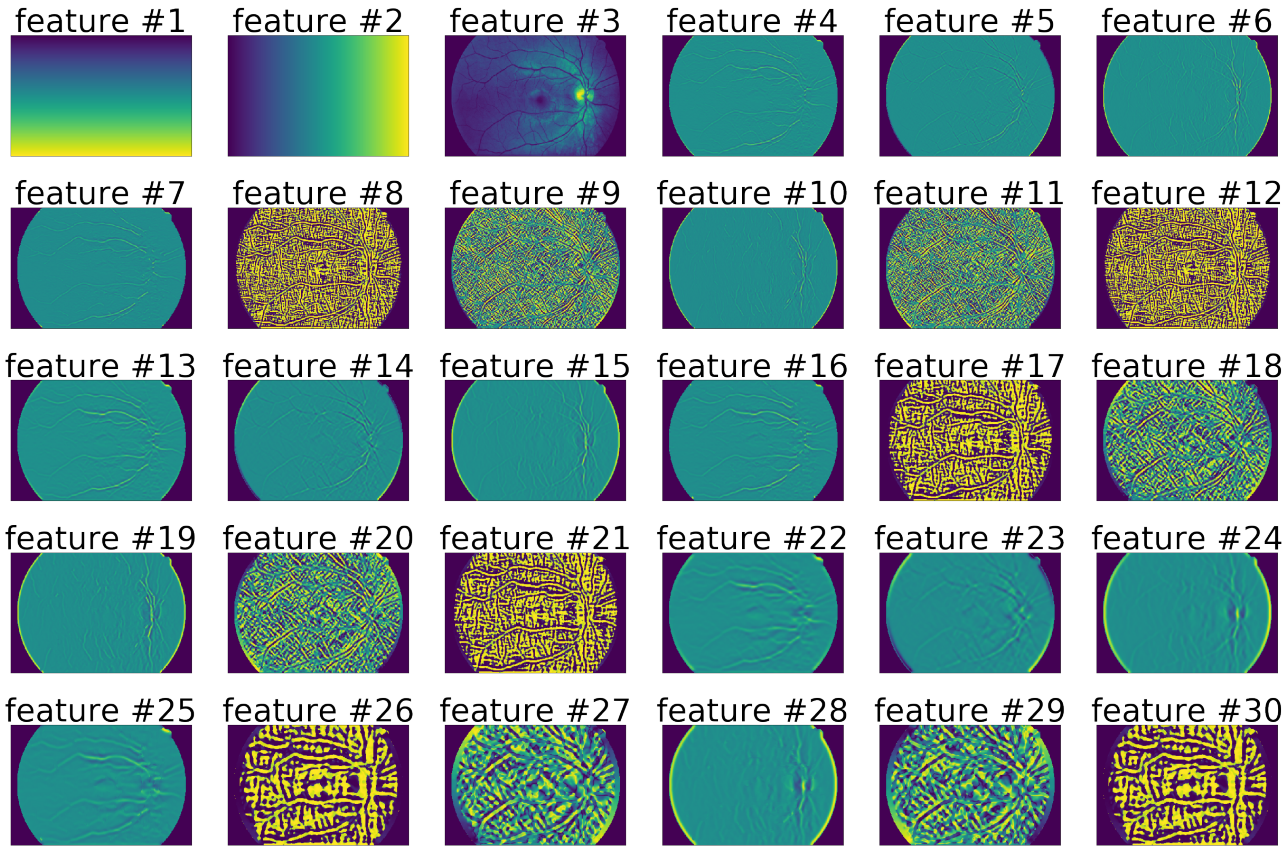


Figure 6: The 30 features extracted from a single image.

Feature 1: x-position	Feature 2: y-position	Feature 3: intensity	Feature 4: entry of hessian, sigma=1	Feature 5: entry of hessian, sigma=1	Feature 6: entry of hessian, sigma=1
Feature 7: eigenvalue sigma=1	Feature 8: eigenvector sigma=1	Feature 9: eigenvector sigma=1	Feature 10 eigenvalue sigma=1	Feature 11 eigenvector sigma=1	Feature 12: eigenvector sigma=1
Feature13: entry of hessian sigma=2	Feature 14: entry of hessian sigma=2	Feature 15: entry of hessian sigma=2	Feature 16: eigenvalue sigma=2	Feature 17: eigenvector sigma=2	Feature 18: eigenvector sigma=2
Feature 19: eigenvalue sigma=2	Feature 20: eigenvector sigma=2	Feature 21: eigenvector sigma=2	Feature 22: entry of hessian sigma=4	Feature 23: entry of hessian sigma=4	Feature 24: entry of hessian sigma=4
Feature 25: eigenvalue sigma=4	Feature 26: eigenvector sigma=4	Feature 27: eigenvector sigma=4	Feature 28: eigenvalue sigma=4	Feature 29: eigenvector sigma=4	Feature 30: eigenvector sigma=4

Figure 7: The interpretation of the features from fig. 6

4 Baseline Model: Muli-layer perceptron

A multilayer perceptron (MLP) is a basic feed-forward artificial neural network consisting of input layer, hidden layers and output layers. Since the goal is to predict probabilities, the final layer uses the standard logistic activation function (SLF), making the MLP perform a regression task.

The respective sizes of input layer, first hidden layer, second hidden layer and output layer are:

$$F \times 32 \times 32 \times 1$$

Each layer is fully connected to the next one, with one trainable parameter per connection. A bias of the size of the sink layer is added. Thus the number of trainable parameters is:

$$(30 * 32) + 32 + (32 * 32) + 32 + (32 * 1) + 1 = 2081$$

```
MLP(
  (fc_layer1): Linear(in_features=30, out_features=32, bias=True)
  (fc_layer2): Linear(in_features=32, out_features=32, bias=True)
  (out_layer): Linear(in_features=32, out_features=1, bias=True)
  (relu): LeakyReLU(negative_slope=0.2)
  (sigmoid): Sigmoid()
)
```

Figure 8: The MLP architecture consists of two fully connnected hidden layers of 32 nodes each and a single node output layer to indicate the vessel probability. Leaky ReLUs with slope 0.2 and a SLF were used as their respective activation functions. Each layer has an added bias vector of the same size as the layers output size.

Each layer's training weights of shape (out_features) are initialized from $\mathcal{U}(-\sqrt{k}, \sqrt{k})$, where $k = \frac{1}{\text{in\_features}}$. The bias vector of shape (out_features), has it's values initialized from $\mathcal{U}(-\sqrt{k}, \sqrt{k})$ where $k = \frac{1}{\text{in\_features}}$.

5 Mean Field Network

5.1 Workflow

Figure 9 gives an overview of the training process with the MFN at the center. We already covered feature extraction in section 3 and the MLP in section 4. Section 5.3 will cover the choice of potentials which will justify the need for S_ij (section 5.3.2) and inform the update equation. Section 5.5 is going to derive the update equation. Section 5.8 will motivate the use of the dice loss as loss function.

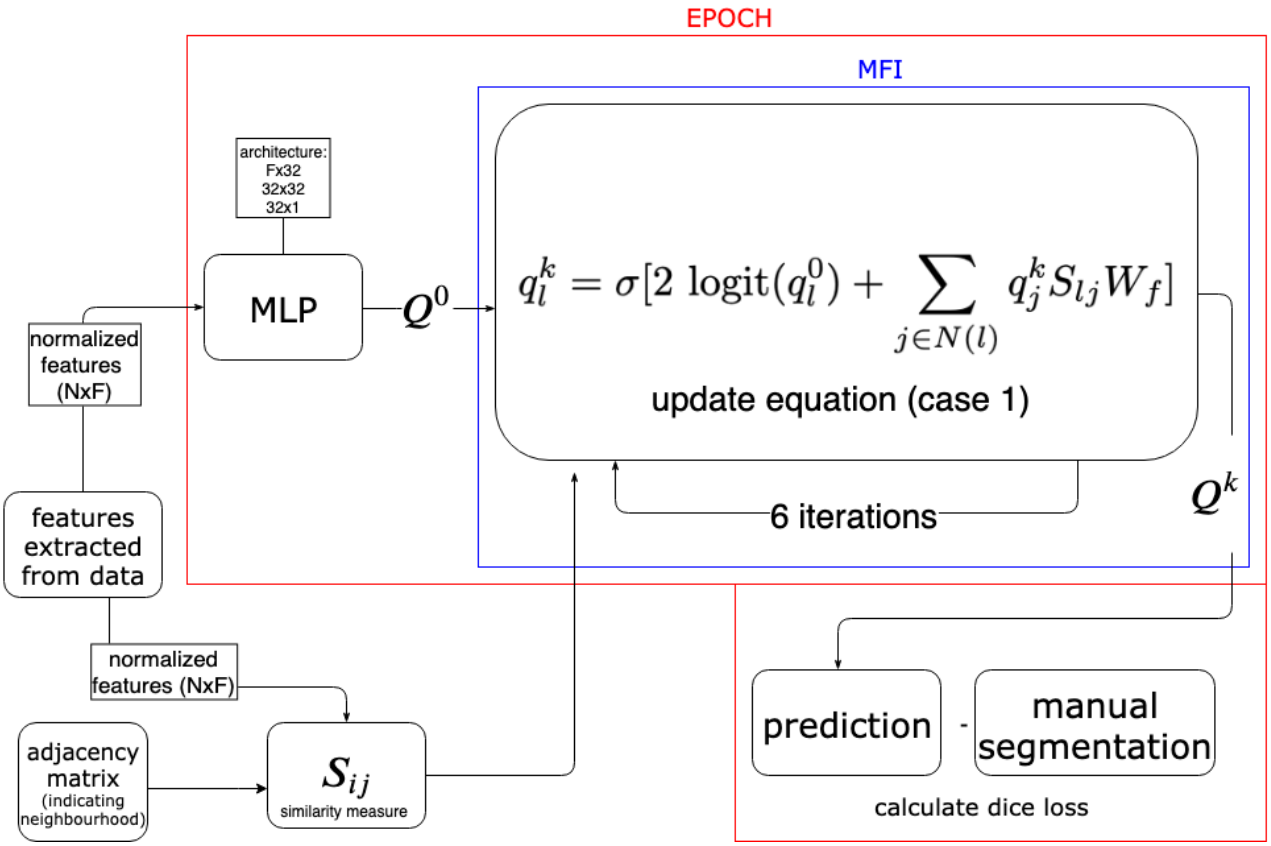


Figure 9: The workflow of training a MFN with case 1 pairwise potential.

In the mean field iterations, only the pairwise potential is updated. The unary potential is given by the MLP. Since MLP and MFI are jointly trained, their weights influence each other. This can be tested empirically by deactivating the MFI and comparing the predictions of the MLP alone with the predictions of the MLP in the MFN.

5.2 Network architecture

The Mean Field iterations can be interpreted as as feed-forward operations in a neural network. Thus, both unary and pairwise potentials form part of a neural network. In our particular case, this networks architecture looks as seen in fig. 10.

```
MFN(
  (fc_layer1): Linear(in_features=30, out_features=32, bias=True)
  (fc_layer2): Linear(in_features=32, out_features=32, bias=True)
  (out_layer): Linear(in_features=32, out_features=1, bias=True)
  (relu): LeakyReLU(negative_slope=0.2)
  (sigmoid): Sigmoid()
  (pwLayer): Linear(in_features=31, out_features=1, bias=True)
)
```

Figure 10: The MFN architecture adds a pairwise potential layer to the MLP architecture seen in fig. 8.

The respective sizes of input layer, first hidden layer, second hidden layer and output layer are:

$$F \times 32 \times 32 \times 1 \times 31 \times 1$$

Like in the MLP, each layer of the MFN is fully connected to the next one, with one trainable parameter per connection. A bias of the size of the sink layer is added. Thus the number of trainable parameters is:

$$(30 * 32) + 32 + (32 * 32) + 32 + (32 * 1) + 1 + (31 * 1) + 1 = 2113$$

This is only 32 more parameters than the MLP.

5.3 Choice of energy potentials

5.3.1 Unary potential

The unary potential is chosen to be a Multilayer perceptron (MLP). Being a neural network, it can be trained together with the MFN via backward propagation and is thus a natural choice for a unary potential. Additionally by running the MLP without message passing, we get a simple baseline which we will later compare to the MFN. To translate the probability output of the MLP into an energy, we make use of the logit function.

$$\mathit{logit} : [0, 1] \to \mathbb{R}.$$

$$\mathit{logit}(p) = \log \frac{p}{1 - p}$$

The unary potential of a feature vector x_i is thus:

$$\psi_u(x_i) = \text{logit}(\text{MLP}(x_i)) = \text{logit}(Q_i^0(x_i)) \tag{28}$$

5.3.2 Pairwise potential

With the pairwise similarity we want to describe a trainable similarity measure for a given pair of pixels, in order to take the context of a pixel into account when predicting it's label. To express the similarity between two feature vectors A and B, a combination of the cosine and the euclidean similarity function is chosen. Yi Li and Wei Ping [19] have shown that cosine similarity can be an effective measure for energy functions in MFNs. The cosine similarity takes the dot product of two vectors and divides it by the product of the vector norms. Dividing by the norms ensures the similarities are in range $[-1, 1]$. $cos(A, B)$ is equal to 1 if A and B are parallel and have the same sense, -1 if they are parallel but opposed and 0 if they stand at a 90 degree angle. We make the

assumption that feature vectors that stand at a small angle to each other indicate that the underlying pair of pixels are similar.

$$\begin{aligned} cos(A, B) &= \frac{A \cdot B}{||A|| \cdot ||B||} \\ &= \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \end{aligned}$$

The euclidean similarity is defined here as $\frac{2}{1+\text{ euc\_dist}} - 1$, which puts it into the same range as cosine similarity: $[-1, 1]$. The motivation for the euclidean similarity $euc(A, B)$ is it's straightforward interpretation as the norm of the distance vector between A and B, a measure which intuitively captures how close the values of A and B are to each other. The euclidean distance is:

$$\text{euc\_dist}(A, B) = \sqrt{\sum_i (A_i - B_i)^2}$$

Since in this model, the euclidean distance is over each single feature dimension, it becomes a distance between scalars, which simplifies to the absolute value of the difference between scalars.

$$\text{euc\_dist}(a, b) = \sqrt{(a-b)^2} \tag{29}$$

$$\text{euc\_dist}(a, b) = |a - b| \quad \text{(absolute value)} \tag{30}$$

These similarities are computed for each edge (i, j). Let the total number of edges in our model be N_e. Using the similarity functions from above, we define a cosine similarity vector over all edges, called $C_{ij} \in \mathbb{R}^{\mathbb{N}_e \times 1}$ and an euclidean similarity matrix $E_{ij} \in \mathbb{R}^{N_e \times F}$, which are stacked into a single matrix $S_{ij} \in \mathbb{R}^{N_e \times (F+1)}$ for joined learning by the network.

$$S_{ij} = \begin{pmatrix} cos(x_i, x_j) \\ euc(x_i, x_j) \end{pmatrix} = \begin{pmatrix} C_{ij} \\ E_{ij} \end{pmatrix} \in \mathbb{R}^{F+1}$$

This means that the similarity matrix $S \in \mathbb{R}^{E \times F+1}$ between two pixels i and j contains the norm of the difference between each feature dimension of $x_i \in \mathbb{R}^F$ and $x_j \in \mathbb{R}^F$ as well as the cosine similarity between the entire feature vectors $cos(x_i, x_j) \in \mathbb{R}$. The similarities over the entire CRF are thus:

$$S = \begin{pmatrix} C \\ E \end{pmatrix} \in \mathbb{R}^{N_e \times (F+1)}$$

S is scaled by a trainable weight vector $W_f \in \mathbb{R}^{(F+1)times1}$ (fig. 3) and a bias term $b_f \in \mathbb{R}^{F+1}$ is added. All entries of b_f are identical. This makes the number of trainable parameters equal to:

$$31 + 1 \text{ (weights matrix } W_f + \text{ bias } b_f)$$

All parameters are initialized according to a uniform distribution, the same as the MLP layers. Additionally, an indicator function $\mathbb{1}(y_i, y_j)$ determines - based on the pixel labels - whether or not the similarity is accounted for. We will test three different indicator functions for this model, dubbed case 1, case 2 and case 3. Finally this defines the pairwise potential as:

$$\psi_p(x_i, x_j) = \mathbb{1}(y_i, y_j) \cdot S_{ij} \cdot W_f$$

During message passing, the similarity function determines which messages are passed while the indicator function defines how the messages are passed. In the following, three different conditions for

message passing are explored, dubbed case 1, 2 and 3 for simplicity.

Because the similarities are known in advance, they are computed before the model is engaged. Thus the mean field combines 3 things: the preliminary probability given by the unary potential, the static similarity information S_{ij} that was precomputed and the trainable matrix W_f that will weight every connection individually. Note that W_f is updated by back-propagation at the end of each image that is passing through the network (a batchsize of 1 is used) but not throughout the message passing steps, which will all make use of the same W_f for a given epoch and image.

Now that we fully defined the pairwise potential, we can calculate the ELBO. **??** requires us to determine the expectation of the log joint probability, which is equal to the energy function. We thus need to determine the expected value of $E(X, Y)$ for each case of pairwise potential.

Case 1: Rewarding message passing: vessel agreement

The energies of cases where the labels are both vessel (they 'agree' on vessel) are added. Thus the expectation of the pairwise potential becomes:

$$\begin{aligned}
\mathbb{E}_Q\left[\psi_p(y_i, y_j)\right] &= \sum_i \sum_{j \in N_i} \mathbb{E}_Q\left[\mathbb{1}\left[y_i = y_j = 1\right]\right] \cdot S_{ij} W_f \qquad \text{(since } S_{ij} \text{ and } W_f \text{ are independent of } Y\text{)} \\
&= \sum_i \sum_{j \in N_i} Q_i^k(y_i = 1) Q_j^k(y_j = 1) \cdot S_{ij} W_f \qquad \text{(}\mathbb{E}\left[\mathbb{1}\right] \text{ is 0 for all terms where } y_i = 0\text{)} \\
&= \sum_i \sum_{j \in N_i} q_i^k q_j^k \cdot S_{ij} W_f \qquad (31)
\end{aligned}$$

Case 2: Rewarding message passing: vessel and background agreement

The energies of all cases in which a pairs of neighbouring pixels whose labels correspond to each other (i.e. 'agree') are added. This covers the case where both pixels have the label 'vessel' and the one where both are 'background'. Thus, the expectation of the pairwise potential becomes:

$$\begin{aligned}
\mathbb{E}_Q\left[\sum_i \sum_{j \in N_i} \psi_p(y_i, y_j)\right] &= \sum_i \sum_{j \in N_i} \mathbb{E}_Q\left[\mathbb{1}\left[y_i = y_j = 1\right]\right] \cdot S_{ij} W_f + \mathbb{1}\left[y_i = y_j = 0\right] \cdot S_{ij} W_f \qquad (32) \\
&= \sum_i \sum_{j \in N_i} \Big[Q_i^k(y_i = 1) Q_j^k(y_j = 1) + Q_i^k(y_i = 0) Q_i^k(y_i = 0) \Big] \cdot S_{ij} W_f \qquad (33) \\
&= \sum_i \sum_{j \in N_i} \Big[q_i^k q_j^k + (1 - q_i^k)(1 - q_j^k) \Big] \cdot S_{ij} W_f \qquad (34)
\end{aligned}$$

Case 3: Rewarding and punishing potential

We add the energies of cases where the labels agree, and substract in cases where they disagree. This is expressed by 4 indicator functions, covering all possible combinations of labels for the pixel pair. Thus the expectation of the pairwise potential becomes:

$$\begin{aligned}
\mathbb{E}_Q\left[\sum_i \sum_{j \in N_i} \psi_p(y_i, y_j)\right] &= \sum_i \sum_{j \in N_i} \mathbb{E}_Q[\mathbb{1}\left[y_i = y_j = 1\right]] \cdot S_{ij} W_f \qquad (35) \\
&+ \mathbb{E}_Q[\mathbb{1}\left[y_i = y_j = 0\right]] \cdot S_{ij} W_f \qquad (36) \\
&- \mathbb{E}_Q[\mathbb{1}\left[y_i = 1, y_j = 0\right]] \cdot S_{ij} W_f \qquad (37) \\
&- \mathbb{E}_Q[\mathbb{1}\left[y_i = 0, y_j = 1\right]] \cdot S_{ij} W_f \qquad (38)
\end{aligned}$$

Solving the expectation over the binary indicator function yields:

$$\mathbb{E}_Q\left[\sum_i \sum_{j\in N_i} \psi_p(y_i, y_j)\right] = \sum_i \sum_{j\in N_i} Q^k(y_i=1)Q^k(y_i=1)\cdot S_{ij}W_f \tag{39}$$

$$+ Q^k(y_i=0)Q^k(y_i=0)\cdot S_{ij}W_f \tag{40}$$

$$- Q^k(y_i=1)Q^k(y_i=0)\cdot S_{ij}W_f \tag{41}$$

$$- Q^k(y_i=0)Q^k(y_i=1)S_{ij}W_f \tag{42}$$

$$= \sum_i \sum_{j\in N_i} \left[q_i^k q_j^k + (1-q_i^k)(1-q_j^k) - q_i^k(1-q_j^k) - (1-q_i^k)q_j^k\right]\cdot S_{ij}W_f \tag{43}$$

5.4 Specific Derivation of ELBO

Central to the design of a MFN are the derivation of the ELBO and the update equation, which are specific to the choice of energy function. The ELBO can be derived for the respective energy potentials by inserting them into the equation eq. (17).
The expectation of the pairwise potential $\psi_p(y_i, y_j)$ will depend on the chosen case. Thus we are going to derive the ELBO for the 3 cases below.

5.4.1 Case 1

For each pixel i, it's energy function $E(X,Y)$ gets updated by summing over the neighbours of i: $\sum_j [\psi_p(x_i, x_j)]$. Based on the definition of ELBO (eq. (17)) using the Q^k notation this time, to highlight the use of the latest guess of distribution. Note that Q is the distribution over all pixels $Q = Q_{1\dots N}$, whereas Q_i denotes the distribution over pixel i.

$$\begin{aligned}
ELBO &= \mathbb{E}_{Q^k(Y)}\left[\log P(X,Y)\right] + H_Q(Y) \\
ELBO &= \underbrace{\mathbb{E}_{Q^k(Y)}\left[E(X,Y)\right]}_{*} \underbrace{-\mathbb{E}_{Q^k(Y)}\left[\log Q(Y)\right]}_{**} \\
&= q_i^k \ \mathrm{logit}(q_i^0) + (1-q_i^k)\ \mathrm{logit}(1-q_i^0) \\
&\quad + q_i^k q_j^k \cdot S_{ij}W_f \\
&\quad - \left[q_i^k \log q_i^k + (1-q_i^k)\log(1-q_i^k)\right]
\end{aligned}$$

$$\begin{aligned}
* &= \mathbb{E}_{Q^k}\left[\sum_i \left[\psi_u(y_i) + \sum_{j\in\mathbb{N}_i}\psi_p(y_i,y_j)\right]\right] && \text{(definition of energy function, eq. (27))} \\
&= \sum_i \left[\mathbb{E}_{Q_i^k}\left[\psi_u(y_i)\right] + \sum_{j\in\mathbb{N}_i}\mathbb{E}_{Q_{ij}^k}\left[\psi_p(y_i,y_j)\right]\right] && \text{(linearity of expectation)} \\
&= \sum_i \left[\mathbb{E}_{Q_i^k}\left[\mathrm{logit}(Q_i^0)\right] + \sum_{j\in\mathbb{N}_i} q_i^k q_j^k S_{ij}W_f\right] && \text{(using eq. (28) and eq. (31))} \\
&= q_i^k \mathrm{logit}(q_i^0) + (1-q_i^k)\mathrm{logit}(1-q_i^0) + \sum_{j\in\mathbb{N}_i} q_i^k q_j^k S_{ij}W_f
\end{aligned}$$

$$
\begin{aligned}
** &= -\sum_{y\in\{0,1\}} Q^k(Y=y)\log Q^k(Y=y) \\
&= -\Big[Q^k(Y=1)\log Q^k(Y=1) + Q^k(Y=0)\log Q^k(Y=0)\Big] \\
&= -\Big[q_i^k \log q_i^k + (1-q_i^k)\log(1-q_i^k)\Big] \qquad \text{(see equation 25)}
\end{aligned}
$$

5.4.2 Case 2

When adding the energies of pixels that agree to be background, we need to change $\mathbb{E}_Q \sum_i \sum_j \psi_p(x_i, x_j)$ as seen in section 5.3.2. We obtain thus:

$$
\begin{aligned}
ELBO &= \mathbb{E}_{Q^k(Y)}\left[E(X,Y)\right] + H_Q(Y) \\
&= q_i^k \ \text{logit}(q_i^0) + (1-q_i^k)\ \text{logit}(1-q_i^0) \\
&\quad + [q_i^k q_j^k + (1-q_i^k)(1-q_j^k)]\cdot S_{ij}W_f \\
&\quad - [q_i^k \log q_i^k + (1-q_i^k)\log(1-q_i^k)]
\end{aligned}
$$

5.4.3 Case 3

And by substracting the energies of the "disagreeing" pixels:

$$
\begin{aligned}
ELBO &= \mathbb{E}_{Q^k(Y)}\left[E(X,Y)\right] + H_Q(Y) \\
&= q_i^k \ \text{logit}(q_i^0) + (1-q_i^k)\ \text{logit}(1-q_i^0) \\
&\quad + [q_i^k q_j^k + (1-q_i^k)(1-q_j^k) - q_i^k(1-q_j^k) - (1-q_i^k)(1-q_j^k)]\cdot S_{ij}W_f \\
&\quad - [q_i^k \log q_i^k + (1-q_i^k)\log(1-q_i^k)]
\end{aligned}
$$

5.5 Specific Derivation of update equation

To achieve the goal of maximizing the ELBO, MFNs update the approximate distribution Q iteratively until convergence to a local optimum. In the following, the update equation achieving this optimization is derived based on the model parameters and the mean field assumptions.

The goal is to find the probability Q_l of a given pixel l that maximizes ELBO, i.e. $Q_l = \underset{Q}{\text{argmax}}\ \text{ELBO}(Q)$ as seen in eq. (23). Because we are looking for a maximum of the ELBO function, we set the derivative to 0.

5.5.1 Case 1

$$\frac{\partial ELBO}{\partial q_l^k} = \text{logit}(q_l^0) - \text{logit}(1-q_l^0) + \sum_{j \in N(l)} q_j^k S_{lj} W_f - [1 + \log q_l^k] = 0 \tag{44}$$

$$\log(q_l^k) - \log(1-q_l^k) = \log(\frac{q_l^0}{1-q_l^0}) - \log(\frac{1-q_l^0}{q_l^0}) + \sum_{j \in N(l)} q_j^k S_{lj} W_f]$$

(move all q_l^k terms to the left hand side)

$$\log \frac{q_l^k}{(1-q_l^k)} = [\log(\frac{q_l^0}{1-q_l^0}) - \log(\frac{1-q_l^0}{q_l^0}) + \sum_{j \in N(l)} q_j^k S_{lj} W_f] \tag{45}$$

$$q_l^k = \sigma[\log(\frac{q_l^0}{1-q_l^0}) - \log(\frac{1-q_l^0}{q_l^0}) + \sum_{j \in N(l)} q_j^k S_{lj} W_f]$$

(since σ is the inverse function of logit)

$$= \sigma[\text{logit}(q_l^0) - \text{logit}(1-q_l^0) + \sum_{j \in N(l)} q_j^k S_{lj} W_f] \tag{46}$$

$$= \sigma[\text{logit}(q_l^0) + \text{logit}(q_l^0) + \sum_{j \in N(l)} q_j^k S_{lj} W_f] \tag{47}$$

$$q_l^k = \sigma[2\ \text{logit}(q_l^0) + \sum_{j \in N(l)} q_j^k S_{lj} W_f] \tag{48}$$

Thus the update equation for case 1 is:

$$q_l^k = \sigma[2\ \text{logit}(q_l^0) + \sum_{j \in N(l)} q_j^k S_{lj} W_f] \tag{49}$$

5.5.2 Case 2

$$\frac{\partial ELBO}{\partial q_l^k} = \text{logit}(q_l^0) - \text{logit}(1-q_l^0) + \sum_{j \in N(l)} q_j^k - (1-q_j^k) S_{lj} W_f - [1 + \log q_l^k] = 0$$

$$= \text{logit}(q_l^0) - \text{logit}(1-q_l^0) + \sum_{j \in N(l)} (2q_j^k - 1) S_{lj} W_f - [1 + \log q_l^k] = 0$$

$$q_l = \sigma[2\ \text{logit}(q_l^0) + \sum_{j \in N(l)} (2q_j^k - 1) S_{lj} W_f] \quad \text{(the rest of the steps are identical to eq. (48))}$$

5.5.3 Case 3

$$\frac{\partial ELBO}{\partial q_l^k} = \text{logit}(q_l^0) - \text{logit}(1-q_l^0) + \sum_{j \in N(l)} q_j^k - (1-q_j^k) S_{lj} W_f - [1 + \log q_l^k] = 0$$

$$= \text{logit}(q_l^0) - \text{logit}(1-q_l^0) + \sum_{j \in N(l)} (4q_j^k - 2) S_{lj} W_f - [1 + \log q_l^k] = 0$$

$$q_l^k = \sigma[2\ \text{logit}(q_l^0) + \sum_{j \in N(l)} (4q_j^k - 2) S_{lj} W_f] \quad \text{(the rest of the steps are identical to eq. (48))}$$

5.6 Hyperparameters and learning metrics

Hyperparameters denote all those parameters of the model that are not trained by back propagation, but chosen by the data scientist. Through using a subset of the training data (the validation set), we can determine which parameters work well on this set and thus likely also generalize to the test set, provided we do not overfit.

In this case, the hyperparameters include:

- learning rate = 0.005
- batchsize = 1
- mean field iterations (number of message passing steps) = 6 for MFN, 0 for baseline MLP
- indicator function/condition for message passing (case 1, 2 and 3)
- optimizer = Adam
- loss function = dice-loss

Message passing was split into 3 different cases as described in section 5.3.2.

5.7 Adam

The learning rate was set empirically based on the performance of the network. For the MFN models, slight changes in learning rate did often lead to the training loss getting stuck. The Adam optimizer (name derived from adaptive moment estimation) [20] uses derivatives to find the local minimum of the loss function, similar to stochastic gradient descent (SGD). Unlike SGD, it uses partial derivatives of each parameter independently instead of using gradients.

5.8 Dice loss

The dice score or F1-score is a similarity measure between predicted values and true values. It corresponds to the harmonic mean between precision and recall. As a reminder, precision (Pr) and recall (Re) are defined as follows:

$$Pr = \frac{TP}{TP + FP}$$
$$Re = \frac{TP}{TP + FN}$$

(Note: the recall is also sometimes called sensitivity).

Which makes the dice score:

$$\text{Dice score} = \frac{2}{\frac{1}{Pr} + \frac{1}{Re}} \tag{50}$$
$$= \frac{2}{\frac{TP+FP}{TP} + \frac{TP+FN}{TP}} \tag{51}$$
$$= \frac{2TP}{2TP + FP + FN} \tag{52}$$

The dice score also has an interpretation in set theory. If we define A to be the set containing the predicted elements (in our case the pixels) and B to be the true set of elements, then $|A \cap B| = TP$. Further, $|A| = TP + FP$ and $|B| = TP + FN$.

Let A be the predicted segmentation and B be the manual segmentation, i.e. the ground truth. The dice-score is defined as two times the number of pixels that are in both sets divided by the sum of the set sizes.

$$\frac{2|A \cap B|}{|A| + |B|}$$

In the case of binary classification:

$$\frac{2TP}{2TP + FP + FN}$$

The dice-loss is simply $1 -$ dice similarity and both dice-score and dice-loss range between $[0, 1]$.

When using binary vectors a and $b \in 0, 1^N$ instead of sets, where 0 encodes background and 1 encodes vessel, the dice-score can be rewritten as follows.

$$\text{dice score} = \frac{2|A \cap B|}{|A| + |B|} = \frac{2\sum_{i=1}^{N} \mathbf{a_i} \cdot \mathbf{b_i}}{\left(\sum_{i=1}^{N} \mathbf{a_i}\right) + \left(\sum_{i=1}^{N} \mathbf{b_i}\right)}$$

Where $\cdot$ is the inner vector product / scalar product. The interesting thing about this vectorized form of the dice score is that it also works for real-valued vectors. Hence, this formula can be implemented for vectors containing probabilities, such as the ones we are dealing with. Note that not only the predictions, but also the ground truth contains probabilities due to the interpolation from resizing the images.

6 Results

In this section, the results of training MLP and the MFN case 1, 2 and 3 are presented. The final scoring metrics on the test set can be seen in table 1. All hyperparameters were set as described in section 5.6 and are the same for all 4 models, except for the number of epochs, which was 1001 for MLP and 201 for the MFNs.

The performance metrics chosen for evaluating success on the test set are accuracy (Acc), sensitivity (Se) and specificity (Sp), since they are the most frequently used metrics in the vessel segmentation literature [7].

$$Acc = \frac{\text{number of correct prediction}}{\text{total number of predictions}} = \frac{TP + TN}{TP + TN + FP + FN} \tag{53}$$

$$Se = \frac{TP}{TP + FN} \tag{54}$$

$$Sp = \frac{TN}{TN + FP} \tag{55}$$

6.1 MLP

The results of the MLP serve as a baseline to the MFN experiments. Since MFN use the MLP as unary potential, they should perform at least as good as the stand-alone MLP. Figure 11 shows that the training loss converged to 0.3 after about 570 epochs. The validation loss could not break 0.38, indicating some slight overfitting.

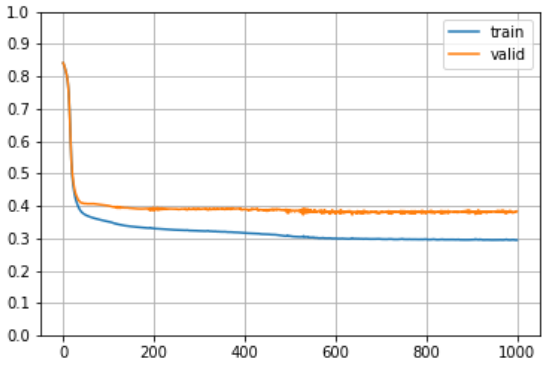
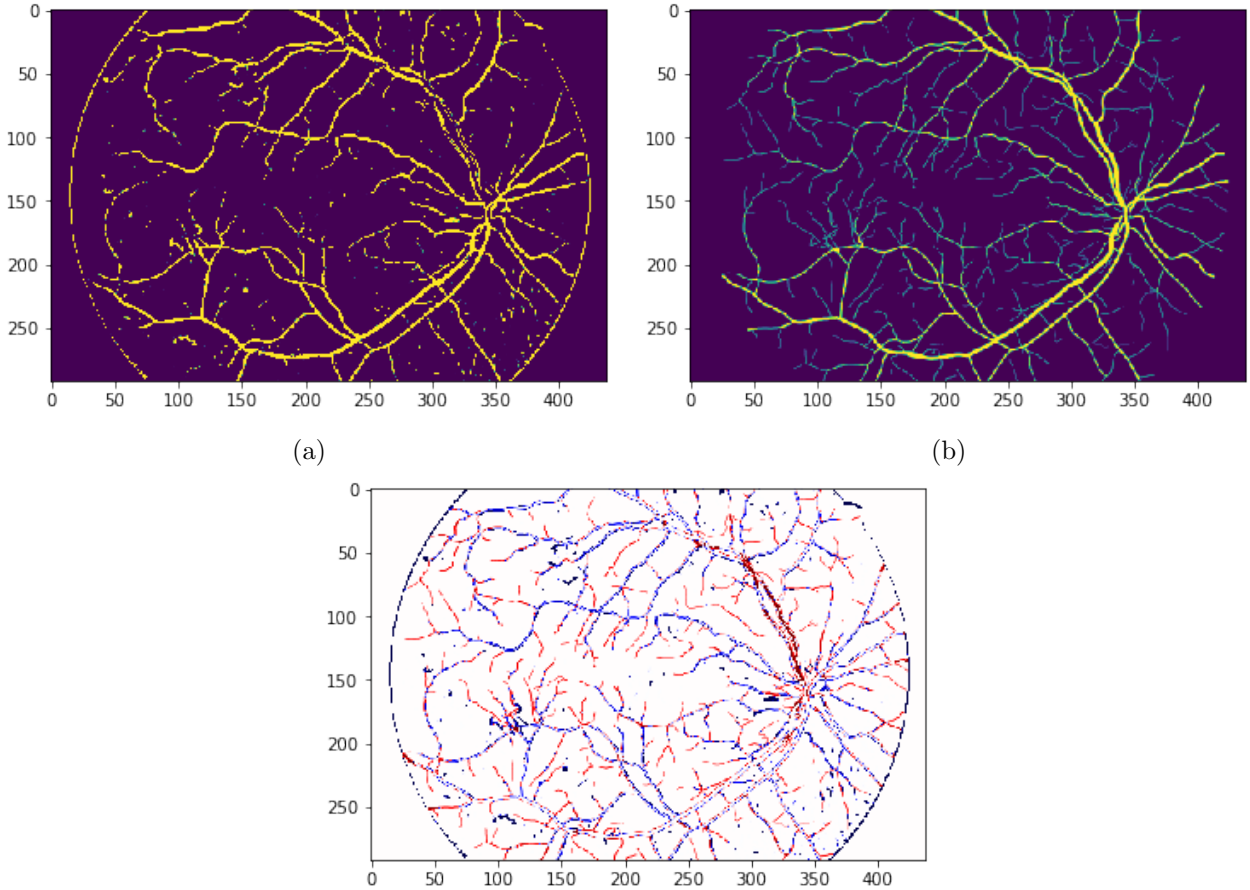


Figure 11: MLP. Loss plot for 1001 epochs.

Figure 12a shows the prediction of the MLP - trained for 1001 epochs - on an image from the test set. The thick vessels can all be observed in the prediction although the sometimes appear discontinuous, as can be seen in the split vessels above the fovea. The thinner vessels are often not predicted or appear discontinuous. The border of the FoV is misinterpreted as vessel, possibly due to some misleading information in the gradient features of pixels at the border.

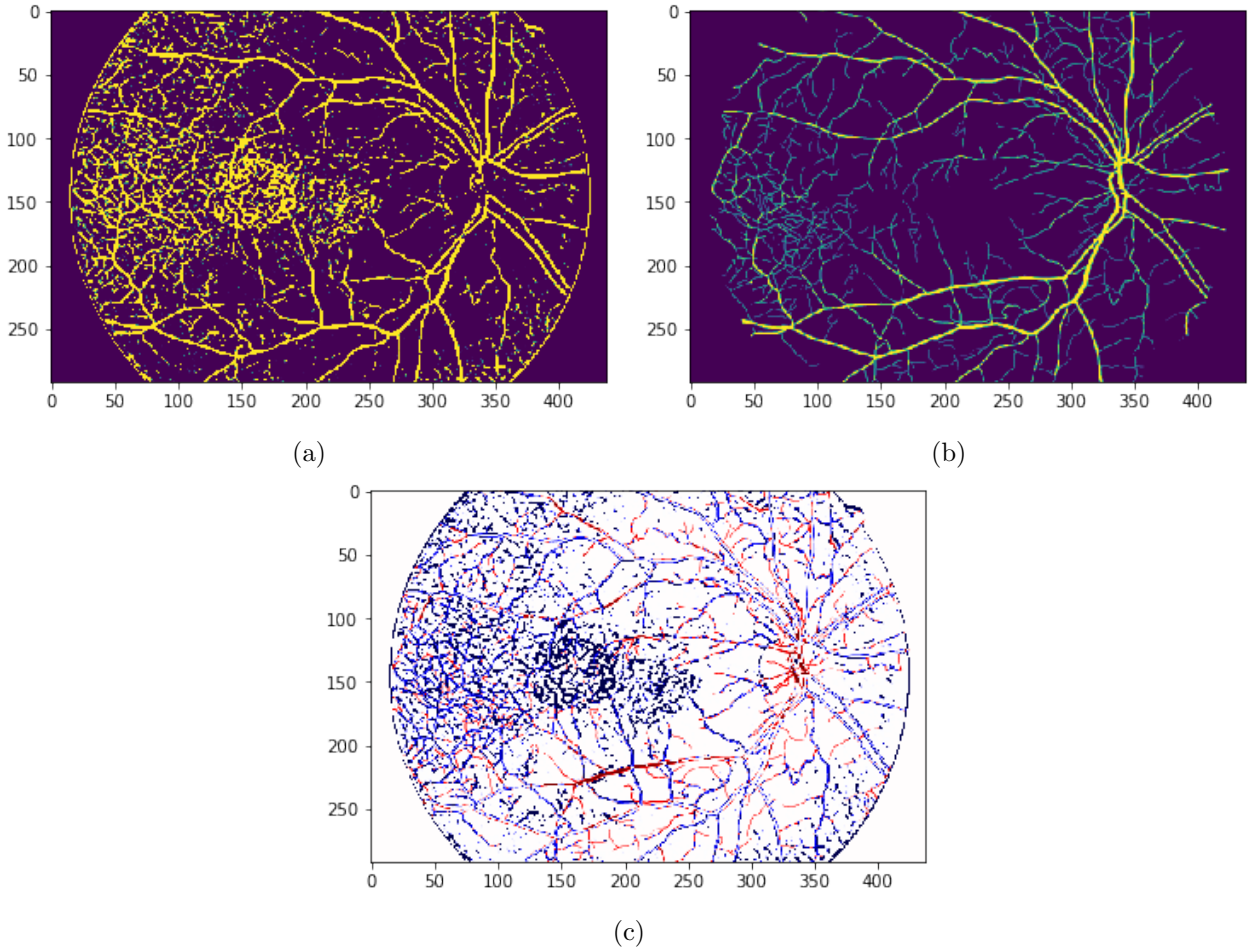


(a) (b)

(c)

Figure 12: a) MLP prediction on test image after 1001 epochs of training. b) Corresponding ground truth image. c) Differences: red = false negative, blue = false positive. This image's prediction has a sensitivity of 78.73% and a specificity of 96.17%

For pictures with a lot of small vessels, the error rate of the MLP is higher . Figure 13b shows a fundus image of a patient with diabetic retinopathy. The unary potential is not able to pick up on this finesse and vastly overpredicts vessels. The high number of false positives (blue in Figure 13c) gives a (relatively) low specificity of 86.19%. This highlights the shortcomings of the baseline model.



(a) (b)

(c)

Figure 13: a) MLP prediction on test image after 1001 epochs of training. b) Corresponding ground truth image. c) Differences: red = false negative, blue = false positive. This images prediction has a sensitivity of 80.05% and a specificity of 86.19%

6.2 MFN

Since the mean-field network extends the MLP model (which updates the pixels prediction based on their own features), by also taking information of neighbouring pixels into account via the pairwise potential, the MFN models are expected to outperform MLP. Furthermore, using more inclusive indicator functions, which draw information from different combinations of labels, is further expected to improve predictions.

The MFN models 1, 2 and 3 were each trained for 201 epochs.

6.2.1 MFN: Case 1

Figure 14 shows the training and validation loss over 201 epochs. Similarly to the MLP loss plot, neither training nor validation dice-loss could reach a value below 0.3. Here, the validation loss is almost 10% lower than for the MLP. Surprisingly, the validation loss is even lower than the training loss which is a phenomenon occurring for all 3 MFN cases (see section 6.2.2 and section 6.2.2). This

could happen by chance and is not a finding that was replicated when changing the learning rate or the weight initialization.

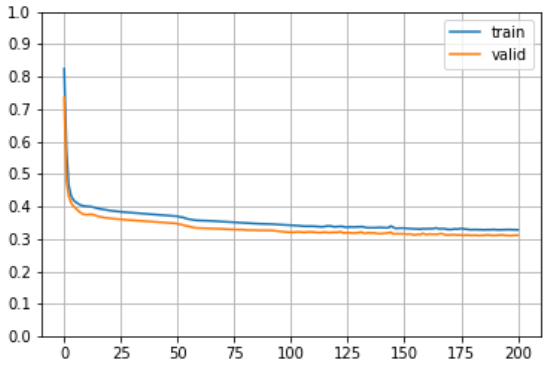


Figure 14: Case 1. Loss plot for 201 epochs

The ELBO is expected to increase across epochs and across message passing iterations until convergence. As seen before, the ELBO is defined as: $ELBO = \mathbb{E}[\psi_u] + \mathbb{E}[\psi_p] + H$. Optimization guarantees that the ELBO increases, except in the case where an overshoot happens where the algorithm follows the gradient such that it passes the local optimum. The ELBO will increase both from one epoch to the next (due to back propagation) as well as from one message passing to the next. Measuring $\mathbb{E}[\psi_u]$ indicates how strong the role of the unary potential was in the optimization, and $\mathbb{E}[\psi_p]$ does so for the pairwise potential. H is expected to decrease as it measures the amount of information gained by further back propagation and message passing steps. In 15, the ELBO is plotted every 60 epochs. It can be seen that optimization is driven by the MLP ($\mathbb{E}[\psi_u]$), as well as the message passing $\mathbb{E}[\psi_p]$. Message passing has it's largest effect at the second iteration.

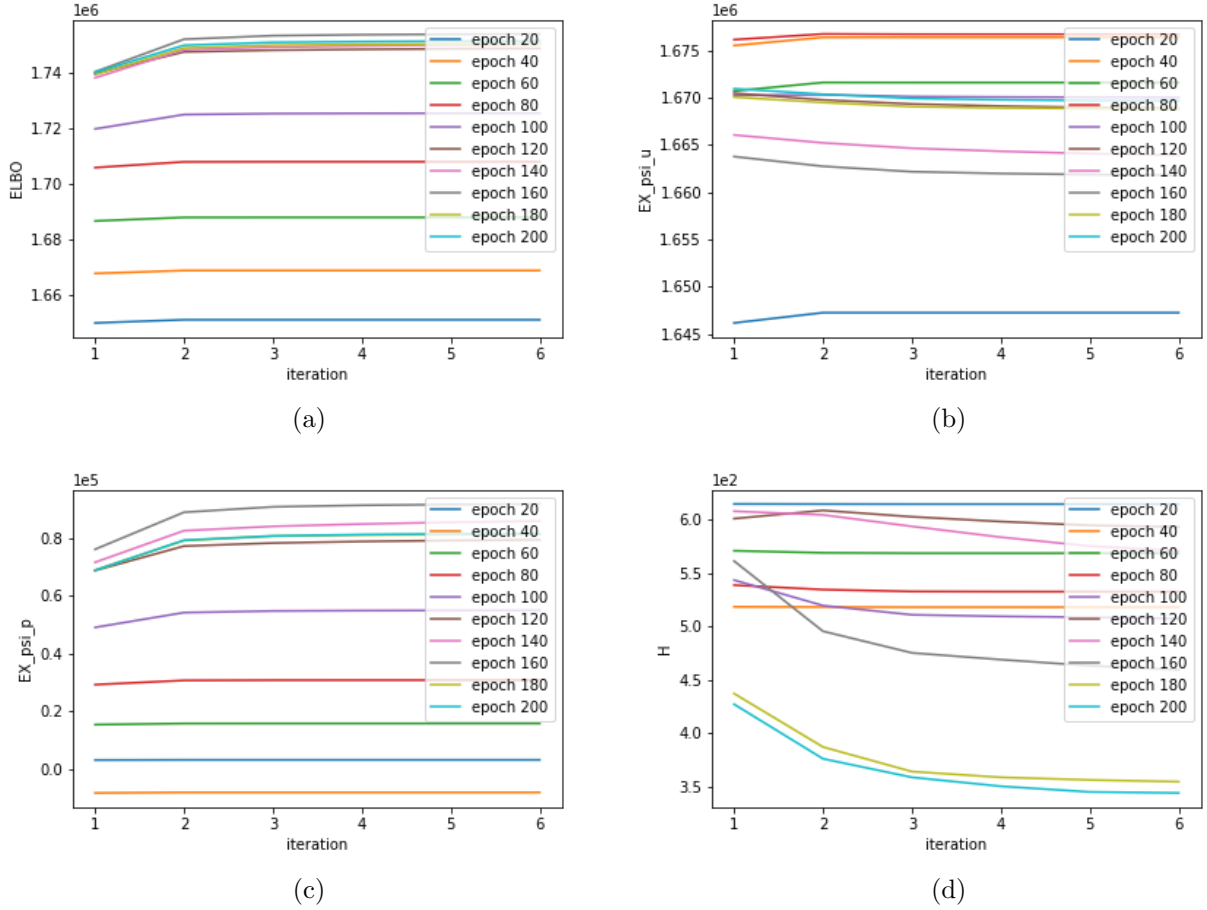


Figure 15: Case 1. $ELBO(a) = \mathbb{E}[\psi_u](b) + \mathbb{E}[\psi_p](b) + H(c)$. The ELBO is increasing over both mean field iterations and epochs, as is expected. The few exceptions to this rule happen after 120 epochs, when the model is nearly converged. The first 3 message passing steps are the most effective. After about 120 epochs (brown line) very little is to be gained from further epochs, which is confirmed by the loss plot fig. 14

Taking a look at the weight vector W_f (fig. 16) reveals that the most crucial feature for prediction were #3 and #27 - intensity and hessian at $sigma = 4$.

Figure 17 shows the prediction of the MFN case 1 model. Both Se (80.26%% against 78.73%) and Sp (97.27% against 96.17%) exceed the one of the MLP on this image.

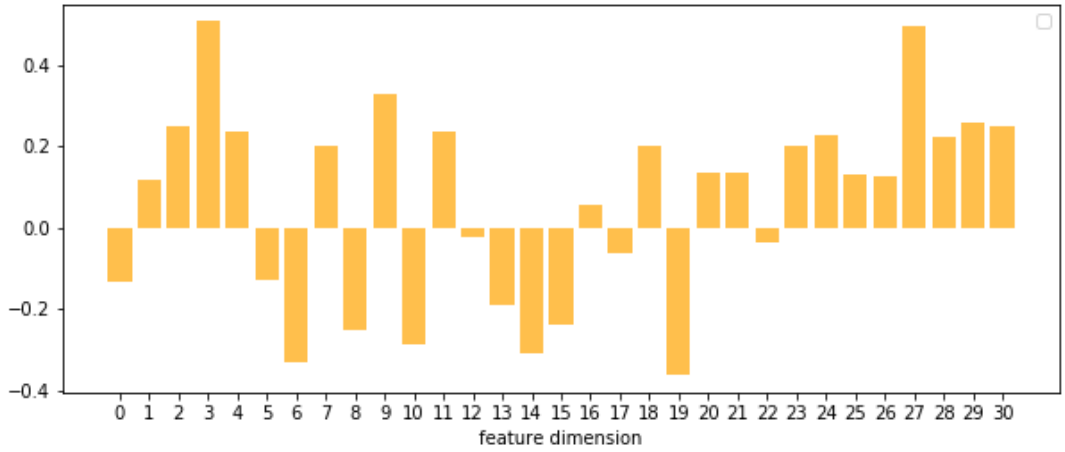
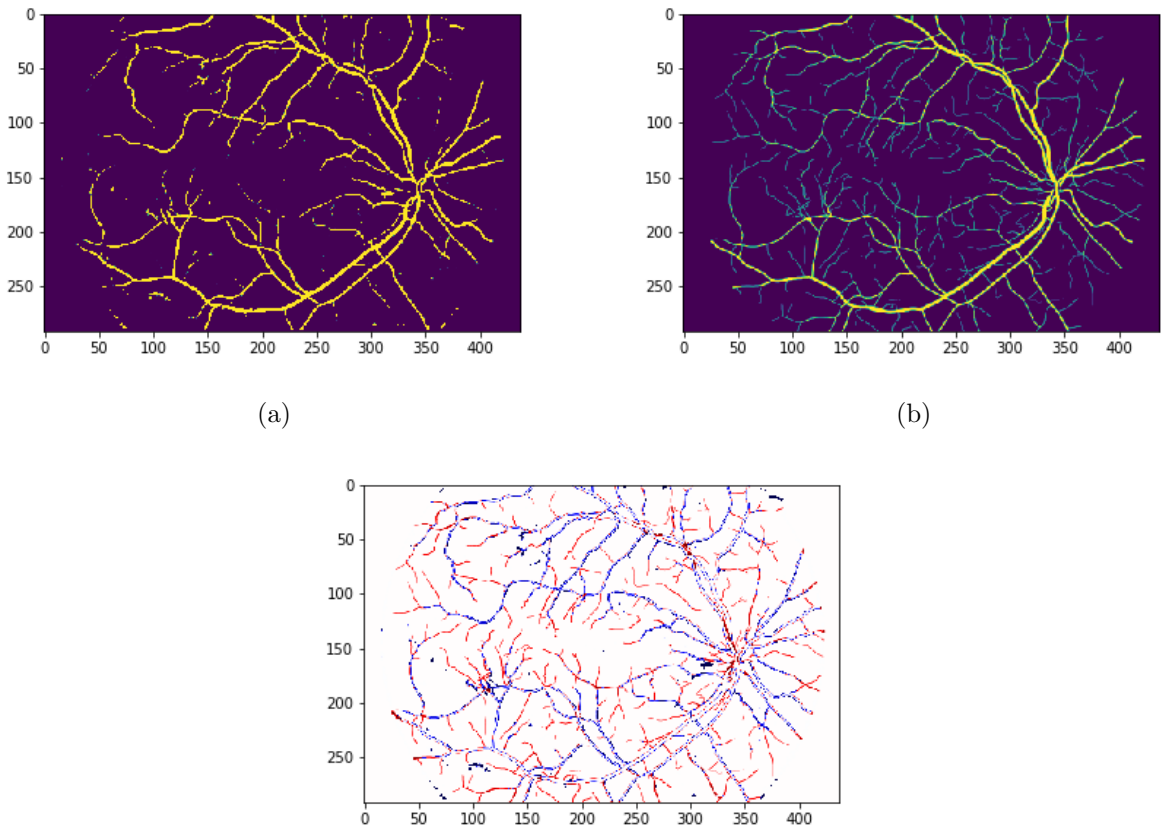


Figure 16: Case 1. Learned entries of the weight vector W_f. The x-axis indicates the feature. #0 = cosine similarity, #1 − #30: features as described in fig. 7.



(a)

(b)

(c)

Figure 17: Case1. a) Prediction on test image after 201 epochs of training. b) Corresponding ground truth image. c) Differences: red = false negative, blue = false positive. This images prediction has a Se of 80.26% and a Sp of 97.27%

6.2.2 MFN: Case 2

The loss plot section 6.2.2 is very similar to the one for case 1. The ELBO plots show the typical increase over epochs and iterations. $\mathbb{E}[\psi_u]$ is strongly driving the ELBO to up, while $\mathbb{E}[\psi_u]$ contributes less and less over later epochs.

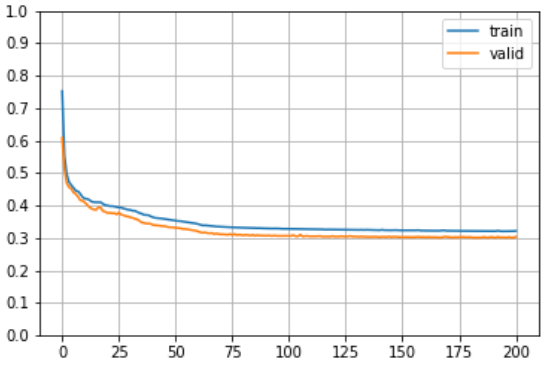


Figure 18: Case 2. Loss plot for 201 epochs, strongly resembling the one for case 1.

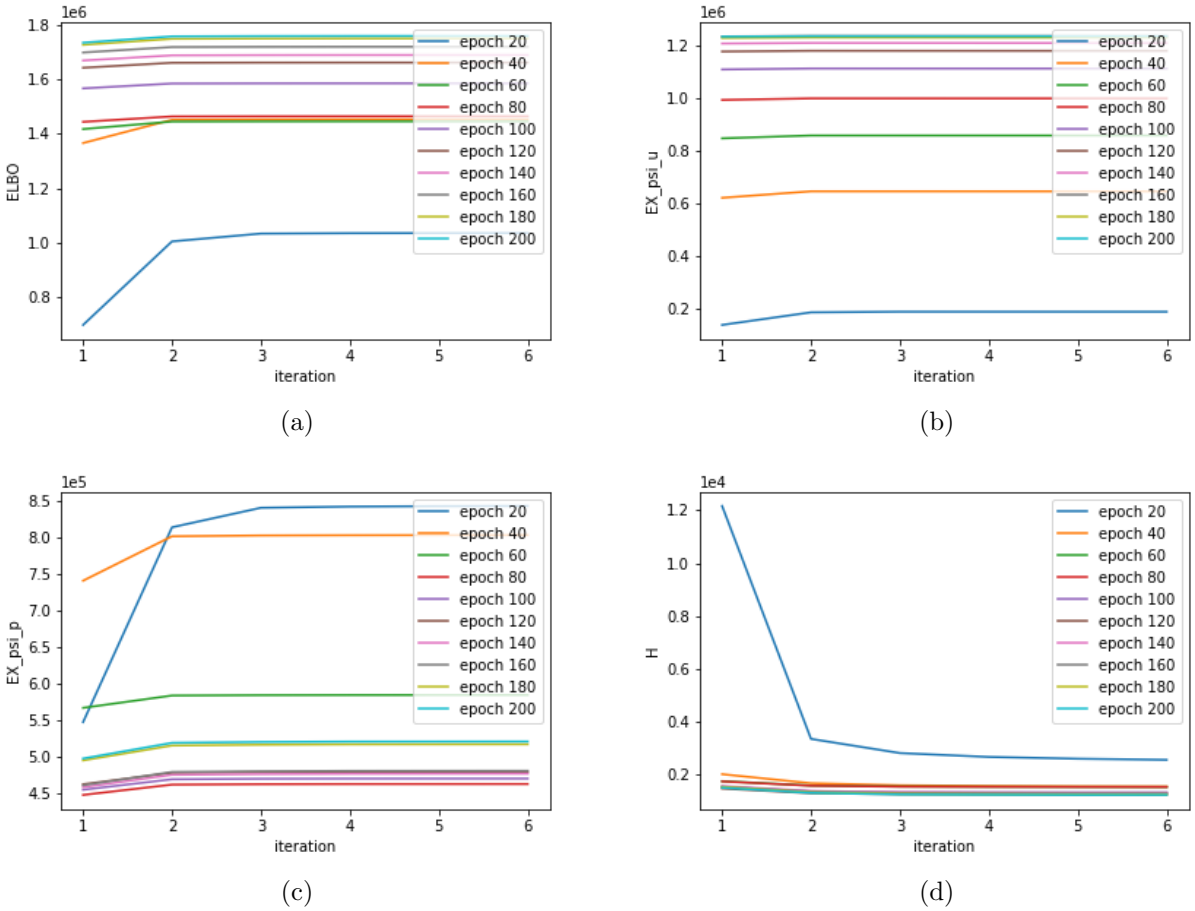


Figure 19: Case 2. $ELBO(a) = \mathbb{E}[\psi_u](b) + \mathbb{E}[\psi_p](b) + H(c)$. The ELBO increases and the entropy decreases as expected. $\mathbb{E}[\psi_u]$ is more effective than $\mathbb{E}[\psi_p]$ in optimizing the ELBO.

For case 2, the weight for intensity strongly exceeds all other feature weights fig. 20.

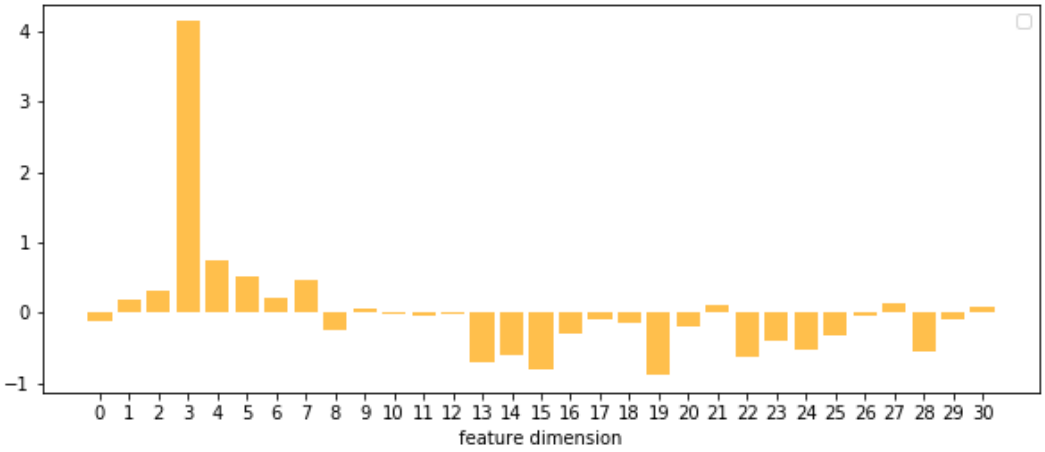


Figure 20: Case 2. Learned weights per feature dimension (W_f) of the pairwise potential. #0 = cosine similarity, #1 − #30: features as described in fig. 6.

6.2.3 MFN: Case 3

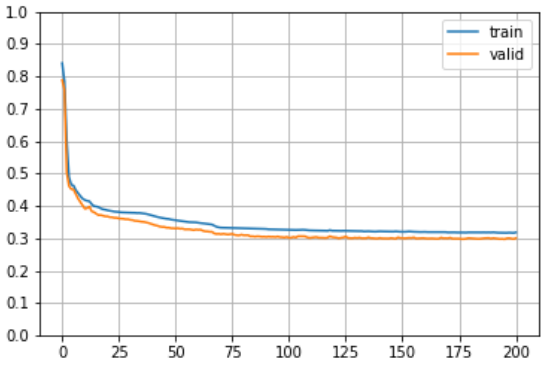


Figure 21: Case 3. Loss plot over 201 epochs, strongly resembling the ones for case 1 and 2.

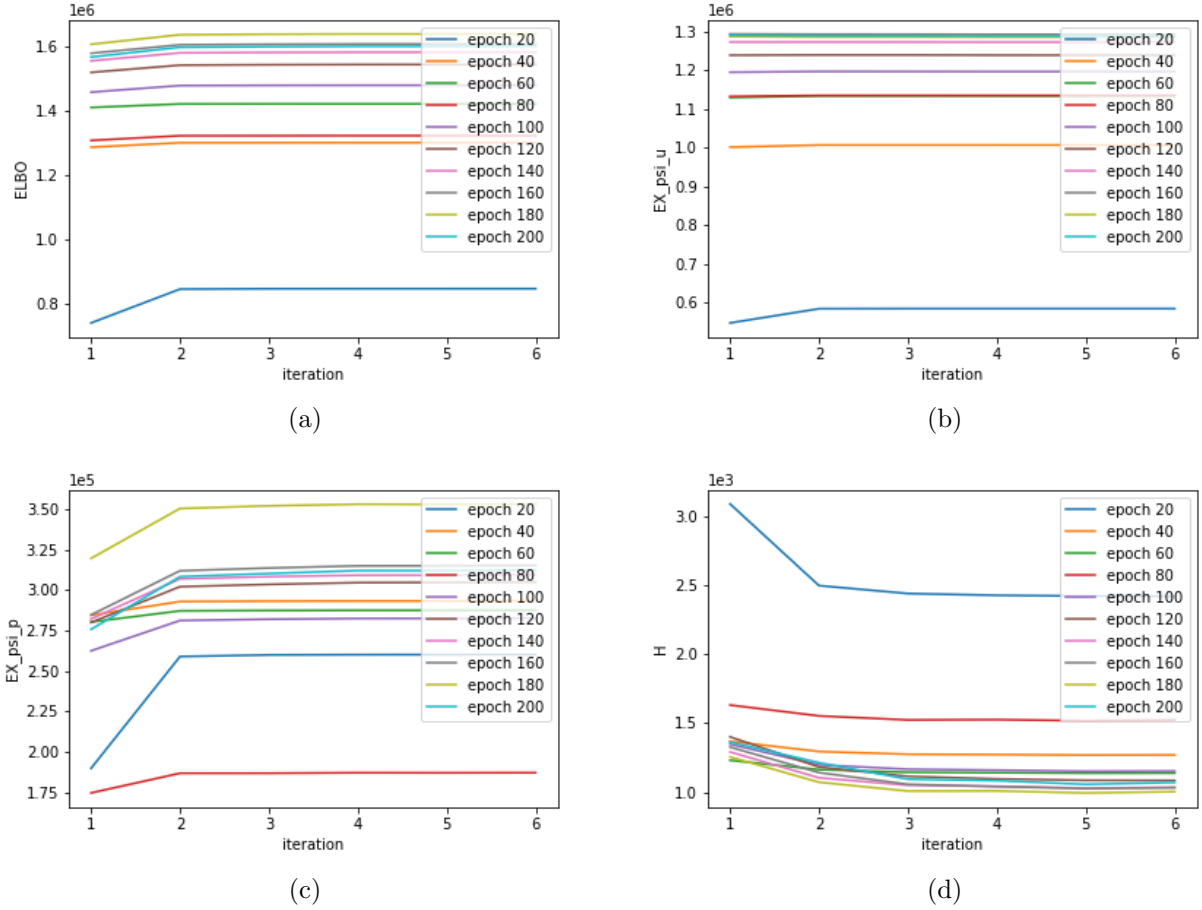


Figure 22: Case 3. $ELBO = \mathbb{E}[\psi_u] + \mathbb{E}[\psi_p] + H$. Plot of ELBO. Having message passing between all neighbouring pairs of pixels, no matter their label, does the best job of maximizing ELBO over both back-propagation and message passing steps. Still, there are a lot of fluctuations visible. The unary potential continues to be responsible for most of the learning.

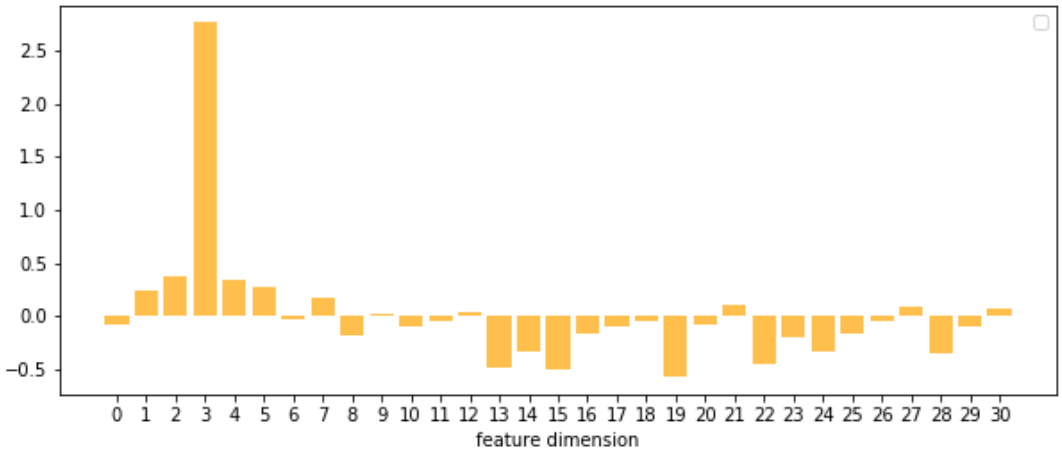
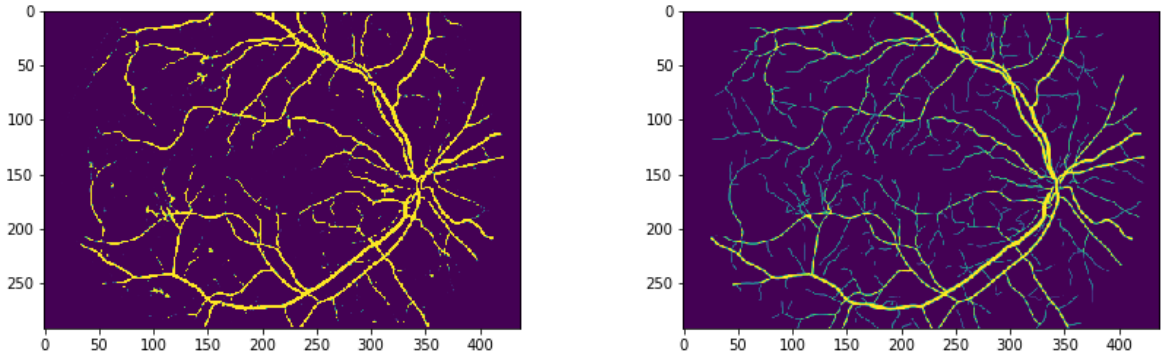
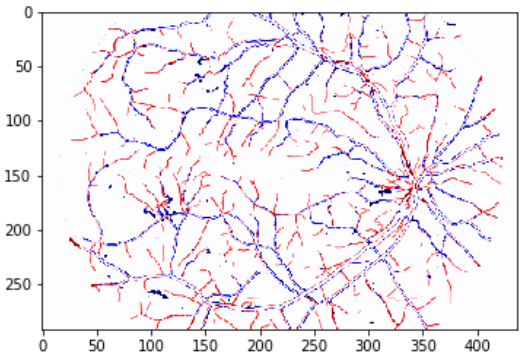


Figure 23: Case 3. Learned weights per feature dimension (W_f) of the pairwise potential. #0 = cosine similarity, #1 − #30: features as described in fig. 6.



(a) (b)



(c)

Figure 24: a) Prediction on test image after 201 epochs of training, case 3. b) Corresponding ground truth image. c) Differences: red = false negative, blue = false positive

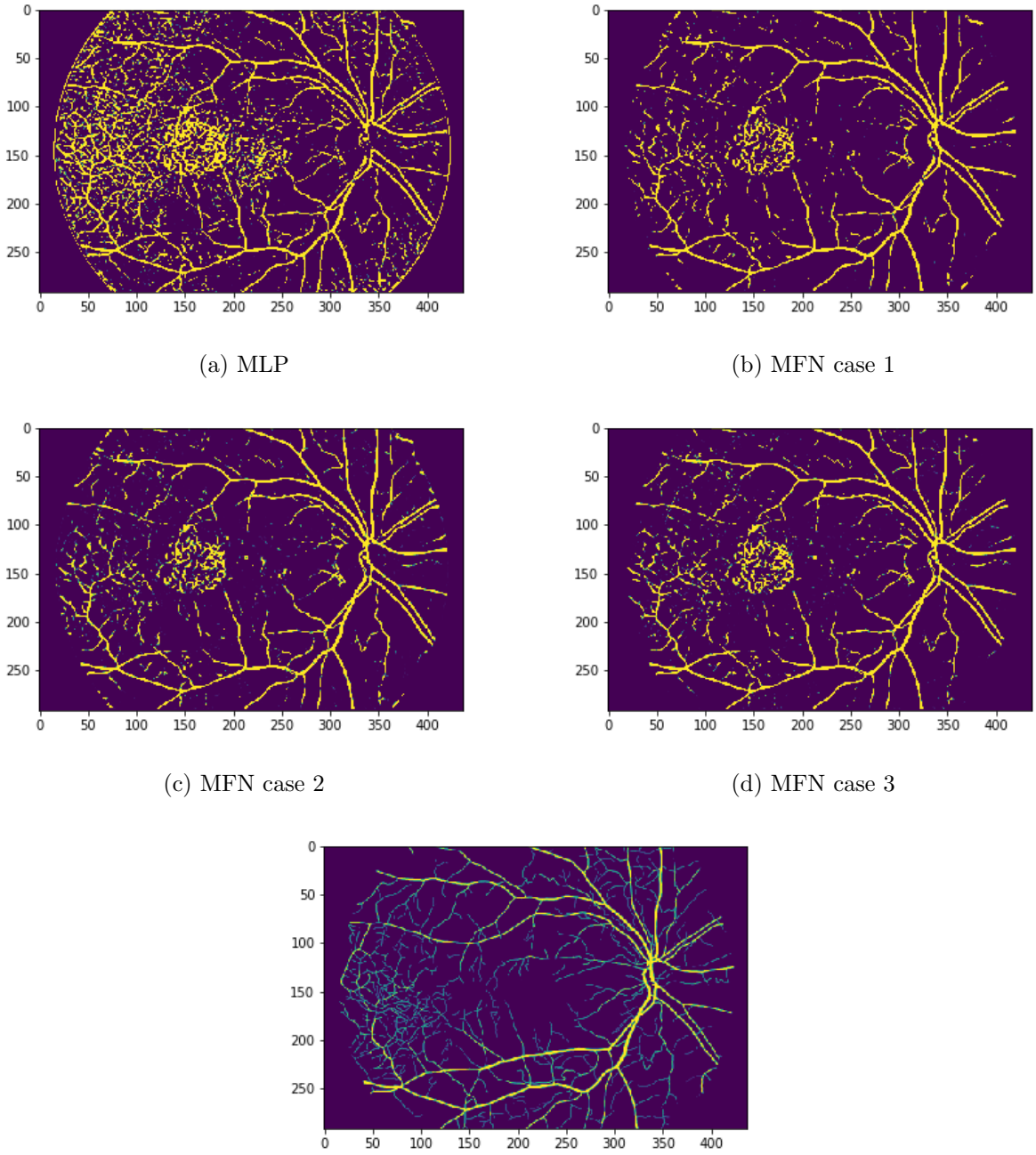
6.3 Overview

Table 1 shows the results on the test set of 9 images.

Model	**dice loss**	**Acc (%)**	**Se (%)**	**Sp (%)**
MLP	0.4057	91.12	86.62	92.56
MFN case 1	0.3240	95.35	79.77	97.01
MFN case 2	0.3180	95.42	81.91	97.07
MFN case 3	0.3128	95.53	82.26	97.11

Table 1: MLP (= MFN with only unary potential, 1001 epochs), MFN case 1 (201 epochs), MFN case 2 (201 epochs), MFN case 3 (201 epochs)

See fig. 25 for a side-by-side comparison of all model predictions on a particularly challenging image.



(a) MLP

(b) MFN case 1

(c) MFN case 2

(d) MFN case 3

(e) Ground truth

Figure 25: The MLP (a) vastly overpredicts vessels in this diabetic retinopathy image from HRF. The differences in the predictions between MFN 1-3 (b)(c) and (d) are very subtle and perhaps most visible at the border of the FoV. This image is particularly challenging to predict on, as it has many fine vessels (e).

7 Discussion

The hypothesis that message passing of the presented form improves prediction when compared to a simply MLP has been mostly confirmed by the results, as did the assumption that more inclusive indicator functions outperform the simpler ones. Considering the dice loss, Acc and Sp as evaluation metrics, the models rank as follows:

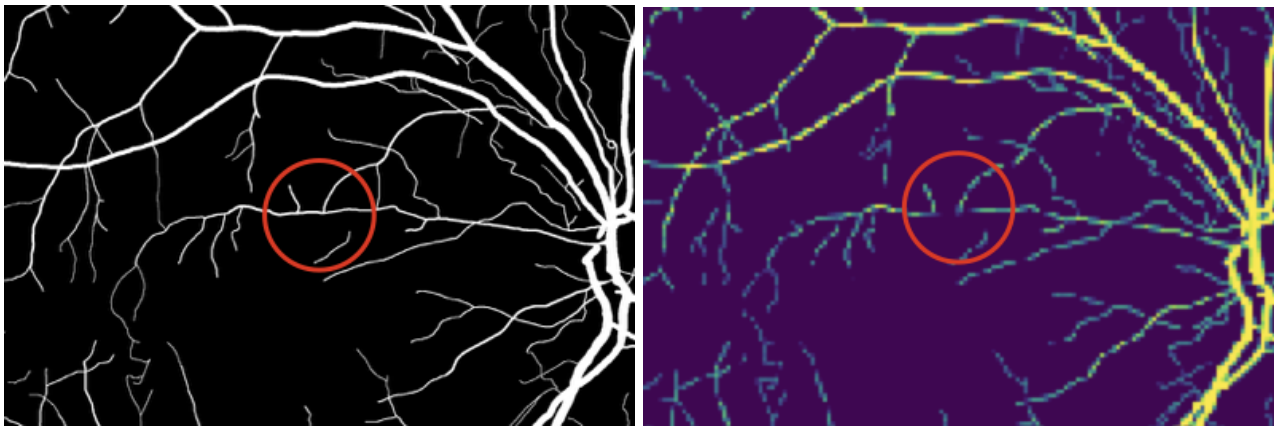
$$\text{MFN } 3 > \text{MFN } 2 > \text{MFN } 1 > \text{MLP}$$

Only for sensitivity (Se) the MLP outperforms the MFNs. This suggests MLP's predictions have a lower amount of false negatives. This however, comes at the expense of a lower Sp. As fig. 13c shows, the MLP is prone to false positives. In general the improvements of MFN predictions are not as high as expected, highlighting the need to either develop more complex energy functions or to choose a more connected CRF.Comparing the MFN cases makes it apparent that more neighbouring information is shown to increase model performance when it comes to designing the indicator function. Integrating the cosine similarity in the pairwise potential seemed to have little effect on prediction, as shown by the small weight values for feature #0 for all 3 MFN cases, whereas the euclidean similarity did contribute more, especially for the intensity feature (feature #3), which was upscaled consistently (see fig. 16, fig. 20 and fig. 23). This means that color intensity is the strongest indicator for whether two pixels share the same or different labels. The experiments also suggest that the MLP works well in conjunction with MFN. As seen in section 5.5, the decision to use a SLF activation function in the MLP as ground model informed the choice of sigma as the normalization function for $E(X, Y)$. This is one of many ways in which the unary potential model and the message passing are intertwined. It should be pointed out that the presented results do not lend themselves to comparison with the result of other work on the HRF dataset, because the images in this work where compressed, making predictions easier.

7.1 Limitations and future work

The findings could suggest that a neighbourhood defined as the 8 adjacent pixels in the regular grid could not be expressive enough to add much useful information to a basic feed forward network, given the energy function presented. Fully connected MFN are in some settings able to give better results on image segmentation tasks, even when using simple potentials based on cosine similarity alone, as showcased by Yi Li and Wei Ping [19]. The idea here is that capturing long range interactions of pixels improves results. However, this requires a lot of RAM, since similarity measures between every possible pair of pixels have to be calculated. For a number of N pixels, there are $\frac{N(N-1)}{2}$ interactions, which have to be pre-computed for every image before training can begin.

Resizing the images can lead to discontinuous vessels (see fig. 26). This is a phenomenon that doesn't occur on high-resolution images, making the findings of this method less generalisable. Dividing the image into patches for prediction or simply getting access to more RAM will solve this limitation.



(a) before resize (b) after resize

Figure 26: Resizing causes discontinuation in the vessels. Patch based prediction is therefore a better alternative to saving RAM.

In the model presented, the cosine and euclidean similarities were merged into one vector and multiplied with a shared training weight that is jointly trained. Another approach would be to give a separate term to euclidean and cosine similarities, each with their own weight vector. This would give the model more freedom possibly resulting in better prediction. In general, the design of the pairwise potential offers many possibilities for similarity functions, that can be combined in a multitude of ways.

Since ELBO optimization is very sensitive to weight initialization [11], further investigation of initializations could proof fruitful. Similarly, a better initial unary potential for the mean field could be achieved by first training the MLP alone, then loading the pretrained MLP for the message passing algorithm instead of jointly training the MLP and the Mean field.

8 Hardware

All scripts were executed on the Google colab servers. In 2019, Colab provides free access to hardware with the following specifications:

GPU: Tesla K80 , compute 3.7, 12GB GDDR5 VRAM, 358.27 GB available

CPU: single core hyper threaded i.e(1 core, 2 threads) Intel Xeon Processors @2.3Ghz (No Turbo Boost) , 45MB Cache

RAM: 12.72 GB Available

References

[1] Chetan L Srinidhi, P. Aparna, and Jeny Rajan. "Recent Advancements in Retinal Vessel Segmentation". In: *Journal of Medical Systems* 41.4 (2017). ISSN: 1573689X. DOI: 10.1007/s10916-017-0719-2.

[2] A. Budai et al. "Robust Vessel Segmentation in Fundus Images". en. In: *International Journal of Biomedical Imaging* 2013 (2013), pp. 1–11. ISSN: 1687-4188, 1687-4196. DOI: 10.1155/2013/154860.

[3] Michael M. Engelgau et al. "The Evolving Diabetes Burden in the United States". In: *Annals of Internal Medicine* 140.11 (2004), p. 945. ISSN: 0003-4819. DOI: 10.7326/0003-4819-140-11-200406010-00035. URL: http://annals.org/article.aspx?doi=10.7326/0003-4819-140-11-200406010-00035.

[4] Ignacio Orlando and Matthew Blaschko. "Learning Fully-Connected CRFs for Blood Vessel". In: *Medical Image Computing and Computer Assisted Intervention (MICCAI)* (2014), pp. 634–641. DOI: 10.1007/978-3-319-10404-179.

[5] Yujia Li and Richard Zemel. "Mean-Field Networks". In: 32.2 (2014), pp. 1–5. ISSN: 0038-1098. DOI: 10.1016/0038-1098(65)90067-0. arXiv: 1410.5884. URL: http://arxiv.org/abs/1410.5884.

[6] Raghavendra Selvan et al. "Graph Refinement based Tree Extraction using Mean-Field Networks and Graph Neural Networks". In: (2018), pp. 1–10. DOI: arXiv:1811.08674v1. arXiv: 1811.08674. URL: http://arxiv.org/abs/1811.08674.

[7] Sara Moccia et al. "Blood Vessel Segmentation Algorithms — Review of Methods, Datasets and Evaluation Metrics". en. In: *Computer Methods and Programs in Biomedicine* 158 (May 2018), pp. 71–91. ISSN: 01692607. DOI: 10.1016/j.cmpb.2018.02.001.

[8] Shuangling Wang et al. "Hierarchical retinal blood vessel segmentation based on feature and ensemble learning". In: *Neurocomputing* 149 (Feb. 2015), pp. 708–717. DOI: 10.1016/j.neucom.2014.07.059.

[9] José Ignacio Orlando and Matthew Blaschko. "Learning Fully-Connected CRFs for Blood Vessel Segmentation in Retinal Images". en. In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2014*. Ed. by Polina Golland et al. Vol. 8673. Cham: Springer International Publishing, 2014, pp. 634–641. ISBN: 978-3-319-10403-4 978-3-319-10404-1. DOI: 10.1007/978-3-319-10404-1_79.

[10] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. en. Adaptive Computation and Machine Learning Series. Cambridge, MA: MIT Press, 2012. ISBN: 978-0-262-01802-9.

[11] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. "Variational Inference: A Review for Statisticians". en. In: *Journal of the American Statistical Association* 112.518 (Apr. 2017), pp. 859–877. ISSN: 0162-1459, 1537-274X. DOI: 10.1080/01621459.2017.1285773. arXiv: 1601.00670.

[12] Xitong Yang. "Understanding the Variational Lower Bound". en. In: (), p. 4.

[13] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. en. Adaptive Computation and Machine Learning. Cambridge, MA: MIT Press, 2009. ISBN: 978-0-262-01319-2.

[14] Xitong Yang. "Understanding the Variational Lower Bound". en. In: (), p. 4.

[15] Agung W. Setiawan et al. "Color Retinal Image Enhancement Using CLAHE". en. In: *International Conference on ICT for Smart Society*. Jakarta, Indonesia: IEEE, June 2013, pp. 1–3. ISBN: 978-1-4799-0145-6 978-1-4799-0143-2 978-1-4799-0144-9. DOI: 10.1109/ICTSS.2013.6588092.

[16] Stephen M Pizer et al. "Adaptive Histogram Equalization and Its Variations". en. In: (), p. 14.

[17] Zhitao Xiao et al. “Diabetic Retinopathy Retinal Image Enhancement Based on Gamma Correction”. In: *Journal of Medical Imaging and Health Informatics* 7 (Feb. 2017), pp. 149–154. DOI: 10.1166/jmihi.2017.1998.

[18] R.C. James. *Advanced calculus*. bibtex*[lccn=66014725]. Wadsworth Pub. Co., 1966. URL: https://books.google.dk/books?id=d5kpAQAAMAAJ.

[19] Yi Li and Wei Ping. “Cancer Metastasis Detection With Neural Conditional Random Field”. en. In: (), p. 9.

[20] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *arXiv:1412.6980 [cs]* (Dec. 2014). arXiv: 1412.6980 [cs].