

# **Hybrid Deep Learning and Machine Learning Approach for Multi-Class Kidney CT Image Classification**

*Report submitted to the SASTRA Deemed to be University in partial fulfillment of the  
requirements for the award of the degree of*

**Master Of Computer Application**

*submitted by*

**ARUN KUMAR G (125176009, MASTER OF COMPUTER APPLICATION)  
PRAVINRAJ S (125176063, MASTER OF COMPUTER APPLICATION)**

**MAY 2025**



**SCHOOL OF COMPUTING  
THANJAVUR,TAMIL NADU,INDIA-613 401**



# SASTRA

ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION

DEEMED TO BE UNIVERSITY

(U/S 3 of the UGC Act, 1956)



THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

T H A N J A V U R | K U M B A K O N A M | C H E N N A I

## SCHOOL OF COMPUTING

THANJAVUR - 613 401

### Bonafide Certificate

This is to certify that the project report titled “ **Hybrid Deep Learning and Machine Learning Approach for Multi-Class Kidney CT Image Classification** ” submitted in partial fulfillment of the requirements for the award of the degree of Master of Computer Application to the SASTRA Deemed to be University , is a bona-fide record of the work done by **Mr. Arun Kumar G** ( Reg. No. 125176009) , **Mr. PravinRaj S** ( Reg. No. 125176063) during the final semester of the academic year 2024-25 , in the **School of Computing**, under my supervision. This report has not formed the basis for the award of any degree, diploma, associateship, fellowship or other similar title to any candidate of any University.

**Signature of Project Supervisor** :

**Name with Affiliation** :

**Date** :

Project Viva voce held on \_\_\_\_\_

**Examiner 1**

**Examiner 2**



# SASTRA

ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION

DEEMED TO BE UNIVERSITY

(U/S 3 of the UGC Act, 1956)



THINK MERIT | THINK TRANSPARENCY | THINK SASTRA


T H A N J A V U R | K U M B A K O N A M | C H E N N A I

**SCHOOL OF COMPUTING**

**THANJAVUR - 613 401**

### Declaration

We declare that the project report titled **“Hybrid Deep Learning and Machine Learning Approach for Multi-Class Kidney CT Image Classification”** submitted by us is an original work done by us under the guidance of **Dr. Lavanya R, AP- III, School of Computing, SASTRA Deemed to be University** during the final semester of the academic year 2024-25, in the **School of Computing**. The work is original and wherever We have used materials from other sources, We have given due credit and cited them in the text of the report. This report has not formed the basis for the award of any degree, diploma, associate-ship, fellowship or other similar title to any candidate of any University.

**Signature of the candidate(s)** :  S. Pravin Raj

**Name of the candidate(s)** : Arun Kumar G, PravinRaj S

**Date** : 06-05-2025

## Acknowledgements

We would like to thank our Honorable Chancellor **Prof. R. Sethuraman** for providing us with an opportunity and the necessary infrastructure for carrying out this project as a part of our curriculum.

We would like to thank our Honorable Vice-Chancellor **Dr. S.Vaidhyasubramaniam** and **Dr. S. Swaminathan**, Dean, Planning & Development, for the encouragement and strategic support at every step of our college life.

We extend our sincere thanks to **Dr. R. Chandramouli**, Registrar, SASTRA Deemed to be University for providing the opportunity to pursue this project.

We extend our heartfelt thanks to **Dr. V. S. Shankar Sriram**, Dean, School of Computing, **Dr. R. Muthaiah**, Associate Dean, Research, **Dr. K.Ramkumar**, Associate Dean, Academics, **Dr. D. Manivannan**, Associate Dean, Infrastructure, **Dr. R. Alageswaran**, Associate Dean, Students Welfare.

Our guide, **Dr. Lavanya R, AP-III** School of Computing was the driving force behind this whole idea from the start. His deep insight in the field and invaluable suggestions helped me/us in making progress throughout our project work. I/We also thank the project review panel members for their valuable comments and insights which made this project better. I/We extend our sincere thanks to **Dr. Pradeepa S, AP - III**, School of Computing, for her valuable guidance and support in helping us complete this project.

# Table of Contents

<b>Title</b>	<b>Page No.</b>
Bona-fide Certificate	ii
Declaration	iii
Acknowledgements	iv
List of Figures	vi
List of Tables	viii
Abbreviation	ix
Notations	x
Abstract	xi
1. Introduction	1
2. Objectives	3
3. Experimental Work / Methodology	5
3.1. Dataset description	5
3.2 Preprocessing and Augmentation	8
3.3 InceptionResNetV2 Model	12
3.4. Feature extraction	13
3.5. Meta-Learning Classifier	14
3.6. Grad-CAM and t-SNE visualization	18
4. Results and Discussion	20
4.1 InceptionResNetV2 Model Result	20
4.2 Meta Learning Results	23
5. Conclusions and Further Work	28
6. References	29
7. Appendix	32
7.1. Similarity Check Report	32
7.2. Sample Source code	32

## List of Figures

Figure No.	Title	Page No.
3.1.1	Work Flow of proposed Model	6
3.1.2	CT images of four classes	7
3.1.3	Model Architecture Diagram	7
3.1.4	Dataset Distribution	8
3.2.1	Dataset split visualization	9
3.2.2	InceptionResnetv2 Overall Network Structure	10
3.5	Meta Learner	13
4.1.1	Training & validation Accuracy and loss graph	21
4.1.2	Confusion Matrix	21
4.1.3	ROC and Precision recall curve	22
4.1.4	Classification Report & Last Conv layer t-SNE	22
4.1.5	Grad-Cam Visualization For All Classes	23
4.2.1	Confusion Matrix for Meta Learning	24
4.2.2	Classification Report for Meta Learning	24
4.2.3	t-SNE Visualization for Meta Learning	25
4.2.4	SVM Classification Result	25
4.2.5	Random Forest Classification Result	25
4.2.6	KNN Classification Result	26
4.2.7	XGBoost Classification Result	26
7.2.1	Importing Libraries	32
7.2.2	Read Data and Storing in Data frame	33
7.2.3	Creating Image Generator	33

7.2.4	Generic Model Creation	34
7.2.5	Model Training	34
7.2.6	Loading Model	34
7.2.7	t-SNE Visualization	35
7.2.8	Feature Extraction	36
7.2.9	Stacking Classifier	36
7.2.10	Evaluation	37

## List of Tables

Table No.	Table Name	Page No.
4.1	Comparison with the proposed model	27



## Abbreviations

UV	Ultraviolet
WHO	World Health Organization
CAD	Computer-aided diagnosis
AI	Artificial Intelligence
DL	Deep Learning
CNN	Convolutional Neural Networks
t-SNE	t-Distributed Stochastic Neighbor Embedding
GRAD-CAM	Gradient-weighted Class Activation Mapping
KL	Kullback-Leibler
SVM	Support Vector Machine
KNN	K-Nearest Neighbors
RBF	Radial Basis Function

## Notations

### English Symbols

B	number of base models
c	number of output channels
d	number of output channels
h	height of the input feature map
w	width of the input feature map
k	size of the convolutional kernel
$A^k$	activations from the final convolutional layer
$Bw_b$	weight assigned to each base model's output by the meta-model
$h_m(x)$	weak learner added at stage m
$L(y_i, F(x_i))$	loss function
$x_i, x_j$	two points in p-dimensional space
$y^b$	prediction from the base models
$y_j$	target values of previous rows
Z	normalizes spatial dimensions

## Greek Symbols

$\Sigma$	Summation
$*$	Multiplication
$\sigma^{(k)}$	Standard deviation
$\mu^{(k)}$	channel-wise mean
$\lambda$	hyperparameter controlling the strength of regularization

## **Abstract**

Kidney-related diseases such as stones, cysts, and tumors remain a pressing concern globally, and there is an increasing demand for robust AI systems to assist in early diagnosis. In this study, we propose a novel hybrid AI framework using InceptionResNetV2 for effective kidney disease classification from CT images. The model was trained on a balanced dataset comprising four classes—Normal, Cyst, Tumor, and Stone—and achieved a classification accuracy of 96.44% as a standalone deep learning model. To enhance interpretability and feature representation, t-SNE was employed for dimensionality reduction and Grad-CAM was used for visualizing discriminative regions. Subsequently, features extracted from the penultimate layer of InceptionResNetV2 were fed into a meta-learning ensemble classifier consisting of SVM, XGBoost, Random Forest, and KNN, with Logistic Regression as the final classifier in a stacking ensemble. The proposed ensemble achieved a remarkable accuracy of 99.32%, outperforming several state-of-the-art approaches reported in the literature. The classification report demonstrated balanced precision, recall, and F1-scores across all classes, validating the robustness of the proposed framework. This work highlights the potential of combining deep learning with classical machine learning in a stacking architecture to significantly boost diagnostic performance for renal diseases..

## **Specific Contribution**

- Arun Kumar G - Model training and t-SNE visualization, Grad-cam, Meta Learning and Final evaluation
- PravinRaj S - data preprocessing, Feature extraction, Machine Learning Algorithms

## **Specific Learning**

- Arun Kumar G- Acquired practical knowledge model training, forming the foundation of the deep learning workflow Gained knowledge in applying t-SNE for visualization, performing Grad-cam analysis, and Meta Learning
- PravinRaj S - Acquired practical knowledge in data preprocessing, deep feature extraction, and the implementation of machine learning classifiers

# CHAPTER 1

## INTRODUCTION

Kidney disease is a significant public health concern, continuing to spread globally despite medical advances. Chronic kidney disease affects over 10% of the world population [2] and ranked 16th among the leading causes of death in 2016, with projections suggesting it may rise to the 5th position by 2040 [3]. Among the various renal disorders, the most frequently occurring are cysts, kidney stones (nephrolithiasis), and renal cell carcinoma (tumors), all of which compromise kidney function and, if left undiagnosed, can lead to complete renal failure [1], [10].

A kidney cyst is a fluid-filled sac that typically presents with water-like density, measured between 0 and 20 Hounsfield units on CT imaging [4][6]. Kidney stones are solid concretions of crystal aggregations affecting approximately 12% of the global population [7]. Renal cell carcinoma, commonly known as kidney cancer, is among the top ten most diagnosed cancers worldwide [8]. For diagnosing these conditions, clinicians often rely on imaging techniques such as X-ray, B-mode ultrasound (US), magnetic resonance imaging (MRI), and computed tomography (CT). Among these, CT imaging offers superior diagnostic clarity through three-dimensional, cross-sectional scans that allow detailed visualization of internal anatomy [9].

However, a major challenge in the early detection of these renal abnormalities is the global shortage of nephrologists and radiologists. For example, South Asia has only about one nephrologist per million people, compared to 25.3 in Europe [12]. Given the scale of the problem and the rapid advancements in artificial intelligence (AI) and deep learning in medical imaging, automated AI-based systems are increasingly being explored as reliable alternatives to aid clinical diagnosis [11].

In this study, we present a novel deep learning and ensemble learning-based pipeline for the automatic classification of kidney conditions—normal, cyst, tumor, and stone—from CT images. Unlike earlier studies that relied on single traditional machine learning algorithms or transformer-based architectures [1], our approach leverages the InceptionResNetV2 deep neural

network to extract high-level features from CT scans. These extracted features are further analyzed using t-distributed stochastic neighbor embedding (t-SNE) for dimensionality reduction and Grad-CAM for model interpretability.

We then constructed a meta-learning ensemble model using the extracted features. The base learners included Support Vector Machine (SVM), XGBoost, Random Forest, and K-Nearest Neighbors (KNN), followed by a Logistic Regression classifier that served as the final decision layer in a stacking ensemble. The ensemble model achieved an outstanding accuracy of 99.32%, significantly outperforming the standalone deep learning model's accuracy of 96.44%, and demonstrated strong generalization across all four classes.

The key contributions of this study are:

- A hybrid deep learning pipeline was developed using InceptionResNetV2 for CT-based kidney disease classification, achieving a base model accuracy of 96.44%.
- Grad-CAM was used to visualize the regions of interest in the CT scans, enhancing interpretability and trustworthiness of the model's predictions.
- A stacking ensemble model was built using classical machine learning classifiers with logistic regression as the meta-learner, which improved classification accuracy to 99.32%.
- Extensive experiments were conducted with precision, recall, and F1-score analyses, demonstrating the robustness of the proposed system in classifying kidney cysts, tumors, stones, and normal images.

The rest of the paper is organized as follows: Section 2 reviews related work on kidney image classification using AI and DL. Section 3 outlines the proposed methodology, including data preprocessing, deep feature extraction, and SVM classification. Section 4 presents the experimental setup and results. Finally, Section 5 concludes the paper and highlights future research directions.

## **CHAPTER 2**

### **OBJECTIVES**

The primary objective of this research is to develop a highly accurate, explainable, and scalable system for the multi-class classification of kidney conditions—including normal, cyst, tumor, and stone—using abdominal and urogram CT images. This is achieved by first utilizing the InceptionResNetV2 model to extract high-level semantic features from the penultimate layer of CT images. To enhance model interpretability and understand the internal feature representation, t-distributed Stochastic Neighbor Embedding (t-SNE) and Gradient-weighted Class Activation Mapping (Grad-CAM) are applied. These techniques provide valuable insights into feature space separability and anatomical focus, respectively.

Building on this foundation, the extracted features are fed into a meta-learning framework that combines the strengths of multiple classical machine learning classifiers: Support Vector Machine (SVM), Random Forest, XGBoost, and K-Nearest Neighbors (KNN). The final classification is performed using Logistic Regression as a meta-learner in a stacking ensemble architecture. This ensemble approach leverages diverse decision boundaries and learning paradigms to increase classification accuracy and generalizability across different kidney conditions.

With an achieved accuracy of 99.32%, the proposed stacking ensemble outperforms previous hybrid models and demonstrates its effectiveness for real-time diagnostic support in clinical environments. Additionally, the inclusion of explainability techniques aligns the model with the growing demand for trustworthy and transparent AI solutions in healthcare.

By combining these two parts—selective deep feature extraction through t-SNE and adaptive classification through meta-learning—the hybrid model developed here markedly lowers computational overhead without compromising on, and sometimes even improving, classification accuracy. Unlike traditional end-to-end frameworks involving huge volumes of data and training time, this method makes the diagnostic process leaner and more efficient for real-world applications in clinical settings where speed and accuracy are key.

In addition, this model provides the foundation for future research on explainable AI in medical diagnosis. By being centered on interpretable and explainable layers and features, it provides a foundation for integrating explainability methods that can deliver visual or quantitative feedback into the decision-making process. It is crucial for establishing trust with healthcare professionals and facilitating the integration of AI-aided diagnosis tools into clinical practice.

In summary, the suggested hybrid kidney disease classification model provides a cost-effective, accurate, and scalable solution that overcomes major drawbacks of common deep learning approaches. Using this approach opens up more accessible and reliable AI-assisted applications in dermatological diagnosis.



## CHAPTER 3

### EXPERIMENTAL WORK/ METHODOLOGY

Figure 3.1 shows the overall architecture diagram for the proposed model for the Kidney disease analysis from CT images. The data sets used for the experiment are described in this section, which also provides a detailed background examination of the deep learning architectures and the various machine learning algorithms used in this work, including transfer learning.

#### 3.1 Dataset description

The data set was gathered from a hospital in Dhaka, Bangladesh, called PACS (Picture archiving and communication system) and workstations, where the patients were already diagnosed with having a kidney tumor, cyst, normal or stone findings. All the subjects of the data set volunteered to participate in the research experiments, and informed consents were taken from them before gathering the data. The experiments and data gathering were already approved by the concerned hospital authorities of Dhaka Central International Medical College and Hospital (DCIMCH).[22]

Both the Coronal and Axial cuts were chosen from both contrast and non-contrast studies with protocol for the entire abdomen and urogram. The Dicom study was then chosen carefully, one diagnosis at a time, and from those they developed a batch of Dicom images of the region of interest for each radiological finding. Then, they removed each patient's information and meta data from the Dicom images and converted the Dicom images to a lossless joint photographic expert group (jpeg/jpg) image format. The Philips IntelliSpace Portal 9.034 application is utilized for data annotation, which is an advanced image visualization tool for radiology images, and the Sante Dicom editor tool<sup>35</sup> is utilized for data conversion to jpg images, which is mainly utilized as a Dicom viewer with advanced features to help radiologists diagnose specific disease findings. After manual conversion and annotation of the data, each image finding was once again checked by a doctor and a medical technologist to recheck the correctness of the data.

The dataset contains 12,446 unique data within it in which the cyst contains 3,709, normal 5,077, stone 1,377, and tumor 2,283. The dataset is available on Kaggle

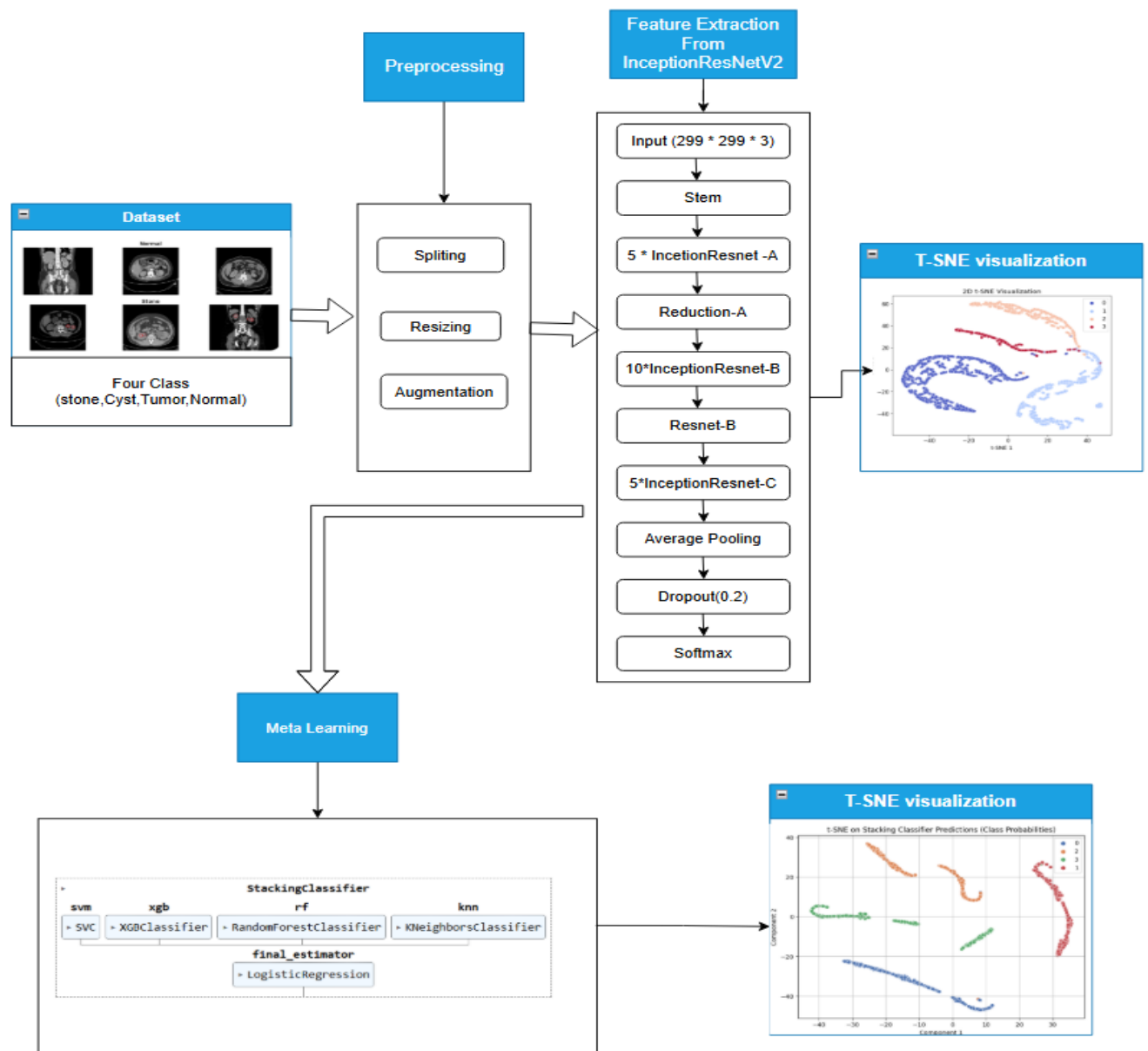


Fig.3.1.1 Work flow of the Proposed Model

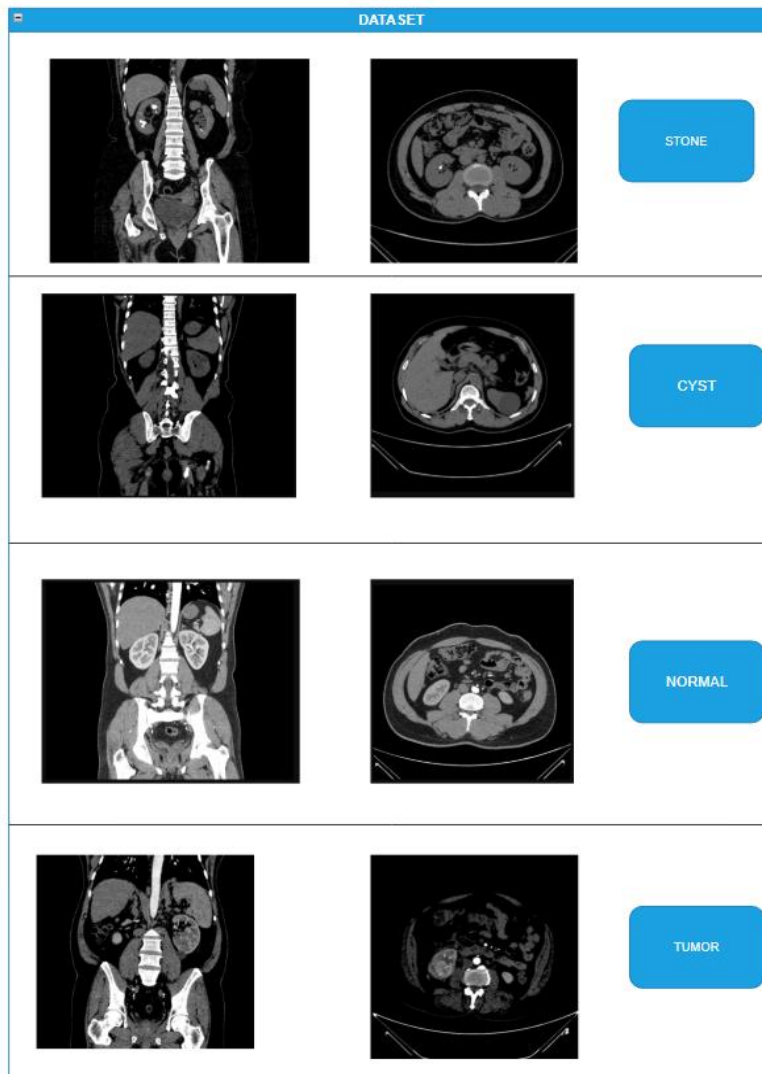


Fig.3.1.2 CT images of Four Classes

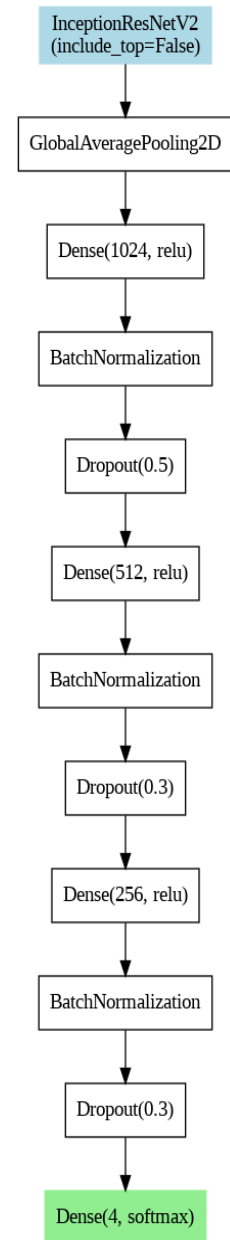


Fig 3.1.3 Model Architecture Diagram

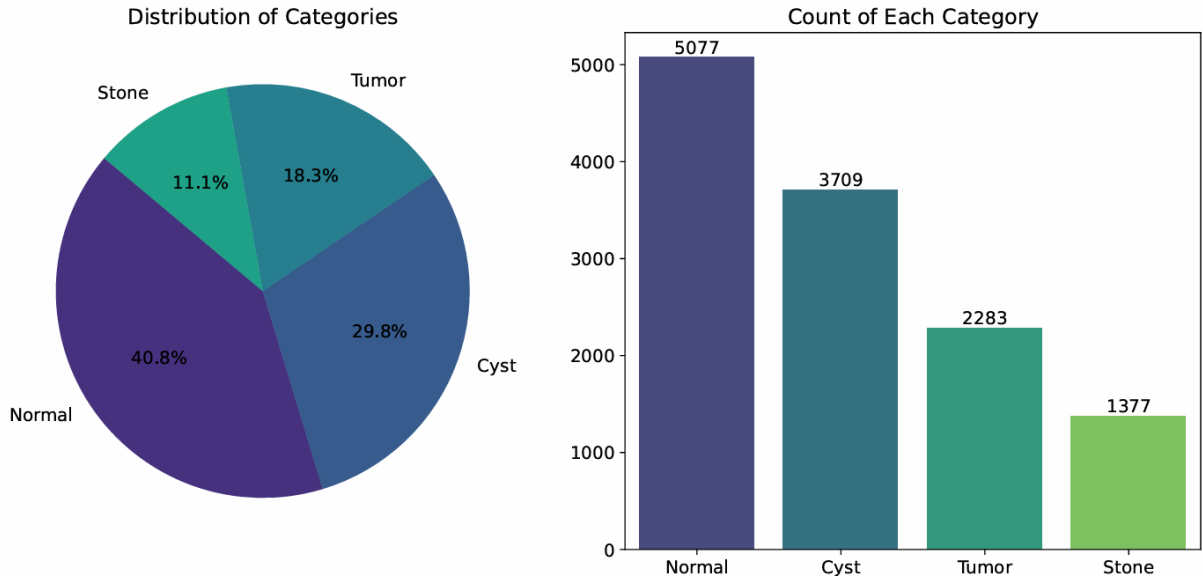


Fig 3.1.4 Dataset Distribution

### 3.2 Preprocessing and Augmentation

The original CT kidney dataset consisted of four categories: Normal (5077 images), Cyst (3709), Tumor (2283), and Stone (1377), exhibiting a clear class imbalance. To address this, a balanced dataset was created by randomly selecting 1300 images from each class, ensuring equal representation across categories. This balanced dataset was then split into training, validation, and testing subsets using a stratified approach: 80% for training and 20% for testing, with 20% of the training data further set aside for validation. This stratification preserved class distributions and supported stable model evaluation.

All CT images were resized to 224×224 pixels to match the input size of the InceptionResNetV2 model. Images were normalized by scaling pixel intensity values to the range [0, 1]. To enhance generalization and reduce overfitting, a comprehensive data augmentation strategy was applied to the training set. This included random rotations ( $\pm 15^\circ$ ), width and height shifts (10%), shear and zoom transformations (10%), and horizontal flipping. These augmentations simulated real-world variability in medical imaging and helped the model learn more robust and invariant features.

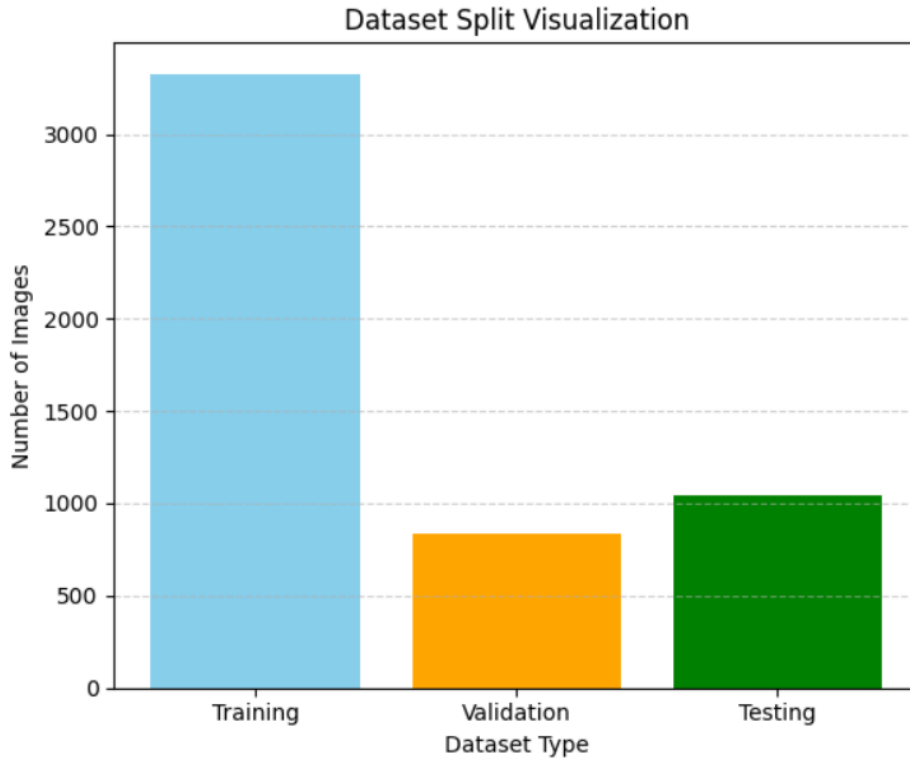


Fig.3.2.1 Dataset Split Visualization

### 3.3 InceptionResNetV2 Model

Inception-ResNet-V2 model is a change from the Inception V3 model, which was inspired by the ResNet paper on Microsoft's residual network. It deepens the network depth while avoiding the problem of gradient disappearance and gradient explosion and improves network performance. The computing power is improved, the network depth is further increased and the nonlinear of the network is increased by decomposing the convolution kernel. This paper is constructed based on the Inception-ResNet-V2 model.

The feature extraction module in the Inception-ResNet-V2 model is composed of three similar modules: Inception-ResNet-A, Inception-ResNet-B and Inception-ResNet-C. First explaining the rectangle in the figure, from left to right in turn from top to bottom for convolution kernel size, convolution operating or pooling operating, channel number, step length, generally to 1 mark, V represents valid, not filling, the default is SAME filling, readers can be seen from the input and output size change also, filter concat expressed as tensor connection, the number of channels after connection is the sum of the former. Among them, Inception-ResNet-B and Inception-ResNet-C

have asymmetric convolution kernels of  $1 \times 7$ ,  $7 \times 1$ ,  $1 \times 3$  and  $3 \times 1$ , compared with the symmetric convolution kernels of Inception-ResNet-A module, such operation can reduce the calculation time during parameter operation and deepen the depth and nonlinear of the network. The feature graph is activated initially by batch normalization processing and Relu activation function after a series of tensor connection and convolution calculation, and finally the result of each module is obtained. After passing through all modules, the result is classified by full connection layer and softmax function. [14]

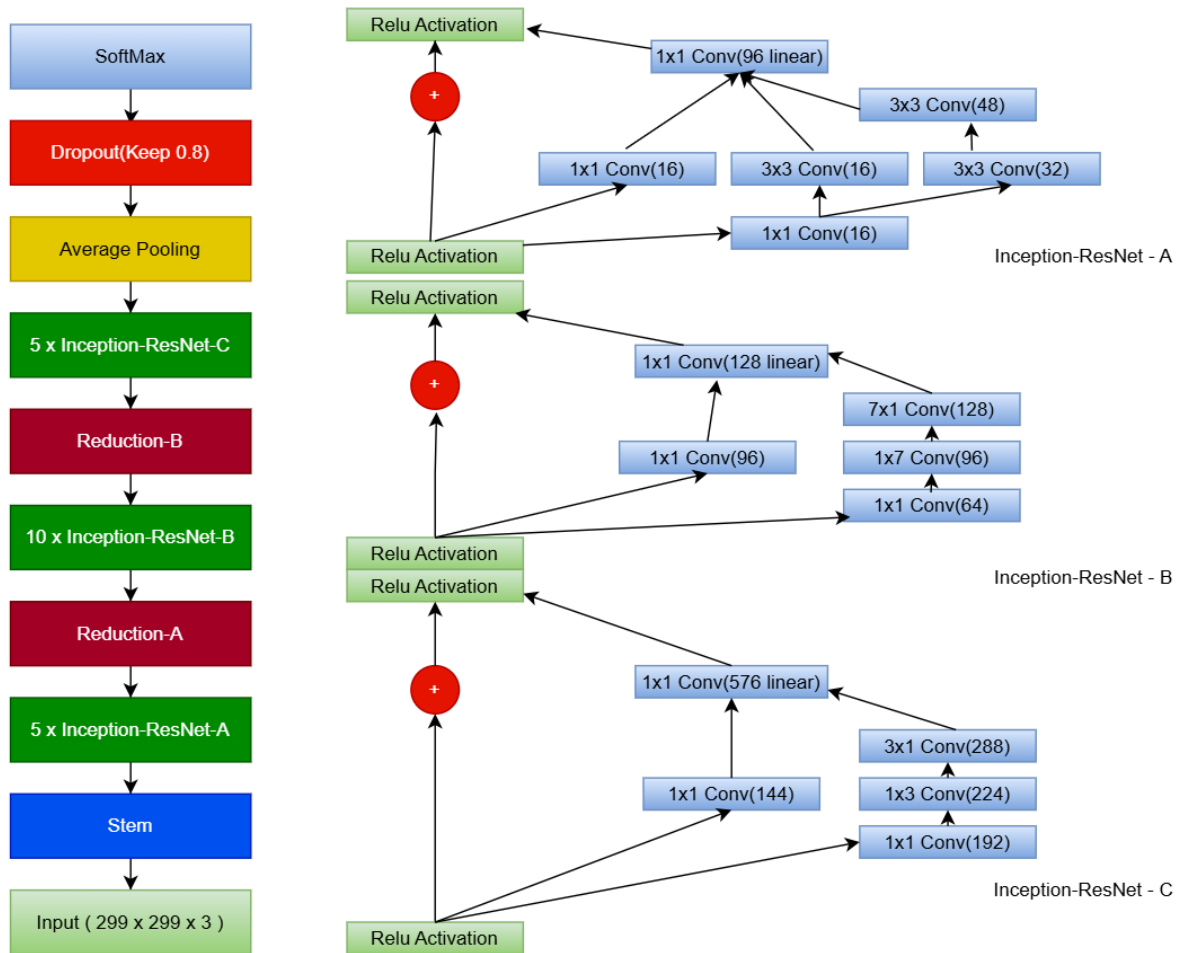


Fig.3.2.2 InceptionResNetv2 Overall Network Structure

### 3.3.1 Stem Block

The **stem block** serves as the initial stage of the InceptionResNetV2 architecture and is responsible

for processing the raw input image into a lower-dimensional but rich feature representation. It typically consists of a series of convolutional layers with increasing filter sizes, combined with max pooling and/or strided convolutions to reduce spatial dimensions while increasing the depth. This early transformation allows the network to extract fundamental edges and textures effectively. Mathematically, each convolution operation applies a filter  $W$  over a region of the input  $X$ , producing an output feature map  $Y$ , using the formula:

$$Y_{i,j,k} = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \sum_{c=0}^{C-1} X_{i+m,j+n,c} \cdot W_{m,n,c,k} + b_k \quad (1)$$

Here,  $M \times N$  is the filter size,  $C$  is the number of input channels, and  $b_k$  is the bias added to each output channel.[15]

### 3.3.2. Inception-ResNet Blocks (A, B, C):

The backbone of the InceptionResNetV2 model comprises three key block types: Inception-ResNet-A, B, and C, each working at different feature map resolutions ( $35 \times 35$ ,  $17 \times 17$ , and  $8 \times 8$  respectively). These blocks blend the multi-path design of the Inception modules with residual connections from ResNet, helping to train very deep networks by addressing the vanishing gradient problem. Each block applies multiple convolutions in parallel (e.g.,  $1 \times 1$ ,  $3 \times 3$ ,  $5 \times 5$ , or asymmetric filters), concatenates their outputs, and projects the result through a  $1 \times 1$  convolution before adding it back to the input using a residual connection. The residual output is given by:

$$Y = X + \alpha F(X) \quad (2)$$

Here,  $F(X)$  represents the transformed input and  $\alpha$  is a scaling factor (typically 0.1) to stabilize gradients during training [15][16]

### 3.3.3. Reduction Blocks (Reduction-A and B):

Reduction blocks are introduced to reduce the spatial size of the feature maps while increasing their depth, thus enabling the model to process higher-level features efficiently with reduced computational cost. These blocks apply max-pooling, strided convolution, and multi-branch down sampling to transition between Inception stages. The output dimensions of these down sampling operations are governed by:

$$\text{Output Size} = \left\lfloor \frac{\text{Input Size} - \text{Kernel Size} + 2 \cdot \text{Padding}}{\text{Stride}} \right\rfloor + 1 \quad (3)$$

Reduction-A typically follows the stem and precedes Inception-ResNet-A, while Reduction-B transitions from block B to C

### 3.4 Feature Extraction

Feature extraction plays a crucial role in reducing the dimensionality of raw image data while preserving the most relevant discriminative information for classification. In this study, a pretrained InceptionResNetV2 model was employed to extract deep features from kidney CT images. Specifically, the activations from the second-to-last dense layer (prior to the softmax output) were used as the feature vector. This layer encodes semantically rich information learned by the deep neural network during training.

Let an input image be denoted as  $x \in \mathbb{R}^{224 \times 224 \times 3}$ . The InceptionResNetV2 model is a composition of convolutional, pooling, and activation layers followed by fully connected layers. The output of the feature extractor can be formulated as a function composition:

$$f = \phi(x) = f_L \circ f_{L-1} \circ \dots \circ f_1(x) \quad (4)$$

Where  $f_i$  represents the transformation (e.g., convolution, activation, normalization, pooling) at the  $i$ -th layer,  $L$  is the index of the second-to-last layer,  $\phi(x) \in \mathbb{R}^d$  is the resulting feature vector for image  $x$ , with  $d$  being the dimension of the flattened output from the penultimate layer.[23]



he dense layer typically applies a linear transformation followed by a nonlinear activation:

$$f^{(dense)}(z) = \sigma(Wz + b) \quad (5)$$

Where  $z \in \mathbb{R}^n$  is the input feature map (after global average pooling),  $W \in \mathbb{R}^{d \times n}$  is the weight matrix,  $b \in \mathbb{R}^d$  is the bias vector,  $\sigma$  is the activation function (e.g., ReLU).

After extracting the deep features  $\phi(x)$ , they were flattened to a 1D vector:

$$\mathbf{v}_x = \text{Flatten}(\phi(x)) \in \mathbb{R}^d \quad (6)$$

These feature vectors  $\mathbf{v}_x$  serve as the input to traditional machine learning classifiers such as SVM, Random Forest, XGBoost, and KNN. This hybrid approach benefits from the expressive power of deep networks for feature representation and the simplicity and robustness of classical models for final classification.

By decoupling feature extraction from classification, the model gains interpretability and efficiency. Moreover, it allows the reuse of a pretrained deep model across different classification algorithms without retraining the entire network.

### 3.5 Meta-learning classifier

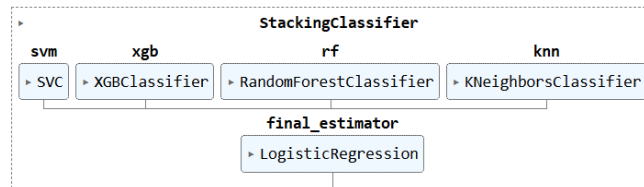


Fig. 3.5 Meta-Learner

Meta-Learning is a method in which predictions of different base models are combined with a meta-classifier[34] as shown in Fig. 3.4.1. The meta-classifier is trained to weigh the base model outputs optimally and combine them for better predictive accuracy.

$$y^{meta} = f_{meta}(\{y^b\}_{b=1}^B) \quad (7)$$

The equation (13) how the meta classifier computes the prediction from all the base models. Here  $B$  is number of base models,  $y^b$  is prediction from the base models,  $f_{meta}$  is final estimator which logistic regression in this case.

In a meta-learning classifier, prediction is performed through a two-tier process involving multiple base models and a meta-model. The base layer consists of multiple individual models  $f_b$ , each trained independently. For every input  $X$ , each base model generates its own prediction  $y^b$  as shown in equation (14). Here,  $b$  indexes each base learner and  $f_b(X)$  denotes the prediction made by the  $b$ -th model on input  $X$ .

$$y^b = f_b(X) \quad \forall b \in \{1, \dots, B\} \quad (8)$$

The predictions from all base learners are then passed to a meta-layer. It learns to combine them optimally. This is done using logistic regression as the meta-model, with L1 regularization. The equation (15) shows the final prediction being class 1, given all base predictions. Here  $y^b$  is prediction from the  $b$ -th base model,  $w_b$  is the weight assigned to each base model's output by the meta-model,  $w_0$  is the bias term in the logistic regression,  $\sigma()$  is the sigmoid activation function,

used to produce the final probability. To prevent overfitting and increase sparsity, L1 regularization constraint is placed on the weight vector  $w$ , such that  $\|w\|_1 \leq \lambda$ . Here  $\|w\|_1$  represents the L1 norm of the weights and  $\lambda$  is a hyperparameter controlling the strength of regularization.[24]

$$P(y = 1 | \hat{y}) = \sigma \left( b = 1 \sum B w_b y^b + w_0 \right) \quad (9)$$

In the meta-classifier, four base models are employed namely XGBoost, SVM, Random Forest, KNN. XGBoost is gradient boosting algorithm that minimizes the loss function with second-order Taylor expansion[36]. It is specifically built for high-speed classification and regression tasks, tuning both speed and accuracy. The equation (18) is the objective function at the  $t$ -th iteration. Here,  $n$  is the number of training samples,  $L(y_i, F^{(t)}(x_i))$  is the loss between the true label and prediction at iteration  $t$ ,  $T$  denotes the number of leaves in the newly added tree,  $\gamma$  and  $\lambda$  are regularization hyperparameters that penalize complexity. The equation (19) is the regularization component encourages simpler trees. Here,  $\gamma T$  discourages the number of leaves in the tree, discouraging overly complex models and  $\frac{\lambda}{2} \|w\|^2$  is a regularization term applied to the leaf weights  $w$ , controlling their magnitude and enhancing generalization.[25]

$$L^{(t)} = \sum_{i=1}^n L(y_i, F^{(t)}(x_i)) + \gamma T + \frac{\lambda}{2} \|w\|^2 \quad (10)$$

$$\Omega(f_t) = \gamma T + \frac{\lambda}{2} \|w\|^2 \quad (11)$$

SVC identifies a hyperplane that maximizes the margin between two classes. SVC supports linear and non-linear classification with kernel functions. It performs well in high-dimensional spaces and is resistant to overfitting, particularly in situations with a clear margin of separation between classes. The equation (20) is the dual optimization objective performed by SVC. Here  $\alpha_i$  are the Lagrange multipliers to be improved,  $K(x_i, x_j)$  is the kernel function which maps the data into a higher-dimensional space to make it linearly separable,  $K(x, z) = e^{-\gamma \|x - z\|^2}$  is the Radial Basis Function (RBF) kernel, where  $\gamma$  controls the width of the Gaussian kernel and finds the influence of each training example. This kernelized formulation allows SVC to handle complex, non-linear decision boundaries efficiently while maintaining strong generalization performance.[26]

$$\max_{\alpha} \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \quad (12)$$

A random forest classifier is a machine learning model that employs many decision trees to classify data, choosing its predictions on the majority vote of the individual trees. It's known for its accuracy and stability. It does this by selecting a random subset of features and training the decision trees on a different subset of the data. The final prediction is made through majority voting[37]. The equation (21) is the prediction rule. Here  $\hat{y}_{RF}$  is the final prediction from the Random Forest,  $\hat{y}_{tree_k}$  is the prediction from the k-th decision tree, mode denotes majority voting over all trees  $k=1 \dots K$ . The equation (22) is the Sampling Correlation Formula which is used to find how much the trees in the forest correlate in their predictions. Here,  $T(x; Z)$  is the output of a decision tree trained on data subset  $Z$  with random feature sampling  $\theta$ ,  $Var_z$  and  $E_z$  denote the variance and expectation over the dataset subsets.[27]

$$\hat{y}_{RF} = \text{mode} \left( \{\hat{y}_{tree_k}\}_{k=1}^K \right) \quad (13)$$

$$\rho(x) = \frac{Var_z[E_\theta[T(x; Z)]]}{Var_z[E_\theta[T(x; Z)]] + E_z[Var_\theta[T(x; Z)]]} \quad (14)$$

The K-Nearest Neighbors (KNN) Classifier is a machine learning algorithm that is non-parametric. It classifies an instance based on the majority class of its k closest neighbors in feature space. The algorithm depends on distance metrics to compute similarity between points[39]. The equation (24) is the prediction rule of KNN. Here,  $N_k(x)$  denotes the set of k nearest neighbors of instance x,  $y_j$  represents the label of the neighbor  $x_j$ . The equation (25) is the formula to calculate distance between the points. Here,  $x_i, x_j$  are two points in p-dimensional space.[28]

$$\hat{y}_{KNN} = mode(\{y_j | x_j \in N_k(x)\}) \quad (15)$$

$$d(x_i, x_j) = \sqrt{\sum_{k=1}^p (x_{ik} - x_{jk})^2} \quad (16)$$

In the proposed meta-learning architecture, a stacked ensemble model was employed to enhance the classification performance using a combination of diverse base learners. These base learners included Support Vector Machine (SVM), Random Forest (RF), XGBoost (XGB), and k-Nearest Neighbors (KNN). Each of these base models was trained using deep features extracted from the InceptionResNetV2 model. Their predicted probabilities served as inputs to a final meta-learner, which was implemented as a standard Logistic Regression classifier.

Logistic Regression estimates the probability of class membership by applying the sigmoid function to a linear combination of the input features. For a given instance  $z=[z_1, z_2, \dots, z_n]$  where each  $z_i$  is the predicted probability from a base learner, the final output is computed as:

$$\hat{y} = \sigma(w_0 + w_1 z_1 + w_2 z_2 + \cdots + w_n z_n)$$

$$\hat{y} = \frac{1}{1 + e^{-(w_0 + \sum_{i=1}^n w_i z_i)}} \quad (17)$$

Where  $\hat{y}$  is the predicted probability of the positive class,  $\sigma(\cdot)$  is the sigmoid function,  $w_0$  is the bias term,  $w_i$  are the learned weights for each base learner output  $z_i$ ,  $n$  is the number of base learners.[29]

The logistic regression meta-learner learns to optimally combine the outputs of the base models to produce a final classification decision. Unlike Lasso-regularized logistic regression, no L1 penalty is applied here; hence, all coefficients are learned without enforced sparsity. This ensemble approach leverages the strengths of multiple classifiers, with the meta-learner effectively learning how to weight each base model's output for improved overall prediction accuracy.

### 3.6 Grad-CAM and t-SNE Visualization :

To better understand the learned feature representations and decision behavior of the InceptionResNetV2 model, two complementary visualization techniques were employed: t-Distributed Stochastic Neighbor Embedding (t-SNE) and Gradient-weighted Class Activation Mapping (Grad-CAM).

t-SNE is a nonlinear dimensionality reduction technique used to visualize high-dimensional feature embeddings in two dimensions. After feature extraction from the penultimate dense layer of the model, t-SNE was applied to project the feature vectors into a 2D plane, helping reveal the intrinsic clustering of samples from different classes. Let  $\mathbf{x}_1, \dots, \mathbf{x}_N \in \mathbb{R}^D$  denote the high-dimensional features, and  $\mathbf{y}_1, \dots, \mathbf{y}_N \in \mathbb{R}^2$  their 2D projections. t-SNE minimizes the Kullback–Leibler (KL) divergence between the pairwise similarities in the high- and low-dimensional space:

$$\text{KL}(P\|Q) = \sum_{i \neq j} p_{ij} \log \left( \frac{p_{ij}}{q_{ij}} \right) \quad (18)$$

Where  $p_{ij}$  and  $q_{ij}$  represent pairwise similarities in high and low dimensions respectively.  $p_{ij}$  is computed using a Gaussian distribution, while  $q_{ij}$  uses a Student-t distribution to handle crowding.[31]

This visualization enabled class separability analysis, demonstrating that features from Normal, Cyst, Tumor, and Stone images formed distinct clusters, indicating good discriminative capability.

Grad-CAM was utilized to generate class-discriminative heatmaps from the final convolutional layer. It highlights the spatial regions in the CT scan that most strongly influenced the model's decision. Given a target class score  $y^c$ , the gradient of the score with respect to the feature maps  $A^k$  of the last convolutional layer is computed:

$$\alpha_k^c = \frac{1}{Z} \sum_i \sum_j \frac{\partial y^c}{\partial A_{ij}^k}$$

$$L_{\text{Grad-CAM}}^c = \text{ReLU} \left( \sum_k \alpha_k^c A^k \right) \quad (19)$$

Where  $\alpha_k^c$  is the importance weight for feature map  $A^k$ ,  $Z$  is the number of pixels in the feature map,  $L_{\text{Grad-CAM}}^c$  is the final heatmap, ReLU is used to retain only the features that positively influence the prediction.[32]

These visualizations serve as a critical interpretability tool, especially in medical imaging tasks, by ensuring that the model's attention is focused on clinically relevant regions (e.g., lesions, stones, or cysts) rather than on irrelevant artifacts.

## CHAPTER 4

### RESULTS AND DISCUSSION

#### 4.1 InceptionResNetv2 model results

The performance of the proposed kidney CT image classification model based on the InceptionResNetV2 architecture was evaluated on a test set comprising 1040 images, with 260 images for each of the four classes: Normal, Cyst, Tumor, and Stone. The classification report and confusion matrix demonstrate the model's high effectiveness across all categories.

The model achieved an overall accuracy of 96%, with a macro-averaged F1-score of 0.96, indicating strong generalization across all classes. In particular, the Cyst class attained a perfect recall of 1.00 and a precision of 0.90, implying that all cyst cases were correctly identified, though a few non-cyst images were misclassified as cysts. The Normal and Tumor classes showed high precision scores of 0.99, with recall values of 0.97 and 0.95 respectively, suggesting minimal false negatives in these categories. The Stone class achieved a precision of 0.99 and recall of 0.93, indicating slightly lower sensitivity compared to other classes, mainly due to misclassification with the Cyst class.

The confusion matrix provides further insights into class-wise misclassifications. Most Normal images (253/260) were correctly classified, with minor confusion primarily with Tumor (3 cases) and Stone (2 cases). The model excelled in Cyst detection, classifying all 260 instances correctly. However, in the Tumor class, 10 images were misclassified as Cyst, while 2 were labeled as Normal. For Stone, 18 instances were incorrectly classified as Cyst, highlighting a pattern of confusion between Cyst and Stone cases, possibly due to overlapping visual features in certain CT slices.





Fig 4.1.1 Training & validation Accuracy and Loss Graph

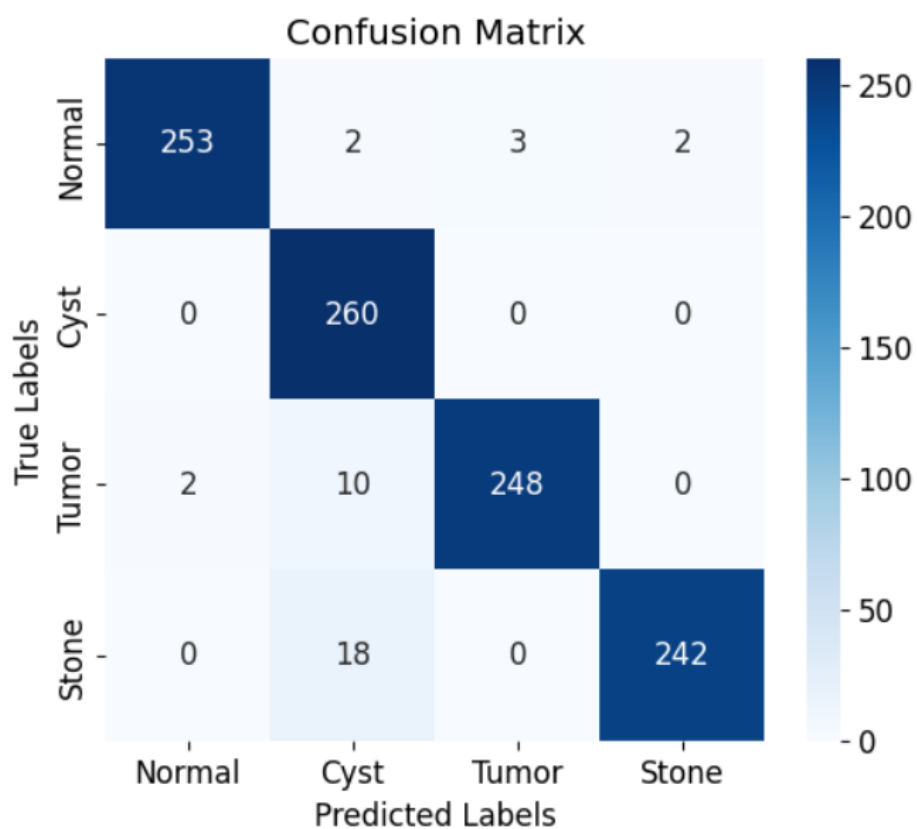


Fig 4.1.2 Confusion Matrix

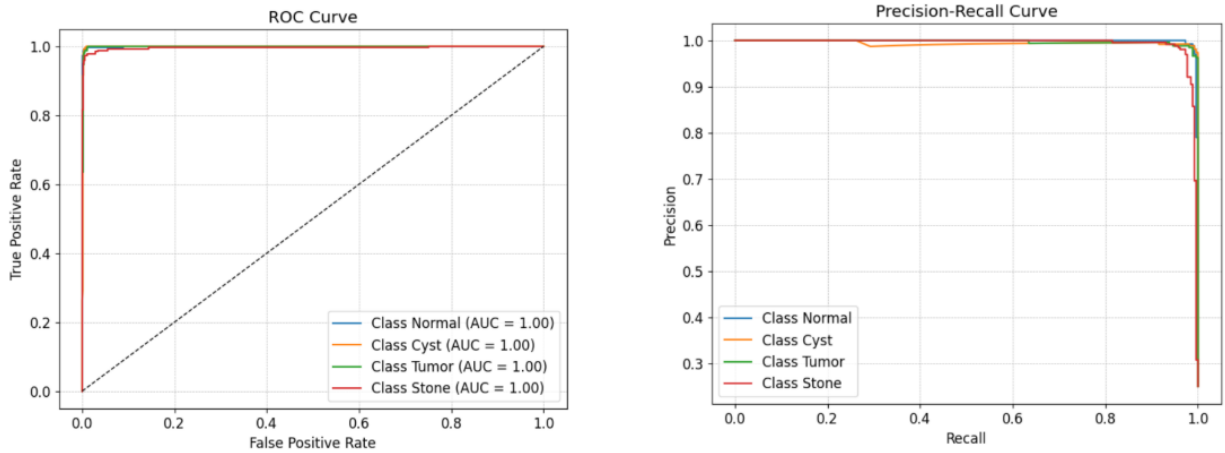


Fig 4.1.3 ROC and Precision-Recall Curve

Classification Report:

	precision	recall	f1-score	support
Normal	0.99	0.97	0.98	260
Cyst	0.98	1.00	0.95	260
Tumor	0.99	0.95	0.97	260
Stone	0.99	0.93	0.96	260
accuracy			0.96	1040
macro avg	0.97	0.96	0.96	1040
weighted avg	0.97	0.96	0.96	1040

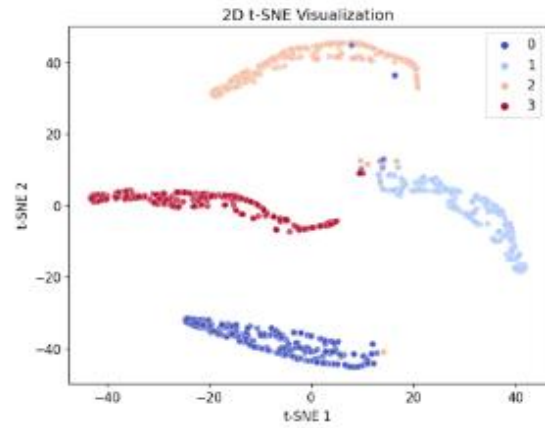


Fig 4.1.4 Classification Report & Last Conv Layer t-SNE visualization

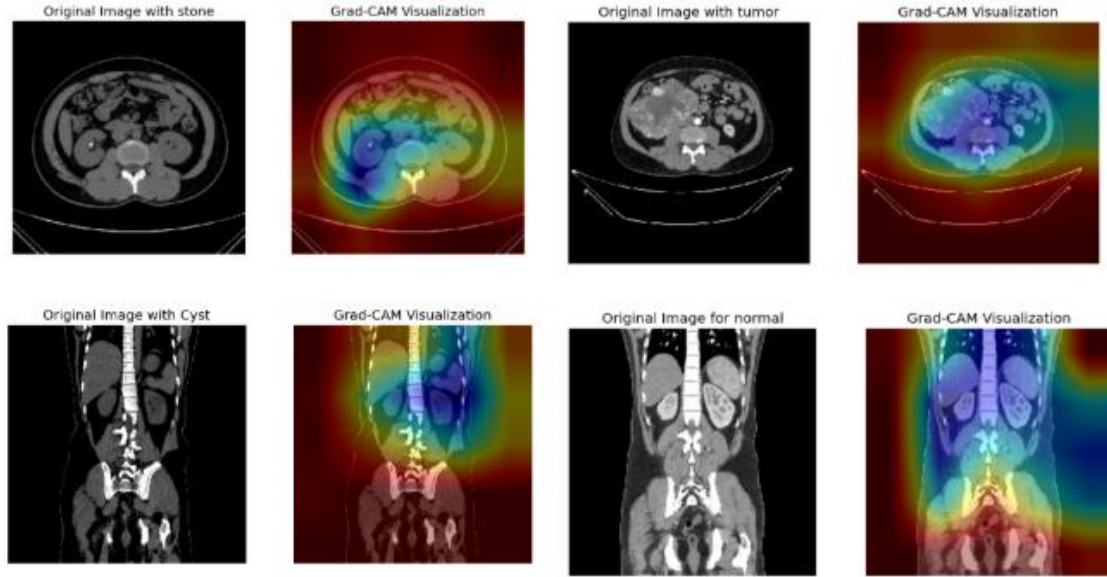


Fig 4.1.5 Grad-cam Visualization for all classes

## 4.2 Meta Learning Results

To further enhance performance, a meta-learning-based stacking ensemble approach was implemented. In this framework, InceptionResNetV2 served as a fixed feature extractor, and extracted deep features were passed to multiple classical machine learning classifiers: Support Vector Machine (SVM), Random Forest, K-Nearest Neighbors (KNN), and XGBoost. Their outputs were then combined using Logistic Regression as the meta-classifier.

This stacking ensemble significantly outperformed the standalone CNN model. It achieved an overall accuracy of 99.33%, with a macro-averaged precision, recall, and F1-score of 99.30%, respectively. Class-wise performance showed exceptional consistency: *Normal*, *Cyst*, *Tumor*, and *Stone* classes all recorded precision and recall values above 99.20%. Notably, the *Cyst* class maintained a perfect recall of 100.00%, and the *Stone* class attained the highest precision of 99.60%, reflecting the model's improved capability in resolving the previous misclassifications observed in the base CNN model.

These results underscore the effectiveness of combining deep learning with classical machine learning via stacking ensembles. The meta-learning approach captured both high-level semantic

features and decision boundaries more robustly than the standalone model, significantly improving classification reliability across all kidney conditions.

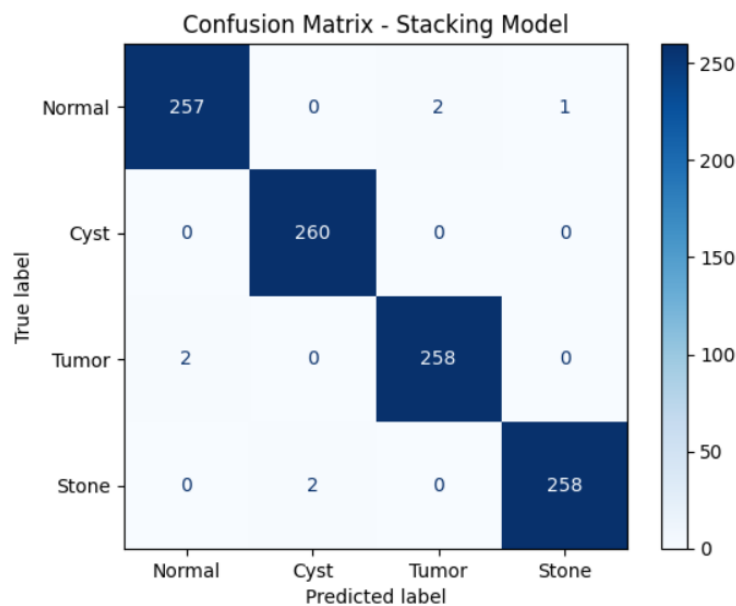


Fig 4.2.1Confusion matrix for Meta Learning

Classification Report (Stacking Ensemble):				
	precision	recall	f1-score	support
0	0.992	0.988	0.990	260
1	0.992	1.000	0.996	260
2	0.992	0.992	0.992	260
3	0.996	0.992	0.994	260
accuracy			0.993	1040
macro avg	0.993	0.993	0.993	1040
weighted avg	0.993	0.993	0.993	1040

Accuracy: 0.9932692307692308

Fig 4.2.2 Classification Report for Meta learning

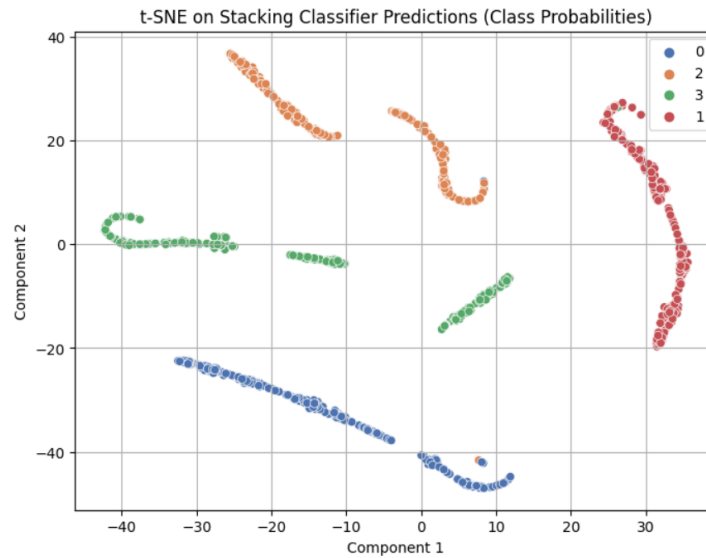


Fig 4.2.3 t-SNE visualization For Meta Learning

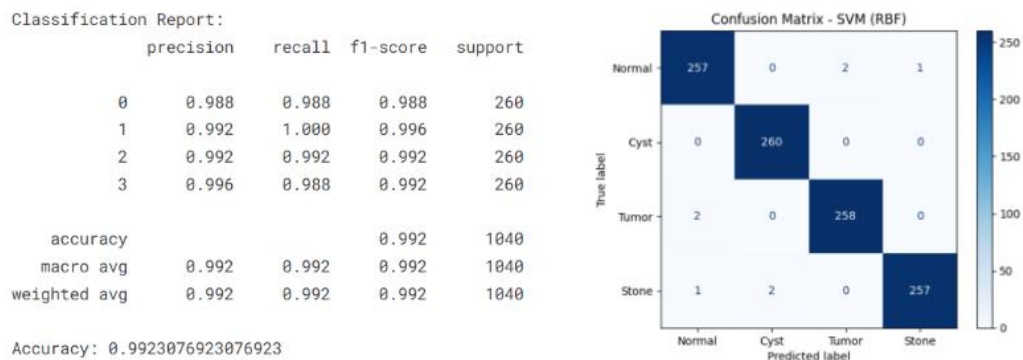


Fig 4.2.4 SVM Classification Result

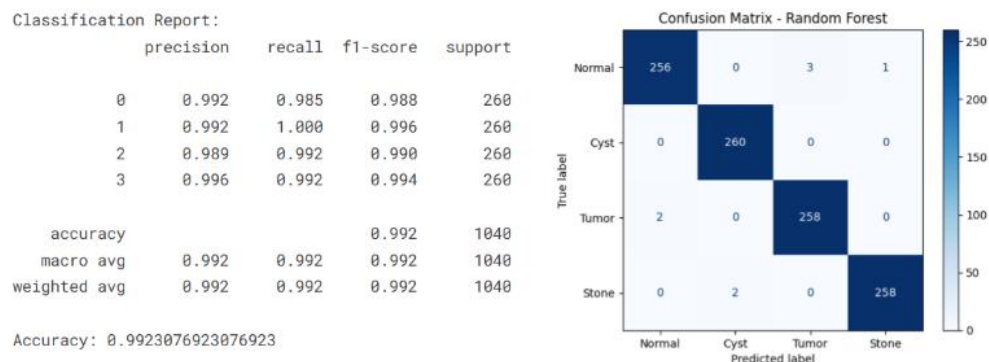


Fig 4.2.5 Random Forest Classification Result

==== K-Nearest Neighbors ====

Classification Report:

	precision	recall	f1-score	support
0	0.992	0.988	0.990	260
1	0.992	1.000	0.996	260
2	0.992	0.992	0.992	260
3	0.996	0.992	0.994	260

accuracy			0.993	1040
macro avg	0.993	0.993	0.993	1040
weighted avg	0.993	0.993	0.993	1040

Accuracy: 0.9932692387692388

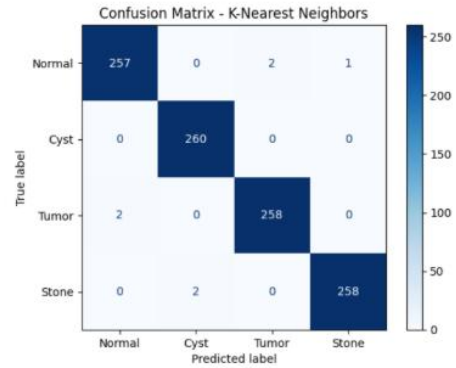


Fig 4.2.6 KNN Classification Result

==== XGBoost ====

Classification Report:

	precision	recall	f1-score	support
0	0.992	0.988	0.990	260
1	0.992	0.992	0.992	260
2	0.989	0.992	0.990	260
3	0.988	0.988	0.988	260

accuracy			0.990	1040
macro avg	0.990	0.990	0.990	1040
weighted avg	0.990	0.990	0.990	1040

Accuracy: 0.9903846153846154

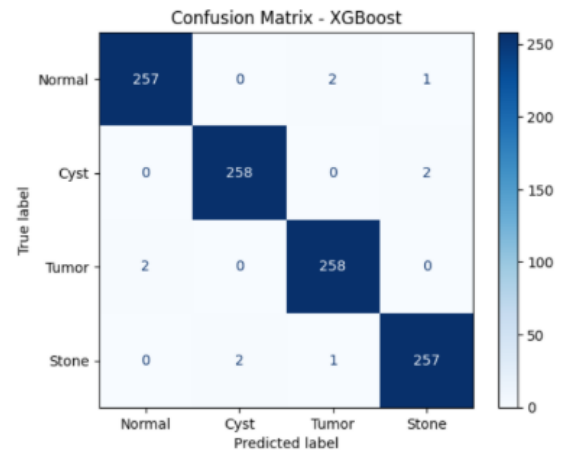


Fig 4.2.7 XGBoost Classification Result

Table 4.1 Comparison with the proposed model

Model	Accuracy	Precision	Recall	F1-score
YOLOv8[17]	82.52%	85.76%	75.28	75.72%
EANET[18]	83.65%	-	-	-
ResNet50[18]	87.92%	-	-	-
ConvNext	89.65%	91%	90%	90%
MobileNetV2[21]	95.29%	-	-	-
EfficientNetV2 - S	95.71%	94%	96%	95%
InceptionV3[21]	97.38%	-	-	-
VGG16[19]	98.00%	-	-	-
CNN[18]	98.66%	-	-	-
Swim Transformer[20]	99.30%	99.15%	99.15%	99.15%
Proposed model	99.33%	99.3%	99.3%	99.3%

## **CHAPTER 5**

### **CONCLUSIONS AND FURTHER WORK**

Kidney disease poses a critical medical challenge, demanding rapid and accurate diagnostic support tools. The proposed model combines deep feature extraction using the InceptionResNetV2 architecture with a meta-learning-based stacking ensemble to significantly enhance classification performance. The methodology was benchmarked against conventional deep learning models and demonstrated superior efficiency and accuracy without increased computational burden. Tested on the CT KIDNEY DATASET: Normal–Cyst–Tumor and Stone, the model achieved an impressive 99.33% classification accuracy, establishing its reliability for automatic kidney abnormality detection. This AI-driven diagnostic aid offers a promising alternative to manual interpretation, supporting radiologists with early and precise decision-making. Future work may involve integration with real-time CT systems and clinical deployment studies to further validate and expand its practical impact.



## CHAPTER 6

### REFERENCES

- [1] A. Hossain et al., "Auto-diagnosis of kidney diseases using vision transformer models with computed tomography images," *Scientific Reports*, vol. XX, no. XX, pp. XX–XX, 2023.
- [2] J. Coresh et al., "Prevalence of chronic kidney disease in the United States," *JAMA*, vol. 298, no. 17, pp. 2038–2047, 2007.
- [3] Global Burden of Disease Study 2016, "Global, regional, and national burden of chronic kidney disease," *Lancet*, 2017.
- [4] A. Grantham, "The diagnosis of renal cysts," *Kidney International*, vol. 50, pp. 1758–1766, 1996.
- [5] M. Slywotzky and R. Bosniak, "Local staging of renal neoplasms with CT," *Radiologic Clinics of North America*, vol. 44, no. 6, pp. 851–866, 2006.
- [6] L. Bosniak, "The Bosniak renal cyst classification," *Radiology*, vol. 179, pp. 621–623, 1991.
- [7] A. Khan et al., "Epidemiology of kidney stone disease," *Nature Reviews Urology*, vol. 13, pp. 369–376, 2016.
- [8] T. Ljungberg et al., "The epidemiology of renal cell carcinoma," *European Urology*, vol. 60, pp. 615–621, 2011.
- [9] D. Miller et al., "CT imaging of the abdomen," *Radiographics*, vol. 20, no. 3, pp. 761–779, 2000.
- [10] M. Foley et al., "Early detection of kidney disease," *Nephrology Dialysis Transplantation*, vol. 22, pp. 181–188, 2007.
- [11] K. Doi, "Computer-aided diagnosis in medical imaging," *IEEE Engineering in Medicine and Biology Magazine*, vol. 17, no. 3, pp. 53–60, 1999.

- [12] International Society of Nephrology, "Global kidney health atlas," ISN Report, 2019.
- [13] A. Hossain et al., "CT Kidney Dataset: Normal-Cyst-Tumor-Stone," Mendeley Data, v1, 2023.
- [14] Peng, C., Liu, Y., Yuan, X., & Chen, Q. (2022). Research of image recognition method based on enhanced Inception-ResNet-V2. *Multimedia Tools and Applications*, 81, 34345–34365.
- [15] Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. (2017). *Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning*. Proceedings of the AAAI Conference on Artificial Intelligence, 31(1).
- [16] He, K., Zhang, X., Ren, S., & Sun, J. (2016). *Deep Residual Learning for Image Recognition*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770–778.
- [17] A. S. Khan, M. A. Khan and A. Mahmood, "Multi-Class Classification of Kidney Diseases Using Deep CNN and Explainable AI," in *IEEE Access*, vol. 12, pp. 26342–26355, 2024, doi: 10.1109/ACCESS.2024.3355691.
- [18] N. Jayachitra and M. Nithya, "Kidney Disease Detection from CT Images using a Customized CNN Model and Deep Learning," in (*ICICICT*), Kannur, India, 2023, pp. 565–570, doi: 10.1109/ICICICT58849.2023.10384368.
- [19] S. P. Sowmya and D. Kumar, "Detecting Multi-Class Kidney Abnormalities Using Deep Learning," in Tumakuru, India, 2022, pp. 1–6, doi: 10.1109/ICSTCEE57018.2022.10008048.
- [20] T. Paul and S. S. Das, "Vision Transformer and Explainable Transfer Learning Models for Auto Detection of Kidney Cyst, Stone and Tumor from CT-Radiography," in *2023 IEEE World AI IoT Congress (AIIoT)*, Seattle, WA, USA, 2023, pp. 1–6, doi: 10.1109/AIIoT58230.2023.10160294.
- [21] A. S. Khan, M. A. Khan, and A. Mahmood, "Kidney Tumor Segmentation and Classification using Deep Neural Network on CT Images," in Kannur, India, 2023, pp. 565–570, doi: 10.1109/ICICICT58849.2023.10384368.

- [22] <https://www.kaggle.com/datasets/nazmul0087/ct-kidney-dataset-normal-cyst-tumor-and-stone> (Dataset)
- [23] Wiatowski, T., & Bölcskei, H. (2018). "A mathematical theory of deep convolutional neural networks for feature extraction," *IEEE Transactions on Information Theory*, vol. 64, no. 3, pp. 1845–1866.
- [24] Finn, C., Abbeel, P., & Levine, S. (2017). "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. 34th Int. Conf. Mach. Learn. (ICML)*, pp. 1126–1135.
- [25] Chen, T., & Guestrin, C. (2016). "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Min. (KDD)*, pp. 785–794.
- [26] Cortes, C., & Vapnik, V. (1995). "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297.
- [27] Breiman, L. (2001). "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32.
- [28] Cover, T., & Hart, P. (1967). "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27.
- [29] Cox, D. R. (1958). "The regression analysis of binary sequences," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 20, no. 2, pp. 215–242.
- [30] J. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, "Meta-learning in neural networks: A survey", vol. 44, no. 9, pp. 5149–5169, Sep. 2022.
- [31] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, Nov. 2008.
- [32] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 618–626.

## CHAPTER 7

## APPENDIX

### 7.1 SIMILARITY CHECK REPORT

### 7.2 SOURCE CODE

#### 7.2.1 IMPORTING LIBRARIES

```
import os
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
import tensorflow as tf
import cv2
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import InceptionResNetV2
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, Flatten, Dropout, GlobalAveragePooling2D, BatchNormalization
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLROnPlateau, CSVLogger, TerminateOnNaN, LearningRateScheduler
from sklearn.model_selection import train_test_split
```

## 7.2.2 READ DATA AND STORING IN DATAFRAME

```
# Define base directory and classes
base_dir = r'/kaggle/input/ct-kidney-dataset-normal-cyst-tumor-and-stone/CT-KIDNEY-DATASET-Normal-Cyst-Tumor-Stone/CT-KIDNEY-DATASET-Normal-Cyst-Tumor-Stone'
classes = ['Normal', 'Cyst', 'Tumor', 'Stone']
label_mapping = {'Normal': 0, 'Cyst': 1, 'Tumor': 2, 'Stone': 3}
```

```
def make_balanced_df(classes, base_dir, samples_per_class=1300):
    data = []
    for label in classes:
        folder_path = os.path.join(base_dir, label)
        all_files = sorted([os.path.join(folder_path, f) for f in os.listdir(folder_path) if
                             f.endswith(('jpg', 'png'))])
        sampled_files = np.random.choice(all_files, samples_per_class, replace=False)
        data.extend([(fp, label) for fp in sampled_files])
    df = pd.DataFrame(data, columns=['file_path', 'label'])
    df['label'] = df['label'].map(label_mapping).astype(str)
    return df

# Load balanced data
df = make_balanced_df(classes, base_dir)
```

## 7.2.3 CREATING IMAGE GENERATOR

```
# Data augmentation
image_generator = ImageDataGenerator(
    rescale=1.0/255,
    rotation_range=15,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.1,
    zoom_range=0.1,
    horizontal_flip=True
)

test_val_generator = ImageDataGenerator(rescale=1.0/255)

# Creating generators
training_set = image_generator.flow_from_dataframe(
    dataframe=train_df, x_col='file_path', y_col='label',
    target_size=(224, 224), batch_size=32, class_mode='categorical', color_mode='rgb', shuffle=True,
)
test_set = test_val_generator.flow_from_dataframe(
    dataframe=test_df, x_col='file_path', y_col='label',
    target_size=(224, 224), batch_size=32, class_mode='categorical', color_mode='rgb', shuffle=False
)

validation_set = test_val_generator.flow_from_dataframe(
    dataframe=valid_df, x_col='file_path', y_col='label',
    target_size=(224, 224), batch_size=32, class_mode='categorical', color_mode='rgb', shuffle=False
)
```

## 7.2.4 GENERIC MODEL CREATION

```

# Model setup
base_model = InceptionResNetV2(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(1024, activation='relu')(x)
x = BatchNormalization()(x)
x = Dropout(0.5)(x)
x = Dense(512, activation='relu')(x)
x = BatchNormalization()(x)
x = Dropout(0.3)(x)
x = Dense(256, activation='relu')(x)
x = BatchNormalization()(x)
x = Dropout(0.3)(x)
output_layer = Dense(4, activation='softmax')(x)
model = Model(inputs=base_model.input, outputs=output_layer)

```

## 7.2.5 MODEL COMPILE AND TRAINING

```

# Compile model
optimizer = tf.keras.optimizers.Adam(learning_rate=1e-4)
model.compile(optimizer=optimizer, loss='categorical_crossentropy', metrics=['accuracy'])

# Callbacks
model_checkpoint = ModelCheckpoint(
    filepath='best_model.keras', monitor='val_accuracy', save_best_only=True, verbose=1, mode='max'
)
early_stopping = EarlyStopping(monitor='val_accuracy', patience=10, mode='max', restore_best_weights=True)
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=3, verbose=1, mode='min', min_lr=1e-6)
csv_logger = CSVLogger('training_log.csv')
callbacks = [early_stopping, reduce_lr, model_checkpoint, csv_logger]

```

```

# Train the model
history = model.fit(
    training_set,
    validation_data=validation_set,
    epochs=50,
    callbacks=callbacks
)

```

## 7.2.6 LOADING MODEL:

```

from tensorflow.keras.models import load_model

# Load your previously trained model
model = load_model("/kaggle/input/inceptionresnetv2-balanced/tensorflow2/default/1/InceptionResNetv2_Balanced.keras")

```

## 7.2.7 t-SNE AND GRAD-CAM VISUALIZATION:

```
# t-SNE Visualization
features = model.predict(test_set)
tsne = TSNE(n_components=2, random_state=42)
features_2d = tsne.fit_transform(features)
plt.figure(figsize=(8, 6))
sns.scatterplot(x=features_2d[:, 0], y=features_2d[:, 1], hue=y_true, palette='coolwarm', alpha=0.7)
plt.xlabel('t-SNE 1')
plt.ylabel('t-SNE 2')
plt.title('2D t-SNE Visualization')
plt.legend()
plt.savefig("tsne_2d.pdf")
plt.show()
plt.close()
```

```
def grad_cam(model, img_array, layer_name):
    grad_model = Model(inputs=model.input, outputs=[model.get_layer(layer_name).output, model.output])
    with tf.GradientTape() as tape:
        conv_outputs, predictions = grad_model(img_array)
        loss = predictions[:, np.argmax(predictions[0])]
        grads = tape.gradient(loss, conv_outputs)
        pooled_grads = K.mean(grads, axis=(0, 1, 2))
        heatmap = tf.reduce_mean(tf.multiply(pooled_grads, conv_outputs), axis=-1)
        heatmap = np.maximum(heatmap, 0)[0]
        heatmap /= np.max(heatmap)
    return heatmap
```

```
def generate_gradcam(model, image_array, last_conv_layer_name, pred_index=None):
    grad_model = tf.keras.models.Model(
        inputs=model.input,
        outputs=[model.get_layer(last_conv_layer_name).output, model.output]
    )
    with tf.GradientTape() as tape:
        conv_outputs, predictions = grad_model(image_array)
        if pred_index is None:
            pred_index = np.argmax(predictions[0]) # Take highest probability class
            class_output = predictions[:, pred_index]

        # Compute gradients
        grads = tape.gradient(class_output, conv_outputs)

        # Compute global average pooling over the gradients
        pooled_grads = tf.reduce_mean(grads, axis=(0, 1, 2))

        # Multiply pooled gradients with feature maps
        conv_outputs = conv_outputs[0]
        heatmap = np.mean(conv_outputs * pooled_grads, axis=-1)

        # Normalize the heatmap
        heatmap = np.maximum(heatmap, 0)
        heatmap /= np.max(heatmap)

    return heatmap
```

## 7.2.8 FEATURE EXTRACTION

```
from tqdm import tqdm

def extract_features(df):
    features = []
    labels = []
    for i, row in tqdm(df.iterrows(), total=len(df)):
        img = load_img(row['file_path'], target_size=(224, 224))
        img_array = img_to_array(img) / 255.0
        img_array = np.expand_dims(img_array, axis=0)
        feature = feature_extractor.predict(img_array, verbose=0)
        features.append(feature.flatten())
        labels.append(row['label'])
    return np.array(features), np.array(labels)
```

```
# Extract features
from tensorflow.keras.preprocessing.image import load_img, img_to_array

X_train, y_train = extract_features(train_df)
X_valid, y_valid = extract_features(valid_df)
X_test, y_test = extract_features(test_df)
```

## 7.2.9 STACKING CLASSIFIER

```
# Define base learners
base_learners = [
    ('svm', SVC(kernel='rbf', probability=True)),
    ('xgb', XGBClassifier(use_label_encoder=False, eval_metric='mlogloss')),
    ('rf', RandomForestClassifier(n_estimators=100, random_state=42)),
    ('knn', KNeighborsClassifier(n_neighbors=5))
]

# Define meta learner
meta_learner = LogisticRegression(max_iter=1000)
```

```
# Create stacking classifier
stacking_model = StackingClassifier(estimators=base_learners, final_estimator=meta_learner, cv=5)
stacking_model.fit(X_train_scaled, y_train)
```



## 7.2.10 EVALUATION

```
# Evaluation
print("Classification Report (Stacking Ensemble):")
print(classification_report(y_test, y_pred, digits=3))
print("Accuracy:", accuracy_score(y_test, y_pred))
```

```
# Convert string labels to integers
y_test_numeric = [int(label) for label in y_test]
y_pred_numeric = [int(label) for label in y_pred]

# Confusion Matrix
cm = confusion_matrix(y_test_numeric, y_pred_numeric, labels=[0, 1, 2, 3])

# Display confusion matrix with class names
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=classes)
disp.plot(cmap=plt.cm.Blues)
plt.title("Confusion Matrix - Stacking Model")
plt.show()
```

```
# Set font style globally
plt.rcParams["font.family"] = "Times New Roman"
plt.rcParams["font.size"] = 12

plt.figure(figsize=(8, 6))

# Ensure y_true is a NumPy array
y_true = np.array(y_true)
for i in range(len(class_names)):
    # Convert y_true to binary labels for class i
    y_true_binary = (y_true == i).astype(int)

    # Compute ROC curve and AUC score
    fpr, tpr, _ = roc_curve(y_true_binary, y_pred_probs[:, i])
    roc_auc = auc(fpr, tpr)

    # Plot ROC curve
    plt.plot(fpr, tpr, label=f'Class {class_names[i]} (AUC = {roc_auc:.2f})')

# Add diagonal reference line
plt.plot([0, 1], [0, 1], 'k--', linewidth=1)

# Labels and title
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc='lower right')
plt.grid(True, linestyle='--', linewidth=0.5)
```