*mathematics*

MDPI

*Article*

# Designing the Architecture of a Convolutional Neural Network Automatically for Diabetic Retinopathy Diagnosis

**Fahman Saeed [1], Muhammad Hussain [2,*], Hatim A. Aboalsamh [2], Fadwa Al Adel [3] and Adi Mohammed Al Owaifeer [4]**

[1] Department of Computer Science, Imam Mohammad Ibn Saud Islamic University (IMSIU), Riyadh 11432, Saudi Arabia
[2] Department of Computer Science, King Saud University, Riyadh 11543, Saudi Arabia
[3] Department of Ophthalmology, College of Medicine, Princess Nourah bint Abdulrahman University, Riyadh 11671, Saudi Arabia
[4] Ophthalmology Unit, Department of Surgery, College of Medicine, King Faisal University, Al-Ahsa 31982, Saudi Arabia
* Correspondence: mhussain@ksu.edu.sa

**Abstract:** Diabetic retinopathy (DR) is a leading cause of blindness in middle-aged diabetic patients. Regular screening for DR using fundus imaging aids in detecting complications and delays the progression of the disease. Because manual screening takes time and is subjective, deep learning has been used to help graders. Pre-trained or brute force CNN models are used in existing DR grading CNN-based approaches that are not suited to fundus image complexity. To solve this problem, we present a method for automatically customizing CNN models based on fundus image lesions. It uses k-medoid clustering, principal component analysis (PCA), and inter-class and intra-class variations to determine the CNN model's depth and width. The designed models are lightweight, adapted to the internal structures of fundus images, and encode the discriminative patterns of DR lesions. The technique is validated on a local dataset from King Saud University Medical City, Saudi Arabia, and two challenging Kaggle datasets: EyePACS and APTOS2019. The auto-designed models outperform well-known pre-trained CNN models such as ResNet152, DenseNet121, and ResNeSt50, as well as Google's AutoML and Auto-Keras models based on neural architecture search (NAS). The proposed method outperforms current CNN-based DR screening methods. The proposed method can be used in various clinical settings to screen for DR and refer patients to ophthalmologists for further evaluation and treatment.

**Keywords:** classification; deep learning; DeepPCANet; diabetic retinopathy; medical imaging; PCA; AutoML; NAS

**MSC:** 68T07



**Citation:** Saeed, F.; Hussain, M.; Aboalsamh, H.A.; Al Adel, F.; Al Owaifeer, A.M. Designing the Architecture of a Convolutional Neural Network Automatically for Diabetic Retinopathy Diagnosis. *Mathematics* **2023**, *11*, 307. https://doi.org/10.3390/math11020307

Academic Editors: Xiang Li, Shuo Zhang and Wei Zhang

Received: 3 November 2022
Revised: 25 December 2022
Accepted: 28 December 2022
Published: 6 January 2023



## 1. Introduction

Diabetes is a leading global health dilemma. One of its serious complications is diabetic retinopathy (DR), which has a prevalence of 34.6% worldwide and is considered a primary cause of blindness among middle-aged diabetic patients [1,2]. A patient has a high DR risk if he or she has had diabetes for a long time or is poorly managed. The DR treatment at its early stage slows down the retinal microvascular degeneration process. Graders manually screen fundus images to detect DR prognosis, which is time-consuming and subjective [3–5]. On the other hand, screening a large number of diabetic patients for the possible prevalence of DR puts a heavy load on graders and reduces their efficiency. It necessitates intelligent systems for DR screening, and many ML-based systems have been proposed that show good results on public data sets. However, their performance is not certain in real DR screening programs, where there are different ethnicities, and the retinal

fundus images are captured using different cameras. These factors affect these systems' performance and remain a challenge in their widespread use [6].

Deep CNN has shown remarkable results in many applications [7–11] and has been employed in DR screening [2,12–15]. A CNN model usually involves a large number of parameters and needs a large amount of data for training. A brute force approach, which has been widely used for DR screening, is to adopt a highly complex CNN model designed for object recognition and pre-trained on the ImageNet dataset and fine-tune them using fundus images [16–19]. As the ImageNet dataset consists of natural images, and the structural patterns of natural images and fundus images are entirely different, the architectures of the fine-tuned models do not adequately encode the fundus images. In addition, the complexity of pre-trained models is very high and not customized to DR screening from fundus images.

Instead, CNN models are manually designed from scratch. The design process starts with a CONV layer of a small width (i.e., the number of filters) and increases the widths of CONV layers by a fixed ratio as the network goes deeper [16–18]. There is no way to know what the depth should be (i.e., the number of layers) of a CNN model; a hit–trial strategy is used to fix the depth. In addition, CNN models are trained using iterative optimization algorithms such as stochastic gradient descent algorithms, and their convergence heavily depends on the initial guess of learnable parameters. Different data-independent [20,21] and data-dependent [22,23] approaches have been proposed to initialize them.

Alternatively, automated machine learning (AutoML) has developed into a significant area of research due to the widespread application of machine learning techniques [24]. AutoML's purpose is to make machine-learning models accessible to those with limited machine-learning prior knowledge. Some of the most commonly used methods for employing machine learning (ML) are easily available and may be used with just one or two lines of code. These systems include Auto-WEKA, Hyperopt-Sklearn, TPOT, Auto-Sklearn, and Auto-Keras [25–32]. Efforts have been made to automate the model selection and tuning hyper-parameters automatically, and so forth. Within the perspective of profound NAS stands for learning, neural architecture search [33], which aims to determine the optimal neural network architecture for a given learning task and dataset, has evolved into a highly effective computational tool for AutoML [34,35]. It achieved competitive performance on the CIFAR-10 and Penn Treebank benchmarks by utilizing a reinforcement learning-based search strategy; consequently, NAS became a mainstream research topic in the machine learning community. NAS is prohibitively expensive and time-consuming in terms of computation [36]. Zoph and Le [33] utilize massive computational resources (800 GPUs for three to four weeks) to achieve their result.

The preceding discussion demonstrates that developing an AutoML-customized lightweight CNN model for DR screening that uses a small subset of the target dataset and consumes fewer resources in a variety of clinical settings is difficult; it entails answering three design questions: (i) what must be the depth of the model, (ii) what must be the width of each of its convolutional (CONV) layer, i.e., the number of its kernels, and (iii) how to initialize the learnable parameters. To address these questions, we propose a constructive data-dependent approach for designing CNN models for DR screening under diverse clinical settings that automatically determine the depth of the model and the width of each CONV layer and initialize the learnable parameters. A custom-designed model takes a fundus image as input and grades it into normal or DR levels. We validated the proposed approach on three datasets: a local DR dataset from King Saud University Medical City, Saudi Arabia, and two benchmark Kaggle datasets: EyePACS [37] and APTOS2019 [38]. Specifically, the main contributions of the paper are as follows:

- We proposed a constructive data-dependent AutoML approach to design lightweight CNN models customized to DR screening under various clinical settings. It automatically determines the depth of the model, and the width of each COVN layer and initializes the learnable parameters using the fundus images dataset.

- To corroborate the usefulness of the proposed approach, we applied it to build an AutoML custom-designed lightweight CNN architecture for three datasets.
- We performed extensive experiments to show that the custom-designed lightweight CNN models compete well with the pre-trained models such as ResNet [17], DenseNet [18], ResNeSt [39], an AutoML NAS method, and other state-of-the-art methods for DR screening.

The layout of the rest of the paper is as follows: the literature view is presented in Section 2, datasets are described in Section 3, the detail of the proposed method is given in Section 4, the detail of experiments and the results are presented in Section 5, and finally, Section 6 concludes the paper.

## 2. Previous Work

Different methods have been introduced for automatic DR screening; an extensive literature review is given in [40–43]. There are some efforts to compress and reduce the complexity of existing pre-trained CNN models by weights pruning [44,45] or filters pruning [46–48]. First, we provide an overview of the previous work on building a deep model and initializing its weights and then give an overview of the state-of-the-art techniques for DR diagnosis.

### *2.1. Data-Dependent and Auto-Deep Models*

Different researchers employed principal component analysis (PCA) in various ways to build deep networks. Chan et al. [49] created an unsupervised two-layer model (PCANet). It is not an end-to-end model and is used only for feature extraction. Philipp et al. [22] used PCA to re-initialize pre-trained CNN models to avoid vanishing or exploding gradient problems. Suau et al. [23] used PCA and correlation to compress the filters of pre-trained CNN models. Seuret et al. [50] employed PCA to initialize the layers of stacked auto-encoders (SAEs). The above PCA-based methods have been employed for designing a CNN-like model for feature extraction, data-dependent re-initialization of the pre-trained models, or compressing their weights to reduce their complexity, but not for the data-dependent design of end-to-end CNN models.

Zhong et al. [51] introduced a method to build a BlockQNN module automatically using the block-wise setup, Q-Learning paradigm, and epsilon-greedy exploration and stacked them to obtain the automatic CNN model. They evaluated their method using CIFAR-10, CIFAR-100, and ImageNet. It needs a lot of computational resources. They used 32 GPUs and got the best CNN model with BlockQNN after three days and Faster BlockQNN after 20 h.

AutoML's initial effort was led by academia and machine learning practitioners, followed by startups and Auto-Weka (2013) [52] from the Universities of British Columbia (UBC). Following that, the University of Freiburg published Auto-Sklearn (2014) [53]. TPOT was created by the University of Pennsylvania [27] in (2015). Following the success of Zoph and Le in [33] in performing comparably to the CIFAR-10 and Penn Treebank benchmarks, other recent efforts to develop NAS have been made [54,55], they incorporate modern design elements previously associated with handcrafted architectures, such as skip connections, which enable the construction of complex, multi-branch networks. To maximize efficiency, state-of-the-art systems employ cell-search spaces [56], which involves configuring only repeated cell architectures rather than the global architecture, and employ gradient-based optimization [57]. Since 2013, Bayesian optimization has achieved several early successes in NAS, resulting in state-of-the-art vision architectures [58]. Google Cloud AutoML based on NAS method is one of the famous auto deep learning models' auto-generation [59]. It utilizes transfer learning and NAS to determine the optimal network architecture and hyper-parameter configuration for that architecture that minimizes the model's loss function [60]. Another method for autoML-based NAS for generating deep learning models is Auto-Keras (2017) [29] from Texas A&M University, which runs on top of Keras, Tensorflow, and Scikit-learn

### *2.2. DR Screening Methods*

Clinical DR screening categorizes a patient based on fundus images into different grades: level 0 (normal), level 1 (mild), level 2 (moderate), level 3 (severe), and level 4 (proliferative). In the state-of-the-art on DR screening, various deep learning-based methods have been proposed, which address mainly three image-level DR grading scenarios: (i) scenario 1 (SC1): normal and different levels of DR severity—a multi-class problem, (ii) scenario 2 (SC2): normal (level 0) vs. DR (levels 1~4)—a two-class problem, (iii) scenario 3 (SC3): non-referral (level 0 and 1) vs. referral (levels 2–4)—a two-class problem. In the following paragraphs, we give an overview of the state-of-the-art best methods. Islam et al. [61] built a hand-designed CNN model consisting of 18 layers and 8.9 million learnable parameters and got on the EyePACS dataset a sensitivity of 94.5%, a specificity of 90.2% for SC2, a sensitivity of (98%) and a specificity of (94%) for SC3. Li et al. [62] introduced two hand-designed CNN models with 11 and 14 layers for feature extraction from the EyePACS dataset. The features from both models are fused and classified using an SVM classifier. They achieved an accuracy of 86.17% for SC1 and an accuracy of 91.05%, a sensitivity of 89.30%, and a specificity of 90.89% for SC2 using five-fold cross-validation. Challa et al. [63] built a CNN model consisting of 10 layers for the EyePACS dataset and obtained an accuracy of 86% for SC1. Tymchenko et al. [64] built an ensemble of 20 CNN models. The ensemble used five versions of each of four pre-trained models: SE-ResNetXt50 with input sizes of 380 × 380 and 512 × 512, EfficientNet-B4, and EfficientNet-B5. It was fine-tuned using the APTOS2019 dataset. They got an accuracy of 91.9%, a sensitivity of 84%, a specificity of 98.1%, and a Kappa of 96.9% for SC1 on the APTOS2019 dataset. Sikder et al. [65] used an ensemble learning algorithm called ET classifier to classify the colored information of the fundus images from the APTOS2019 dataset. They filtered the dataset by removing many noisy samples and achieved an accuracy of 91% and a recall of 89.43% for SC1. DR categorization was performed manually by Sikder et al. (2021) [66]. They conducted significant preprocessing to fundus pictures before extracting the histogram and GLCM features. The APTOS2019 dataset was utilized to validate the procedures, with 75% used for training and 25% for testing. The XGBoost algorithm was used to fine-tune and pick the best features for optimal performance. Classification accuracy for DR (five classes) was 94.20% (95% CI: 93.88–94.51%) for the whole set of features and 93.70% (95% CI: 93.48–93.93%) for the subset.

The above overview of the state-of-the-art methods shows that some used hand-designed CNN models, and others employed pre-trained models and fine-tuning. For creating hand-designed models, the architectures of CNN models were fixed empirically using the hit-and-trial approach. In the case of fine-tuning, the complexity of the pre-trained models is very high and is not customized to the structures of fundus images.

## 3. Materials

We developed and validated custom-designed CNN models using two Kaggle challenge datasets: EyePACS [37] and APTOS2019 [38], and one local dataset collected at King Saud University Medical City (KSU-DR). Each dataset was preprocessed and augmented using the procedure described in Section 4.2.1. KSU-DR and EyePACS were divided into training (80%), validation (10%), and testing (10%). APTOS2019 consists of two sets: public training and public testing; 90% of the public training data was used for training and the remaining 10% for validation and public testing for testing.

### *3.1. KSU-DR*

The data were collected after obtaining approval from King Saud University Medical City's local Institutional Review Board committee. The samples were collected randomly from fundus images of diabetic patients acquired during their routine endocrinologist's appointment at the funduscopic screening clinic. Fundus images were captured with a non-mydriatic fundus camera (3D-OCT-1-Maestro non-mydriasis fundus camera); a 45-degree fundus photo was captured from each eye. All patients were from Saudi Arabia, 44% were

males, and 56% were females. The mean patient age was 53 years; 17% had type 1 diabetes, and 83% had type 2 diabetes. The mean duration of diabetes was 18 years (ranging from 4–42 years). Random samples of 1750 images were selected and graded by two expert ophthalmologists; 1024 were graded as normal, 477 as mild non-proliferative DR, 222 as moderate non-proliferative DR, 20 as severe non-proliferative DR, and 7 as proliferative DR (PDR).

### *3.2. EyePACS*

EyePACS [37] consists of 88,702 color retinal fundus images with varying resolutions up to about 3000 × 2000 pixels [63], collected from 44,351 subjects, but only 35,126 labeled images are available in the public domain; most of the researchers used this set for the proposal of new algorithms [61,67]. We also used 35,126 labeled images to design and evaluate the custom-designed CNN model. The images are graded into normal and 4 DR classes—mild, moderate, severe, and proliferative.

### *3.3. APTOS2019*

APTOS2019 dataset [38] was published by the Asia Pacific Tele-Ophthalmology Society on the Kaggle competition website. Clinical experts graded the images into normal and 4 DR levels (mild, moderate, severe, and proliferative). The public domain version of this database contains 3662 fundus images for training and 1928 fundus images for testing.

## 4. Proposed Method

### *4.1. Problem Formulation*

The problem is to predict whether a subject has normal vision or suffering from DR (with different levels of severity) using his/her retinal fundus images. Formally, let $R^{h\times w\times 3}$ be the space of color retinal fundus images with resolution $h \times w$ and $P= \{1, 2, \ldots, C\}$ be the set of labels where $C$ is the number of classes, which represent different DR grades; in case of two grades (i.e., normal and DR), $C$ = 2, such that $c$ = 1 means normal and $c$ = 2 stands for DR; when there are five grades, $C$ = 5, and $c$ = 1, 2, 3, 4, 5 are the labels for normal, mild, moderate, severe, proliferative DR, respectively. To predict the grade of a patient, we need to define a mapping $\phi : R^{h\times w\times 3} \rightarrow P$ that takes a fundus image $\boldsymbol{x} \in R^{h\times w\times 3}$ and associates it to a label $c \in P$, i.e., $\phi(\boldsymbol{x}) = c$. We model the mapping function $\phi$ using a custom-designed CNN model.

### *4.2. Custom-Designed CNN Model*

The main constituent layer of a CNN model is the CONV layer, and the widely adopted CNN models contain a large number of CONV layers, e.g., VGGNet [68] contains 13 CONV layers. The number of layers (depth) and the number of filters in each layer (width) are fixed manually, keeping in view the ImageNet challenge dataset [69], without following any formal procedure. Retinal fundus images have complex small-scale structures, which form discriminative patterns and are entirely different from those of the natural images in the ImageNet dataset. We design an AutoML CNN model for the DR problem by drawing its architecture from the fundus images; we determine the depth of a model and the widths of its CONV layers in a customized way using the discriminative information in fundus images specific to different DR levels. In this direction, the first design decision is about specifying the search space and extracting discriminatory information. For this purpose, first, we reduce the search space and select the most representative fundus images from the available DR dataset using the K-medoids clustering algorithm [70] and then apply PCA [71] to determine the widths of CONV layers and initialize them. The next design decision is about the depth (i.e., the number of CONV layers). We control the depth using the ratio of the between-class scatter matrix $\boldsymbol{S_b}$ to the within-class scatter matrix $\boldsymbol{S_w}$. Finally, motivated by the design strategy of ResNet [17], we add global pooling layers that follow the last CONV layer, and their outputs are fused and fed directly to a softmax layer. These layers control the drastic increase in the number of learnable parameters (which cause

overfitting). The design process is described in detail below, and its overview is shown in Figure 1.

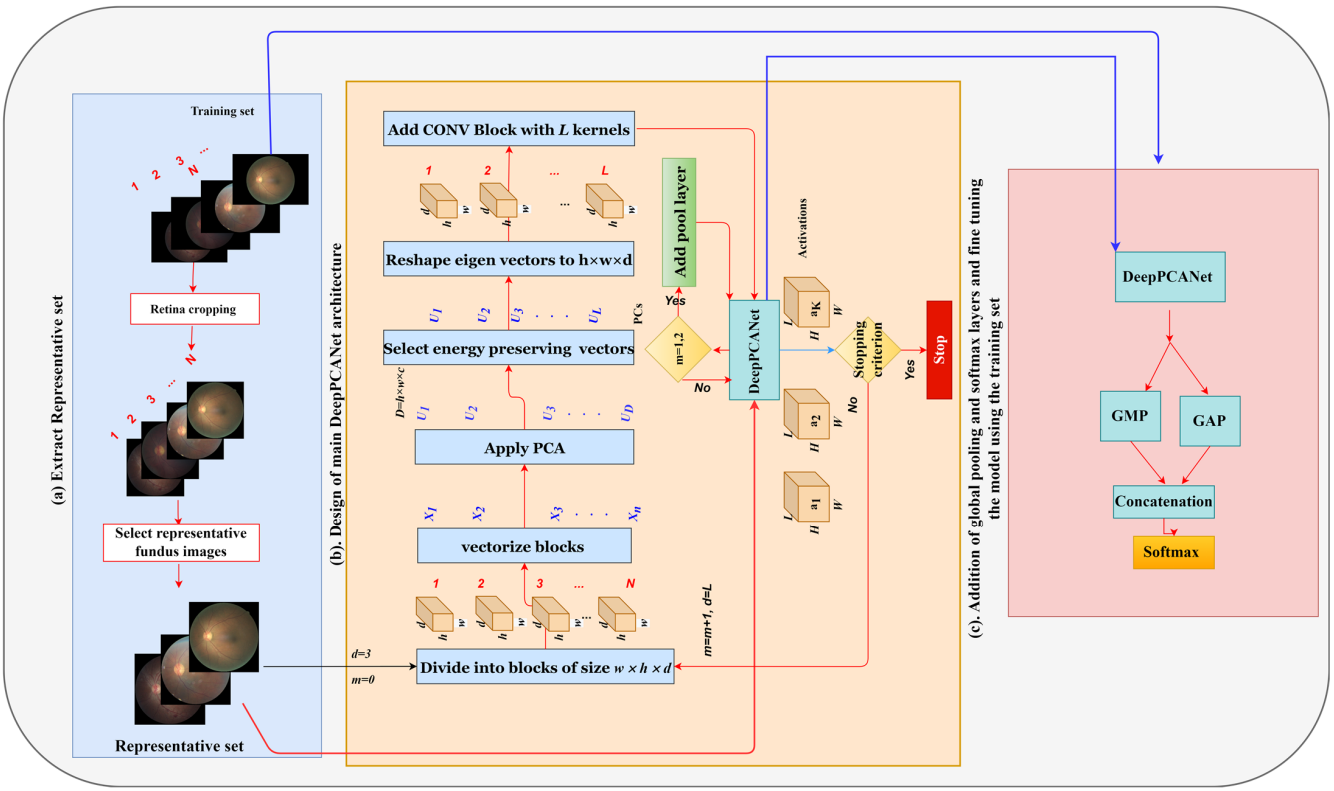


**Figure 1.** Design procedure of DeepPCANet.

#### 4.2.1. Preprocessing

The retinal fundus images are usually not calibrated and are surrounded by a black area, as shown in Figure 1a. To center the retina and remove the black area around it, firstly, the retina circle is cropped, and the background is removed using the method presented in [65], and then it is resized to 512 × 512 pixels. Usually, the DR datasets are imbalanced, i.e., the numbers of images of different classes are significantly different; we increase the data of minority classes using data augmentation. We apply affine transformations to randomly rotate the image with an angle $\theta \in (-180, 180)$.

#### 4.2.2. Selection of Representative Fundus Images

Using the EyePACS training dataset, we choose the most representative fundus images using (K-means [72] and K-medoids [70], and random samples) selection methods to customize a DeepPCANet model and then test it. The discriminative features are extracted from training fundus images for clustering using the efficient LGDBP descriptor proposed in [73]. The number K of clusters for K-means and K-medoids is specified using the gap statistic method [74].

As indicated in Table 1, the K-medoids gave the best results and are the most precise. Due to the fact that K-means gives mean feature vectors as cluster centers, it is inadequate at selecting representative fundus images, and outliers are a serious concern. On the other hand, because the K-medoids algorithm selects representative fundus images as cluster centers, using representative fundus images is appropriate. Both K-medoids and K-means outperform the random fundus image model.

**Table 1.** Comparison between clustering methods based on Eyepacs (SC1).

| Dataset | Model | ACC % | SE % |
|---|---|---|---|
| Eye PACS | Random fundus images | 85.12 | 81.33 |
| | K-means | 89.32 | 83.45 |
| | **K-medoids** | **94.22** | **86.56** |

4.2.3. Designing the Main DeepPCANet Architecture

The design of the AutoML customized architecture of DeepPCANet needs to address two questions, i.e., (i) what should be the depth of the model and (ii) what should be the width of each CONV layer? These questions are addressed by an iterative algorithm, incrementally adding CONV layers, and stopping when a specific criterion is satisfied. It is based on the idea of exploiting discriminative information of fundus images to select the number of kernels in a CONV layer and initialize them. It takes representative fundus images $RI_j$, $j = 1, 2, 3, \ldots, K$ as input and divides them into patches of size $7 \times 7$. The patches are vectorized and used to determine the number of kernels and initialize them. One possible idea is to cluster the patches and select the cluster centers as kernels, but the issue is choosing the number of clusters. We go for a simple and effective procedure, i.e., we employ PCA because it reduces the redundancy and helps to determine the kernels and their number, exploiting the discriminative information in the patches. The principal components (PCs), i.e., the eigenvectors along which the maximum energy is preserved, serve as kernels of the first layer. After computing the PCs, the DeepPCANet is initialized with an input layer and a CONV block (BN+ReLU+CONV) with kernels equal to the number of PCs; the kernels are initialized by reshaping the PCs. Please note that we fix the size of patches to $7 \times 7$ so that the size of kernels of the first CONV layers is $7 \times 7$ following the convention of most of the existing CNN models such as Inception [16], ResNet [17], and DenseNet [18]. Using the current architecture of DeepPCANet, activations $a_j$, $j = 1, 2, 3, \ldots, K$ of representative fundus images $RI_j$, $j = 1, 2, 3, \ldots, K$ are calculated. Inspired by the Fisher ratio [75], using these activations, the ratio of the trace ($TR$) of between-class scatter matrix $S_b$ to the trace of within-class scatter matrix $S_w$ is calculated $TR = \frac{Trace(Sb)}{Trace(Sw)}$ and is used to decide whether to stop or add another CONV block. The new CONV blocks continue to be added as long as TR continues to increase. This criterion ensures that the features generated by DeepPCANet have large inter-class variation and small intra-class scatter. To add a CONV block, the above procedure is repeated with activations $a_j$, $j = 1, 2, 3, \ldots, K$. To reduce the size of feature maps for computational efficiency, pooling layers are added after the first and second CONV blocks. As the kernels and their number are determined from the fundus images, each layer can have a different number of filters. The detail of the design procedure is elaborated in Algorithm 1. It is to be noted that the PCs $(u_i)$, which are used to specify the kernels of a CONV layer, are orthogonal and capture most of the variability in input fundus images, without redundancy, in the form of independent features. The PCs are selected so that the maximum energy is preserved. The energy is measured in terms of the corresponding eigenvalues, i.e., $Energy = \frac{\sum_{l=1}^{L} \lambda_l}{\sum_{j=1}^{D} \lambda_j}$ [23,76] and a threshold value is used to ensure that a certain percentage of energy (e.g., 99%) is preserved. The threshold value of 99% preserves the maximum energy with 209 ($L$) PCs for CONV1 in the EyePACS dataset, as shown in Figure 2. The depth of the AutoML CNN model and the width of each layer are important factors determining the model complexity. Step 7 of Algorithm 1 adaptively determines the best number of kernels that ensure the preservation of the maximum energy of the input image. Step 9 initializes the kernels to be suitable for the DR domain. The selected kernels extract the features from fundus images (five classes) so that the variability of the structures in fundus images is maximally preserved. It is also essential that the features must be discriminative, i.e., have large inter-class variance and small intra-class scatter as we go deeper in the network; it is ensured using the trace ratio

$TR = \frac{Trace(Sb)}{Trace(Sw)}$, the larger the value of the trace ratio, the larger the inter-class variance, and the smaller the intra-class scatter [75]. Step 13 in Algorithm 1 allows adding CONV layers as long as $TR$ increases and determines the data-dependent depth of DeepPCANet. As shown in Figure 3, the maximum ratio is at layers 4, 5, and 16 for KSU-DR, APTOS2019, and EyePACS, respectively. It means that the suitable depth of the DeepPCANet model for the KSU-DR dataset is four layers (Figure 3a), for APTOS2019 it is five layers ((Figure 3b), and for the EyePACS dataset it is sixteen layers (Figure 3c). The model for EyePACS is deeper because it contains many poor quality fundus images, and there is the possibility of label noise because only one expert graded each image in this dataset. Each dataset was collected from a different region and under different conditions using different cameras, so the architecture of the DeepPCANet model is different for each dataset.

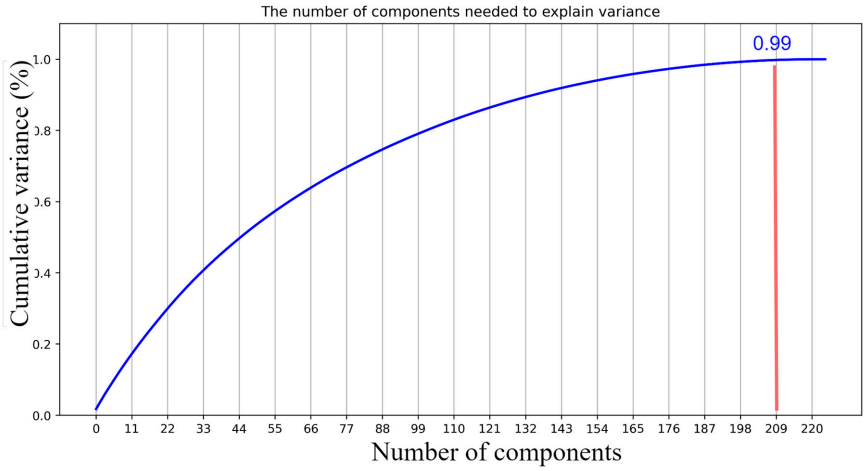


**Figure 2.** Selecting the best threshold. The appropriate threshold is (0.99) and the number of eigenvectors is 209.

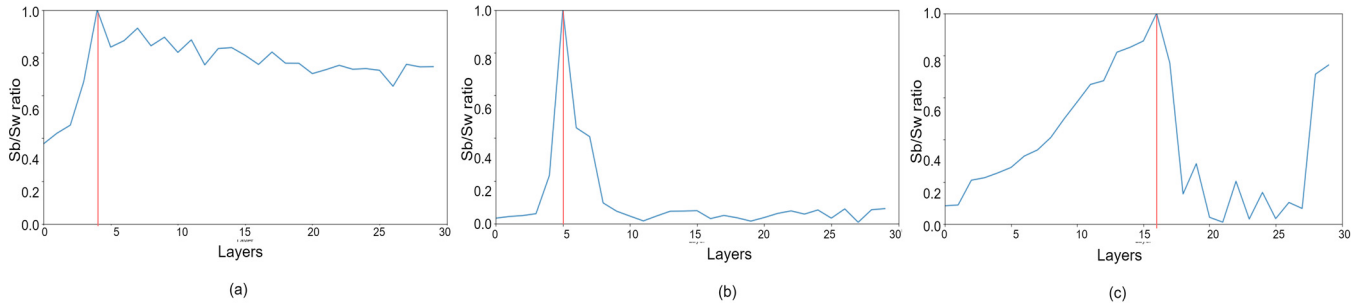


**Figure 3.** Trace ratio between class scatter and within class scatter. The depth for (**a**) APOTOS2019 dataset is 4 layers, (**b**) KSU-DR dataset is 5 layers, and (**c**) EyePACS dataset is 16 layers.

**Algorithm 1.** To design the main DeepPCANet Architecture.

**Input**: Representative fundus images: $RI_j$, $j$ = 1, 2, . . . , $K$ of size $W \times H$ and the class labels $c$ = 1, 2, . . . , $C$; Energy threshold $\varepsilon$
**Output**: The main architecture of DeepPCANet Architecture
**Processing**
**Step 1:** Initialize DeepPCANet with an input layer and set $w$ = 7, $h$ = 7, $d$ = 3, $m$ = 0 (*number of layers*)
**Step 2:** Set $a_j = RI_j$, $j$ = 1, 2, 3, . . . , $K$, and $TRP$ (*previous TR*) = 0.
**Step 3:** Divide $a_j$, $j$ = 1, 2, 3, . . . , $K$, into blocks $b_{ij}$, $i$ = 1, 2, 3, . . . , $B$, $j$ = 1, 2, 3, . . . , $K$, of size $w \times h \times d$, where $d$ is the number of channels (feature maps) in $a_j$ and $B$ is the number of blocks created from each $a_j$.
**Step 4:** Flatten $b_{ij}$ into vectors $\boldsymbol{x}_i \in R^D$, $i = 1, 2, \ldots M$, where $M = K \times B$, and $D = w \times h \times d$.
**Step 5:** Compute zero-center vectors $\phi_i$, $i = 1, 2, \ldots M$ such that $\phi_i = \boldsymbol{x}_i - \bar{\boldsymbol{x}}$, where $\bar{\boldsymbol{x}} = \frac{1}{M}\sum_{l=1}^{M} \boldsymbol{x}_i$.
**Step 6:** Compute the covariance matrix $C = AA^T$, where $A = [\phi_1\ \phi_2 \ldots\ \phi_M]$. Calculate the eigenvalues $\lambda_j$ and eigenvectors $\boldsymbol{u_j}$ $(j = 1, 2, \ldots, D)$ of the covariance matrix $C$.
**Step 7:** Select $L$ eigenvectors $\boldsymbol{u_i}$ $i = 1, 2, \ldots, L$ $(L < D)$ corresponding to the $L$ largest eigenvalues such that $\frac{\sum_{l=1}^{L} \lambda_l}{\sum_{j=1}^{D} \lambda_j} \geq \varepsilon$, where $\varepsilon$ determines the level of energy to be preserved (e.g., $\varepsilon$ = 0.99, for 99% energy preservation).
**Step 8:** The eigenvectors corresponding to the $\frac{\sum_{l=1}^{L} \lambda_l}{\sum_{j=1}^{D} \lambda_j} < \varepsilon$ are summed up to form a single eigenvector, and then stacked at the end of the $L$ eigenvectors.
**Step 9:** Reshape $\boldsymbol{u_i}$, $i = 1, 2, \ldots, L + 1$ to kernels of size $W \times H \times D$ and add the CONV block to DeepPCANet; Update $m = m + 1$.
**Step 10:** If $m$ = 1 or 2, add max pool layers with a pooling window of size 2 × 2 and stride 2 to DeepPCANet.
**Step 11:** Compute the activations $a_j$, $j$ = 1, 2, 3, . . . , $K$ of representative fundus images $RI_j$, $j$ = 1, 2, 3, . . . , $K$ such that $a_j$ = DeepPCANet ($RI_j$).
**Step 12:** Compute the trace ratio between scatter between matrix ($S_b$) and within matrix ($S_w$) as $TR = \frac{Trace(Sb)}{Trace(Sw)}$ where $S_w = \sum_{i=1}^{C} \sum_{j=1}^{n_i} \left(x_j - \mu_i\right)\left(x_j - \mu_i\right)^T$ and $S_b = \sum_{i=1}^{C} n_i\ (\mu_i - \mu)\ (\mu_i - \mu)^t$.
**Step 13:** If $TRP(previous\ TR) \leq TR, set\ TRP = TR$, $W$ = 3, $H$ = 3, $D = L$, and go to Step 3, stop otherwise.

#### 4.2.4. Addition of Global Pool and Softmax Layers

The dimension of the activation of the last CONV block is $W \times H \times L$. If it is flattened and passed to a fully connected (FC) layer, the number of learnable weights and biases of the FC becomes excessively large, which leads to overfitting. To overcome this issue, the activation of the last CONV block is passed simultaneously to global average pooling (GAP), and global max-pooling (GMP) layers [77], which extract the mean and largest feature from each feature map, and these features are fused using a concatenation layer. Both GAP and GMP help to reduce the number of learnable parameters and extract discriminative features from the activation. Finally, a softmax layer is introduced as a classification layer, and the output of the concatenation layer is passed to this layer, as shown in Figure 1c. A dropout layer is also added after the last CONV layer to overcome the overfitting problem.

#### 4.2.5. Finetuning the DeepPCANet Model

After determining the architecture of AutoML's custom-designed DeepPCANet, it is fine-tuned using the training and validation sets. Fine-tuning involves various hyper-parameters: the optimization algorithm, learning rate, batch size, activation function, and dropout probability. We employed the Optuna optimization algorithm [78] to determine the best values of the hyper-parameters. We tested three optimizers (Adam, SGD, and RMSprop), a learning rate between 1e-5 and 1e-1, four batch sizes (5, 10, 15, 20), three activation functions (ReLU, LReLU, and Sigmoid), and dropout probability between 0.25 and 0.50. After training for ten epochs, the Optuna returned the best hyper-parameters for each dataset, as shown in Table 2. The number of kernels in each layer of each model is based on an energy threshold of 0.99. The models for APTOS2019, KSU-DR, and EyePACS

datasets (five classes each) are DeepPCANet-4, DeepPCANet-5, and DeepPCANet-16, respectively, and their specifications are shown in Figure 4. Each dataset has different AutoML architecture because each one is from different ethnicities; the EyePACS dataset is from the USA, APTOS2019 is from India, and KSU-DR is from KSA, as well as the use of retinal images captured using different cameras. To confirm the distinct architectures for the three DR datasets, we combined the extracted K-medoids fundus images into a single dataset, generated a custom DeepPCANet, and tested it on the three datasets. As illustrated in Table 3, the outcome is not as good as that obtained using the customized DeepPCANet for each DR dataset, as illustrated in Tables 4 and 5. Each dataset is from different ethnicities; the EyePACS dataset is from the USA, APTOS2019 is from India, and KSU-DR is from KSA; as well as the use of different retinal images captured using different cameras, so each dataset has a different custom-designed model. After fixing the hyper-parameters, each model is fine-tuned using training and validation sets for 100 epochs. The fine-tuned model is tested using the testing set.

**Table 2.** The best hyper parameters found using Optuna algorithm (SC1).

| Dataset | Activation Function | Learning Rate | Patch's Size | Optimizer | Dropout |
|---|---|---|---|---|---|
| **KSU-DR** | LReLU | 0.0001 | 10 | RMSprop | 0.50 |
| **EyePACS** | LReLU | 0.0055 | 10 | RMSprop | 0.38 |
| **APTOS2019** | LReLU | 0.0007 | 5 | RMSprop | 0.40 |

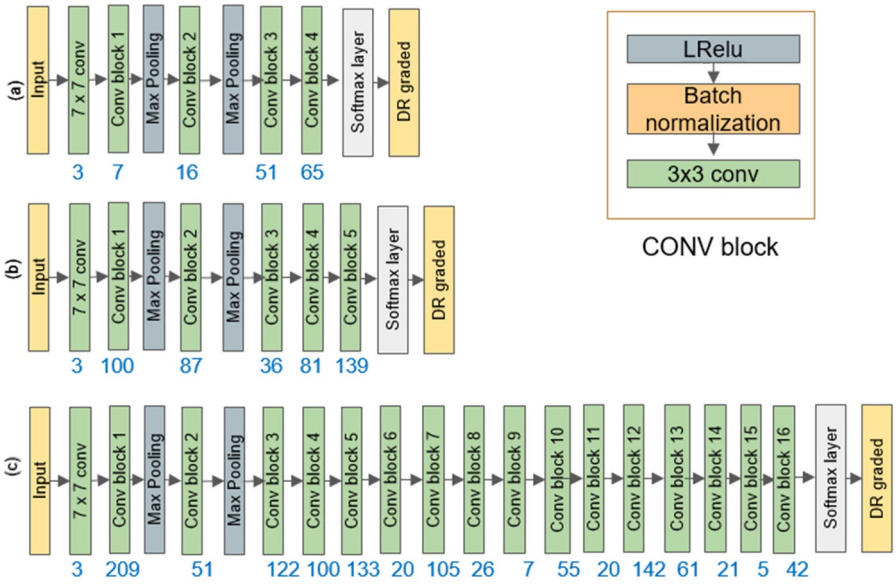


**Figure 4.** DeepPCANet architecture for (**a**) APOTS2019, (**b**) KSU-DR, and (**c**) EyePACS datasets.

**Table 3.** Customize the DeepPCANet by combining the extracted K-medoids fundus images to confirm the distinct architectures for the three DR datasets.

| Dataset | Model | Performance (%) | | | |
|---|---|---|---|---|---|
| | | ACC | SE | SP | Kappa |
| **EyePACS (SC1)** | PCANet model (mixed dataset) | 73 | 28 | 83 | 8 |
| **APTOS2019 (SC1)** | | 88 | 36 | 91 | 51 |
| **KSU-DR (SC2)** | | 80 | 81 | 81 | 59 |

**Table 4.** Comparison between DeepPCANet models and the pretrained models for SC1 scenario, M and K stand for millions and thousands.

| Dataset | Model | #FLOPs | # Parameters | ACC % | SE % | SP % | Kappa % |
|---|---|---|---|---|---|---|---|
| APTOS2019 | ResNet152 | 5.6 M | 60.19 M | 95.25 | 88.22 | 96.97 | 88.15 |
| | DenseNet121 | 1.44 M | 7.98 M | 96.58 | 91.55 | 97.82 | 89.22 |
| | ResNeSt50 | 5.39 M | 27.5 M | 97.11 | 92.29 | 98.2 | 90.82 |
| | **DeepPCANet-4** | **1.36 M** | **63.7 K** | **98.21** | **95.29** | **98.9** | **94.32** |
| EyePACS | ResNet152 | 5.6 M | 60.19 M | 92.25 | 80.74 | 94.9 | 75.16 |
| | DenseNet121 | 1.44 M | 7.98 M | 91.14 | 80.07 | 95 | 74.84 |
| | ResNeSt50 | 5.39 M | 27.5 M | 93.12 | 82.33 | 95.21 | 78 |
| | **DeepPCANet-16** | **2.11 M** | **557.68 K** | **94.22** | **86.56** | **96.30** | **81.64** |

**Table 5.** Comparison between DeepPCANet model and the pretrained models for SC2 scenario.

| Dataset | Model | #FLOPs | # Parameters | ACC % | SE % | SP % | Kappa % |
|---|---|---|---|---|---|---|---|
| KSU-DR dataset | ResNet152 | 5.6 M | 60.19 M | 97.98 | 97.83 | 97.83 | 95.75 |
| | DenseNet121 | 1.44 M | 7.98 M | 98.51 | 96.06 | 98.86 | 96.4 |
| | ResNeSt50 | 5.39 M | 27.5 M | 99.47 | 99.46 | 99.46 | 98.93 |
| | **DeepPCANet-5** | **1.375 M** | **73.66 K** | **99.5** | **99.5** | **99.5** | **98.99** |
| APTOS2019 | ResNet152 | 5.6 M | 60.19 M | 95 | 94.44 | 94.44 | 89.80 |
| | DenseNet121 | 1.44 M | 7.98 M | 99.32 | 98.8 | 98.8 | 98.73 |
| | ResNeSt50 | 5.39 M | 27.5 M | 98.33 | 96.54 | 96.53 | 94.22 |
| | **DeepPCANet-4** | **1.36 M** | **63.7 K** | **99.7** | **99.44** | **99.44** | **99.3** |
| EyePACS | ResNet152 | 5.6 M | 60.19 M | 91.36 | 90.94 | 92.25 | 82.53 |
| | DenseNet121 | 1.44 M | 7.98 M | 91.51 | 91.75 | 91.75 | 82.72 |
| | ResNeSt50 | 5.39 M | 27.5 M | 90.53 | 90.92 | 90.92 | 79.04 |
| | **DeepPCANet-16** | **2.11 M** | **557.68 K** | **94.44** | **94.28** | **94.28** | **88.71** |

## 5. Experiments and Results

This section first describes the evaluation protocol and the experiments performed to evaluate the proposed method and then presents the results.

### *5.1. Evaluation Protocol*

We determined the architecture of the DeepPCANet for each DR dataset and fine-tuned it using the training set of the corresponding DR database; the detail is given in Section 3. After that, the performance of each model was evaluated using the test set of the related database. To validate the usefulness and the superiority of the model design technique, we compared custom-designed models with the widely used state-of-the-art pre-trained CNN models such as ResNet [17], DenseNet [18], and ResNeSt [39], which have shown outstanding performance for various computer vision applications. Additionally, we compared it with AutoML models (Google Cloud AutoML) and Auto-Keras. We fine-tuned the competing models using the same procedure employed for DeepPCANet on each dataset.

For evaluation, we adopted three scenarios SC1 [62,63], SC2 [61,62], and SC3, as described in Section 5.1. We evaluated the AutoML custom-designed models using SC1 and SC2 on APTOS2019 and EyePACS and SC3 in EyePACS. However, the evaluation of the KSU-DR dataset was performed using SC2 because the number of images for five

classes is not enough. In addition, we used four commonly used metrics in medical application and deep learning models: accuracy (ACC), sensitivity (SE), specificity (SP), and Kappa [13,79–82].

#### 5.1.1. Five Class Problem (SC1)

Using the APTOS2019 and EyePACS datasets, we built DeepPCANet-4 and DeepPCANet-16 models, respectively, for SC1 using the respective training sets and fined-tuned them using the corresponding training and validation sets (see detail in Section 3). After fine-tuning, the models were evaluated on test datasets of EyePACS and APTOS2019; the results are shown in Table 4. The results of the ResNet152, DenseNet121, and ResNeSt50 models, fine-tuned using the same training set and evaluated using the same testing set as for from EyePACS and APTOS2019, are also shown in Table 4. The results show that DeepPCANet-4 and DeepPCANet-16 outperform ResNet152, DenseNet121, and ResNeSt50 on both datasets in terms of all metrics; in particular, in both cases, the sensitivity and Cohen's Kappa are higher than those of ResNet152, DenseNet12, and ResNeSt50, Cohen's Kappa is considered a more robust statistical measure than accuracy [83,84]. The DeepPCANet-4 has the lowest number of FLOPs (1.36 M) and learnable parameters (63.7 K) among all competing models, as shown in Table 4.

DeepPCANet-16 has fewer learnable parameters than the pertained ResNet152, DenseNet121, and ResNeSt50 and has fewer FLOPs than ResNet152 and ResNeSt50 models, but slightly greater than DenseNet121. In contrast, it has the best performance in terms of metrics on the EyePACS dataset. ResNeSt50 has better performance than ResNet152 and DenseNet121. To compare the AutoML DeepPCANet to the state-of-the-art AutoML methods, we use the most DR-intensive dataset available, the EyePACS, based on scenario SC1. According to the NAS method [24], we test two AutoML methods; the Google Cloud (vision) AutoML [43] and Auto-Keras [23]. We set up and generated the AutoML using Auto-Keras methods locally using the same device and based on the representative set, fine-tuned the generated CNN model using a training and validation set; then, it was evaluated using the test set as with DeepPCANet-16. For Google Cloud AutoML, we upload the representative, training, validation, and test sets to the Google cloud storage and follow the same evaluation procedure. DeepPCANet-16 outperformed the Google Cloud AutoML, and Auto-Keras has fewer number FLOPs, as shown in Table 6, but its performance is lower than both models. The FLOPs and number of parameters of Google Cloud AutoML are hidden, showing only the precision (PR) and recall (SE) metrics. NAS algorithms are time-consuming and resource-intensive; they typically look for the cell structure, including the topology of the connections and the operation (transformation) that connects each cell. After that, the resulting cell is replicated to construct the neural network [85]. We used a basic simple cell structure throughout our AutoML DeepPCANet (LReLU, batch normalization layer, and CONV layer). The filters in the CONV layers are derived automatically from fundus' lesions and require less time. They optimized both the search architecture and hyper-parameters in NAS algorithms. In contrast, we first derived the optimal DeepPCANet architecture and then used Optuna to optimize the hyper-parameters, as shown in Table 2.

**Table 6.** Comparison between DeepPCANeT-16 and AutoML methods.

| Dataset | Model | #FLOPs | # Parameters | ACC % | SE % | PR % |
|---|---|---|---|---|---|---|
| EyePACS | **Auto-Keras** | 0.31 M | 15 M | 73 | 73 | 53 |
| | **Google-AutoML** | Hidden | Hidden | – | 71.43 | 79.1 |
| | **DeepPCANet-16** | 2.11 M | **557.68 K** | **94.44** | **94.28** | **96.12** |

#### 5.1.2. Two Class Problem (SC2)

We validated the DeepPCANet models' performance using the three datasets for SC2. The custom-designed models DeepPCANet-5, DeepPCANet-4, and DeepPCANet-16

for KSU-DR, APTOS2019, and EyePACS, respectively, which were designed and fine-tuned using only fundus images, outperform the highly complex CNN models such as ResnNet152, DenseNet121, and ResNeSt50, which were trained using ImageNet dataset and fine-tuned using fundus images, in terms of all metrics, as shown in Table 5. Though DenseNet121 outperforms ResNet152 and ResNeSt50 on the three datasets, its performance is not better than the custom-designed models. DeepPCANet-5 involves 1.375 M FLOPs, which is smaller than the number of FLOPs of ResNet152, DenseNet121, and ResNeSt50. The number of learnable parameters of DeepPCANet-5 is 73.66K which is much smaller than those of the pre-trained models ResNet152 (60.19 M), DenseNet121 (7.98 M), and ResNeSt50 (27.5 M). In Figure 5, we provide illustrations of the ROC curves on the three datasets using the four models (customized DeepPCANet, ResNet152, DenseNet121, and ResNeSt50). It indicates that the DeepPCANet models' performance is better than the three pre-trained models on the three datasets.

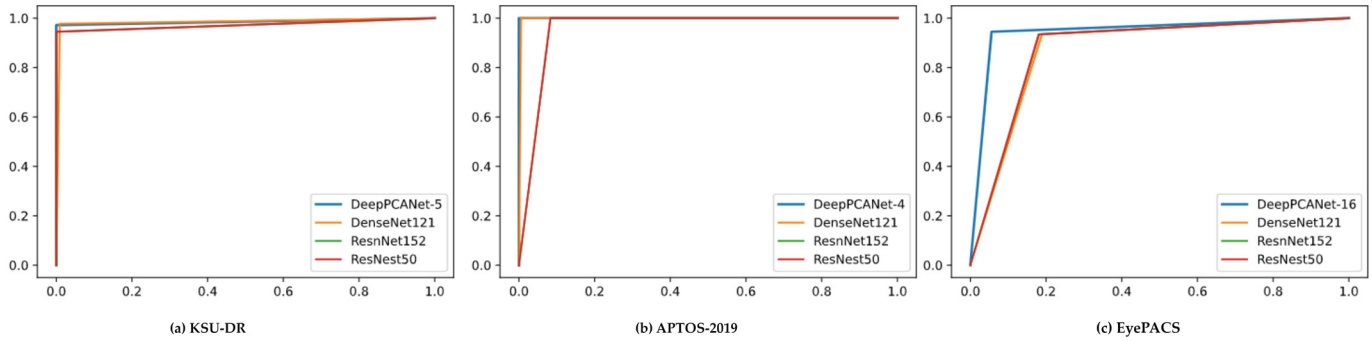


**Figure 5.** ROC curve for custom-designed DeepPCANet and the pretrained models for SC2 scenario and datasets: (**a**) KSU-DR, (**b**) APTOS-2019, (**c**) EyePACS.

### *5.2. Visualization*

To understand the decision-making mechanism of the custom-designed CNN models, we created the visual feature maps using the gradient-weighted class activation mapping (GradCam) visualization method [86]. The visual feature maps of two random fundus images generated by the DeepPCANet-5 model customized for the local KSU-DR dataset are shown in Figure 6d,h. The same fundus images were given blindly to two expert ophthalmologists at King Khalid Hospital of KSU, and they independently specified the lesion regions manually. Though there is a slight difference in the annotations of both experts, they agreed on most of the lesions, as shown in Figure 6b,c for the fundus image Figure 6a from class moderate and Figure 6f,g for the fundus image Figure 6e from class PDR. The visual features maps of the DeepPCANet-5 model highlight the lesions annotated by both experts, as shown in Figure 6d,h. The yellow and orange splatter in Figure 6d,h indicates that the DeepPCANet-5 model makes decisions based on the features learned from the lesion regions.

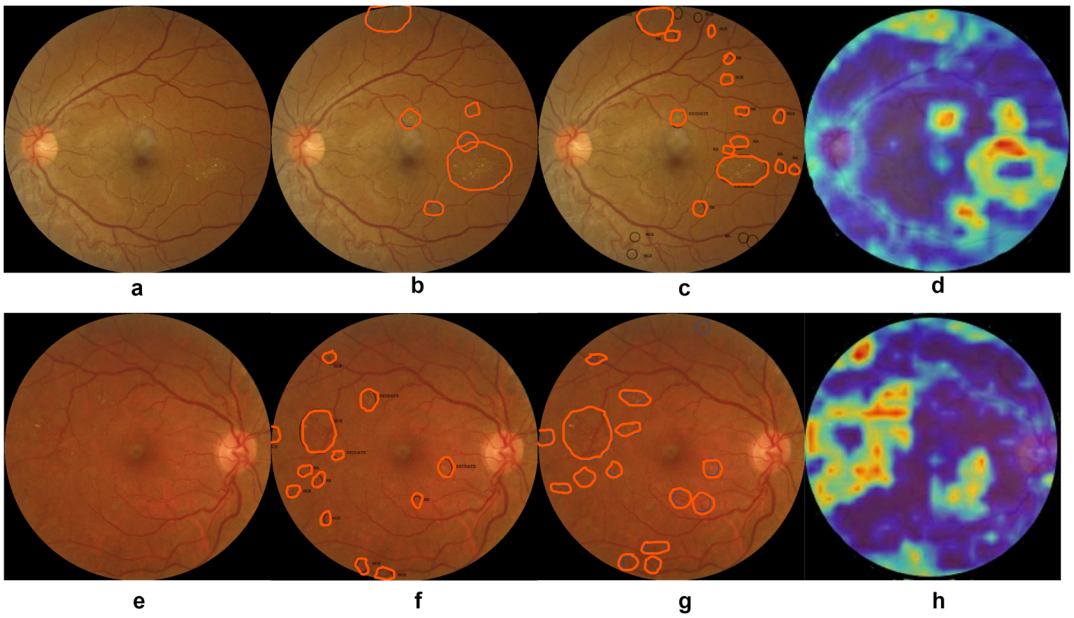


**Figure 6.** Visualization of the decision-making mechanism of DeepPCANet-5 model. (**a**) Fundus image from class moderate, (**b**,**c**) lesions specified by experts 1 and 2, respectively, (**d**) DeepPCANet-5 map (**e**) fundus image from class PDR, (**f**,**g**) lesions specified by experts 1 and 2, respectively, (**h**) DeepPCANet-5 map.

## 6. Discussions

This study proposed a technique to auto-custom-design a DeepPCANet model for a target DR dataset. The depth of the model and the width of each layer is not specified randomly or by exhaustive experiments. The custom-designed DeepPCANet models for DR screening have small depths and varying widths of CONV layers and involve a small number of learnable parameters. The results of the AutoML DeepPCANet models customized for the KSU-DR, APTOS2019, and EyePACS datasets (presented in Tables 4–6) demonstrate that it outperforms the well-known highly complex pre-trained models ResNet152, DenseNet121, and ResNeSt50, as well as AutoML from Google and Auto-Keras that was fine-tuned using the same DR datasets. Generally, the DeepPCANet got competitive performance with a small number of layers and parameters. As shown in Table 4, the custom-designed DeepPCANet models for the three datasets have a small number of parameters in thousands against that number in millions of ResNet152, DenseNet121, and ResNeSt50. DeepPCANet-4 and DeepPCANet-5 have fewer FLOPs than all pre-trained models and have better performance. The DeepPCANet-16 has fewer FLOPs than that of ResNet152 and ResNeSt50 and also has better performance. Though DenseNet121 has fewer FLOPs than DeepPCANet-16, it has the least performance and a large number of parameters. The reason for the lightweight structures and superior performance of custom-designed DeepPCANet models is that their architectures have been directly drawn from the fundus images, unlike the state-of-the-art CNN models, which are mainly designed for object detection. In addition to comparing the custom-designed DeepPCANet models with famous pre-trained models, it is essential to validate their effectiveness in DR screening by comparing them to the state-of-the-art methods on two challenging datasets (APOTS2019 and EyePACS). DeepPCANet-4 generated for SC1 on the APTOS2019 dataset outperforms the state-of-the-art methods on the same dataset in terms of accuracy, sensitivity, specificity, and Kappa, as shown in Table 7. The DeepPCANet-4 based on the five-class problem (SC1) and APTOS2019 dataset outperforms the method presented in Sikder et al. (2021), which used handcrafted features and needs a long processing time for fundus image preprocessing and extracting features. Though the method by Tymchenko et al., 2020 [64] outperforms the DeepPCANet-4 in the Kappa score for the five-class problem (SC1), it has less accuracy, sensitivity, and specificity, and it is based on a highly complex ensemble of 20 CNN models.

For the same scenario, the DeepPCANet-16 designed for EyePACS outperforms the existing methods in accuracy and specificity. The method by Islam et al., 2018 [61] obtained higher sensitivity, but their model is more complex, and it was tested on 4% of the dataset, as shown in Table 7. For the SC2 (normal vs. DR levels), DeepPCANet-4 outperforms the method by Tymchenko et al. [64] in all metrics on APTOS2019. In this scenario, on the EyePACS dataset, as shown in Table 7, the DeepPCANet-16 is better than other methods in accuracy, sensitivity, and specificity; the method by Islam et al., 2018 [61] is slightly better than DeepPCANet-16 in sensitivity, but it was tested only on 4% of the EyePACS dataset. The method of Chetoui et al., 2020 [87] is better than DeepPCANet-16, whereas they used transfer learning based on Inception-Resnet-v2, which has high complexity and a large number of parameters. It consists of five convolutional layers, each followed by batch normalization, two pooling layers, forty-three inception modules, three residual connections, the pooling of global averages, and the use of two fully connected layers in conjunction with the rectified linear unit (ReLU); whereas, DeepPCANet-16 is a 16-layer structure that employs the basic CONV setup. The DeepPCANet-16, based on the EyePACS dataset for the SC3 (0 and 1 vs. DR levels), obtained less accuracy than Colas et al. [67] and Islam et al. [61] but obtained higher sensitivity and specificity, which are more important and robust than accuracy in the medical applications [88].

**Table 7.** Comparison between DeepPCANet models and state-of-the-art methods.

| Paper | Method | Dataset | Performance (%) | | | |
|---|---|---|---|---|---|---|
| | | | ACC | SE | SP | Kappa |
| **Five classes (SC1)** | | | | | | |
| **Sikder et al., 2019 [65]** | Colored features extraction using ensemble | APTOS2019 | 91 | 89.54 | - | - |
| **Tymchenko et al., 2020 [64]** | An ensembled models with 3 CNN architectures fficientNet-B4, EfficientNet-B5, and SE-ResNeXt50 | APTOS2019 | 91.9 | 84 | 98 | 96.9 |
| **Shorfuzzaman et al., 2021 [89]** | CNN-based transfer learning ensemble | aptos2019 | 96.2 | 94.00 | - | - |
| **Sikder et al. (2021)** | Histogram and GLCM features tuning using XGBoostand genetic algorithm. | aptos2019 | 94.20 | - | - | - |
| **DeepPCANet-4** | **DeepPCANet model customized for APTOS2019 dataset** | **APTOS2019 (test: 40% of public dataset)** | **98.21** | **95.28** | **98.85** | **94.32** |
| **Lahmar et al., 2021 [90]** | Transfer learning (MobileNet V2) | APTOS2019 (SC1) | 93.09 | 89.27 | 92.69 | - |
| **Islam et al., 2018 [61]** | A CNN model consisting of 18 layers with 3 × 3 and 4 × 4 kernels | EyePACS (test: 4% of dataset) | | 94.5 | 90.2 | |
| **Li et al., 2019 [62]** | Deep learning model based on DCNN | EyePACS (test: 10% of dataset) | 86.17 | - | - | - |
| **DeepPCANet-16** | **DeepPCANet model customized for EyePACS dataset** | **EyePACS (test: 10% of public dataset)** | **94.22** | **89.56** | **96.30** | **81.64** |

**Table 7.** *Cont.*

| Paper | Method | Dataset | Performance (%) | | | |
|---|---|---|---|---|---|---|
| | | | ACC | SE | SP | Kappa |
| **Normal vs. DR (all DR levels) (two classes) (SC2)** | | | | | | |
| **Tymchenko et al., 2020 [64]** | Ensembled models with 3 CNN architectures EfficientNet-B4, EfficientNet-B5, and SE-ResNeXt50 | APTOS2019 | 99.3 | 99.3 | 99.3 | 98.6 |
| **DeepPCANet-4** | **DeepPCANet model customized for APTOS2019 dataset** | **APTOS2019 (test: 10% of public dataset)** | **99.7** | **99.44** | **99.44** | **99.3** |
| **Islam et al., 2018 [61]** | A CNN model consisting of 18 layers with 3 × 3 and 4 × 4 kernels | EyePACS (test: 4% of public dataset) | | 94.5 | 90.2 | - |
| **Li et al., 2019 [62]** | Features extraction using deep learning model based on DCNN and SVM classification | EyePACS | 91.05 | 89.30 | 90.89 | - |
| **Chetoui et al., 2020 [87]** | Pretrained Inception-Resnet-v2 DCNN | EyePACS (SC2) | 97.9 | 95.8 | 97.1 | 98.6 |
| **DeepPCANet-16** | DeepPCANet model customized for EyePACS dataset | EyePACS (test: 10% of public dataset) | 94.44 | 94.28 | 94.28 | 88.71 |
| **Non-referral (Normal and DR grad 1) vs. referral (DR grade 2 to highest grade) (two classes) (SC3)** | | | | | | |
| **Colas et al., 2016 [67]** | A CNN model. End to end training. | EyePACS (train: 89%, test: 11%) | 96.2 | 66.6 | 94.6 | |
| **Islam et al., 2018 [61]** | A CNN model | EyePACS (train: 96%, test: 4%) | 98 | 94 | | |
| **DeepPCANet-16** | DeepPCANet model derived from EyePACS dataset | EyePACS (test: 10% of public dataset) | 94.59 | 94.86 | 94.87 | 89.02 |

## 7. Conclusions

We introduced an approach to building an AutoML data-dependent CNN model (DeepPCANet) customized for DR screening automatically. This approach tackles the limitations of the available annotated DR datasets and the problem of a vast search space and a huge number of parameters in a deep CNN model. It bauto-lightweightghtweight CNN model customized for a target DR dataset using k-medoid clustering, principal component analysis (PCA), and inter-class and intra-class variations. The DeepPCANet model is data-dependent, and each DR dataset has its appropriate AutoML architecture. The customized models, DeepPCANet-5 for the local KSU-DR dataset, DeepPCANet-4 for APTOS2019, and DeepPCANet-16 for the EyePACS dataset outperform the pre-trained very deep and highly complex ResNet152, DenseNet121, and ResNeSt50 models fine-tuned using the same datasets and procedure. The performance, complexity, and number of parameters of the customized DeepPCANet models are significantly less than ResNet152 and ResNeSt50. Though DenseNet121 has fewer FLOPs than DeepPCANet-16, it has the least performance and a large number of parameters. On the EyePACS dataset, compared to the Google Cloud AutoML and Auto-Keras, DeepPCANet-16 based on SC1 obtained better performance with fewer parameters. Using the EyePACS dataset, DeepPCANet-16 also compared to the state-of-the-art methods (for SC2 and SC3), the DeepPCANet-16 has less complexity and parameters and has competitive performance. The DeepPCANet fails to predict DR grade from fundus images, which have poor quality. It could not correctly grade some poor quality fundus images from the EyePACS dataset; each image in this dataset was graded by only one expert from the geographic region of California, which can potentially lead to annotation bias. How the DeepPCANet can reliably predict the DR

grade from poor quality fundus images is a subject of future work. Additionally, how the DeepPCANet can be generalized with different fundus datasets is a subject of future work.

**Author Contributions:** Conceptualization, F.S. and M.H.; data curation, F.S., F.A.A. and A.M.A.O.; formal analysis, F.S., F.A.A. and A.M.A.O.; funding acquisition, methodology, F.S. and M.H.; project administration, M.H. and H.A.A.; resources, M.H. and H.A.A.; software, F.S.; supervision, M.H. and H.A.A.; validation, F.S.; visualization, F.S.; writing original draft, F.S.; writing—review and editing, M.H. All authors have read and agreed to the published version of the manuscript.

**Funding:** The authors extend their appreciation to the Deputyship for Research and Innovation, Ministry of Education in Saudi Arabia for funding this research work through the project no. (IFKSURG-2-108).

**Data Availability Statement:** Public domain datasets were used for experiments.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Yau, J.W.Y.; Rogers, S.L.; Kawasaki, R.; Lamoureux, E.L.; Kowalski, J.W.; Bek, T.; Chen, S.-J.; Dekker, J.M.; Fletcher, A.; Grauslund, J.; et al. Global Prevalence and Major Risk Factors of Diabetic Retinopathy. *Diabetes Care* **2012**, *35*, 556–564. [CrossRef] [PubMed]
2. Quellec, G.; Charrière, K.; Boudi, Y.; Cochener, B.; Lamard, M. Deep image mining for diabetic retinopathy screening. *Med. Image Anal.* **2017**, *39*, 178–193. [CrossRef] [PubMed]
3. Sreejini, K.; Govindan, V. Retrieval of pathological retina images using Bag of Visual Words and pLSA model. *Eng. Sci. Technol. Int. J.* **2019**, *22*, 777–785. [CrossRef]
4. Hemanth, D.J.; Deperlioglu, O.; Kose, U. An enhanced diabetic retinopathy detection and classification approach using deep convolutional neural network. *Neural Comput. Appl.* **2020**, *32*, 707–721. [CrossRef]
5. Stolte, S.; Fang, R. A Survey on Medical Image Analysis in Diabetic Retinopathy. *Med. Image Anal.* **2020**, *64*, 101742. [CrossRef]
6. Ting, D.S.W.; Pasquale, L.R.; Peng, L.; Campbell, J.P.; Lee, A.Y.; Raman, R.; Tan, G.S.W.; Schmetterer, L.; Keane, P.A.; Wong, T.Y. Artificial intelligence and deep learning in ophthalmology. *Br. J. Ophthalmol.* **2019**, *103*, 167–175. [CrossRef]
7. Gu, J.; Wang, Z.; Kuen, J.; Ma, L.; Shahroudy, A.; Shuai, B.; Liu, T.; Wang, X.; Wang, G.; Cai, J.; et al. Recent advances in convolutional neural networks. *Pattern Recognit.* **2018**, *77*, 354–377. [CrossRef]
8. Grigorescu, S.; Trasnea, B.; Cocias, T.; Macesanu, G. A survey of deep learning techniques for autonomous driving. *J. Field Robot.* **2020**, *37*, 362–386. [CrossRef]
9. Abou Arkoub, S.; Hajjam El Hassani, A.; Lauri, F.; Hajjar, M.; Daya, B.; Hecquet, S.; Aubry, S. Survey on Deep Learning Techniques for Medical Imaging Application Area. In *Machine Learning Paradigms*; Springer: Cham, Switzerland, 2020; pp. 149–189.
10. Mohamadou, Y.; Halidou, A.; Kapen, P.T. A review of mathematical modeling, artificial intelligence and datasets used in the study, prediction and management of COVID-19. *Appl. Intell.* **2020**, *50*, 3913–3925. [CrossRef]
11. Zhu, H.; Shu, S.; Zhang, J. FAS-UNet: A Novel FAS-Driven UNet to Learn Variational Image Segmentation. *Mathematics* **2022**, *10*, 4055. [CrossRef]
12. Lam, C.; Yu, C.; Huang, L.; Rubin, D. Retinal lesion detection with deep learning using image patches. *Investig. Ophthalmol. Vis. Sci.* **2018**, *59*, 590–596. [CrossRef] [PubMed]
13. Gao, Z.; Li, J.; Guo, J.; Chen, Y.; Yi, Z.; Zhong, J. Diagnosis of Diabetic Retinopathy Using Deep Neural Networks. *IEEE Access* **2019**, *7*, 3360–3370. [CrossRef]
14. Shaik, N.S.; Cherukuri, T.K. Hinge attention network: A joint model for diabetic retinopathy severity grading. *Appl. Intell.* **2022**, *52*, 15105–15121. [CrossRef]
15. Gao, Z.; Jin, K.; Yan, Y.; Liu, X.; Shi, Y.; Ge, Y.; Pan, X.; Lu, Y.; Wu, J.; Wang, Y.; et al. End-to-end diabetic retinopathy grading based on fundus fluorescein angiography images using deep learning. *Graefe's Arch. Clin. Exp. Ophthalmol.* **2022**, *260*, 1663–1673. [CrossRef] [PubMed]
16. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 1–9.
17. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
18. Huang, G. Densely connected convolutional networks. CVPR 2017. *arXiv* **2016**, arXiv:1608.06993.
19. Saini, M.; Susan, S. Diabetic retinopathy screening using deep learning for multi-class imbalanced datasets. *Comput. Biol. Med.* **2022**, *149*, 105989. [CrossRef]
20. Glorot, X.; Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, Sardinia, Italy, 13–15 May 2010; pp. 249–256.
21. He, K.; Zhang, X.; Ren, S.; Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In Proceedings of the IEEE International Conference on Computer, Santiago, Chile, 7–13 December 2015; Volume 22, pp. 1026–1034.

22. Krähenbühl, P. Data-dependent initializations of convolutional neural networks. *arXiv* **2015**, arXiv:1511.06856.
23. Suau, X.; Apostoloff, N. Filter Distillation for Network Compression. In Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV), Piscataway, NJ, USA, 1–5 March 2020; pp. 3129–3138.
24. Singh, V.K.; Joshi, K. Automated Machine Learning (AutoML): An overview of opportunities for application and research. *J. Inf. Technol. Case Appl. Res.* **2022**, *24*, 75–85. [CrossRef]
25. Thornton, C. Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 11–14 August 2013; pp. 847–855.
26. Komer, B.; Bergstra, J.; Eliasmith, C. Hyperopt-sklearn: Automatic hyperparameter conTableuration for scikit-learn. In *ICML Workshop on AutoML*; Citeseer: Austin, TX, USA, 2014; Volume 9.
27. Olson, R.S.; Bartley, N.; Urbanowicz, R.J.; Moore, J.H. Evaluation of a tree-based pipeline optimization tool for automating data science. In Proceedings of the Genetic and Evolutionary Computation Conference, Denver, CO, USA, 20–24 June 2016; pp. 485–492.
28. Feurer, M.; Springenberg, J.; Hutter, F. Initializing bayesian hyperparameter optimization via meta-learning. In Proceedings of the AAAI Conference on Artificial Intelligence, Austin, TX, USA, 25–30 January 2015; Volume 29. No. 1.
29. Jin, H.; Song, Q.; Hu, X. Auto-keras: An efficient neural architecture search system. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, New York, NY, USA, 4–8 August 2019; pp. 1946–1956.
30. Doke, A.; Gaikwad, M. Survey on Automated Machine Learning (AutoML) and Meta learning. In Proceedings of the 2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT), Kharagpur, India, 6–8 July 2021; IEEE: Piscataway, NJ, USA, 2021.
31. Stojadinovic, M.; Milicevic, B.; Jankovic, S. Improved predictive performance of prostate biopsy collaborative group risk calculator when based on automated machine learning. *Comput. Biol. Med.* **2021**, *138*, 104903. [CrossRef]
32. Aloraini, T.; Aljouie, A.; Alniwaider, R.; Alharbi, W.; Alsubaie, L.; AlTuraif, W.; Qureshi, W.; Alswaid, A.; Eyiad, W.; Al Mutairi, F.; et al. The variant artificial intelligence easy scoring (VARIES) system. *Comput. Biol. Med.* **2022**, *145*, 105492. [CrossRef]
33. Zoph, B.; Le, Q.V. Neural architecture search with reinforcement learning. *arXiv* **2016**, arXiv:1611.01578.
34. Domhan, T.; Springenberg, J.T.; Hutter, F. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In Proceedings of the Twenty-fourth International Joint Conference on Artificial Intelligence, Buenos Aires, Argentina, 25–31 July 2015.
35. Tan, R.Z.; Chew, X.; Khaw, K.W. Neural Architecture Search for Lightweight Neural Network in Food Recognition. *Mathematics* **2021**, *9*, 1245. [CrossRef]
36. Liu, H. Hierarchical representations for efficient architecture search. *arXiv* **2017**, arXiv:1711.00436.
37. Kaggle. Diabetic Retinopathy Detection (Kaggle). 2019. Available online: https://www.kaggle.com/c/diabetic-retinopathy-detection/data (accessed on 26 September 2021).
38. Kaggle. APTOS Blindness Detection. 2019. Available online: https://www.kaggle.com/c/aptos2019-blindness-detection (accessed on 26 September 2021).
39. Zhang, H. Resnest: Split-attention networks. *arXiv* **2020**, arXiv:2004.08955.
40. Mateen, M.; Wen, J.; Hassan, M.; Nasrullah, N.; Sun, S.; Hayat, S. Automatic Detection of Diabetic Retinopathy: A Review on Datasets, Methods and Evaluation Metrics. *IEEE Access* **2020**, *8*, 48784–48811. [CrossRef]
41. Soomro, T.A.; Afifi, A.J.; Zheng, L.; Soomro, S.; Gao, J.; Hellwich, O.; Paul, M. Deep learning models for retinal blood vessels segmentation: A review. *IEEE Access* **2019**, *7*, 71696–71717. [CrossRef]
42. Asiri, N.; Hussain, M.; Al Adel, F.; Alzaidi, N. Deep learning based computer-aided diagnosis systems for diabetic retinopathy: A survey. *Artif. Intell. Med.* **2019**, *99*, 101701. [CrossRef]
43. Bhandari, S.; Pathak, S.; Jain, S.A.; Deshmukh, V. A Review on Swarm intelligence & Evolutionary Algorithms based Approaches for Diabetic Retinopathy Detection. In Proceedings of the 2022 IEEE World Conference on Applied Intelligence and Computing (AIC), Sonbhadra, India, 17–19 June 2022; IEEE: Piscataway, NJ, USA, 2022.
44. Han, S.; Mao, H.; Dally, W.J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv* **2015**, arXiv:1510.00149.
45. He, Y.; Zhang, X.; Sun, J. Channel pruning for accelerating very deep neural networks. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 1389–1397.
46. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv* **2017**, arXiv:1704.04861.
47. Jiang, C.; Li, G.; Qian, C.; Tang, K. Efficient DNN Neuron Pruning by Minimizing Layer-wise Nonlinear Reconstruction Error. In Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18), Stockholm, Sweden, 13–19 July 2018; Volume 2018, pp. 2298–2304.
48. Choudhary, T.; Mishra, V.; Goswami, A.; Sarangapani, J. A transfer learning with structured filter pruning approach for improved breast cancer classification on point-of-care devices. *Comput. Biol. Med.* **2021**, *134*, 104432. [CrossRef]
49. Chan, T.H.; Jia, K.; Gao, S.; Lu, J.; Zeng, Z.; Ma, Y. PCANet: A simple deep learning baseline for image classification? *IEEE Trans. Image Process.* **2015**, *24*, 5017–5032. [CrossRef] [PubMed]

50. Seuret, M.; Alberti, M.; Liwicki, M.; Ingold, R. PCA-initialized deep neural networks applied to document image analysis. In Proceedings of the 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), Kyoto, Japan, 9–15 November 2017; IEEE: Piscataway, NJ, USA, 2017.
51. Zhong, Z.; Yang, Z.; Deng, B.; Yan, J.; Wu, W.; Shao, J.; Liu, C.L. Blockqnn: Efficient block-wise neural network architecture generation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, *43*, 2314–2328. [CrossRef] [PubMed]
52. Kotthoff, L.; Thornton, C.; Hoos, H.H.; Hutter, F.; Leyton-Brown, K. Auto-WEKA. In *Automated Machine Learning*; Springer: Cham, Switzerland, 2019; pp. 81–95.
53. Feurer, M.; Klein, A.; Eggensperger, K.; Springenberg, J.; Blum, M.; Hutter, F. Auto-sklearn: Efficient and robust automated machine learning. In *Automated Machine Learning*; Springer: Cham, Switzerland, 2019; pp. 113–134.
54. Real, E.; Aggarwal, A.; Huang, Y.; Le, Q.V. Regularized evolution for image classifier architecture search. In Proceedings of the AAAI Conference on Artificial Intelligence, Palo Alto, CA, USA, 22 February–1 March 2022; No. 01. Volume 33.
55. Zoph, B.; Vasudevan, V.; Shlens, J.; Le, Q.V. Learning transferable architectures for scalable image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018.
56. Pham, H.; Guan, M.; Zoph, B.; Le, Q.; Dean, J. Efficient neural architecture search via parameters sharing. In Proceedings of the International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; PMLR 80. pp. 4095–4104.
57. Liu, H.; Simonyan, K.; Yang, Y. Darts: Differentiable architecture search. *arXiv* **2018**, arXiv:1806.09055.
58. Bergstra, J.; Yamins, D.; Cox, D. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In Proceedings of the International Conference on Machine Learning, Atlanta, GA, USA, 17–19 June 2013; PMLR 28. pp. 115–123.
59. Bisong, E. Google AutoML: Cloud vision. In *Building Machine Learning and Deep Learning Models on Google Cloud Platform*; Springer: Berkeley, CA, USA, 2019; pp. 581–598.
60. Bisong, E. *An Overview of Google Cloud Platform Services. Building Machine Learning and Deep Learning Models on Google Cloud Platform*; Springer: Berkeley, CA, USA, 2019; pp. 7–10.
61. Islam, S.M.S.; Hasan, M.M.; Abdullah, S. Deep Learning based Early Detection and Grading of Diabetic Retinopathy Using Retinal Fundus Images. *arXiv* **2018**, arXiv:1812.10595.
62. Li, Y.-H.; Yeh, N.-N.; Chen, S.-J.; Chung, Y.-C. Computer-assisted diagnosis for diabetic retinopathy based on fundus images using deep convolutional neural network. *Mob. Inf. Syst.* **2019**, *2019*, 6142839. [CrossRef]
63. Challa, U.K.; Yellamraju, P.; Bhatt, J.S. A Multi-class Deep All-CNN for Detection of Diabetic Retinopathy Using Retinal Fundus Images. In Proceedings of the International Conference on Pattern Recognition and Machine Intelligence, Tezpur, India, 17–20 December 2019; Springer: Cham, Switzerland, 2019; pp. 191–199.
64. Tymchenko, B.; Marchenko, P.; Spodarets, D. Deep Learning Approach to Diabetic Retinopathy Detection. *arXiv* **2020**, arXiv:2003.02261.
65. Sikder, N.; Chowdhury, M.S.; Arif, A.S.M.; Nahid, A.A. Early Blindness Detection Based on Retinal Images Using Ensemble Learning. In Proceedings of the 2019 22nd International Conference on Computer and Information Technology (ICCIT), Dhaka, Bangladesh, 18–20 December 2019; IEEE: Piscataway, NJ, USA, 2019.
66. Sikder, N.; Masud, M.; Bairagi, A.K.; Arif, A.S.M.; Nahid, A.A.; Alhumyani, H.A. Severity classification of diabetic retinopathy using an ensemble learning algorithm through analyzing retinal images. *Symmetry* **2021**, *13*, 670. [CrossRef]
67. Colas, E.; Besse, A.; Orgogozo, A.; Schmauch, B.; Meric, N.; Besse, E. Deep learning approach for diabetic retinopathy screening. *Acta Ophthalmol.* **2016**, *94*. [CrossRef]
68. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
69. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009; IEEE: Piscataway, NJ, USA, 2009.
70. Zhang, Q.; Couloigner, I. A new and efficient k-medoid algorithm for spatial clustering. In Proceedings of the International Conference on Computational Science and Its Applications, Singapore, 9–12 May 2005; Springer: Berlin/Heidelberg, Germany, 2005; pp. 181–189.
71. Wold, S.; Esbensen, K.; Geladi, P. Principal component analysis. *Chemom. Intell. Lab. Syst.* **1987**, *2*, 37–52. [CrossRef]
72. Likas, A.; Vlassis, N.; Verbeek, J.J. The global k-means clustering algorithm. *Pattern Recognit.* **2003**, *36*, 451–461. [CrossRef]
73. Saeed, F.; Hussain, M.; Aboalsamh, H.A. Method for Fingerprint Classification. U.S. Patent 9530042, 13 June 2016.
74. Tibshirani, R.; Walther, G.; Hastie, T. Estimating the number of clusters in a data set via the gap statistic. *J. R. Stat. Soc. Ser. B (Stat. Methodol.)* **2001**, *63*, 411–423. [CrossRef]
75. Mika, S.; Ratsch, G.; Weston, J.; Scholkopf, B.; Mullers, K.R. Fisher discriminant analysis with kernels. In *Neural Networks for Signal Processing IX: Proceedings of the 1999 IEEE Signal Processing Society Workshop (Cat. No. 98TH8468), Madison, WI, USA, 25 August 1999*; IEEE: Piscataway, NJ, USA, 1999.
76. Al Asbahi, A.A.M.H.; Gang, F.Z.; Iqbal, W.; Abass, Q.; Mohsin, M.; Iram, R. Novel approach of Principal Component Analysis method to assess the national energy performance via Energy Trilemma Index. *Energy Rep.* **2019**, *5*, 704–713. [CrossRef]
77. Cook, A. Global Average Pooling Layers for Object Localization. 2017. Available online: https://alexisbcook.github.io/2017/global-average-pooling-layers-for-object-localization (accessed on 23 November 2022).

78. Akiba, T.; Sano, S.; Yanase, T.; Ohta, T.; Koyama, M. Optuna: A next-generation hyperparameter optimization framework. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, New York, NY, USA, 4–8 August 2019; pp. 2623–2631.
79. Yu, M.; Wang, Y. Intelligent detection and applied research on diabetic retinopathy based on the residual attention network. *Int. J. Imaging Syst. Technol.* **2022**, *32*, 1789–1800. [CrossRef]
80. Chowdhury, A.R.; Chatterjee, T.; Banerjee, S. A Random Forest classifier-based approach in the detection of abnormalities in the retina. *Med. Biol. Eng. Comput.* **2019**, *57*, 193–203. [CrossRef]
81. Zhang, W.; Zhong, J.; Yang, S.; Gao, Z.; Hu, J.; Chen, Y.; Yi, Z. Automated identification and grading system of diabetic retinopathy using deep neural networks. *Knowl.-Based Syst.* **2019**, *175*, 12–25. [CrossRef]
82. Zebin, T.; Rezvy, S. COVID-19 detection and disease progression visualization: Deep learning on chest X-rays for classification and coarse localization. *Appl. Intell.* **2021**, *51*, 1010–1021. [CrossRef]
83. Ben-David, A. Comparison of classification accuracy using Cohen's Weighted Kappa. *Expert Syst. Appl.* **2008**, *34*, 825–832. [CrossRef]
84. Fernández-Fuentes, X.; Mera, D.; Gómez, A.; Vidal-Franco, I. Towards a fast and accurate eit inverse problem solver: A machine learning approach. *Electronics* **2018**, *7*, 422. [CrossRef]
85. Shu, Y.; Wang, W.; Cai, S. Understanding architectures learnt by cell-based neural architecture search. *arXiv* **2019**, arXiv:1909.09569.
86. Selvaraju, R.R.; Cogswell, M.; Das, A.; Vedantam, R.; Parikh, D.; Batra, D. Grad-Cam: Visual explanations from deep networks via gradient-based localization. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 618–626.
87. Shorfuzzaman, M.; Hossain, M.S.; El Saddik, A. An Explainable Deep Learning Ensemble Model for Robust Diagnosis of Diabetic Retinopathy Grading. *ACM Trans. Multimed. Comput. Commun. Appl. (TOMM)* **2021**, *17*, 1–24. [CrossRef]
88. Lahmar, C.; Idri, A. On the value of deep learning for diagnosing diabetic retinopathy. *Health Technol.* **2021**, *12*, 89–105. [CrossRef]
89. Chetoui, M.; Akhloufi, M.A. Explainable end-to-end deep learning for diabetic retinopathy detection across multiple datasets. *J. Med. Imaging* **2020**, *7*, 044503. [CrossRef] [PubMed]
90. Reynolds, C.R.; Fletcher-Janzen, E. *Handbook of Clinical Child Neuropsychology*; Springer: Boston, MA, USA, 2013; ISBN 978-0-387-70708-22.