Eye diseases classification using Transfer Learning

Abstract— **Ophthalmic illnesses such as cataracts, glaucoma, age-related macular degeneration, and diabetic retinopathy, have a considerably negative effect on world health and cause vision loss and blindness. Effective screening and diagnosis techniques are required due to the illnesses' rising prevalence in order to enable prompt treatment and preventative measures. The objective of this project is to develop a Deep Learning model that can recognize eye diseases from retinal images of the eye using the TensorFlow and Keras libraries. This project's introduction is shown in Section I. A description of the dataset can be found in Section II. The specific processes of data preprocessing are shown in Section IIIA. The machine learning model's architecture is described by IIIB. Method for hyperparameter adjustment offered by the IIIC.**

Keywords — Ophthalmic illnesses, Eye diseases detection, CNN, DL, DenseNet-121, VGG-16, ResNet-152

I. Introduction

The eyes are essential organs that provide us the ability to see the world around us. Nevertheless, a number of eye conditions can impair our eyesight and lower our quality of life. The three most prevalent eye conditions, cataracts, diabetic retinopathy, and glaucoma, impact millions of individuals globally. Cataracts are a disorder where the eye's natural lens gets hazy, impairing vision and causing blurriness. A consequence of diabetes called diabetic retinopathy damages the blood vessels in the retina and impairs eyesight. Glaucoma is a group of eye disorders that damage the optic nerve and can cause irreversible vision loss if left untreated. For many eye illnesses to be successfully treated and vision to be preserved, early identification is crucial. Recent developments in technology and medical imaging have made it simpler to identify certain eye illnesses early on, enabling timely intervention and more favorable results. Every area of life has seen extensive usage of machine learning techniques, and medicine is no different. The monitoring method for eye illnesses described in this study may be used to detect diabetic retinopathy, cataracts, and glaucoma as well as healthy eyes. Transfer learning methods including VGG16, ResNet152, and DenseNet121 are used in the suggested system. Changes were made to the model to increase accuracy because a certified system has to have a high detection rate.The following is this paper's primary focus: 1) The causes and signs of the aforementioned three categories of eye illnesses were investigated. 2) For the aforementioned three eye disorders, a transfer learning-based eye disease discrimination system is suggested. 3) To increase the effectiveness of the proposed system, a grid search hyperparameter selection approach is suggested. The structure of this essay is as follows: The associated study, including the three disorders' etiologies, symptoms, and ocular criteria, are introduced in the second section. The suggested ocular illness detection system's ML algorithm's theoretical features are presented in the third section. The system implementation based on machine learning algorithm and integrated learning approach are introduced in detail in the fourth section, along with the data set that was employed. The outcomes of employing each approach are presented in the fifth part, which also compares how well each method performs and examines its qualities. The complete material is summed up in the sixth section.

II. Related Work

In order to diagnose eye disorders, Machine Learning and Deep Learning models have been extensively used. In the report Eye Diseases Classification Using Deep Learning published in 2021, Kulkarni, P [7]used fundus images for the screening of different eye diseases of ocular pathologies enabling efficient management of potential blinding eye diseases. The objective machine-learning classification model for glaucomatous optic discs developed in this study exposes the classificatory parameters to support clinical glaucoma therapy.

In order to identify diabetic retinopathy and diabetic macular edema in retinal pictures, Dr. Lily Peng and the Google AI team created a deep learning model utilizing convolutional neural networks (CNNs). Some state-of-the-art techniques have been utilized in this project including: Inception-v3 architecture and implemented the ensemble model by combining the predictions from various CNN models trained on an identical task. The accuracy rates for this ensemble model were considerably higher, with an AUC of 0.993 for diabetic retinopathy and 0.993 for diabetic macular edema.

III. Dataset

The dataset consisted of four directories eaching containing approximately 1,000 colorful eyeball images of people with normal eyes, cataracts, glaucoma and diabetes. The dataset is from Kaggle[4][5], and the images there are collected from various sources like IDRiD, Oculus recognition, HRF etc.

In modern medicine, eye pictures are the main way to judge eye diseases, and different eye diseases will show different lesion characteristics in the images. For example, a doctor can judge a patient, who suffers from diabetes if the eye image has “cotton wool” spots, hard exudate, abnormal growth of blood vessels and hemorrhages. All these features

will be displayed on the professional medicine eye images. This is also the main basis for machine learning judging.

IV. Methodology

A. Exploratory Analysis

The dataset contains 1038 images in cataracts directory, 1098 images in diabetes directory, 1007 images in glaucoma directory and 1074 images in normal directory. Sample images are shown below. The numbers of the four types of data are fairly even, however, the size of each image are not the same. It is necessary to change them into the same size during the preprocessing so that they can be input into the model properly.

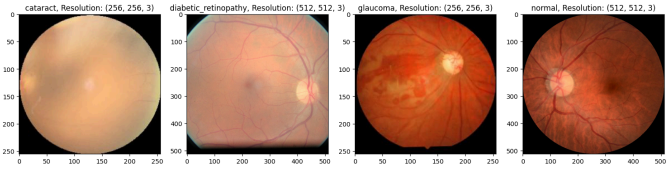

Fig. 1. Eye diseases sample picture

B. Preprocessing

First, all images are traversed and all images that do not meet JFTF standards are removed. After this process, 201 images were removed. The following is the distribution of images after the removal process.

	Labels	image_number
0	normal	1074
1	cataract	938
2	glaucoma	906
3	diabetic_retinopathy	1098

Fig. 2. Category and label images

Secondly, every image is resized to have size 224 by 224, and every pixel value is divided by 255 so the model can learn the data more efficiently.

Thirdly, normalized data[6] is frequently more effective for ML training. As a result, the range for all features having numerical values is set to 0.0 to 1.0, therefore, every pixel value is divided by 255.

Finally, a batch of size 64 is added, the machine will take 64 images at a time so that it can learn faster and more generalized.

C. Machine Learning Models

Some major science and technology organizations created transfer learning models to increase their effectiveness. This project first uses Densenet121 imported from Keras [10]as the model for training, and then uses another two transfer learning models that contain convolutional layers that good at image processing (VGG16 and Resnet152) for comparison.

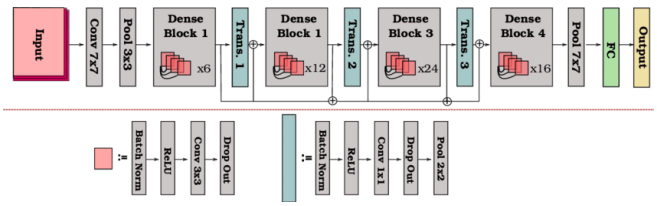

Fig. 3. Architecture of DenseNet-121

In 2016, Huang et al. unveiled the convolutional neural network architecture known as DenseNet-121. A convolutional neural network (CNN) with four dense blocks, each including many convolutional layers with tiny filter sizes, makes up the 121 layers of DenseNet-121. The transition layer after the CNN decreases the dimensionality of the feature maps. The feedforward connections between the dense blocks allow the output of each layer to be passed as input to each next layer in the block.

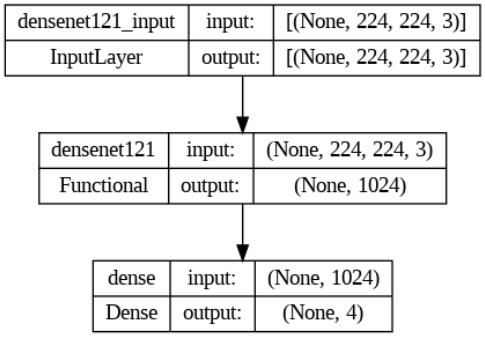

Fig. 4. Initial transformer learning model

The image above shows the architecture of the model, at input layer, the model takes images that have size 224*224*3, then feed the image into the transfer learning model, the results are then input into a dense layer which produces categorical output.

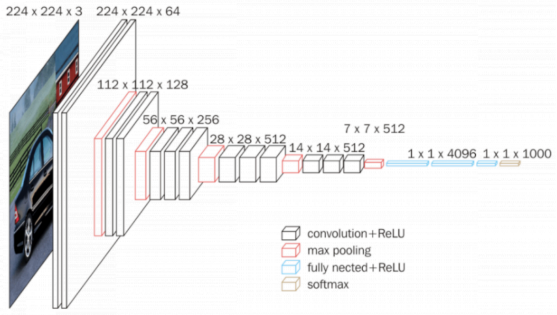

Fig. 5. Architecture of VGG-16

Simonyan and Zisserman from the Visual Geometry Group (VGG) at the University of Oxford first presented the VGG-16 convolutional neural network design in 2014. One of the most well-liked and often applied deep learning architectures for image categorization is this one. VGG-16 has 16 layers, comprising 3 fully linked layers and 13 convolutional layers. A max pooling layer with a 2x2 filter and a stride of 2 pixels is placed after the convolutional layers, which are grouped in groups of two or three and have tiny 3x3 filters and a stride of

one pixel. The convolutional layers carry out feature extraction, while the max pooling layers are utilised to minimize the spatial dimensions of the feature maps. The final categorization is carried out using the completely linked layers, which are at the network's edge. The final convolutional layer's output is flattened and fed into three fully connected layers: the first, with 4096 neurons, the second, with 4096 neurons, and the third, with 1000 units of softmax, which stands for each of the 1000 categories in the ImageNet dataset.

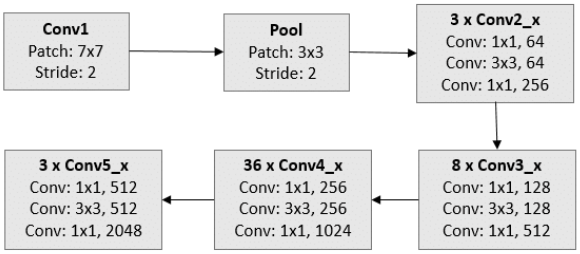

Fig. 6. architecture of ResNet-152

Kaiming He et al. first presented the ResNet-152 convolutional neural network architecture in 2015 as an advancement over the ResNet-50 design. There are 152 layers in ResNet-152, starting with a string of convolutional layers with tiny 3x3 filters and a stride of 1 pixel, then a number of residual blocks. Multiple convolutional layers with tiny filters are included in each residual block, which is followed by a shortcut connection that adds the input to the output of the block. The network can learn residual mappings and skip parts of the convolutional layers as a result of the shortcut connection. This can assist to avoid the vanishing gradient problem and allow the network to learn more detailed representations. The network's final layers are a fully connected layer, a softmax layer, and a global average pooling layer. The final convolutional layer's feature maps are averaged throughout their spatial dimensions in the global average pooling layer, which lowers the number of parameters and helps guard against overfitting.

D. Frozen Layer:

In our case, the frozen layer refers to a network architecture that fixes the weights of some layers—typically the early levels—while allowing the weights of the other layers to change during training. In transfer learning, a pre-trained model is adjusted for a new classification job using this method.

Below is the basic step of how to implementation:

1. Choose a model that has already been trained: Opt for a CNN architecture that has already been trained on a sizable dataset, such as ImageNet. Architectures that are in demand include VGG, ResNet, or Inception.
2. The dataset should be pre-processed in order to get it ready for the new classification task. This entails scaling down photos, standardizing pixel values, and maybe using data augmentation methods.
3. The final layers of the pre-trained model, such as the fully connected and softmax layers, which are particular to the initial classification task, should be removed or modified. In their place, add fresh layers made just for the new classification assignment.
4. Layers can be frozen by setting the "trainable" attribute of those layers to False. This keeps their weights from changing throughout the workout. Early layers of the network are typically frozen since they have acquired general qualities that are applicable to a variety of applications. Edges, textures, and color blobs are examples of these broader characteristics.
5. Develop the model: On your dataset, train the modified model. Only the weights of the remaining layers will be modified during training because the weights of the frozen layers are fixed. This enhances the network's performance while utilizing the knowledge that is already there in the frozen layers to perform the new categorization task.

E. Hyperparameter tuning:

Hyperparameter is added to the model, gridsearch is used to find the optimal hyperparameter of each model, and then the optimal hyperparameter of each model is brought in for further training, and the most suitable model for data is found after comparing the results.

An activation function frequently employed in neural networks and other machine learning methods is the sigmoid function, also referred to as the logistic function. The sigmoid function is particularly effective for describing probabilities because it converts any input value to a value between 0 and 1. The sigmoid function has the following mathematical equation:

$$\sigma(x) = 1 / (1 + \exp(-x)) \quad (1)$$

Here, $\sigma(x)$ denotes the output of the sigmoid function for a given input x, and $\exp(-x)$ indicates the exponential function with base e (about 2.71828 in Euler's number) raised to the power of -x.

The tanh function provides a seamless change between output values and a zero-centered output by mapping each input value to a value between -1 and 1. Comparing this to the sigmoid function, which is not zero-centered and has a range of 0 to 1, can be useful in some circumstances.

Mathematical equation for the tanh function is:

$$\tanh(x) = (\exp(x) - \exp(-x)) / (\exp(x) + \exp(-x)) \quad (2)$$

Here, the exponential functions $\exp(x)$ and $\exp(-x)$ with base e (Euler's number, roughly 2.71828) raised to the power of x and -x, respectively, describe the result of the exponential function $\tanh(x)$ for an input x.

By adding nonlinearity to the model using the ReLU function, the model is given the ability to recognize intricate patterns and connections between inputs and outputs. In comparison to other activation functions, ReLU is computationally efficient.

The mathematical equation for the ReLU function is:

$$ReLU(x) = \max(0, x) \quad (3)$$

ReLU(x) in this case denotes the result of the ReLU function given an input of x. Between 0 and the input value x, the function returns the highest possible value. In essence, the function returns x if x is positive and returns 0 if x is negative or zero.

Dropout is a regularization method that neural networks utilize to avoid overfitting and enhance model generalization. Overfitting is a condition in which a model performs well on the training set but poorly on unobserved data because it learns the noise in the training data. The risk of a neuron momentarily "dropping out" or becoming inactive during training is determined by the dropout rate, which is a hyperparameter.

The dropout rate ranges from 0 to 1, with 0 denoting no dropout (i.e., all neurons remain active during training) and 1 denoting complete dropout. (which is not useful in practice). In general, dropout rates range from 0.2 to 0.5, depending on the model's architecture and the particular issue being tackled.

In gradient-based optimization techniques, such as those used to train neural networks and other machine learning models, the learning rate is a hyperparameter. It chooses the size of the step when changing the model's parameters to reduce the loss function. While a higher learning rate can hasten convergence, it also increases the chance of overshooting the ideal solution or inducing oscillations. A smaller learning rate results in slower convergence but may also yield higher precision.

The image below shows the model architecture with hyperparameters. For the grid search setup, the options for learning rate are 0.1, 0.001and 0.0001. The options for dropout rate are 0.5 and 0.8, the options for activation function.

After running grid search for each model, the best hyperparameter is retrieved. DenseNet121{'active': 'relu', 'dropout': 0.5, 'learning_rate': 0.001}, VGG16{'active': 'relu', 'dropout': 0.8, 'learning_rate': 0.001}, ResNet152{'active': 'tanh', 'dropout': 0.5, 'learning_rate': 0.0001}. Then the hyperparameters are set for each model, then the results will be outputted and scores are compared and analyzed.

TABLE I HYPER-PARAMETER CONFIGURATION OF TRANSFER MODELS

	Active	Dropout rate	Learning rate
DenseNet121	relu	0.5	0.001
VGG16	relu	0.8	0.001
ResNet152	tanh	0.5	0.0001

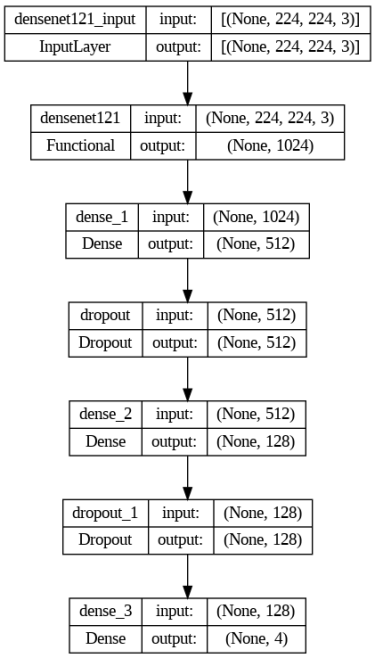

Fig. 6. CNN + Transfer learning model

V. EVALUATION SCORE RESULTS AND ANALYSIS

The following image shows the scores of our model:

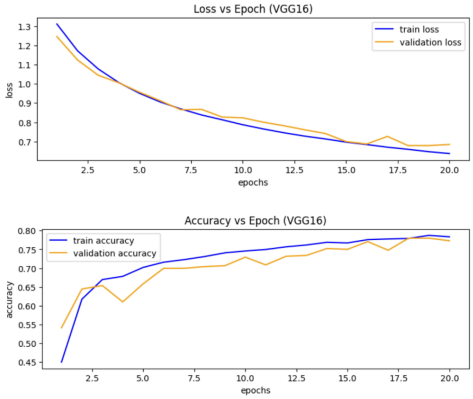

Fig. 7. Iteration/epochs vs loss, accuracy

Test accuracy: 80.60% (VGG16)

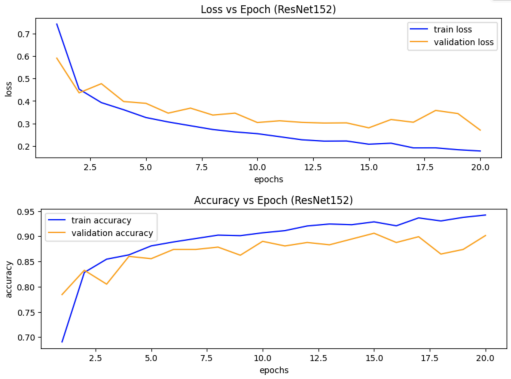

Fig. 8. Iteration/epochs vs loss, accuracy

Test accuracy: 84.64% (ResNet152)

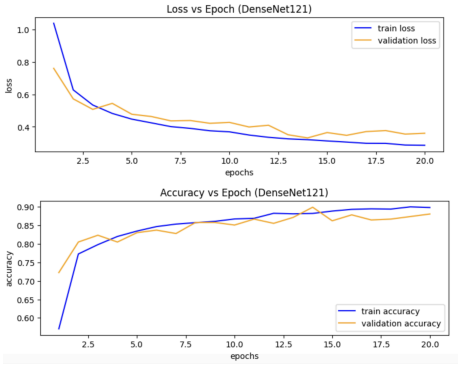

Fig. 9. Iteration/epochs vs loss, accuracy

Test accuracy: 91.28% (DenseNet121)

A. After Tuning:

Hyperparameter tuning, however Given the variety of datasets, problems, and model architectures, it is not always certain that hyperparameter tuning would enhance a particular assessment metric, such as precision.

In a particular situation, hyperparameter adjustment might reduce the precision score for a transformer model for a number of reasons:

1. Overfitting: It's possible that the tweaked hyperparameters cause the training data to become overfitted, which would make the model perform poorly on fresh, untainted data. A reduced precision rating may be the outcome of this.
2. Imbalance in the dataset: A biased model could result from a dataset with an unbalanced class distribution. In this situation, tweaking the hyperparameters could accidentally favor the majority class while degrading the precision for minority classes.
3. In the case of noisy data, hyperparameter tweaking may end up fitting the noise rather than the underlying pattern. As a result, the test set's precision score may suffer and generalization may be weak.

		Actual Class	
		Positive (**P**)	Negative (**N**)
Predicted Class	Positive (**P**)	True Positive (**TP**)	False Positive (**FP**)
	Negative (**N**)	False Negative (**FN**)	True Negative (**TN**)

Fig. 10. Confusion Matrix

Precision score / Recall score / F1 score explanation:

Positive: It is proven that the instance belongs to the class that the classifier is attempting to identify.

Negative: It is determined that the instance does not correspond to the class we are trying to identify.

True Positive: TP | False Positive: FP

False Negative: FN | True Negative: TN

Precision is an indicator of the number of positive forecasts made are correct (TP). Precision = (TP) / (Total Predict Positive) Precision refers to how accurate/precise your model is in terms of how many of the positive outcomes that were predicted turned out to be positive.

The model's recall score gauges how well it can identify positive samples. When recall is high, the model is less likely to anticipate negative outcomes incorrectly. The recall formula is:

Recall = (TP) / (Total Actual Positive)

F1 Score is needed when you want to seek a balance between Precision and Recall. Utilizing the two metrics, it offers a balanced assessment of precision and recall. The F1 score is determined by:

F1_score = 2 * ((Precision * Recall) / (Precision+Recall))

Using these three assessment standards[9], in particular for binary classification problems where unbalanced datasets are common, allows for a more full understanding of the effectiveness of the model.

TABLE II PRECISION, RECALL AND F1 SCORES COMPARISON

	DenceNet121	VGG16	ResNet152
Precision	0.901	0.853	0.393
Recall	0.898	0.853	0.448
F1-score	0.899	0.850	0.387
Accuracy	0.9128	0.8060	0.8464

VI. NEXT STEPS

A. Model optimization:

Talk about prospective changes to the model's architecture, such as experimenting with various transfer learning models that have already been trained, changing the number of layers or neurons, or tweaking the model's hyperparameters. Mention how to tune hyperparameters using methods like grid search, random search, or Bayesian optimization.

B. Data augmentation:

Describe the methods that could be utilized to enhance the training dataset's diversity and the extrapolated abilities of the model, such as scaling, rotating, flipping, and color modifications.

C. Combining further details:

Discuss the advantages of training the model on larger or more varied datasets, such as combining images from various sources, capturing variations in imaging technology and patient populations, or classifying images of various eye diseases.

D. Model interpretability:

Talk about how crucial it is to comprehend how the model decides to act, and propose using tools like Grad-CAM, LIME, or SHAP to display and analyze the model's forecasts.

E. Clinical integration:

Describe the procedures needed to include the created model into clinical workflows, including establishing an easy-to-use user interface, establishing a safe and flexible platform for model deployment, and taking care of any privacy and legal problems.

F. Continuous model improvement:

Mention the necessity of updating the model with fresh data, continuously observing how the model performs in practical contexts, and resolving any biases or restrictions as they manifest.

Conclusion

Due to the complexity of the eye, the wide range of illnesses with similar symptoms, and the frequent occurrence of subtle and gradual changes, diagnosing eye disorders can be difficult.

In this paper, we introduce several transfer learning methods. DenceNet121, VGG16 and ResNet152 build our base model. Freeze the previous train ability, keep the original fixed weight, and enhance the learning ability of small data sets. In the initial step, all performance is based on the entire transfer model, and the output layer is classified with four features to explore the powerful capabilities of the transfer learning model. Afterwards, hyperparameter tuning is added to improve the accuracy of each model. Based on the characteristics of each model, we tune the activation function, dropout rate, and learning rate to find specific parameters suitable for each model. To accommodate these adjustments, three hidden layers are added at the end of the model. After the hyperparameters were implemented, the accuracy of all models increased by about 5-7%. However, it is not always certain that hyperparameter tuning enhances specific evaluation metrics, such as accuracy. In our model, both DenceNet121 and VGG16 have slight increases, but ResNet152 contains specific overfitting. All of this is also shown on Precision, Recall and F1 scores.

References

[1]. Neurohive, "VGG16 architecture," in Neurohive, published on August 13, 2019, https://neurohive.io/en/popular-networks/vgg16/

[2]. A schematic illustration of the DenseNet-121 architecture. Accessed: Apr. 07, 2023. [Online]. Available: https://www.researchgate.net/figure/A-schematic-illustration-of-the-DenseNet-121-architecture-82_fig5_334170752

[3]. K. He, X. Zhang, S. Ren, and J. Sun, "The basic architecture of Resnet152," in ResearchGate, 2015. [Online]. Available: https://www.researchgate.net/figure/The-basic-architecture-of-Resnet152_fig2_324961229

[4]. Alcatere, "Eye diseases CNN," in Kaggle, 2019. [Online]. Available: https://www.kaggle.com/code/alcatere/eye-deseases-cnn

[5]. Gunavenkata D., "Eye diseases classification," in Kaggle, 2021. [Online]. Available: https://www.kaggle.com/datasets/gunavenkatdoddi/eye-diseases-classificat

[6]. K. M. Ali Alheeti and K. Mc Donald-Maier, "Intelligent intrusion detection in external communication systems for autonomous vehicles," Syst. Sci. Control Eng., vol. 6, no. 1, pp. 48–56, 2018.

[7]. Kulkarni, P.,(2021). Eye disease classification using deep learning. 2021 IEEE 2nd International Conference on Intelligent Sustainable Systems (ICISS), 38-41. https://iovs.arvojournals.org/article.aspx?articleid=2776689

[8]. Mavroudi, E., Koutsouri, T., Giakoumidis, A., & Tsioukas, V. (2021). Evaluation of different machine learning algorithms for classification of eye diseases. 2021 6th International Conference on Advanced Information Systems and Engineering (ICAISE), 100-104. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6029465/

[9]. F. Salo, M. Injadat, A. B. Nassif, A. Shami, and A. Essex, "Data mining techniques in intrusion detection systems: A systematic literature review," *IEEE Access*, vol. 6, pp. 56046–56058, 2018.

[10]. G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected Convolutional Networks," *arXiv.org*, 28-Jan-2018. [Online]. Available: https://arxiv.org/abs/1608.06993. [Accessed: 12-Apr-2023].