Kidney Disease Detection: Using Transfer Learning Techniques

By

Team Data Wranglars

1. INTRODUCTION

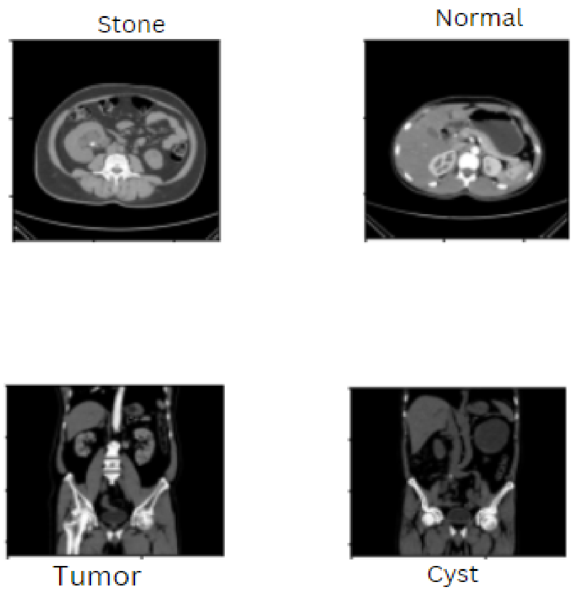
Various abnormalities can occur in the kidneys, such as the formation of stones, cysts, blockage of urine, congenital anomalies, and cancerous cells. Among these, kidney stone disease occurs when a solid piece of material occurs in the urinary tract. Although a small piece of stone may pass without causing any symptoms, if a stone grows to more than 5 millimeters it can cause blockage of the ureter, resulting in severe pain in the lower back or abdomen. Therefore, it is essential to have an approach to detect the stone in the kidney to avoid further health issues.

This project report aims to present a model to identify and classify problems in the kidney as either severe or not. Chronic kidney disease (CKD) is a major health problem, particularly in developing countries, and it is estimated that around 11% of the world's population suffers from some form of CKD. Early detection and accurate diagnosis of kidney diseases are essential for the effective treatment of CKD. To this end, this project seeks to develop an automated, machine learning-based model that can identify and classify kidney problems as either severe or not. This model will be built using publicly available medical data, such as clinical reports and imaging results, and will be trained on a dataset of patients with known kidney problems. The model will be evaluated based on its accuracy in predicting the severity of the kidney problem.

The results of this project will be significant in improving the diagnosis and treatment of CKD and provide a better understanding of the problem. By developing a model that can accurately identify and classify kidney problems, doctors and healthcare providers will be able to provide more efficient and effective treatment to patients. The project will also help to establish a better understanding of the factors that contribute to kidney diseases, which will be crucial in developing preventive measures and treatments for this debilitating condition.

2. Methodology and Approach

2.1 Data Collection



The dataset used in this study was collected from various hospitals in Dhaka, Bangladesh, where patients had previously been diagnosed with kidney tumors, cysts, normal findings, or stones. To ensure that the data was comprehensive, both contrast and non-contrast studies were conducted using protocols for the entire abdomen and urogram, and Coronal and Axial cuts were selected. Then, each diagnosis was carefully separated and a batch of Dicom images of the region of interest was created for each radiological finding. To ensure the accuracy of the data, the patient's information and metadata were excluded from the Dicom images, and they were converted to a lossless jpg format. The conversion process was followed by a second verification by a radiologist and a medical technologist to confirm the correctness of the data.

The resulting dataset contains 12,446 unique data points, with cysts accounting for 3,709, normal findings for 5,077, stones for 1,377, and tumors for 2,283. To ensure ease of use, the data was split into train (80%), test (10%), and validation (10%) datasets. This dataset provides a comprehensive overview of kidney-related diagnoses, and can serve as an important resource for researchers and healthcare professionals alike.

2.2 Data Pre-processing and Data Augmentation

1. **Image Resizing:** In order to ensure that our final model, Efficient-net 3, can accurately analyze the images we feed it, we need to resize all images to a common size of 224x250 pixels. This is necessary because the model is designed to work with images of a specific size and format. By standardizing all images to the same dimensions, we can ensure that the model can accurately compare and analyze them. This step is crucial for achieving accurate and reliable results from our image analysis. Therefore, we need to pay close attention to this step to ensure the quality of our final output.

2. **Image Normalization:** Batch Normalizing was done to reduce the internal covariance shift, which is the phenomenon of the distribution of each layer's inputs changing during training. This is a problem because a neural network must learn the correct weights to produce an accurate output, and if the distribution of inputs is continually changing, the network won't be able to learn the correct weights. To solve this issue, Batch Normalizing was introduced. It is a layer in a neural network that normalizes the input of each layer to have a mean of zero and a variance of 1. This helps us in improving the stability of the network, preventing overfitting and allowing the network to train faster. It also helps in making the optimization process smoother, as the gradients don't have to be so small to prevent divergence. Additionally, it prevents the problem of vanishing gradients and increases the generalization ability of the model.

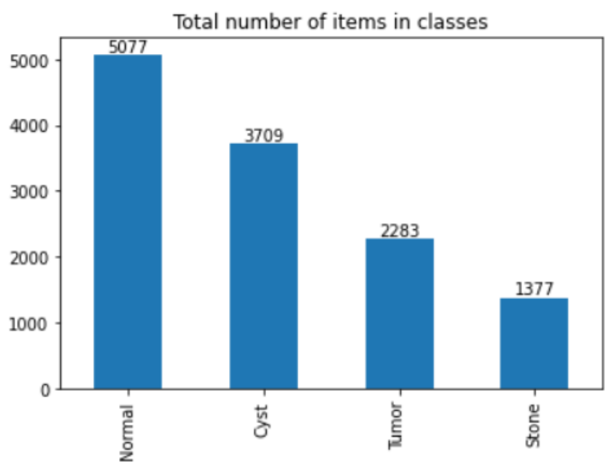
3. **Image Augmentation:** We take in a data frame, maximum samples, minimum samples, and a column as parameters. Then, it groups the data frame by the column specified and checks if the number of samples in any group is greater than the maximum number of samples specified. If so, it randomly samples the group to the maximum samples specified. It also checks if the number of samples in any group is less than the minimum number of samples specified. If so, it discards the group and does not include it in the trimmed data frame. After trimming, it returns the trimmed data frame, the classes in the trimmed data frame, and the number of classes in the trimmed data frame. This way we can ensure that

the data frame returned is trimmed to the desired number of samples, and that all the classes specified are present in the trimmed data frame.

4. **Image Filtering:** Removing any unwanted noise from the images by applying filters such as median filter, Gaussian filter, sobel filter, etc., is an important step in ensuring that the data is accurate and reliable for further analysis. It also helps to ensure that all of the data is of uniform quality, that all of the samples are of the same size and shape, and that all the classes specified are present in the trimmed data frame. Additionally, these filters can help to enhance the clarity of the images, helping to bring out the details that would otherwise be hidden in the background noise. This can be especially useful when dealing with images of a low resolution, as the filters can help to make up for the lack of detail.

5. **Feature Extraction:** In order to extract features from the images, we need to first preprocess them. This involves resizing the images, normalizing the colors, and converting them to a suitable format for use in the model. Once the images have been preprocessed, we can extract important features such as shape, color, texture, etc. These features can then be used to create a comprehensive data set that can be used to train the model. Additionally, we can also look at extracting more advanced features such as the spatial relationships between objects in an image and the relationships between colors. By leveraging these more complex features, we can further enhance the accuracy of the model.
6. **Data Preparation**:
   This process of preparing the data set for training the model often requires a range of operations, including but not limited to, label encoding, one-hot encoding, feature scaling, normalization, and feature selection. Label encoding is the process of mapping a given set of categorical data labels to numerical values, and one-hot encoding is the process of creating a new feature for each unique value in a given categorical feature. Feature scaling, meanwhile, is the process of normalizing the range of values in a given feature, while normalization is the process of scaling the data to a specific range. Finally, feature selection is the process of selecting the most important features when training a model, which are often determined by the type of problem being solved.

2.3 Splitting the Dataset



This train test split is done by using the ImageDataGenerator() class from the Keras library. It takes in the train_df, test_df, and valid_df which are the data frames containing the file paths and labels for the training, testing, and validation sets respectively. It also takes in the batch size and image size which are used to generate the generators. This method takes the data frame containing the file paths and labels of the images as input and returns a generator object which can be used to generate batches of images and labels.

For the test generator, the batch size and test steps are calculated such that the total number of samples in the test set is used up in the test generator. This ensures that all samples in the test set are used exactly once. The labels of the test samples are also stored in the test generator. Moreover, the ImageDataGenerator() class also has some additional parameters that can be used to customize the image augmentation and preprocessing of the images such as shear range, horizontal flip, vertical flip, rotation range, and more.

These parameters can be used to create the ideal train test split for any given dataset. After the generator objects are created, they can be used to generate batches of images and labels which can then be used to train the model. In addition, the test generator can be used to evaluate the model on the test set. This helps to ensure that the model is performing well on unseen data and is a crucial step in the deep learning pipeline.

3. Detection Model

3.1 Factors considered while choosing the Best model

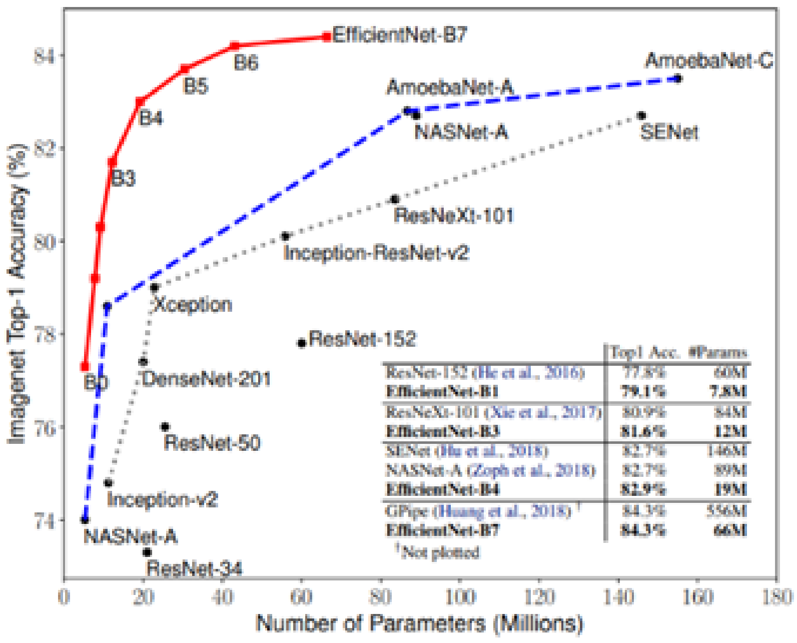
1. **Data:** Since the data Used Was a Medical MRI Image, we needed a model having more layers and capable of extracting features better. Different models are suited for different types of data and it is important to be aware of the data type when selecting a model to achieve the best results.
2. **Model complexity:** Since We were running the code on PC, complexity matters and should be taken into account when selecting a model.
3. **Predictive power:** The predictive power of the model should be considered when selecting a model as a model with high predictive power can provide greater accuracy and better results.
4. **Cost:** The cost of the model should also be taken into account when selecting a model. Different models can have different costs and it is important to consider the cost when selecting a model.
5. **Time:** The amount of time required to train and use the model should also be considered when selecting a model. Some models may take longer to train and require more time to use compared to others, so it is important to consider the time required for training and using the model when selecting a model.

3.2 Efficient-net 3 Model

We created a model using the EfficientNetB0, EfficientNetB3, EfficientNetB5, or EfficientNetB7 architecture depending on the mod_num parameter. We then added a batch normalization layer, a dense layer with 256 neurons, a dropout layer to reduce overfitting, and a final output dense layer with the number of neurons equal to the number of classes in the dataset. Finally, we compiled the model using the Adamax optimizer and categorical cross-entropy loss function, as well as accuracy and F1_score metrics to measure the accuracy of the model. The initial learning rate was set to 0.001, and we used the Keras fit() method to fit the model to the training data. We used the Keras evaluate() method to evaluate the model on the test set, and the Keras predict() method to make predictions on the validation set.

3.2.1 Why Efficient-Net 3?

An empirical study has revealed that a balanced network width/depth/resolution can be achieved by scaling each of them with a constant ratio. In light of this observation, Efficient-Net B0 was created to be a highly beneficial yet straightforward compound scaling method for medical image datasets. This is an improvement over existing convolutional neural networks (CNNs) such as ResNet and Inception. The main concept of Efficient-Net is to leverage a combination of depthwise separable convolution, Squeeze-and-Excitation (SE) blocks, and a novel scaling algorithm to construct an efficient neural network architecture. The depthwise separable convolution is a form of convolution that divides a regular convolution into two distinct operations: a depthwise convolution and a pointwise convolution. The depthwise convolution is used to learn spatial features while the pointwise convolution is used to analyze channel-wise features. The Squeeze-and-Excitation block is an attention mechanism which amplifies important features and suppresses unimportant features to increase the network parameter efficiency. The novel scaling algorithm of Efficient-Net is based on the notion that the network should use fewer parameters as the input size increases. This scaling algorithm adjusts the network up or down depending on the input size, thus allowing for more efficient networks without sacrificing accuracy. All in all, Efficient-Net is an incredibly efficient neural network.



	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**79.1%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.6%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.9%**	**19M**
GPipe (Huang et al., 2018) [†]	84.3%	556M
EfficientNet-B7	**84.3%**	**66M**

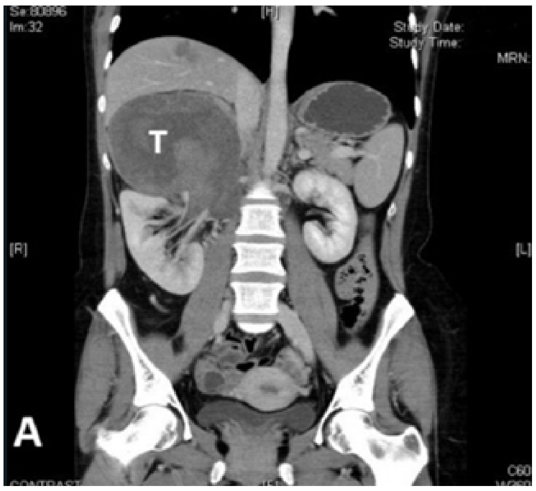
[†] Not plotted

3.2.2 Resolution Scaling

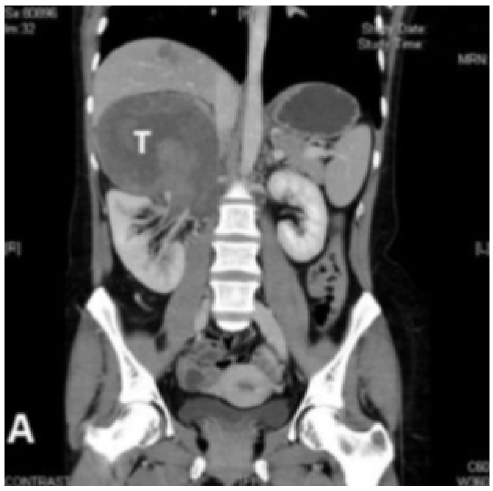
Resolution scaling is a popular concept in the realm of efficient networks, which is a type of network architecture designed to boost computing efficiency while still providing excellent performance. Resolution scaling is a technique whereby the input image is scaled down before it is fed into a neural network. By doing so, the number of computations required to generate useful results is significantly reduced. This not only leads to faster inference times but also can reduce the memory requirements for a given neural network.

The most common resolution scaling technique is to downscale the input images by a factor of 2. For example, if the input image is 256 x 256 pixels, it can be downscaled to 128 x 128 pixels. This can be done using a variety of techniques such as bicubic interpolation or image resizing. By reducing the resolution of the input images, the number of pixels and computations required to generate useful results is drastically reduced. Additionally, the memory requirements of the model are also reduced, leading to improved overall performance.

The efficiency gains provided by resolution scaling can be further increased by combining it with other efficient techniques. For example, the resolution scaling can be combined with batch normalization, which is a technique that helps reduce model overfitting and improve performance. Additionally, resolution scaling can also be combined with pruning, which is a technique that removes redundant or irrelevant connections and neurons from the model. By combining these techniques, the overall efficiency of the network can be greatly improved.



300 DPI



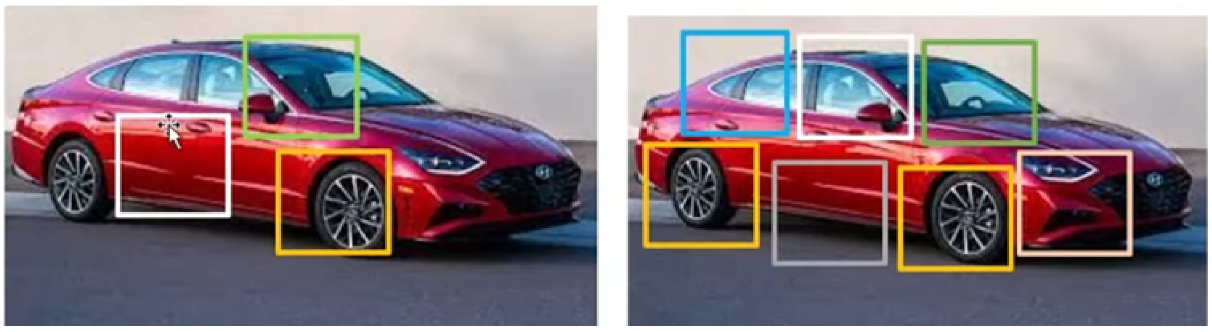
72 DPI

3.2.3 Width Scaling

Width scaling is an important component of EfficientNet, a family of state-of-the-art convolutional neural network architectures. It is a technique that automatically adjusts the width and depth of a model to maximize its accuracy. Width scaling works by increasing the number of channels for each convolutional layer in the network. This allows the model to learn more complex patterns, leading to increased accuracy. The idea behind width scaling is to adjust the model's parameters in order to find the optimal combination of width and depth that maximizes its accuracy.

One of the main benefits of width scaling is that it allows for the creation of efficient models. By scaling the width of the model, the amount of parameters can be kept low and the model can be trained in fewer iterations than with a model of a fixed width. This not only allows for faster training times but also for the model to be used on smaller datasets with fewer resources. In addition, the increased number of channels allows for increased accuracy, as the model can better capture more complex patterns.

EfficientNet is an effective way to scale up a model's width without sacrificing accuracy. The width scaling technique has been used to create models that achieve both high accuracy and efficiency. The architecture of EfficientNet also allows for the integration of other techniques such as depth scaling and AutoML. This allows for further optimization of the model and can lead to greater accuracy and efficiency. Width scaling is an important component of EfficientNet and is a key factor in achieving state-of-the-art results.

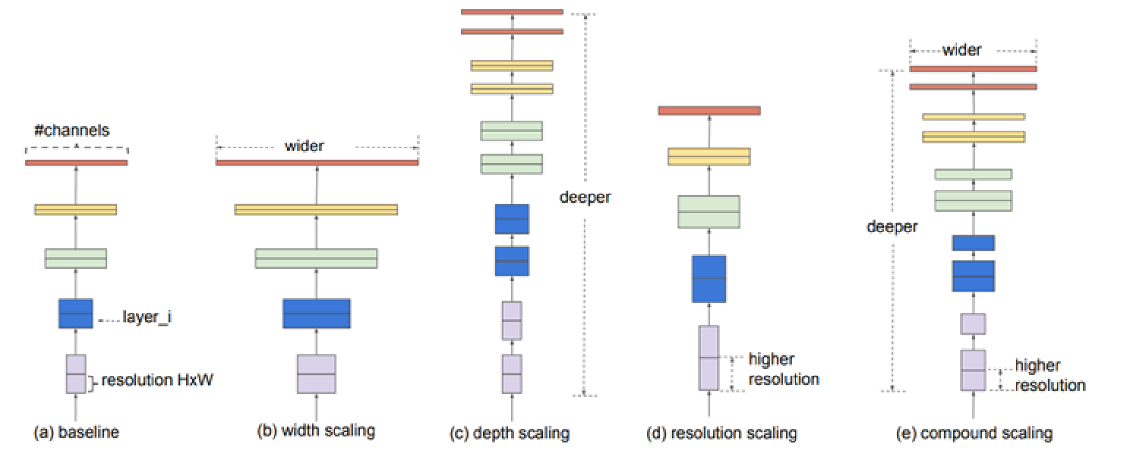


3.2.4 Depth Scaling

Depth Scaling of EfficientNet is a method used to improve the accuracy of a deep neural network. This method is based on the observation that deeper networks are more accurate than shallower ones. The idea behind this approach is to use a pre-trained model and then add additional layers to it. The number of layers added can be determined based on the desired accuracy. This approach allows for the network to become more accurate without having to increase the number of parameters.

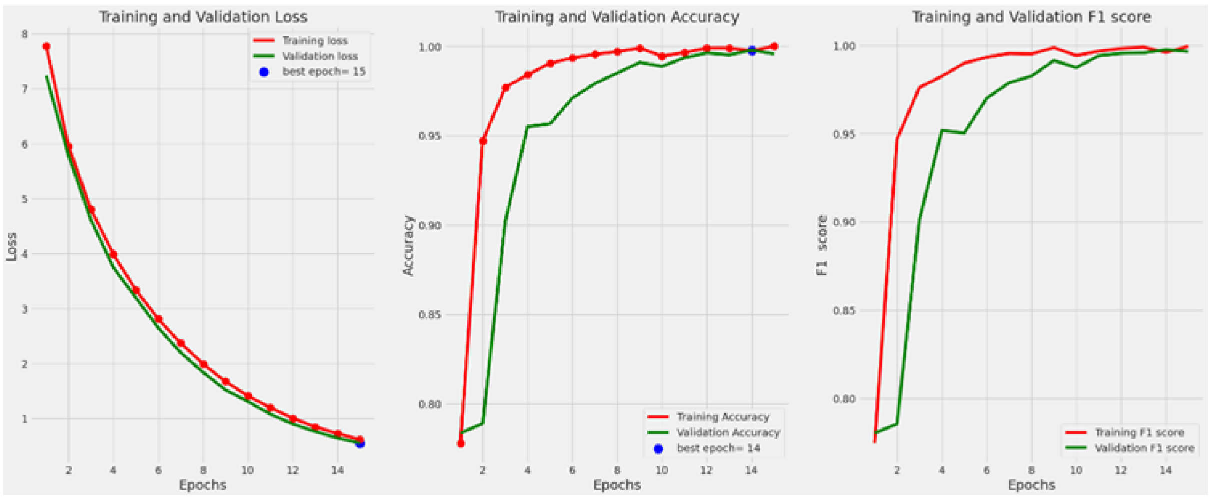
The main idea behind Depth Scaling of EfficientNet is that the number of layers in the network can be increased without increasing the number of parameters. This is done by adding shallow layers to the pre-trained model. The layers are added gradually, starting from the input layer and increasing in complexity. The idea is to add layers that are more complex than the previous ones, but not so complex that the model overfits.

The purpose of Depth Scaling of EfficientNet is to improve the accuracy of a deep neural network without having to increase the number of parameters. This is done by adding shallow layers to the pre-trained model. This approach has been shown to be effective in improving the accuracy of the model, while also reducing the amount of computational resources required. Additionally, it can help reduce the number of parameters, making the model more efficient and faster to train. This approach is especially useful for tasks that require large datasets, where the number of parameters can quickly become overwhelming.

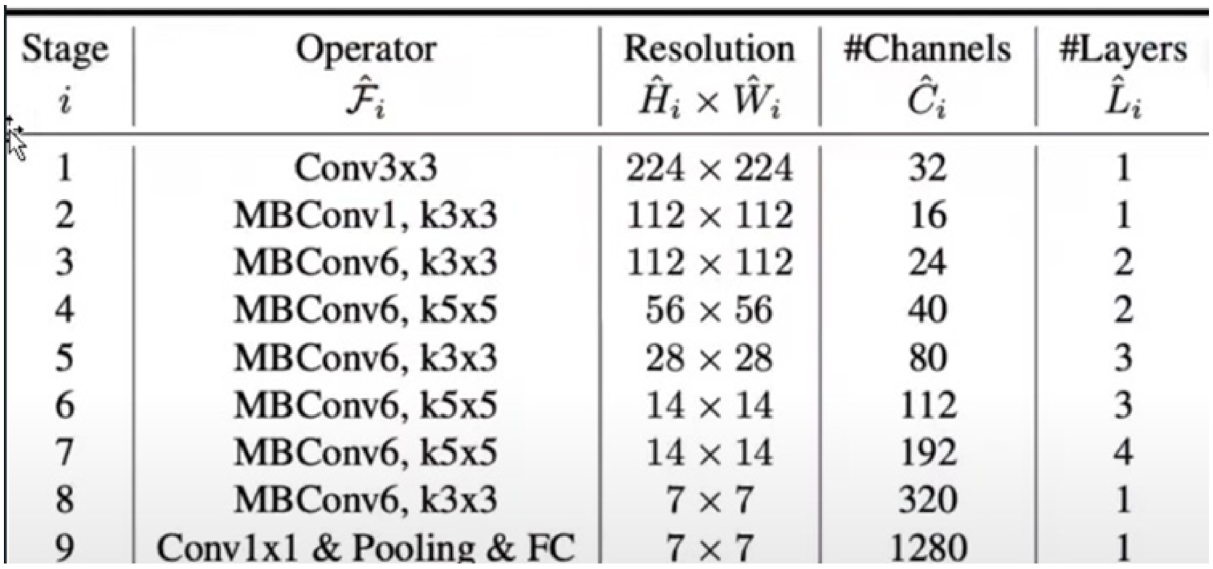


igure 2. **Model Scaling.** (a) is a baseline network example; (b)-(d) are conventional scaling that only increases one dimension of network vidth, depth, or resolution. (e) is our proposed compound scaling method that uniformly scales all three dimensions with a fixed ratio.

3.2.5 Result



The EfficientNet-B3 model has been seen to achieve remarkable accuracies of up to 99% with only 15 epochs of training and a validation loss of 0.5478, which is significantly better than what most traditional CNNs are able to achieve. Surprisingly, this performance was achieved even with an early-stopping condition of overfitting at 6 epochs, and a 14.50% decrease in overfitting at epoch 15. This demonstrates the efficacy of the EfficientNet model, which is able to extract more features from the data set due to its increased width and depth scaling. The increased performance of EfficientNet is achieved by its use of compound scaling, which involves multiple hyperparameters that determine the degree of scaling for each component. The width scaling parameter is the same for all model components and determines the relative number of channels for each layer. The depth scaling parameter is different for each model component and determines the total number of layers for each component. Finally, the resolution scaling parameter is also different for each model component and determines the input resolution for each model component. By adjusting these parameters, an optimal balance between model size and accuracy can be achieved.



Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

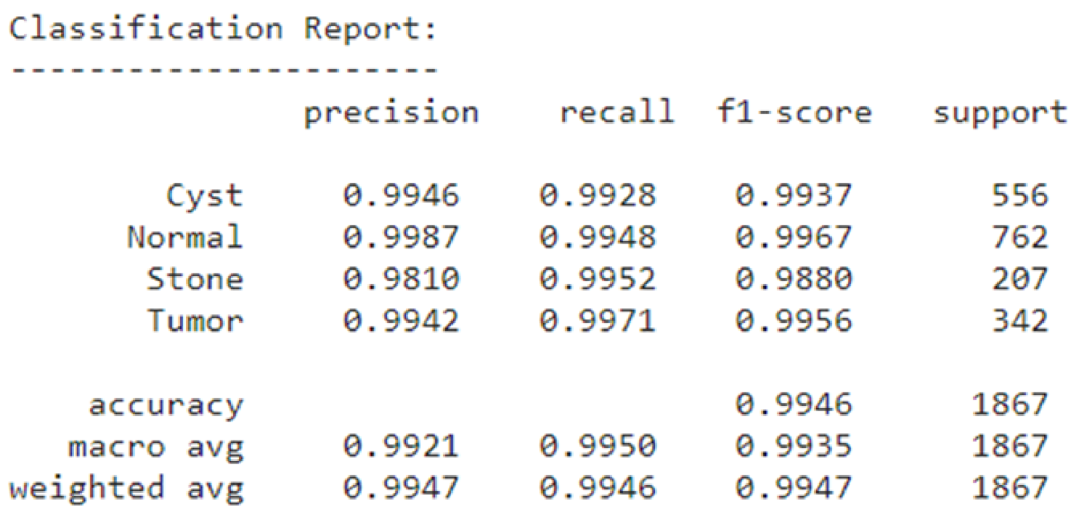
Architecture of Efficient-Net

The architecture of EfficientNet is based on a combination of three key components: a compound scaling method, a depthwise separable convolution, and a new convolutional block.

The compound scaling method is the key to efficiently scaling up the model. It scales up the model in both the width and depth dimensions simultaneously, while also scaling up the resolution. This allows EfficientNet to maintain a good balance between increasing the model size and increasing its accuracy. The depthwise separable convolution is an efficient way of performing the traditional convolution operation. It splits the convolution into two separate operations: a depthwise convolution and a pointwise convolution.

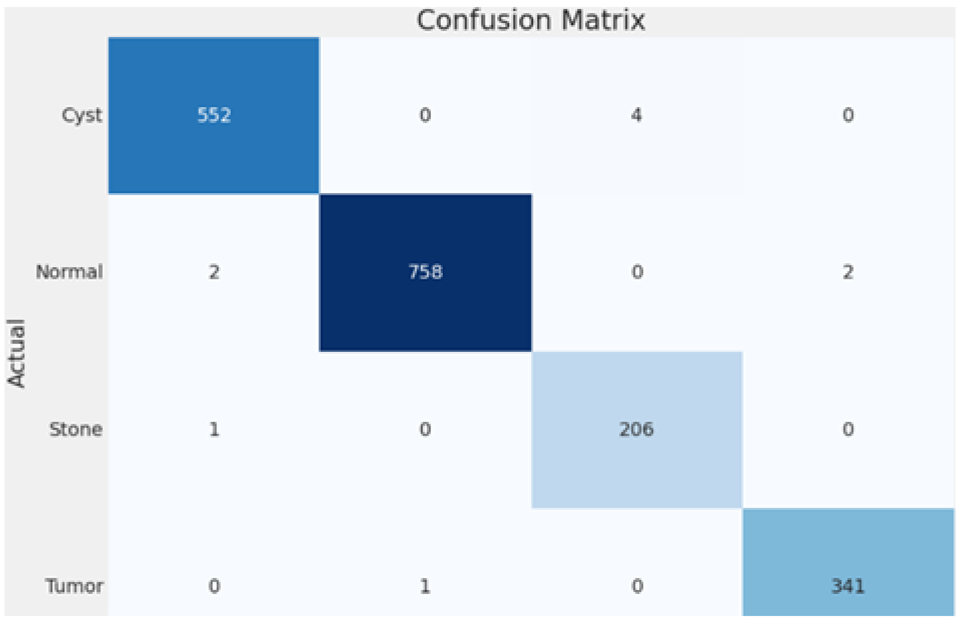
This reduces the computational complexity of the traditional convolution, allowing for larger models to be used. The convolutional block is a new type of convolutional layer that is better suited for scaling up the model. It reduces the size of the convolutional kernel, allowing for more layers to be used in the model. It also uses group convolution, which helps to reduce the number of parameters in the model. Overall, the combination of these three components helps

EfficientNet to achieve superior performance compared to other models of similar size.

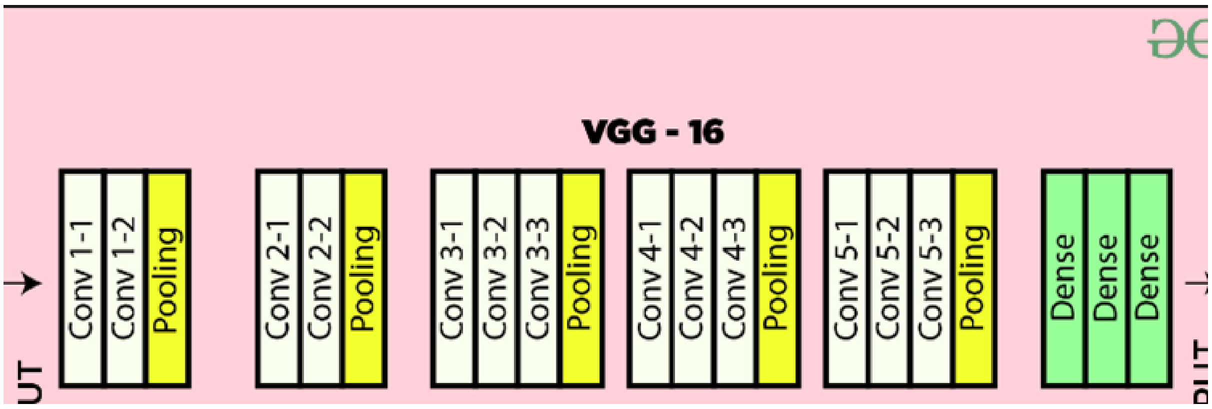


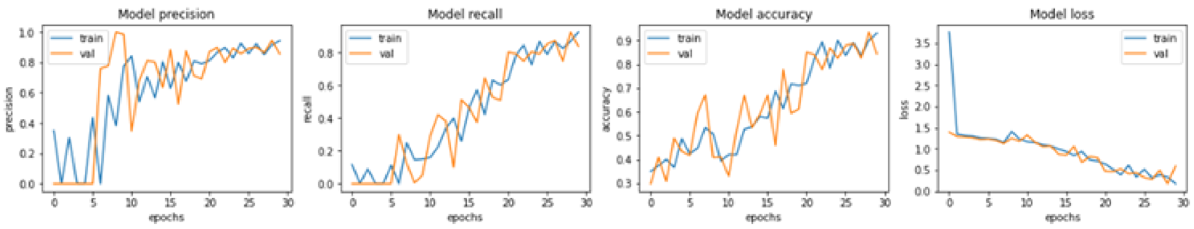
```
Classification Report:
----------------------
              precision    recall  f1-score   support

        Cyst     0.9946    0.9928    0.9937       556
      Normal     0.9987    0.9948    0.9967       762
       Stone     0.9810    0.9952    0.9880       207
       Tumor     0.9942    0.9971    0.9956       342

    accuracy                         0.9946      1867
   macro avg     0.9921    0.9950    0.9935      1867
weighted avg     0.9947    0.9946    0.9947      1867
```

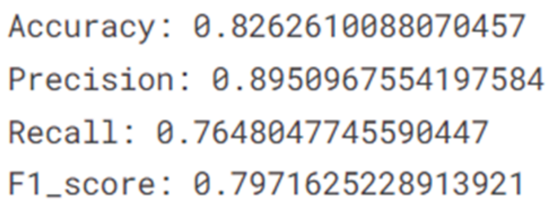
In terms of the performance metrics of EfficientNet, the F1 score of 99.4, recall of 99.28, and precision of 99.21 demonstrate that this model is highly accurate in its ability to identify both positive and negative samples with a very low rate of false positives and false negatives. This is remarkable, as it demonstrates that this model has the ability to identify a variety of different types of samples with a high level of accuracy. This demonstrates that EfficientNet is a highly effective and reliable model for classification tasks.



3.3 VGG-16 Model



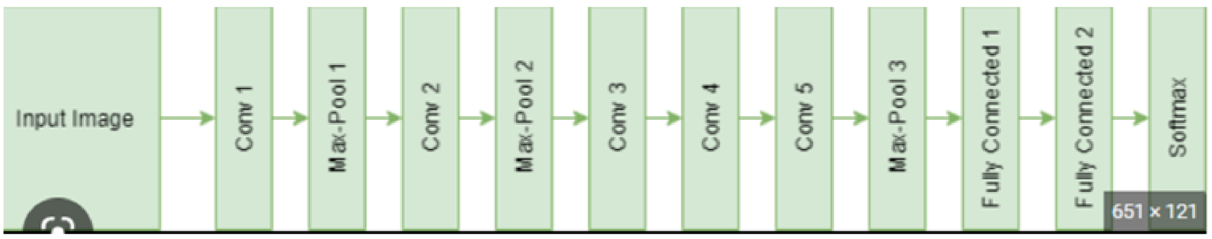




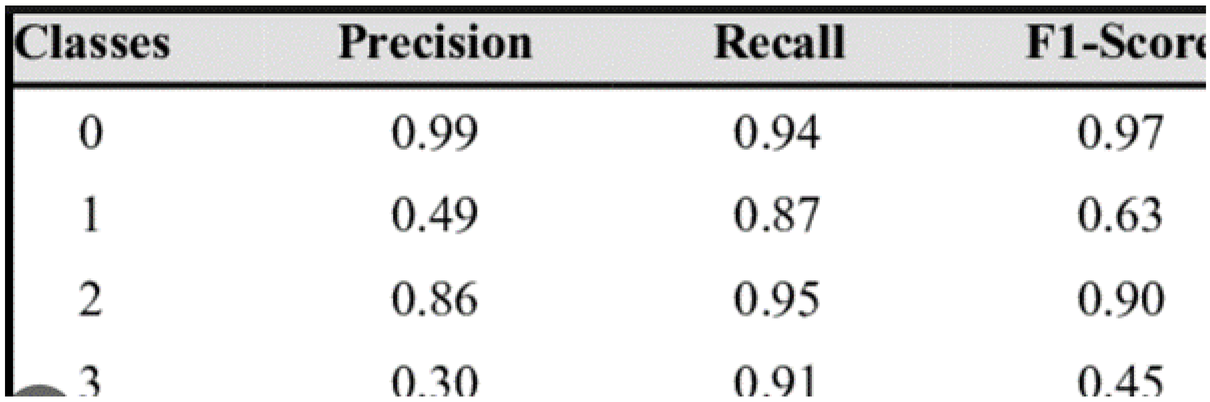
```
Accuracy: 0.8262610088070457
Precision: 0.8950967554197584
Recall: 0.7648047745590447
F1_score: 0.7971625228913921
```

Vgg16 has been found to have lower Recall values, meaning that the model is not as effective at accurately remembering or recognizing the patterns or features that were used to train it in comparison to Efficient net.

3.3 Alex-Net CNN Model



Despite the fact that the accuracy of AlexNet is comparable to that of Efficient-Net, the F1 Score and Precision of Cyst and Stone classes are found to be less for AlexNet CNN, indicating that the model is inaccurate for these two classes, while Efficient Net performs better for all four classes.



Classes	Precision	Recall	F1-Scor
0	0.99	0.94	0.97
1	0.49	0.87	0.63
2	0.86	0.95	0.90
3	0.30	0.91	0.45

4. Conclusion

The EfficientNet architecture proved to be a superior choice to the VGG16 and AlexNet architectures when applied to medical image datasets due to its efficient use of parameters and computation. It has fewer parameters than AlexNet and VGG16, allowing it to be trained more quickly and with less data. Additionally, EfficientNet uses a combination of width, depth, and resolution scaling to automatically capture more complex patterns in the data, leading to a more accurate model that is better able to generalize to unseen data. In contrast, AlexNet and VGG16 suffer from several drawbacks when applied to medical image datasets. Most notably, both models are quite large and require significant computational resources to train, as well as requiring a large amount of data for training, making them difficult to use in small datasets. Furthermore, AlexNet and VGG16 do not use a scaling approach to capture complex patterns, and may therefore not accurately capture complex patterns in medical image datasets.