Deep Learning for Diabetic Eye Disease: A Comparative Analysis of Detection and Grading Models

Hyeong Cho **Erin Rafferty**

hjc423 eor2014

Abstract

This project explores VGG16, DenseNet, ResNet50, and AlexNet in classifying retina images for diabetic retinopathy classifications test. We use a dataset from Asia Pacific Tele-Ophthalmology Society (APTOS) that has labels made by clinicians on diabetic retinopathy status. We train and compare the performance of these models and found that VGG16 performed best out of the four models.

The Python notebook containing our code can be found on https://github.com/hyeong-joon-cho/dl-final

Introduction

Diabetic retinopathy stands as a significant microvascular complication of diabetes mellitus, impacting millions with health on a global scale (Stitt et al. 2016). Diabetes mellitus harms multiple organ systems, but the eyes are particularly vulnerable to its effects. Diabetic retinopathy emerges as a degenerative affliction of the retina, marked by the rapid erosion of visual acuity. Its onset often precedes other complications, meaning that early detection in retina images can prevent further health complications.

Because of its importance and relevance, we chose to study detecting diabetic retinopathy in retina images. New advanced deep learning models are known to be useful in image classification tasks, so we decided to focus our project at this intersection.

Our project compares AlexNet, VGG16, DenseNet, and ResNet50 using dataset by Asia Pacific Tele-Ophthalmology Society (APTOS). We describe the differences in the models' architectures and analyze their performance. We also share limitations in our research and suggest directions for improvements.

Literature Survey

There is an abundance of research on diabetic retinopathy in both the medical and machine learning communities. We limit our literature survey to the latter, focusing on detection of retina images using different models.

Kazakh-British et al. proposed an autonomous algorithm and preprocessing of images using anistotropic filters for detection (Kazakh-British, Pak, and Abdullina 2018). Khalifa et al. studied various deep transfer learning models that consisted of relatively small number of layers and achieved high accuracy results (Khalifa et al. 2019). Wu et al. proposed a vision transformer based model to classify diabetic retinopathy in images (Wu et al. 2021). Chetoui et al. proposed an interesting approach using Federated Learning, a type of distributed machine learning, with vision transformer architecture in detection (Chetoui and Akhloufi 2023).

We highlighted a few research papers that uses different deep learning models to classify retinal images for diabetic retinopathy. But these are not comprehensive of all the research and progress that have been made. Due to its importance, there continues to more research and breakthroughs on this topic using different datasets and architectures. In our project, we try to present results in a more consistent way, despite using a relatively small dataset.

Data and Model Overview

The dataset that was used in our project comes from APTOS (APTOS). It is collection of retina images using a fundus photography under different imaging conditions. Labels were added by clinicians for each image on a scale of 0 to 4, according to the following scale: 0 - No DR, 1 - Mild, 2 - Moderate, 3 - Severe, 4 - Proliferative DR. The dataset is about 10 gb in size.

Models we looked at are VGG16, DenseNet, ResNet50, and AlexNet.

VGG16 (Visual Geometry Group 16) is a CNN architecture, consisting of 16 layers, including 13 convolutional layers and 3 fully connected layers. VGG16 has a simple and uniform architecture with small convolutional filters (3x3) stacked on top of each other.

DenseNet is short for Densely Connected Convolutional Networks. It is a CNN architecture introduced to address the vanishing gradient problem. DenseNet introduces dense connections between layers, where each layer receives feature maps from all preceding layers as input.

ResNet50 is a type of ResNet architecture that allows training of very deep neural networks. It uses residual connections, where the input from one layer is added to the output of another layer further down the network. It contains 50 layers of residual blocks.

AlexNet is also a CNN architecture that gained fame after winning ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012. It consists of 5 convolutional layers followed by 3 fully connected layers and incorporates techniques such as dropout and ReLU activation functions to prevent overfitting.

Image Preprocessing

In order to enhance the quality of input images and improve model performance, we applied various preprocessing techniques to the APTOS dataset. We adopted the techniques outlined by Nakhon Ratchasima, a participant in the APTOS 2019 Blindness Detection competition hosted by Kaggle.

To focus on relevant areas of the images, dark regions were identified and removed based on a set pixel intensity threshold. This step eliminates distractions and enhances the visibility of important features within the images.

A circular mask was applied to the center of the images to emphasize the area around the retina. This technique helps isolate key details essential for diagnosing diabetic retinopathy and reduces the influence of less relevant peripheral image content.

Gaussian blurring was applied to the image to reduce noise and minimize high-frequency details that could interfere with image analysis. This clarifies the images by making subtle features more noticeable and less likely to be obscured by random noise.

Model and Training Details

Training for all models was conducted on Google Colab utilizing an NVIDIA A100 GPU. This powerful GPU significantly accelerated the training process, efficiently handling the extensive computational demands of our deep learning models.

The ImageDataGenerator in Keras is used to augment image data through real-time transformations, enhancing the diversity and size of the training dataset. This generator is configured to apply horizontal and vertical flips, perform rotations of up to 180 degrees, and support random zoom adjustments up to 20%. These augmentations help avoid overfitting by preventing the model from learning to recognize only the specific details present in the training images, thereby improving its ability to generalize to new unseen images. This approach is particularly valuable given the small size of our dataset.

All models were trained for 10 epochs, as our aim was comparison rather than optimization. Binary cross-entropy was used as the loss function for all models.

AlexNet

- **Architecture**: Modified AlexNet architecture suitable for 512x512 image inputs, with several layers of convolution followed by max-pooling and dense layers. Uses ReLU activation, Local Response Normalization, and dropout layers.
- **Parameters**: 512,332,165 trainable parameters
- **Training Strategy**: Employs the Adam optimizer with a learning rate of 0.00001.

In our implementations of VGG16, DenseNet, and ResNet50, we incorporate a set of custom layers designed to fine-tune the networks for our classification task.

The first of these layers is a global average pooling layer, which replaces the conventional flattening step used in CNNs. This layer reduces the dimensionality of the feature maps while preserving spatial hierarchies. This is done to maintain the integrity of the image features.

Following the global average pooling layer, a dropout layer with a dropout rate of 0.5 is employed. This layer helps prevent overfitting by randomly zeroing out a portion of the input units during training. This forces the network to learn more robust features.

The architecture concludes with an output layer comprising a dense layer with five units and a sigmoid activation function. This setup is designed to output the probability for each of the five classes.

This combination of custom layers is used across the three models to standardize the output structure.

VGG16

- **Architecture**: Utilized the pre-trained VGG16 architecture provided by Keras, which includes weights trained on the ImageNet dataset, but with the fully connected layers removed to accommodate the custom layers outlined above.
- **Parameters**: 9,441,797 trainable parameters
- **Training Strategy**: Employs the Adam optimizer with a learning rate of 0.00005. The top 13 layers are frozen to retain learned features from ImageNet, and the layers beyond these are made trainable. This allows the latter layers of the model to be fine-tuned on our dataset, enhancing task specific performance.

DenseNet

- **Architecture**: Utillizes the pre-trained DenseNet121 architecture provided by Keras, which includes weights trained on the ImageNet dataset, but with fully connected layers removed to accommodate our custom layers.
- **Parameters**: 6,958,981 trainable parameters
- **Training Strategy**: Employs the Adam optimizer with a learning rate of 0.00005. Given that the initial DenseNet121 layers are frozen to retain the pre-trained ImageNet features, only the custom layers added at the end are trainable. This strategy focuses on fine-tuning the model to our specific dataset.

ResNet50

- **Architecture**: Utillizes the pre-trained ResNet50 architecture provided by Keras, which includes weights trained on the ImageNet dataset, but with fully connected layers removed to accommodate our custom layers.
- **Parameters**: 23,544,837 trainable parameters

Model	Accuracy	Loss	Runtime
AlexNet	0.8218	0.2546	123s
VGG16	0.9182	0.1894	1782s
DenseNet	0.6691	0.1861	1792s
ResNet50	0.7964	0.1326	1798s

Fig. 1: Model Performance Metrics

- **Training Strategy**: The Adam optimizer is employed with a learning rate of 0.00005. The training is conducted with the initial layers of ResNet50 set as non-trainable to maintain the robust features learned from ImageNet, while newly added custom layers are trainable, as in the models described above.

Results and Discussion

Model Performance Overview

AlexNet showed moderate accuracy at 82.18% with a loss of 0.2546, achieving these results in the shortest runtime of 123 seconds. This suggests AlexNet is relatively efficient, making it suitable for environments where computational resources or time are limited

VGG16 achieved the highest accuracy among the tested models at 91.82%, at the cost of a significantly longer training time (1782 seconds) and a loss of 0.1894. This performance indicates a strong model capability but raises concerns about efficiency and practicality due to the long training duration.

DenseNet had the lowest accuracy at 66.91% with a loss similar to that of VGG16 at 0.1861. The runtime for DenseNet was slightly longer than for VGG16 at 1792 seconds, which, considering its lower accuracy, suggests issues with model tuning or overfitting.

ResNet50 provided a balance between accuracy and loss (79.64% and 0.1326 respectively) but had a training runtime comparable to DenseNet and VGG16 at 1798 seconds. The lower loss indicates effective learning but the time cost and moderate accuracy present challenges for real-world applications.

Results are summarized in Fig. 1.

Comparative Analysis

Our results indicate significant variability in performance across the models. VGG16, while the most accurate, requires extensive computational resources, which might pose a problem in tiem-sensitive or resource-constrained environments. In contrast, AlexNet, despite its lower accuracy, offers a trade-off between speed and performance.

DenseNet's performance was less efficient in terms of accuracy and time, suggesting that our configuration may not be optimal for the dataset used. ResNet50, with the lowest loss, shows promising efficiency but with high computational demands as reflected in its runtime.

The differing performances of these models underline the importance of model selection depending upon specific use cases. For example, VGG16 may be preferred in scenarios where accuracy is of great importance and computational resources are plentiful. Where time constraints and resource limitations are present, AlexNet might be a better choice.

It is worth noting that the performance metrics presented could potentially be improved as our implementations of each model may not have been fully optimized for maximum efficiency or accuracy.

Our results highlight the trade-offs between accuracy, loss, and training runtime in different convolutional neural network architectures. Each model presents unique advantages and limitations, and choice of model should be tailored to specific operational constraints and performance goals. These findings provide valuable insight into the application of deep learning models in image classification tasks.

Limitations and Future Work

An obvious extension of this project is to include more models and parameters to test, such as ImageNet, GoogleNet, and other variations of ResNet. Different datasets could have been analyzed to study how or if the models perform differently. A more detailed audit of the datasets with respect to demographic information, for example, is also a good direction of research to ensure fairness across different subpopulations. An analysis of the results, particularly the incorrect predictions is something to look at as well. For example analyzing false negatives, incorrectly predicting an image as "No" or "Mild" when an image actually has "Severe" or "Proliferative" can lead to dire consequences. However, with limited resources, we fell short of the more comprehensive analysis we initially aimed to do.

Conclusion

In our project we aimed to train and assess several deep learning models using the diabetic retinopathy dataset in order to determine which model can be most effectively utilized. We assessed performance beyond accuracy, but not to the level of exhaustiveness we initially sought out to do, as explained in the **Limitations and Future Work** section. Despite some setbacks, however, we are optimistic that our project, is fruitful and useful as it compared the different models for the same datasets and other controllable constants.

In conclusion we evaluated four popular deep learning architectures, VGG16, DenseNet, ResNet50, and AlexNet, for diabetic retinopathy detection in retinal images. We looked at the accuracy values, losses, and runtime, concluding that VGG16 performed the best overall.

a conclusion section explaining what you originally intended to do, and what you have accomplished.

References

APTOS. ???? APTOS Dataset - Kaggle. https://www.kaggle.com/competitions/aptos2019-blindness-detection/data.

Chetoui, M.; and Akhloufi, M. A. 2023. Federated Learning for Diabetic Retinopathy Detection Using Vision Transformers. *BioMedInformatics*.

Kazakh-British, N. S. P.; Pak, A. A.; and Abdullina, D. 2018. Automatic Detection of Blood Vessels and Classification in Retinal Images for Diabetic Retinopathy Diagnosis with Application of Convolution Neural Network. In *Proceedings of the 2018 International Conference on Sensors, Signal and Image Processing*, SSIP '18, 60–63. New York, NY, USA: Association for Computing Machinery. ISBN 9781450366205.

Khalifa, N. E. D. M.; Loey, M.; Taha, M. H.; and Mohamed, H. N. E. T. 2019. Deep Transfer Learning Models for Medical Diabetic Retinopathy Detection. *Acta Informatica Medica*, 27: 327 – 332.

Ratchasima, N. ???? https://www.kaggle.com/code/ratthachat/aptos-eye-preprocessing-in-diabetic-retinopathy.

Stitt, A. W.; Curtis, T. M.; Chen, M.; Medina, R. J.; McKay, G. J.; Jenkins, A.; Gardiner, T. A.; Lyons, T. J.; Hammes, H.-P.; Simo, R.; et al. 2016. The progress in understanding and treatment of diabetic retinopathy. *Progress in retinal and eye research*, 51: 156–186.

Wu, J.; Hu, R.; Xiao, Z.; Chen, J.; and Liu, J. 2021. Vision Transformer-based recognition of diabetic retinopathy grade. *Medical Physics*, 48(12): 7850–7863.
(Stitt et al. 2016) (Ratchasima)