



# KIDNEY DISEASE DETECTION

**MD EKRAMUDDIN**  
**ROLL NO. CSE214044**  
**MD SHAHID**  
**ROLL NO. CSE214057**

**MINOR PROJECT PRESENTATION**  
**4TH YEAR : 7TH SEM**  
**TOPIC : KIDNEY DISEASES DETECTION**  
**USING DEEP LEARNING MODEL**

*Under guidance of*  
***Dr. Souvik Sengupta***  
*Associate Professor at Aliah University*  
*[CSE Department]*

# OVERVIEW



## Objective

Develop a robust machine learning framework to enhance the accuracy of kidney disease detection.



## Methodology

Integrated a function to preprocess and classify new kidney CT scan images using the trained model, outputting the predicted disease label



## Dataset

The dataset includes kidney-related medical imaging data, categorized into four classes :- Normal-Cyst-Tumor-Stone



## Results

The models demonstrated significant improvements in accuracy and performance metrics, leading to better predictions.



## Future Work

Expand the dataset to include diverse imaging modalities for better generalization. Focus on integrating additional preprocessing methods to handle complex imaging noise.

# INTRODUCTION

## KIDNEY DISEASES: PREVALENCE, IMPACT, AND THE NEED FOR EARLY DETECTION

KIDNEY DISEASES AFFECT MILLIONS WORLDWIDE, WITH CHRONIC KIDNEY DISEASE (CKD) BEING A LEADING GLOBAL HEALTH BURDEN.

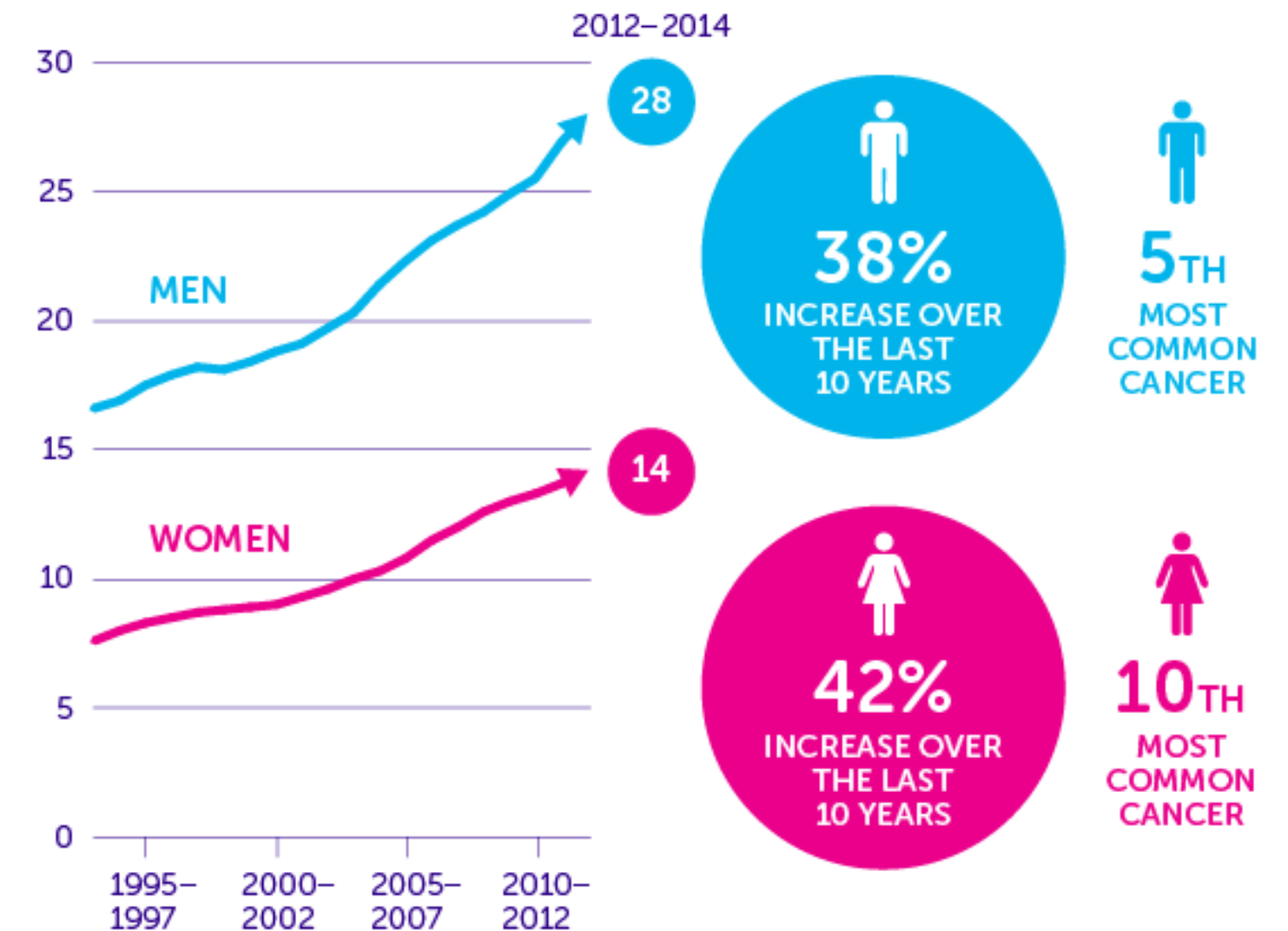
EARLY DETECTION IS CRITICAL, AS CKD OFTEN PROGRESSES SILENTLY UNTIL ADVANCED STAGES, LEADING TO SEVERE COMPLICATIONS LIKE KIDNEY FAILURE, CARDIOVASCULAR DISEASES, OR DEATH.

THE LACK OF EARLY SYMPTOMS MAKES TIMELY DIAGNOSIS CHALLENGING, EMPHASIZING THE IMPORTANCE OF ADVANCED DIAGNOSTIC TOOLS FOR EARLY-STAGE DETECTION.

## RISING KIDNEY CANCER RATES

### KIDNEY CANCER INCIDENCE RATES

PER 100,000 MEN AND WOMEN, 3 YEAR ROLLING AVERAGES, UK 1993–2014



Source: [cruk.org/cancerstats](http://cruk.org/cancerstats)

LET'S BEAT CANCER SOONER  
[cruk.org](http://cruk.org)



# WHY DEEP LEARNING FOR KIDNEY DISEASE DETECTION ?

- **ENHANCED DETECTION ACCURACY:**
- **AUTOMATION:**
- **HANDLING BIG DATA:**
- **FEATURE EXTRACTION:**
- **REAL-TIME APPLICATION:**

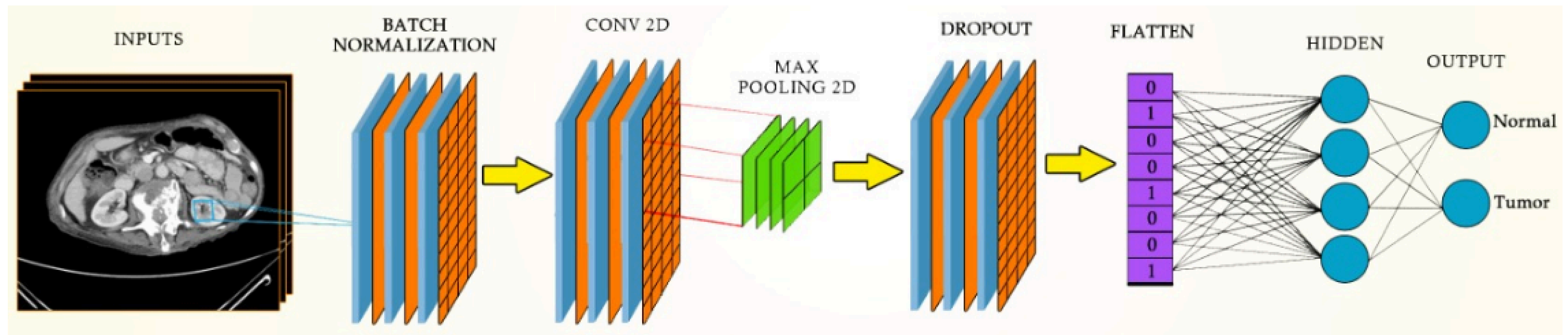
DETECT ABNORMALITIES WITH HIGHER PRECISION THAN TRADITIONAL METHODS.

ENSURING CONSISTENT AND REPRODUCIBLE RESULTS.

PROCESS LARGE VOLUMES OF IMAGING DATA

TECHNIQUES LIKE CLAHE ENHANCE IMAGE CONTRAST

DIAGNOSTIC SUPPORT, STREAMLINING CLINICAL WORKFLOWS AND ENABLING TIMELY INTERVENTIONS.





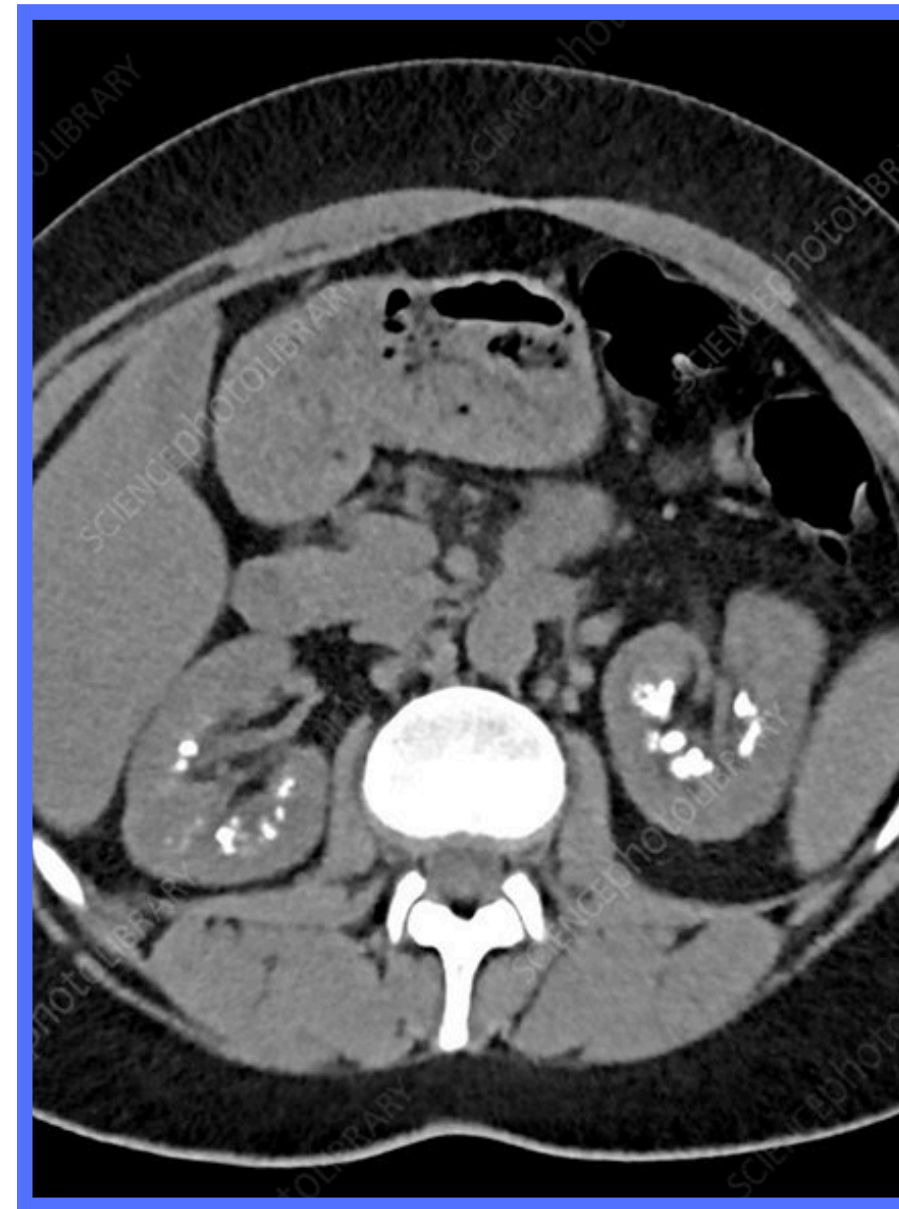
# IMAGING MODALITIES X-RAY & CT-SCAN

**X-RAY:** ARE WIDELY USED AS A PRELIMINARY DIAGNOSTIC TOOL IN MEDICAL IMAGING DUE TO THEIR AFFORDABILITY AND ACCESSIBILITY.

LIMITATION: THEY PROVIDE 2D IMAGES, WHICH MAY MISS CRITICAL DETAILS OR UNDERLYING ABNORMALITIES.

**CT SCAN:** SCANS OFFER A MORE ADVANCED IMAGING OPTION, PROVIDING HIGH-RESOLUTION, CROSS-SECTIONAL, AND 3D IMAGES OF THE KIDNEYS AND SURROUNDING STRUCTURES.

This project utilizes CT scan images acquired from a Siemens SOMATOM scanner.



**CT- SCAN :** Top View  
detecting stone in Kidney

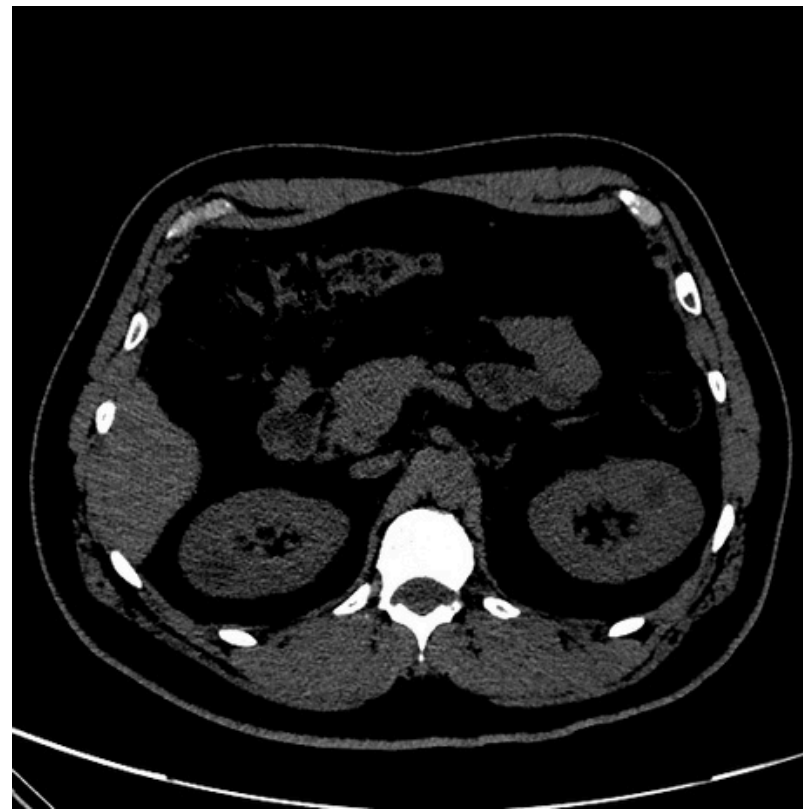


**X Ray :** Side View, not  
suitable for early detection.

# IMAGING MODALITIES : CT-SCAN [Classes]



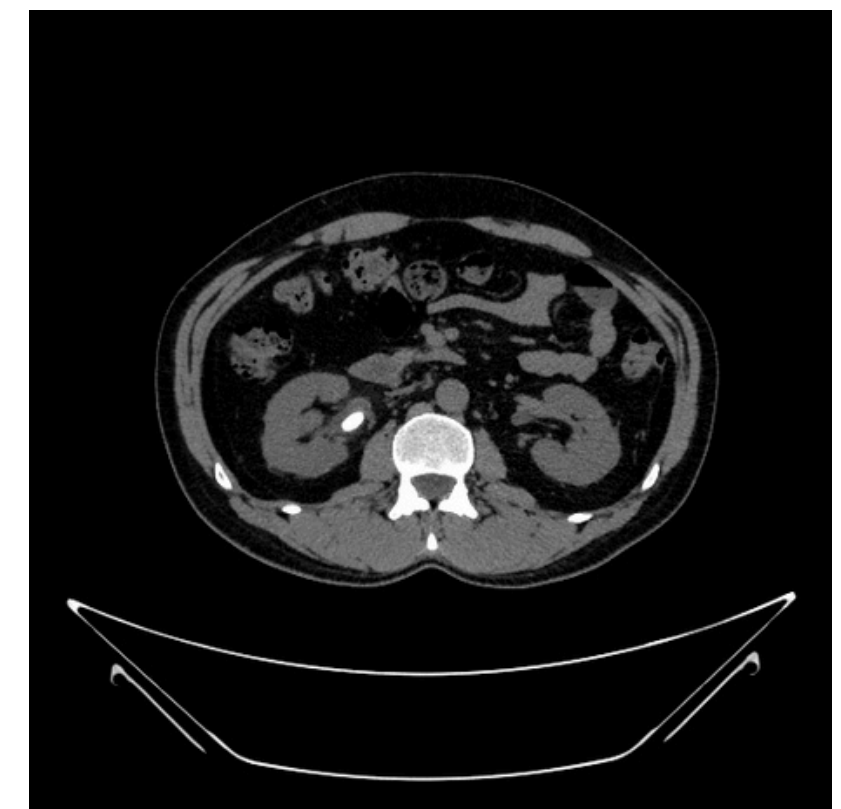
**TUMOR**



**CYST**



**NORMAL**



**STONE**

- **Normal** : Healthy kidneys without abnormalities in structure, size, or function.
- **Cysts** : Fluid-filled sacs that form on or within the kidney. Kidney cysts are generally benign but may indicate polycystic kidney disease (PKD) in some cases.
- **Tumor** : Abnormal growths in the kidney, which may be benign (e.g., angiomyolipomas) or malignant
- **Stones** : Hard mineral and salt deposits that form within the kidneys, causing pain and potential blockage.

# CHALLENGES OF TRADITIONAL METHOD AND SOLUTIONS

## CHALLENGES

- LACK OF STANDARDIZED DATASETS
- NOISE IN IMAGING DATA:
- DIFFICULTY IN EARLY-STAGE DETECTION:

## SOLUTION

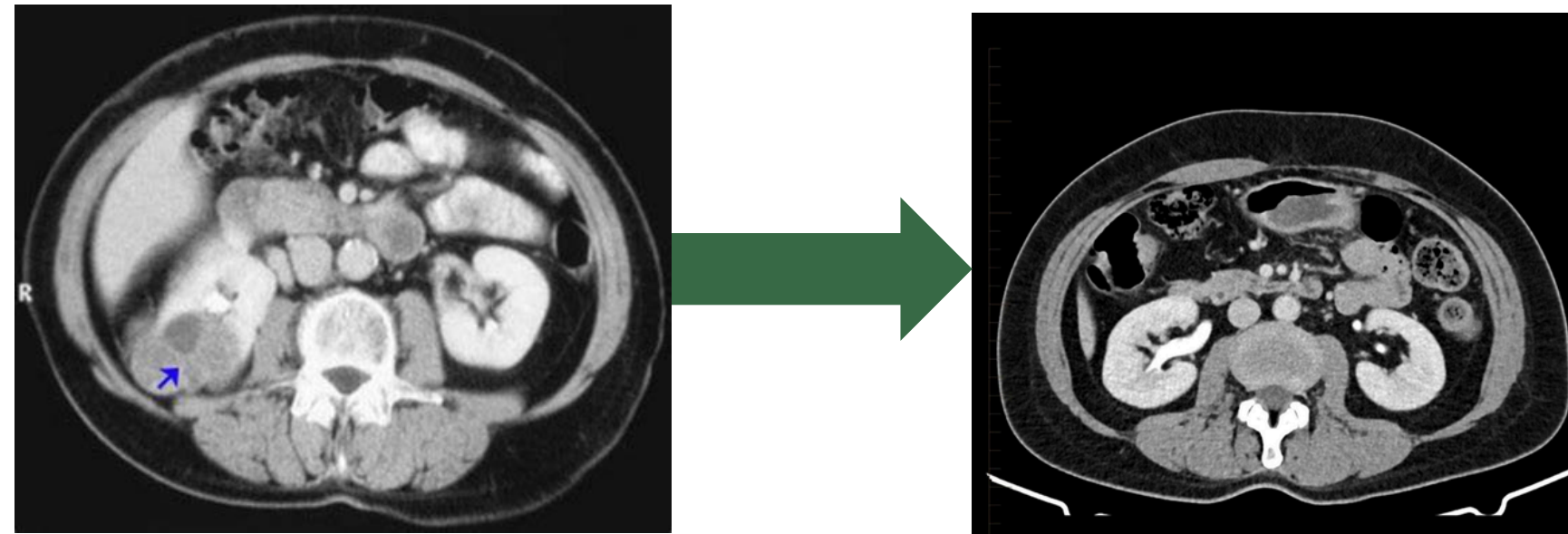
- THE CODE USES DATASET PREPROCESSING TECHNIQUES LIKE CLAHE (CONTRAST LIMITED ADAPTIVE HISTOGRAM EQUALIZATION) TO STANDARDIZE THE QUALITY OF IMAGING DATA AND IMPROVE FEATURE EXTRACTION,
- CLAHE ENHANCES IMAGE CONTRAST BY REDUCING NOISE AND HIGHLIGHTING ESSENTIAL FEATURES IN THE KIDNEY IMAGING DATA, ENABLING THE MODEL TO DETECT ABNORMALITIES MORE EFFECTIVELY.
- THE PROJECT INCORPORATES STATE-OF-THE-ART MODELS WHICH ARE HIGHLY CAPABLE OF EXTRACTING INTRICATE PATTERNS FROM IMAGING DATA. THESE MODELS ARE TRAINED ON PREPROCESSED DATASETS, IMPROVING ACCURACY IN DETECTING EARLY-STAGE KIDNEY DISEASES.

# DATASET OVERVIEW

- **Dataset Name** : CT KIDNEY DATASET: Normal-Cyst-Tumor and Stone
- **Source** : Kaggle
- **About Dataset** : Dataset was collected from PACS (Picture archiving and communication system) from different hospitals in Dhaka, Bangladesh where patients were already diagnosed with having a kidney tumor, cyst, normal or stone findings.
- **Data Type**: CT scan slices
- **Classes**: 4 ( Normal, Cyst, Tumor, Stone )
- **Dataset contains**
  - cyst 3,709,
  - normal 5,077,
  - stone 1,377,
  - tumor 2,283
- **Total** : 12,446 unique data
- **Image Format**: Originally DICOM, converted to grayscale for analysis



# IMPLEMENTATION OF CLAHE



BEFORE

AFTER

## SNIPPET OF CLAHE

```
def preprocess_image_with_CLAHE(image_path):  
    """  
    Reads an image from the given path, converts it to grayscale,  
    applies CLAHE, and resizes it to the target size.  
    """  
    image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE) # Convert to grayscale  
    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))  
    enhanced_image = clahe.apply(image) # Apply CLAHE  
    resized_image = cv2.resize(enhanced_image, (128, 128)) # Resize to (128, 128)  
    return resized_image
```

### Step 1: Load the Image

Read the image file from the given path. Convert it to grayscale (black and white) for simplicity and focus on intensity values.

### Step 2: Apply CLAHE (Contrast Limited Adaptive Histogram Equalization)

Use CLAHE to enhance the contrast of the image.

### Step 3: Resize the Image

Resize the enhanced image to a smaller, standard size of 128x128 pixels. This resizing ensures uniformity and reduces computational requirements for machine learning.

### Step 4: Return the Processed Image

The final enhanced and resized image is returned, ready for use in the model.

## LOAD AND PREPARE DATASET

1. **Define Categories and Base Directory** : Specify the base\_dir (main folder) containing subfolders for each category (e.g., cyst, normal, tumor, stone).
2. **Preprocess Each Image** : Combine the folder path and image name to get the full image path. Use the preprocess\_image\_with\_CLAHE function to preprocess the image: Add the preprocessed image to the images list and its corresponding category label to the labels list.
3. **Handle Errors Gracefully** : If any image fails to load or preprocess, print an error message with the file name.
4. **Convert to NumPy Arrays**: Images: Convert the list of processed images to a NumPy array and reshape it to include a channel dimension (shape:  $[-1, 128, 128, 1]$  for grayscale). Labels: Convert the labels list to a NumPy array.
5. **Normalize the Images**: Normalize the pixel values of the images by dividing them by 255.0, scaling them to a range of  $[0, 1]$ .

## Splitting Data into Training and Testing Set

**Training Data** (X\_train, y\_train): 80% of the dataset, used to train the model.

**Testing Data** (X\_test, y\_test): 20% of the dataset, used to evaluate the model's performance.


```
# 6. Split Data into Training and Testing Sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

# Building the Model

The model is a Convolutional Neural Network (CNN) designed to classify images into 4 categories (e.g., cyst, normal, tumor, stone). Here's a breakdown of its layers:

- ➡ • **Input Layer:** Accepts input images of size 128x128x1 (128x128 pixels, 1 grayscale channel).
  - ➡ • **First Convolutional Layer:** Conv2D(32, (3, 3)): Applies 32 filters of size 3x3 to the image to detect low-level features like edges.
  - ➡ • **First Pooling Layer:** MaxPooling2D(2, 2): Reduces the spatial size (height and width) by selecting the maximum value from every 2x2 region.
  - ➡ • **Second Convolutional Layer:** Conv2D(64, (3, 3)): Adds 64 filters for learning more detailed features. Followed by a MaxPooling2D(2, 2) layer for further dimensionality reduction.
  - ➡ • **Third Convolutional Layer:** Conv2D(128, (3, 3)): Applies 128 filters to capture even finer details in the image. Followed by another MaxPooling2D(2, 2) layer.
- Flatten Layer:
- ➡ • **Flatten():** Converts the 2D feature maps from the previous layer into a 1D vector, making it suitable for the dense (fully connected) layers.
  - ➡ • **Dense (Fully Connected) Layer:** Dense(128, activation='relu'): Adds 128 neurons to learn complex patterns from the extracted features.

# Building the Model

- 
- **Output Layer:** Dense(4, activation='softmax'): Final layer with 4 neurons, one for each class (cyst, normal, tumor, stone).  
Softmax Activation: Converts the output into probabilities for multi-class classification.

## Summary of the Model

- The CNN progressively extracts features from the images (edges → patterns → high-level details).
- It outputs a probability for each of the 4 classes, and the highest probability determines the predicted class.
- This architecture balances simplicity and performance, making it effective for the classification task.

### SNIPPET OF MODEL

```
# 7. Build Model
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(128, 128, 1)),
    MaxPooling2D(2, 2),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(2, 2),
    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D(2, 2),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(4, activation='softmax') # 4 categories for classification
])
```



## COMPILE MODEL

```
# 8. Compile Model
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

**PURPOSE:** PREPARES THE MODEL FOR TRAINING BY SPECIFYING THE OPTIMIZER, LOSS FUNCTION, AND EVALUATION METRICS.

## TRAIN MODEL

```
# 9. Train Model
history = model.fit(X_train, y_train, epochs=5, validation_split=0.2, batch_size=32)
```

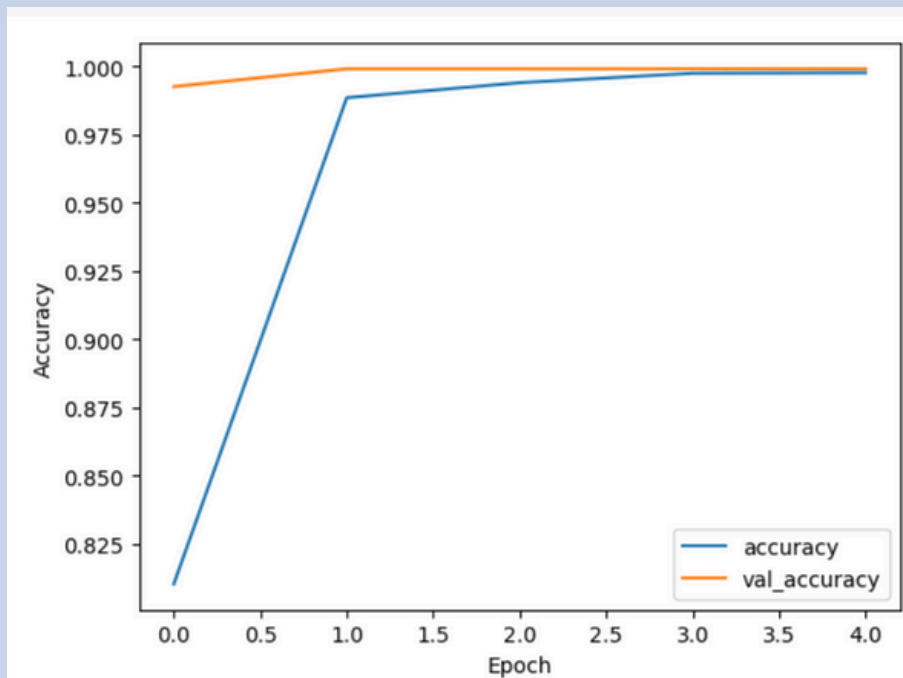
**PURPOSE:** TRAINS THE MODEL USING THE TRAINING DATA TO LEARN PATTERNS AND FEATURES FOR CLASSIFICATION.

## EVALUATE MODEL

```
# 10. Evaluate Model
test_loss, test_acc = model.evaluate(X_test, y_test)
print(f"Test Accuracy: {test_acc * 100:.2f}%")
```

```
154/154 ————— 175s 1s/step - accuracy: 0.9983 - loss: 0.0067 - val_accuracy: 0.9992 - val_loss: 0.0043
48/48 ————— 14s 297ms/step - accuracy: 0.9989 - loss: 0.0061
Test Accuracy: 99.87%
```

# Plotting accuracy and loss

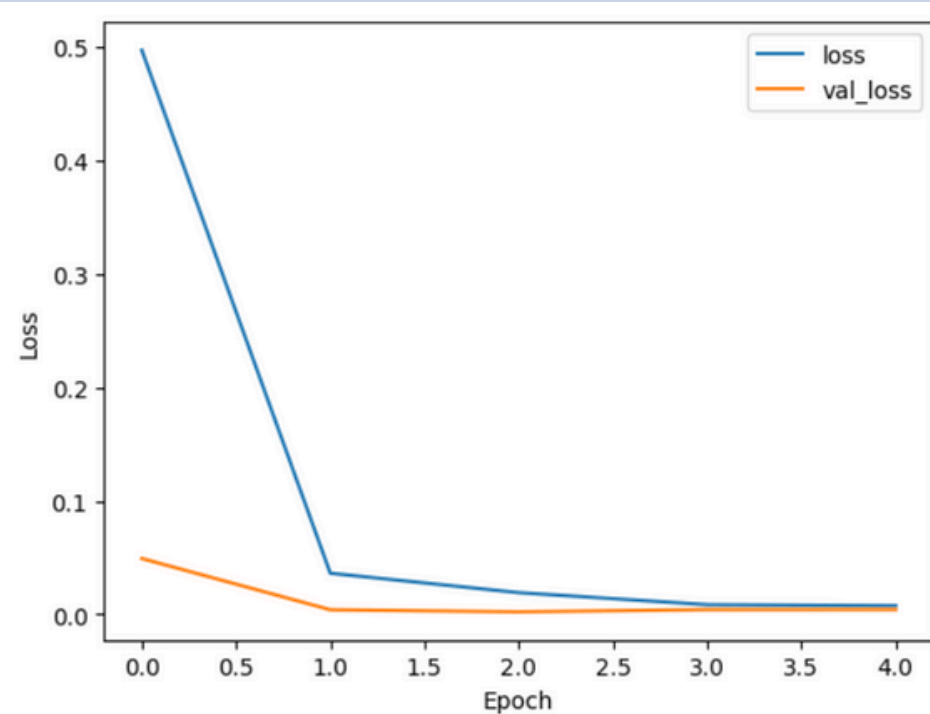


## Accuracy vs. Epoch

**Blue Line (Training Accuracy):** Shows how well the model performed on the training dataset for each epoch. The accuracy quickly increases and approaches 1.0, indicating the model learned the training data very well.

**Orange Line (Validation Accuracy):** Represents the model's accuracy on unseen (validation) data. It starts high and remains stable, suggesting the model generalizes well without overfitting.

**Conclusion:** The model achieves high accuracy on both training and validation data, indicating effective learning.



## Loss vs. Epoch


**Blue Line (Training Loss):** Indicates the model's error on the training dataset. It decreases rapidly, showing the model improves as it learns.

**Orange Line (Validation Loss):** Reflects the error on the validation dataset. It starts low and remains steady, which is a good sign of stability and no overfitting.


**Conclusion:** The model's loss decreases for training, and the validation loss remains low, confirming the model is well-trained and **balanced**.

# USING THE TRAINED MODEL TO PREDICT THE DISEASE.


```
new_image_path = '/content/drive/MyDrive/Dataset/CT-KIDNEY-DATASET/dataset/Final Test/Normal/Normal- (111).jpg'
print(f"Predicted Disease: {predict_disease(new_image_path, model)}")
```

1/1  0s 62ms/step  
Predicted Disease: Normal


```
new_image_path = '/content/drive/MyDrive/dataset/CT-KIDNEY-DATASET-Normal-Cyst-Tumor-Stone/CT-KIDNEY-DATASET-Normal-Cyst-Tumor-Stone/Cyst/Cyst- (10).jpg'
print(f"Predicted Disease: {predict_disease(new_image_path, model)}")
```

1/1  0s 210ms/step  
Predicted Disease: Cyst

```
new_image_path = '/content/drive/MyDrive/Dataset/CT-KIDNEY-DATASET/dataset/Final Test/Tumour/Tumor- (139).jpg'
print(f"Predicted Disease: {predict_disease(new_image_path, model)}")
```

1/1  0s 27ms/step  
Predicted Disease: Tumor

```
new_image_path = '/content/drive/MyDrive/dataset/CT-KIDNEY-DATASET-Normal-Cyst-Tumor-Stone/CT-KIDNEY-DATASET-Normal-Cyst-Tumor-Stone/Stone/Stone- (1).jpg'
print(f"Predicted Disease: {predict_disease(new_image_path, model)}")
```

1/1  0s 31ms/step  
Predicted Disease: Stone

# CONCLUSION

- **High Accuracy Achieved:** The model demonstrates excellent training and validation accuracy, indicating that it has successfully learned to classify kidney diseases (Cyst, Tumor, Stone, etc.) effectively.
- **Low Loss Values:** Both training and validation loss values are minimal, showcasing the model's stability and ability to generalize well to unseen data without overfitting.
- **Effective Preprocessing:** The use of CLAHE preprocessing and normalization significantly enhances image features, contributing to the model's performance.
- **Successful Multi-Class Classification:** The model can accurately differentiate between multiple classes (e.g., Cyst, Tumor, Stone), as evidenced by consistent predictions during testing.
- **Generalization Ability:** Validation metrics confirm the model's capability to handle new images, making it a reliable tool for real-world applications.
- **Practical Application:** The function to predict diseases on new images ensures ease of use and scalability, supporting potential deployment in medical diagnostics.



**Thank you !**

