USING MULTI-INPUT DEEP LEARNING NETWORKS IN MEDICAL IMAGES FOR THE AUTOMATIC DETECTION OF PATHOLOGICAL STATES

Beatriz De Oliveira Noronha Filipe
beatriz.n.filipe@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal
Outobro 2021

ABSTRACT

One of the most common ways to detect a pathological state, through diagnosis, is by using medical images such as in vivo confocal microscopy, fundus photography, etc. These diagnoses are demanding, since it is needed a specialized health professional and there has been an increase of data regarding this type of diagnoses, making it too hard to manage. This increase in data availability associated with the increase in computational power has instigated the application of AI algorithms in this area. This study tested the hypotheses of multi-input deep learning networks for both images of cornea and retina, to distinguish between healthy and pathological states in diabetic neuropathy and diabetic retinopathy, respectively. It was proposed a multi-input architecture with different networks, such as LeNet-5, AlexNet, GoogLeNet and ResNet, and different optimizers (mini-batch SGD, RMSprop, Adam) with different learning rates (0.01, 0.001, 0.0001). All these variations were only applied in corneal images, and the one that showed a better performance was selected to be tested in retinal images. From the corneal images trials, it was concluded that the combination that best performed was GoogLeNet with Adam optimizer of 0.01 learning rate, since it achieved an accuracy of 99.40% in leave-one-out cross-validation. In retinal images, this architecture achieved a satisfactory accuracy of 84.44%, although the F1-score of 0.125 showed that the model is not reliable. Nevertheless, this study demonstrates the capability of the proposed multi-input network in the classification of healthy and pathological state, and encourages the investigation with more images and other diseases.

Index Terms— Convolutional neural networks, Deep Learning, Diabetic Neuropathy, Diabetic Retinopathy

1. INTRODUCTION

1.1. Deep learning

Since the decade of 1950 that the idea of simulating human reasoning with computers has been developed by computer scientists, the so-called Artificial intelligence or AI. However, this vision was in stand by, until the end of the century, due to lack of data and computational power. More recently, with an increase in data availability and improvements in computer hardware, this field started developing algorithms such as Machine Learning (ML) and Deep Learning (DL).[1]

Convolutional Neural Networks (CNNs), is one of the most used deep learning architectures it was inspired in the cat's visual cortex, an idea developed by Hubel and Wiesel. The cat's visual cortex divides the visual field into specific regions that are processed by different cells and react to different external stimulation in those regions.[2] Some cells only react to vertical orientation of the objects (edges), others only to horizontal orientation, others to movement and then further on the information obtained is merged and processed by a layered architecture.[3] For this reason it is the most adequate for pattern recognition in images, as it obtains the detected features for every position in the images, and after the updating of the weights in each fragment of the image these are averaged and applied to all sets of weights.[4]

1.2. Diabetic neuropathy and retinopathy

Diabetic neuropathy (DN) and retinopathy (DR) are serious and common complications of type 1 and type 2 diabetes.
DN is a type of nerve damage caused by long-term high blood sugar levels and loss of neurotrophic support given by insulin.[5] Early symptoms are numbness, pain, or weakness in hands or feet. Many studies associate this disease to morphological changes in peripheral nerve fibers, also mentioned as diabetic peripheral neuropathy (DPN), like corneal nerve fibers.[6] [7] [8] To diagnose DPN the most common technique used is in vivo confocal microscopy, that provides three non-overlaping images of each eye of the patient that constitute a mosaic.
DR, on the other hand, is one of the leading causes of visual deterioration and blindness in young adults and adults.[9] This condition is characterized by higher permeability in the vessels, growth of new vessels and connective tissue on the surface of the retina and into the vitreous. In most cases, in the early stages, it is asymptomatic, so there is or should be a normal clinical check up of patients with diabetes regard-

ing the neural retinal damage and microvascular changes, to better prevent and treat the disease as early as possible. The diagnosis of DR is normally based on functional changes such as electroretinography (ERG), retinal blood flow and retinal blood vessel caliber, but for early stages imaging techniques are used such as fundus photography[10]. Unlike the previous diagnostic technique, the DR with retinal images has usually one image for each eye that are both used to see if the patient has DR or not.

Both diagnoses require a clinical professional of the area, that are few for the demand since the number of cases has been increasing in the last years. For this reason there has been a wave of applications of different deep learning algorithms, mainly CNNs, to automate the detection of DR and DPN.[11]

1.3. Proposed approach

The present study focuses on a multi-input CNN approach, inspired by the paper[12]. This different type of architecture was influenced by the cornea nerve plexus data, leveraging the fact that these data has 3 images from different parts of the eye that are important to determine the presence of an abnormality. This multi-input CNN architecture was adapted to other networks such as LeNet-5, AlexNet, GoogLeNet and ResNet, with different optimizers (mini-batch SGD, Adam and RMSprop) and learning rates (0.01, 0.001 and 0.0001). The goal was to analyze how these architectures would perform in the data and in which way it is influenced.

2. METHODS

2.1. Data

2.1.1. Cornea

The data set used consisted of 252 images of healthy subjects and 252 images of pathological(type I and II diabetes) subjects, so a total of 84 subjects were included in this study, has the ones illustrated in fig.1. The images covered a field of $400 * 400 \mu m^2$ ($384 * 384$ pixels), acquired trough Heidelberg Retina Tomograph (HRT-II) with the Rostock Cornea Module (Heidelberg Engineering GmbH, Heidelberg, Germany). Image acquisition was performed in different clinical centers and each image was anonymized to eliminate patient information. They were obtained from the supervisor *Fabio Scarpa* from *Univeristy of Pádua*.

Pre-processing Due to the curvature of the cornea, problems can arise from the peripheral area of the images such as spatial distortion partial volume effect (some structures adjacent can appear in the images), illumination drift and blurring. So one of the first steps included in the algorithm while reading the images into the CNN was to shorten the external area of the images by 10 pixels, followed by a resizing by a factor of 0.7. Being the input of the network an $256 * 256$ image.

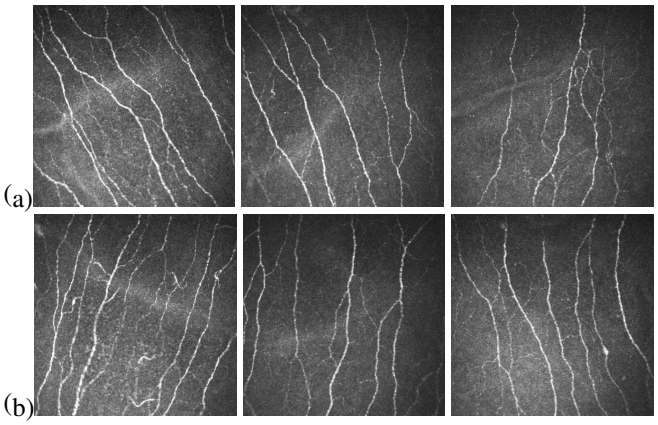


Fig. 1: In vivo confocal images of the cornea of a healthy patient. (a) right eye. (b) left eye

As it was mentioned above, the data set consists of 3 non-overlapping images for each eye, so the data was organized randomly in sets of three to prevent bias in the result, caused by the position of the images. Other techniques used to augment the data were flipping horizontally the images and organizing each set of three images in different configurations, so that each set of 3 images is represented in 3 different ways. In this way, the data set in the end is constituted of 504 blocks of 3 images (3 blocks for each one of the eyes of 84 subjects).

2.1.2. Retina

The data set used was of 506 images of healthy subjects and 506 images of pathological subjects (mild, moderate, severe or proliferative DR), giving a total of 506 subjects included in this study. The images covered a field of $2592 * 3888$ pixels, taken under a variety of imaging conditions, from an open-source data [13] (fig.2).

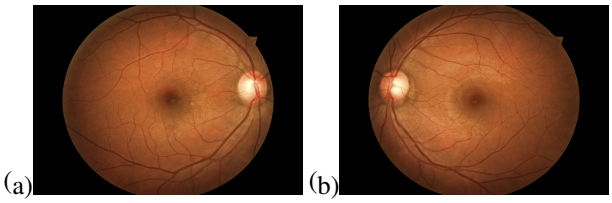


Fig. 2: Fundus photography images of the retina of a healthy patient. (a) right eye. (b) left eye.

Pre-processing

The specific images are too large to fit in a deep learning model, which causes memory issues in the performance, for that reason the data was subjected to pre-processing. Firstly the image was converted to a gray scale image, then the edges were cut, in such a way that it included practically only the area of the retina and no background (40 pixels vertically and 1400 pixels horizontally), and lastly it was resized to a $256 *$

256 to better fit the model.

Regarding the network itself, this type of data has only one image of each eye for one subject. So the images from left and right eye were given as an input to the network. In the same way has the corneal nerve plexus data, the data was also randomly organized, and augmented by flipping the images horizontally. This way each set of 2 images were represented in 2 different ways, and the whole data set was 1012 blocks of 2 images (2 blocks for each one of the 506 subjects).

2.2. Deep learning model

The multi-input CNN represented, in fig.3, is composed by 6 convolutional and pooling layers followed by two fully connected layers and one activation layer, which in the case of the present study had a sigmoid function. C1-C3 were convolutional layers with size of $7*7$ with 32 kernels, followed by S1 a pooling layer of stride 2. From S1 the outputs are all concatenated and afterwards transformed by C4 of size $3*3$ with 16 kernels, S2 pooling layer with stride 4, C5 and C6 convolutional layer with 32 kernels and size $3*3$ and lastly by and S4 pooling layers with stride 2. From the output of S4 the data is flattened and a dropout layer reduces the quantity of data, to prevent overfitting. Then FC1 and FC2 are applied, the first one with 32 neurons and the second with 3, and the result is obtained through a FC with a sigmoid activation function and 1 neuron.

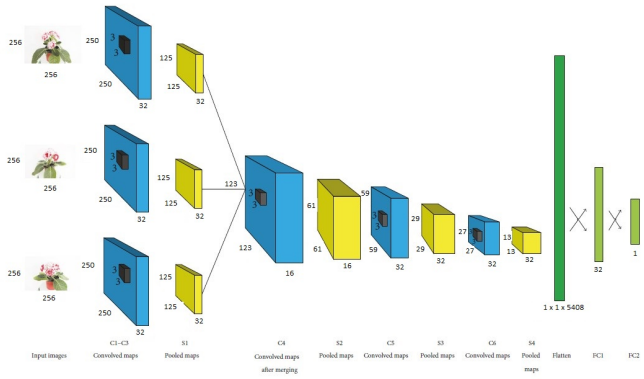


Fig. 3: Three-input CNN from the paper [12]

This model was afterwards adapted to other networks such as: LeNet-5 [14], AlexNet [15], GoogLeNet [16] and ResNet [17]. This adjustment between architectures was done by changing C1-C3 and S1 layers (from the original architecture) to the first convolutional and pooling layers of the other architectures; the rest of the network was applied after the merge of the input layers, after C1-C3 and S1. This adjustment was done for every architecture separately, where it was also tested different optimizers such as mini-batch Stochastic gradient descent (SGD), root-mean-square propagation (RMSprop) and adaptive moment estimation (Adam); for 0.01, 0.001 and 0.0001 learning rates.

2.3. Tests and Evaluation metrics

To evaluate all different models the data set was divided into train and validations sets, which is mentioned in the study as **train/validation split**, and the values of accuracy and loss in validation for the last epoch were used to measure the performance, for both single and multi-input CNNs. From the higher validation accuracy and lower validation loss it was afterwards performed *leave-one-out cross-validation* and also extracted the *confusion matrix*.

In **Leave-one-out cross-validation** the training and validation set were altered so that only one block of three images, equivalent to the diagnostic of one eye, was validated at a time and the remaining were included in the training set. This way, all blocks of three images were subjected to validation individually. The measures obtained were the mean value of the validation accuracy of the last epoch in each *training/validation set* trial, and the percentage of validation accuracies different from 1.0.

Confusion Matrix, on the other hand, was obtained by predicting the results, of the training set, with the model chosen has best. This method gives a matrix, as the one represented in table 1, whereby comparing the results predicted with the true labels it calculates the true positives (TP), false positives (FP), true negatives (TN) and false negatives (FN). From which is possible to obtain metrics such as accuracy, that provides the ratio between correct predictions and the total number of predictions; sensitivity, also called true positive rate, which measures the ratio of true positive predictions; specificity, or true negative rate, that determines the proportion of correct negatives compared with all predicted negatives; and F1-score, it quantifies the precision of the classifier as well as its robustness. All this metrics are given in values between 0 and 1, although accuracy, sensitivity and specificity can be also represented has a percentage (%). [18][19]

		Actual values	
		True	False
Predicted values	True	TP	FP
	False	FN	TN

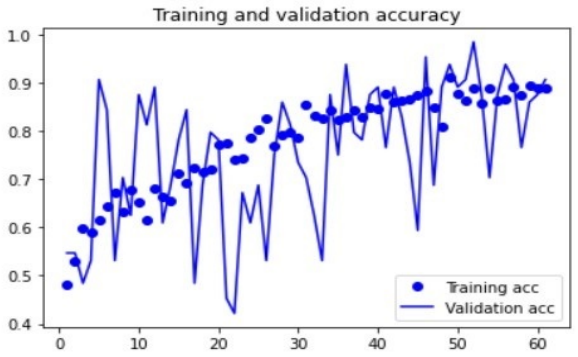
Table 1: Representation of the confusion matrix, where TP means true positive, FP means false positive, FN means false negative and TN means true negative.

3. RESULTS

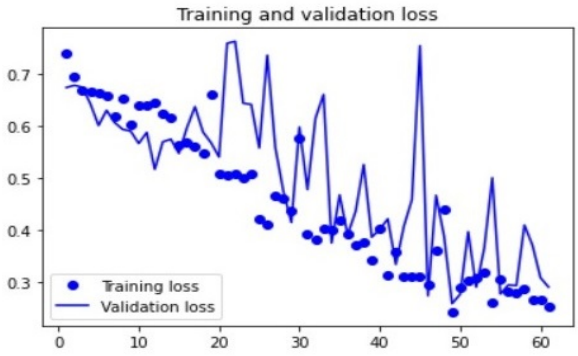
In this section the metrics obtained are presented and analyzed for all three different methods chosen. Overall the methods applied took between 30 minutes and 1 hour with the use of GPU from google colab, apart from *leave-one-out cross-validation* which took 18 hours, with 64Gb GPU from the server in Universitá di Padova.

3.1. Single-input CNN

To better compare the results between the proposed model and a conventional single-input CNN, it was performed the analysis of corneal images using the example network from [20]. This network was constituted by four convolutional and pooling layers, followed by a dropout of 0.5 and three fully connected layers. The results are represented by fig.4, where it is possible to see that the last epoch had values of 88.82% accuracy, 0.2515 loss in training and 90.62% accuracy, 0.2901 loss in validation, after 61 epochs.



(a)



(b)

Fig. 4: Analysis of accuracy and loss in the training and validation of a single-input CNN.[20]The accuracy in (a) and loss in (b) values are marked in the y axis and the number of epochs in the x axis.

3.2. Train/validation split

3.2.1. Cornea

In this section, are presented results for all combinations of architectures, optimizers and respective learning rates. For these tests the batch_size was 60 for training and 24 for validation, i.e. validation had only one batch; the steps_per_epoch were given by the number of samples (training/validation) divided by the batch_size; with 200 epochs and EarlyStoping, described in the previous section, as callback.

Multi-input network (original)

To evaluate this network the results are presented by the table 2. Overall the performance was better for higher learning rates, in all optimizers, and the best performance was obtained by mini-batch SGD with 0.01 learning rate. This resulted in 100% accuracy, 1.940E-04 loss in training and, 91.67% accuracy and 0.1588 loss in validation, with 61 epochs.

Table 2: Values from the last epoch for each optimizer applied with the architecture of the network of the study [12]

Optimizers	Lr	Val Accuracy	Val Loss	Last epoch
Adam	0.0001	0.1250	5.2007	61
	0.001	0.5000	4.5053	61
	0.01	0.9167	0.6795	63
RMSprop	0.0001	0.5000	5.5376	61
	0.001	0.2500	11.2973	61
	0.01	0.7917	0.4937	73
SGD	0.0001	0.25	2.1963	61
	0.001	0.5000	3.9969	61
	0.01	0.9167	0.1588	61

LeNet-5 As for LeNet-5 the table 3 presents the results. The model appears to have an unsatisfactory performance in validation accuracy, probably generated by overfitting, although during training all models seem to perform really well. Concerning the learning rate, the results were also inconclusive since it has a constant validation accuracy in Adam optimizer, an increase with higher learning rate in RMSprop optimizer and in mini-batch SGD the best learning rate is given by 0.001 and the other two have the same value.

Table 3: Values from the last epoch for each optimizer applied with the architecture of LeNet-5 CNN.

Optimizers	Lr	Val Accuracy	Val Loss	Last epoch
Adam	0.0001	0.5000	2.2408	61
	0.001	0.5000	7.8549	63
	0.01	0.5000	11.6819	61
RMSprop	0.0001	0.5000	3.6244	61
	0.001	0.6250	0.7861	61
	0.01	0.6250	4.3196	68
SGD	0.0001	0.3750	1.9172	61
	0.001	0.6250	1.2333	61
	0.01	0.3750	5.1471	61

AlexNet The table 4 describes the AlexNet results. By looking at the table, it is possible to notice that there is an increase in validation accuracy aligned with the increase of the learning rate. Regarding the best result it can be observed

by the figures and confirmed in the table that the best one occurred in mini-batch SGD with 0.01 learning rate, with values of 100% accuracy, 3.000E-04 loss in training, and 87.50% accuracy, 1.2767 loss in validation, after 73 epochs.

Table 4: Values from the last epoch for each optimizer applied with the architecture of AlexNet CNN.

Optimizers	Lr	Val Accuracy	Val Loss	Last epoch
Adam	0.0001	0.5000	16.8937	61
	0.001	0.7500	4.7785	66
	0.01	0.7917	1.7941	86
RMSprop	0.0001	0.6250	8.2155	61
	0.001	0.3333	11.1866	63
	0.01	0.7083	67.0281	135
SGD	0.0001	0.5000	1.9899	61
	0.001	0.6250	3.1305	61
	0.01	0.8750	1.2767	73

GoogLeNet

In the case of GoogLeNet, the results are given by the table 5. The optimizer that had the best performance was Adam with 0.01 learning rate, this can be noticed in the figures and also confirmed by the table, since it achieved 99.76% accuracy, 0.0080 loss in training, and 100% accuracy, 0.0000 loss in validation, after 89 epochs. Regarding the learning rate, it is observed that the validation accuracy increases with the increase of the learning rate.

Table 5: Values from the last epoch for each optimizer applied with the architecture of GoogLeNet CNN.

Optimizers	Lr	Val Accuracy	Val Loss	Last epoch
Adam	0.0001	0.5000	5.7099	61
	0.001	0.7917	2.2100	61
	0.01	1.0000	0.0000	89
RMSprop	0.0001	0.3750	2.2920	61
	0.001	0.5000	1.8160	61
	0.01	0.7083	1	129
SGD	0.0001	0.5000	3.4957	61
	0.001	0.6250	0.9412	61
	0.01	0.7083	6.7843	86

ResNet

Lastly, the performance of ResNet architecture is presented by the table 6. The model that best performed was the 0.01 RMSprop, as the figures illustrate and the table corroborates, since the values obtained were 100% accuracy, 3.030E-09 loss in training and 87.50% accuracy, 0.8005 loss in validation after 156 epochs. Concerning the learning rate, in the case of ResNet it is also not very consistent, since only in RMSprop the validation accuracy increases with the learning rate and the other two optimizers have the lowest value of validation accuracy when the learning rate is 0.001.

Table 6: Values from the last epoch for each optimizer applied with the architecture of ResNet CNN.

Optimizers	Lr	Val Accuracy	Val Loss	Last epoch
Adam	0.0001	0.5000	2.0261	61
	0.001	0.3750	1.5761	67
	0.01	0.8333	0.3634	94
RMSprop	0.0001	0.2500	6.4950	61
	0.001	0.8134	0.7083	70
	0.01	0.8750	0.8005	156
SGD	0.0001	0.5000	0.7152	61
	0.001	0.2500	26.5744	61
	0.01	0.7500	6.9600	95

From these results, it can be said that the GoogLeNet architecture had the best performance, considering accuracy and loss in validation. For this reason the next results were measured using this architecture, as well as the Adam optimizer with learning rate 0.01 (fig.5).

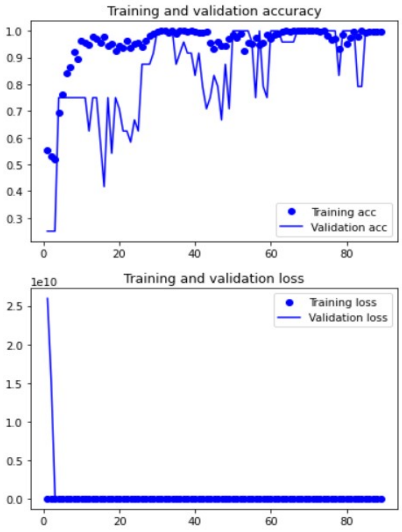


Fig. 5: Analysis of losses and accuracy in the training and validation of the GoogLeNet architecture with Adam 0.01 learning rate optimizer. The accuracy, in the top graphics, and loss, in the bottom graphics, values are marked in the y axis and the number of epochs in the x axis.

3.2.2. Retina

For retina, it was tested the GoogLeNet algorithm with Adam 0.01, as mentioned, but it was added ReduceLROnPlateau as

callback, together with Earlystopping. ReduceLROnPlateau function monitors a quantity, in this case 'val_loss', and if no improvement is observed for a 'patience' number of epochs (10), the learning rate is reduced. For this task the batch_size was 28 for training and 20 for validation; the steps_per_epoch were given by the number of samples (training/validation) divided by the batch_size; with 200 epochs. The results regarding retinal images are represented by fig.6.

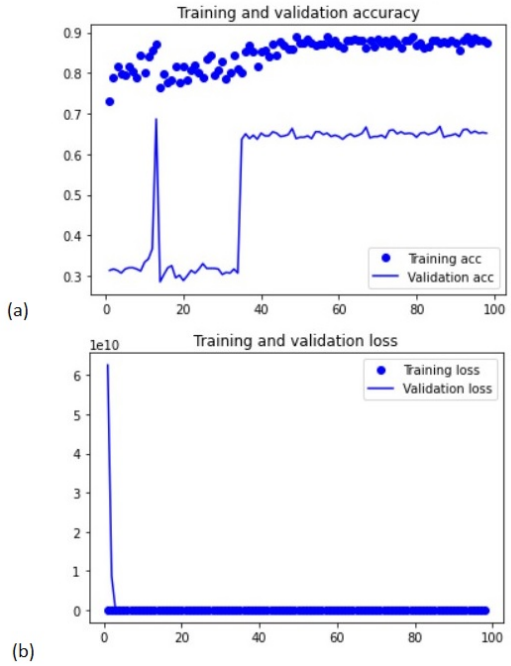


Fig. 6: Analysis of loss and accuracy in the training and validation of the GoogLeNet architecture with Adam optimizer of 0.01 learning rate. The accuracy in (a) and loss in (b) values in the y axis and the number of epochs in the x axis.

The results obtained in the last epoch were 87.30% accuracy, 0.3444 loss in training and 65.17% accuracy, 1.2765 loss in validation; it was also noticed that after 98 epochs the learning rate had reduced to 1.0000E-05. Although the mechanisms used were with the goal to minimize overfitting, it seems that it is still present in the results.

3.3. Leave-one-out cross-validation

From the results in the subsection below, it was tested the GoogLeNet architecture with Adam (0.01 learning rate) in the Leave-one-out cross-validation. This method, as it was described before, took a block of three images (one eye) in each iteration as validation, so it ran 168 times in the same conditions as the *train/validation split* for GoogLeNet (Adam 0.01). The mean accuracy obtained, from all last epochs, was 99.40%, which is the best one compared also with the literature.

3.4. Confusion matrix

As it was mentioned before, for these measurements it was taken the GoogLeNet and the Adam optimizer. The models were trained and the training set, in both cases, was used to predict the results. From the comparison between the predicted results and the original labels, the confusion matrix was then obtained.

3.4.1. Cornea

To test these data, it was only used the Adam as optimizer, with 0.01 learning rate. The confusion matrix is represented by table 7

		Actual values	
		True	False
Predicted values	True	69	7
	False	2	74

Table 7: Values of confusion matrix from GoogLeNet (with Adam 0.01 learning rate) optimizer.

From this table the values of accuracy, sensitivity, specificity and F1-score were respectively 94.08%, 97.18%, 91.35% and 0.9388. These results were quite satisfactory, it shows that the model is robust and has high accuracy, as well as sensitivity and specificity.

3.4.2. Retina

For retina, it was used, also the model already trained in the section *train/validation split*, GoogLeNet architecture with Adam optimizer (0.01 learning rate), and ReduceLROnPlateau and EarlyStopping as callbacks. The confusion matrix obtained is illustrated in the table 8.

		Actual values	
		True	False
Predicted values	True	2	25
	False	3	150

Table 8: Confusion matrix of GoogLeNet network with Adam 0.01.

The results calculated from the table were: 84.44% accuracy, 40.00% sensitivity, 85.71% specificity and 0.125 F1-score. Although accuracy and specificity were good, F1-score show's that the model has very little robustness and therefore the results are not reliable.

4. LIMITATIONS

Although the results were satisfactory, there were limitations to the study. Overall, the limitations of the study included

a small data set for both diagnosis. In DPN the corneal image data set was small in number and on the characteristics of the data itself, since it only covered small regions instead of all cornea regions. The single-input trial with corneal data was not optimized for augmentation neither for the best architecture found. The multi-input, on the other hand, had also some optimizing issues that should be further worked on. The *train/validation split* metrics were not measured in the best way, considering that it tests the model in such a way that a different training and validation set are picked in each iteration, so the comparison between the different networks and optimizers has not the same initial conditions (it is random). As for the *confusion matrix*, it had the same data set in the training and the testing phase, which is not ideal, since the test set should introduce new data.
Specifically regarding DR besides the lack of data, this diagnosis was for the patient, i.e. the input images were a retinal image of each eye (two inputs), instead of three images of the same eye has it was for cornea. This difference in inputs as well as the use of the same network might also have impacted the results regarding accuracy and the other metrics of the model. Another factor, might have been the pre-processing. Since the images were of higher resolution, they were shrunk to fit in the model and turned to grayscale images, which can cause a deterioration in the quality of the data.

5. CONCLUSION

Regarding the implementation of the different algorithms to the corneal images, it can be said that GoogLeNet (with 0.01 learning rate Adam optimizer) was the one that best performed with a mean accuracy of 99.40% which also is better then the single-input trial. It achieved in all three different evaluation methods a high accuracy, and also achieved higher values in sensitivity, specificity and F1-score, which confirms that the method is robust. When comparing these results with the ones from the literature, the multi-input network outperforms them in all cases.

For retinal images, the proposed multi-input network has been tested for the first time on a small data set and the literature results are better then the ones obtained for both accuracy, sensitivity and specificity. This can be due to the fact that only one type of optimizer was tested, maybe a different optimizer, learning rate, or type of architecture (LeNet-5, AlexNet, ResNet) should be considered. Another approach would be to have more samples in the training set or use better mechanisms to overcome overfitting.

Considering all mentioned above and the goal of the thesis, it can be concluded that this dissertation obtained good results regarding the application of multi-input networks to corneal images for DPN diagnosis. Although there were some limitations, the results were higher than in the literature, however it is still needed more research with much more data and more trials to confirm the robustness of the model. In retina, although it was not the best performance, it can be further on explored with different CNN architectures, more data or by including transfer learning, which will tackle the overfitting issue. The proposed model shows that multi-input networks can be very helpful in differentiating between healthy and pathological states, in diagnosis with more than one input or even in multi-modal diagnoses. It also encourages the investigation on different types of data and diseases.

6. REFERENCES

[1] Charu C Aggarwal et al., "Neural networks and deep learning," *Springer*, vol. 10, pp. 978–3, 2018.

[2] Manjunath Jogin, MS Madhulika, GD Divya, RK Meghana, S Apoorva, et al., "Feature extraction using convolution neural networks (cnn) and deep learning," in *2018 3rd IEEE international conference on recent trends in electronics, information & communication technology (RTEICT)*. IEEE, 2018, pp. 2319–2323.

[3] Nicola Strisciuglio and Nicolai Petkov, "Brain-inspired algorithms for processing of visual data," pp. 105–115, 2019.

[4] Geoffrey Hinton, "Deep learning—a technology with the potential to transform health care," *Jama*, vol. 320, no. 11, pp. 1101–1102, 2018.

[5] James L Edwards, Andrea M Vincent, Hsinlin T Cheng, and Eva L Feldman, "Diabetic neuropathy: mechanisms to management," *Pharmacology & therapeutics*, vol. 120, no. 1, pp. 1–34, 2008.

[6] Gérard Said, "Diabetic neuropathy—a review," *Nature clinical practice Neurology*, vol. 3, no. 6, pp. 331–340, 2007.

[7] V Bansal, J Kalita, and UK Misra, "Diabetic neuropathy," *Postgraduate medical journal*, vol. 82, no. 964, pp. 95–100, 2006.

[8] Shanshan Wei, Faqiang Shi, Yuexin Wang, Yilin Chou, and Xuemin Li, "A deep learning model for automated sub-basal corneal nerve segmentation and evaluation using in vivo confocal microscopy," *Translational Vision Science & Technology*, vol. 9, no. 2, pp. 32–32, 2020.

[9] Tien Yin Wong and Neil M Bressler, "Artificial intelligence with deep learning technology looks into diabetic retinopathy screening," *Jama*, vol. 316, no. 22, pp. 2366–2367, 2016.

[10] Nikos Tsiknakis, Dimitris Theodoropoulos, Georgios Manikis, Emmanouil Ktistakis, Ourania Boutsora, Alexa Berto, Fabio Scarpa, Alberto Scarpa, Dimitrios I.

Fotiadis, and Kostas Marias, "Deep learning for diabetic retinopathy detection and classification based on fundus images: A review," *Computers in Biology and Medicine*, vol. 135, pp. 104599, 2021.

[11] Michael David Abràmoff, Yiyue Lou, Ali Erginay, Warren Clarida, Ryan Amelon, James C Folk, and Meindert Niemeijer, "Improved automated detection of diabetic retinopathy on a publicly available dataset through integration of deep learning," *Investigative ophthalmology & visual science*, vol. 57, no. 13, pp. 5200–5206, 2016.

[12] Yu Sun, Lin Zhu, Guan Wang, and Fang Zhao, "Multi-input convolutional neural network for flower grading," *J. Electr. Comput. Eng.*, vol. 2017, pp. 9240407:1–9240407:8, 2017.

[13] "Diabetic retinopathy detection," https://www.kaggle.com/c/diabetic-retinopathy-detection/data, Accessed: 2021-08-30.

[14] Mostafa Gazar, "Lenet-5 in 9 lines of code using keras," https://medium.com/@mgazar/lenet-5-in-9-lines-of-code-using-keras-ac99294c8086, Accessed: 2021-08-1.

[15] Richmond Alake, "Googlenet," https://towardsdatascience.com/implementing-alexnet-cnn-architecture-using-tensorflow-2-0-and-keras-2113e090ad98, Accessed: 2021-08-1.

[16] "Code for plotting the keras model graph," https://github.com/Machine-Learning-Tokyo/DL-workshop-series/blob/master/Part%20I%20-%20Convolution%20Operations/ConvNets.ipynb, Accessed: 2021-08-1.

[17] Sachin Mohan, "Keras implementation of resnet-50 (residual networks) architecture from scratch," https://machinelearningknowledge.ai/keras-implementation-of-resnet-50-architecture-from-scratch/, Accessed: 2021-08-1.

[18] Mohit Khanna, "Classification problem: Relation between sensitivity, specificity and accuracy," https://www.analyticsvidhya.com/blog/2021/06/classification-problem-relation-between-sensitivity-specificity-and-accuracy/, Accessed: 2021-09-30.

[19] Aditya Mishra, "Metrics to evaluate your machine learning algorithm," https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234, Accessed: 2021-09-30.

[20] Francois Chollet, *Deep learning with Python*, Simon and Schuster, 2021.