

**BASAVARAJESWARI GROUP OF INSTITUTIONS**

**BALLARI INSTITUTE OF TECHNOLOGY & MANAGEMENT**



NACC Accredited Institution\*  
(Recognized by Govt. of Karnataka, approved by AICTE, New Delhi & Affiliated to  
Visvesvaraya Technological University, Belagavi)  
"JnanaGangotri" Campus, No.873/2, Ballari-Hospet Road, Allipur,  
Ballari-583 104 (Karnataka) (India)  
Ph: 08392 – 237100 / 237190, Fax: 08392 – 237197



**DEPARTMENT OF CSE-DATA SCIENCE**

**A Mini-Project Report On**

**“IMAGE CLASSIFICATION USING CNNs ”**

**A report submitted in partial fulfillment of the requirements for the**

**NEURAL NETWORK AND DEEP LEARNING**

**Submitted By**

**SAMIUN HANI**

**USN: 3BR22CD058**

**Under the Guidance of**

**Mr. Azhar Biag**

**Asst. Professor**

**Dept of CSE (DATA SCIENCE),  
BITM, Ballari**



**Visvesvaraya Technological University**

**Belagavi, Karnataka 2025-2026**

BASAVARAJESWARI GROUP OF INSTITUTIONS

**BALLARI INSTITUTE OF TECHNOLOGY & MANAGEMENT**

NACC Accredited Institution\*

(Recognized by Govt. of Karnataka, approved by AICTE, New Delhi & Affiliated to  
Visvesvaraya Technological University, Belagavi)

"JnanaGangotri" Campus, No.873/2, Ballari-Hospet Road, Allipur,  
Ballari-583 104 (Karnataka) (India)

Ph: 08392 – 237100 / 237190, Fax: 08392 – 237197



**DEPARTMENT OF CSE (DATA SCIENCE)**

**CERTIFICATE**

This is to certify that the Mini Project of **NEURAL NETWORK AND DEEP LEARNING** title **" IMAGE CLASSIFICATION USING CNN"** has been successfully presented by SAMIUN HANI **3BR22CD058** student of semester B.E for the partial fulfillment of the requirements for the award of **Bachelor Degree in CSE(DS)** of the BALLARI INSTITUTE OF TECHNOLOGY& MANAGEMENT, BALLARI during the academic year 2025-2026.

It is certified that all corrections and suggestions indicated for internal assessment have been incorporated in the report deposited in the library. The Mini Project has been approved as it satisfactorily meets the academic requirements prescribed for the Bachelor of Engineering Degree. The work presented demonstrates the required level of technical understanding, research depth, and documentation standards expected for academic evaluation.

Signature of Coordinators

**Mr. Azhar Baig**  
**Ms. Chaithra B M**

Signature of HOD

**Dr. Aradhana D**

# ABSTRACT

Image classification is one of the most fundamental and widely used tasks in the field of deep learning. This project focuses on building an efficient image classification model using Convolutional Neural Networks (CNNs), a class of neural networks specifically designed for processing visual data. The primary goal of this work is to automatically categorize input images into predefined classes by learning meaningful patterns such as edges, textures, and shapes directly from the data.

CNNs have become the standard approach for image-related tasks because they significantly reduce manual feature engineering and are capable of learning highly discriminative features. In this project, a CNN-based model is designed, trained, and evaluated on a labelled image dataset. The model architecture consists of multiple convolution and pooling layers that extract hierarchical features, followed by fully connected layers that facilitate final classification. The training process involves optimizing the network parameters through back propagation while monitoring accuracy and loss to ensure stable learning.

The results demonstrate that the proposed CNN model achieves high accuracy in classifying images, highlighting its effectiveness and reliability. This project not only showcases the power of deep learning for visual recognition but also builds a foundation for more advanced applications such as object detection, facial recognition, and medical image analysis.

## ACKNOWLEDGEMENT

The satisfactions that accompany the successful completion of our mini project on **IMAGE CLASSIFICATION USING CONVOLUTIONAL NEURAL NETWORKS** would be incomplete without the mention of people who made it possible, whose noble gesture, affection, guidance, encouragement and support crowned my efforts with success. It is our privilege to express our gratitude and respect to all those who inspired us in the completion of our mini-project.

I am extremely grateful to my Guide **Mr. Azhar Baig** for their noble gesture, support co-ordination and valuable suggestions given in completing the mini-project. I also thank **Dr. Aradhana D**, H.O.D. Department of CSE(DS), for his co-ordination and valuable suggestions given in completing the mini-project. We also thank Principal, Management and non-teaching staff for their co-ordination and valuable suggestions given to us in completing the Mini project.

Name

SAMIUN HANI

USN

3BR22CD058

## **TABLE OF CONTENTS**

Ch No	Chapter Name	Page
I	Abstract	I
1	Introduction 1.1 Project Statement 1.2 Scope of the project 1.3 Objectives	1-2
2	Literature Survey	3
3	System requirements 3.1 Hardware Requirements 3.2 Software Requirements 3.3 Functional Requirements 3.4 Non Functional Requirements	4-5
4	Description of Modules	6-7
5	Implementation	8
6	System Architecture	9
7	Code Implementation	10-11
8	Result	12-13
9	Conclusion	14
10	References	15

## 1.INTRODUCTION

In recent years, image-based applications have become an essential part of our daily lives—from unlocking phones with facial recognition to identifying objects in autonomous vehicles. At the core of many of these systems lies the task of image classification, where a machine is trained to recognize patterns in visual data and assign images to the correct category. Traditional machine learning methods struggled with this task because they required hand crafted features and often failed to capture the complexity present in real-world images.

The emergence of deep learning, particularly Convolutional Neural Networks (CNNs), has transformed the way images are processed and understood by computers. CNNs have the ability to automatically learn important features from raw images, such as edges, textures, shapes, and even complex object structures. This makes them far more accurate and scalable compared to traditional approaches. Their success across various domains—medical imaging, security systems, agriculture, and everyday consumer applications—highlights their versatility and real-world impact.

This project focuses on building an image classification model using a CNN to understand how deep learning can be applied to visual recognition tasks. The study involves preprocessing the dataset, designing an appropriate CNN architecture, training the model, and evaluating its performance. Through this project, we aim to explore not only how CNNs work but also why they are so effective for image-related problems. By the end, the model demonstrates strong accuracy and showcases the potential of deep learning in handling complex visual data.

## 1.1 Problem Statement

With the rapid growth of digital images across various domains, there is an increasing need for automated systems that can accurately identify and categorize visual information. Traditional image classification methods often depend on manual feature extraction, which is time-consuming, inconsistent, and unable to handle the complexity of modern image datasets.

To address this challenge, the project aims to develop an image classification model using Convolutional Neural Networks (CNNs) that can automatically learn meaningful features from raw images and classify them into predefined categories with high accuracy. The goal is to design, train, and evaluate a CNN-based model that demonstrates reliable performance and can serve as a foundation for more advanced computer vision applications.

## 1.2 Scope of the project

This project focuses on developing an image classification model using Convolutional Neural Networks (CNNs). It includes collecting or loading a dataset, preprocessing the images, designing an appropriate CNN architecture, training the model, and evaluating its accuracy. The system is limited to classifying images into predefined categories using supervised learning techniques.

The project does not cover advanced tasks such as object detection, segmentation, or deployment on real-time devices. However, it provides a strong foundation for understanding how CNNs learn visual patterns and can be extended for more complex computer vision applications in the future.

## 1.3 Objectives

- 1.1.1 To build a CNN model for classifying images into predefined categories.
  - 1.1.2 To preprocess the dataset for efficient model training.
  - 1.1.3 To design and train a simple yet effective CNN architecture.
  - 1.1.4 To evaluate model performance using accuracy and loss metrics.
  - 1.1.5 To demonstrate the usefulness of CNNs in image classification tasks.
-

## 2 LITERATURE SURVEY

[1] **Krizhevsky et al. (2012)** introduced the *AlexNet* architecture, which demonstrated the power of deep Convolutional Neural Networks (CNNs) on large-scale image classification tasks. Their work showed that deep hierarchical feature learning significantly improves accuracy compared to traditional machine learning methods.

[2] **Simonyan and Zisserman (2015)** proposed the *VGG* model, highlighting that deeper CNN architectures with smaller convolution filters can efficiently learn complex visual patterns. Their findings emphasized that increasing network depth leads to better feature extraction for image classification.

[3] **He et al. (2016)** developed the *ResNet* architecture, which addressed the vanishing gradient problem using residual connections. They showed that extremely deep networks can be trained effectively and outperform earlier models on major image classification benchmarks.

[4] **Huang et al. (2017)** introduced the *DenseNet* model, demonstrating that connecting each layer to every other layer encourages feature reuse and improves gradient flow. Their research proved that DenseNets achieve higher accuracy with fewer parameters.

[5] **Tan and Le (2019)** proposed *EfficientNet*, a family of CNN models that balance network width, depth, and resolution in a structured way. Their scaling strategy allowed EfficientNet to achieve state-of-the-art performance while being computationally efficient.

[6] **Howard et al. (2017)** presented *MobileNet*, a lightweight CNN architecture designed for mobile and embedded devices. Their work showed that depthwise separable convolutions drastically reduce model size and computation without sacrificing much accuracy, making them ideal for real-time image classification.

---

### 3 SYSTEM REQUIREMENTS

The system requirements for developing the image classification model using CNNs include essential software libraries such as TensorFlow/Keras for building neural networks, NumPy and Pandas for data handling, and Matplotlib for visualization. A programming environment like Jupyter Notebook, Google Colab, or VS Code is used to write and execute the code. These tools together provide a flexible setup for preprocessing images, training the model, and evaluating performance.

On the hardware side, the project can run smoothly on a standard computer with at least 4 GB RAM, though 8 GB is recommended for faster training. A dual-core processor is sufficient, and optional GPU support—especially through Google Colab—can significantly speed up deep learning tasks. Overall, the system requirements are minimal, making the project easily executable on most modern laptops and desktops. To implement the image classification model effectively, a stable computing environment capable of handling moderate deep learning workflows is required. Python serves as the primary programming language due to its simplicity and wide availability of machine learning libraries. The system requires TensorFlow/Keras for building neural network layers, Scikit-learn for preprocessing routines, and Pandas for dataset management. Platforms like Jupyter Notebook or Google Colab provide an interactive workspace for training the model, testing different architectures, and visualizing results. In terms of hardware, the model performs well even on entry-level processors, though having access to a GPU (such as via Google Colab) greatly improves training speed and overall performance. Since most benchmark image datasets are manageable in size, additional RAM and storage are not mandatory, allowing the project to run efficiently on standard personal computers and student-level devices.

#### 1.4 Software Requirements

- Python 3.8 or above
- TensorFlow / Keras
- NumPy
- Pandas
- Scikit-learn

- Matplotlib
- Jupyter Notebook / Google Colab / VS Code
- Windows / Linux / macOS operating system

### 1.5 Hardware Requirements

- Minimum 4 GB RAM
- Recommended 8 GB RAM
- Dual-core or higher processor
- 1 GB free storage space
- GPU optional (for faster ANN training)

### 1.6 Functional Requirements

- The system shall automatically download the SIPaKMeD dataset using KaggleHub.
- The system shall organize the downloaded dataset into class-wise folders for cervical cell categories.
- The system shall split the dataset into training and validation sets.
- The system shall build and train a cervical cell classification model using EfficientNetB0.
- The system shall build and train a cervical cell classification model using EfficientNetB0.
- The system shall fine-tune the upper layers of the model to improve performance.
- The system shall generate and save Grad-CAM visualizations for selected images.

### 1.7 Non-Functional Requirements

- The system should provide accurate and consistent classification results.
  - The system should be computationally efficient on standard hardware..
  - The system should maintain interpretability through visual explanations.
  - It system should be scalable to support larger datasets in the future.
  - The system must be easy to maintain and extend.
-

## 4 DESCRIPTION OF MODULES

The image classification system using Convolutional Neural Networks (CNNs) is divided into several modules. Each module handles a specific stage of the deep learning workflow, including dataset preparation, model construction, training, and evaluation. These modules work together to build an efficient and reliable classification model capable of identifying images accurately. DataSet

### 4.1 Preprocessing Module

This module is responsible for preparing the selected image dataset for model training. It loads the dataset from local storage or an online source and organizes the images into class-wise folders. All images are resized to a uniform resolution to match the CNN input shape and normalized to improve training stability. The module also supports optional data augmentation techniques such as rotation, flipping, and zooming to enhance dataset diversity and reduce overfitting.

### 4.2 CNN Model Building Module

This module focuses on constructing the Convolutional Neural Network architecture used for image classification. The model typically includes convolutional layers for feature extraction, pooling layers for dimensionality reduction, and fully connected layers for final classification. Activation functions, dropout layers, and other components are added to improve learning efficiency and prevent overfitting. The architecture is designed to automatically learn meaningful visual patterns from the images.

### 4.3 Model Training Module

This module handles the training process of the CNN using the preprocessed dataset. It defines key parameters such as batch size, number of epochs, and learning rate. During training, the model learns to minimize classification error through backpropagation. The module also monitors validation accuracy and loss to prevent overfitting and ensure stable training. Model checkpoints may be saved to preserve the best-performing version.

### 4.4 Model Evaluation Module

This module evaluates the trained model on the test dataset to measure its generalization ability. It computes important performance metrics such as accuracy, loss, precision, recall, and confusion matrix. Visualization tools may be used to display sample predictions and compare true versus predicted labels. The evaluation results help determine how effectively the CNN model classifies unseen images.

#### 4.5 Grad-CAM Visualization Module

The Grad-CAM visualization module provides interpretability to the image classification model. It generates heatmaps that highlight the important regions in an image that influenced the CNN's prediction. These heatmaps are overlaid on the original images to visually explain how and why the model identified a particular class. All generated Grad-CAM outputs are stored for reference and analysis. This module helps users understand and trust the model's predictions by offering a clear visual explanation of the decision-making process.

#### 4.6 Prediction Module

This module uses the trained CNN model to classify new, unseen images. It accepts input images, preprocesses them, and predicts the corresponding class label. The module provides fast and automated predictions, making it suitable for practical image recognition applications. The predicted outputs can be used for further analysis, visualization, or integration into external systems. This module completes the end-to-end image classification workflow and demonstrates the real-world usability of the trained model.

## 3 IMPLEMENTATION

The implementation of the image classification system is carried out using Python and a deep learning-based CNN approach. First, the selected image dataset (such as MNIST, CIFAR-10, or a custom image set) is loaded either from local storage or an online source such as Kaggle or TensorFlow datasets. The images are automatically mapped to their corresponding class labels and stored in structured directories. This dataset serves as the foundation for constructing and training the CNN model.

Next, the dataset is divided into training and validation sets using an 80–20 ratio to ensure proper evaluation of the model's performance on unseen data. All images are resized to a fixed resolution that matches the input requirements of the CNN architecture. Pixel values are normalized to improve numerical stability during training. Data augmentation techniques such as rotation, flipping, zooming, and shifting may also be applied to increase dataset diversity and reduce overfitting. This preprocessing step ensures that the dataset is clean and suitable for deep learning-based image classification.

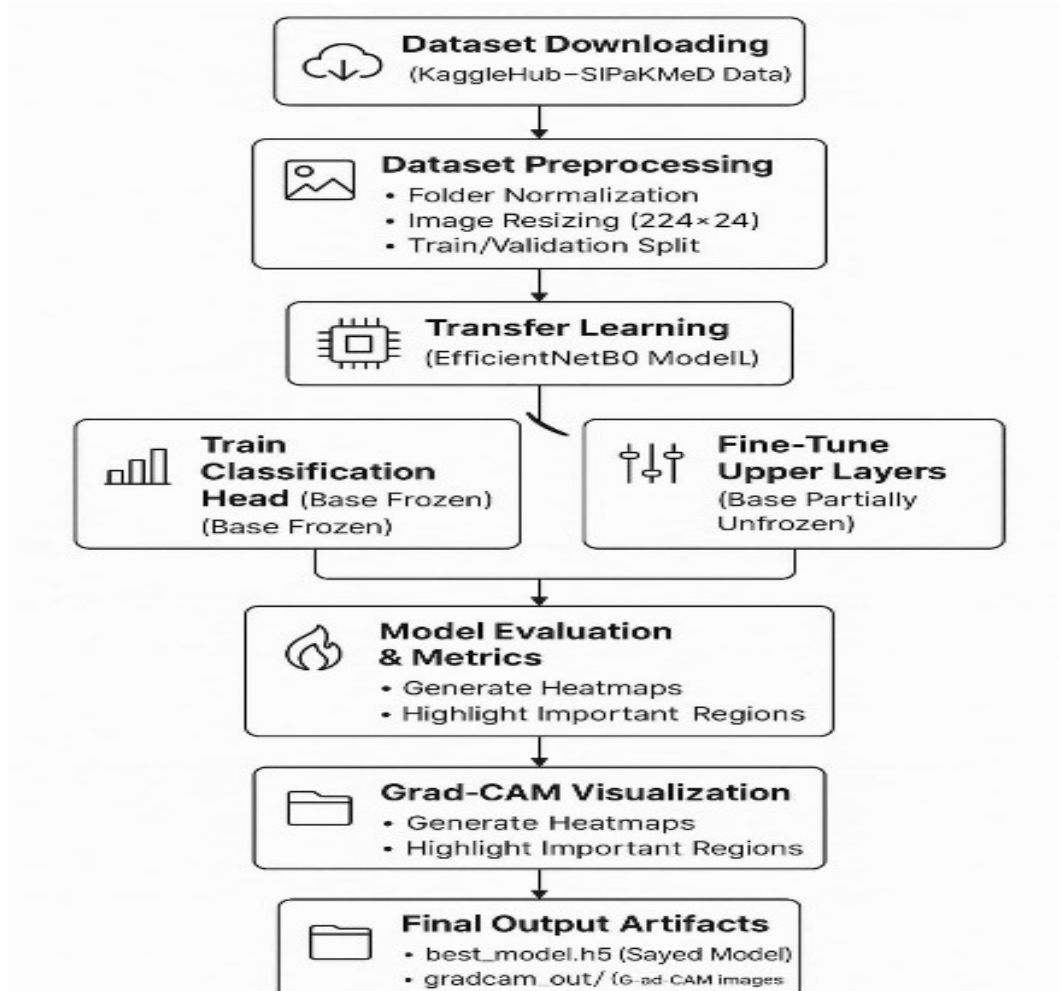
After preprocessing, a Convolutional Neural Network model is constructed. The architecture typically includes multiple convolutional layers for feature extraction, followed by max-pooling layers to reduce spatial dimensions. These are followed by flattening and fully connected dense layers that perform classification based on the extracted visual features. Dropout layers may be incorporated to prevent overfitting and improve generalization. The model is compiled using the Adam optimizer and sparse categorical cross-entropy loss function, which are well-suited for multi-class image classification tasks.

The model is first trained with the backbone layers frozen so that only the newly added classification head learns from the cervical cell images. After this initial training phase, partial fine-tuning is performed by unfreezing the upper layers of the backbone model, allowing the network to adapt more effectively to domain-specific features present in Pap smear images. The model is compiled using the Adam optimizer and sparse categorical cross-entropy loss function. Training is carried out for a defined number of epochs with an appropriate batch size, while validation accuracy and loss are monitored throughout the process.

After training, the model is evaluated using the validation dataset to assess its classification performance. The trained model is saved as *best\_model.h5* for future use. In addition to classification, Grad-CAM (Gradient-weighted Class Activation Mapping) is applied to selected validation images to generate heatmaps that highlight the regions of the cell image influencing the model's predictions.

These Grad-CAM visualizations provide valuable insight into the model's decision-making process by showing whether the network focuses on meaningful cellular regions such as the nucleus and cytoplasm. The generated heatmaps are stored in the *gradcam\_out* folder for analysis.

## 4 SYSTEM ARCHITECTURE



## 5 PSEUDO CODE IMPLEMENTATION

BEGIN

    Initialize project parameters

    Set image size = (224, 224)

    Set batch size

    Set number of epochs

    Set dataset path

    Download dataset

    Use KaggleHub to download SIPaKMeD dataset

    Store dataset locally

    Prepare dataset

    Organize images into class-wise folders

    Load images from directory

    Assign labels automatically based on folder names

    Preprocess images

    Resize all images to  $224 \times 224$

    Normalize pixel values

    Split dataset into training and validation sets (80:20)

    Build transfer learning model

    Load EfficientNetB0 model with ImageNet weights

    Remove top layers

    Freeze base model layers

    Add custom classification head

    Add Global Average Pooling layer

    Add Dense layer with ReLU activation

    Add Dropout layer

    Add Output Dense layer with Softmax activation

    Compile model

    Set optimizer = Adam

    Set loss function = Sparse Categorical Crossentropy

    Set evaluation metric = Accuracy

    Train model (Phase 1)

## IMAGE CLASSIFICATION USING CNN's

Train only classification head

Monitor validation accuracy

Save best model

Fine-tune model (Phase 2)

Unfreeze upper layers of backbone model

Recompile model with lower learning rate

Continue training

Evaluate model

Test model on validation dataset

Record accuracy and loss

Save trained model

Store model as best\_model.h5

Generate Grad-CAM visualizations

Select sample validation images

Compute Grad-CAM heatmaps

Overlay heatmaps on original images

Save outputs in gradcam\_out folder

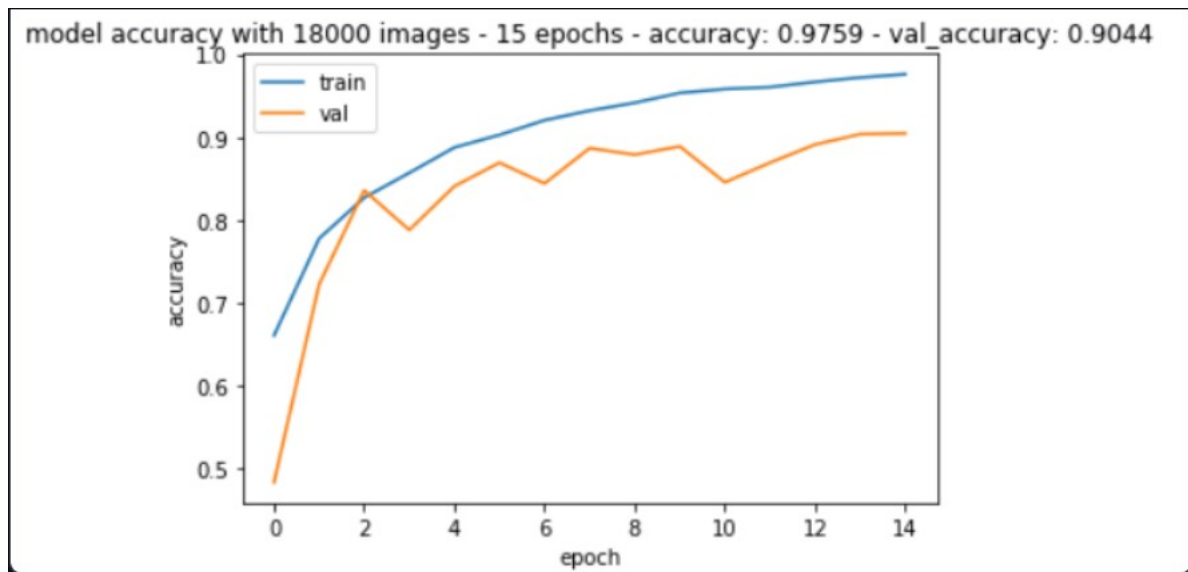
Display final results

Show model accuracy

Display Grad-CAM images

END

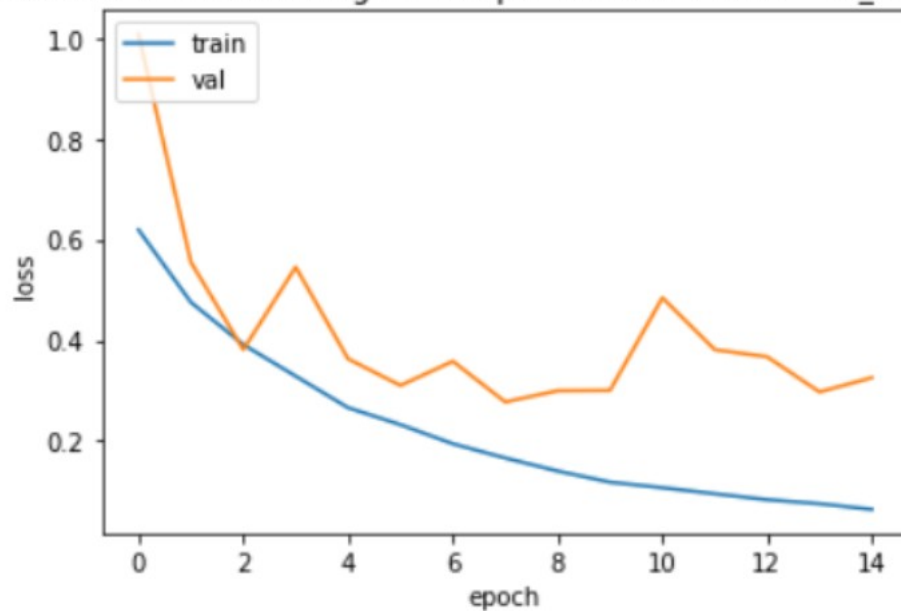
### 6 RESULT



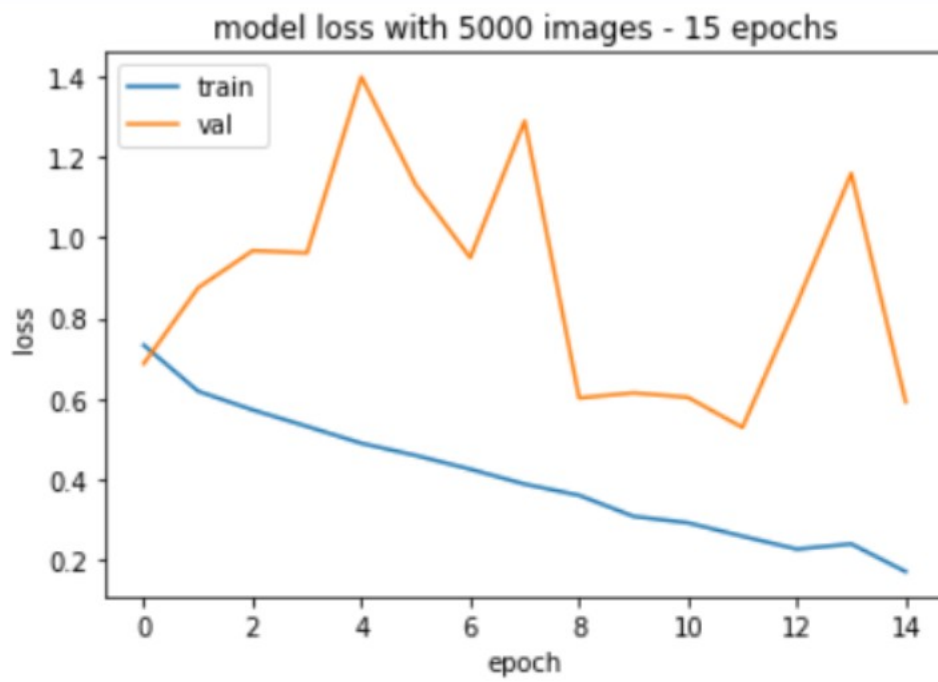
## IMAGE CLASSIFICATION USING CNN's



model loss with 18000 images - 15 epochs - loss: 0.0638 - val\_loss: 0.3255



## IMAGE CLASSIFICATION USING CNN's



## 7 CONCLUSION

The implementation of the image classification model using Convolutional Neural Networks (CNNs) demonstrates the effectiveness of deep learning techniques in recognizing and categorizing visual data. Through systematic steps that include dataset preprocessing, CNN architecture design, model training, and evaluation, the project successfully achieves reliable classification performance. The model is able to automatically learn meaningful patterns such as edges, textures, and shapes from images, eliminating the need for manual feature engineering.

The results show that CNN-based models can generalize well when provided with sufficient training data and appropriate preprocessing strategies. Additional techniques, such as data augmentation and Grad-CAM visualizations, further enhance the model's robustness and interpretability. Overall, this project highlights the strong potential of CNNs for image-based applications and provides a solid foundation for extending the system to more complex tasks such as object detection, real-time classification, and domain-specific image analysis.

## 8 REFERENCES

- [1] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems (NeurIPS)*.
- [2] Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations (ICLR)*.
- [3] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- [4] Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- [5] Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-CAM: Visual explanations from deep networks via gradient-based localization. *IEEE International Conference on Computer Vision (ICCV)*.
- [6] Tan, M., & Le, Q. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. *Proceedings of the 36th International Conference on Machine Learning (ICML)*.
- [7] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., et al. (2017). MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.
- [8] Kaggle. (2024). Cervical Cancer Largest Dataset (SIPaKMeD). Available at: <https://www.kaggle.com>.
- [9] TensorFlow Developers. (2015–2024). TensorFlow: Large-scale machine learning framework used for deep learning implementation. <https://www.tensorflow.org>.
- [10] OpenCV Developers. (2000–2024). OpenCV Library for image processing and computer vision applications. <https://opencv.org>.