Mesopotamian journal of Big Data
Vol. **(2025)**, 2025, **pp**. 369–393
DOI: https://doi.org/10.58496/MJBD/2025/023 , ISSN: 2958-6453
https://mesopotamian.press/journals/index.php/BigData

Research Article

Concise Comparison of CNN Models On a Specified Dataset

Humam K. Yaseen[1, *], Saif S. Kareem[2], Bashar I. Hameed[1], Salam K. Abdullah[3]

[1] *Computer Science Department, Al-Imam Al-Adham University College, Baghdad, Iraq*

[2] *Al-Nahrain University, Baghdad, Iraq*

[3] *Department of computer engineering, Al.nukhba University College, Baghdad, Iraq*

ARTICLEINFO

Article History

Received 26 Aug 2025
Revised 19 Sep 2025
Accepted 02 Oct 2025
Published 29 Oct 2025

Keywords

Deep learning
Convolutional neural network
Comparative study
CNN models

ABSTRACT

Recently, interest in Deep Learning (DL), which is a subset of Machine Learning (ML), has emerged. The most famous and used from the DL is the Convolutional Neural Network (CNN). CNN is particularly effective in image processing. There are many duties in image processing that CNN can do, i.e., segmentation, classification, object detection, facial recognition, etc. Image classification is one of the most important applications due to its relevance to various fields, including the healthcare industry and others. One of the challenges researchers face is selecting the appropriate algorithm for the classification task, particularly when dealing with binary or multi-class classification. This paper attempts to compare these algorithms depending on a specific dataset which have four classes, each having a balanced number of medical images. The main thing that the paper focuses on is the power of these algorithms in image classification when placed in the same conditions. This paper also makes a comparison inside the model itself by using three scenarios. The first one involves binary classification, the second uses three classes from the dataset, while the third scenario uses the entire number of classes. The best result among the models is going to AlexNet with an accuracy of 91.92%, and the DenseNet169 with an accuracy of 91.48%. Finally, this paper highlights the differences among state-of-the-art algorithms, particularly in their application to binary and multi-classification tasks.

1. INTRODUCTION

Convolutional Neural Networks (CNNs) have revolutionized the field of computer vision, offering powerful tools for image classification, object detection, and segmentation. These networks are built to mimic the visual processing system of the human brain, enabling them to learn spatial hierarchies of features from images. Convolutional neural networks have been extensively implemented in both research and industrial initiatives as a result of their numerous benefits, including weight sharing, down-sampling, and local connections.

1.1 Background

CNNs are family of deep learning models tailored to process data with a grid-like structure, such as images. Motivated by the structure of the animal visual cortex, CNNs can learn continuously across multiple processing levels of raw input. Several CNN architectures have been introduced over the years, introducing new ideas to address issues like vanishing gradients, computational inefficiency, and inefficient feature reuse. For example, VGGNet proposed a plain and uniform architecture with small convolutional sizes, and ResNet solved the problem of performance degradation in very deep networks by using residual connections [1]. DenseNet then further promoted information propagation and reuse of features by connecting all layers to every other layer in a feedforward manner [2]. More recently, EfficientNet proposed a compound model scaling method that scales up all dimensions of depth/width/resolution evenly, gaining higher accuracy and efficiency [3].

1.2 *Motivation*

Introduction some of the significant break-throughs in computer vision and deep learning have been achieved by CNNs. Due to the fast development of thousands of CNN architectures, analyzing and comparing those is important for many reasons:

*Corresponding author. Email: humam.khalid@imamaladham.edu.iq, humamk84@gmail.com

- Performance optimization: Various CNN architectures have different trade-offs for the accuracy, speed, and resource utilization. In this paper, a comparison results were made among these algorithms to determine the best models.
- Understanding strength and weakness: Every CNN architecture has its powerfull in fields and weaknesses in another fields. E.g., some of them could be good for high resolution images, while others could be good on small datasets. So, it is useful to compare algorithms to show the differences because this allows researchers to select the best model for their problem.
- Guiding future research: Comparisons further shows the effects of architectural designs and techniques in CNN-based simulations. These insights, if well founded can be used to inspire future research, to improve existing models, or to design a new architecture.
- Application specific needs: CNNs may also need different requirements in different applications, such as real-time processing for video data or high accuracy for medical imaging. Comparing CNN algorithms lets researchers select models that are better adapted to their application, resulting in better performance.
- Community collaboration: Publication of comparative results promotes research community collaboration. Releasing research on CNN performance allows researchers to stand on each other's shoulders, ultimately driving increased lack of innovation and better model performance.

1.3 *Challenges*

In order to make a meaningful comparison among CNN models, several factors must be considered by the researcher, including choosing an appropriate dataset, training conditions, evaluation metrics, and so on. Many challenges face the comparisons of the CNN model, down there some of them:

- Dataset variability: Many datasets can be used for comparisons, each of which differs from others in size, number of classes, balance or not, and so on. These differently makes discrepancies in results.
- Training conditions: the change of hyperparameter of the models, such as learning rate, batch size, may change the results.
- Evaluation metrics: many of evaluation metrics may be used in the comparisons which may confuse the researcher.
- Computational resources: running the models on different machines may lead to different results even for the same model under the same conditions.

1.4 *Contributions*

This paper addressing some contributions that may guide researchers in their papers.

- Architectural Insights: Providing an insight into the model's architecture, analysis them, and make a comparison including key features, use cases, parameter count, and others.
- Solo model comparative: This paper making a comparative for each model with three scenarios.
- Models' comparative: comparing using the evaluation metric among the model used.
- Metrics: suggested the benchmark evaluation metrics can used for the comparison studies.

1.5 *Objectives*

This paper aims to create a comprehensive understanding of CNN model performance, facilitate informed decisions in model selection, and ultimately contribute to advancements in deep learning applications. The following are some of the objectives of this research:

- Advancement in DL research: where the comparative study can guide future research and development in the field of DL.
- Improved model selection: providing a clear understanding of model performance can guide researchers to make a good decision when selecting a model.
- Contribution to industrial innovation: Optimizing CNN models for specific applications can drive innovations in various fields, such as healthcare, autonomous driving, manufacturing, and security.

1.6 *Significances*

The comparison of CNN models may exceed academic interest to other fields. This research can ultimately contribute to the evolution of Artificial Intelligence (AI) technologies and their integration into everyday applications.

The remainder of the paper is organized as follows: Section 2 presents related work, Section 3 describes the methodology and explains the CNN methodology, and also explains the architecture of the models used in this paper, while Section 4 discusses the results of models using the three scenarios. Finally, Section 5 concludes the paper.

2. RELATED WORKS

The DenseNet model was used for liver lesions prediction in [4]. The authors used this model to mark a significant advancement in CNN architectures in medical image. The paper attempts to mitigate vanishing gradient issues, enhancing feature extraction and representation learning critical to reach an accurate diagnosis. The strength of the work comes from the effective features extracted by the DensNet model, which is reflected in the results showing promising outcomes. While the model was tested on a large liver scan dataset and showed a promising results, the use of another dataset remains a concern. The authors also did not use another state-of-the-art models to compare the performance of DenseNet with others. In this paper, there is no analysis about the dataset that highlights its limitations, which could affect the performance of the model and then influence applicability in real-world frameworks. The authors in [5]use VGG16, ResNet, and AlexNet models to enhance white blood cell classification. The authors also develop a novel meta-heuristic optimization algorithm by leveraging the unique strengths of each CNN in feature extraction and representation learning. The study reports significant improvements in cell type classification. However, the paper has some limitations, among them is that it doesn't thoroughly evaluate how each individual architecture contributes to the overall performance, that provides a clearer understanding of their impacts. Additionally, the effectiveness of the proposed optimization algorithm isn't compared to existing methods, leaving questions about its relative advantages. The authors also did not discuss the risks of overfitting or high computational demands, which are crucial in clinical applications. In general, the study yields good results and highlights the promise of combining advanced CNN architectures for medical diagnosis in hematology; however, a deeper exploration of its methodologies and limitations would strengthen its relevance and applicability. The authors in [6] present a fusion approach utilizing Inception module to detect pulmonary nodules. The method achieved an overall accuracy of 92.5%, which demonstrate high effectiveness in identifying both malignant and benign nodules. The model detects malignant cases by showing sensitivity of 91%, and a specificity of 90%. Additionally, the model yielded an F1 score of 89%, highlighting a good balance between precision and recall. Also, the model can distinguish between positive and negative instances by result 94% of the Area Under the Receiver Operating Characteristic Curve (AUC-ROC). Although the authors show a good result in their paper but still there some limitations such as the unbalanced dataset used which may not show the diversity of pulmonary nodules, the complexity of the Inception module takes long training times and increased computational resource requirements, which could hinder practical implementation in real-time clinical settings. The paper [7] introduces an automatic diagnosis system of COVID-19 based on the EfficientNet Convolutional Neural Network to extract the efficient feature properties. It demonstrates a high accuracy on diagnosing chest X-ray images; thus, its suitability for rapid and accurate clinical use has been presented. The authors in their paper show good results with 95% accuracy, around 93% sensitivity, indicating its effectiveness in detecting true positive cases, and a specificity of about 96%, showcasing its ability to accurately identify non-COVID-19 cases. The study is generally good, but it has several limitations among them is that it might have used a small and relatively homogeneous dataset, which may limit the model's generalization to different populations and imaging protocols. The EfficientNet may produce overfitting, especially with the limited dataset. In addition, there has been no complete comparison to other state-of-the-art methods, and so it is difficult to tell the pros and cons of this EfficientNet method. The survey introduced by A. Kamilaris et al. [8] studies the use of CNN in agriculture. The strength of the study comes from the focusing on the success of the CNN in manipulating large amounts of complex data in this field, such as sensor data and images, which significantly enhances the agriculture field such as monitoring crop, detection of disease and forecasting the yield. The study also makes comparisons among several research works in the field to show the performance of the CNN architectures and methodology in agriculture, and highlights how CNNs improve the field of precision agriculture with better image analysis and data analysis. By comparing sensor types with CNN performers, the study shows the AI performance to enhance activities in farming markedly. The study also marks some challenges in using CNN with the agriculture field, one of these is the poor quality and quantity of labelled data, which can decrease the efficiency of the model. Also, the computational resources may be expensive, especially for farmers who may not have the technology. Investigating the industrial food packaging hyperspectral classification by a CNN was suggested in [9]. The primary strength of the paper is to show how we can defect detection in packaging materials using CNN on hyperspectral information, and also show how it increased accuracy. The paper focused on ability of CNNs to effectively handle the high dimensionality of hyperspectral data which help in identification of food packaging materials and contaminants and thus to improve product safety and compliance with regulatory standards. The paper discusses that obtaining dataset could be time consuming and costly, so it is one of the major challenges in industrial. Over fitting also may

reduce the capability of generalization especially if the model is trained on a limited variety of packaging materials. However, the paper also has notable limitations. One potential issue is the computational intensity of CNNs, which may require significant processing power and resources, making it less accessible for smaller operations. The authors in [10] provides an extensive review of using CNN with transfer learning in the medical imaging. They spotlight the benefits of the approach from better disease discovery, support for pre-trained models to reduce training time and data needs. The strength of the paper by discussion the architecture development of the CNN models that increase the efficiency of the model with each development. Many challenges are discussed, such as a lack of labeled medical data, the complexity of interpretable models, and the necessity of strong validation methods, the training of some models may require computational resources, which may not available for all organizations. The research also addressed the benefits of using the transfer learning strategy with the CNN model especially in medical diagnostics and patient care which gives a strong classification rate which is important in this field. Although this study demonstrates notable strengths, it may not adequately address the interpretability of the models. Interpretability is essential for building trust among healthcare professionals and ensuring clinical applicability. The survey in [11] offers a comprehensive on several CNN models by exploring the architecture design, key enhancements, and also the use of the models in many domains presenting the advantages of CNNs on feature representations and generalization to other tasks, and show the challenges, such as the computational cost and the interpretability of models, as well as the future directions of research development. In other words, it highlights the huge role played by CNN models in technology advancement and the room for further innovation in various fields. The limitation of this paper is focusing on theoretical frameworks and algorithms rather than providing real-world case studies or practical implementations. CNNs and their implementation in radiology are reviewed in [12] by presenting the architecture and working of CNNs, and underscore their potential analyzing and interpreting medical images efficiently. The other strength of this work is its discussion of various applications, which showcases the versatility of CNNs in addressing different challenges in radiology. One of the healthcare system challenges is the availability of data discussed in this study which is a difficult issue due to privacy concerns and the need for expert labeling. Also, the resource needed for models may be high which be posing a barrier for some healthcare institutions. While the paper makes a good discussion about CNN application in radiology medical images, the paper might not sufficiently address the challenges of interpretability and trust in CNN predictions, which are crucial for clinical adoption, which is considered a limitation of the work. In the final, the review emphasizes the prominent role CNNs can have to transform radiological workflow and patient care. M. A. Saleem et al. [13] makes a comparison on popular models like ResNet, Inception, and EfficientNet and analyses their trade-offs in terms of accuracy, computational, and training efficiency. The findings reveal the trend in CNNs including deeper structures of the network and new layers performing feature extraction. The strength of this paper is the improvement in feature extraction capabilities and training efficiency, for the ResNet as an example the improvement is employs skip connections to facilitate deeper network, while DenseNet cconnects every layer to every other layer, improving information flow and reducing redundancy. The dependence on a specific model is considered a weakness of the framework because it may overlook other significant models that could offer valuable insights or alternatives. Overall, this paper discusses the state-of-the-art CNN models, which give an insight into the strengths and weaknesses of these models and the need to continue for research to enhance them and overcome the limitations.

3. MATERIALS AND METHODS

This section discusses the convolutional neural networks, including the primary contents and also the state-of-the-art models that play a crucial rules in image classification.

3.1 Convolutional Neural Networks

CNN has shown remarkable success in many computer vision and machine learning applications. On this topic, there have been many great articles, and several great open source packages for implementing CNNs. CNN applies to many tasks, including image based ones. CNN applies to object recognition in images, image classification, and image semantic segmentation, etc. In this paper we would focus on the domain of image classification, otherwise referred to as categorization. In every image in the problem of image classification, there is a dominant object that occupies a significant portion of the image. The object class that a picture belongs to (dog, airplane, bird, and so on) is the class that the particular image represents[14, 15].

3.1.1 CNN architecture

Convolutional Neural Networks (CNN) are used to process input images in the form of 3D tensors through several layers: convolution, pool, I/O layer, and full connected layers. Each layer takes the input and passes it on to the next, until we get the last output. If we are training for classification, the output will be a vector of class activations, computed using a softmax. The last layer has a loss function such as cross-entropy, but it directly computes and optimizes the difference between predictions and ground truth which is made possible due to a simpler backpropagation path.

a. Convolution layer

The convolution layer is a fundamental building block in CNN, which is used to perform the convolution operation on the input data (e.g. images)[16]. Convolutional layers play a crucial role in feature extraction in the CNNs, which lets the

network to capture the spatial hierarchies in images. Using a sequence of filters in combination with various operations such as stride and padding, these layers are capable to transform input data into useful representations for further processing[17]. Filters, or kernels, are tiny matrices (for example 3×3 or 5×5) that learn to recognize various patterns such as edges, textures, or shapes[3]. The filter is moved over the input image and for each position, the element-wise multiplication is taken and the results are all summed together (as shown in Figure 1), resulting in a single value. This operation results in a feature map, which highlights feature that filter could detect[18]. The depth of filter is depend on the depth of the image, for example, for color images, filters are 3-D with a depth of 3 (one for each RGB input channels)[19].

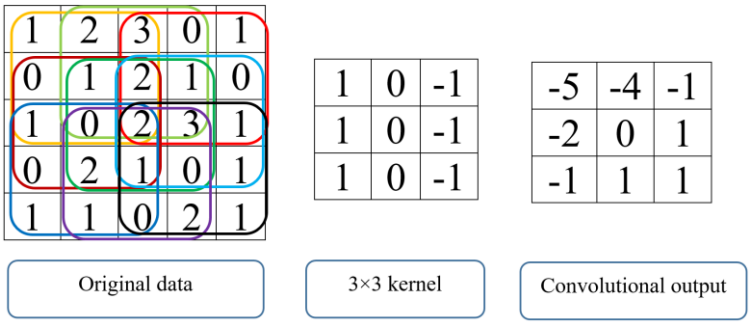

Figure 1: Convolutional layer

b. Pooling layer

The feature map of convolutional layer cannot preserve the position information of the feature very well. Thus, even small perturbations e.g. cropping or rotation will produce completely different feature maps. To deal with this problem, down-sampling was used in convolution layers. This is achieved by placing a pooling layer after the nonlinearity layer. This pooling is an operation that helps to achieve invariance to small translations of the input. A transitional invariance implies that small shifts in the input will not warp the value distributions of most of the pooled outputs[20]. Pooling layer has different types for different purposes. The most popular one is max pooling, where maximum value of a local region of feature map is chosen, which effectively preserves prominent features and reduces the dimension, as shown in Figure 2. Another type is average pooling, which computes the average of the elements of a region, thus smoothing features as shown in Figure 3[21].

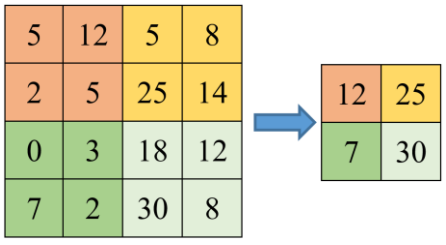

Figure 2: Max pooling

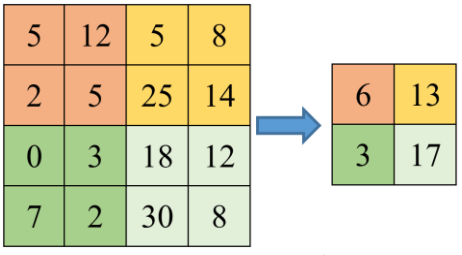

Figure 3: Average pooling

c. Activation function

Activation functions are critical pieces in neural network that bring non-linearity to the model and can help make the network learn more complex patterns in the data. The most frequently used activation function is the ReLU (Rectified Linear Unit) defined in (1) [22]

$$f(x) = \max(0, x) \qquad (1)$$

This function is also fast to compute and can help alleviate the vanishing gradient problem, but it can cause the "dying ReLU" problem, where neurons die and become inactive[23]. Another common option is the sigmoid function showed in (2) mapping input values to the range [0, 1], for binary classification tasks, but it tends to vanish gradients for big inputs[24].

$$f(x) = \frac{1}{1+e^{-x}} \qquad (2)$$

The tanh function illustrated in (3) returns values on rang −1 to 1, which usually leads to better convergence than the sigmoid, however, the issue of gradient dying out is similar to the sigmoid[25].

$$f(x) = \tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \qquad (3)$$

For multi-class classification, the softmax function illustrated in (4) is widely used, which normalizes scores into a probability distribution summing to one, though it is prone to outliers[26]. Lastly, the category of leaky ReLU addresses the dying ReLU problem by assigning positive gradients to negative inputs, but more complex structure is added to the network by having to learn the negative slope[23].

$$f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \qquad (4)$$

Every different activation function has its pros and cons, and proper activation function is vital to rail the performance of neural networks.

d. Fully connected layer

Fully connected (FC) layer, is an important component in neural networks in which every neuron in the layer is fully connected to every neuron in its preceding layer[27]. This rich connectivity in turn lets the network process information in a globally collective manner, with each neuron taking input from all neurons in the preceding layer. There is a weight between each connection, and a bias within each neuron, and both of these are trained to minimize the loss function[28]. The weighted sum input for each neuron is calculated, and a non-linear activation function is applied to allow the network to learn complex patterns. Fully connected layers also work as both: to aggregate the features learnt by previous layers, as well as to make the final predictions for the output. In classification models, the output fully connected layer typically has neurons equal to the classes, and a softmax activation function to output probabilities for the classes. In general, the fully connected layers connect the input to the output of a network, enabling to make predictions based on input data[27, 28].

e. Input layer

The input layer is the initial layer of a neural network, which takes and organizes data into a network format. Its number of neurons equals that of the input features (e.g., for a 28×28 pixel grayscale image, it has 784 neurons). This layer does not compute anything on its own, it just forwards the well-padded data to the next layers in a way that allows the convolutional architecture to grasp certain patterns and correlations between the concepts[29].

f. Output layer

The output layer is the last layer in a neural network that generates the predictions of the model after processing the input data. Its architecture and activation function is customized to the problem at hand. For classification problems, the output layer usually contains as many neurons as there are classes and applies a softmax activation function to return raw scores to probabilities. And finally, the output layer outputs the net's computed results, which the network can then use to make decisions about the input data[30].

3.1.2 Forward run

After training, the CNN model can be utilized for prediction with a forward pass through the network. For an image classification task, the input image x1 is mapped through each layer in turn to produce a set of intermediate outputs up to the estimate xL for the posteriors of the C categories[31]. In calculating the ultimate prediction, the category with the largest predicted probability will be chosen. It's worth to mention that there is no need of the loss layer in prediction, since it is just for parameters learning during training[32].

3.1.3 Stochastic Gradient Descent

The loss function is minimized using Stochastic Gradient Descent (SGD) algorithm to optimize the CNN parameters[33]. In training, a forward pass takes the input training example, does the prediction and compares to the target to calculate the loss. We update the parameters based on the gradient of loss to each of those parameters using the equation in (5):

$$w^i = w^i - \eta \frac{\partial z}{\partial w^i} \qquad (5)$$

Where η is the learning rate. Instead of updating parameters on all training examples at once, SGD updates parameters on mini-batches to achieve a balance between efficiency and stability, with input as 4D tensor with mini-batches. This is since it serves to smooth out disturbances to the loss function and is the common way to train CNNs[33].

3.1.4 Back propagation

Partial derivatives with respect to the last layer in CNN are the easiest to calculate since XL directly modifies the loss z by means of parameters WL. Two gradients are computed for each layer: $\frac{\partial z}{\partial WL}$ for parameter updates, and $\frac{\partial z}{\partial XL}$ for error backpropagation[34]. This backpropagation, which utilises the chain rule, facilitates more effective propagation of errors from the output layer through to the layers prior to it. Using the computed gradients $\frac{\partial z}{\partial X_{i+1}}$ can be produced and the updates of Wi and Xi can be expressed in terms of these gradients that are easier to compute given the connections defined by the parameters of the layer[32].

3.2 Overview

The heterogeneity among CNN architectures demonstrate the on-going nature of visual data processing challenges, and the race for more efficient, more accurate models. Every kind of CNN brings its particular characteristics and innovations designed to solve particular problems, deal with computational constraints or improve to performances. This article will review different types of CNN models describing their architectures and details.

3.2.1 LeNet-5

LeNet-5 is an early convolutional neural network (CNN), introduced by Yann LeCun in the late 80s and early 90s [35]. LeNet-5 was mainly created for handwritten digit data such as MNIST. The architecture has multiple layers: it begins with an input layer with 32x32 pixels, then two convolutional layers are used to extract features with filters, followed by two pooling layers which reduce the dimensionality of the data, while still preserving important information. LeNet-5, for instance, contains six filters at the first convolutional layer and sixteen at the second convolutional layer, which are then passed through fully connected layers that give you predictions for digits 0 to 9 [35]. LeNet-5 not only successfully utilizes backpropagation for training, introduces pooling layers to deal with growing complexity, but also activations, described with Sigmoid or more recently Tanh for the more recent incarnations; ReLU would be used today. Figure 4 shows the architecture of LeNet-5. LeNet model has served as the basis of newer CNN models, and remains a core part of the deep learning, and computer vision, community, especially for tasks such as image classification, handwriting recognition, object detection and document analysis. The model is considered the groundwork for the modern CNN model and is characterized by being easy to learn and understand, making it suitable for teaching CNN to beginners. Also effectiveness for simple classification tasks is another advantage of LeNet, which demonstrates the validation of CNN in feature extraction. Additionally, LeNet requires the lowest computational resources compared to other models, making it suitable for organizations or individuals with limited resources. In the other hand, there are many limitations to using LeNet. For example but not limited to, LeNet is not deep enough to capture complex patterns, also the model suffers from overfitting espicially with large datasets, and finally, the model lacks the techniques like dropout or skip connection[36].

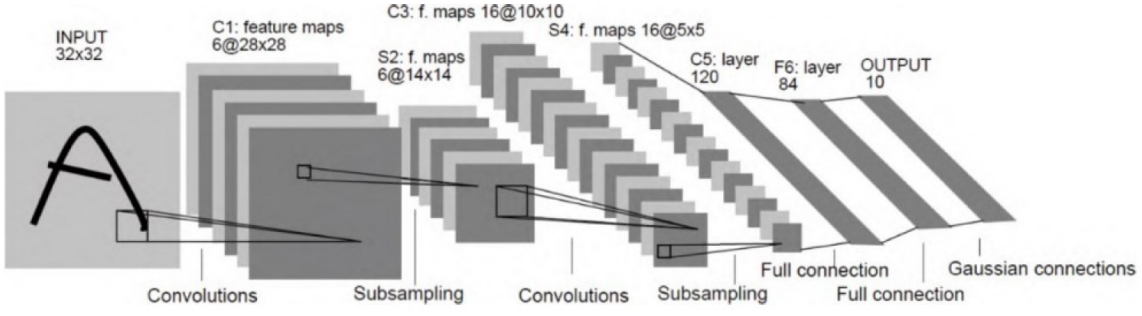

Figure 4: LeNet-5 architecture [37]

3.2.2 AlexNet

AlexNet is a pioneering convolutional neural network architecture that significantly revolutionized deep learning, particularly in the field of image classification. Created by Alex Krizhevsky and his colleagues in 2012, it was the winning architecture of the ImageNet Large Scale Visual Recognition Challenge that year [38], showcasing the potential of deep learning in processing intricate visual data. AlexNet is composed of eight layers with learnable parameters, including five convolutional layers followed by three fully connected layers, as shown in Figure 5. It utilizes large filters in the initial layers to identify low-level characteristics and smaller filters in the later layers to distinguish more intricate patterns[39]. One of

AlexNet's significant contributions is the use of the ReLU activation function, which eliminates the vanishing gradient problem and expedites training compared to previous activation functions such as Sigmoid and Tanh[40]. AlexNet also uses dropout regularization to prevent overfitting, as well as data augmentation to enhance the model's robustness. AlexNet's influence is enormous and extensive. Although it only directly contributed to one area, it encouraged a vast amount of research in CNN architecture from which various inferences can be drawn[39]. One of the most significant merits of AlexNet is its capability to work on large datasets. It is also very important in modern CNN architecture. Despite its advancements, AlexNet has its drawbacks. Its architecture is still considered to be shallow. The heavy parameter also makes it a lot less efficient than the new models[41].

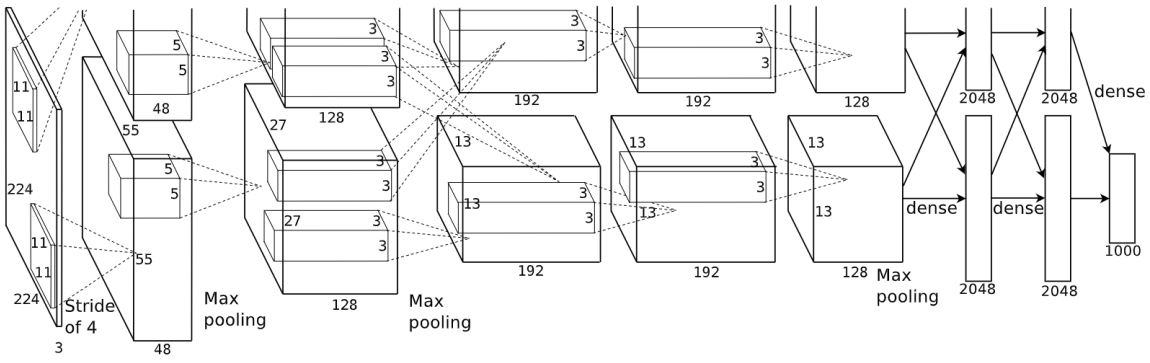

Figure 5: AlexNet architecture [39]

3.2.3 VGGNet

VGGNet belongs to a class of convolutional neural network (CNN) architectures, is well known for its simplicity and is effective in the task of image classification. VGGNet developed by the Visual Geometry Group at the University of Oxford was a finalist in the 2014 ImageNet [38] Large Scale Visual Recognition Challenge (ILSVRC) and came to the public attention due to its second place performance in the classification task. VGGNet is distinctive due to its very small (3×3) convolution filters, and its significant depth (e.g., 16-19 weight layers)[42]. VGG16 possesses about 138 million parameters while VGG19 has about 143 million parameters. This architecture helps the network to learn more complex features while maintaining a manageable number of parameters. Small filters are used, so it can capture fine details of the image and spatial hierarchies, making it's accuracy high. In order to decrease dimensionality of the image while keeping the significant transitions, VGGNet uses multiple layers of the convolution and max pooling [43], also three fully connected layers are added to the network for calculating the final classification results. For the purpose of accelerating the network convergence and improving the model generalization, VGGNet employs the ReLU as the activation function. VGGNet use tiny 3×3 convolutions and is able to learn very good image representations for a classification task. The network starts with an input layer with 224×224 RGB images as input. It is comprised of 5 convolutional blocks and then max-pooling layers to downsample spatial dimensions while preserving essential primitives[42]. The starting block has 64 filter, the second block has 128 filter and the next two blocks have 256 and 512 filters, while the last two blocks have three sets of Conv layers[42]. Following the convolutional layers, the VGGNet contains three fully connected layers, the first two having 4096 neurons and the output layer with 1000 neurons to classify 1000 classes. The architecture uses the Rectified Linear Unit (ReLU) activation function in the blocks, and dropout on the fully connected layers for generalization. The main advantage of VGGNet lies in its transfer learning property, consequently also being employed in various applications rather than image classification as well, such as feature extraction and fine-tuning to particular applications[44]. Despite being computationally expensive by modern standards, its simplicity and rapid performance, served as a solid baseline and an inspiration for many of the subsequent CNN architectures. Also, VGGNet's use of small convolutional filters as resulting depth has become a trend among the following networks, demonstrating the fact that depth of the networks is important for the overall performance. VGGNet is known for its usage of stacked 3×3 convolution filters for its simplicity and uniform design for depth. Such an arrangement is beneficial for more accurate feature extraction. Its simplicity and performance on image classification tasks are notable. However, VGGNet have some weaknesses points, which include the excessive number of parameters that translates to high memory usage and disproportionate computational cost, make the model ineffective in low-resource spaces[45].

3.2.4 ResNet

ResNet or Residual Network, is an architecture tailored to fathom the problem of training very deep neural networks. ResNet uses a form of residual learning introduced in "Deep Residual Learning for Image Recognition [46]", which enables networks to learn residual mappings rather than unreferenced mappings. This is done by means of skip connections that skip one or more layers and thus prevent the vanishing gradient problem, hence allowing the training of networks with hundreds or thousands of layers as shown in Figure 6. Stacked residual blocks architecture, each block containing two or three

convolutional layers, batch-normalization, and ReLU activation[47]. ResNet, which has shown state-of-the-art performance on a variety of image recognition tasks, pushing the error rates substantially lower in both standard benchmark data sets such as ImageNet [38]. There are many versions of ResNet e.g. ResNet18, ResNet34, ResNet50, ResNet101, and ResNet152 depending on the depth of the network [1], this variations allow flexibility in choosing depending on the need of balancing performance and resource constraints[46]. ResNet's scalability and ability to generalize well across various tasks are significant strengths. Nevertheless, ResNet's identity shortcuts may limit the model's representation, while deeper versions of the model tend to be more expensive computationally[48].

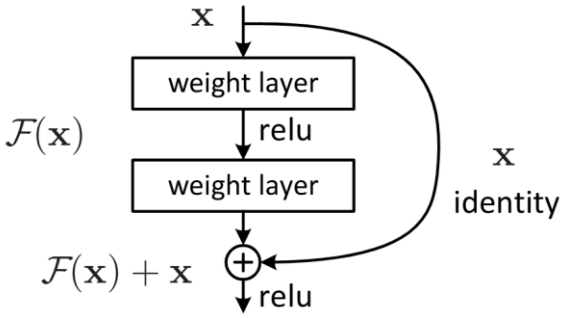

Figure 6: Residual block

3.2.5 Inception

Inception is a deep learning neural network architecture that is prepared for overcoming some issues of image classification by reaching a better local minimum. It is based on Inception modules, i.e., it uses multiple convolutional filter sizes (1×1, 3×3, and 5×5) in parallel inside one layer [49]. This method enables the network to learn multiple types of features simultaneously. An important concept is factorization where large convolutions are factorized into smaller ones, for example using two 3×3 convolutions sequentially as opposed to a single 5×5 convolution [49]. This decreases the number of parameters and computational cost and retain the expressiveness of the model. Also, Inception model includes 1×1 convolutions with purpose to reduce dimensionality, thereby reducing the computational burden before applying more complex convolutions. The architecture encourages a regular depth and width of layers, a good performance can be achieved on benchmark datasets, such as ImageNet [38, 50]. In general, the Inception model has been praised for its good tradeoff between efficiency and accuracy in many computer vision tasks [49, 51]. The Inception model has a different number of parameters across versions. Inception-v1 considers nearly 5 million parameters, meanwhile Inception-v2 lowers this to close to 4.5 million, using factorization. As the subsequent version, Inception-v3 refines the architecture slightly, with about 23 million parameters. These trade-offs reflect the differences in the costs of model complexity, computation and performance at different levels of the resource-requirement spectrum for different tasks, allowing a user to select an appropriate version of the model according to their unique requirements. Inception is modular and flexible, hence it's suitable to achieve high performance at a low cost. This comes with a catch though. Its complex architecture can be difficult to implement and fine tune especially for beginners who are not versed with the design principles[49].

3.2.6 EfficientNet

EfficientNet is developed based on a principled network scaling method that uniformly scales all dimensions of depth/width/resolution. The base network of the architecture, called EfficientNet-B0 uses depthwise separable convolutions. This approach decreases the amount of parameters and computations by decoupling the convolution operation into two layers: depthwise convolution and pointwise convolution[52]. This model use Swish activation function which outperform better with this model than the conventional ReLU activations[53]. EfficientNet use scaling, where all dimensions of the network including depth, width, and resolution, are increased together. The function of scaling is make a trade-off between model complexity and efficiency[52]. EfficientNet comes in a different sizes, beginning with EfficientNet-B0, which has around 5.3 million parameters, and goes all the way up to EfficientNet-B7, which has about 66 million parameters, the performance is outperform a little bit better at each version[54]. The architecture is arranged in a stem layer to process the input image, and multiple blocks of depthwise separable convolutions. All of the blocks already incorporate batch normalization and activation functions which also help accelerating the computation. And at the end of the model global average pooling and a fully connected layer for classification purpose has been added[52]. Its architectural and scaling strategy enable it to be adopted in a variety of computer vision tasks where it strikes a satisfying trade-off among both accuracy and efficiency[52]. The model efficiency and ability to generalize well on a very broad range of datasets are its strengths. Nevertheless, its application of neural architecture search for design optimization could result in extremely high up-front computational costs that may be too expensive for some users[52].

3.2.7 NASNet

Neural Architecture Search Network (NASNet) is a convolutional neural network architecture model that was developed with neural architecture search methods. NASNet is designed to search for the optimal architecture which can perform efficiently and accurately, using search algorithm[55]. There are two major versions: NASNet-Large and NASNet-Mobile. NASNet-Large is meant to have high accuracy and is designed to be used in applications where there is a lot of computational resources while it have over 88M parameters[56]. In contrast, NASNet-Mobile [57] dedicates for mobile and edge devices by seeking to minimize the model size and computation cost while retaining competitive performance with approximately 23M parameters. The architecture is based on cells which are the basic building block of NASNet stacked on top of each other, where normal cells compute features while reduction cells perform feature map downsampling. A cell is manually designed to learn patterns which can be represented by various operations such as convolutions, pooling, or batch normalization [55]. One of the most innovations of NASNet is that it employs reinforcement learning for searching the optimal architecture. That is, it's the process of training a controller LSTM network how to produce architectures, then testing those architectures' performance on some task. The best networks are employed to retrain the controller, making it learn and become optimized [55]. NASNet achieves state-of-the-art performance on multiple image classification datasets validating the effectiveness of the architecture optimization. NASNet-Large is the highest performing one in terms of accuracy and NASNet-Mobile is the low cost alternative for real-time applications on resource-constrained devices [57]. The greatest advantage of NASNet is its automatic design process, eliminating the need for tuning the architecture manually. The drawback of this is that the computation of resources consumed during search is extremely high, and hence it is less adapted to smaller organizations or researchers with limited resources[58].

3.2.8 Xception

Xception is a convolutional neural network architecture that's similar to the Inception model, but is based on depthwise separable convolutions. Xception is a model architecture developed for image classification by François Chollet in 2017 with the goal of increasing model efficiency [59]. The architecture of the Xception model involves depthwise separable convolution layers in place of traditional convolution layers. The depthwise convolutional operator use a single learned filter for each input channel, followed by a 1×1 convolutional filter to combine the filtered maps. The benefit of this strategy is the reduction of parameters and computational cost, and also making accuracy distinct or even further improved than the other network [59]. The architecture of Xception consists of 14 convolutional layers organized into 36 depthwise separable convolution blocks, and then a last global average pooling layer, and finally with a fully connected layer for the classification process. The model contain about 22 million parameters [59, 60], and can be efficiently deployed across different applications, one of the many good things Xception has going for it is it's capability of capturing very complex patterns in an efficient way. The model shown good results on image classification benchmark ImageNet dataset, outperforming many previous models with regard to the accuracy [59, 60]. Xception is a substantial improvement over previous deep learning architectures, and it gives an effective and efficient framework for image classification applications. Its creative depthwise separable convolutions usage also makes it appealing to researchers in computer vision [61]. The model strengths include the efficient use of model parameters and its ability to support large-scale datasets very efficiently. Overfitting, particularly when used with smaller datasets, can occur in Xception[62].

3.2.9 DenseNet

DenseNet is a convolutional neural network architecture proposed by Gao Huang et al. [63] in 2017. The main novelty of DenseNet is its dense connectivity which allows every layer to receive from all preceding layers, thereby encouraging feature reuse and better gradient propagation through the entire network [63]. DenseNet is composed of a series of dense blocks, where each layer in a block is connected to all subsequent layers. Inside each block, the output of each layer is appended, and constitutes, to the input of the following layer. Such architecture differs from the typical network in which layers are sequentially connected. Such densely connected layers help to optimize parameter by reusing shared information identified at lower layers in the network [64]. A transition layer is added after each dense block to reduce spatial size and number of feature maps. This dense blocks with transition layers combination allows a model with a manageable complexity and computational cost while delivering high performance. DenseNet networks differ in depth, such as DenseNet-121, DenseNet-169, and DenseNet-201, where the number refer to the number of layers [63, 65]. DenseNet is known to be efficient and competitive in accuracy on ImageNet compared to traditional network structures with non-negligible less parameters used. The efficiency is mainly achieved by the dense connectivity, which suppresses the over-fitting and promotes the reuse of the feature [63, 66]. The architecture has achieved superior performance on a number of computer vision tasks, such as image classification, object detection, and semantic segmentation. Due to such potential, DenseNet is widely adopted in academia and industry [64, 67] and it can be more accurate with fewer parameters. In sum, DenseNet is a milestone in architecture design for deep learning, which demonstrates the effectiveness of dense connections for feature learning and efficiency [65, 66]. The Dense connectivity encourages feature reuse and reduces the number of parameters. This architecture has better accuracy and reduces the network to train. Another advantage of DenseNet is its efficiency and robust performance in image classification tasks. Its dense connectivity is, on the other hand, greedy in terms of large GPU memory and training time, which can be limiting in resource-constrained environments[62].

All of the previous CNN models have several common characteristics, as shown in Table . They all use convolutional layers to process input images, which makes them good at learning spatial hierarchies [67]. The majority of architectures make use of the ReLU activation function, which is designed to alleviate the problem of vanishing gradients, while other variations incorporate alternatives such as Sigmoid or Swish [68]. Pooling layers like max pooling are often added to the neural network architecture to perform down sampling on the feature maps and decrease the spatial dimensions and computational cost [69]. A large number of networks also have regularization methods such as dropout and batch normalization to avoid overfitting and improve the generalization [70]. All of these models are mainly designed for image classification and achieved good results on image classification benchmarks such as ImageNet and MNIST [71]. Transfer learning, which enables a model developed on one task to be fine-tuned on another and is effective in the event that limited data is available, is also a feature used in many state-of-the-art architectures [42]. Moreover they can be implemented with commonly-used deep learning libraries such as TensorFlow, Keras, and PyTorch making it accessible for adoption and experimentation [72, 73]. Popular optimizers such as Adam, SGD and RMSProp are used among different architectures in order to update the weights whilst training the model, demonstrating the core principles that exploit the benefits of CNNs in feature extraction and generalized pattern recognition [74].

Table I: CNN models comparison

Model	Year	Depth	Key features	Parameter count	Input size	Activation function	optimizer	Transfer learning support	use cases
LetNet-5	1998	7	Fisrt CNN, uses conv and subsampling layer	60K	32×32	Sigmoid	SGD	✗	Handwritten digit recognition
AlexNet	2012	8	introducing ReLU, dropout, and data augmentation	60M	227×227	ReLU	SGD and Momentum	✓	image classification
VGG	2014	16-19	uses small filters, deep architecture	138M	224×224	ReLU	Adam	✓	image classification, transfer learning
Inception	2014	22	inception modules for multiscale feature extraction	7M	224×224	ReLU	RMSProp	✓	Image classification, object detection
ResNet	2015	18-34-50-101-151	uses residual connections to combat vanishing gradients	25-60M	224×224	ReLU	Adam	✓	Image classification, object detection
Xception	2017	71	Depthwise separable conv for efficiency	22M	299×299	ReLU	Adam	✓	Image classification, transfer learning
DenseNet	2017	121-169-201	Dense connectivity pattern for feature reuse	8-30M	224×224	ReLU	Adam	✓	Image classification, segmentation
NASNet	2018	403	Neural architecture search for optimized architecture	88M	224×224	ReLU	Adam	✓	Image classification, transfer learning
EfficientNet	2019	7	compound scaling method for optimizing accuracy and efficiency	5-66M	224×224	Swish	Adam	✓	Image classification, mobile application

3.3 Experimental environment

The models were implemented using Python 3.12.7 and TensorFlow 2.19.0. Each model was trained for 10 epochs. ADAM is the optimizer used with a learning rate of 0.0001. The models were implemented on a HP Zbook workstation with Windows 10 64-bit, CPU: i7-6820HQ, RAM: 32GB DDR5, and GPU: 8 GB. There are no preprocessing procedures applied to the images, except for adjusting the image dimensions to suit the models.

3.4 Performance metrics

In order to evaluate the models and demonstrate the effectiveness of each model, state-of-the-art performance measures are used. In order to work with these metrics, each model has its own Confusion Matrix (CM), which can be used to evaluate the model's performance. Four metrics found in CM, True Positive (TP) which indicate the positive instances that were correctly classified, True Negative (TN) which indicate the negative instances that were correctly classified, False Positive (FP) which indicate the positive instances that were incorrectly classified, and Flase Negative (FN) which indicate the negative instances that were incorrectly classified[75]. Table makes a description of the performance measures with their formula.

Table II: Performance metrics description

Metric	Description	Formula
Accuracy	Ratio of the corrected classified instances to the total instances	$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$
Precision	Estimate the accuracy of positive predictions made by a model	$Presicion = \frac{TP}{TP + FP}$
Recall or True Positive Rate (TPR)	The ability of a model to identify all relevant instances of a particular class	$Recall = \frac{TP}{TP + FN}$
F1-score	Evaluate the effectiveness of a classification model, particularly in situations where there is an imbalance between the positive and negative classes	$F1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$
False Positive Rate (FPR)	Calculates the proportion of actual negative instances that are incorrectly classified as positive	$FPR = \frac{FP}{FP + TN}$
Area Under Curve (AUC)	Give an outline of the model's ability to distinguish between positive and negative classes	$AUC = \sum_{i=1}^{n-1} (\frac{TPR_i + TPR_{i+1}}{2} \times (FPR_{i+1} - FPR_i))$

3.5 Dataset

It is often difficult to deal with a dataset with a huge number of images especially with unbalanced data, so in this paper we deal with a balanced dataset called "OCT images balanced version" [76]. This dataset contains 32064 retinal optical coherence tomography images classified into 4 classes as shown in the Table , while the Figure 7 shows a sample of the images. For all models used in this paper, the dataset was split into 80% for training and 20% for validation.

Table III: Number of images per each class

Class	CNV	DME	DRUSEN	NORMAL
Image number	8016	8016	8016	8016

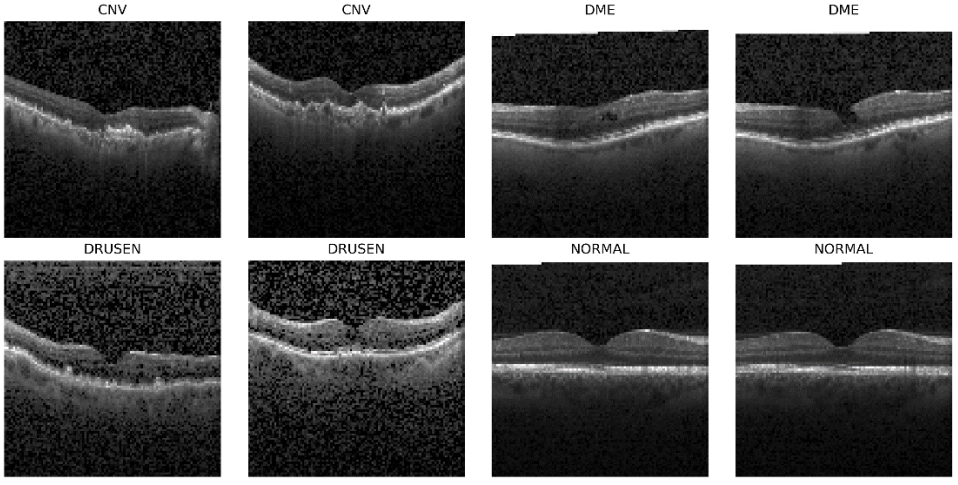

Figure 7 : Sample of images per each class in the dataset

4. RESULTS

The state-of-the-art models discussed in section three was evaluated in depth to assess the efficiency of these models on a specific dataset. In the following section we describe the results due to performance metrics.

While the dataset has four classes, eleven state-of-the-art models were implemented using three scenarios: the first with two classes, the second with three classes, and the third with four classes. Most of these models have two or more versions, each of which is evaluated separately.

As listed in the Table , the reported classification metrics of VGG16 and VGG19 on different scenarios provide us with some useful information on the models' performance. In the first scenario, VGG19 clearly outperforms VGG16, achieving an accuracy of 86.87% compared to 81.67%. Precision and recall are also improved in the VGG19. This gives us an F1-score of 86.85% and an AUC of 86.72%, which signifies that VGG19 works well in detecting and classifying samples in the binary classification. A strong performance in all metrics indicates that VGG19 possesses a balanced capability to capture true positives and reduce false positives. In the second scenario, both architectures perform worse than in the first scenario; VGG19 is still better than VGG16 in the accuracy metric by about 6%. Additionally, both models are not very accurate and sensitive, especially VGG16, which has a precision of only 55.76% and an F1 score of the same value. In this case, both models obtained a low value for AUC, showing that the positive and negative instances are difficult to distinguish accurately. The findings suggest that the poorer performance in the second scenario warrants further investigation. For the third scenario, the case is more alarming, where both models score low in most metrics. VGG16 exceeded in accuracy compared to the three classes scenario, and was approximately equal to VGG19, which is worse than the three classes scenario. The same weaknesses of precision and recall are found, especially for VGG16 with 51.12% in precision and 47.06% in recall. The same happens with the F1-scores, being even slightly inferior, with the VGG19 (54.22%) not being able to describe those positive instances effectively. The discriminative power of AUC is also low, which are 64.71 % for VGG16 and 65.31 % for VGG19. In conclusion, the work highlights that VGG19 is consistently better than VGG16. However, both models suffer from heavy degradation, especially in the (3 and 4) classes scenario. The better performance from VGG19 is presumably due to its deeper structure and more layers, which enable VGG19 to extract more intricate features and patterns from the dataset. This improved ability for feature extraction makes them generalise better, which is why they obtained higher precision, recall, and F1-scores when compared to VGG16. The clear decline in performance numbers between the three and four classes scenario indicates that the model needs further fine-tuning with other parameters and data augmentation.

Table IV: VGG performance

Class No.	Model	Accuracy	Precision	Recall	F1-score	AUC
2	VGG16	81.67%	84.9%	77.92%	81.26%	79.79%
2	VGG19	86.87%	87.13%	86.57%	86.85%	86.72%
3	VGG16	70.51%	55.76%	55.76%	55.76%	66.82%
3	VGG19	76.87%	69.78%	65.3%	67.74%	73.98%
4	VGG16	73.53%	51.12%	47.06%	49.01%	64.71%
4	VGG19	73.98%	62.35%	47.97%	54.22%	65.31%

The results of the ResNet models in all scenarios for the performance metrics indicate a generally low level of efficacy as shown in Table . In the first scenario with the two classes, while ResNet50 achieved the highest accuracy among the three models, it still falls short at just over 52%. The result shows that precision and recall scores are similarly low, indicating that the models struggle to accurately identify positive instances. For the second scenario, we observe a significant increase in accuracy in all models except ResNet50 which drops in accuracy. In this scenario, the drop occurred in the other metrics, indicating that many of the positive and negative instances are not being correctly classified. In scenario three, the ResNet models demonstrate improvements in all models in terms of accuracy, with ResNet151 achieving the highest accuracy of 62.54%, though precision remains critically low. Overall, ResNet50 tends to perform the best among the three models in the two classes scenaio, while ResNet101 and ResNet151 show similar performance in the second and third scenarios. The results show low performance across all metrics in all scenarios, which can be assigned to several factors, including the complexity of the dataset and potential overfitting, as well as the architectures' inability to capture the relevant features effectively. In general, for all the variants which have reported the results, although ResNet50 is found to be the best among

all the models, it could be observed that deeper models, say ResNet101 and ResNet151 are not always better for performance. This is a wonderful example of when less is more. In summary, while ResNet50 shows relatively good results in the binary classification, all models have unacceptable results and thus exhibit significant limitations. To improve the classification accuracy and reliability, the models must be refined further and augmented with additional data.

Table V: ResNet performance

Class No.	Model	Accuracy	Precision	Recall	F1-score	AUC
2	ResNet50	52.1%	55.14%	49.79%	52.33%	50.95%
2	ResNet101	47.83%	51.06%	49.38%	50.21%	48.61%
2	ResNet151	50.48%	48.32%	52.6%	50.37%	51.54%
3	ResNet50	38.3%	37.47%	37.4%	37.44%	53.08%
3	ResNet101	55.37%	37.26%	33.06%	35.03%	49.79%
3	ResNet151	55.08%	32.62%	37.98%	35.1%	52.14%
4	ResNet50	58.72%	24.25%	32.69%	27.84%	52.16%
4	ResNet101	62.38%	29.48%	24.76%	26.92%	49.84%
4	ResNet151	62.54%	25.07%	29.38%	27.06%	52.0%

The performance of the NASNet models for the three scenarios illustrated in Table shows notable challenges in classification effectiveness. In the first scenario, NASNet Mobile achieves a higher accuracy compared to NASNet Large, but it struggles with low precision and recall. This indicates it can correctly identify some instances, but it frequently misclassifies others, which means it misses a significant number of false positives and negatives. The performance of NASNet Large, despite its slightly higher precision, suggests that it is better at identifying true positives but still fails to maintain a strong overall performance. In the second scenario with three classes, the NASNet Mobile also exceeds NASNet Large slightly in terms of accuracy. But for both models exhibit similar low levels of precision and recall, pointing to a shared difficulty in effectively classifying this class. This consistent underperformance across both models suggests that the complexity of the dataset or the inherent characteristics of using three classes may be contributing factors. When examining the third scenario, NASNet Mobile, as a trend, outperformed in terms of accuracy, but its precision and recall remain low. NASNet Large shows a decline in performance metrics, indicating that it struggles even more than in previous classes. The overall trend across all classes highlights a significant need for improvement in capturing true positive instances and reducing misclassifications. Finally, NASNet Mobile performs better in terms of accuracy across all the scenarios, but NASNet Large achieves higher precision and recall metrics in the first scenario. The low performance across all metrics for both models suggests that there are some limitations of using these models, such as the complexity of the dataset, that hinder effective learning and classification. So to enhance performance many procedures may be made, such as further model tuning, data augmentation, or exploring alternative architectures will be crucial in addressing these issues and improving overall accuracy, precision, and recall.

Table VI: NasNet performance

Class No.	Model	Accuracy	Precision	Recall	F1-score	AUC
2	NASNet Mobile	58.95%	34.10%	25.02%	28.86%	50.45%
2	NASNet Large	50.31%	52.19%	49.88%	51.01%	50.1%
3	NASNet Mobile	56.19%	36.52%	34.28%	35.37%	50.71%
3	NASNet Large	54.03%	32.54%	30.83%	31.66%	48.55%
4	NASNet Mobile	62.74%	32.95%	25.48%	28.74%	50.32%
4	NASNet Large	61.43%	24.1%	23.17%	23.62%	48.93%

Model Xception has fair but modest results in all the scenarios considered, as shown in Table . In first experiment the model hav'nt very powerful performance but obtains balanced performance among the metric, means it has a stable detection for the relevant instances. While in the three class scenario, the model's performance is a little lower, and all the metrics are at low level except for accuracy, which indicates the persistent difficult in distinguishing all the instancess. A similar trend is also observed on the last experiment where the measures show a very poor performance of precision and recall. In general, although Xception is performance-balanced, the results indicates a room for improvement. Modifications to the model architecture or training process might improve the performance of this model.

Table VII: Xception performance

Class No.	Model	Accuracy	Precision	Recall	F1-score	AUC
2	Xception	48.24%	49.56%	46.54%	48.01%	48.29%
3	Xception	54.48%	32.83%	31.54%	32.18%	48.98%
4	Xception	62.0%	24.74%	24.06%	24.16%	49.41%

Varying levels of effectiveness in classification are highlighted across different scenarios illustrated in the performance metrics of the Inception models in Table . In the first scenario, Inception V2.0 stands out with the highest performance in all metrics, suggesting it has a better grasp of the class characteristics compared to the other versions. The overall results are almost similar for all the versions, for example, the precision of Inception V4.0 is lower than V2.0 by 0.4%, demonstrating similar precision levels, indicating that they struggle to identify true positives effectively. In the second scenario, all Inception versions show an increase in accuracy. Inception V3.0 came in first place with 55.68%, making it the best in this scenario among the other versions. This increase in accuracy does not prevent precision and recall from decreasing, which means that not all versions effectively distinguish between positive and negative instances. The trend is continuous in scenario three, where the increase in accuracy was found, and the precision and recall are also decreased. The low F1-scores across the scenarios further emphasize the difficulties faced in balancing precision and recall, leading to overall mediocre performance. Overall, Inception V2.0 is the best version among the other in both scenario one and two, demonstrating a better ability to capture the nuances of these classes. The Inception V2.0 focuses on optimizing the depth and width of the network, so this refinement allows it to capture more intricate patterns within the data, leading to better classification accuracy. At the same time, Inception V3.0 leads in scenario 2, because of its advanced convolutional techniques, including factorized convolutions, which allow for a more efficient representation of the input data. This leads to better feature extraction, enabling the model to capture relevant patterns more effectively. With each version of the Inception models, there are an refinements in architecture, such as improved convolutional layers, batch normalization, and optimized layer connections, which enable newer models like Inception V2.0 and V3.0 to capture complex patterns in the data better, leading to improved accuracy and classification performance. The accuracy of different versions of Inception model is illustrated in Figure 8.

Table VIII: Inception V1.0 performance

Class No.	Model	Accuracy	Precision	Recall	F1-score	AUC
2	Inception V1.0	49.54%	49.94%	48.23%	49.1%	49.55%
2	Inception V2.0	51.55%	50.9%	52.5%	51.69%	51.56%
2	Inception V3.0	49.78%	49.56%	49.03%	49.29%	49.78%
2	Inception V4.0	49.46%	50.5%	50.09%	50.3%	49.44%
3	Inception V1.0	55.0%	33.02%	32.41%	32.71%	49.46%
3	Inception V2.0	54.86%	33.13%	32.15%	32.63%	49.36%
3	Inception V3.0	55.68%	32.63%	33.71%	33.16%	50.01%
3	Inception V4.0	55.53%	33.44%	33.27%	33.35%	49.55%
4	Inception V1.0	63.34%	26.04%	26.57%	26.31%	50.96%

4	Inception V2.0	66.18%	26.03%	32.19%	28.78%	53.77%
4	Inception V3.0	62.7%	24.2%	25.17%	24.68%	49.95%
4	Inception V4.0	62.28%	25.11%	24.67%	24.89%	49.85%

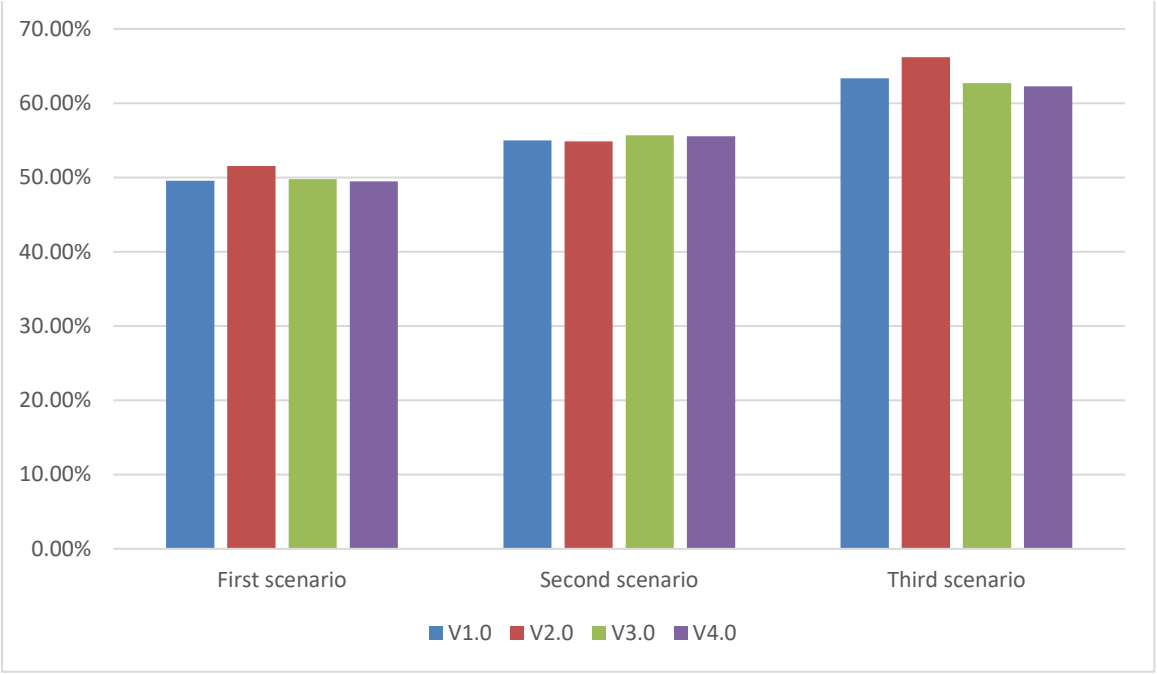

Figure 8: The Inception versions comparison

The evaluation of various EfficientNet models across different scenarios provides valuable insights into their performance metrics, including accuracy, precision, recall, F1-score, and AUC as shown in Table . This analysis will delve into the strengths and weaknesses of each model, highlighting the best performers in specific classes and discussing the factors contributing to lower performance in others. The best version of EfficintNet models in the first scenario is EfficientNet-B0 in terms of accuracy of 53.97%. This model balances precision (50.17%) and recall (58.28%), resulting in an F1-score of 53.92%. The result of recall which is relatively high indicates that it successfully identifies a significant portion of the positive cases, which is crucial in applications such as medical diagnostics where missing a case can have serious implications. For the second scenario, EfficientNet-B2 emerges as the best performer with an accuracy of 55.98%. Although it has a relatively low precision of 34.28%, its recall of 33.90% indicates that it can identify a fair number of true positive cases. The result of F1-score suggests that the model while it struggles with false positives, it still captures a relevant portion of positive cases, making it useful in scenarios where some false positives can be tolerated. Finally, in the last scenario, the superiority of the accuracy among the models goes to EfficientNet-B5 with 68.86%, while achieved a precision of 53.72% and a recall of 29.90%, resulting in an F1-score of 38.41%. Although it showcases strong accuracy, its low recall presents challenges in capturing all positive cases, indicating that it may be overconfident in its predictions. This highlights the need for further refinement to improve its ability to generalize across different classes. The results in Table and its analysis show a specific limitation for each model that impact its overall performance, for example, EfficientNet-B0 is superior in the first scenario, but the metrics can hinder its effectiveness in other scenarios, suggesting a lack of generalizability. As the EfficientNet-B1 model shows low performance in all scenarios, it indicates that the model may be too simplistic or not well-tuned according to the complexity of the data. While, EfficientNet-B2, although performing best in the second scenario, it faces challenges with high false positives, potentially due to overfitting or inadequate feature extraction. Despite EfficientNet-B5 performed well in the third scenario, the model suffered from low recall, missing many positive cases, so this tendency to overestimate its predictions could limit its practical application. Lastly, EfficientNet-B4 consistently underperforms across all classes, achieving accuracy below 50%. Its low precision and recall highlight its inability to learn essential features, rendering it unsuitable for effective deployment.

Table IX: EfficientNet performance

Class No.	Model	Accuracy	Precision	Recall	F1-score	AUC
2	EfficientNet-B0	53.97%	50.17%	58.28%	53.92%	54.27%
2	EfficientNet-B1	49.85%	49.49%	49.83%	49.66%	49.85%
2	EfficientNet-B2	52.72%	55.16%	53.9%	54.52%	52.65%
2	EfficientNet-B3	51.21%	50.86%	50.29%	50.27%	51.2%
2	EfficientNet-B4	46.84%	46.91%	45.86%	46.38%	46.84%
2	EfficientNet-B5	49.49%	49.47%	49.16%	49.31%	49.49%
2	EfficientNet-B6	50.58%	51.15%	50.57%	50.86%	50.58%
2	EfficientNet-B7	50.06%	49.8%	50.11%	49.95%	50.06%
3	EfficientNet-B0	54.4%	32.12%	31.52%	31.82%	48.79%
3	EfficientNet-B1	53.11%	30.62%	29.55%	30.08%	47.43%
3	EfficientNet-B2	55.98%	34.28%	33.9%	34.09%	50.52%
3	EfficientNet-B3	54.49%	33.56%	31.46%	32.47%	49.12%
3	EfficientNet-B4	53.47%	31.39%	30.87%	31.13%	48.01%
3	EfficientNet-B5	62.57%	32.98%	32.88%	32.93%	53.48%
3	EfficientNet-B6	55.36%	32.96%	33.12%	33.04%	49.78%
3	EfficientNet-B7	55.5%	33.31%	33.19%	33.25%	49.94%
4	EfficientNet-B0	64.43%	28.54%	28.83%	28.68%	52.5%
4	EfficientNet-B1	62.23%	25.0%	24.57%	24.78%	49.78%
4	EfficientNet-B2	65.36%	23.95%	30.21%	26.72%	52.43%
4	EfficientNet-B3	63.25%	26.93%	26.1%	26.51%	51.0%
4	EfficientNet-B4	47.65%	20.17%	22.01%	21.05%	40.78%
4	EfficientNet-B5	68.86%	53.72%	29.9%	38.41%	58.75%
4	EfficientNet-B6	62.38%	24.38%	24.7%	24.77%	49.84%
4	EfficientNet-B7	62.29%	24.56%	24.59%	24.58%	49.72%

The results of DenseNet models, as shown in Table , illustrate the superiority of the models compared to others. In the first scenario, DenseNet169 is the superior model, achieving an impressive accuracy of 91.48% compared to other models. This high accuracy is complemented by the results of other metrics which all above 90%. This results shows that the increased in depth of model allows for a more nuanced understanding of complex features within the data, leading to better generalization and performance. The enhanced feature extraction capabilities likely explain its leading position in this class, as it can capture more intricate patterns that are crucial for accurate classification. DenseNet121, while still effective, falls behind with an accuracy of 85.34%. The restriction of the model's ability to learn the complex representations which is necessary for optimal performance comes from the shorter depth of this model. DenseNet201, despite its additional layers,

performed slightly better than DenseNet121 with an accuracy of 90.21%, but, it did not match or surpass DenseNet169. So, merely increasing depth does not always yield proportional improvements in results, potentially due to overfitting or diminishing returns on feature extraction. In the second scenario, DenseNet201 although decreases the accuracy, but it achieved the highest among other models with 82.66%, showcasing its capability to handle this particular classification task effectively, but also the model's precision (73.88%) and recall (74.09%) reveal that it struggles with false positives and negatives, which can impact its reliability in practical applications. DenseNet121 and DenseNet169 displayed lower accuracies of 80.19% and 78.57%, respectively. On the other hand, DenseNet201 is capable of learning detailed features, but it still suffers from some noise that occurs due to the complexity, leading to lower precision and recall rates. This suggests that the model might be too intricate for the underlying data distribution in this scenario, leading to ineffective generalization. The final experiment showed a contrast in model performance. DenseNet121 again takes the lead with an accuracy of 84.12%. However, overall performance metrics across all models in this class were lower compared to the first and second scenarios. The final experiment showed a contrast in model performance. DenseNet121 again takes the lead with an accuracy of 84.12%. However, overall performance metrics across all models in this class were lower compared to the first and second scenarios. DenseNet201 shows dismal performance with an accuracy of 51.54% only, this result shows that the increased model complexity may exacerbate the challenges of learning from potentially noisy or ambiguous data. Finally, DenseNet169 achieved an accuracy of 66.10% which indicates that the dense connections of this architecture do not well capture the features relevant to this scenario. The drop in performance metrics across all models in this class highlights the need for tailored model architectures that can better accommodate the specific characteristics of the data. The accuracy of different versions of Inception model is illustrated in Figure 9.

Table X :DenseNet performance

Class No.	Model	Accuracy	Precision	Recall	F1-score	AUC
2	DenseNet121	85.34%	85.71%	86.12%	85.91%	85.31%
2	DenseNet169	91.48%	94.96%	91.89%	93.4%	91.3%
2	DenseNet201	90.21%	89.67%	90.85%	90.26%	90.21%
3	DenseNet121	80.19%	70.45%	70.1%	70.27%	77.68%
3	DenseNet169	78.57%	67.78%	67.92%	67.85%	75.9%
3	DenseNet201	82.66%	73.88%	74.09%	73.99	80.51%
4	DenseNet121	84.12%	67.86%	68.61%	68.23%	78.93%
4	DenseNet169	66.1%	67.67%	67.58%	67.62	66.02%
4	DenseNet201	51.54%	32.85%	32.72%	32.78%	47.45%

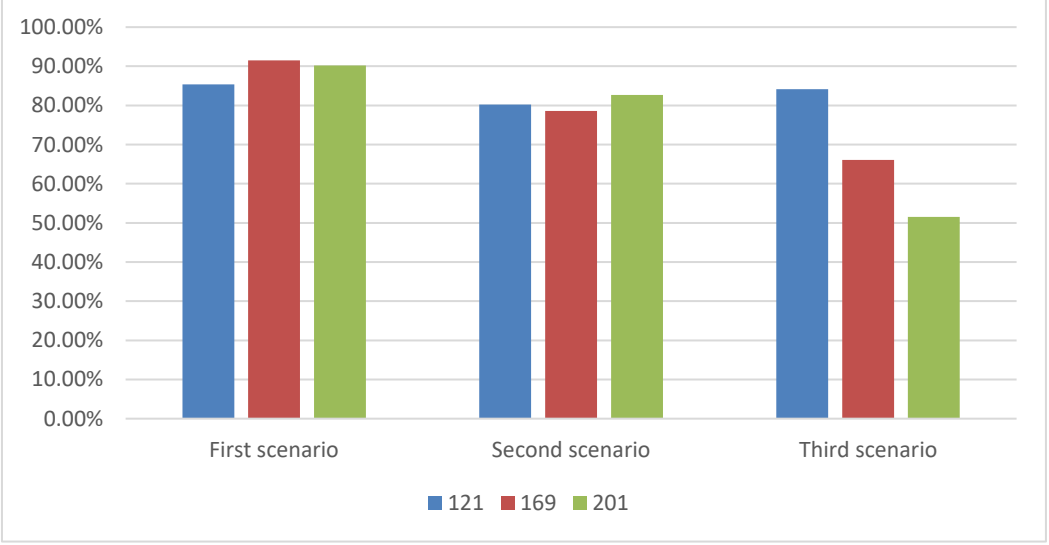

Figure 9: The DenseNet versions comparison

The performance of AlexNet across the three scenarios, as shown in Table , indicates generally good results. We will start with the first scenario where AlexNet achieved an impressive accuracy of 91.92%, indicating a strong ability to classify

this category correctly. This high accuracy is supported by a precision of 91.52%, an AUC of 91.93%, and an F1-score of 91.87%. The recall achieved a good result with 92.22% which means that the model is effective in identifying true positives, making it reliable for applications where the correct identification of this class is critical. The superiority of AlexNet in this scenario can be attributed to its architecture, which effectively captures relevant features through its layered approach. The model's ability to learn from a diverse set of features allows it to generalize well to the data, leading to high performance in both precision and recall. For the second scenario, when analyzing the results, we notice that the model recorded less accuracy than the previous one. The trend is also for the precision and recall metrics, highlighting potential challenges in correctly identifying instances of this scenario. While the F1-score struggles with false negatives, as evidenced by the lower recall. The AUC of 82.22% suggests that the model has a moderate ability to distinguish between classes, but there is room for improvement. The decline in performance in this scenario could be attributed to the complexity or variability of the data. AlexNet, while effective for many tasks, may not capture the intricacies of this particular dataset as well as deeper architectures. This limitation can lead to misclassifications, resulting in lower precision and recall. For the third scenari, AlexNet achieved an accuracy of 89.3%, which is a moderate performance compared to first but an improvement over the second. The precision of 79.71% and recall of 77.5% indicate that while the model is generally effective, it still faces challenges in identifying true positives. The F1-score of 78.59% reinforces this point, indicating a balance between precision and recall. The AUC of 85.4% suggests that the model is reasonably good at distinguishing between classes, though not as strong as in the first scenario. Although the model has moderate accuracy which can recognize patterns, it may not be able to fully leverage the features necessary for optimal classification.

Table XI: AlexNet performance

Class No.	Model	Accuracy	Precision	Recall	F1-score	AUC
2	AlexNet	91.92%	91.52%	92.22%	91.87%	91.93%
3	AlexNet	84.58%	79.85%	74.46%	77.06%	82.22%
4	AlexNet	89.3%	79.71%	77.5%	78.59%	85.4%

The performance of LeNet model among the three scenarios clearly demonstrates that classification ability is not good, as shown in Table . In the two classes scenario, LeNet achieves better accuracy than in the first scenario. However, it is still struggling, and its precision and recall indicate that the model has difficulties correctly identifying the relevant instances. The low recall value implies many true positives are not caught and result in untrustworthy predictions. In the three classes scenario, the accuracy deteriorates even more. There are not so many relevant instances, indeed, and the model leaves out many of them. This reveals a critical issue that needs to be addressed. In the last scenario, the model still performs badly as it has poor metrics in all the categories. There is a significant lack of capability to accurately discriminate between the relevant samples, leading to a low number of useful positive assignments.

Table XII: LeNet performance

Class No.	Model	Accuracy	Precision	Recall	F1-score	AUC
2	LeNet	51.05%	50.37%	52.35%	51.34%	51.07%
3	LeNet	58.52%	36.74%	38.87%	37.78%	53.4%
4	LeNet	65.43%	30.61%	30.85%	30.73%	53.86%

5. Discussions

Figure 10 illustrates the relationship among the top three models, showing the superiority of AlexNet in all scenarios. This superiority of AlexNet comes from the innovative architecture of the model, where it has multiple convolutional and pooling layers, enabling it to learn complex features from images, and also the use of ReLU as an activation function, which helps to reduce the vanishing gradient problem, leading to faster training and better performance compared to others. While DenseNet suffers little from AlexNet, it is still a good choice in image classification, especially with the good performance shown in this paper. The good results of both DenseNet 169 and 201 come from the effective architecture of the model, particularly the dense connections, which help alleviate the vanishing gradient problem by providing more direct paths for gradients during backpropagation, leading to more effective training. Also the model mitigates the risk of overfitting by reusing features, which leads to a reduction in the number of parameters. Finally, we must note that not always does the big

depth of the model lead to a good result, as shown in the Figure 10, which illustrates the superiority of DenseNet169 over DenseNet201 in all scenarios except the second scenario.

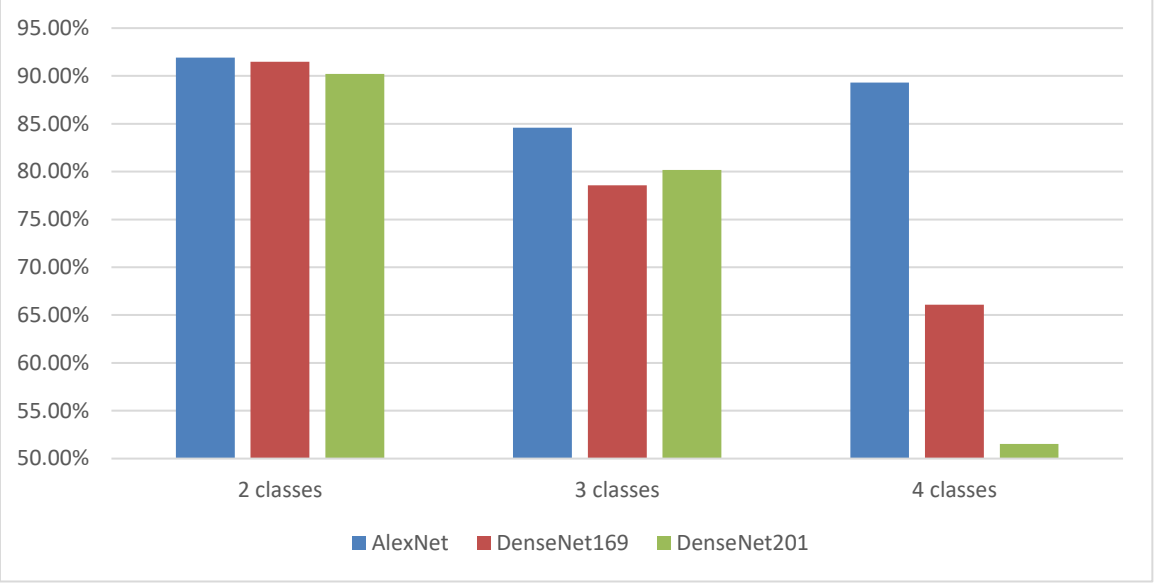

Figure 10: Top three models comparison

Table 20 shows a comparison with work in [77]. Two scenarios were used, the first is the binary classification, while the second uses the whole images in the dataset. Although the table shows superiority of the paper in [77], the AlexNet model performs well. The cause of this superiority lies in the preprocessing procedures implemented in [77], whereas our paper did not perform any preprocessing on the images to highlight the strengths and weaknesses of the models. We are confident that if preprocessing were applied, the performance would be even better than that of [77].

Table XIII: AlexNet comparison with [70]

Model	Accuracy
[77] binary classification	100%
AlexNet binary classification	91.92%
[77] whole dataset	96.6%
AlexNet whole dataset	89.3%

Due to the great performance without preprocessing for the AlexNet and DenseNet models, many practical implications for real-world applications are benefited from these models, especially in medical diagnostics, which most use the classification of images. This significance comes from their ability to automate complex image processing tasks, which leads to improved efficiency, accuracy, and decision-making.

6. CONCLUSION

In the past decade, CNNs have been widely used in image classification, because they are able to automatically learn and extract features. This has significantly enhanced the classification accuracy on analysis of a wide range of different applications. CNN can encode spatial hierarchies present in the data efficiently due to their hierarchical design and therefore they perform better in solving difficult classification problems that the traditional methods cannot solve properly. Their performance in the field lends the value that prompts more exploration on deep learning models to achieve new limits of what can be done in image classification.

By analyzing the results, the AlexNet model achieved an impressive accuracy of 91.92% with a good precision and recall making it stand out as a notable performer overall along with DenseNet169 and DenseNet201, particularly in binary classification. We conclude that these excel at correctly identifying true positives while maintaining a low false positive rate, which is critical for applications requiring high reliability. So we can understand that the modification of this model

represented by dense connectivity, allows for efficient feature utilization, leading to superior performance in complex classification tasks.

In contrast, the EfficientNet-B4 model makes the worst performer across all models with an accuracy in the binary classification of only 46.84%, with similarly low precision and recall values, indicating a failure to classify instances effectively. This worst performance of this model may come from the simplistic architecture of the model which limit its ability to learn the necessary features from the data, leading to high misclassification rates. The results highlight that despite being lightweight, EfficientNet-B4 does not provide the robustness needed for complex image classification tasks, particularly in scenarios where precision and recall are paramount. This low performance is not only in EfficientNet-B4 but also in another models such as ResNet101, Xception, Inception, EfficientNet-B1, and EfficientNet-B5. Many resons for this low performance of the models, such as model depth, over complexity, absence of parameter optimization, the inability to handle complex data. Many strategies can be done to enhance performance, such as increasing the model depth which could increase the number of extract effective features, utilizing techniques such as transfer learning can significantly improve classification accuracy, implementing advanced regularization methods, data augmentation, and ensemble techniques may also help mitigate overfitting and improve generalization.

The key findings from this study highlight the importance of model architecture and thus achieving high classification performance. As an example, DenseNet169's superior performance illustrates the benefits of using deeper networks with efficient feature extraction capabilities. These findings are considered important for practitioners in many fields such as medical imaging, where classification instances accuratlly is critical. The ability to reliably identify conditions from images can lead to improved diagnostic outcomes and patient care. According to the saying "No one fits all", the study highlights the necessity for careful model selection based on the specific characteristics of the dataset and the application requirements. For instance, while lightweight models like ResNet50 may be suitable for applications with limited computational resources, they fall short in accuracy and reliability in more demanding tasks. The study also showed that there are many reliable models such as AlexNet and DenseNet169 can handle more complex datasets and provide higher accuracy, precision, and recall, making them suitable for various practical applications.

Although the excellent performance of the AlexNet and DenseNet models, there are many limitations to using them. The AlexNet can be computationally intensive, particularly on larger datasets, and it can overfit when trained on small datasets, as it has a large number of parameters. Finally, AlexNet could be considered shallow according to its architecture, which may limit its ability to capture more complex features in data. While the DenseNet can lead to high memory usage, particularly with deeper networks, according to its architecture, especially with dense connections which also lead to longer training times compared to simpler architectures.

In conclusion, the results of these models illuminate the capabilities and limitations of various architectures in image classification tasks. DenseNet169 stands out as a leader in performance, while EfficientNet-B4 demonstrates the pitfalls of overly simplistic models. The insights gained from this evaluation can guide future research and practical applications, emphasizing the need for robust and reliable models in critical domains.

For future work, increasing the number of datasets will help illustrate the performance and effectiveness of the models. Also, preprocessing is performed on the datasets, including augmentation and image enhancement, to demonstrate the performance of the models before and after these procedures.

Conflicts of Interest

The author's disclosure statement confirms the absence of any conflicts of interest.

Funding

The authors had no institutional or sponsor backing.

Acknowledgment

The authors expresses appreciation to their institutions for their continuous support and access to relevant research materials.

7. References

[1] R. U. Khan, X. Zhang, R. Kumar and E. O. Aboagye, "Evaluating the performance of resnet model based on image recognition," in *Proceedings of the 2018 international conference on computing and artificial intelligence*, 2018, pp. 86-90. doi: https://doi.org/10.1145/3194452.3194461.

[2] D. Singh, V. Kumar and M. Kaur, "Densely connected convolutional networks-based COVID-19 screening model," *Applied Intelligence,* vol. 51, pp. 3044-3051, 20212021.DOI: https://doi.org/10.1007/s10489-020-02149-6.

[3] Y. Hu, C. Tian, J. Zhang and S. Zhang, "Efficient image denoising with heterogeneous kernel-based CNN," *Neurocomputing,* vol. 592, p. 127799, 20242024.DOI: https://doi.org/10.1016/j.neucom.2024.127799.

[4] N. N. Prakash, V. Rajesh, D. L. Namakhwa, S. D. Pande and S. H. Ahammad, "A DenseNet CNN-based liver lesion prediction and classification for future medical diagnosis," *Scientific African,* vol. 20, p. e01629, 20232023.DOI: https://doi.org/10.1016/j.sciaf.2023.e01629.

[5] K. A. Fathy, H. K. Yaseen, M. T. Abou-Kreisha and K. A. ElDahshan, "A novel meta-heuristic optimization algorithm in white blood cells classification," *CMC-Comput Mater Contin,* vol. 75, no. 1, pp. 1527-1545, 20232023.DOI: 0.32604/cmc.2023.036322.

[6] G. Zheng, G. Han and N. Q. Soomro, "An inception module CNN classifiers fusion method on pulmonary nodule diagnosis by signs," *Tsinghua Science and Technology,* vol. 25, no. 3, pp. 368-383, 20192019.DOI: 10.26599/TST.2019.9010010.

[7] G. Marques, D. Agarwal and I. De la Torre Díez, "Automated medical diagnosis of COVID-19 through EfficientNet convolutional neural network," *Applied soft computing,* vol. 96, p. 106691, 20202020.DOI: https://doi.org/10.1016/j.asoc.2020.106691.

[8] A. Kamilaris and F. X. Prenafeta-Boldú, "A review of the use of convolutional neural networks in agriculture," *The Journal of Agricultural Science,* vol. 156, no. 3, pp. 312-322, 20182018.DOI: https://doi.org/10.1017/S0021859618000436.

[9] L. D. Medus, M. Saban, J. V. Francés-Víllora, M. Bataller-Mompeán and A. Rosado-Muñoz, "Hyperspectral image classification using CNN: Application to industrial food packaging," *Food Control,* vol. 125, p. 107962, 20212021.DOI: https://doi.org/10.1016/j.foodcont.2021.107962.

[10] A. W. Salehi, S. Khan, G. Gupta, B. I. Alabduallah, A. Almjally *et al.*, "A study of CNN and transfer learning in medical imaging: Advantages, challenges, future scope," *Sustainability,* vol. 15, no. 7, p. 5930, 20232023.

[11] Z. Li, F. Liu, W. Yang, S. Peng and J. Zhou, "A survey of convolutional neural networks: analysis, applications, and prospects," *IEEE transactions on neural networks and learning systems,* vol. 33, no. 12, pp. 6999-7019, 20212021.DOI: 10.1109/TNNLS.2021.3084827.

[12] R. Yamashita, M. Nishio, R. K. G. Do and K. Togashi, "Convolutional neural networks: an overview and application in radiology," *Insights into imaging,* vol. 9, pp. 611-629, 20182018.DOI: https://doi.org/10.1007/s13244-018-0639-9.

[13] M. A. Saleem, N. Senan, F. Wahid, M. Aamir, A. Samad *et al.*, "Comparative analysis of recent architecture of convolutional neural network," *Mathematical Problems in Engineering,* vol. 2022, no. 1, p. 7313612, 20222022.DOI: https://doi.org/10.1155/2022/7313612.

[14] F. A. Hashim, Y. M. Mohialden and N. M. Hussien, "Hybrid Feature Selection and Ensemble Classification for Climate Change Indicators: A Machine Learning Approach," *Terra Joule Journal,* vol. 1, no. 2, p. 8, 20252025.DOI: https://doi.org/10.64071/3080-5724.1021.

[15] L. L. Scientific, "PREDICTING FRAUD: A MACHINE LEARNING APPROACH TO SECURE TRANSACTIONS IN CREDIT CARD SYSTEM," *Journal of Theoretical and Applied Information Technology,* vol. 103, no. 14, 20252025.

[16] M. El Sakka, M. Ivanovici, L. Chaari and J. Mothe, "A review of CNN applications in smart agriculture using multimodal data," *Sensors,* vol. 25, no. 2, p. 472, 20252025.DOI: https://doi.org/10.3390/s25020472.

[17] M. A. Islam, M. Kowal, S. Jia, K. G. Derpanis and N. D. Bruce, "Position, padding and predictions: A deeper look at position information in cnns," *International Journal of Computer Vision,* vol. 132, no. 9, pp. 3889-3910, 20242024.DOI: https://doi.org/10.1007/s11263-024-02069-9.

[18] H. Bakır, "Evaluating the impact of tuned pre-trained architectures' feature maps on deep learning model performance for tomato disease detection," *Multimedia Tools and Applications,* vol. 83, no. 6, pp. 18147-18168, 20242024.DOI: https://doi.org/10.1007/s11042-023-17503-2.

[19] S. S. Kareem, B. I. Hameed and H. K. Yaseen, "Enhanced Mutual Authentication Scheme for Fog Computing using Blockchain Technology," *Journal of Advanced Research Design,* vol. 141, no. 1, pp. 163-188, 20262026.DOI: https://doi.org/10.37934/ard.141.1.163188.

[20] M. Sun, Z. Song, X. Jiang, J. Pan and Y. Pang, "Learning pooling for convolutional neural network," *Neurocomputing,* vol. 224, pp. 96-104, 20172017.DOI: https://doi.org/10.1016/j.neucom.2016.10.049.

[21] S. Tammina, "Transfer learning using vgg-16 with deep convolutional neural network for classifying images," *International Journal of Scientific and Research Publications (IJSRP),* vol. 9, no. 10, pp. 143-150, 20192019.DOI: http://dx.doi.org/10.29322/IJSRP.9.10.2019.p9420.

[22] C. Banerjee, T. Mukherjee and E. Pasiliao Jr, "An empirical study on generalizations of the ReLU activation function," in *Proceedings of the 2019 ACM southeast conference*, 2019, pp. 164-167. doi: https://doi.org/10.1145/3299815.3314450.

[23] L. Lu, Y. Shin, Y. Su and G. E. Karniadakis, "Dying relu and initialization: Theory and numerical examples," *arXiv preprint arXiv:1903.06733*, 20192019.DOI:https://doi.org/10.4208/cicp.OA-2020-0165.

[24] S. Mugunthan and T. Vijayakumar, "Design of improved version of sigmoidal function with biases for classification task in ELM domain," *Journal of Soft Computing Paradigm (JSCP),* vol. 3, no. 02, pp. 70-82, 20212021.DOI: https://doi.org/10.36548/jscp.2021.2.002.

[25] A. Kumar and S. S. Sodhi, "Some modified activation functions of hyperbolic tangent (TanH) activation function for artificial neural networks," in *International conference on innovations in data analytics*, 2022, pp. 369-392: Springer. doi: https://doi.org/10.1007/978-981-99-0550-8_30.

[26] M. Wang, S. Lu, D. Zhu, J. Lin and Z. Wang, "A high-speed and low-complexity architecture for softmax function in deep learning," in *2018 IEEE asia pacific conference on circuits and systems (APCCAS)*, 2018, pp. 223-226: IEEE. doi: 10.1109/APCCAS.2018.8605654.

[27] S. S. Basha, S. R. Dubey, V. Pulabaigari and S. Mukherjee, "Impact of fully connected layers on performance of convolutional neural networks for image classification," *Neurocomputing,* vol. 378, pp. 112-119, 20202020.DOI: https://doi.org/10.1016/j.neucom.2019.10.008.

[28] W. Yuan, X. Gu, Z. Dai, S. Zhu and P. Tan, "Neural window fully-connected crfs for monocular depth estimation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 3916-3925. doi: https://doi.org/10.48550/arXiv.2203.01502.

[29] L. Mohammadpour, T. C. Ling, C. S. Liew and A. Aryanfar, "A survey of CNN-based network intrusion detection," *Applied Sciences,* vol. 12, no. 16, p. 8162, 20222022.DOI: https://doi.org/10.3390/app12168162.

[30] J. Ayeni, "Convolutional neural network (CNN): the architecture and applications," *Appl. J. Phys. Sci,* vol. 4, no. 4, pp. 42-50, 20222022.DOI: https://doi.org/10.31248/AJPS2022.085.

[31] S. M. Muhammed, G. Abdul-Majeed and M. S. Mahmoud, "Weighting Heart Disease Criteria Using Multi-Criteria Decision-Making," in *2023 Al-Sadiq International Conference on Communication and Information Technology (AICCIT)*, 2023, pp. 342-346: IEEE. doi: 10.1109/AICCIT57614.2023.10218189.

[32] P. Murugan, "Feed forward and backward run in deep convolution neural network," *arXiv preprint arXiv:1711.03278*, 20172017.DOI: https://doi.org/10.48550/arXiv.1711.03278.

[33] J. Yang and G. Yang, "Modified convolutional neural network based on dropout and the stochastic gradient descent optimizer," *Algorithms,* vol. 11, no. 3, p. 28, 20182018.DOI: https://doi.org/10.3390/a11030028.

[34] S. M. Muhammed, G. Abdul-Majeed and M. S. Mahmoud, "Early prediction of chronic heart disease based on electronic triage dataset by using machine learning," in *2023 Al-Sadiq International Conference on Communication and Information Technology (AICCIT)*, 2023, pp. 131-136: IEEE. doi: 10.1109/AICCIT57614.2023.10218241.

[35] Y. LeCun, P. Haffner, L. Bottou and Y. Bengio, "Object recognition with gradient-based learning," in *Shape, contour and grouping in computer vision*: Springer, pp. 319-345 1999.

[36] M. F. Ferraz, B. K. Friesel and O. Spinczyk, "Pros and Cons of Executable Neural Networks for Deeply Embedded Systems," in *Proceedings of the 2023 Workshop on Compilers, Deployment, and Tooling for Edge AI*, 2023, pp. 16-20. doi: https://doi.org/10.1145/3615338.3618118.

[37] M. Lu and N. Ke, "LeNet-5 handwritten digit recognition based on deep learning," in *International Conference on Optics, Electronics, and Communication Engineering (OECE 2024)*, 2024, vol. 13395, pp. 977-982: SPIE. doi: https://doi.org/10.1117/12.3049259.

[38] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li *et al.*, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, 2009, pp. 248-255: Ieee. doi: 10.1109/CVPR.2009.5206848.

[39] A. Krizhevsky, I. Sutskever and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems,* vol. 25, 20122012.DOI: https://doi.org/10.1145/3065386.

[40] K. A. Fathy, H. K. Yaseen, M. T. Abou-Kreisha and K. A. ElDahshan, "A novel meta-heuristic optimization algorithm in white blood cells classification," *Comput., Mater. Continua,* vol. 75, no. 1, pp. 1527-1545, 20232023.DOI: 0.32604/cmc.2023.036322

[41] P. Kalaiarasi and P. Esther Rani, "A comparative analysis of AlexNet and GoogLeNet with a simple DCNN for face recognition," in *Advances in Smart System Technologies: Select Proceedings of ICFSST 2019*: Springer, pp. 655-668 2020.

[42] M. T. Abou-Kreisha, H. K. Yaseen, K. A. Fathy, E. A. Ebeid and K. A. ElDahshan, "Multisource smart computer-aided system for mining COVID-19 infection data," in *Healthcare*, 2022, vol. 10, no. 1, p. 109: MDPI. doi: https://doi.org/10.3390/healthcare10010109.

[43] L. Wang, S. Guo, W. Huang and Y. Qiao, "Places205-vggnet models for scene recognition," *arXiv preprint arXiv:1508.01667*, 20152015.DOI: https://doi.org/10.48550/arXiv.1508.01667.

[44] M. S. Majib, M. M. Rahman, T. S. Sazzad, N. I. Khan and S. K. Dey, "Vgg-scnet: A vgg net-based deep learning framework for brain tumor detection on mri images," *IEEE Access,* vol. 9, pp. 116942-116952, 20212021.DOI: 10.1109/ACCESS.2021.3105874.

[45] Z. Li, B. Li, S. G. Jahng and C. Jung, "Improved vgg algorithm for visual prosthesis image recognition," *IEEE Access,* vol. 12, pp. 45727-45739, 20242024.DOI: 10.1109/ACCESS.2024.3380839.

[46] K. He, X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770-778.

[47] M. Subhi, O. F. Rashid, S. A. Abdulsahib, M. K. Hussein and S. M. Mohammed, "Anomaly Intrusion Detection Method based on RNA Encoding and ResNet50 Model," *Mesopotamian Journal of CyberSecurity,* vol. 4, no. 2, pp. 120-128, 20242024.DOI: https://doi.org/10.58496/MJCS/2024/011.

[48] T. S. Prajwal and I. AK, "A comparative study Of RESNET-pretrained models for computer vision," in *Proceedings of the 2023 Fifteenth International Conference on Contemporary Computing*, 2023, pp. 419-425. doi: https://doi.org/10.1145/3607947.3608042.

[49] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818-2826.

[50] S. Barratt and R. Sharma, "A note on the inception score," *arXiv preprint arXiv:1801.01973*, 20182018.DOI: https://doi.org/10.48550/arXiv.1801.01973.

[51] H. K. Yaseen and A. M. Obaid, "Big data: Definition, architecture & applications," *JOIV: International Journal on Informatics Visualization,* vol. 4, no. 1, pp. 45-51, 20202020.DOI: http://dx.doi.org/10.30630/joiv.4.1.292.

[52] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*, 2019, pp. 6105-6114: PMLR.

[53] X. Wang, H. Ren and A. Wang, "Smish: A novel activation function for deep learning methods," *Electronics,* vol. 11, no. 4, p. 540, 20222022.DOI: https://doi.org/10.3390/electronics11040540.

[54] K. Kansal, T. B. Chandra and A. Singh, "ResNet-50 vs. EfficientNet-B0: multi-centric classification of various lung abnormalities using deep learning," *Procedia Computer Science,* vol. 235, pp. 70-80, 20242024.DOI: https://doi.org/10.1016/j.procs.2024.04.007.

[55] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *arXiv preprint arXiv:1611.01578*, 20162016.DOI: https://doi.org/10.48550/arXiv.1611.01578.

[56] W. K. Mohammed, M. A. Taha and S. M. Mohammed, "A novel hybrid fusion model for intrusion detection systems using benchmark checklist comparisons," *Mesopotamian Journal of CyberSecurity,* vol. 4, no. 3, pp. 216-232, 20242024.DOI: https://doi.org/10.58496/MJCS/2024/024.

[57] B. Zoph, V. Vasudevan, J. Shlens and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8697-8710.

[58] K. Radhika, K. Devika, T. Aswathi, P. Sreevidya, V. Sowmya *et al.*, "Performance analysis of NASNet on unconstrained ear recognition," in *Nature inspired computing for data science*: Springer, pp. 57-82 2019.

[59] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251-1258.

[60] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510-4520.

[61] M. A. Hasan and K. Dey, "Depthwise Separable Convolutions with Deep Residual Convolutions," *arXiv preprint arXiv:2411.07544*, 20242024.DOI: https://doi.org/10.48550/arXiv.2411.07544.

[62] F. Salim, F. Saeed, S. Basurra, S. N. Qasem and T. Al-Hadhrami, "DenseNet-201 and Xception pre-trained deep learning models for fruit recognition," *Electronics,* vol. 12, no. 14, p. 3132, 20232023.

[63] G. Huang, Z. Liu, L. Van Der Maaten and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700-4708. doi: https://doi.org/10.48550/arXiv.1608.06993.

[64] S. Li, Y. Sun, G. G. Yen and M. Zhang, "Automatic design of convolutional neural network architectures under resource constraints," *IEEE Transactions on Neural Networks and Learning Systems,* vol. 34, no. 8, pp. 3832-3846, 20212021.DOI: 10.1109/TNNLS.2021.3123105.

[65] Y. Hou, Z. Wu, X. Cai and T. Zhu, "The application of improved densenet algorithm in accurate image recognition," *Scientific Reports,* vol. 14, no. 1, p. 8645, 20242024.DOI: https://doi.org/10.1038/s41598-024-58421-z.

[66] M. Wu, J. Zhou, Y. Peng, S. Wang and Y. Zhang, "Deep learning for image classification: a review," in *International Conference on Medical Imaging and Computer-Aided Diagnosis*, 2023, pp. 352-362: Springer. doi: https://doi.org/10.1007/978-981-97-1335-6_31.

[67] C.-L. Zhou, L.-M. Ge, Y.-B. Guo, D.-M. Zhou and Y.-P. Cun, "A comprehensive comparison on current deep learning approaches for plant image classification," in *Journal of Physics: Conference Series*, 2021, vol. 1873, no. 1, p. 012002: IOP Publishing. doi: 10.1088/1742-6596/1873/1/012002.

[68] S. R. Dubey, S. K. Singh and B. B. Chaudhuri, "Activation functions in deep learning: A comprehensive survey and benchmark," *Neurocomputing,* vol. 503, pp. 92-108, 20222022.DOI: https://doi.org/10.1016/j.neucom.2022.06.111.

[69] R. Nirthika, S. Manivannan, A. Ramanan and R. Wang, "Pooling in convolutional neural networks for medical image analysis: a survey and an empirical study," *Neural Computing and Applications,* vol. 34, no. 7, pp. 5321-5347, 20222022.DOI: https://doi.org/10.1007/s00521-022-06953-8.

[70] C. F. G. D. Santos and J. P. Papa, "Avoiding overfitting: A survey on regularization methods for convolutional neural networks," *ACM Computing Surveys (Csur),* vol. 54, no. 10s, pp. 1-25, 20222022.DOI: https://doi.org/10.1145/351041.

[71] X. Li, Y. Chen, Y. Zhu, S. Wang, R. Zhang *et al.*, "Imagenet-e: Benchmarking neural network robustness via attribute editing," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 20371-20381.

[72] M. M. Yapıcı and N. Topaloğlu, "Performance comparison of deep learning frameworks," *Computers and Informatics,* vol. 1, no. 1, pp. 1-11, 20212021.Available: https://dergipark.org.tr/en/pub/ci/issue/60236/769457#article_cite.

[73] H. A. Al-Tameemi, G. G. Shayea, M. Al-Zubaidie, Y. L. Khaleel, M. A. Habeeb *et al.*, "A Systematic review of metaverse cybersecurity: Frameworks, challenges, and strategic approaches in a quantum-driven era," *Mesopotamian Journal of CyberSecurity,* vol. 5, no. 2, pp. 770-803, 20252025.DOI: https://doi.org/10.58496/MJCS/2025/045.

[74] S. Prasher, L. Nelson and M. Jagdish, "Potato leaf disease prediction using RMSProp, Adam and SGD optimizers," in *2023 International Conference on Advancement in Computation & Computer Technologies (InCACCT)*, 2023, pp. 343-347: IEEE. doi: 10.1109/InCACCT57535.2023.10141714.

[75] B. I. H. Nazar Salih Absulhussein, Humam K. Yaseen, Nebras H. Ghaeb, Mohamed Ksantini, "An Automated Detection and Classification of Retinopathy of Prematurity Stages Using SWIN Transformer," *Practice and Applications,* vol. 21, no. 2, pp. 228-240, 20262026.DOI: https://doi.org/10.54216/FPA.210215.

[76] M. Berrimi. (2019, 5-Dec-2024). *OCT images Balanced version (Version 3 ed.)*. Available: https://www.kaggle.com/datasets/mohamedberrimi/oct-images-balanced-version?resource=download

[77] D. S. Kermany, M. Goldbaum, W. Cai, C. C. Valentim, H. Liang *et al.*, "Identifying medical diagnoses and treatable diseases by image-based deep learning," *cell,* vol. 172, no. 5, pp. 1122-1131. e9, 20182018.DOI: 10.1016/j.cell.2018.02.010 External Link.