

CHRONIC KIDNEY DISEASE DETECTION USING DEEP LEARNING

A Project Report Phase - II

Submitted in partial fulfillment of the requirements for the award of Bachelor of
Engineering degree in Electronics and Communication Engineering

by

KEERTI G. (39130223)

MANI MOZHI R. (39130280)



DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

SCHOOL OF ELECTRICAL AND ELECTRONICS

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

APRIL - 2023



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with "A" grade by NAAC
Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119
www.sathyabama.ac.in

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of Keerti G. (39130223) and Mani Mozhi R. (39130280) who carried out the project entitled "Chronic Kidney Disease Detection Using Deep Learning" under our supervision from November 2022 to April 2023.

Dr. N. M. NANDHITHA, M.E, Ph.D.

Internal Guide

Dr. T. RAVI, M.E., Ph.D.,
Head of the Department

25.04.2023

Submitted for Viva voce Examination held on _____

Internal Examiner

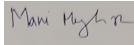
External Examiner

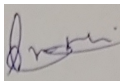
DECLARATION

We KEERTI G. (39130223), MANI MOZHI R. (39130280) hereby declare that the Project Report entitled “**Chronic Kidney Disease Detection using Deep learning**” done by us under the guidance of Dr. N. M. Nandhitha, M.E., Ph.D. is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Electronics and Communication Engineering.

DATE: 25.04.2023

PLACE: Chennai

1. 

2. 

SIGNATURE OF THE CANDIDATES

ACKNOWLEDGEMENT

We are pleased to acknowledge our sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. We are grateful to them.

We convey our thanks to **Dr. N. M. NANDHITHA, M.E., Ph.D., Professor & Dean, School of Electrical and Electronics** and **Dr. T. RAVI, Ph.D., Professor & Head, Department of Electronics and Communication Engineering** for providing us necessary support and details at the right time during the progressive reviews.

We would like to express our sincere and deep sense of gratitude to our **Project Guide Dr. N. M. NANDHITHA M.E., Ph.D.**, for her valuable guidance, suggestions and constant encouragement which paved way for the successful completion of our project work.

We wish to express our thanks to all Teaching and Non-Teaching staff members of the Department of Electronics and Communication Engineering who were helpful in many ways for the completion of the project.

ABSTRACT

One of the most important problems in contemporary life is kidney disease, which affects millions of people globally. Blood metabolite creatinine has a significant positive correlation with glomerular filtration rate (GFR). Since measuring GFR is challenging, the amount of chronic kidney disease (CKD) is first determined by considering the creatinine level. A creatinine test added to a routine fitness examination should detect CKD. Creatinine testing isn't included in the routine health check in many countries because additional items for a full examination are more expensive. For diagnosing various conditions, cross-sectional slices of the human kidneys are created using computed tomography, a narrow-beam x-ray imaging technology. Using several deep-learning techniques, computer tomography pictures have been successfully categorised and segmented. However, it has proved challenging for healthcare professionals to comprehend the model's precise judgments, leading to a "black box" system. This study suggested a lightweight tailored convolution neural network to identify kidney cysts, stones, and tumours to get around these problems. The suggested CNN model outperformed other innovative techniques and achieved an accuracy of 99.52. The suggested study gives practitioners definitive and clear data with improved outcomes and enhanced interpretative ability. Finally, the flask framework is used to predict web pages.

TABLE OF CONTENTS

S. No	TITLE	Page No.
	ABSTRACT	v
	LIST OF FIGURES	viii
	LIST OF TABLES	x
1	INTRODUCTION	1
	1.1 CNN - An Overview	6
	1.2 Problem definition	12
	1.3 Objectives	12
2	LITERATURE SURVEY	13
	2.1 Overview of Literature	13
	2.2 Literature Summary	20
3	PROPOSED METHODOLOGY FOR CHRONIC KIDNEY DISEASE DETECTION	21
	3.1 Research Datasets	21
	3.2 Block Diagram of The Model	21
	3.3 Scope of the Work	25
	3.4 Sequential CNN for CKD Detection	25
4	WORKFLOW OF THE CNN MODEL FOR CHRONIC KIDNEY DISEASE DETECTION	31
	4.1 Process Flow of The Model	31
	4.1.1 Data Collection	32
	4.1.2 Data Preprocessing	33
	4.1.3 Model Training	35

4.1.4	Web Implementation	37
4.2	Performance Metrics	38
4.3	Standard	40
4.4	Constraints and Trade-Offs	41
5	RESULTS AND DISCUSSIONS	43
5.1	Datasets Used	43
5.2	Model Performance Analysis	45
5.3	Summary of the Model	48
6	CONCLUSION AND FUTURE WORK	49
6.1	Conclusion	49
6.2	Future Work	49
	REFERENCES	50
	APPENDIX	53

LIST OF FIGURES

Figure No	Description	Page No
1.1	Difference between normal kidney and diseased kidney	2
1.2	Stages of Kidney Disease	3
1.3	Indication of kidney disease based on GFR Level	3
1.4	Albumin level in a healthy Kidney Vs a damaged kidney	4
1.5	Deep Neural Network	6
1.6	Perceptron	7
1.7	Multilayer Perceptron	7
1.8	Convolutional Neural Network	8
1.9	Recurrent Neural Network	8
1.10	Working of CNN	9
1.11	Layers in CNN	10
1.12	Pooling	11
1.13	Max Pooling and Average Pooling	11
4.1	Flow diagram of the model's processing	31
4.2	Sequence diagram of the model	32
4.3	Flask Framework Architecture	38
5.1	Cyst samples	44
5.2	Normal samples	44

5.3	Stone samples	44
5.4	Tumor samples	44
5.5	Confusion matrix of the model	45
5.6	Epoch graphs of the model	46
5.7	Web interface outputs	47
5.8	Overall Architecture of the model	48

LIST OF TABLES

Table No	Description	Page No
3.1	Sample of the text Dataset Collected	22
4.1	Summary of the CNN model's layers	36
5.1	Summary of datasets collected	43
5.2	Model Performance analysis	45
5.3	Epoch summary of the model	46

CHAPTER 1

INTRODUCTION

Kidneys are critical organs within the human frame that perform a ramification of important features. Humans have two fist-sized kidneys. Their most important reason is to filter the blood. It removes waste products and extra water and turns them into urine. It additionally facilitates preserve the body's chemical stability, controls blood pressure, and produces hormones. Kidney disorder affects greater than 750 million human beings worldwide. Kidney disorder is a condition that affects human beings worldwide, however disorder prevalence, detection, and treatment vary broadly. Kidney failure is the main reason of loss of life in modern-day society. This scenario is exacerbated by smoking, binge drinking, high cholesterol, and numerous other risk factors.

Situations that injure your kidneys and lessen their ability to keep you healthy by filtering waste from your blood constitute chronic kidney disease. Wastes might accumulate to excessive levels for your blood and make you feel sick if renal disease develops. Additionally, kidney disease increases your risk of developing heart and blood vessel problems. Those problems could develop slowly over a long period of time. Chronic renal disease can frequently be prevented from going worse with early detection and treatment. As a kidney condition worsens, renal failure could develop, necessitating dialysis or a kidney transplant to maintain life.

The difference between a normal kidney and a kidney with the disease is depicted in the image below:

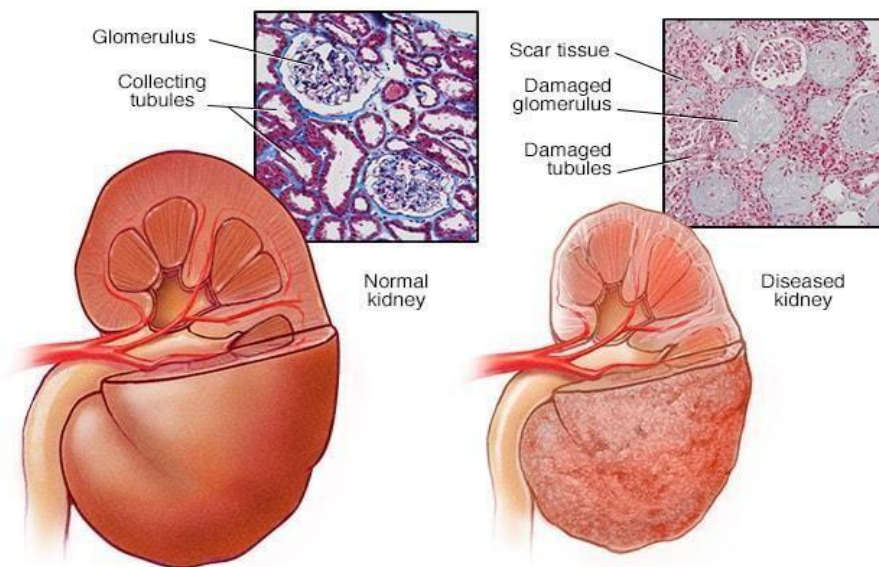


Fig 1.1 Difference between normal kidney and diseased kidney

In the above shown image (Fig 1.1) diseased kidney is compared to a normal kidney based on parameters like glomerulus, tubules etc.

Five stages of renal disorders were defined by the National renal Foundation (NKF). This helps clinicians provide the best care possible because each level necessitates individual exams and treatments. The severity of a kidney ailment is assessed using the glomerular filtration rate (GFR), a math formula that takes into account an individual's age, gender, and serum creatinine level (as detected by a blood test). One of the most important indicators of renal function is creatinine, a waste product produced by muscular activity. When the kidneys are functioning well, they remove creatinine from the blood; however, as renal function deteriorates, blood levels of creatinine rise.

- **Stage 1** with normal or high GFR (GFR > 90 mL/min)
- **Stage 2** Mild CKD (GFR = 60-89 mL/min)
- **Stage 3A** Moderate CKD (GFR = 45-59 mL/min)
- **Stage 3B** Moderate CKD (GFR = 30-44 mL/min)

- **Stage 4** Severe CKD (GFR = 15-29 mL/min)
- **Stage 5** End Stage CKD (GFR <15 mL/min)







Stage of CKD	STAGE 1	STAGE 2	STAGE 3A	STAGE 3B	STAGE 4	STAGE 5
eGFR	90 or greater	Between 60 and 89	Between 45 and 59	Between 30 and 44	Between 15 and 29	Less than 15
Level of kidney damage	 Mild kidney damage	 Mild kidney damage	 Mild to moderate kidney damage	 Mild to moderate kidney damage	 Moderate to severe kidney damage	 End-stage kidney disease. Kidneys are close to failure or have completely failed. You will need to start dialysis or have a kidney transplant.

Fig 1.2 Stages of Kidney Disease

Various techniques available for the detection of kidney disease are as follows: Blood test for GPR, Creatinine Test, Urine test for Albumin, and Urine Albumin-to-creatinine ratio (UACR). An overview of these techniques is provided below:

A blood test is taken to check kidney function. The results of the test mean the following:

- a GFR of **60 or more** is in the normal range.
- a GFR of **less than 60** may mean you have kidney disease.
- a GFR of **15 or less** is called kidney failure. Most people below this level need dialysis or a kidney transplant.

GFR level test and an indication of each level are shown in the figure below

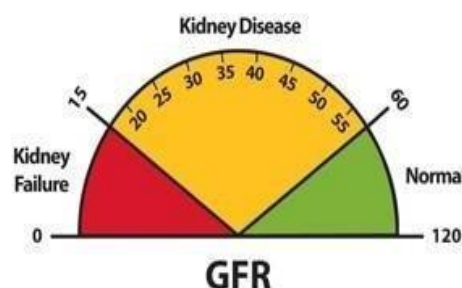


Fig 1.3 Indication of kidney disease based on GFR Level

In the above figure (Fig 1.3) the GFR level is categorized based on the range below 15 which is kidney failure, GFR of less than 60 and more than 15 which indicates kidney disease and above 60 which is normal.

A waste product of our body's routine breakdown of muscle tissue is creatinine. Creatinine is eliminated from your blood by the kidneys. Your blood's concentration of creatinine is used to calculate your GFR. The level of creatinine rises as renal disease gets worse.

A protein called albumin was found to be useful in blood. Albumin can no longer shunt into the urine in a healthy kidney. A few albumins should flow into the urine in cases of damaged kidneys. Your urine should include as little albumin as possible. Albuminuria is the presence of albumin in the urine. The difference in the level of albumin in a healthy Kidney and a damaged kidney is shown in the figure below

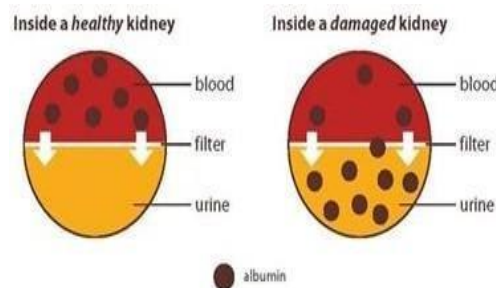


Fig 1.4 Albumin level in a healthy Kidney Vs a damaged kidney

In Fig 1.4 we can see that the albumin is filtered by a healthy kidney whereas albumin is not filtered and is more in urine.

The urine albumin-to-creatinine ratio (UACR) test calculates the ratio between the albumin and creatinine concentrations in a urine sample. The UACR is used to calculate how much albumin might flow into the urine in a 24-hour period. An albumin urine test result of

- **30 mg/g** or much less is ordinary
- **Extra than 30 mg/g** may be a signal of kidney disease

A lot of these strategies are invasive in nature. Consequently, it necessitates a non-invasive method for the diagnosis of kidney illnesses. massive work is completed in this location.

Ebrahime Mohammed Senan et al. (2021) In their study they assisted specialists in investigating CKD prevention options by early diagnosis using device learning techniques. Recursive function elimination (RFE) was used to choose the most crucial capabilities. The support vector machine (SVM), K-nearest neighbour (KNN), choice tree, and random forest were the four category algorithms used in this study. All of the class algorithms delivered overall performance that was encouraging. The random forest algorithm was found to perform better than all other algorithms, achieving 100% accuracy, precision, recall, and F1-score for all measures. A multiclass statistical analysis was used to test and evaluate the system, and the empirical results of the SVM, KNN, and decision tree algorithms revealed high values for the accuracy measure of 96.67%, 98.33%, and 99.17%.

Njoud Abdullah Almansour et al. (2019) completed experiments; the implication of the related attributes modified all missing values in the dataset. Then, with the help of fine-tuning the parameters and running multiple trials, the optimised parameters for the Artificial Neural Network (ANN) and Support Vector Machine (SVM) approaches had been established. With accuracies of 99.75% and 97.75%, respectively, the actual findings of the studies showed that ANN performed better than SVM, suggesting that the outcome is quite promising. It was found that ANN performed better than SVM overall, with an accuracy of 99.75% utilizing the optimised features, while SVM had an accuracy of 97.75%.

Rui Gao et al. (2021) , had a look at Serum Raman spectra of healthy and CRF Patients were identified using a convolutional neural network (CNN), and in comparison, the effects were identified using a more advanced version of AlexNet. To further investigate the effect of a little random noise on the experimental results, a unique amplitude of noise was introduced to the spectral information of the samples. The accuracy of classifying the spectra using a CNN and a stepped-

forward AlexNet was 79.44% and 95.22%, respectively. The accuracy of the categorization was no longer significantly impacted by the presence of noise. In this examination, CNN's accuracy may reach 95.22 percent, up from 89.7 percent in the previous examination. The findings of this study demonstrated that serum Raman spectra and CNN may be used together to diagnose CRF, and that minor random noise no longer significantly impedes the results of records evaluation.

However, in all the above approaches, the prediction rate is not satisfactory. Hence it is necessary to develop an efficient technique for the classification of images. To do so, it is necessary to have a deep insight in the working of Convolutional Neural Networks.

1.1 CNN - AN OVERVIEW

Deep learning techniques are based on neural networks, also known as synthetic neural networks (ANNs), or simulated neural networks (SNNs), a subset of device mastering. A number of node layers, an input layer, one or more additional hidden layers, and an output layer are all components of artificial neural networks (ANNs). Each node, or synthetic neuron, is connected to another and has a weight and threshold that go along with it. A node is activated and sends information to the community's next layer if its output price is higher than the desired threshold price. In every other situation, no information is communicated to the community's next level. Illustration of the process of deep neural network is shown in the figure below

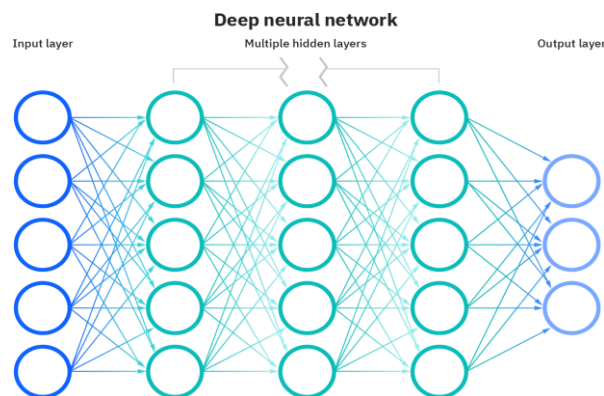


Fig 1.5 Deep Neural Network

As we can see in the above figure (Fig 1.5) the deep neural network contains many hidden layers which make it possible to segment the function of a neural network into particular data manipulations.

Different Models of Neural Network

The first neural network was the perceptron, developed by Frank Rosenblatt in 1958. It is the most basic type of neural network and consists of a single neuron. and shown in the below figure

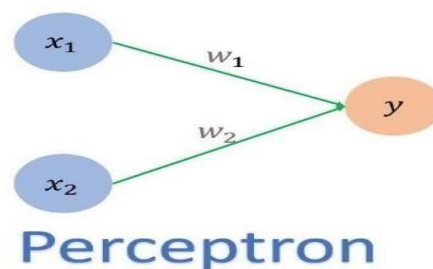


Fig 1.6 Perceptron

Multi-layer perceptrons (MLPs)

Feed-forward neural networks, are made up of an input layer, one or more hidden layers, and an output layer. While these neural networks are also frequently referred to as MLPs, it is important to note that they are made up of sigmoid neurons rather than perceptron. The structure of the multilayer perceptron is shown in the figure below

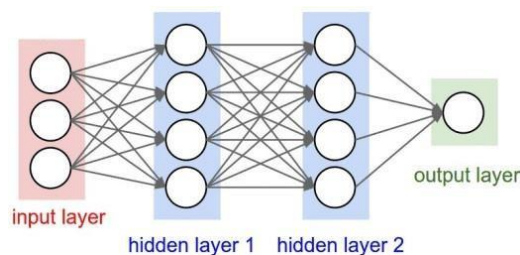


Fig 1.7 Multilayer Perceptron

We can see the different layers and the working of a Multilayer Perceptron in figure 1.7.

Similar to feedforward networks, convolutional neural networks (CNNs) are used in computer vision, pattern recognition, and photo recognition applications. To recognise patterns in a picture, those networks use concepts from linear algebra, specifically matrix multiplication. An example of a convolutional neural network is shown in the figure below

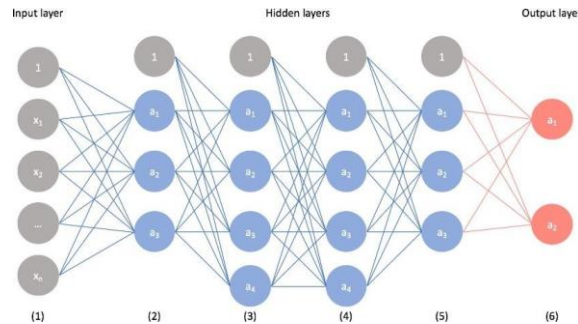


Fig 1.8 Convolutional Neural Network

From Fig 1.8 we can understand the structure and working of the convolutional neural network.

Recurrent neural networks (RNNs)

Using their feedback loops, are diagnosed. When using time-series data to forecast future outcomes, such as stock market projections or income forecasting, those learning algorithms are typically used. A recurrent neural network is shown in the figure below:

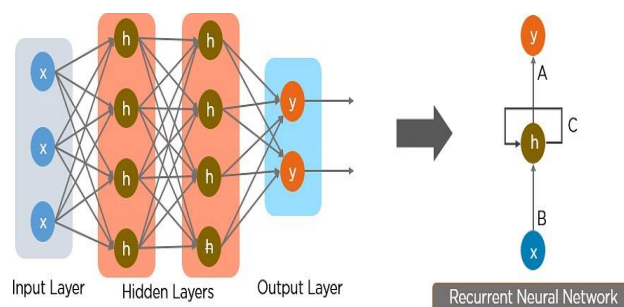


Fig 1.9 Recurrent Neural Network

The above-shown Figure (Fig 1.9) illustrates the processing of each layer and the usage of feedback loops.

Working of CNN

Artificial neurons are arranged in multiple layers to form convolutional neural networks. Synthetic neurons are mathematical operations that compute the activation price as the weighted average of several inputs. A ConvNet generates various activation capabilities when you input a photo, and each layer passes these capabilities on to the following layer. The working of CNN is illustrated in the figure below.

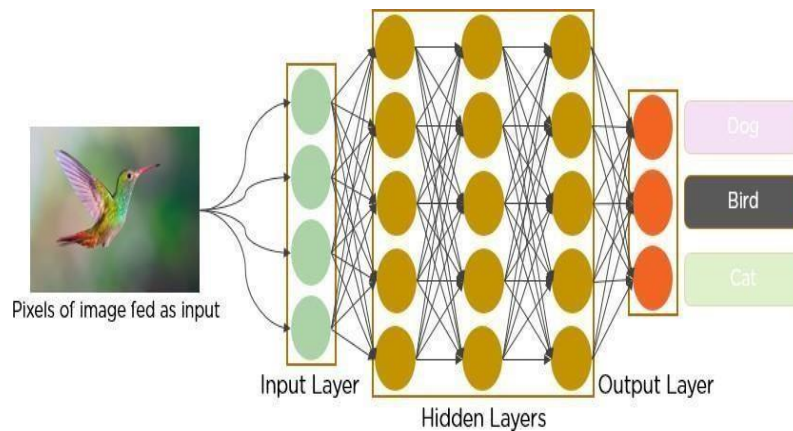


Fig 1.10 Working of CNN

The above image (Fig 1.10) shows how an image is processed by a CNN using different layers and how the image is classified.

Layers in CNN

Horizontal or diagonal edges are among the basic capabilities that are frequently extracted from the initial layer. This output is immediately advanced to the next layer, which finds more complex capabilities as well as corners or combined edges. As we get further into the neighborhood, it might be able to recognize even more intricate capabilities like items, faces, etc.

The procedure taking place in each layer is depicted in the image below.

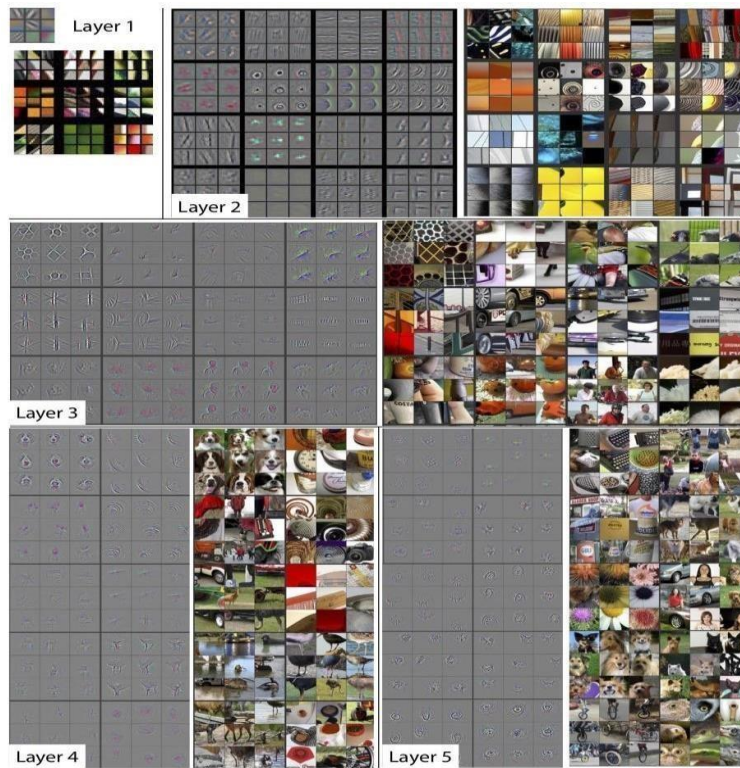


Fig 1.11 Layers in CNN

As we can see from Fig 1.11, The classification layer generates a series of confidence ratings (values between 0 and 1) based entirely on the activation map of the final convolution layer that indicate how likely it is that the image is a "elegance." For instance, if your ConvNet can identify cats, dogs, and horses, the very last layer's output will be the likelihood that the input image contains any of those animals. Any of those animals.

A method for helping the community understand characteristics regardless of their location inside the image is pooling in convolutional neural networks, which generalizes features extracted using convolutional filters. There are various approaches to pooling. The two most frequently utilized procedures are common pooling and max-pooling.

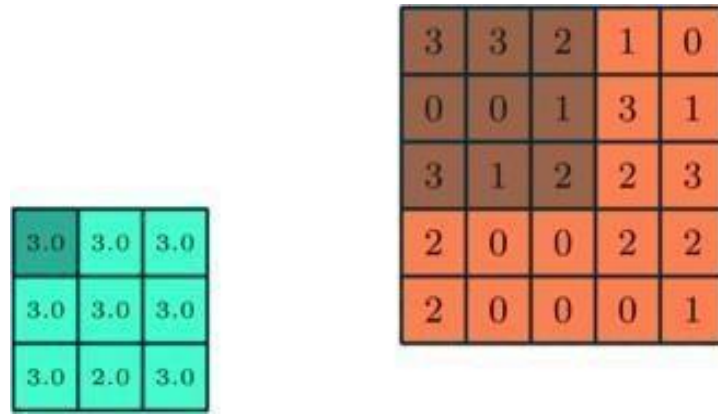


Fig 1.12 Pooling

Therefore, in Max Pooling, we determine the maximum value of a pixel from a portion of the image that is covered by the kernel. Additionally, Max Pooling functions as a noise suppressant. It does de-noising along with dimensionality reduction and completely discards the noisy activations. Instead, common pooling delivers the mean of all the data from the area of the image that is covered by the kernel's resource. Average Pooling without a doubt plays dimensionality reduction as a noise suppressing mechanism.

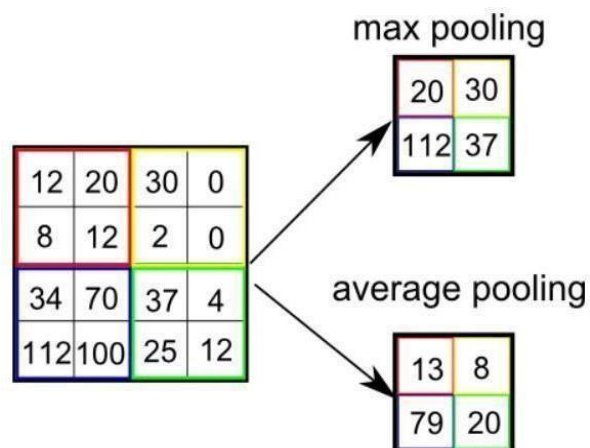


Fig 1.13 Max pooling and Average pooling

Having understood the problem statement and the working of Convolutional Neural Networks, an efficient CNN based algorithm can be developed for the classification of images. Also, the performance can be measured in terms of suitable metrics.

1.2 PROBLEM DEFINITION

The proposed work aims to develop deep-learning neural community architectures for the detection of continual Kidney disorder detection. The proposed work additionally targets reading the overall performance of the proposed techniques by suitable performance metrics.

1.3 OBJECTIVES

The detailed objectives are listed below:

1. To accumulate chronic kidney disease images the use of a suitable diagnostic modality.
2. To understand the effect of metadata on the algorithm for the detection of kidney illnesses.
3. To select an appropriate deep studying architecture for the detection of kidney sickness.
4. To appraise the performance of the chosen structure.
5. To increase a customized deep getting to know community for the detection of kidney sicknesses.
6. To assess how well the suggested method performs in comparison to the traditional deep learning strategy for detecting renal disease.

CHAPTER 2

LITERATURE SURVEY

A detailed literature survey on the convolutional techniques available for the detection of CKD is given in this chapter. Biology behind CKD, diagnostic techniques, and Machine Learning techniques for the identification of kidney disease are explained in this chapter.

2.1 OVERVIEW OF LITERATURE SURVEY

Biology Behind Chronic Kidney Disease

Paola Romagnani et al. (2017) In this study described recurrent urine abnormalities, anatomical abnormalities, or reduced excretory renal characteristics suggesting of a loss of functional nephrons are used to define chronic kidney disease (CKD). They also suggested that the majority of CKD patients were more likely to develop cardiovascular disease and pass away. They came to the conclusion that the pillars of treatment for CKD patients should be early detection or prevention, blood pressure management, suppression of the renin-angiotensin system, and condition-specific therapies. Cardiovascular health and quality of life are impacted by CKD headaches, such as anemia, metabolic acidosis, and secondary hyperparathyroidism, which necessitate diagnosis and treatment.

Angela C Webster et al. (2017) According to the most recent international guidelines, this condition is defined as having decreased kidney function, independent of the underlying cause, as measured by a glomerular filtration rate (GFR) of less than 60 mL/min per 7.3 m² or indicators of kidney damage, or both. Diabetes and high blood pressure were examined as the two main causes of CKD. Many people were found to be asymptomatic or to have non-specific symptoms including fatigue, itching, or appetite loss. The risk of CKD progression and death is increased when proteinuria is present. It was shown that those with CKD were 5 to 10 times more likely to pass away before developing advanced kidney disease. It was determined that the fitness-related quality of lives for those with CKD is much poorer than for the general population and decreased as GFR decreased. Interventions targeted at specific symptoms that supported educational or lifestyle

choices, establishing a favorable differentiation for people living with CKD.

Dervla M. Connaughton et al. (2019) revealed that there have been over 500 monogenic causes of chronic kidney disease (CKD) that have been strongly associated with pediatric populations. Less research has been done on how common monogenic causes are in adults with CKD. To assess the viability of identifying monogenic causes of CKD, whole exome sequencing (WES) was carried out in a multi-Centre cohort of 114 families with 138 affected people who had the disease. The affected adults came from 78 families with a positive family history, 16 families with redundant-renal traits, and 20 families without a family history or extra-renal features. In 42 of 114 families (37), a pathogenic mutation in a known CKD gene was discovered. In 36 afflicted families with a confirmed family history of CKD, 69 affected families with extra-renal symptoms, and only 15 affected families without a confirmed family history or extra-renal abnormalities, a monogenic etiology was identified. When a monogenic cause was identified in 42 families, WES confirmed the clinical opinion in 17 of those families (40), corrected the clinical opinion in 9 of those families (22), and established an opinion for the first time in 16 of those families (38), where CKD with an unclear etiology was present. In the early stages of medication development, it was proposed that WES might be a crucial tool for determining the root cause of CKD in adults.

Adriano Luiz Ammirati (2020) suggested that the advanced cardiovascular hazard relates to chronic renal disease, which is widely prevalent (10–13% of the population), incurable, progressive, and present. Most of the time, cases with this condition are asymptomatic, only seldom exhibiting consequences typical of renal failure. Treatment options include relief therapy (hemodialysis, peritoneal dialysis, and organ transplantation) or conservative care (patients without a recommendation for dialysis, typically those with glomerular filtration rates above 15 ml/nanosecond). According to the findings of this investigation, conservative therapy for habitual order Disease assisted with hepatitis B vaccination, the treatment of problems (anemia, bone conditions, cardiovascular conditions), and the prescription for order relief remedy. It also slowed down the course of order dysfunction.

Rajiv Agarwal et al. (2020) suggested that the advanced cardiovascular hazard relates to chronic renal disease, which is widely prevalent (10–13% of the population), incurable, progressive, and present. Diabetes was cited as the primary cause of chronic kidney disease (CKD) in cases with this pathology and as a risk factor for the development of CKD into end-stage kidney disease (ESKD). Additionally, it was claimed that CKD in those with type 2 diabetes (T2D) was linked to a shorter life span. As a result, it was determined that although individuals with early CKD in T2D were frequently omitted from randomized studies of renal failure, they were far more likely to die than advance to ESKD. The FIDELIO-DKD and FIGARO-DKD trials examined a therapy without a glucose-lowering effect and were prespecified superiority studies as opposed to safety trials. These studies were discovered to have the ability to slow the progression of CKD and provide cardiorenal protection in T2D patients with CKD at all stages.

Hui- Ju Tsai et al. (2021)- According to this study, environmental adulterants such heavy essence, PM, and related chemicals including phthalates, melamine, and BPA play a role in the etiology of CKD. Most environmental nephrotoxics cause CKD through pathogenic pathways that were demonstrated. More thorough longitudinal studies and experimental designs with precise and quantified measures of environmental exposure were required to establish unproductive connections and cure-response associations between exposure to environmental adulterants and order complaints for a variety of exposure situations. In conclusion, their research confirms the need for non supervisory approaches to pollution control and the mitigation or prevention of exposure to environmental health risks.

Roser Torra et al. (2021) suggested that inherited diseases (IKDs) were one of the major factors in early-onset habitual complaints (CKD) and were to blame for at least 10 to 15 cases of adult order relief remedies (KRT). Recent studies have shown that monogenic diseases have a high likelihood of causing adult instances of CKD. There are a number of factors that contribute to inheritable order disorders being difficult to detect in most cases, including the fact that adult nephrologists are generally uninformed about IKDs, the existence of atypical phenotypes, and the lack of widespread availability and affordability of inheritable testing. In place of reporting

only cystic complaints and hiding other IKDs under markers like "assorted" or "other," it was suggested that registries could help to understand the burden of IKDs by routinely grouping all IKDs in their periodic reports, as was done for glomerulonephritis or interstitial conditions. In this situation, inheritable testing might be useful, but it should be utilized carefully and with a thorough understanding of IKDs.

Diagnostic Techniques for Kidney Diseases

Zainuri Saringat et al. (2019) attempted to establish whether the stage of CKD was acute or chronic based on the symptoms or characteristics seen in a specific case. To do this, a categorization model with stages of severity for renal disease cases was proposed. The proposed classification model, which is based on the eight supervised classification algorithms ZeroR, Rule Induction, Support Vector Machine, Naive Bayes, Decision Tree, Decision Stump, k- Nearest Neighbour, and Classification through Regression, was also put to the test. Accuracy, precision, and recall were used to estimate each classifier's performance. The outcomes showed that the regression classifier performs well during the kidney diagnostic process.

Mohamed Elhoseny et al. (2019) developed Density based Feature Selection (DFS) and Ant Colony based Optimisation (D- ACO) algorithm for Chronic Kidney Disease (CKD). By using DFS prior to the building of the ACO-based classifier, the recommended intelligent system eliminated features that were unnecessary or inapplicable. Three phases—Velicet preprocessing, Feature Selection (FS), and classification—were included in the recommended D-ACO framework. The suggested D- ACO algorithm initially beat the previous strategies with improved classification performance in a number of ways as compared to the current procedures. Overall, the proposed D- ACO method was designed to be a useful classifier for identifying CKD.

Adeola Ogunleye et al. (2019) The extreme gradient boosting (XGBoost) approach was chosen as the base model in this paper due to its outstanding performance among numerous conventional and contemporary AI algorithms that were examined in the context of CKD. Additionally, the model was improved, and

the best whole model that was trained on all the features had testing accuracy, sensitivity, and specificity that were all equal to or greater than 1.000. In contrast, the suggested full model attained independently 1.000, 1.000, and 1.000 for accuracy, sensitivity, and specificity. The accuracy, sensitivity, and specificity of a simplified model containing roughly half of the entire features were 1.000, 1.000, and 1.000, respectively.

Mehdi Hossein Zadeh et al. (2020) In this paper, an IoT multimedia data-based diagnostic prediction model for CKD and its severity was developed. Since there are many factors that affect CKD and the volume of IoT multimedia data is typically very large, different features were chosen in different groups of multimedia datasets for CKD in order to evaluate performance measures of CKD prediction and its level determination using various classification techniques. These features were chosen based on physicians' clinical observations and experiences as well as previous studies. The experimental results showed that using decision tree (J48) classifiers in Support Vector Machine (SVM), Multi-Layer Perception (MLP), and Nave Bayes classifiers, the applied method with the suggested selected features produced 97% accuracy, 99% sensitivity and precision, and 95% specificity. Additionally, it was found that the proposed feature set, when compared to other datasets with various features, can reduce the execution time.

Ning Shang et al. (2021) built a portable and expandable electronic CKD phenotype to enable extensive experimental and genetic research of kidney features and to facilitate early disease detection. The program automatically staged cases on the albuminuria by glomerular filtration rate grid (also known as the "A-by-G" grid) using a combination of rule-based and machine-learning approaches. 451 chart reviews from three different medical systems were used to manually validate the algorithm, and the results showed an overall positive predictive value of 95 for CKD cases and 97 for healthy controls. A few of the traits' large-scale activities were also displayed. Their results supported the idea that CKD could not be a single phenotype but rather a collection of genetically and phenotypically diverse illnesses that includes numerous Mendelian subtypes and disorders of the inheritable determination that is more sophisticated. Automated CKD subtype determination and more precise methods for estimating GFR in adult and pediatric cases with

various ancestry backgrounds are likely to be future developments in e-phenotyping for CKD.

Conventional Techniques for Kidney Disease Detection

OlivierJ. Wouters et al. (2015) advised a better understanding of the benefits and drawbacks of various CKD treatment modalities and emphasized the necessity of doing so in order to improve health outcomes and save costs. He claimed that CKD, like many other chronic diseases, was essentially a broad measure of overall health. The majority of cases with mildly or relatively depressed eGFRs have comorbidities that are more relevant to their present and future health than a CKD diagnosis. The effectiveness of early therapies has diminished due to how difficult it is for doctors to recognize cases of progressing CKD. Despite the fact that alternative biomarkers that may improve CKD identification and prognosis were becoming more and more accessible, it was determined that it was unlikely that any panacea would totally alleviate the complaint burden of CKD.

YuryE. Glazyrin et al. (2020) suggested that the most prevalent causes of chronic kidney disease (CKD) are glomerulonephritis, hypertension, and diabetic nephropathy. A differential diagnosis and a suitable therapy are required at the very early stages because CKD from multiple causes may not become apparent until renal function is noticeably compromised. With high confidence (97.8), the k-nearest neighbors algorithm demonstrated the possibility of separating a healthy group from renal patients in general. According to proteomics data of plasma, this algorithm was also shown to be the most effective of the three examined for differentiating the groups of cases with diabetic nephropathy and glomerulonephritis (96.3 of correct conclusions). However, utilizing the k- closest neighbors classifier and urine proteome data, the group of hypertensive nephropathies was successfully distinguished from all other renal cases. In addition to being able to distinguish hypertensive and diabetic nephropathy in the early stages based not on individual biomarkers but rather on the entire proteomic composition of urine and blood, the tested algorithms demonstrated good capacities to separate the various groups across proteomic data sets, which could be helpful to avoid invasive intervention for the verification of the glomerulonephritis subtypes.

Feasibility of Deep Learning Neural Networks for Kidney Disease Detection

Gabriel R. Vásquez- Morales et al. (2019) In order to predict if a person was at risk of getting chronic kidney disease (CKD), this study developed a neural network-based classifier. Two different demographic groups' medical care data were used to train the model. People in Colombia who were diagnosed with CKD in 2018 are shown on the one hand, and a sample of those who were not given this diagnosis are shown on the other. They used data from medical procedures conducted on people between the years 2009 and 2018 to develop the model in this study. The training data set for this work consisted of 1000 persons. They created a neural network strategy using the new dataset, and it was 95 percent accurate at predicting the likelihood of having chronic kidney disease. The most comparable study's accuracy was 89.7, so this was a fantastic outcome.

Iliyas Ibrahim Iliyas et al. (2020) recommended that the investigation of Deep Neural Network (DNN) is becoming a focus. They added that disregarding renal dysfunction could result in chronic kidney disease and eventual mortality. They employed a DNN model to predict whether or not each case had CKD by obtaining a dataset from Bade General Hospital consisting of 400 cases with 10 attributes. The model's accuracy was 98 percent. When the model was tested using a set of data that weren't included during the training procedure, they were successful in categorizing the kidney disease dataset into CKD and non-CKD with 98 overall accuracies. The DNN model that was chosen turned out to be efficient and appropriate for the vaticination of renal illness.

Vijendra Singh et al. (2022) This investigation offers a fresh deep literacy model for CKD early detection and vaccination. The goal of this investigation was to create a deep neural network and evaluate its effectiveness in comparison to other modern machine learning techniques. All database missing values were replaced throughout tests using the relevant features' normal. Machine learning models were given selected features for classification reasons. By reaching 98 accuracies, the proposed Deep neural model surpassed the four other classifiers (SVM, KNN, Logistic regression, Random Forest, and Naive Bayes classifier). Nephrologists may find the suggested method helpful in the CKD diagnosis.

Muthiah.M. A, Logashanmugam.E, and Nandhitha.N.M (2021) This work investigated the convolutional neural networks' potential for classifying floral photos. This model predicted had a precision of 82.60 independently among the different Convolutional Neural Network infrastructures. Independent accuracy for VGG19 was 66.19. The accuracy of DENSENET201 was 92.6 independently. Independently, RESNET18 had an accuracy of 84.59. Independently, RESNET50 has an accuracy of 89.8. RESNET101 independently measured accuracy at 89.2. The highest accuracy is achieved by Densenet201 for the classification of flowers dataset. The impact of a huge dataset on the performance of the suggested techniques can be evaluated, even if the offered algorithms were scalable in nature. The simulation position is the last point of work.

2.2 LITERATURE SUMMARY

From the surveyed literature, it is found that the performance of the techniques is dependent on the type of images used as inputs, type of the diagnostic technique used for acquiring images and nature of the classifier. Many modifications cannot be done on the classifier algorithms. Hence, the proposed work must be in such a way to increase the overall performance of the system.

CHAPTER 3

PROPOSED METHODOLOGY FOR CHRONIC KIDNEY DISEASE DETECTION

3.1 RESEARCH DATASETS

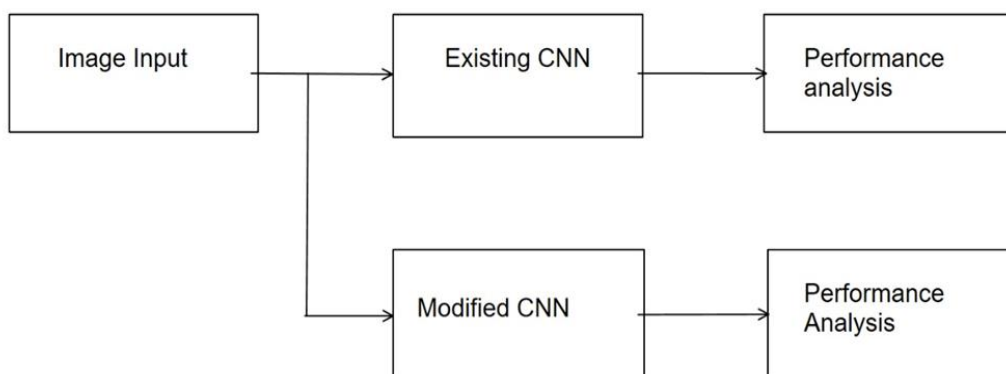
The sets of data used for the CNN model's implementation were gathered from well-known web dataset sources like Kaggle as well as from samples of MRI scans from hospitals and diagnostic labs. The collected datasets were then classified according to their class and were stored as individual databases for easy implementation and training of the model developed.

3.2 BLOCK DIAGRAM OF THE MODEL

The proposed methodology used CNN as the base to achieve the solution for the problem statement stated at the beginning of the research.

A Convolutional Neural Networks is a built-in integrated algorithm that can discriminate between different components and devices in the image by assigning importance (learnable weights and biases) to each one. A subclass of neural networks called convolutional neural networks (CNN or ConvNet) are mostly employed in applications that combine speech and picture recognition. CNNs are appropriate for this case since their convolutional layer decreases the high dimensionality of snapshots.

The block diagram demonstrating the overall process of scope is as follows:



Let us understand the overview of each block in detail

Image Input

The photograph input stated here has been obtained after a conversion of the pictorial database to the textual dataset. This textual database was downloaded from outside database websites. The database was produced from records from four hundred patients of different age organizations containing 24 unique functions. The missing numerical and nominal values were updated using the propose and mode statistical analysis approaches. since visible facts are proven to present better efficiency with CNN than textual records, we shift our records input to the photograph dataset. A sample of the collected textual dataset is shown in the table below

Table 3.1 Sample of the text Dataset Collected

id	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	bu	sc	sod	pot	hemo	pcv	wc	rc
0	48	80	1.02	1	0	1	1	0	0	121	36	1.2	0	0	15.4	44	7800	5.2
1	7	50	1.02	4	0	1	1	0	0	0	18	0.8	0	0	11.3	38	6000	0
2	62	80	1.01	2	3	1	1	0	0	423	53	1.8	0	0	9.6	31	7500	0
3	48	70	1.005	4	0	1	0	1	0	117	56	3.8	111	2.5	11.2	32	6700	3.9
4	51	80	1.01	2	0	1	1	0	0	106	26	1.4	0	0	11.6	35	7300	4.6
5	60	90	1.015	3	0	1	1	0	0	74	25	1.1	142	3.2	12.2	39	7800	4.4
6	68	70	1.01	0	0	1	1	0	0	100	54	24	104	4	12.4	36	6400	0
7	24	0	1.015	2	4	1	0	0	0	410	31	1.1	0	0	12.4	44	6900	5
8	52	100	1.015	3	0	1	0	1	0	138	60	1.9	0	0	10.8	33	9600	4
9	53	90	1.02	2	0	0	0	1	0	70	107	7.2	114	3.7	9.5	29	12100	3.7
10	50	60	1.01	2	4	1	0	1	0	490	55	4	0	0	9.4	28	6500	0
11	63	70	1.01	3	0	0	0	1	0	380	60	2.7	131	4.2	10.8	32	4500	3.8
12	68	70	1.015	3	1	1	1	1	0	208	72	2.1	138	5.8	9.7	28	12200	3.4
13	68	70	0	0	0	1	1	0	0	98	86	4.6	135	3.4	9.8	38	6000	0
14	68	80	1.01	3	2	1	0	1	1	157	90	4.1	130	6.4	5.6	16	11000	2.6
15	40	80	1.015	3	0	1	1	0	0	76	162	9.6	141	4.9	7.6	24	3800	2.8
16	47	70	1.015	2	0	1	1	0	0	99	46	2.2	138	4.1	12.6	39	6200	0

As we can see in the above figure, different features like Blood Pressure, Albumin level, RBC, WBC, Platelet Count, Hemoglobin Level, Level of Appetite, and many more are collected from each patient. To increase the efficiency of detection, this dataset is replaced with direct 2D image dataset and is taken as input for the neural network.

Existing CNN

There are several CNN models under existence in the present time. Some of the popular ones used are **DarkNet**, **ResNet**, **GoogleNet**, **LeNet**, **MobileNet**, and many others. Each of these models possess their own level of sensitivity and accuracy in prediction of the intended result based on the image input given. After an implementation trial on multiple CNN models, based on the level of accuracy shown, the suitable model will be opted for the advancement of the model.

Performance Analysis

The performance of any CNN model is done based on certain performance analysis parameters some of which are sensitivity, specificity, accuracy, probability of reduction

.

(i) *Sensitivity:*

It's a measure of how well a machine-learning version can discover practical instances. it's also known as the actual fantastic charge (TPR) or consider. Sensitivity is used to evaluate the model's overall performance as it permits us to look what the number of high-quality times the version can perceive properly.

(ii) *Specificity:*

It determines a model's ability to predict if a remark no longer belongs to a selected category. It calls for an understanding of the version's performance when the statement absolutely belongs to each different category than the only being taken into consideration.

(iii) *Accuracy:*

Version accuracy is defined because the quantity of classifications a version effectively predicts divided by the full quantity of predictions made. it is a way of assessing the overall performance of a version, however truly not the only possible way.

(iv) *Probability of reduction/Confusion matrix:*

It serves as a performance indicator for a classification problem involving machine learning in which there may be two or more classes as an output. There are four

possible anticipated and actual value combinations in the table.

Modified CNN

The primary parameter that can be modified to improve any model is the **number of layers**. When the number of layers is increased, the efficiency of prediction of the model will show an increase. Apart from the number of layers, we can also modify other parameters of the existing models to improve the efficiency of working of our new model. Some of these are:

- Number of strides
- Filter size
- Number of filters
- Number of convolutional layers
- Different pooling function
- Different fully connected network.

Modifying some or all these parameters helps optimize the working of the model to a better extent.

Performance analysis

The working efficiency of the new model developed can be estimated from the performance analysis parameters as mentioned earlier. With the increased no of layers and other optimizations in the new model, the performance analysis parameters are modified as follows:

- (i) *Sensitivity*: With an increased number of layers and other changes, the sensitivity of the entire model will get optimized in a positive aspect.
- (ii) *Specificity*: With the enhanced model, the specificity in prediction of the result will show an increase.
- (iii) *Accuracy*: With suitable modifications applied, the net accuracy of the model will show a rise and is improved to a larger extent.

- (iv) *Confusion matrix or probability of reduction:* With required modifications done, the new model developed will possess a better and highly optimized confusion matrix and a greater probability of reduction.

3.3 SCOPE OF THE WORK

1. The suggested approach may evaluate the risk of CKD for public health using widely accessible health data.
2. High-risk individuals can be identified through screening and urged to undergo a creatinine test for additional assurance.
3. By doing this, we can lessen the negative effects of CKD on public health and make it easier for many people to get early detection in cases when a creatinine test is not accessible.
4. These models, when used in conjunction with a nephrologist's experience, can speed up and lower the cost of a patient's CKD diagnosis.
5. The feature extraction and learning from images demonstrate significant potentials for ML in both general and medical applications. The application of it in CKD cases would be in later study.

3.4 SEQUENTIAL CNN FOR CKD DETECTION

The basic algorithm used to implement this detection model is CNN as discussed in the previous sections. To be specific, the nature of CNN used to implement the model is 'sequential CNN'. Let us understand the basics on the type of neural network used followed by the working of the model.

Sequential CNN is one of neural network model developed with multiple trained convolutional layers that efficiently works at understanding the required parameter to detect and detects the intended results with the utmost accuracy. The basic idea behind sequential CNN is that it involves multiple hidden layers working within the overall model that involve in the training stage and giving best results in

the testing and validation stages. Let us have an initial understanding on convolutional layers and their requirement in any neural network model through the following sections.

Convolutional Layers:

Convolutional Layers act as the foundation of any neural model as they act as the main source of database containing the information about the nature of inputs about to receive and the knowledge on the factor by which classification of the inputs is to be done. In other words, they are the sources of the model containing the kernels important for classification.

The convolutional neural layers perform an elementwise dot product with the data results from the previous layers and the pre-trained kernels available with the current layer. This dot product process helps the model generate multiple intermediate datasets thus increasing the number of datasets available for the model to refer thus improving its accuracy of prediction. Hence the final output from the entire convolutional neuron would be the summarized results of the preceding layers along with the learnt kernels.

The size of these existing kernels with the model is a hyper parameter defined by the designer of the network while building the model. For an improved comprehension of the model, it is important for us to get an idea on the various hyper parameters that are user-defined and those assumed by the neural network based on the inputs from the designer. It is as well important to know what model parameters are and how do each of these impacts the results of the model.

Hyper parameters:

Hyper parameters are the parameters declared by the designer for the model well before its training begins. In other words, the metrics defined by the user before the start of the start of the training process of the model are called the hyper parameters or the top-level parameters. These parameters are given the utmost importance in the development of the neural model because, once defined, no value of any hyper parameter can be varied at any stage of the model's life cycle. Also, the values of these parameters cannot be seen from the resultant model. They play

a significant role only during the training stage of the model.

Some of the commonly used hyper parameters in the development of any CNN model are:

a) Padding:

Padding commonly refers to the feature when the model extends its filter size beyond the declared kernel. This is one of the commonly declared hyper parameter where the model finds this padding useful during its training stage. Zero padding is one of the common methods of padding adopted due to its simplicity and higher accuracy. This strategy of padding is used by most of the high performing convolutional models like AlexNet.

b) Kernel size:

Kernel size indicates the size of the filter to be taken by the model in turn deciding the final dimension of the sliding window to be derived from the input supplied to the model. The least the value of kernel the more is the efficiency of the model and its accuracy of detection. This is because when lesser kernel size is declared for the model, it involves in multiple stages of refinement thereby getting a deeper detail on the dataset available as well as helping the model turn out as a deeper one with multiple layers of processing.

c) Stride:

Stride refers to the pixel shift to be incorporated by the model on the kernel existing with it. The lesser the stride value considered, the greater is the efficiency. If a higher stride is considered by the model, a lesser amount of feature extraction happens, and the least depth of learning is achieved by the model. If the stride is taken more, it leads to better feature extraction and better turnup of the model.

d) Batch Size:

This parameter specifies the total subsets and number of datasets to be considered in each subset for the model to take as training data.

e) Learning Rate:

This specifies how many times the model must adjust itself to account for the expected error each time its weights are changed. This enables us to determine how frequently the model parameters are cross-checked.

f) Number of Epochs:

Epochs generally refer to the number of iterative cycles of training to be taken up by the model. This is basically decided based on the validation error and accuracy of the model once we run the model. Based on the required accuracy and error reduction, this epoch number is updated before starting the training of the model.

Model Parameters:

Model parameters on the other hand are concerned internally with the model. Alike the hyper parameters being user-defined, model parameters are taken values by the model itself based on its learning from the training stage using the hyper parameters. These model parameters are initialized in the start with some initialization values which in the later stage of execution are given values by the model based on the learning and assumptions of the model being built. Some of the common model parameters are:

- a) Coefficients of linear regression models
- b) Weights of pixels
- c) Biases
- d) Cluster centroids

One of the notable advantages with sequential CNN is that it is customizable at any stage of the neural network's implementation based on the requirement. This major advantage adds up to the fact of using this model for the implementation of this project concerned with the medical sector where accuracy place a significant role.

Activation Functions:

Activation functions are the key performers of any CNN model that will help the model convert multiple inputs from previous layers and the inbuilt kernels into a combined single output. They are also called 'transfer functions' which aim at mainly making the model more non-linear. The more non-linear the model is, the more

complex is neural connection becomes and better becomes its performance and understanding on the intended result.

Though activation functions are used in all 3 layers of CNN, the input, hidden and output layers, the role of them become most prominent in the hidden layer. Activation functions are implemented once each neural network node has been processed. The most typical activation methods employed by CNN are:

- Rectified Linear Activation (ReLU)
- Logistic (Sigmoid)
- Hyperbolic Tangent (Tanh)

Among all these types, the most used activation function at modern times is ReLU.

a) Rectified Linear Activation:

Being the modern activation function used for classification, ReLU shows out a better result and good performance with all scenarios of classification especially outperforming in multi-class classifications. The present implementations of CNN models all being multi-class classifications, utilize ReLU to attain a maximum efficiency. They mainly omit the vanishing gradient issue seen with the other two activation functions and avoid negative dimensions in the process of modelling. The default range of values taken by ReLU are from 0 to all positive values.

Hence ReLU is opted as the activation function for this model.

b) Sigmoid activation:

Sigmoid activation also called the Logistic activation is mainly used in the logistic regression classification algorithm. Unlike ReLU, the range of values considered by sigmoid are between 0 and 1. The more relative the output is, the closer it is to 1. Hence these functions are main highly useful for binary classifications.

c) Softmax:

This is another activation function used as the final layer of any CNN model that will ensure if the model's final output sums up to 1. In other terms, softmax aims at making the model's output into more of a probabilistic fashion. The predicted outputs from the ReLU layers won't be probabilistic in nature and it is the job of the softmax function to scale those outputs in probabilistic terms and give out that as the model's final predicted output, which will later be useful for classification of the inputs given.

Pooling Layers:

Pooling layers mainly involve at reducing the spatial extent of the model's outputs at each layer as well as at the final output. They mainly involve in gradually decreasing the dimensionality of the layer-wise outputs and perform a reduction in dimension to the model's final output. Among the two available types of pooling, max and average pooling, due to its increasing efficiency than the latter, max pooling is selected as the pooling layer for this model's implementation. In max pooling, based on the kernel (filter) size and the stride length configured as hyper parameters before the training of the model, max pooling strides the kernel and chooses only the largest value at each kernel slice(tensor) given as input and pushes that as the detected output to the corresponding layer.

Flatten Layer:

This is another important layer required by any common CNN model for an effective prediction. Here the flatten layer converts the multi-dimensional matrix from the pooling layers in the network to a 1-dimensional or linear vector for easier computation and prediction by the softmax function mentioned in the earlier section. Presence of flatten layer makes the task of softmax much easier and helps achieve a more accurate prediction. The flatter matrix helps both in reducing the resource consumption of the model and in predicting accurate results. Our model also uses on flatten layer before softmax function.

The main platforms used for the build and implementation of the CNN model are **PYCHARM IDE** and **ANACONDA**. These interface units along with a standard PC can help us build and implement this model for easy detection of CNN. Let us understand the workflow and the process cycle of the CNN model in the following sections.

CHAPTER 4

WORKFLOW OF THE CNN MODEL FOR CHRONIC KIDNEY DISEASE DETECTION

The working of the model can be basically understood as 5 different stages of its implementation, basically the 5 modules of processing wherein at each stage, the modules move on enhancing the model's learning and prediction accuracy at each step. Let us understand the model's implementation as 5 different modules in the from the following sequence diagrams and in detail from the sections below

4.1 PROCESS FLOW OF THE MODEL

The following flowchart demonstrates the overall process flow of the model.

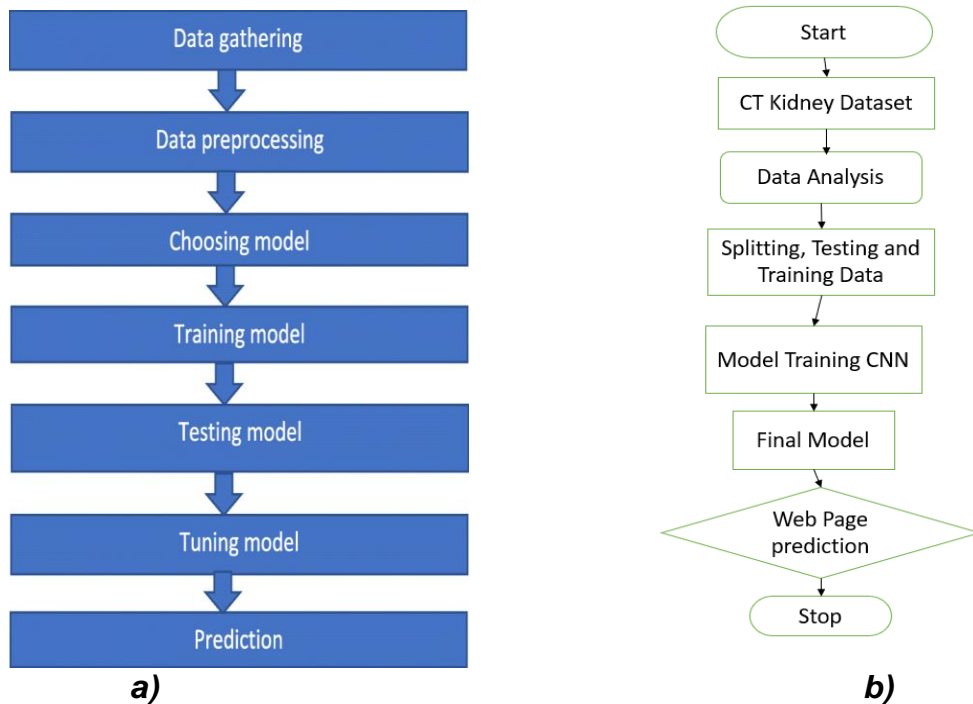


Fig 4.1 (a), (b) Flow diagram of the model's processing

Having understood the process flow from the figure 4.1, the sequence of process happening in the model from the sequence diagram given below.

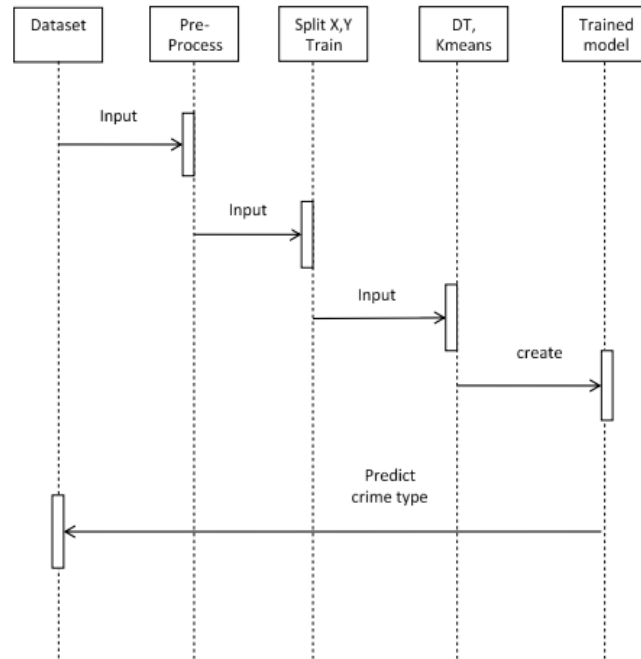


Fig 4.2 Sequence diagram of the model

With the sequence diagram describing the sequence flow of the model's working, let us try understanding the model's working in a detailed manner from the following sections.

4.1.1 Data Collection

The dataset was gathered via PACS (Picture archiving and communication system) at various hospitals in Dhaka, Bangladesh, where patients had already been identified as having kidney stones, cysts, tumours, or normal findings. With protocol for the whole abdomen and urogram, the Coronal and Axial cuts were both chosen from contrast and non-contrast examinations. Then, we prepared a batch of Dicom images of the region of interest for each radiological finding from the carefully chosen Dicom study, one diagnosis at a time. The Dicom photos were then converted to a lossless jpg image format after we removed each patient's information and meta data. After the conversion, a radiologist and a medical technologist again verified each image finding to reconfirm the correctness of the data.

The format of datasets used in this implementation are 2D images. The image datasets are divided into two major categories, training, and test datasets. To each division, a set of images of 4 classes namely, cyst, stone, normal and tumor are imported, and this classification is mainly used as the base dataset for the implementation of the model.

4.1.2 Data Preprocessing:

This module of the model's development cycle focuses at doing the prerequisite processing and formatting of the collected datasets from the previous module to make the input dataset understandable and acceptable to the model. Preprocessing here refers to performing all the transforms, configuration, reshaping and other basic changes to the datasets given as inputs to the model from the data collection.

Importance of data preprocessing:

Real time data collected from various physical conditions or from different sources of data are generally noisy, may involve disturbances, inaccuracies within or even the presence of duplicate entries as well is possible. To map those faulty entries from human collected or any other source of data, data preprocessing becomes important in any model's development cycle. Some commonly adopted data preprocessing methodologies are as follows:

- *Sampling*: Involves selecting a representative subset from each group of data.
- *Transformation*: Manipulates multiple raw data entries to produce a single input.
- *Denoising*: Involves in removing noise from the input dataset.
- *Feature extraction*: Involves in extracting specific attributes or characters from the input data significant for enhancing the model's performance.

Some of the most important steps involved in data preprocessing are:

(a) Data profiling:

In this step, the classes and nature of the collected dataset is analyzed. It aims

understanding the quality of the input data collected. The process starts with surveying the collected datasets and at the end mapping out which are the datasets relevant for the result along with the quality of the relevant dataset thus segregated.

(b) Data cleansing:

This step involves reconfiguring bad data from the segregated ones, removing duplicate data and balancing the missing data entries thus bringing in a balance in the available dataset.

(c) Data reduction:

This step involves in reducing the dataset's size by removing redundancies and with the help of component analysis find out the relevant quantity and quality of data and reduce it to such a way that it becomes easier for the model to understand and develop.

(d) Data transformation:

This step mainly classifies the collected input data into different classes (like training, testing and validation) as per the requirement of both the developer and the developing model.

(e) Data enrichment:

Here the transformed dataset is further enriched by various feature engineering techniques to achieve an optimal balance of the data distribution such that the working time of the model to train itself and come up with the accurate prediction is not exceeded beyond the expected one.

Some of the data preprocessing modules included for the optimization of the proposed model are:

- *Rescaling:*

Scaling or normalization of data is a general procedure in data preprocessing wherein the datasets are standardized into a standard format bringing them within a range to make the training process faster, accurate and efficient. The rescaling is set here to **1/255** bringing them within the range **[0,1]**.

- *Target size:*

Target size here refers to the final resized tuple which denotes the dimension of the input images given as dataset to the model. In this model, the target size is configured to a final dimension of **200x200** pixels and hence the model resizes the input data to the target size dimensions.

- *Color mode:*

Color mode refers to the color code taken by the model for its datasets and further processing. With CNN, two different color modes are possible namely the color ('RGB') format and the grayscale (Black and white) format. Since the collected input by itself is in grayscale, we are configuring the model to proceed with **grayscale** as the color mode throughout its development cycle.

- *Class mode:*

Class mode refers to the number of classes or categories the CNN model will use and accordingly classify the input thus provided. Two possible classes available are 'binary' and 'categorical' where binary refers to classification into 2 labels and categorical refers to classification into multiple labels. Since the input datasets we assume here are of 4 different classes, we configure the class mode as **'categorical'**.

- *Batch size:*

As described earlier, batch size refers to the number of images taken by the model as input for each cycle of process. In other words, it defines the number of images in each batch. Here, we are configuring the batch size as **100 images**.

4.1.3 Model Training:

Having done the collection and preprocessing of data, we now can train the model with the readily available datasets. We are Initializing a sequential model of CNN and adding up multiple layers to it using the Keras API with the TensorFlow backend based on the accuracy we intended to achieve.

Six convolutional layers with ReLU activation make up the model's architecture, which is followed by layers that down sample the feature maps called max-pooling layers. The feature maps are flattened after the convolutional layers and then run through a fully connected layer with 512 hidden units and a ReLU activation. To categorize the input images into one of the four classes, the output layer with 4 neurons and a softmax activation function is added last. It uses the 'rmsprop' optimizer and 'categorical_crossentropy' as the loss function, both of which are suited for multi-class classification jobs.

Precision, recall, accuracy, and loss are used as evaluation metrics for the model, and these metrics are defined in the METRICS variable that is used in the testing section to analyze the efficiency of the model.

The structure of the layers and the corresponding dimensions assumed for each layer along with its structural arrangement can be understood from the following table.

Table 4.1 Summary of the CNN model's layers

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 198, 198, 32)	320
max_pooling2d (MaxPooling2D)	(None, 99, 99, 32)	0
conv2d_1 (Conv2D)	(None, 97, 97, 32)	9248
max_pooling2d_1 (MaxPooling2D)	(None, 48, 48, 32)	0
conv2d_2 (Conv2D)	(None, 46, 46, 64)	18496
max_pooling2d_2(MaxPooling2D)	(None, 23, 23, 64)	0
conv2d (Conv2D)	(None, 198, 198, 32)	320
max_pooling2d (MaxPooling2D)	(None, 99, 99, 32)	0
conv2d_1 (Conv2D)	(None, 97, 97, 32)	9248
max_pooling2d_1 (MaxPooling2D)	(None, 48, 48, 32)	0
conv2d_2 (Conv2D)	(None, 46, 46, 64)	18496
max_pooling2d_2(MaxPooling2D)	(None, 23, 23, 64)	0
conv2d_3 (Conv2D)	(None, 21, 21, 64)	36928
conv2d_3 (Conv2D)	(None, 21, 21, 64)	36928

Now that the training and building of the model is complete, its efficiency of working and prediction accuracy is understood from its real time implementation using flask framework detailed in the section below

4.1.4 Web Implementation:

The final module explains the real time implementation of the developed model for effective usage in the practical world. This implementation is achieved using a web interface developed with the help of flask framework. Flask framework is the intermediate module used here to interlink the backend neural model developed with the frontend web interface developed using the leading frameworks HTML, CSS, and JavaScript.

Flask

A lightweight web framework for Python called Flask makes it simple and quick to create web apps. The Model-View-Controller (MVC) architecture pattern underlies how Flask operates. In this pattern, the model represents the data, the view the user interface, and the controller serves as a bridge between both.

Routes in Flask

Routes are the URLs that the user can visit in the web interface. We can define routes in Flask by using the **app. route ()** decorator and specifying the URL pattern as a parameter.

Frontend

The part of a website that a user first interacts with is called the front end. It includes every element that users interact with, including buttons, the navigation menu, text, colors, and styles. It also includes photographs, videos, graphs, and tables.

The procedure followed to implement flask framework are as follows:

- The front end was created using HTML, CSS, and JavaScript. Python web application development was done with Flask.
- Importing the Flask class was the first step in the procedure. After that, a

class instance was created.

- The name of the application's module or package was supplied as the '___name___' option. To know where to look for resources like templates and static files, Flask needs this information. This declaration was made as a result.
- Then, to tell Flask which URL should invoke our method, we utilize the route () decorator. The message that should be displayed in the user's browser is returned by this method.

These steps are done for the interlink of the model's framework with the web interface created as the frontend. The summary and workflow of the flask framework can be understood from the following diagram.

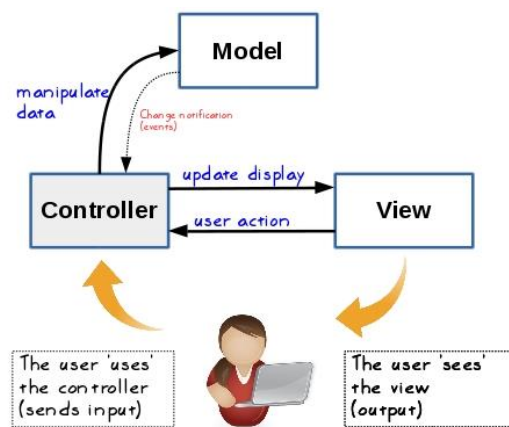


Fig 4.3 Flask Framework Architecture

As seen from the diagram above, usage of flask, a simple framework interlinked as a library or package with python, we can easily interlink the algorithm we formulated with the real-time interface thus making this development a user-serving tool.

4.2 PERFORMANCE METRICS:

The model's efficiency of working is understood by its analysis which can be accomplished using suitable performance analysis metrics detailed in the following sections.

Precision:

The proportion of positive instances relative to all projected positive instances is what is referred to as a model's accuracy. It is given by the formula:

$$\frac{TP}{TP + FP}$$

Recall:

The percentage of positive instances out of all actual positive instances is known as the true positive rate or recall of any model.

$$\frac{TP}{TP + FN}$$

Accuracy:

The ratio of the sum of true positive and true negative examples to the entire number of forecasts determines the accuracy of any model. This is one of the most used metrics to judge a model. The worse happens when classes are imbalanced. It is given by the formula:

$$\frac{TP + TN}{TP + FP + TN + FN}$$

Logarithmic Loss:

The way that Logarithmic Loss or Log Loss operates is by correcting incorrect classifications. The classifier must assign probabilities to each class for each sample when using Log Loss. Assuming there are N samples spread throughout M classes, the log loss is determined as follows:

$$LogarithmicLoss = \frac{-1}{N} \sum_{i=1}^N \sum_{j=1}^M y_{ij} * \log(p_{ij})$$

Where y_{ij} denotes whether or not sample i belongs to class j and p_{ij} is the likelihood that sample i does. Log Loss exists on the range $[0,1]$, and it has no upper bound. A log loss that is closer to 0 denotes precision.

Confusion matrix:

Confusion matrix is a common technique used in any CNN classification to understand the overall working of the model and it helps summarize the overall nature of prediction of the model.

When the classes of the model have unequal number of observations in each class, then classification accuracy might be misleading in such cases. In those instances, confusion matrix stands as a greater alternative to understand the overall working of the model, its accuracy and nature of predictions in each class and helpful to find out under what instances and in which classifications the model is going wrong.

Epoch Graph:

Epochs are basically defined as the number of iterations or cycles taken by the model to analyse the training dataset passed on to the model. The training dataset is usually divided into batches with certain number of data images in each batch. The number of epochs is generally declared as a hyper parameter and its value decides the total no of iterations the total training dataset moves in and gets processed by the model.

4.3 STANDARD

The IEEE standard adapted for this project is **IEEE 2801-2022**.

IEEE 2801-2022: Standard for the Performance and Safety Evaluation of Artificial Intelligence Based Medical Device:

Terminology

This suggested practice outlines pleasant methods for configuring a nice control system for information units used in artificial intelligence (AI) in medical devices. It covers the full lifecycle of managing a dataset, including but not limited to data collecting, switching, utilizing, storing, maintaining, and updating. This recommended practice aims to improve universal statistics excellence and develop standards for fine control of information units for scientific AI.

4.4 CONSTRAINTS AND TRADE-OFFS

Constraints

The constraints encountered during the development of the CNN model are as follows:

- *Insufficiency of data:*

Initially with considering textual datasets for processing, the Neural model started giving outputs with lesser accuracy due to insufficient quantity of data. This has been now resolved with taking 2D image datasets as input to the CNN model.

- *Less accuracy in prediction:*

When an attempt was made to implement the model with textual data, the result obtained did not show the expected accuracy. This led to the change of input dataset from textual format to images. After shifting to image datasets, the accuracy achieved was seen to be much better than the former case.

Trade-Offs

The possible trade-offs that can be faced with utilizing this tool are as below:

- *Memory consumption:*

Since image datasets are used as input datasets, the memory consumption by the datasets for storage and by the model for processing the datasets can be a bit more. Hence sufficient memory units and processors are essential.

- *Cost of implementation:*

With the usage of image dataset, the overall cost of initializing and implementing the model will be slightly elevated.

- *Power consumption:*

Due to pixel processing of images, the net power consumed by the model for the processing and prediction will be slightly more than that consumed with textual data.

Despite the compromise in the above-mentioned parameters, the **overall accuracy** of the model, its **efficiency of prediction** and the **net validity** in the result are **increased double fold** than the existing prediction models. In this way, a positive outcome is expected from this project work.

CHAPTER 5

RESULTS AND DISCUSSIONS

In this section, the model is tested on its prediction accuracy using the test and validation datasets kept classified as mentioned earlier in the data collection section and its work efficiency is measured in numbers as well using the analysis parameters considered in the training phase. Apart from using the test and validation datasets classified earlier, it is equally important to measure the model's accuracy of detection in each case. Among the commonly available analysis parameters, we are using 4 different analysis metrics which are precision, recall, accuracy, and loss.

DATASETS USED

The summary of the datasets collected and used in the training of the model is tabulated in the following table. The number of images in each class and division is summarized in the following table. (Table 5.1)

Table 5.1 Summary of datasets collected

Dataset nature Image Class	Training dataset (No of images)	Validation Dataset (No of images)	Test Dataset (No of images)
Cyst	2000	800	800
Normal	3000	1000	900
Stone	500	400	300
Tumor	1000	550	550

Now that the number of images in each class is understood from the table 5.1, a sample of images from each class is shown in the following figures.

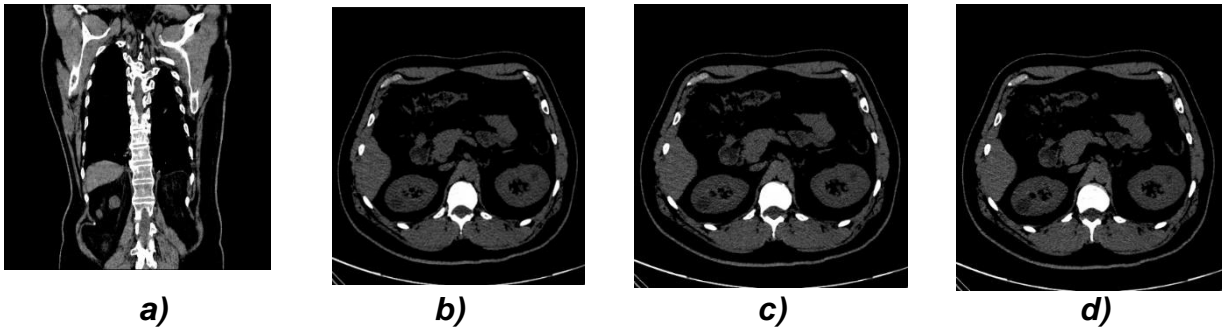


Fig 5.1.(a), (b), (c), (d) Cyst samples

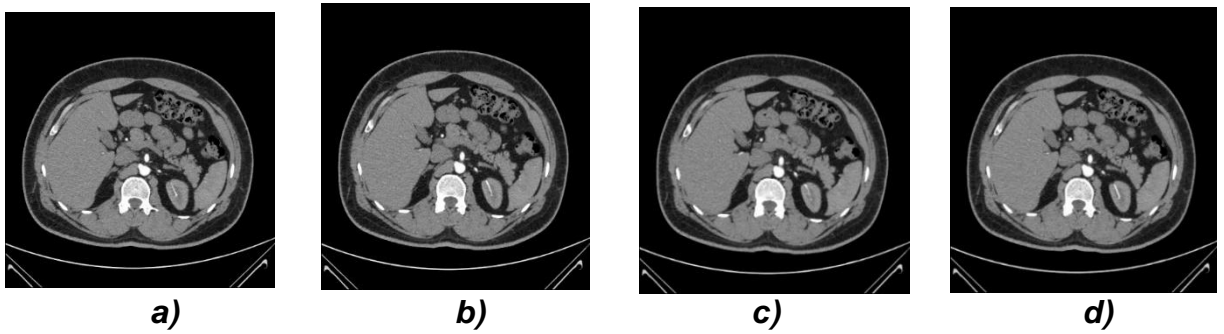


Fig 5.2.(a), (b), (c), (d) Normal samples

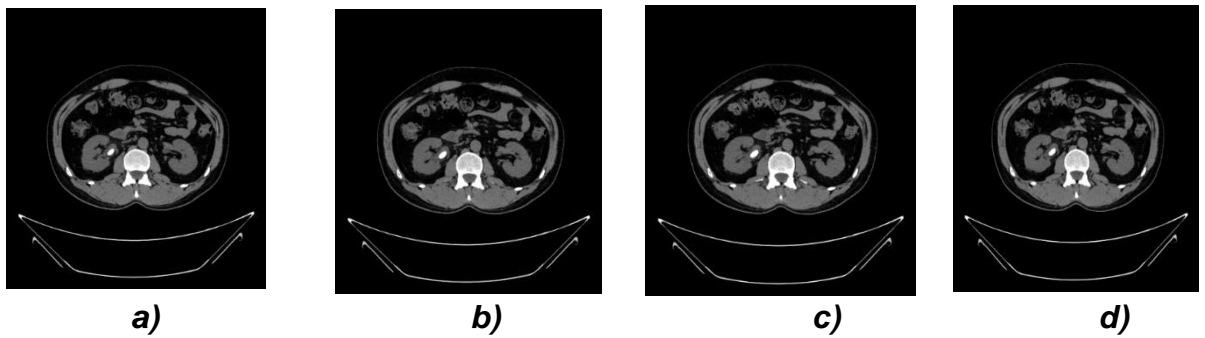


Fig 5.3.(a), (b), (c), (d) Stone samples

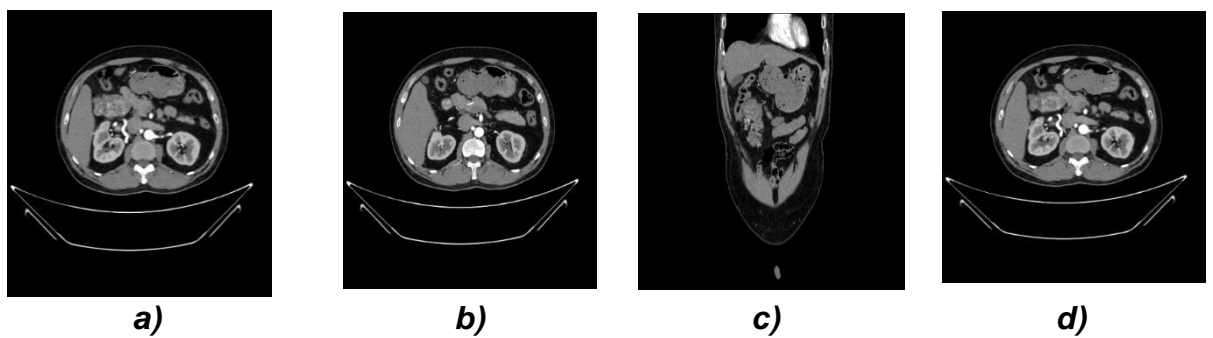


Fig 5.4.(a), (b), (c), (d) Tumor samples

5.1 MODEL PERFORMANCE ANALYSIS

The performance measures described in the preceding chapter are used to analyze the model's performance. These metrics are calculated for each and every class, and the accompanying table provides an overview of the analysis.

Table 5.2 Model Performance analysis

Class of image	Precision	Recall	Accuracy	Loss
Cyst	98.58	97.54	97.07	26.98
Normal	98.47	98.43	98.64	5.78
Stone	99.67	99.01	99.87	11.23
Tumor	99.32	98.72	99.03	20.78

The overall performance analysis of the model can be summarized as follows:

OVERALL PRECISION: **98.7435**.

OVERALL RECALL: **98.0852**.

OVERALL ACCURACY: **98.8817**.

OVERALL LOGARITHMIC LOSS: **0.0079**.

The overall confusion matrix achieved with the model for CKD classification is as follows:

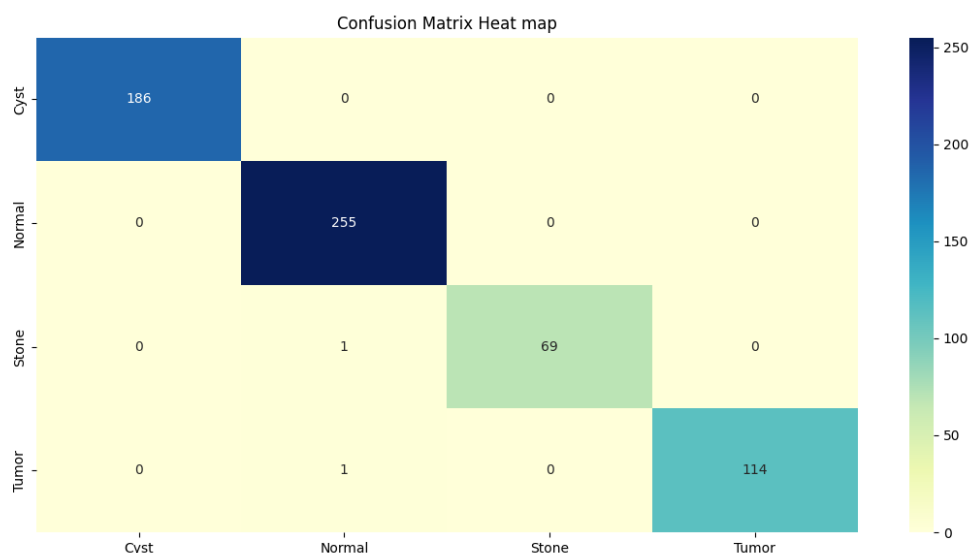


Fig 5.5 Confusion matrix of the model

The epochs considered for the development of our model and their process at each iteration can be understood from the following table.

Table 5.3 Epoch summary of the model

Epoch Number	Training				Validation			
	Accuracy	Precision	Recall	Loss	Accuracy	Precision	Recall	Loss
1	0.5489	0.6491	0.3229	1.1325	0.5484	0.6152	0.4177	1.1297
2	0.7427	0.7906	0.6857	0.6531	0.8935	0.9064	0.8742	0.2698
3	0.9313	0.9373	0.9250	0.2108	0.9871	0.9871	0.9871	0.0580
4	0.9687	0.9706	0.9672	0.0991	0.9968	0.9968	0.9968	0.0125
5	0.9844	0.9858	0.9839	0.0629	0.9968	0.9968	0.9968	0.0079

The table 5.3 also explains the performance efficiency reached at every epoch implementation. For this model, a total of 4 epochs are configured and the above table summarizes the values of the performance metrics at every epoch of the model's training cycle. The corresponding epoch graph plot comparing the results between the training and the validation datasets at each epoch are shown in the following figure.

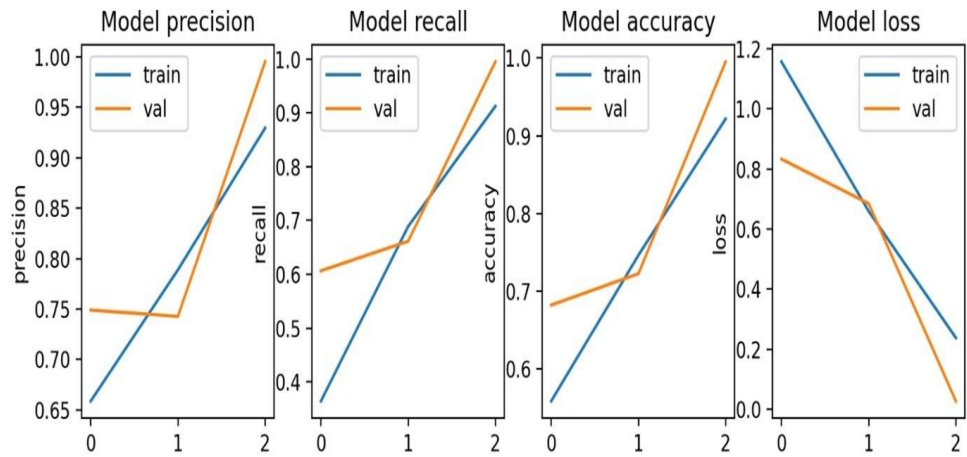
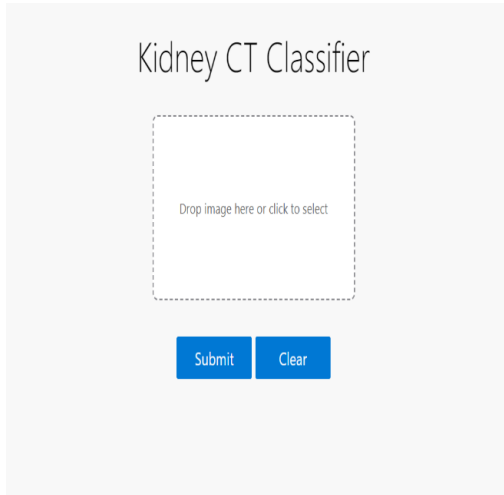


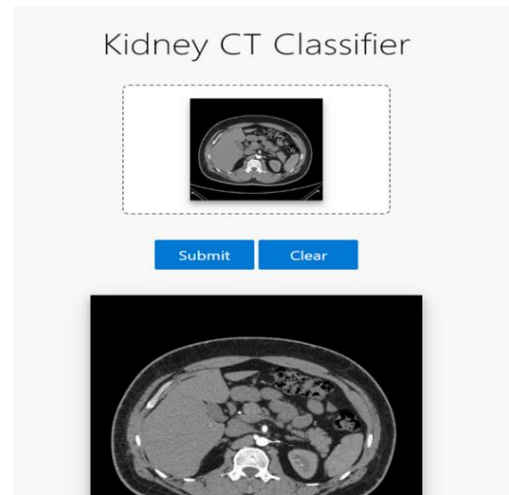
Fig 5.6 Epoch graphs of the model

Web interface outputs:

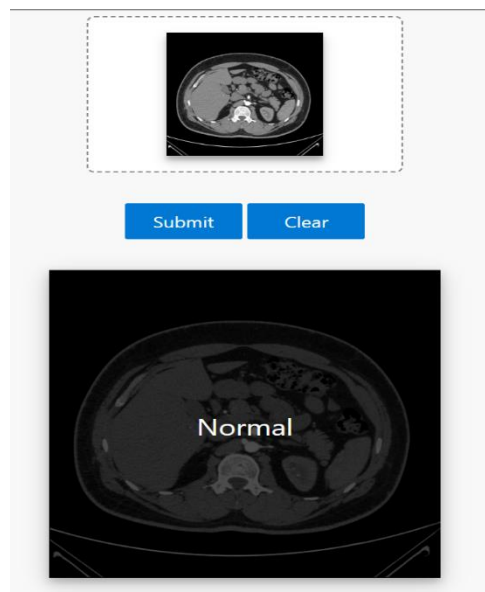
The following images show the real time implementation of the tool using a web interface, making it a user-friendly and interactive system.



(a)



(b)



(c)

Fig 5.7 (a), (b), (c) Web interface outputs

Having a glance on the efficiency of the model's working, its overall architecture can be understood from the following figure:

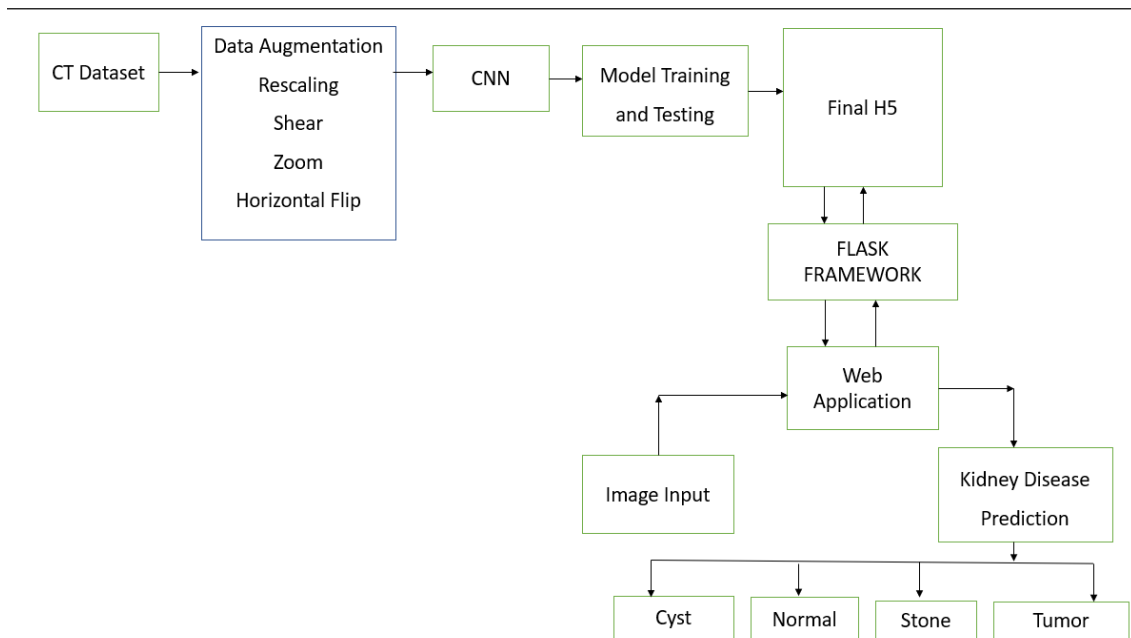


Fig 5.8 Overall Architecture of the model

5.3 SUMMARY OF THE MODEL

The summary of the model's implementation and its working can be jotted down as the following points:

- Collection of suitable image datasets for the model.
- Preprocessing of the collected datasets for easier processing and understanding of the model.
- Training of the model with the collected datasets along with configuring the basic and the most important hyper parameters which will decide the latter efficiency of the model.
- Building the user interacting web tool using flask interface framework for interlinking the frontend website and the backend working CNN model.
- Testing the model with test datasets and validating the working of the model after analysis of the same using the chosen performance analysis metrics.
- Implementing the model in the practical world for interactive help to its users.

CHAPTER 6

CONCLUSION AND FUTURE WORK

Having briefed the detailed working of the model, the performance metrics considered for its evaluation, and the resultant efficiency and performance scores achieved, this chapter focuses on deriving an overall conclusion on the problem statement assumed at the beginning of the research and details the future enhancement that are possible with the current state of implementation.

6.1 CONCLUSION

Kidney abnormality is a major health concern globally, and the use of deep learning models in kidney disease diagnosis has shown promise in recent years. Specifically, in the detection of kidney cysts, stones, and tumours, a lightweight customized convolution neural network was proposed, which outperformed various cutting-edge techniques and attained an accuracy of 99.52%. The proposed model also provides better interpretive power, enabling clinicians to understand the specific decisions made by the model. Additionally, the deployment of the model using Flask framework for web page prediction makes it accessible to clinicians and patients, allowing for more efficient and accurate kidney disease diagnosis. Future research in this area should focus on expanding the scope of the model to detect other kidney abnormalities and on validating the model's performance on larger datasets.

6.2 FUTURE WORK

Future research in this area should focus on

- Expanding the scope of the model to detect other kidney abnormalities and on validating the model's performance on larger datasets.
- Expanding the model's capabilities to identify other kidney problems.

REFERENCES

- [1] Adeola Ogunleye et al., XGBoost Model for Chronic Kidney Disease Diagnosis,
- [2] IEEE/ACM Transactions on Computational Biology and Bioinformatics, Volume:17,2019, Issue: 6, pp.2131 – 2140.
- [3] Adriano Luiz Ammirati, Chronic Kidney Disease, Rev assoc med bras,2020 66(SUPPL 1): S3-S9.
- [4] Angela C Webster et al. (Chronic Kidney Disease), The Lancet, Volume 389, Issue10075, 25–31 2017, Pages 1238-1252.
- [5] Dervla M. Connaughton et al. Monogenic causes of Chronic Kidney Disease inadults, Clinical investigation, 2019, volume 95, issue 4, p914-928.
- [6] Ebrahime Mohammed Senan et al., Diagnosis of CKD Using Effective Classification Algorithms and Recursive Feature Elimination Techniques, Journal ofHealthcare Engineering Volume 2021, Article ID 1004767, 10 pages.
- [7] Gabriel R. Vásquez-Morales et al., Explainable Prediction of Chronic Renal Disease in the Colombian Population Using Neural Networks and Case-Based Reasoning, IEEEAccess - Special section on data-enabled intelligence for digital health, VOLUME 7, 2019, PP (99):1-1.
- [8] Hui-Ju Tsai et al., Environmental Pollution and Chronic Kidney Disease,2021, Int JMed Sci. 2021; 18(5): 1121–1129.
- [9] Iliyas Ibrahim Iliyas et al. Prediction of Chronic Kidney Disease Using Deep Neural Network, ArXiv-eess-Electrical Engineering and Systems Science,2020.
- [10] Jacek Rysz et al. Novel Biomarkers in the Diagnosis of Chronic Kidney Diseaseand the Prediction of Its Outcome, Int. J. Mol. Sci. 2017, 18(8), 1702.

- [11] Ji-Cheng Lv et al., Prevalence and Disease Burden of Chronic Kidney Disease, Renal Fibrosis: Mechanisms and Therapies, 2019, pp.3–15.
- [12] Jiongming Qin et al., A Machine Learning Methodology for Diagnosing Chronic Kidney Disease, IEEE Access (Volume: 8), 2019, pp. 20991 – 21002.
- [13] Kamyar Kalantar-Zadeh et al., Chronic Kidney Disease, The Lancet, Volume 398, 2021, Issue 10302, Pages 786-802.
- [14] LiHe et al., Impact of high, low, and non-optimum temperatures on Chronic Kidney Disease in a changing climate, 1990–2019: A global analysis, Environmental Research, Volume 212, 2022, Part A, 113172.
- [15] Mehdi Hossein Zadeh et al., A diagnostic prediction model for chronic kidney disease in internet of things platform, 2020, Multimed Tools Appl 80, 16933–16950.
- [16] Mohamed Elhoseny et al., Intelligent Diagnostic Prediction and Classification System for Chronic Kidney Disease, Scientific Reports, volume 9, 2019, Article number: 9583.
- [17] Muthiah.M. A, Logashanmugam. E, and Nandhitha.N.M., Feasibility of Convolutional Neural Networks (CNN) for Content Based Image Retrieval of Images from Hadoop, Smart Intelligent Computing and Communication Technology.
- [18] Ning Shang Et al., Medical records-based Chronic Kidney Disease phenotype for clinical care and “big data” observational and genetic studies), npj Digital Medicine volume 4, Article number: 70 (2021).
- [19] Njoud Abdullah Almansour et al., Neural network and support vector machine for the prediction of chronic kidney disease: A comparative study, Comput Biol Med, 2019 Jun; 109:101-111.

[20] Olivier J. Wouters et al., Early Chronic Kidney Disease: diagnosis, management and models of care, *Nature Reviews Nephrology* volume 11, 2015, pages 491 – 502.

[21] Paola Romagnani et al., Chronic Kidney Disease, *Nature Reviews Disease Primers*, volume 3, 2017, Article number: 17088.

[22] Rajiv Agarwal et al., Investigating new treatment opportunities for patients with Nephrology Dialysis Transplantation, *Nephrology Dialysis Transplantation*, Volume 37, Issue 6, 2022, Pages 1014–1023.

[23] Roser Torra et al., Genetic kidney diseases as an underrecognized cause of Chronic Kidney Disease: the key role of international registry reports, *Clinical Kidney Journal*, Volume 14, Issue 8, 2021, Pages 1879–1885.

[24] Rui Gao et al., Recognition of chronic renal failure based on Raman spectroscopy and convolutional neural network, *Photodiagnosis and Photodynamic Therapy*, Volume 34, June 2021, 102313.

[25] Vijendra Singh et al., A Deep Neural Network for Early Detection and Prediction of Chronic Kidney Disease, *Diagnostics* 2022, 12(1), 116.

[26] Yury E. Glazyrin et al., Proteomics-Based Machine Learning Approach as an Alternative to Conventional Biomarkers for Differential Diagnosis of Chronic Kidney Diseases, *Int J Mol Sci*, 2020;21(13):4802.

[27] Zainuri Saringat et al., Comparative analysis of classification algorithms for chronic kidney disease diagnosis, *Bulletin of Electrical Engineering and Informatics*, Vol.8, No.4, 2019, pp. 1496~1501.

APPENDIX

CODE:

//INDEX HTML CODE FOR WEB INTERFACE DESIGN

```
<div class="main">
  <div class="title">
    <h3>Kidney CT Classifier</h3>
    <!-- <p>
      <small>A web app demo</small>
    </p> -->
  </div>
  <div class="panel">
    <input id="file-upload" class="hidden" type="file" accept="image/x-
png,image/gif,image/jpeg" />
    <label for="file-upload" id="file-drag" class="upload-box">
      <div id="upload-caption">Drop image here or click to select</div>
      <img id="image-preview" class="hidden" />
    </label>
  </div>
  <div style="margin-bottom: 2rem;">
    <input type="button" value="Submit" class="button" onclick="submitImage();" />
    <input type="button" value="Clear" class="button" onclick="clearImage();" />
  </div>
  <div id="image-box">
    <img id="image-display" />
    <div id="pred-result" class="hidden"></div>
    <svg id="loader" class="hidden" viewBox="0 0 32 32" width="32" height="32">
      <circle id="spinner" cx="16" cy="16" r="14" fill="none"></circle>
    </svg>
  </div>
</div>
```

//JAVASCRIPT CODE FOR WEBPAGE OPTIMIZATION

```
var fileDrag = document.getElementById("file-drag");
var fileSelect = document.getElementById("file-upload");

// Add event listeners

fileDrag.addEventListener("dragover", fileDragHover, false);
fileDrag.addEventListener("dragleave", fileDragHover, false);
fileDrag.addEventListener("drop", fileSelectHandler, false);
fileSelect.addEventListener("change", fileSelectHandler, false);

function fileDragHover(e) {

    // prevent default behaviour

    e.preventDefault();

    e.stopPropagation();

    fileDrag.className = e.type === "dragover" ? "upload-box dragover" : "upload-
box";

}

function fileSelectHandler(e) {

    // handle file selecting

    var files = e.target.files || e.dataTransfer.files;

    fileDragHover(e);

    for (var i = 0, f; (f = files[i]); i++) {

        previewFile(f);

    }

}

var imagePreview = document.getElementById("image-preview");
var imageDisplay = document.getElementById("image-display");
```



```

var uploadCaption = document.getElementById("upload-caption");
var predResult = document.getElementById("pred-result");
var loader = document.getElementById("loader");

function submitImage() {
    // action for the submit button
    console.log("submit");

    if (!imageDisplay.src || !imageDisplay.src.startsWith("data")) {
        window.alert("Please select an image before submit.");
        return;
    }

    loader.classList.remove("hidden");
    imageDisplay.classList.add("loading");

    // call the predict function of the backend
    predictImage(imageDisplay.src);
}

```

```

function clearImage() {
    // reset selected files
    fileSelect.value = "";

    // remove image sources and hide them
    imagePreview.src = "";
    imageDisplay.src = "";
    predResult.innerHTML = "";

    hide(imagePreview);
    hide(imageDisplay);
}

```

```

hide(loader);

hide(predResult);

show(uploadCaption);

imageDisplay.classList.remove("loading");
}

function previewFile(file) {

  // show the preview of the image

  console.log(file.name);

  var fileName = encodeURIComponent(file.name);

  var reader = new FileReader();

  reader.readAsDataURL(file);

  reader.onloadend = () => {

    imagePreview.src = URL.createObjectURL(file);

    show(imagePreview);

    hide(uploadCaption);

    // reset

    predResult.innerHTML = "";

    imageDisplay.classList.remove("loading");


    displayImage(reader.result, "image-display");

  };
}

const getBase64FromUrl = async (url) => {

  const data = await fetch(url);

  const blob = await data.blob();

```

```

return new Promise((resolve) => {

  const reader = new FileReader();

  reader.readAsDataURL(blob);

  reader.onloadend = () => {

    const base64data = reader.result.replace("data:", "")

      .replace(/^.+/, "");

    resolve(base64data);

  }

});

}

async function predictImage() {

  var image_preview = document.getElementById('image-preview');

  var base64Data = await getBase64FromUrl(image_preview.src);

  var base64DataString = String(base64Data);

  var requestBody = JSON.stringify({

    "data":base64DataString

  });

  console.log(requestBody);

  fetch("/predict", {

    method: "POST",

    headers: {

      "Content-Type": "application/json"

    },

    body: requestBody

  })

```

```

.then(resp => {
  if (resp.ok)
    resp.json().then(data => {
      displayResult(data);
    });
})

.catch(err => {
  console.log("An error occurred", err.message);
  window.alert("Oops! Something went wrong.");
});
}

function displayImage(image, id) {
  // display image on given id <img> element
  let display = document.getElementById(id);
  display.src = image;
  show(display);
}

function displayResult(data) {
  // display the result
  // imageDisplay.classList.remove("loading");
  hide(loader);
  predResult.innerHTML = data.result;
  show(predResult);
}

function hide(el) {

```

```

// hide an element

el.classList.add("hidden");

}

function show(el) {

// show an element

el.classList.remove("hidden");

}

```

//CODE FOR MODEL CONFIGURATION AND WORKFLOW

```

from tensorflow.keras.preprocessing.image import ImageDataGenerator ,
load_img , img_to_array
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, Flatten, MaxPool2D, Dense
import matplotlib.pyplot as plt

import numpy as np
#from skimage import transform
#import splitfolders
from sklearn.metrics import confusion_matrix
from sklearn.metrics import precision_recall_fscore_support
import seaborn as sns

import splitfolders
splitfolders.ratio("CT-KIDNEY-DATASET-Normal-Cyst-Tumor-Stone/CT-KIDNEY-
DATASET-Normal-Cyst-Tumor-Stone/CT-KIDNEY-DATASET-Normal-Cyst-
Tumor-Stone/CT-KIDNEY-DATASET-Normal-Cyst-Tumor-
Stone",output="./dataset", seed=7, ratio=(0.90,0.050, 0.050))

train_datagen = ImageDataGenerator(rescale=1/255)
valid_datagen = ImageDataGenerator(rescale=1/255)
test_datagen = ImageDataGenerator(rescale=1/255)

```

```
train_dataset = train_datagen.flow_from_directory('dataset/train/',
                                                  target_size=(200, 200),
                                                  color_mode='grayscale',
                                                  class_mode='categorical',
                                                  batch_size=100,
                                                  )
```

```
test_dataset = test_datagen.flow_from_directory('dataset/test/',
                                                  target_size=(200, 200),
                                                  class_mode='categorical',
                                                  color_mode='grayscale',
                                                  batch_size=100,
                                                  shuffle=False
                                                  )
```

```
valid_dataset = valid_datagen.flow_from_directory('dataset/val/',
                                                  target_size=(200, 200),
                                                  class_mode='categorical',
                                                  batch_size=100,
                                                  color_mode='grayscale',
                                                  )
```

```
model = Sequential()
```

```
model.add(Conv2D(32, (3,3), activation='relu',
input_shape=train_dataset.image_shape))
model.add(MaxPool2D(2))
model.add(Conv2D(32, (3,3), activation='relu'))
model.add(MaxPool2D(2))
model.add(Conv2D(64, (3,3), activation='relu'))
model.add(MaxPool2D(2))
model.add(Conv2D(64, (3,3), activation='relu'))
model.add(MaxPool2D(2))
```

```
model.add(Conv2D(128, (3,3), activation='relu'))
```

```

model.add(MaxPool2D(2))
model.add(Conv2D(128, (3,3), activation='relu'))
model.add(MaxPool2D(2))
model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dense(4, activation='softmax'))
model.summary()
import keras

METRICS = [
    'accuracy',
    keras.metrics.Precision(name='precision'),
    keras.metrics.Recall(name='recall')
]
model.compile(optimizer='rmsprop', loss='categorical_crossentropy',
metrics=METRICS)

Info = model.fit(train_dataset,
                 validation_data=valid_dataset,
                 epochs=3,
                 )

fig, ax = plt.subplots(1, 4, figsize=(20, 3))
ax = ax.ravel()
for i, met in enumerate(['precision', 'recall', 'accuracy', 'loss']):
    ax[i].plot(Info.history[met])
    ax[i].plot(Info.history['val_' + met])
    ax[i].set_title('Model {}'.format(met))
    ax[i].set_xlabel('epochs')
    ax[i].set_ylabel(met)
    ax[i].legend(['train', 'val'])
predictions = model.predict(test_dataset)
diseases_labels = []

for key, value in train_dataset.class_indices.items():

```

```

    diseases_labels.append(key)
def evaluate(actual, predictions):
    pre = []
    for i in predictions:
        pre.append(np.argmax(i))
    accuracy = (pre == actual).sum() / actual.shape[0]
    print(f'Accuracy: {accuracy}')
    precision, recall, f1_score, _ = precision_recall_fscore_support(actual, pre,
    average='macro')
    print(f'Precision: {precision}')
    print(f'Recall: {recall}')
    print(f'F1_score: {f1_score}')
    fig, ax = plt.subplots(figsize=(20,20))
    conf_mat = confusion_matrix(actual, pre)
    sns.heatmap(conf_mat, annot=True, fmt='.0f', cmap="YlGnBu",
    xticklabels=diseases_labels, yticklabels=diseases_labels).set_title('Confusion
    Matrix Heat map')
    plt.show()
    evaluate(test_dataset.classes, predictions)
    model.evaluate(test_dataset)

```

//CODE FOR WEB INTERFACE IMPLEMENTATION

```

# Flask
from flask import Flask, redirect, url_for, request, render_template, Response,
jsonify, redirect
from werkzeug.utils import secure_filename
from gevent.pywsgi import WSGIServer

# TensorFlow and tf.keras
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.applications.imagenet_utils import preprocess_input,

```



```

decode_predictions
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
# from tensorflow.keras.applications.mobilenet_v2 import MobileNetV2
# Some utilities
import numpy as np
from util import base64_to_pil
# Declare a flask app
app = Flask(__name__)
# model = MobileNetV2(weights='imagenet')
# Model saved with Keras model.save()
MODEL_PATH = 'models/KidneyDiseasesModel.h5'
model = load_model(MODEL_PATH)
model.make_predict_function() # Necessary
print('loading model & Start serving...')
print('Model loaded. Check http://127.0.0.1:5000/')
def model_predict(img, Model):
    img = img.resize((200, 200))
    img_gray = img.convert('L')
    test_image = image.img_to_array(img_gray)
    test_image = np.expand_dims(test_image, axis=0)
    # Be careful how your trained model deals with the input
    # otherwise, it won't make correct prediction!
    # x = preprocess_input(test_image, mode='tf')
    preds = Model.predict(test_image)
    return preds
def determine_case(x):
    rtn = ""
    if x == 0:
        rtn = "Cyst"
    elif x == 1:
        rtn = "Normal"
    elif x == 2:
        rtn = "Stone"

```

```

elif x == 3:
    rtn = "Tumor"
    return rtn
@app.route('/', methods=['GET'])
def index():
    # Main page
    return render_template('index.html')
@app.route('/predict', methods=['GET', 'POST'])
def predict():
    if request.method == 'POST':
        print('-----')
        print('incoming request :')
        # Get the image from post request
        img = base64_to_pil(request.json['data'])
        # Making the prediction
        prediction = model_predict(img, model)[0]
        index_of_max = np.argmax(prediction)
        case = determine_case(index_of_max)
        print('prediction : ', case)
        print('-----')
        # Serialize the result, you can add additional fields
        return jsonify(result=case)
    return None
if __name__ == '__main__':
    # app.run(port=5002, threaded=False)
    # Serve the app with gevent
    http_server = WSGIServer(('0.0.0.0', 5000), app)
    http_server.serve_forever()

```