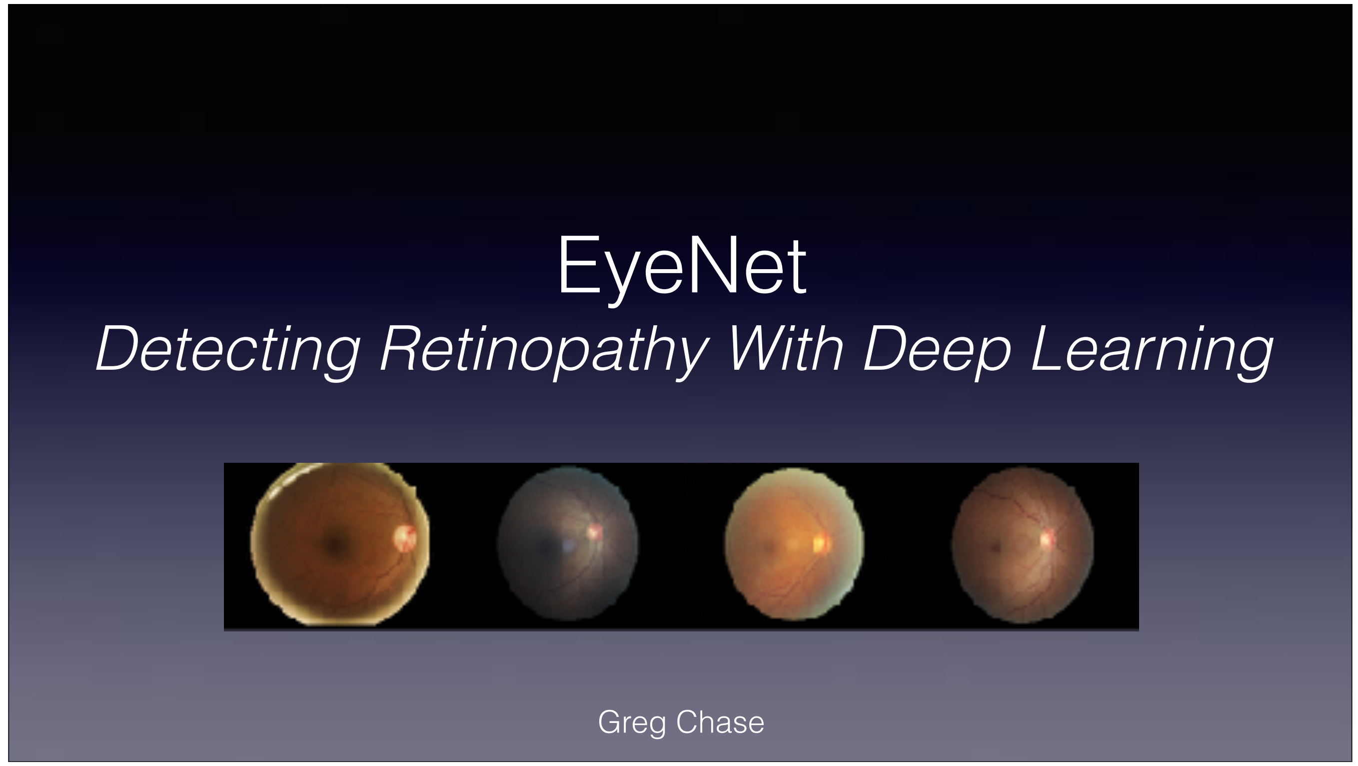
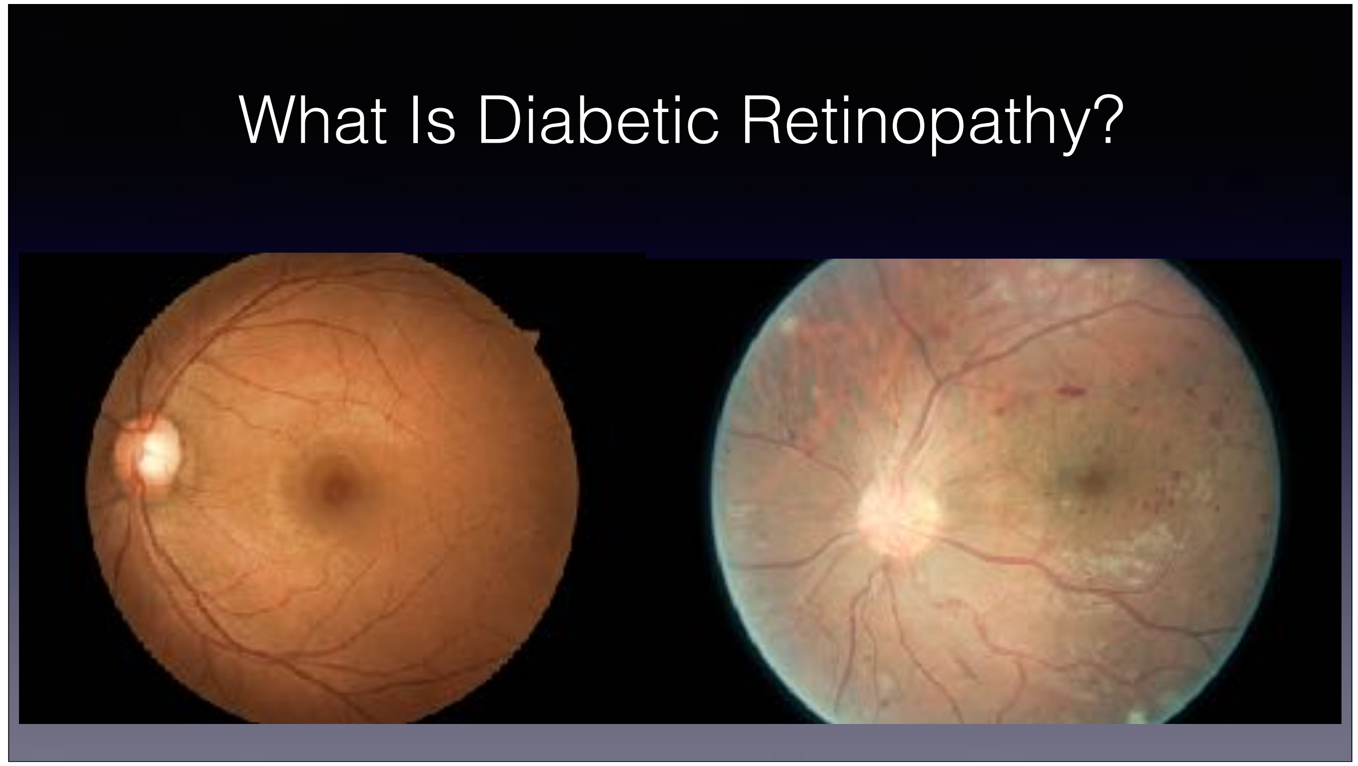


Hi everyone! My name is Greg Chase. Prior to Galvanize, I was a data engineer. But even during undergrad, I've always held a deep appreciation for doctors and what they do. During Galvanize, I decided to merge my interest in deep learning with the medical field. The product of that is EyeNet, a neural network that's designed to detect diabetic retinopathy.



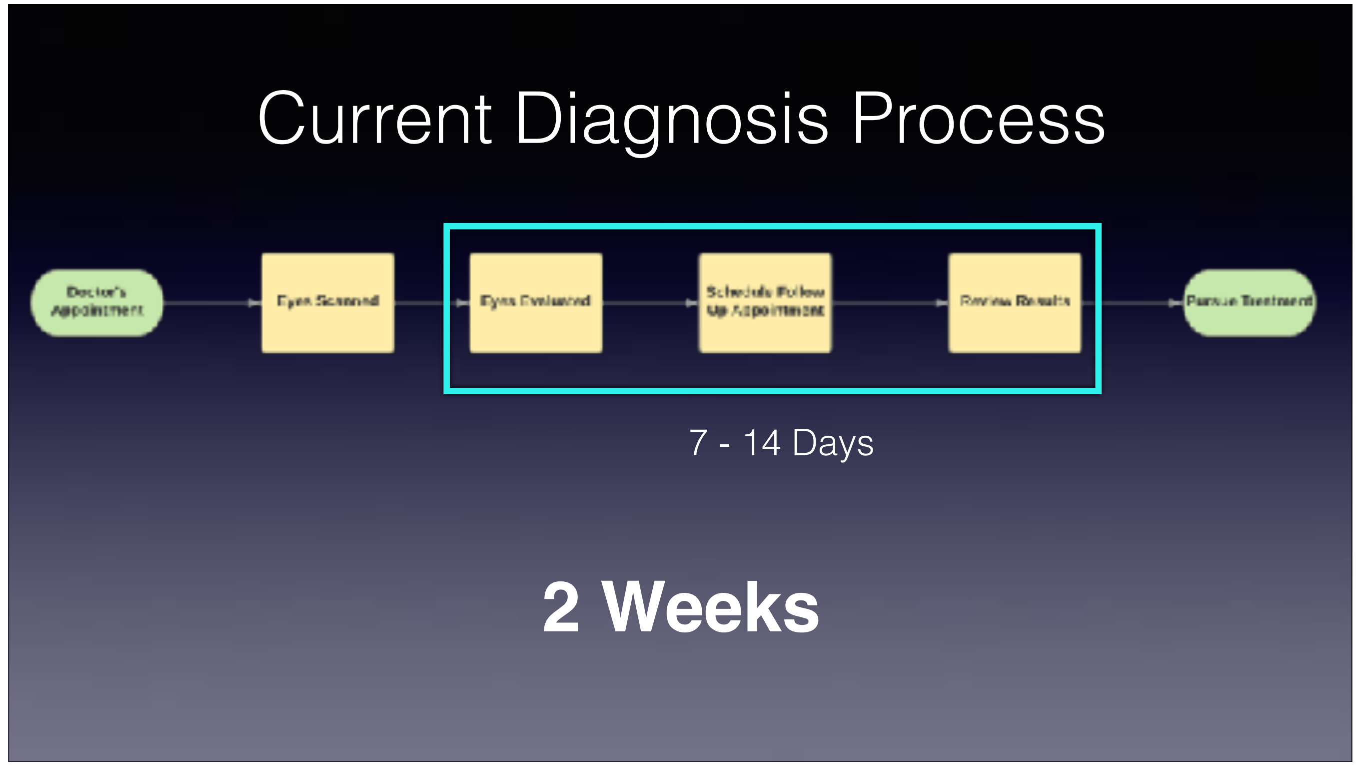
To those unfamiliar, diabetic retinopathy is a disease that affects the eyes of diabetics. Anyone with diabetes is at risk, and the probability of contracting this increases as you age.

The problem arises when it’s left untreated. You see a perfectly healthy eye on the left. But when left untreated, the disease advances into proliferative retinopathy. The blood vessels constrict, blood pockets form in the eye, and this results in what we know as blindness.

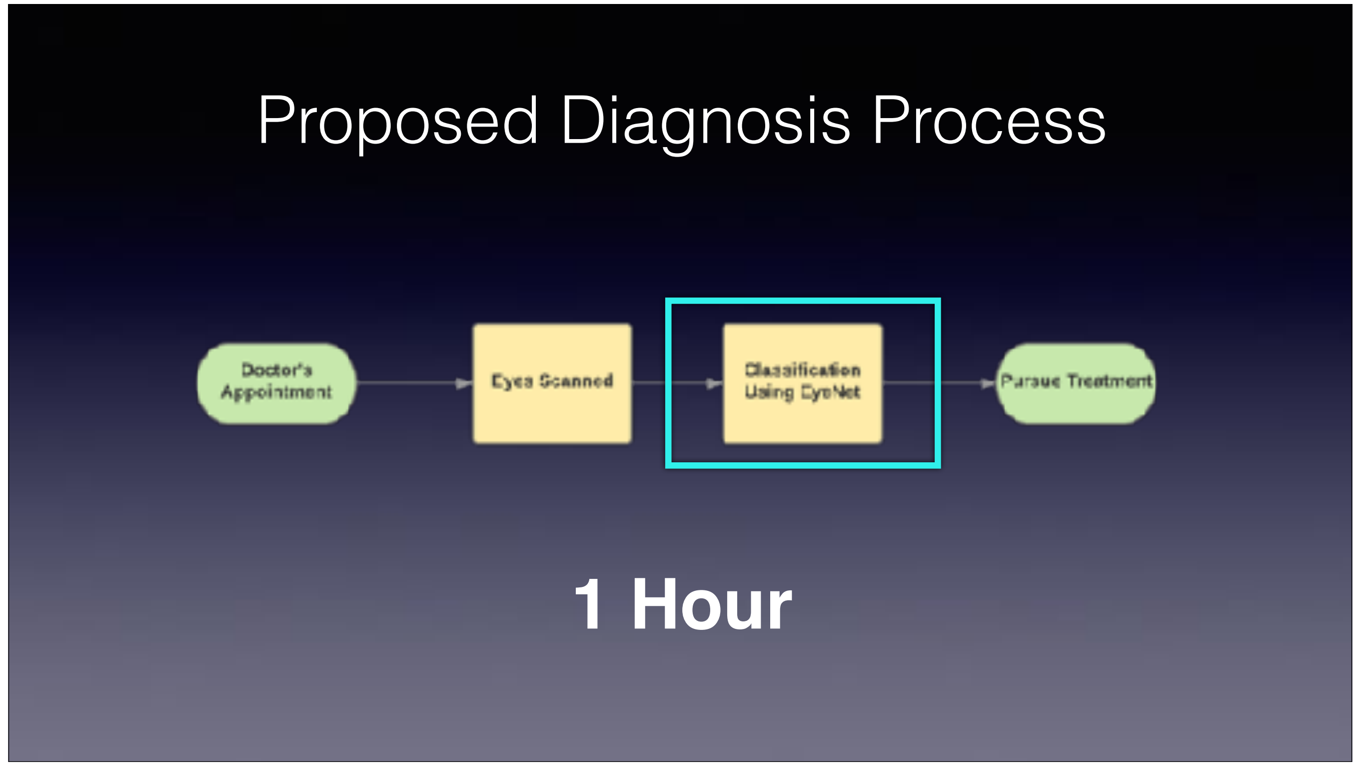
93 Million

Number of people living with retinopathy worldwide

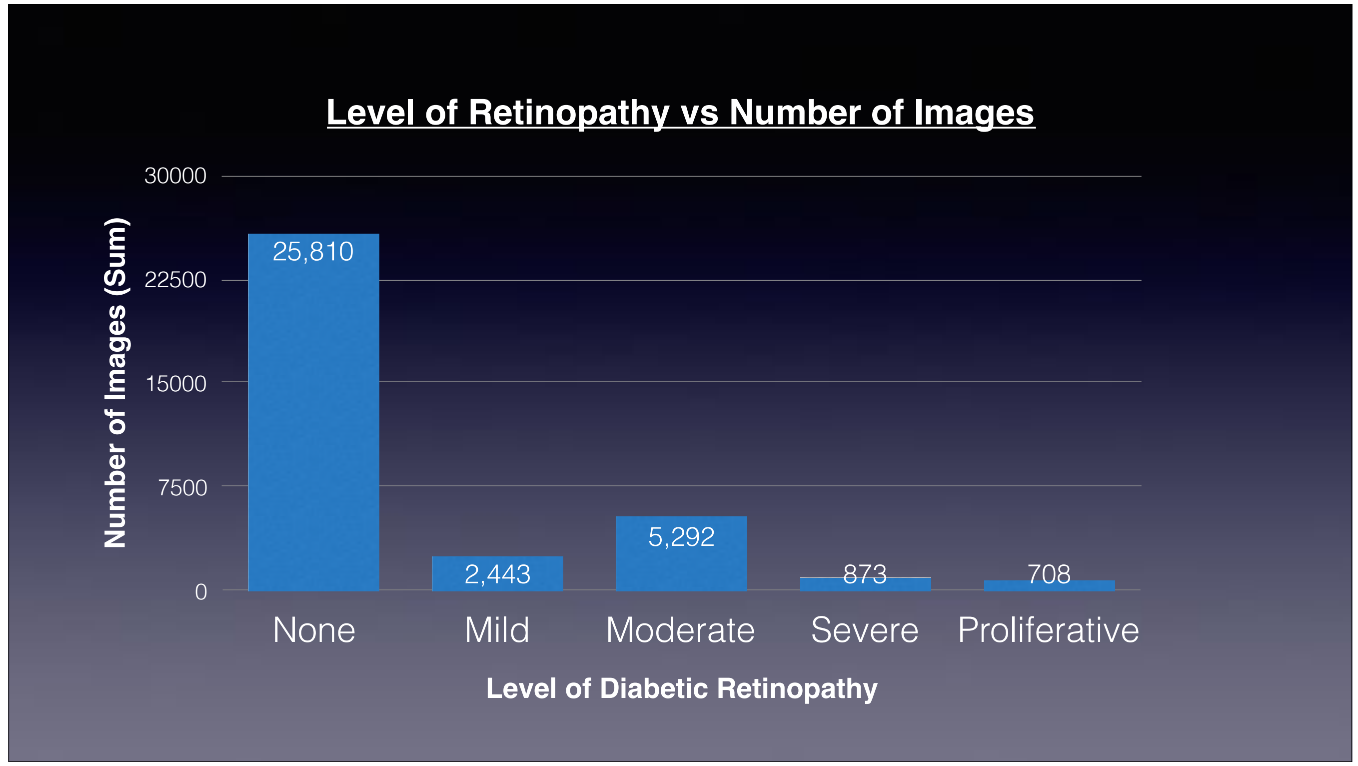
The number I want to start off with is 93 million. This is the number of people living with some degree of retinopathy worldwide.



The current diagnosis process is pretty long. You have to go in, get your eyes scanned. Then the doctor has to examine them manually. Due to doctor patient confidentiality, you can't disclose results over the phone. So you have to schedule a follow up appointment. As we all know, life happens, so you may reschedule. This could take 2 weeks, which isn't a rational time frame in 2017.



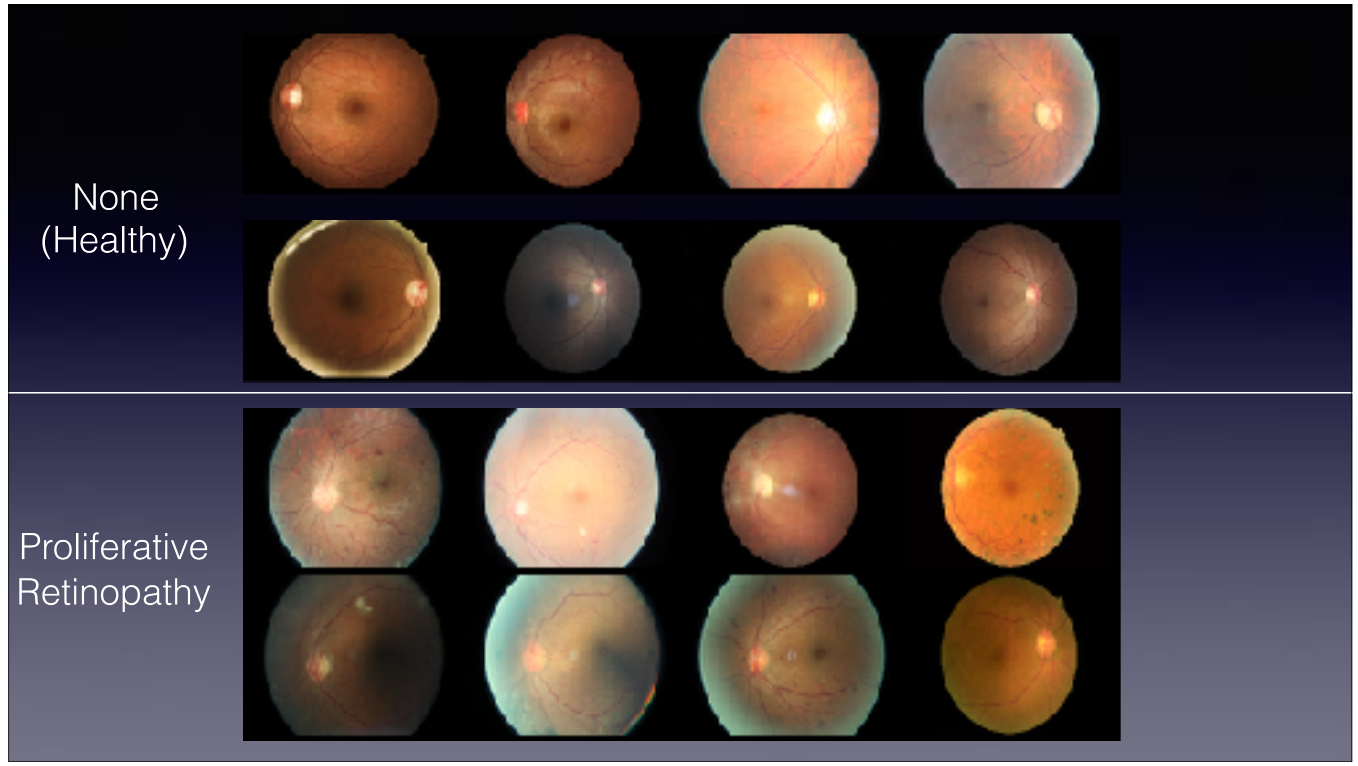
The proposed process would be the following. You go in, get your eyes scanned, and EyeNet would act as an assistant (not replacement) for your doctor. The total time, between filling out paperwork, getting your eyes scanned, and talking with your doctor, is 1 hour max. Orders of magnitude faster than the current process.



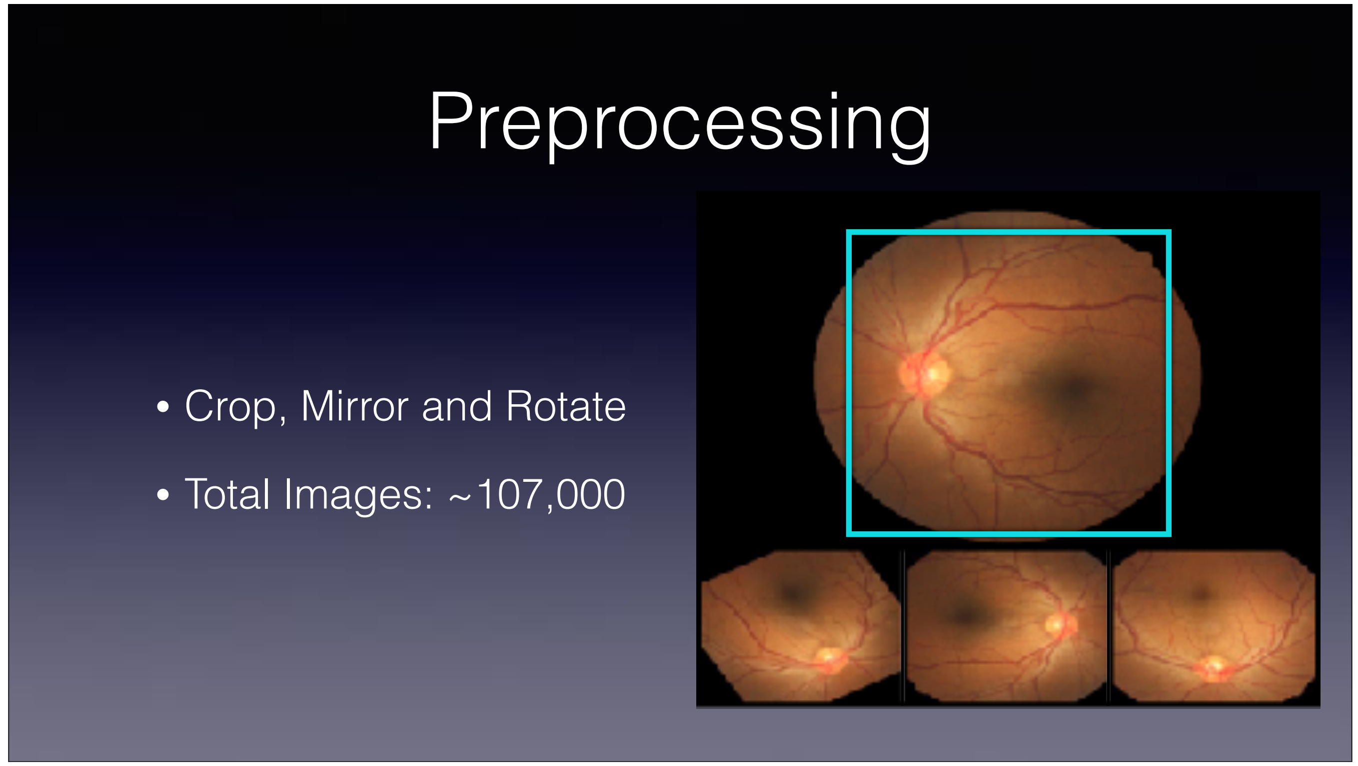
So, how do you solve a problem that affects this many people, and reduce the time frame to an hour?

We can do this with a convolutional network. Algorithms that are the gold standard for image classification.

Well, we first start with data. This is taken from a 2015 Kaggle challenge, but it's incredibly messy. The first issue is the class imbalance. The majority of people in this dataset don't have any sign of retinopathy!

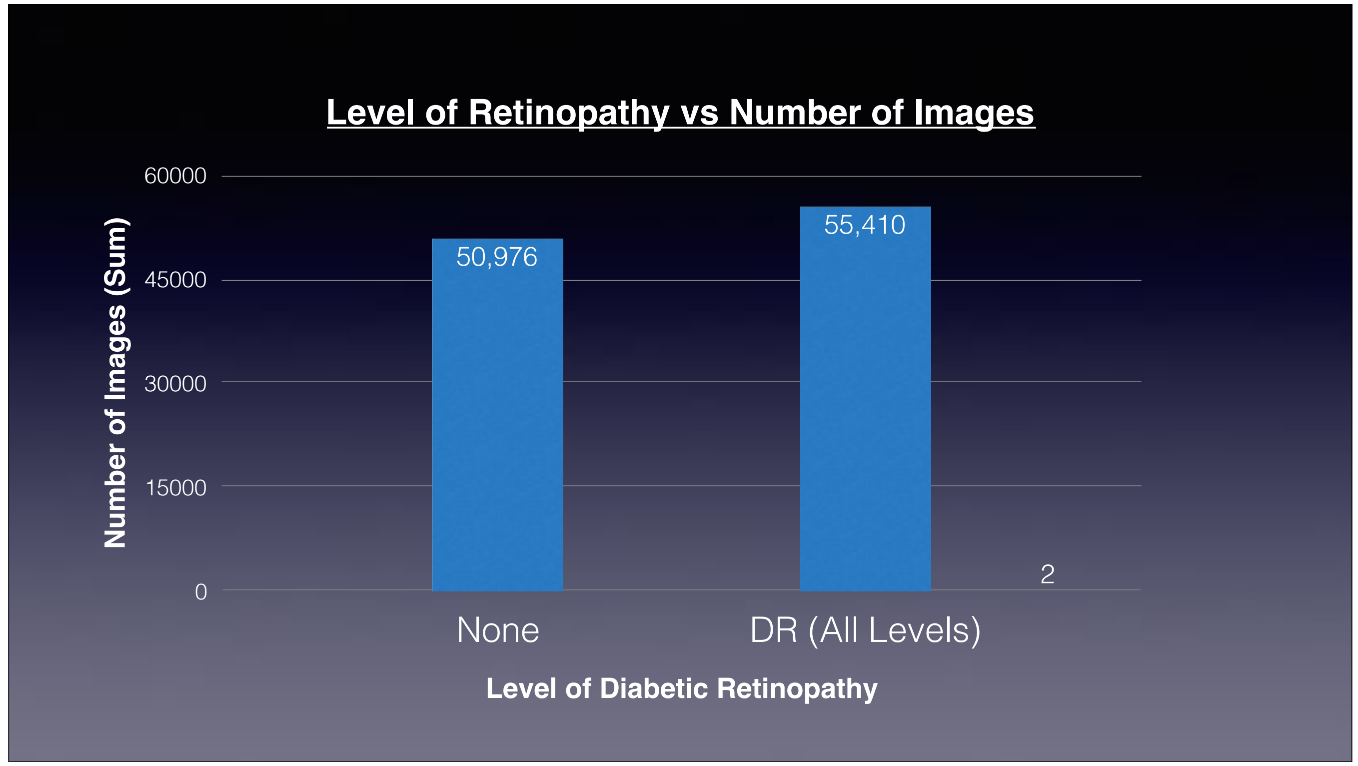


Additionally, the eyes have a huge amount of variance. To most of you in the audience, these look the same. But deep learning is so good, it can do it's own feature engineering to figure out the differences.

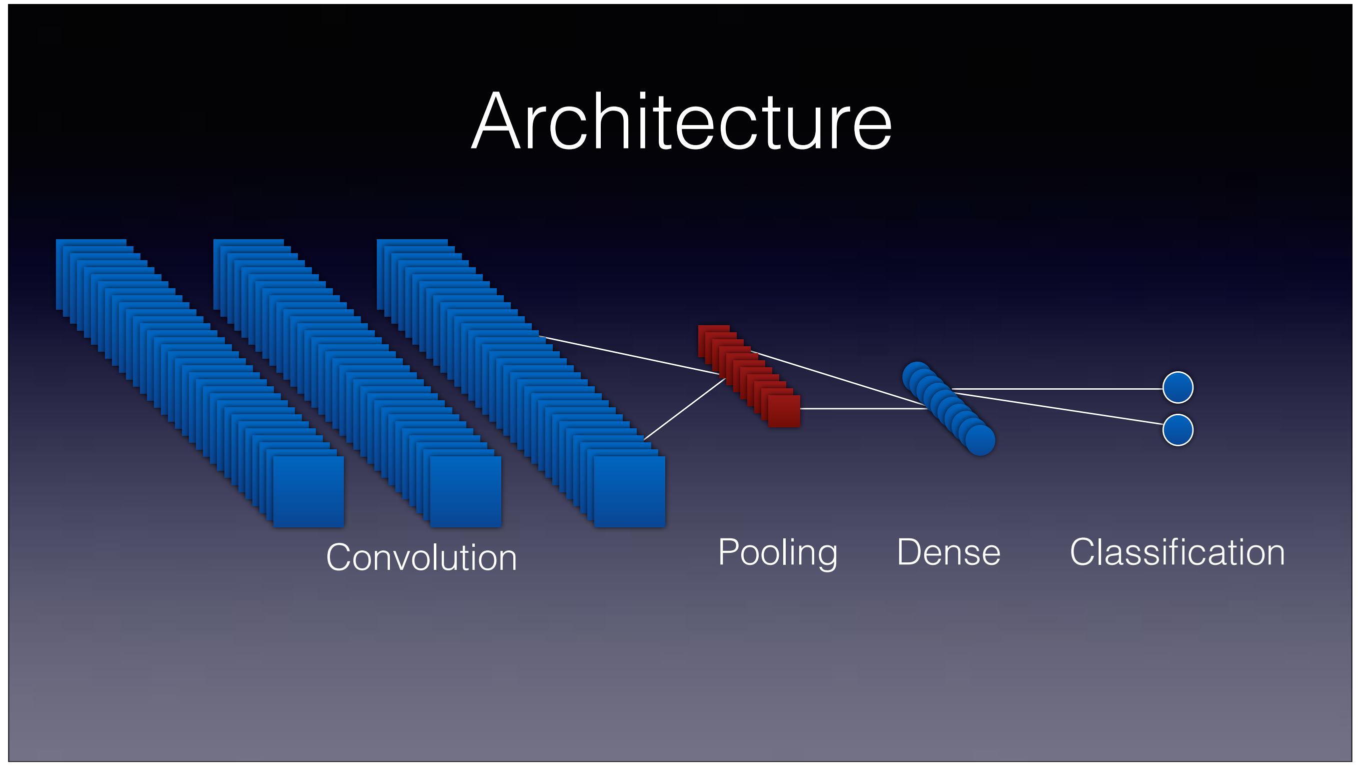


The biggest step (and breakthrough) was preprocessing. Instead of using the whole image, I cropped only the most important part of every image.

After this, I rotated and flipped all the images. This data augmentation allows us to take the total size from 35,000 images to 107,000 images.



Finally, I changed the classifier to be two categories, as opposed to five. The classifier now detects any degree of retinopathy, or not. You'll also noticed that the classes are much more balanced. The difference is now ~4,500 images.

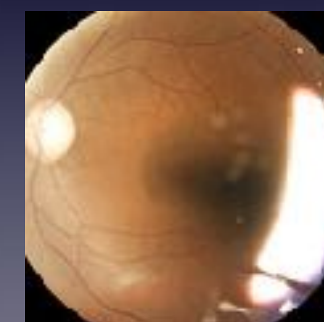


After preprocessing, the data gets fed through EyeNet. The architecture is pretty simple. Three convolution layers, all identical, followed by a pooling layer. The data is then taken through a dense layer, and finally a binary classifier, signifying if you have retinopathy or not.

Results

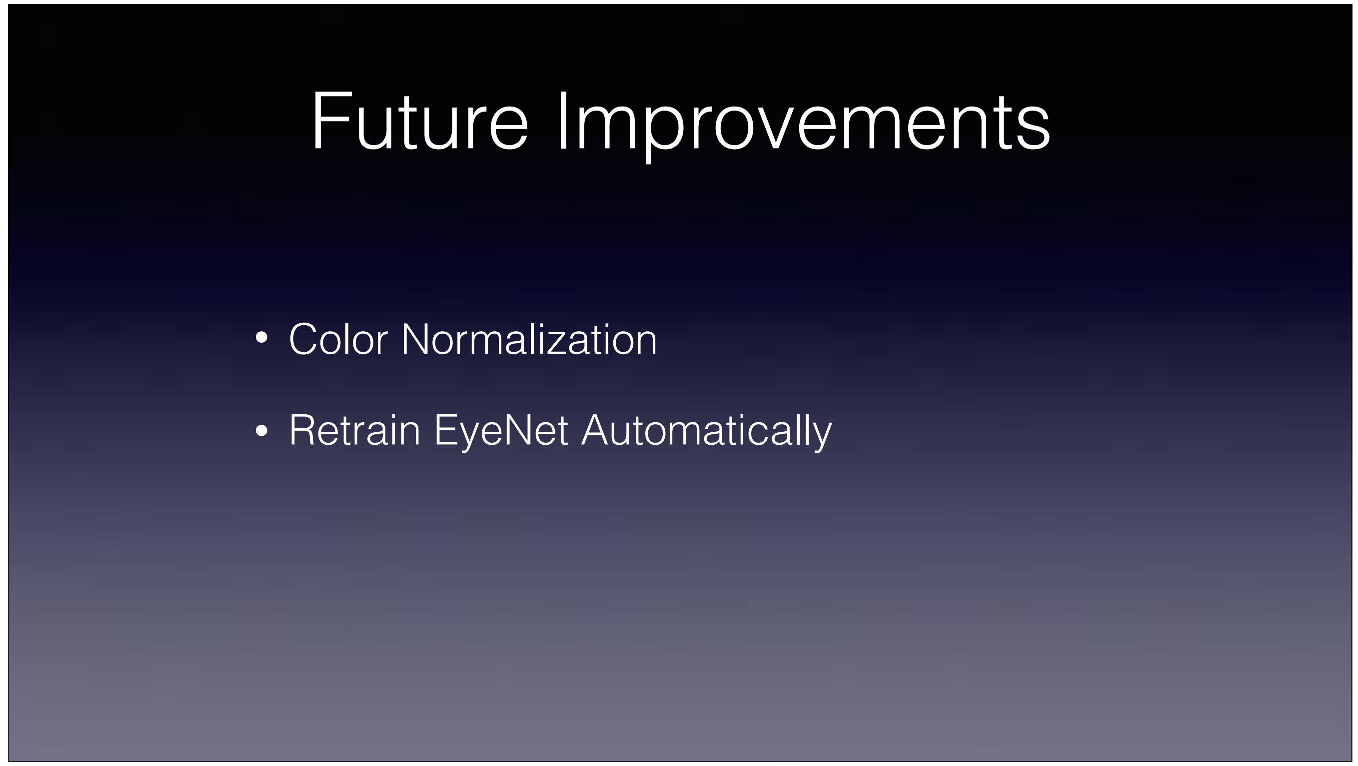
Metric	Value
Accuracy (Train)	82%
Accuracy (Test)	78%
Precision	85%
Recall	**77%**

- Image quality
- Dropping “bad” images



The results were pretty interesting. The accuracy is pretty good, but what we care about is the recall. In medicine, it’s better to be safe than sorry. So by classifying the true positives and false negatives, we’re able to screen more people, even if EyeNet thinks you have retinopathy in the slightest degree.

The results are in part due to two problems: image quality variance, and not dropping “bad” images. As you can see, the image on the left probably isn’t optimal for the classifier. So in EyeNet 2.0, I’d look to drop “bad” images, based on a few criteria, generally based on the color space of each image.



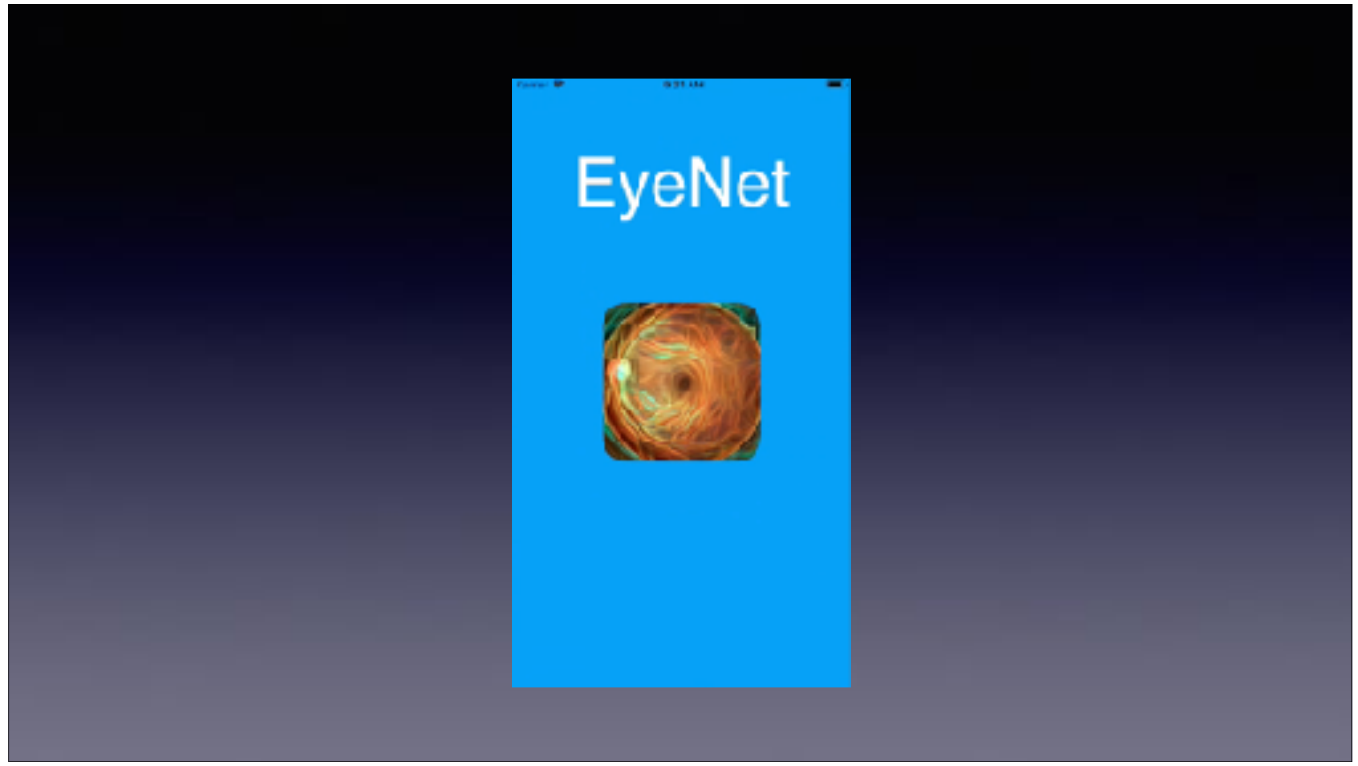
As for the future of EyeNet, their are two big steps that need to happen. The first is color normalization. A variety of techniques exist, but the first run through had mixed results. So moving forward, this is the big priority.

The second is to retrain EyeNet automatically. Based on all the criteria that defines a “good” image, EyeNet could retrain itself after every new classification, and learn the way humans do.

But, there is one more thing.

A lot of you probably have an iPhone in the audience. Hard to believe they were released 10 years ago. But today, we can actually do machine learning on the phone. So during the cohort, I embarked to try and develop a proof of concept on the iPhone.

It’s worth mentioning, this isn’t being processed on the cloud - these images can be processed on device! This can be done with CoreML, and architecture released by Apple in June 2017. You can write your model in native Python, and convert the trained model to a .mlmodel file. Some assembly is required in Xcode, but you now have the ability to put your machine learning models on a phone!



Here’s the proof of concept, and what it looks like. So imagine two images came through the office, and are now ready to be processed. The first shows no sign of retinopathy, and EyeNet is pretty sure of that. Conversely, the second image is the polar opposite, and it’s pretty confident of retinopathy existing!

Since this is processed on device, I want you to imagine something: the ability to take some equipment, plus your phone to remote areas around the world. You’d be able to provide treatment nearly anywhere, and get the treatment to those that need it the most.

In 2017, we have the ability to create a classifier that affects 93 million people, retrain it automatically, and the ability to do it on your phone.

Contact

in/gregwchase

github.com/gregwchase

@gregwchase

That’s EyeNet, I’m Greg Chase, and this is my contact information. If you have any questions, I’ll take them now.