מכללה אקדמית הדסה ע.ר.
Hadassah Academic College R.A.
Department of Computer Science
מכללה אקדמית הדסה ירושלים
החוג למדעי המחשב

CLASSIFYING KERATOCONUS AND ASTIGMATISM WITH DEEP LEARNING

Final Project Report
2017

Gil Mor
gilmor11@gmail.com

Supervised by: Dr. Yoram Yekutieli

August, 2017

In collaboration with The Optometry Department in Hadassah College
Prof. Ariela Gordon-Shaag
Dr. Einat Shneor

Abstract

The aim of this project was to diagnose Keratoconus and Regular Astigmatism corneal disorders in corneal images, using deep learning with convolutional neural networks. Our data set consisted of 250 corneal images - an extremely small amount of data for training a deep neural network for an image recognition task. We overcame this challenge by using 'transfer learning' – utilizing the knowledge learned by a pre-trained learning algorithm to solve a new problem that the algorithm was not trained to solve. In our case, we took a neural network pre-trained on the general-purpose ImageNet data set and fine-tuned it to recognize corneal disorders. The technique achieved 93.6% classification accuracy on the Healthy, Keratoconus and Regular Astigmatism classes in a short training time.

Table of Contents

Table of Figures

Table of Tables

Introduction

Corneal Disorders

The cornea is the front part of the eye that covers the iris and the pupil. The cornea plays a major role in human vision, it refracts light into the eye and it is responsible for two-thirds of the eye's refractive power. For the cornea to refract light correctly, it should have a round shape and smooth surface. Some eye disorders distort the cornea topography and cause visual impairment.

Astigmatism

Astigmatism is the general term for corneal surface irregularity. There are two main types of Astigmatism: regular astigmatism and irregular astigmatism. Regular astigmatism is the most common type and is characterized by an irregularity that resembles a symmetric 'bow-tie'. Irregular astigmatism is characterized by an asymmetric irregularity. It can be caused by an eye-injury or Keratoconus disease (Nordqvist 2017). Regular astigmatism is a common disorder, which can be treated with corrective lenses in the majority of cases (Nordqvist 2017).
A corneal topology map can clearly show different corneal conditions.

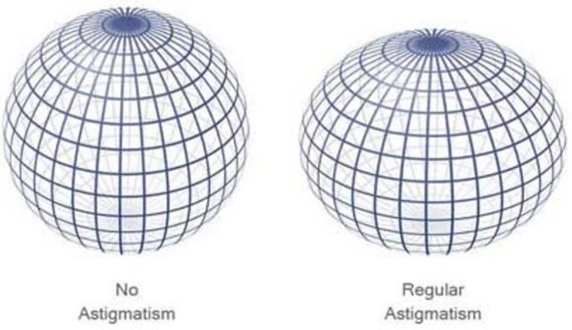


Figure 1. Healthy, round cornea vs regular astigmatism cornea.

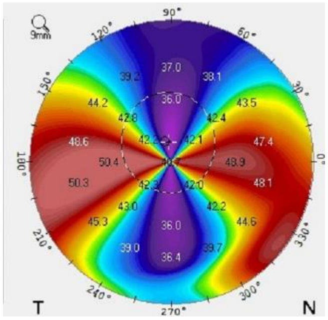


Figure 2. Corneal topology map of regular astigmatism. You can see the bow-tie iregularity.

Keratoconus

Keratoconus Disease (KC) is an eye condition that is characterized by progressive thinning and weakening of the cornea. The weakening of the cornea causes its topography to change, leading to irregular astigmatism. The round cornea starts to protrude outward. Its shape changes to a cone-like structure. This results in significant visual impairment due to irregular light refraction. The onset of KC usually occurs in the second decade of life. The early signs of KC are thin cornea followed by very subtle topography changes. The mild changes are considered undetectable without the use of dedicated computer-assisted videokeratography systems (Li et al. 2009).

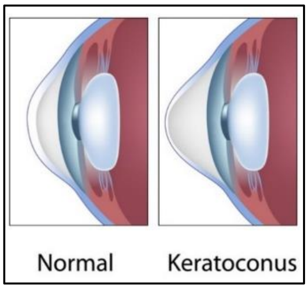


Figure 3. Healthy Cornea vs. KC Cornea

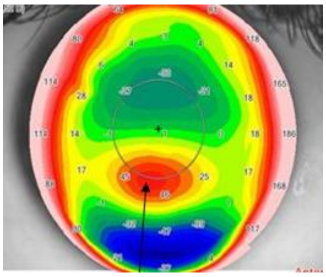


Figure 4. Topology map of KC cornea. The red area bellow the center is the bulged cornea being pulled down by gravity.

Aims

This project is part of a multidisciplinary umbrella project conducted in Hadassah Academic College in Jerusalem. The goal of the umbrella project is to create a cheap and mobile system for detecting Astigmatism with a focus on Keratoconus eye disease. The system is to be used as a screening solution and/or for self-examination.

This sub-project mission is to detect the conditions in eye-images and classify it as Healthy, KC, KC suspect, or as regular astigmatism. The goal is to implement a robust and flexible system that could be extended to different disorders and adapted to different types of input data, i.e., a different corneal imaging technique. For this purpose, the system should be able to classify unconstrained images without any form of feature engineering that could tie it to a specific type of input.

Image Classification, Machine Learning, Feature Engineering, and Deep Learning

One of the challenges in computer vision lies in the fact that general images are unconstrained – Images are taken from different angles in different lightning conditions. They might be blurred and the objects in them can appear at different locations and come in different sizes and aspects. They are full of 'noise' – spare data that covers the underlying concept. Machine learning algorithms were proven very useful for computer vision. They can learn the rules behind different variations of the same concept. However, handling unconstrained visual data is still a hard task for machine learning.

Feature engineering is the process of reengineering the input data for a learning algorithm. The term feature engineering is vague and has no uniform definition. Here we will refer to feature engineering as data preprocessing that applies domain-specific knowledge to simplify the data for a learning algorithm. Applying domain-specific knowledge means that a human researcher has to customize the procedure for the problem at hand after examination. According to this definition, generic preprocessing that could be used on any type of input is not considered as feature engineering. Feature engineering could involve putting constrains on the data like centering images around some pre determined anchor point, or extracting prominent features that closely represent the original data with less parameters. Feature engineering is a research and labor-intensive task. According to Andrew Ng*: "Coming up with features is difficult, time-consuming, requires expert knowledge"* (Ng 2015). Moreover, it might be input specific, so changing the type of input could mean repeating the procedure.

Until recent developments in deep learning, conventional machine-learning algorithms could not handle high dimensionality visual data (Chen and Lin 2014). Deep learning algorithms automatically extract prominent features by training on large amounts of data.

Approach

Deep Learning with Convolutional Neural Networks (CNN) is a powerful tool for classification of unconstrained visual data. This makes these systems ideal for this project's objectives.

Challenges Posed by Deep Learning Approach

1. Training a CNN from scratch requires large amounts of images due to the high number of parameters it contains. A CNN needs to tune millions of parameters to classify images. Many CNNs are trained on the ImageNet data set for the ILSVRC competition (Russakovsky et al. 2015). This data set contains 1,200,000 images categorized into 1000 categories such as nature, animals, species, day-to-day objects and much more.

2. Training a CNN requires large computational power. Training from scratch can take several days even with multiple GPUs working in parallel (Krizhevsky et al. 2012).

Solution: Transfer Learning (AKA Inductive-Transfer or Fine-Tuning)

Transfer learning is a technique that addresses the two problems posed by deep learning. In transfer learning, we take a pre-trained net and resume its training on a new data set. Transfer-Learning relies on the fact that the lower layers of CNNs that were trained on natural images contain common, generic, low-level features like edges and color blobs, regardless of the specific images in their data set.

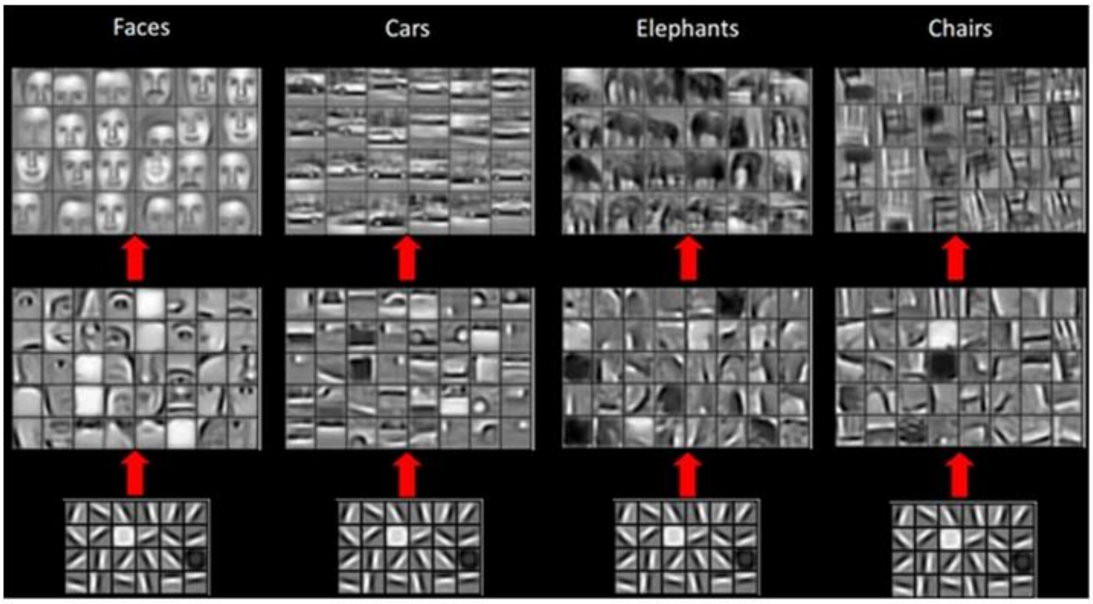


Figure 5. Illustration of layers in CNNs trained on different types of images. In all cases, the lower layers learned edges and color blobs while higher layers learn data-specific features.

Using transfer learning makes it possible to fine-tune a CNN on a small data set in a relatively short time. Using transfer learning is effective when the features learned by the pre-trained net can help in the classification of the new data set.

Background

Keratoconus Detection

Advanced Keratoconus can be detected with the naked eye (Very bulged cornea), while detecting early Keratoconus is done with dedicated equipment. An early device for examining the corneal topography is the Keratoscope or Placido's disk. Placido's disk works by projecting a series of concentric light rings on the cornea. On a normal, symmetric, cornea the rings should reflect in an evenly spaced, circular manner. On an asymmetric cornea, the rings will reflect in a deformed and skewed manner.

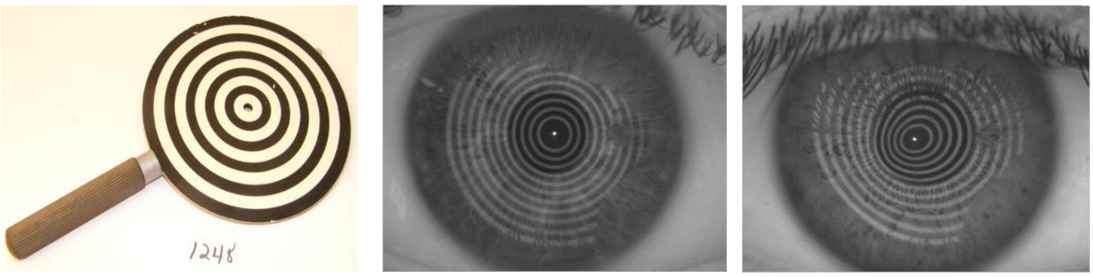


Figure 6. Left: Placido's disk (Keratoscope). The disks would reflect on the patient's cornea as the doctor looks through the peephole. Middle: The disks as reflected on a normal cornea. Right: The disks as reflected on a KC cornea.

Modern Devices for KC diagnosis

For detection of early KC or KC suspects, computerized corneal analysis devices that employ additional or other imaging techniques are used. These devices measure corneal thickness (pachymetry) and produce anterior and posterior corneal topography maps. They are considered state of the art in detecting early signs of KC.

Problem Statement

This project faces several technical challenges:

Using partial corneal info – Advanced corneal analysis systems usually provide anterior and posterior corneal topography maps along with corneal thickness map and posterior chamber depth (Anderson no-year). Our data set is composed only of raw Placido disk images. Placido's disks can measure the anterior corneal surface topography but they are not used for measuring the corneal thickness (pachymetry) or the posterior corneal surface (Miles et al. 2016). This is not supposed to be an issue when classifying KC patients, where the corneal surface is already deformed, but it could raise a problem when trying to classify KC suspect or early KC that can have no visible or very mild topography irregularities. In the case of detecting KC suspects or early KC, the corneal thickness map is the main feature that is examined.

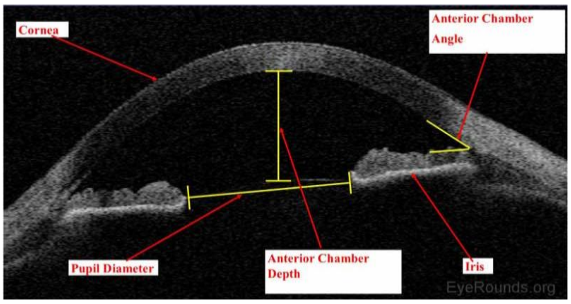


Figure 7. Eye Image taken with Scheimpflug camera. Scheimpflug image is used to produce posterior corneal curvature and corneal thickness maps.

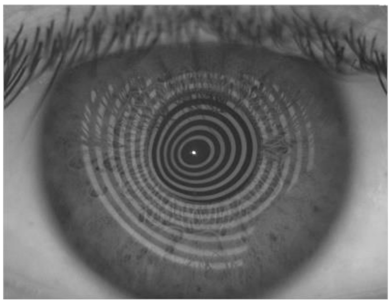


Figure 8. Placido Disk image of KC cornea as taken by the Placido-Scheimpflug based SIRIUS Schwind device.

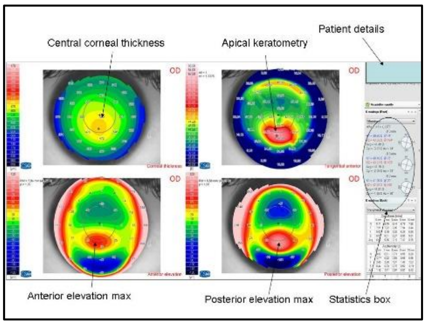


Figure 9. 4 corneal maps produced from both placido images and scheimpflug images.

Low amounts of data – Transfer learning is a solution for small data sets. Nevertheless, there is still a need to have enough data to train the network for the new task. We have 257 grey-scale images of size 256X256 divided into four categories: 103 images of healthy, 86 images of KC, 37 images of KC suspects, and 31 images of regular astigmatism. 257 images are considered an extremely low amount of data for any machine-learning task.

Classifying KC suspects - KC suspects are defined as people with either too thin cornea and/or topography changes with no visual impairment (Li et al. 2009). This means that their Placido images can show anything from symmetric Placido disk reflection to asymmetry that resembles mild KC. Trying to train any machine-learning algorithm with this input would raise some difficulties. This problem is intensified by the fact that we only have 37 images of KC suspects.

Related Projects

Three projects focusing on the classification of KC were already carried out under the umbrella project. One project by Ifergan and Koretz (Ifergan and Koretz 2016) used ‘Crowd Wisdom’ for classification. In crowd-wisdom, volunteers from the general population can access the images via the internet and offer their classifications. The average result determines the final diagnosis. The results displayed high accuracy in classification of normal and KC images, and lower accuracy in classification of KC suspects. The project experienced major difficulties in recruiting volunteers.

Two other projects, by M.Sc. student Babi (Babi 2015) and B.Sc. student Elharar (Elharar 2016), respectively, employed machine learning for classification but they used excessive feature engineering procedures in order to reduce the problem's dimensions. The last project by Elharar reduced each image by a single numerical value that was fed into the learning algorithm. The results of these two projects displayed high accuracy in classification of healthy vs. KC (97.5%) and lower accuracy when classifying healthy vs. KC and KC suspects (84%). In the latter, the cases were categorized in two groups; one consisting of KC and KC suspects, the other consisting of healthy. The experiment resulted in 29% of KC and KC suspects being wrongly classified as healthy, the vast majority of these being KC suspects.

Advantages over Previous Projects

1. **Advantages over crowd-wisdom:** Does not rely on the availability of human classifiers.
2. **The advantages over previous machine learning projects:**
   2.1. Free of feature engineering: The two projects relied heavily on feature engineering in order to achieve good accuracy with a linear classifier. Freeing the system from feature engineering offers several advantages:
   2.1.1. Minimize effort: Feature engineering consumed a large amount of the project's time.
   2.1.2. Robustness: Feature engineering could mistakenly rely on patterns that exist in the currently available data but may be absent in newly introduced data. This is obviously a problem when introducing a new type of data, but even newly introduced similar data could display variations that were not taken into account. Such was the case of the centering procedure that was carried out in the previous projects. The procedure relied on a bright dot that is projected on the center of the eye by the imaging device. In recent

additions to the data set, we received several images with a deformed cornea that would pose a problem to the centering procedure.

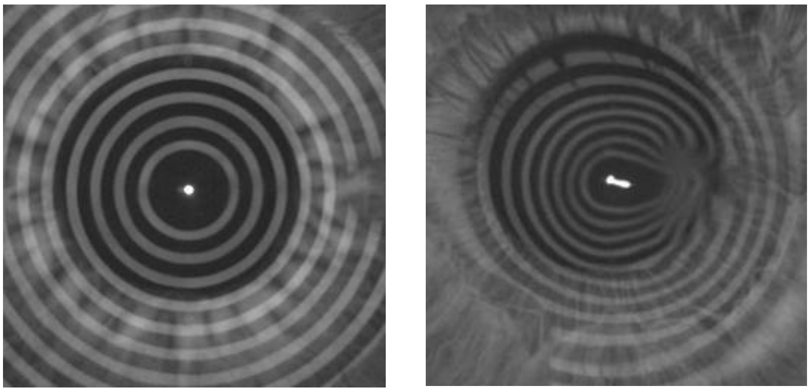


Figure 10. The bright white dot was the anchor point for the image centering process. The stretched bright dot would pose a problem to the centering algorithm used in previous projects.

2.2. **Using open-source framework:** We used an acclaimed deep learning framework developed by experts in machine learning and computer vision. Using a popular open source framework has a few benefits:

1. Support in the form of updates, feature requests, bug fixes, and support forums.
2. Ability to fully customize the software to your needs.

Requirements

The project needs to meet the following demands:

1. The input data will be Placido disk images.

2. There will be no feature-engineering procedures.

3. After the training, the deployed network will classify images to Normal and KC with high accuracy and KC suspects with reasonable accuracy.

4. After training, the deployed network will classify a single image in no longer than a few seconds.

5. The project will follow medical confidence guidelines.

6. Intellectual property – The project will be kept under IP confidentiality.

7. Research and Development will follow Scientific and Academic standards.

8. All failures to achieve any of the goals will be documented and explained in accordance to scientific and academic standards.

Specifications and Design

Caffe Deep Learning Framework

Caffe (Jia et al. 2015) is a popular deep learning framework developed in Barkley University. It offers an API for configuring, training, testing, and deploying neural networks. The Caffe Model-Zoo offers numerous trained networks for download and pycaffe provides a convenient python API.

Choosing a pre-trained CNN

We chose Caffe's reference net, caffenet (Donahue 2014), as our pre-trained CNN. Caffenet was chosen for its performance and tutorials availability. It was trained on the general ImageNet data set for ILSVRC. We did search for a network trained on data that is more similar to our target problem, such as a net trained to find circles or evaluate symmetry but no such network was found. We could only hope that the very general knowledge learned from ImageNet would be helpful in classifying our data.

Computational Infrastructure

Although CNNs are currently built to run on GPUs, Caffe allows to run its models on CPU. The fact that we are using transfer learning with a very small data set is what allowed us to run the net on a personal laptop with a moderate dual core CPU clocking at 2.6 GHz and 4GB RAM and still get results in a sensible timeframe. After several months, the PC CPU gave up and could no longer handle the load of running the net. That is when we installed Caffe on a GPU instance of 'Amazon Elastic Compute Cloud' service (AWS EC2). The boost in speed was by a factor of 60-100. It is worth mentioning that we tried running the net on a powerful, CPU only, EC2 instance and did not notice almost any increase in speed compared to the PC. The conclusion is that even a strong CPU with sufficient RAM does not contribute to the speed of Caffe's models (It is possible that installing caffe with specialized CPU performance enhancement libraries would speed it up, but it would not come close to running on GPUs).

Overview of the Training Process

We trained a CNN to classify images by supervised-learning. The final goal is to train a classifier that will be able to predict the labels of new, unlabeled data with high accuracy. The net is trained on a training set and its performance is evaluated on a separate, unlearned, validation set. During training, the net attempts to classify the training samples and update its weights (its learnable parameters) in order to minimize the classification errors. The net preforms a validation routine on the validation set every specified number of iterations so that we can track the learning process. This enables us to check how well the learning converges.

Cross-Validation

Cross-validation is a technique that helps to evaluate the generalization of a classifier. In cross validation, we partition our data into complementary validation sets so that every image appears once in one of the validation sets. We then average the results over all validation sets. This is contrary to the naïve approach where we would arbitrarily partition the data into one training set and one validation set (the split is around 80/20) and make the classification. The obvious downside of the naïve approach is that not all images are validated (classified). The other downside is that it requires a bigger validation set, which means a smaller training set. It is obvious that one cannot deduce the classifier performance with certainty from the results of one validation set. The randomness introduced by the naïve approach becomes greater as the data set is smaller. With a small data set such as ours, every image makes a difference.
We used cross-validation where the test-set was 5% of the relevant classes.

CNN Specific Details

CNNs are composed of layers where each layer has its own weights. The net learns by optimizing its weights in order to minimize the classification error on the training set. The weights span an 'error-landscape'
$(\mathbb{R}^n \ space\ where\ n\ is\ the\ size\ of\ the\ net\ weights)$. Each state of the weights is associated with an error-value on that 'landscape', and the current state of the weights is the current location of the net in that space. The goal in learning is to find minima in the error-landscape (The minima is not always a global minima. It could be one of many local minima). Caffenet uses SGD algorithm (Stochastic Gradient Descent) to optimize its weights in the learning process. Generally, SGD searches for minima in the error-landscape by following the negative of the gradients downhill.
CNNs learn by iterating over the data repeatedly. In each iteration the net propagates a batch of the training data through its layers in a forward pass ('forward propagation'), makes predictions in the last layer, and then optimizes the weights in a backward pass ('backward propagation').
In each experiment, we trained the net for 50-100 epochs (an epoch passes when the net had seen all training samples). During training we saved 5-10 'snapshots' of the current state of the learned weights, we then used these snapshots to classify the images from the validation set in order to get the predicted label and class-probabilities for each image.

Hyperparameters Tuning

Hyperparameters determine how the net will update its weights during the back propagation stage. A right configuration of the hyperparameters is necessary for the learning to converge (that is, if learning is possible). A wrong configuration of the hyperparameters will cause the error rate to diverge. CNNs are configured for Big-Data so the hyperparameters are adjusted for fast and aggressive learning. Caffenet came with its original configuration that was used to train it on 1.2 million images. Our data set consists of 257 images. It requires a much more subtle learning process. For the learning process to converge, we had to fine-tune some of the hyperparameters. Since there are almost no guidelines on how to fine-tune hyperparameters for transfer learning with a data set of this size we are going to list our configuration:

* **Learning rate:** Determines the 'step size' that the net will take when following the gradient descent. Too large steps could make it miss 'slopes' or 'canyons' (minima or ways to get to minima) and/or get completely lost in the error-landscape. Too small steps could lead to longer training times in the best case. In the worst case, small steps could lead to the net getting stuck in local minima or saddle points and prevent it from reaching adequate minima.
Caffenet learning rate was 0.1. We lowered it by a factor of 100 to 0.001.

* **Momentum:** Supposed to simulate the physical properties of momentum, i.e., factor previous weight updates into the current next. The usual value, 0.9, did not work for us, possibly because of the small size of our training set. It is possible that momentum made the learning too aggressive. Furthermore, Momentum might be more useful with big training sets that do not fit in one training batch like ours.
Caffenet Momentum was 0.9. We set it to 0.5.

* **Batch size:** How many images the net can learn in a single iteration. The larger the batch size the more stable the gradient, since SGD averages the gradient over all training samples in the batch. Our small data set allowed us to use the optimal batch size, which is the size of the entire training set. Caffenet was trained on 1.2 million images with a batch size of 128.

* **Weight decay:** decrease the size of the weights after each iteration. Prevents over-fitting. Supposed to bigger for smaller training sets.
Caffenet weight decay was 0.0005. We increased it to 0.001.

* **End to End fine-tuning:** We chose to resume training of all layers and increase the learning rate of the last layers as opposed to freezing the state of the first layers and only train the last ones. However, both methods worked.

* **Individual layer learning rates:** caffenet has individual learning rate multipliers in each layer. We increased the learning rate of the last layer by a factor of 20, the second to last layers by a factor of 10, and third to last by a factor of 5.

* **Learning rate decay:** The learning rate should decrease when the learning converges, in order to allow the net to settle on minima.
Keeping the learning rate fixed through out the training worked for us.

These configurations were mostly successful but we ended up using the Adam optimization method (Kingma et al. 2017) which adjusts the learning rate dynamically during training. We used Caffe's lenet Adam solver (configuration). Unfortunately, the solver did not work 'out of the box' and we had to lower its base (initial) learning rate from 0.001 to 0.0001. After that, we did see faster convergence with more aggressive overfitting after reaching the peak of the learning process (which is good since you have no doubt about when your net started to over fit).

Data Preparation

As stated, the most important constrain on this project is not to use feature engineering. Our images are eye close-ups of size 587X440. They include eyelids and lashes. Caffenet expects images of size 256X256 for training (internally, it augments the training data by making five different crops of size 227X227). So we cropped the images to 256X256 in a naïve manner, i.e., without centering on the white dot projected by the imaging device.

Results

Results Summary

After we found a working configuration for the Hyperparameters and the learning converged, the net achieved high accuracy rates on the Healthy, KC and Regular Astigmatism classes in short training times (the training took 5 minutes in each cross-validation iteration). The net classified Healthy vs. KC with 97% accuracy. The overall classification accuracy on the Healthy, KC, and Regular Astigmatism classes was 93.6% - where Regular Astigmatism class was classified with 83% accuracy. Classifying the KC suspects proved to be hard. In the best experiment, 32% of the suspects were correctly classified as suspects and 54% were classified as non-healthy. 45% of the suspects were classified as Healthy.

Experiments

Terminology

* **Training Loss:** A measurement of the classification error during training. Measures how well the net learns to classify the training set. Smaller values mean less error. Should start high and decrease in time. Derived from the net's weights.

* **Test Loss:** Same as training loss only on the validation set.

* **Test Accuracy:** The portion of correct predictions during validation:

$$\frac{correct\ predictions}{number\ of\ predictions}$$

* **Overfitting:** A state where a continuous decrease in training loss is not followed by an increase in test accuracy. The net 'over learns' the training data. It starts to learn the specific details of the training samples instead of the general rules that will help it classify the validation set. If the training set is a little different from the validation set – learning the specifics of the training set will hurt the performance on the validation set.

* **Confusion Matrix**: A table used to measure the classifier performance. The table shows the classifier predictions by classes. The diagonal of the table shows correct predictions e.g., Healthy was classified as Healthy, while any other cell shows some wrong predictions, e.g., Healthy was classified as class KC.

* **Best epoch:** We deemed the best epoch to be the one with maximum test accuracy and minimal overfitting. For each experiment, we will display the confusion matrix for the best epoch.

* **Learning Graph legend:**

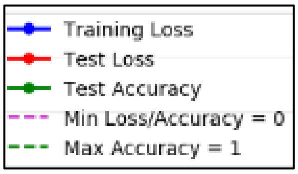


Figure 11. Learning graph legend.

* **CLY:** The Regular astigmatism data set is sometimes referred to as 'CLY', which is short for Cylinder that is the type of corrective lens used to correct regular astigmatism vision defects.

Healthy vs. KC

103 images of healthy and 86 images of KC. In each cross-validation iteration, there were 10 images in validation set and 179 images in the training set.

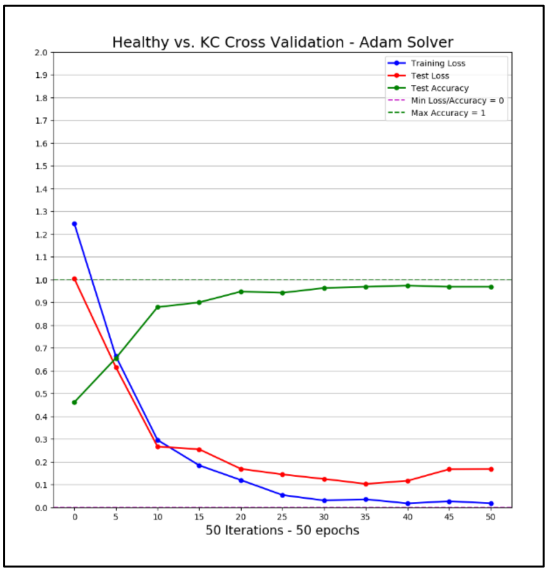
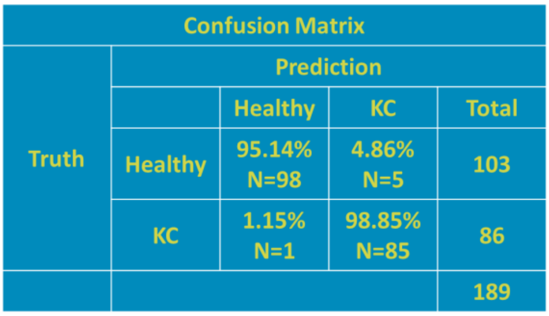


Figure 12. Graph showing Healthy vs. KC performance measurements by epochs. Maximum accuracy was 97%.



Confusion Matrix				
	Prediction			
		Healthy	KC	Total
Truth	Healthy	95.14% N=98	4.86% N=5	103
	KC	1.15% N=1	98.85% N=85	86
				189

Table 1. Healthy vs. KC Confusion Matrix of epoch 35.

This first experiment yielded very good results. The net achieved very high accuracy on a small data set of unconstrained images in short training time of 5 minutes. Only one KC image was classified as healthy while five healthy images were classified as KC.

On Amazon GPU instance, it took the net approximately two and a half minutes to reach maximum accuracy before it started overfitting (the increase in test loss after the 35th iteration). On a PC, it took between an hour and two hours.

Minimum Number of Images Required to Train the Net on Healthy vs. KC

After the success of the first experiment, it was interesting to check the minimum number of images required to train the net to separate between Healthy and KC. We reached a conclusion that approximately 15 images of each class could be enough to train the net for this new task but it is by no means as stable as training with more data.

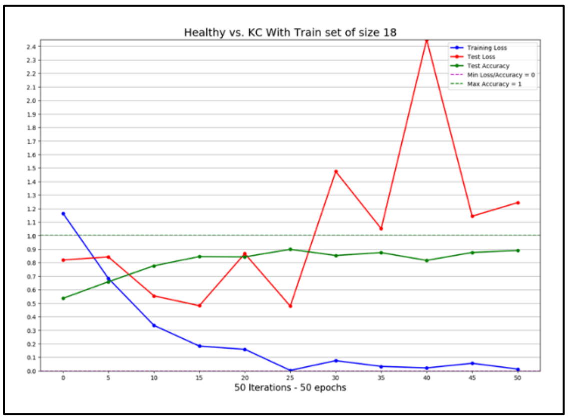


Figure 13. Training the net on healthy vs. KC with 18 images in the validation set. The net does reach 90% accuracy but the learning isn't stable and there's strong overfitting.

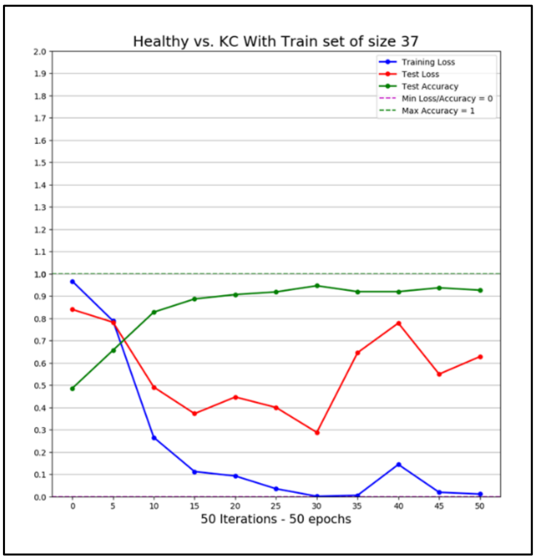


Figure 14. Training the net on healthy vs. KC with 37 images in the validation set. The net reaches 95% accuracy and the learning is more stable.

Healthy vs. KC vs. KC Suspects

103 images of healthy, 86 images of KC, and 37 images of KC Suspects.
The net did not manage to classify the KC suspects.

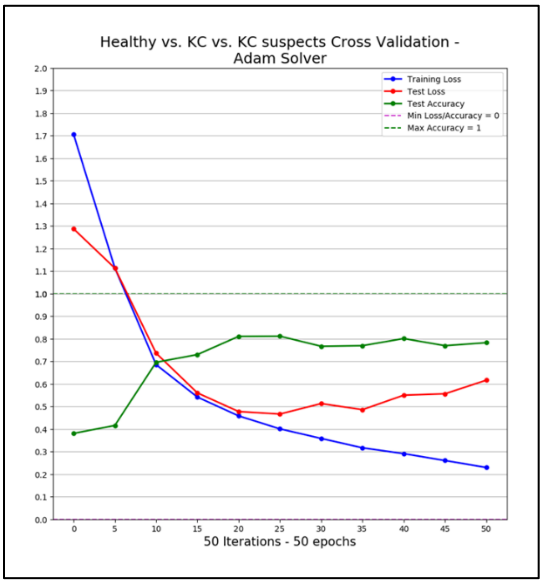
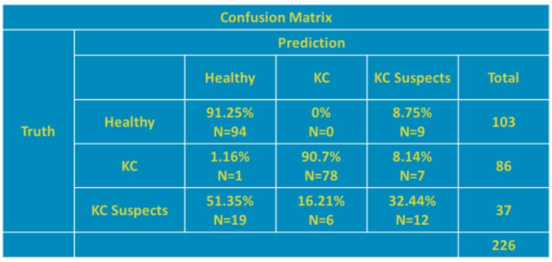


Figure 15. Healthy vs. KC vs. KC Suspects performance measurements by epochs. Adding KC Suspect reduced the overall accuracy to 80%.



Confusion Matrix					
	Prediction				
		Healthy	KC	KC Suspects	Total
Truth	Healthy	91.25% N=94	0% N=0	8.75% N=9	103
	KC	1.16% N=1	90.7% N=78	8.14% N=7	86
	KC Suspects	51.35% N=19	16.21% N=6	32.44% N=12	37
					226

Table 2. Healthy vs. KC vs. KC Suspects Confusion Matrix - Epoch #25.

51% of the Suspects were classified as healthy. Introducing the suspects set did some damage to the learning of the healthy class – 9 healthy images were classified as KC and 7 KC images were classified as KC Suspects (although these errors are of less concern).

Healthy vs. KC and KC Suspects

Another viable option was to label the KC Suspects as sick and see if this approach yields higher accuracy. After all, it might be enough to know that something is wrong with the suspects, so that they could be further examined.

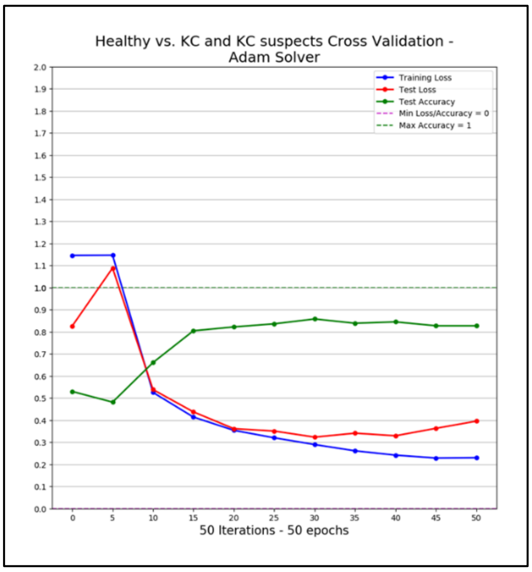
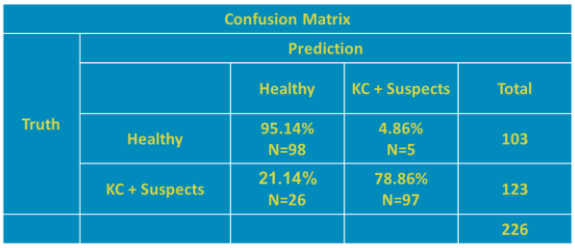


Figure 16. Healthy vs. KC + KC Suspects performance measurements by epochs. Although the accuracy reached 86%, Closer examination of the classifier predictions show that, the performance is worse than when Suspects were labeled as Suspects and not as KC.



Confusion Matrix				
		Prediction		
Truth		Healthy	KC + Suspects	Total
	Healthy	95.14% N=98	4.86% N=5	103
	KC + Suspects	21.14% N=26	78.86% N=97	123
				226

Table 3. Healthy vs. KC + KC Suspects Confusion Matrix - Epoch #30.

Although the overall accuracy is higher, the performance in this experiment was in some ways worse than that of the healthy vs. KC vs. KC Suspects experiment.

When the suspects were labeled as suspects, nine healthy images and seven KC images were labeled as suspects. Those errors accounted for 17% error rate. However, they are less serious than classifying KC suspects as healthy. In this experiment, out of the 26 KC and suspects that were classified as healthy 25 were suspects, while in the previous experiment it was 19. That means that 67.5% of the suspects were classified as healthy compared to 51% in the previous experiment. The conclusion is that giving the net the option to classify the suspects as suspects is better than labeling them as KC. This result is intuitive – labeling the suspects as KC forces the net to choose between two categories so that if a suspect image looks more healthy than sick, the net will choose to classify it as healthy. However, if that suspect image looks more suspicious than healthy, then given a choice, the net might classify it as a suspect.

The Problem Posed by KC Suspects

We conducted tens of experiments with different hyperparameters but we could not pass the 50% accuracy bar for the suspects. Except for hyperparameters, the problem could also be accounted for class imbalance. The number of suspect images is much lower than the number of healthy and KC images. Therefor, the net makes more weight optimization for the healthy and KC classes. The question is how much it affects the learning process. To test this hypothesis, we augmented the KC suspects set with variations of the images. Augmenting the input data is a common procedoure. It is suppose to make the net more resiliant to small variations in the data.

Healthy vs. Augmented KC Suspects

We created two variations of each KC suspect image, one blurred and one with high contrast.

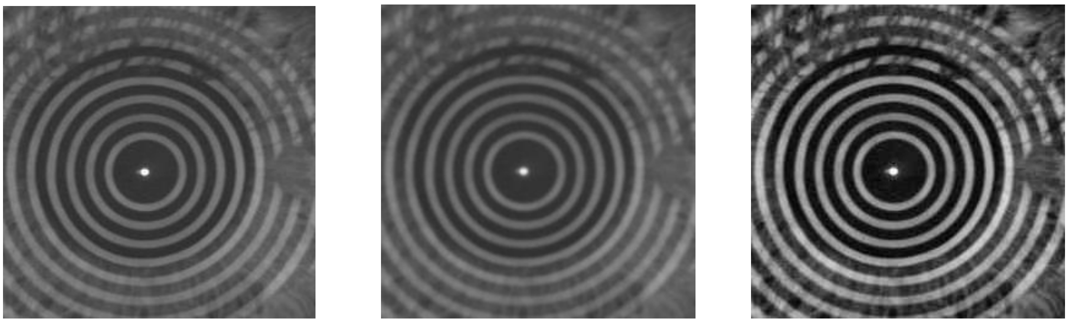


Figure 17. Original and augmented versions of one KC suspect image. Left: original, middle: blurred version, right: high contrast version.

The result was KC suspects set with 111 images. We then performed cross-validation routine of 103 healthy images vs. the augmented KC suspects set. Note that we although we trained on the both the original and augmented images, we only validated the original images.

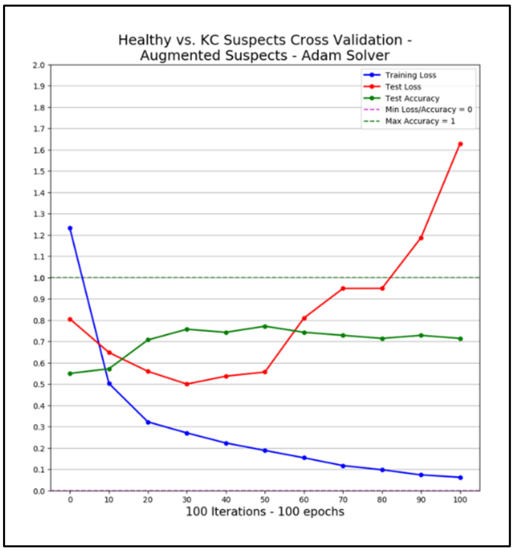
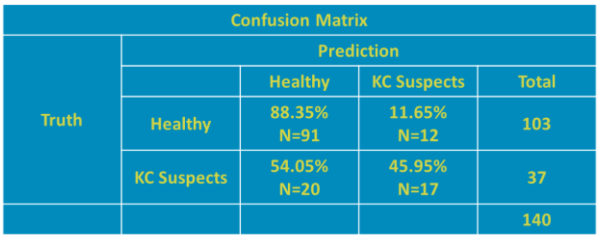
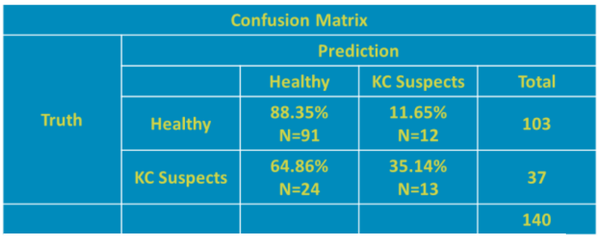


Figure 18. Healthy vs. augmented KC Suspects performance measurements by epochs. Adding KC Suspect reduced the overall accuracy to 80%.



Confusion Matrix				
	Prediction			
		Healthy	KC Suspects	Total
Truth	Healthy	88.35% N=91	11.65% N=12	103
	KC Suspects	54.05% N=20	45.95% N=17	37
				140

Table 4. Healthy vs. augmented KC Suspects Confusion Matrix - Epoch #30. Only original images were validated.



Confusion Matrix				
	Prediction			
		Healthy	KC Suspects	Total
Truth	Healthy	88.35% N=91	11.65% N=12	103
	KC Suspects	64.86% N=24	35.14% N=13	37
				140

Table 5. For comparison, confusion-matrix of Healthy vs. the original KC Suspects set - Epoch #20.

The results show that the augmentation did not help the net to classify the suspects. 54% of the suspects were labeled as Healthy while 11.6% of the healthy were labeled as suspects. This is a slight improvement from the healthy vs. the original KC suspects set experiment where 64% of KC suspects were labeled as healthy.

The problem with the failure to classify the KC suspects is that there is always the possibility that some configuration or some augmentation might bring the break through. We spent a good part of the project trying different approaches.

At last, some form of break through came with the addition of the regular astigmatism data set. The break through was not in succeeding to classify the suspects, instead, the success with the regular astigmatism data set shed some light on the KC suspects problem.

The regular astigmatism data set consists of 31 images. Many of those images display only subtle deformations. To the naked eye, most of them are visually close to the healthy class. On the other hand, the images that do display asymmetry, show different kind of irregularities than KC. The net will have to learn how to classify not only by the symmetry of the Placido disks, but also by the type of skewness. Therefor, the regular astigmatism data set also looks hard to classify.

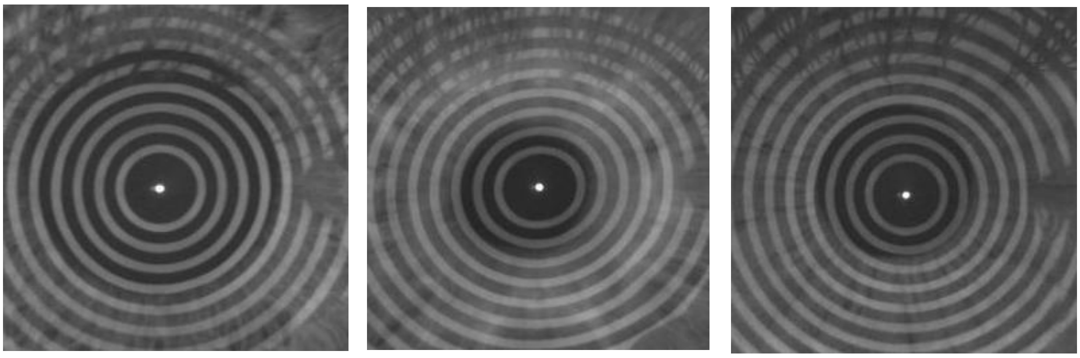


Figure 19. 3 Placido images demonstrating the Regular Astigmatism set. Left: Regular astigmatism that resembles a healthy cornea. Middle: Regular astigmatism that shows a horizontal 'bow-tie' deformation. Right: Regular astigmatism with slight deformations.

Healthy vs. KC vs. Regular Astigmatism

103 healthy images, 86 KC images, and 31 regular astigmatism images. The net showed good results in this experiment. Only 10% of the regular astigmatism images were classified as

healthy. The net reached 94% accuracy.

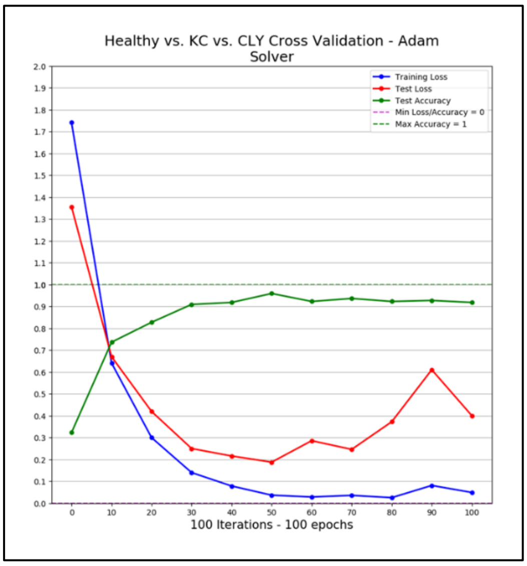
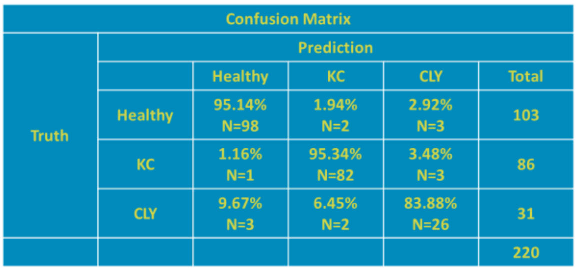


Figure 20. Healthy vs. KC vs. regular astigmatism performance measurements by epochs.



Confusion Matrix					
	Prediction				
		Healthy	KC	CLY	Total
Truth	Healthy	95.14% N=98	1.94% N=2	2.92% N=3	103
	KC	1.16% N=1	95.34% N=82	3.48% N=3	86
	CLY	9.67% N=3	6.45% N=2	83.88% N=26	31
					220

Table 6. Healthy vs. KC vs. regular astigmatism Confusion Matrix - Epoch #50.

Healthy vs. KC vs. KC Suspects vs. Regular Astigmatism

103 healthy images, 86 KC images, 37 KC Suspects images, and 31 regular astigmatism images. The results are similar to previous experiments involving KC suspects, with minor improvements. In this experiment 54% of the KC suspects, we are classified as one of the non-healthy classes. The suspects set damaged the learning of the other sets, lowering the accuracy to 80%.

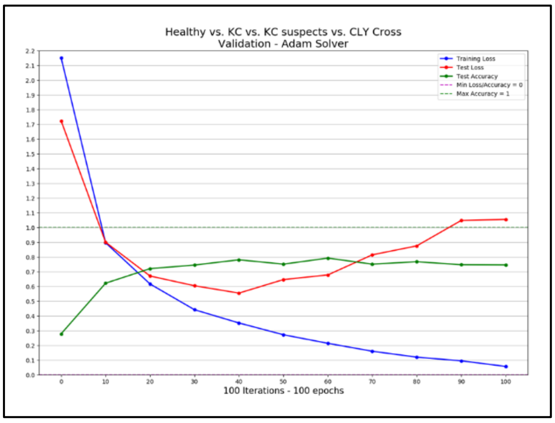
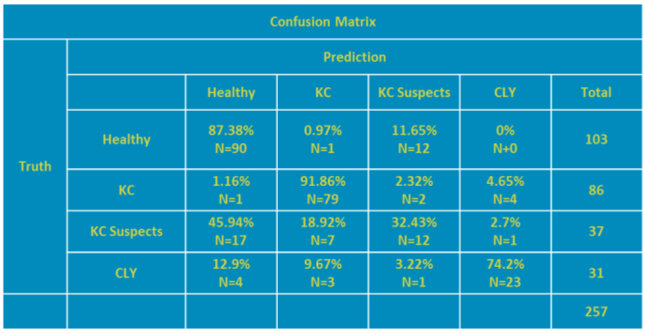


Figure 21. Healthy vs. KC vs. KC Suspects vs. regular astigmatism performance measurements by epochs.



Confusion Matrix						
	Prediction					
		Healthy	KC	KC Suspects	CLY	Total
Truth	Healthy	87.38% N=90	0.97% N=1	11.65% N=12	0% N+0	103
	KC	1.16% N=1	91.86% N=79	2.32% N=2	4.65% N=4	86
	KC Suspects	45.94% N=17	18.92% N=7	32.43% N=12	2.7% N=1	37
	CLY	12.9% N=4	9.67% N=3	3.22% N=1	74.2% N=23	31
						257

Table 7. Healthy vs. KC vs. KC suspects vs. regular astigmatism Confusion Matrix - Epoch #40.

Healthy vs. Regular Astigmatism

For comparison with healthy vs. KC suspects:
103 images of healthy vs. 31 images of regular astigmatism. The net achieved 96% accuracy.

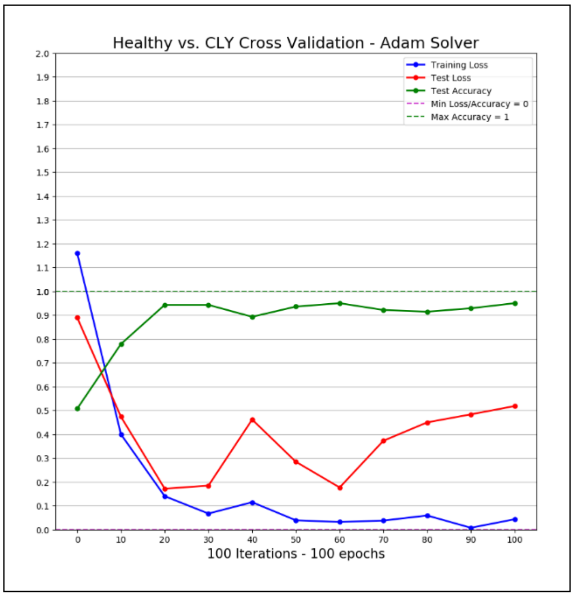
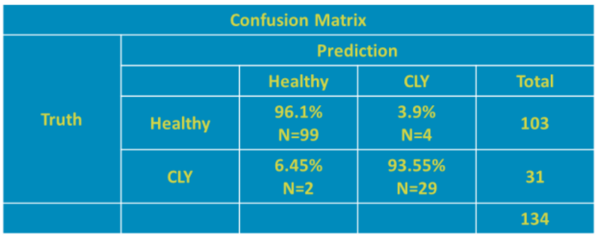


Figure 22. Healthy vs. regular astigmatism performance measurements by epochs.



Confusion Matrix				
Truth		Prediction		
		Healthy	CLY	Total
	Healthy	96.1% N=99	3.9% N=4	103
	CLY	6.45% N=2	93.55% N=29	31
				134

Table 8. Healthy vs. regular astigmatism Confusion Matrix - Epoch #20.

Conclusions

1. Using transfer learning on the healthy, KC, and Regular Astigmatism data sets proved to be extremely useful. Healthy and KC classes were classified with state-of-the-art accuracy without any feature engineering. Healthy, KC, and Regular Astigmatism were classified with 93.6% accuracy. While 10% of the regular astigmatism were classified as healthy, it is possible that adding more images of that class will improve the performance.

2. The success in the classification of the regular astigmatism data set offers some insights on the failure to classify the KC suspects. The regular astigmatism data set is smaller than that of the KC suspects. Moreover, many of the images in that set are more visually close to the healthy class than to the KC class. Those images display mild corneal asymmetry, far less dramatic than the asymmetry introduced by KC. Nevertheless, this dataset was classified with much higher accuracy. The accuracy of healthy vs. regular astigmatism was 96% while the accuracy of healthy vs. KC suspects was 72%. The accuracy of healthy vs. the augmented KC suspects set was 76%.

These observations suggest couple of things:

2.1. Caffenet can be resilient to class imbalance. There is no hard correlation between the degree of class imbalance and caffenet performance. During training on the healthy and regular astigmatism data sets, caffenet saw 3 times more healthy images. However, it did not prevent it from achieving 96% accuracy.

2.2. Caffenet is able to extract prominent features in order to classify the input data, even when those features are subtle. When classifying regular astigmatism vs. Healthy vs. KC, the net needs to learn two concepts: It needs to learn that subtle irregularities separate the astigmatism from the healthy. It also needs to learn the differences between the irregularities displayed by KC and regular astigmatism.

The success with the regular astigmatism set implies that the Placido images do not hold enough visual information that would enable the net to learn the difference between healthy and KC suspects.

Future Work

Future Work Regarding the Classification Project

1. Gather more images of regular astigmatism and other corneal disorders.

2. Categorize KC suspects into sub-categories:
   a. KC suspects that were diagnosed based on their corneal thickness only.
   b. KC suspects with thin cornea and curvature irregularities.
   c. KC suspects with curvature irregularities and normal corneal thickness.

If Placido images do not hold information regarding corneal thickness, then sub-category a might be unclassifiable.

The suspect's parameters should be extracted from the comprehensive corneal topography maps that were used to diagnose their condition.

3. Using more advanced CNN architecture. Caffenet is a variant of the AlexNet CNN architecture that was developed in 2012 (Krizhevsky et al. 2012). Since then, new architectures were developed that achieved far better results in ILSVRC.

Future Work Regarding the Umbrella Project

The ability to detect KC suspects with Placido images is very limited. The umbrella project faces two options:

1. Use Placido camera as a compact and cheap imaging technique – thus giving-up on trying to detect KC suspects based on corneal thickness.
2. Search for an alternative imaging technique that could extract data about the corneal thickness.

References

Li, Xiaohui, Huiying Yang, and Yaron S. Rabinowitz. “Keratoconus: Classification Scheme Based on Videokeratography and Clinical Signs.” Journal of cataract and refractive surgery 35.9 (2009): 1597–1603. PMC.
Retrieved: 30 June 2017.

Nordqvist, Christian. "Astigmatism: Causes, Symptoms, and Treatments." Medical News Today. MediLexicon International, 28 June 2017.
Retrieved: 29 June 2017.
http://www.medicalnewstoday.com/articles/158810.php

Ng, Andrew. "Machine Learning and AI via Brain simulations". Stanford University. 2015-03-23.
Retrieved: 30 June 2017.
https://forum.stanford.edu/events/2011/2011slides/plenary/2011plenaryNg.pdf

X. W. Chen and X. Lin, "Big Data Deep Learning: Challenges and Perspectives," in IEEE Access, vol. 2, no. , pp. 514-525, 2014. doi: 10.1109/ACCESS.2014.2325029.
Retrieved: 30 June 2017.
http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6817512&isnumber=6705689

Olga Russakovsky*, Jia Deng*, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg and Li Fei-Fei. (* = equal contribution) ImageNet Large Scale Visual Recognition Challenge. IJCV, 2015.
Retrieved: 30 June 2017.
http://image-net.org/

Miles F. Greenwald, Brittni A. Scruggs, Jesse M. Vislisel, Mark A. Greiner. Corneal Imaging: An Introduction. University of Iwoa Health Care. October 19, 2016.
Retrieved: 30 June 2017.
http://webeye.ophth.uiowa.edu/eyeforum/tutorials/Corneal-Imaging/Index.htm

Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama: “Caffe: Convolutional Architecture for Fast Feature Embedding”, 2014; arXiv:1408.5093.
Retrieved: 30 June 2017.

Donahue, Jeff. GitHub repository. Aug 13, 2014.
Retrieved: 30 June 2017.
https://github.com/BVLC/caffe/tree/master/models/bvlc_reference_caffenet

Anderson, Dianne. “Understanding Corneal Topography”. American Optometric Association.
Retrieved: 30 June 2017.
https://www.aoa.org/Documents/optometric-staff/Articles/Understanding-Corneal-Topography.pdf

Kingma, Diederik P., and Jimmy Ba. "Adam: A Method for Stochastic Optimization." [1412.6980] Adam: A Method for Stochastic Optimization. 3rd International Conference for Learning Representations, 30 Jan. 2017.
Retrieved: 30 June 2017.
https://arxiv.org/abs/1412.6980.

Krizhevsky Alex, Sutskever Ilya, Hinton E. Geoffrey. “ImageNet Classification with Deep Convolutional Neural Networks”. Advances in Neural Information Processing Systems 25 (NIPS 2012). 2012.
Retrieved: 28 July 2017.
https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

Babi Pavel. “Automated Diagnosis of Keratoconus”. Final Project in the department of CS in Hadassah Academic College in Jerusalem. 2015.
No retrieval date.

Ifergan Aviad, Koretz Shmuel. “Classifying Keratoconus vs. Healthy using crowd-wisdom”. Final Project in the department of CS in Hadassah Academic College in Jerusalem. 2016.
No retrieval date.

Elharar Rotem. “Automated Diagnosis of Keratoconus”. Final Project in the department of CS in Hadassah Academic College in Jerusalem. 2016.
Retrieved: January 2017.