Diabetic retinopathy - Diagnosis with Convolutional Neural Networks (CNNs)

André Oliveira, Diogo Ribeiro Santos, Gonçalo Eloy,
Gonçalo Matos, Rafael Chamusca
Nova IMS
Campus de Campolide, 1070-312 Lisboa
Email: {20222156, 20222152, 20222174, 20221194, 20222162}@novaims.unl.pt

***Abstract*—Purpose. This study aims to evaluate the efficacy of Convolutional Neural Networks (CNNs) in diagnosing diabetic retinopathy using Kaggle's dataset. Methods. Different CNN architectures were tested, including dropout, local histogram equalization, and data augmentation strategies. The VGG-16 model was also employed for comparison. Results. All models demonstrated high accuracy in identifying diabetic retinopathy, with the VGG-16 model showing exceptional performance (AUC score: 0.97). Conclusions. The study concludes that CNNs are highly effective in diagnosing diabetic retinopathy, highlighting their potential utility in medical image analysis and diagnostics.**

***Index Terms*—Diabetic Retinopathy, CNNs, Retinal Image Analysis, Machine Learning in Ophthalmology, Transfer Learning, Vision Loss Prevention, Medical Imaging Automation**

I. Introduction

A. Objective

The purpose of this study is to evaluate the performance of Convolutional Neural Networks (CNNs) for the automatic detection of diabetic retinopathy (DR).

B. Real-world significance

The increasing prevalence of Diabetic Retinopathy among diabetes patients highlights the critical need for early detection and effective treatment to prevent vision loss. Current manual screening methods are slow and error-prone, creating a demand for an automated, precise diagnostic tool. This need is amplified by the rising diabetes rates and the scarcity of ophthalmologists, underscoring the importance of developing a reliable automated system for the accurate detection and grading of Diabetic Retinopathy, allowing for timely interventions and personalized treatments.

C. Related work

There is literature applying CNNs to the detection of DR, dealing with binary and multiclass classification tasks (stage of DR) and feature detection (laterality of the eye, i.e., left vs. right) [1] [2] [3]. The main results are presented in Table 1, except for [3], that does not report statistics. There are also Kaggle projects using our dataset, whose results are pictured in Table 2.

TABLE I
Main results in the literature

Study	Dataset size	Specificity	Sensitivity	AUC
[1]	763,848			0.989
[2]	88,252	0.92	0.80	

TABLE II
Main results in Kaggle

Project	Precision	Recall	F1-score	Accuracy
1	0.94	0.94	0.94	0.94
2				0.97
3	0.95	0.94	0.94	0.94

D. Dataset description

The dataset was retrieved from Kaggle via an API. The preprocessed dataset consists of 2,838 retina scan images, taken under a variety of imaging conditions. The original dataset is available here. These images are 224x224, 3, meaning that the image is a 224 by 224 pixel square, with three color channels (RGB - Red, Gree, Blue). An ophtalmologist has rated the presence of diabetic retinopathy in each image, according to the following scale (class size in parenthesis): 0 - No DR (1,430) and 1 - DR (1,408). The dataset was also divided into train, validation and test sets, as per Table 3:

TABLE III
Train, validation and test set characteristics

Class	Train	Validation	Test	Total
No DR	1,026	286	118	1,430
DR	1,050	245	113	1,408
Total	2,076	531	231	2,838

II. Methods

A. Preprocessing

Scaling data before applying it to a Convolutional Neural Network (CNN) is a good practice. It speeds up training, avoids numerical instability, prevents bias, and ensures better weight initialization and consistency across images. So, pixel values were scaled to a range of 0 to 1 by dividing by 255 since RGB values range from 0 to 255. Min-max scaling is preferable to standardization, which entails a nonlinear transformation of the data. Scaling was applied to the training, validation and test sets.

B. Models

Our approach combines Convolutional Neural Networks (CNNs) and transfer learning. We use a single CNN in different versions with increasing complexity. CNNs offer several significant advantages for classification tasks in computer vision:

- Feature Learning: Unlike traditional image processing methods, CNNs have the ability to automatically and adaptively learn spatial hierarchies of features from input images. They can capture important features without the need for manual feature extraction.
- Local Connectivity: In CNNs, each neuron is connected only to a small region of the input image (receptive field), reducing the number of parameters and computations compared to fully connected networks. This locality and translation invariance make them highly efficient for image data.
- Shared Weights Architecture: CNNs use shared weights in convolutional layers, meaning the same filter (weights) is used across the entire input image. This not only reduces the memory footprint but also allows the network to learn features that are useful across the whole image.
- Robustness to Image Variations: CNNs are less sensitive to the location and orientation of features in the image due to their pooling layers and translational invariance, enhancing their ability to recognize objects regardless of their position in the field of view.
- Depth and Hierarchical Feature Extraction: CNNs can have deep architectures with multiple layers, enabling the extraction of complex, high-level features from the raw input image. Lower layers typically identify simple features like edges or textures, while deeper layers can identify more complex features.

C. Transfer learning

Transfer learning is a machine learning technique where a model developed for one task is reused as the starting point for a model on a second task. It's particularly useful in deep learning where large datasets and computational resources are required to train models from scratch. The VGG16 model, developed by the Visual Graphics Group at Oxford, is a popular example of a pre-trained model used in transfer learning. Originally trained on a large dataset (ImageNet) for image classification tasks, its architecture, consisting of 16 convolutional layers, is adept at capturing complex features from images, making it highly effective for various image recognition tasks. By fine-tuning VGG16 on specific datasets, it can be adapted for different image classification problems with reduced training time and resources.

D. Topology

The key aspects of the models' topologies are as follows:

1) Base models (v1, v2, v3, v4, v5):

- Common structure: All five versions of the model have a common structural foundation:
  - Architecture: Each model is a Sequential CNN, which is typical for image classification tasks. The structure has convolutional layers followed by max-pooling layers, a flattening step, and fully connected (dense) layers.
  - Input Shape: Holds the raw pixel values of the image. The input shape for all models indicates that color images (3 channels) of size 224x224 pixels are expected.
  - Activation Functions: ReLU (Rectified Linear Unit) is used for intermediate layers, providing non-linearity without significant computational cost. The sigmoid activation function in the final layer is suited to binary classification (Diabetic Retinopathy vs. No Diabetic Retinopathy).
  - Compilation: All models are compiled with the 'adam' optimizer and '`binary_crossentropy`' loss function, which are standard choices for binary classification tasks.
- Layer Configuration:
  - Input: Here, an image having the width of 224, height of 224, and three color channels R, G, B (depth = 3).
  - Convolutional Layers: Each model begins with two convolutional layers (with 32 and 64 filters respectively) with a kernel size of 3x3 and ReLU activation. They compute a dot product between the filter weights and a small region they are connected to in the input volume. These layers are designed to extract features from the images.
  - Pooling Layers: Following each convolutional layer, there's a max-pooling layer with a pool size of 2x2 to reduce the spatial dimensions of the feature maps.
  - Flattening: The output from the convolutional and pooling layers is flattened into a one-dimensional vector before being passed to the dense layers.
  - Dense Layers: After flattening, there are dense layers for classification. They are fully connected, in the sense that each neuron in these layers will be connected to all the inputs from the previous layer. The final output layer uses a sigmoid activation function, suitable for binary classification.
- Version Differences
  - Base Model v1: The initial model, serving as the foundation for subsequent versions.
  - Base Model v2: Introduces data augmentation during training, including rotation, width shift, height shift, shear, and zoom. Uses RMSprop optimizer with a learning rate of 1e-4 for compilation.
  - Base Model v3: Introduces a dropout layer for regularization after the flattening step. Dropout is used

to prevent overfitting. Nodes were randomly dropped with a probability p=0.5.

  - Base Model v4: Applies normalization (samplewise) and local histogram equalization to the training data. The data augmentation settings are tuned further.
  - Base Model v5: Similar to v4 but removes normalization from the data generator and fine-tunes data augmentation parameters.

2) VGG-16 Model:

- This model leverages the VGG-16 architecture, a pre-trained model known for its performance in image classification. Leveraging a model refers to the process of using a pre-existing model or framework to enhance or simplify the development of a new model. This is where we take a model trained on one task and repurpose it for a different but related task.
- Training is conducted with specified callbacks like ModelCheckpoint and EarlyStopping to enhance performance and prevent overfitting.
- It includes custom top layers (two dense layers with 4096 units each and a dropout layer) while the VGG-16 base is used for feature extraction.
- The model is initially trained without fine-tuning and later with fine-tuning of the last two layers.

3) Data augmentation:

- Purpose of Data Augmentation:
  - Combat Overfitting: By introducing variability in the training data, the model is less likely to overfit to the noise and specific patterns in the training set.
  - Model Generalization: It helps in creating a more robust model that can generalize better to new, unseen data by simulating different conditions.
- Implementation via TensorFlow's ImageDataGenerator: This tool allows for the on-the-fly transformation of images during training, which helps to artificially expand the dataset.
- On-the-Fly Augmentation: Instead of augmenting the images and then adding them to the dataset, the augmentation happens in real-time as the images are fed into the model during training. Each image is randomly transformed every time it is passed through the network during training. Since the transformations are applied randomly and on-the-fly, the model rarely, if ever, sees the exact same image twice. This introduces a wide variety of variations from a single image, enhancing the model's ability to generalize without the need for physically storing additional augmented images.
- Types of Augmentation Used:
  - Rotation: Images are rotated by up to 40 degrees (`rotation_range=40`), which helps the model learn from various rotated perspectives of the retina.
  - Width and Height Shifts: The images are randomly shifted in width and height within a specified range (`width_shift_range=0.2` and `height_shift_range=0.2`), simulating variations in image alignment.
  - Shear Transformation: The images undergo shear transformations (`shear_range=0.2`), which can simulate changes in angle or perspective.
  - Zoom: There is a zoom augmentation (`zoom_range=0.2`), which helps the model learn from images that are variously zoomed in or out.
  - Horizontal Flip: Images are flipped horizontally (`horizontal_flip=True`), which is logical for retinal images as flipping does not change the diagnosis.
  - Fill Mode: The `fill_mode='nearest'` parameter is used to fill in new pixels that might be created due to rotations or shifts.

E. Evaluation and Performance Metrics

Models are evaluated based on training and validation accuracy, loss, and ROC AUC (Receiver Operating Characteristic - Area Under Curve) score. Accuracy is a primary metric indicating the proportion of correctly classified images (both DR and No DR) out of the total number of images in the test set. It gives a general idea of how well the model performs across both classes. Recall (or Sensitivity) shows the proportion of actual positives (DR) that were identified correctly. It's crucial for medical diagnoses where missing a positive case can be critical. F1-Score, the harmonic mean of precision and recall, provides a balance between these two metrics. ROC AUC Score is a performance measurement for the classification at various thresholds settings. The AUC represents a degree or measure of separability - how well the model is able to distinguish between classes. In medical image classification, a high AUC is desirable as it indicates the model's capability to differentiate between healthy and diseased cases with high confidence. The combination of these metrics provides a well-rounded picture of the model's performance, considering not only its accuracy but also its precision, recall, and ability to distinguish between classes. This is particularly important in medical applications like DR diagnosis, where the cost of false negatives (missed diagnoses) or false positives (unnecessary treatments) can be significant.

III. Results and Discussion

A. Results

The results are displayed in Table 4.

TABLE IV
METRICS OF EACH MODEL

Value 1	Value 2	Value 3	F1-score	Accuracy	AUC
v1	0.92	0.92	0.92	0.92	0.92
v2	0.93	0.93	0.93	0.93	0.93
v3	0.91	0.91	0.91	0.91	0.91
v4	0.94	0.94	0.94	0.94	0.94
v5	0.93	0.93	0.93	0.93	0.93
VGG16	0.97	0.97	0.97	0.97	0.97

B. Error analysis

The VGG16 model's generalization is evaluated by comparing training and validation performance. The accuracy per epoch graph shows a convergence of training and validation lines, suggesting the model is learning patterns applicable to unseen data. Similarly, the loss graph indicates that the validation loss decreases and stabilizes in line with the training loss, a behavior expected from a well-generalizing model. In terms of classification errors, the confusion matrix reveals an equitable distribution of false positives and negatives, with a total of 9 errors out of 231 cases. This distribution indicates that the model's ability to generalize does not skew towards one class. The error analysis confirms the VGG16 model's proficiency in generalizing from training to unseen data, evidenced by consistent accuracy and loss results across training, validation, and test sets. The model's balanced classification of errors also suggests a reliable predictive performance, underscoring its potential for application in clinical settings.

C. Discussion

The evolution of our models shows a progression from a basic CNN architecture to more sophisticated versions incorporating advanced techniques like dropout, local histogram equalization, and varied data augmentation strategies. These enhancements are aimed at improving the model's ability to generalize and perform accurately on unseen data, particularly important in medical image analysis. Table shows a consistent performance across versions 1 to 5, with slight improvements in later versions. This suggests incremental enhancements in the models' abilities to classify correctly. The VGG16 model outperforms the other versions in all metrics, indicating its superior capability in this classification task. The 0.97 AUC score compares well with the 0.989 score reported in [1], with a dataset 269 times larger than our own. Its recall is far larger than the sensitivity reported in [2] of 0.80, whose dataset is 31 times larger. This model also performs better than three Kaggle projects using the same dataset. This could be attributed to its deeper and more complex architecture, which allows for better feature extraction. In fact, its results lead us to drop the hyperparameter tuning initially planned. The metrics for all models are relatively high (above 0.90), which suggests that all models perform well for this particular task. This is especially notable in critical fields like medical diagnostics, where high accuracy and recall are crucial. The similar values of precision, recall, and F1-score across the models indicate a balanced performance. There's no significant trade-off between precision and recall, which is often a challenge in classification tasks.

IV. Conclusion

The report explores the effectiveness of CNNs in detecting diabetic retinopathy. It emphasizes the increasing need for automated tools in medical imaging due to the rising prevalence of diabetic retinopathy and the limitations of manual screening. The study utilizes Kaggle's dataset, experimenting with different CNN architectures and techniques like dropout, local histogram equalization, and various data augmentation strategies. It concludes that all the models, especially the VGG-16, show promising results in accurately classifying diabetic retinopathy, underscoring the potential of CNNs in medical diagnostics.

References

[1] G. Quellec, M. Lamard, B. Lay, A. L. Guilcher, A. Erginay, B. Cochener, and P. Massin, "Instant automatic diagnosis of diabetic retinopathy," *arXiv preprint arXiv:1906.11875*, 2019.

[2] J. G. Nam, S. Park, E. J. Hwang, J. H. Lee, K.-N. Jin, K. Y. Lim, T. H. Vu, J. H. Sohn, S. Hwang, J. M. Goo *et al.*, "Development and validation of deep learning–based automatic detection algorithm for malignant pulmonary nodules on chest radiographs," *Radiology*, vol. 290, no. 1, pp. 218–228, 2019.

[3] N. Singh and R. C. Tripathi, "Automated early detection of diabetic retinopathy using image analysis techniques," *International Journal of Computer Applications*, vol. 8, no. 2, pp. 18–23, 2010.