Diagnosing Diabetic Retinopathy

Project Category: Life Sciences

https://github.com/jaigupta/cs229-proj

Blanco, Ignacio
iblanco@stanford.edu

Gupta, Jai
jaigupta@stanford.edu

Kaur, Amrita
amritak@stanford.edu

November 19, 2020

1 Abstract

In this work, we compare the use of different methodologies in the task of predicting the correct class for Diabetic Retinopathy, a medical condition where retina is damaged, from images. Apart from social impact, the problem is also interesting from technical perspective because it is expensive to obtain labelled dataset for such problems, but a large amount of unlabelled dataset may be avaialble. In this work, we use GANS to improve our results using semi supervised learning by making use of the unlablled dataset as well and compare it with various other approaches.

2 Introduction

Diabetic retinopathy (DR) eye disease is a medical condition where the retina is damaged due to diabetes. It is estimated to affect over 93 million people worldwide, and it is the leading cause of blindness in the developing world. 80% of the patients who have had diabetes for 20 years or more develop DR eye disease. This disease is hard to diagnose, because it is initially asymptomatic, until it is too late and the patient loses their vision. Currently, the detection of DR is a time-consuming and manual process that requires specialists and several days to yield a diagnosis, which leads to lost follow up, miscommunication, and delayed treatment. However, as the number of individuals with diabetes continue to grow, the current methods and techniques become insufficient.

The need for an automated and thorough detection process has been pointed out, and previous efforts have made some progress with image classification, pattern recognition, and machine learning, using digital color fundus photographs. Nonetheless, we propose using cutting-edge techniques that can optimize and improve the existing methods to obtain a more accurate and faster diagnosis. After obtaining results, we will compare our models with existing methods to fine-tune and optimize the decision-making process of the algorithms.

3 Related Work

There have been many attempts in this area. Kaggle [5] hosts this dataset [1] and some of the leading solutions use Finetuning on top of Inception V3, Resnet-50 and VGG and the use of attention in the final layers to focus on the right part of the image.

4 Dataset and Features

This dataset [1] contains 82GB of high-resolution retina images taken using fundus photography. For each patient, photos of both the left and right eyes are included. The images in this dataset come from a wide variety of clinics over an extended period of time, and therefore were taken using different types of cameras, greatly influencing the visual appearance of the photos. While some photos display the eye as one may see it anatomically, others are inverted due to the use of a microscope condensing lens.

The dataset includes 35k labelled and 42.5k unlabelled images. For the labeled training data, a clinician has rated the presence and severity of diabetic retinopathy (DR) in each eye using the following scoring system: 0 for no DR, 1 for mild DR, 2 for moderate DR, 3 for severe DR, and 4

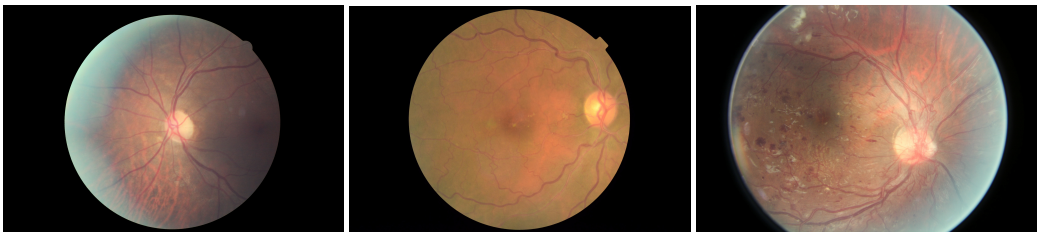

Figure 4.1: Examples of retina images from the dataset. Left: healthy left eye with no DR (0), Middle: right eye with moderate DR (2), Right: left eye with proliferative DR (4).

for proliferative DR. Examples of images and their labels can be seen in Figure 4.1

All of our models take raw images as the input. The images are then preprocessed to reduce them to a resolution of 224x224x3. There are no additional features.

5 Models

To start with, we wanted to setup the dataset and run some small models. With this goal, we tested the following models:

5.1 Linear classifier

We flatten the complete image (224x224x3) into a single dimension and pass it through a linear layer with 5 outputs (since we have 5 output classes).

5.2 Deep CNN classifier

This model uses a simple CNN based architecture with 6 layers of 2D Convolutional neural network with 128 kernels each, 2x2 strides to half the output size at each layer and relu activation. At the end we flatten the output and pass it through a linear layer with 5 outputs.

5.3 Resnet50

This model uses Resnet50 architecture but does not use any pretrained checkpoint. A linear layer is added on top of Resnet50 with 5 outputs.

5.4 Frozen Resnet50 + Classification layer

This model uses a classification layer on top of the output from a Resnet50 backbone used for feature extraction. Resnet50 is frozen during the training process and the weights are loaded from a pretrained model on imagenet dataset.

5.5 Finetuned Resnet50 + Classification layer

This model is same as above except that the layers of Resnet50 model are not frozen.

5.6 Frozen Resnet50 + deep classifier

This model too uses Resnet50 backbone. The layers of Resnet50 are frozen, but we add two linear layers after it (with a ReLU non-linearity in between). The first layer has a output dimension of 1024, while the last layer has 5 outputs.

Common setup
For all the above models, We used categorical cross entropy as our loss function. Generally the output of the above models should be passed through a softmax layer, but we used Tensorflow API which allows us to pass the logits directly to the loss function, removing the need for the softmax layer. We used Adam optimizer in all cases with a learning rate of 0.001. We used an accuracy metric to evaluate the performance on the eval dataset. For Resnet50 [7], we used an existing checkpoint from TF Hub [6].

5.7 Generative approach for modelling from unlabeled dataset

One thing to note about this dataset is that we have a large amount of unlabelled data as well. In our future experiments, we want to ask here is whether we can make use of this additional unlabelled dataset to improve our prediction.

GANS provides a powerful technique to model the distribution of a dataset to differentiate between fake (generated) images and real images (from dataset). While this does not directly help us with our problem, we hypothesize that this can help provide an inductive bias in the model that can help in the real problem.

Essentially, we will have a generator, and a discriminator. The generator will produce fake images, and the discriminator has to produce a probability distribution over the k classes along with predicting whether the input is real (from dataset) or fake (created by generator). We base our work on the Semi-Supervised learning approach from [2]. [3] achieved at the time state-of-the-art result using this method, including MNIST. Hence, we think this might be a good direction to explore.

We evaluate two Generator and two discriminator models for this setup:

5.7.1 Basic Generator

This model takes a 128 dimensional gaussian noise as input, and first maps it to a 7*7*64 image using a fully connected layer followed by reshape. This is followed by 5 layers of 2D transpose convolutional layers with kernel size of 3x3 and strides of 2x2. Hence each of these layers double the height and width finally reaching at 224*224. All the layers use 64 filters except for the last convolutional layer which uses 3 filters. We use relu activation everywhere except for the last layer. Each of the convolutional layers are preceded by batch normalization and relu layers which draw inspiration from similar architecture in ResNet.

5.7.2 Big Gans Generator

This model is based on the BigGans architecture [8]. It consists of small blocks with few 2d transpose convolutional layers along with a residual like connection. Each of these blocks double the size of the output image. We start with 4*4 image and after 6 such layers, we get output of size 256*256. We discard the 16 pixels on each side to get generated images of size 224*224. The batch normalization as mentioned in the paper is conditioned on random noise. We use it along with relu prior to the transpose convolution layers. Additionally, we use a self attention layer prior to the last block.

5.7.3 Basic Discriminator

This model uses 5 layers of convolutional neural network where each layer halves the dimension of the image at each layer. Each of these layers are preceded by a batch normalization and a relu layer. After this, the output is flattened and fed to two fully connected networks. The first network has one output denoting the predicted logit for whether the image is real or fake. The second network has 5 output denoting the predicted logit for each class.

5.7.4 Resnet50 Discriminator

This model uses ResNet50 architecture as the backbone architecture for feature generation. The output is then fed to two fullyconnected networks with 1 and 5 outputs respectively similar to the basic discriminator.

6 EXPERIMENTS

We split the dataset into three sets, train, evaluation and test, in a ratio of 80:10:10. In table 6.2, we present the accuracy on the train and test datasets. For the test dataset, we chose the best checkpoint from the evaluation dataset.

Model	**Train Accuracy**	**Test Accuracy**
Linear Classifier	0.581	0.5761
Deep CNN classifier	0.7396	0.7292
Resnet50	0.7339	0.7288
Frozen Resnet50 + Classification layer	0.5897	0.5653
Finetuned Resnet50 + Classification layer	0.7358	0.73
Frozen Resnet + Deep Classifier	0.5699	0.5307
Basic Generator + Basic Discriminator	-	0.7331
BigGans Generator + Resnet50 Discriminator	-	0.7782

Table 6.1: Performance of different models on train and test dataset

As expected, the linear classifier does not perform very well, but gives us a reasonable baseline. Surprisingly, the next two models in order of accuracy, "Frozen Resnet50 + Classification layer" and "Frozen Resnet50 + Classification layer", do not perform very well either. We hypothesize that this is because of the fact that it was pretrained on a very different dataset (ImageNet) and the extracted features from that dataset are not very useful in our experiment. This is further corroborated by the fact that the other two Resnet models, "Resnet50" and "Finetuned Resnet50" perform equally well. It is worth noticing that the Deep CNN classifier performs equally well, which suggests that increasing the complexity of the model is not helping much at this point.

We do not have train accuracy for the GANS based setup since we didn't log it while training but later calculated it on the eval dataset. The training just worked on reducing the loss functions for generator and discriminator and we logged that, but the accuracy was not logged. The GANS based setups, as expected perform better than a classifier only setup. The BigGans and Resnet50 based setup perform exceptionally well and we think that with the right set of hyperparameter tuning (which we didn't have a time to do), can perform even better. Few samples of the fake images from the generator are in Figure 6.1.

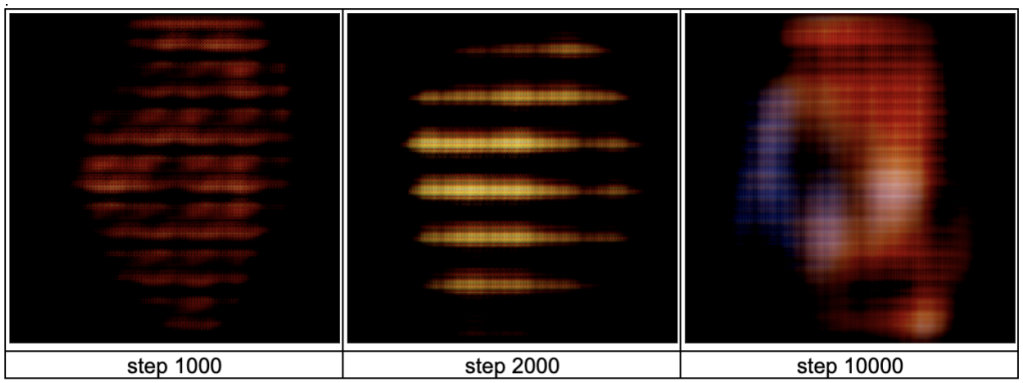

Figure 6.1: Sample of fake images from the generator at various steps.

Error Analysis: The dataset contains images where labels can only be analyzed by doctors who are experienced in the field. Hence, we were not able to perform much error analysis by looking at the failing examples and trying to find pattern. But we did create the confusion matrix evaluated on 10% of the eval dataset as present in Table 6.2. The labels and prepdictions are strongly correlated along the diagonal. The off diagonal elements decrease evenly as we move away from the diagonal. This is intuitive since the labels denote how mild/severe the retina damage is, and errors are mostly between nearby labels.

-	**pred=0**	**pred=1**	**pred=2**	**pred=3**	**pred=4**
label=0	156	20	10	6	1
label=1	17	190	20	12	5
label=2	31	55	385	39	19
label=3	5	7	8	54	13
label=4	2	1	5	8	55

Table 6.2: Confusion matrix computed on 10% of the eval dataset for GANS model

7 Conclusion and Future Directions

The experiments showed that the most accurate model tested was BigGans Generator + Resnet50 Discriminator. We believe that this conclusion is important, because it does not only show the value of using a generative model to improve our accuracy, but it also forges the first steps for future work. The next steps for this project involve two main pillars. First, we would try and analyze how our model will respond to Resnet101 and then even Resnet101+BigGans Generator. Due to the complexity and the abundance of the data, a more intensive model could increase the accuracy of the data, despite the longer time that it would take. Secondly, the information displayed from BigGans Generator + Resnet50 Discriminator also leads to believe that if we look for more specific hyperparameter tuning and let the model run for more number of iterations, we can further improve the accuracy.

8 Contributions

Jai Gupta: Worked on final report, wrote all the code for the preprocessing and models, wrote the sections on modeling, next steps and experiments, tested the models to determine accu-

racy, worked on references

Ignacio Blanco: Worked on the poster, wrote the introduction and error analysis, went to office hours to talk about project with TAs, emailed TAs and professors about project

Amrita Kaur: Worked on the poster, wrote the section on dataset, worked on references, created the data table, went to office hours to talk about project with TAs

REFERENCES

[1] EyePACs (2015, Feb). Diabetic Retinopathy Detection. Version 1. Retrieved October, 2020 from https://www.kaggle.com/c/diabetic-retinopathy-detection/data.

[2] Augustus Odena *Semi-Supervised Learning with Generative Adversarial Networks*, ArXiv preprints, 1606.01583, 2016

[3] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen *Improved Techniques for Training GANs*, ArXiv preprint, 1606.03498, 2016

[4] Mayo Clinic (2018,May) Diabetic Retinopathy – Symptoms and Causes. Retrieved October, 2020 from https://www.mayoclinic.org/diseases-conditions/diabetic-retinopathy/symptoms-causes.

[5] Kaggle *https://www.kaggle.com*

[6] TensorFlow *https://www.tensorflow.org/api_docs/python/tf*

[7] TensorFlow Hub, Resnet-50 Feature Vector *https://tfhub.dev/tensorflow/resnet_50/feature_vector/1*

[8] Andrew Brock, Jeff Donahue, Karen Simonyan *Large Scale GAN Training for High Fidelity Natural Image Synthesis*, ArXiv preprints, 2019