**Kidney Disease Detection: Using Transfer Learning Techniques**

By

**Team Data Wranglars**

**1. INTRODUCTION**

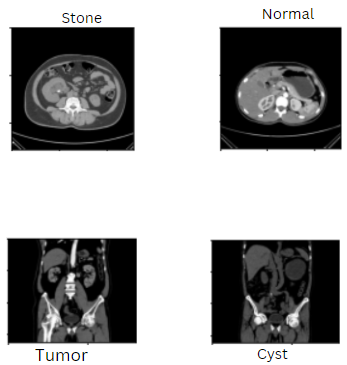
Various abnormalities can occur in the kidneys, such as the formation of stones, cysts, blockage of urine, congenital anomalies, and cancerous cells. Among these, kidney stone disease occurs when a solid piece of material occurs in the urinary tract. Although a small piece of stone may pass without causing 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**



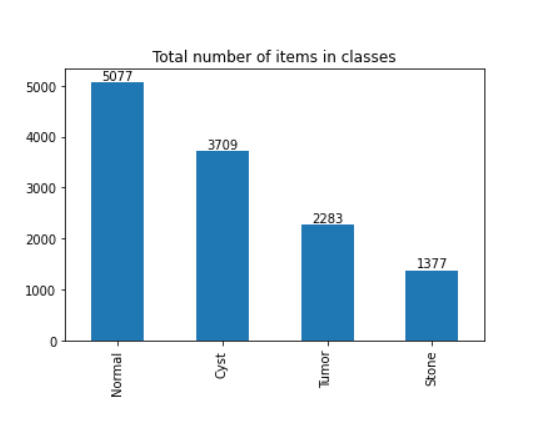
The dataset used in this study was collected from various hospitals in Dhaka, Bangladesh, where patients had previously been diagnosed with kidney tumors, cysts, normal findings, or stones. To ensure that the data was comprehensive, both contrast and non-contrast studies were conducted using protocols 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 Spliting 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 dataframes containing the filepaths 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 dataframe containing the filepaths 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 crossentropy 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.