# **CHAPTER** **THREE**

# **SYSTEM ANALYSIS AND DESIGN**

# **INTRODUCTION**

This research explores a supervised machine learning classification problem (image classification) using the Convolutional Neural Network deep learning algorithm. This chapter discusses some approaches to solving this problem then elucidates the various tools and implementation methods employed in order to solve this problem using the selected approach.

# **RESEARCH DESIGN**

# **MODES OF APPROACH**

There two major approaches to image classification are the usage of traditional image processing, and the use of deep learning models.

Traditional image processing encompasses steps including image acquisition, preprocessing, enhancement, restoration, color processing, segmentation, feature extraction, compression, analysis, and recognition. These steps involve manipulating pixel values and their relationships within the image, employing mathematical operations, filters, and algorithms to extract information and make sense of, or classify the image content.

There are various deep learning methods used in image classification and some of them include:

1. **Convolutional Neural Networks (CNNs):** CNNs are the primary and most widely used deep learning architecture for image classification tasks. They develop and adapt their own image filters and they consist of a set of layers that enable them perform image classification tasks.
2. **Transfer Learning:** This method involves taking a pre-trained CNN model such as Visual Geometry Group (VGG), Residual Network (ResNet), and Inception. Then fine-tuning it on a specific image classification task. Transfer learning is particularly useful when there is a limited amount of labeled data for a particular classification task.
3. **Residual Networks (ResNets):** ResNets introduce skip connections to alleviate the vanishing gradient problem in deep networks.
4. **Inception Networks:** Inception networks, particularly versions like InceptionV3 and InceptionV4, use a combination of multiple filters and pooling operations at different scales.
5. **Densely Connected Convolutional Networks (DenseNet):** DenseNet connects each layer to every other layer in a feed-forward fashion. It encourages feature reuse and performs vey accurately in image classification tasks.

Below are some of the most accurate CNN architectures alongside their accuracy score on the ImageNet dataset

|  |  |
| --- | --- |
| METHOD/ARCHITECTURE | ACCURACY |
| Residual Networks (ResNet-152) | 96.4% |
| Vision Transformer (ViT-L/14) | 90.2% |
| Dense Network (DenseNet-201) | 84.4% |
| EfficientNet-B7 | 84.4% |

# **RESEARCH METHODOLOGY**

This research employs quantitative methods i.e. it aims to measure and quantify phenomena using numerical data which in this case refers to image classification with the numerical data being the accuracy of the employed model. The best model for this task is the Convolutional Neural Network and the reason for this has been fleshed out in the previous sub-section. An illustration of the CNN architecture is provided below.



Fig. 1. Architecture of a Convolutional Neural Network

The algorithm architecture for the system development is given as follows

* Target images to be used for the development of the model would be extracted from the CIFAKE dataset which contains 60,000 synthetically-generated images and 60,000 real images (collected from CIFAR-10) with each image being 32x32 pixels. The dataset contains two classes - REAL and FAKE. For REAL, images were collected from Krizhevsky & Hinton's [CIFAR-10 dataset](https://www.cs.toronto.edu/~kriz/cifar.html) and for the FAKE images, the equivalent of CIFAR-10 with Stable Diffusion version 1.4 was generated. The dataset contains 100,000 images for training (50,000 per class) and 20,000 for testing (10,000 per class).
* The convolutional layer or layers is then constructed. In order to extract essential characteristics from the input images, the convolutional layer computes the convolutional operation of the images using matrix filters. In comparison to the input images, the filters have the same dimensions but reduced constant parameters. A 2D activation map is created by sliding the filter across the whole image input step by step and estimating the dot product between the weights of the filters and the value of the input image.
* Construction of an activation function. The activation function determines the output from an input fed to a neuron and the set of inputs fed to the layer as a whole by determining whether or not a neuron’s input to the network is important. It thereby removes unnecessary values such as unwanted pixels. The Rectified Linear Unit (ReLU) activation function would be used to the non-linear form of images.
* Reduction of the size of the array using max pooling algorithm. In max pooling, the output value for each pooling region is simply the maximum value of the input values within that region. This has the effect of preserving the most salient features in each pooling region, while discarding less relevant information. The result is a pooled feature map which then undergoes a process called “flattening” to convert the data into a single column so as to conform with neural network input.
* The classification algorithm is implemented and a mathematical tool called the SoftMax function is then constructed to transform a vector of real values into a probability distribution. Each element is exponentiated to make it positive, and then it divides the result by the total of all the exponentiated values to normalize it. This helps the output appear in a probabilistic form.
* Once the training of the neural network is completed, we test it using the test data and the accuracy of the network is determined.

# **FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS**

The functional requirements of the system are:

* A user should be able to securely upload an image via a form
* The system should provide feedback on the authenticity of the uploaded image
* The system should be able to accept images of various dimensions

The non-functional requirements or quality attributes of the system are as follows:

* The web interface provided should be user-friendly and easy to use
* The web interface should be responsive and accessible (hosted) online
* The system should protect user data from unauthorized access

# **SYSTEM REQUIREMENTS**

# **3.4.1** **OPERATING SYSTEM REQUIREMENTS**

* Ubuntu 16.04 or higher (64-bit)
* macOS 10.12.6 (Sierra) or higher (64-bit) *(no GPU support for TensorFlow)*
* Windows Native - Windows 7 or higher (64-bit) *(no GPU support after TensorFlow 2.10)*
* Windows WSL2 - Windows 10 19044 or higher (64-bit)

# **HARDWARE REQUIREMENTS**

* AVX Instructions enabled CPUs
* NVIDIA® GPU card with CUDA® architectures 3.5, 5.0, 6.0, 7.0, 7.5, 8.0 and higher. *(for GPU build)*

# **SOFTWARE REQUIREMENTS**

* Python 3.9–3.11
* Pip version 19.0 or higher for Linux (requires manylinux2014 support) and Windows. pip version 20.3 or higher for macOS.
* Windows Native Requires Microsoft Visual C++ Redistributable for Visual Studio 2015, 2017 and 2019
* Microsoft Visual Studio Code

# **FLOWCHART AND OTHER DIAGRAMS**

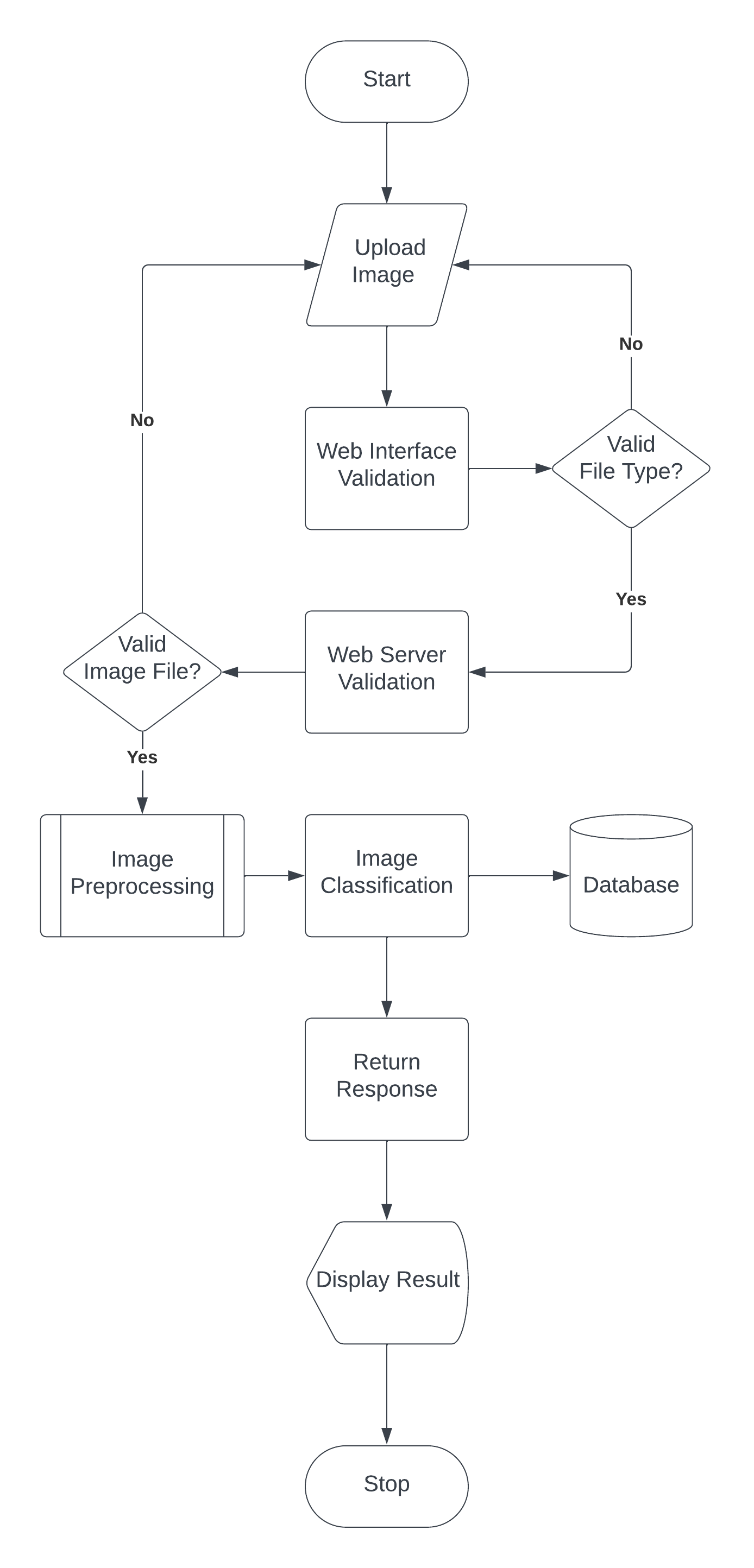


Fig. 2. System Flow Chart

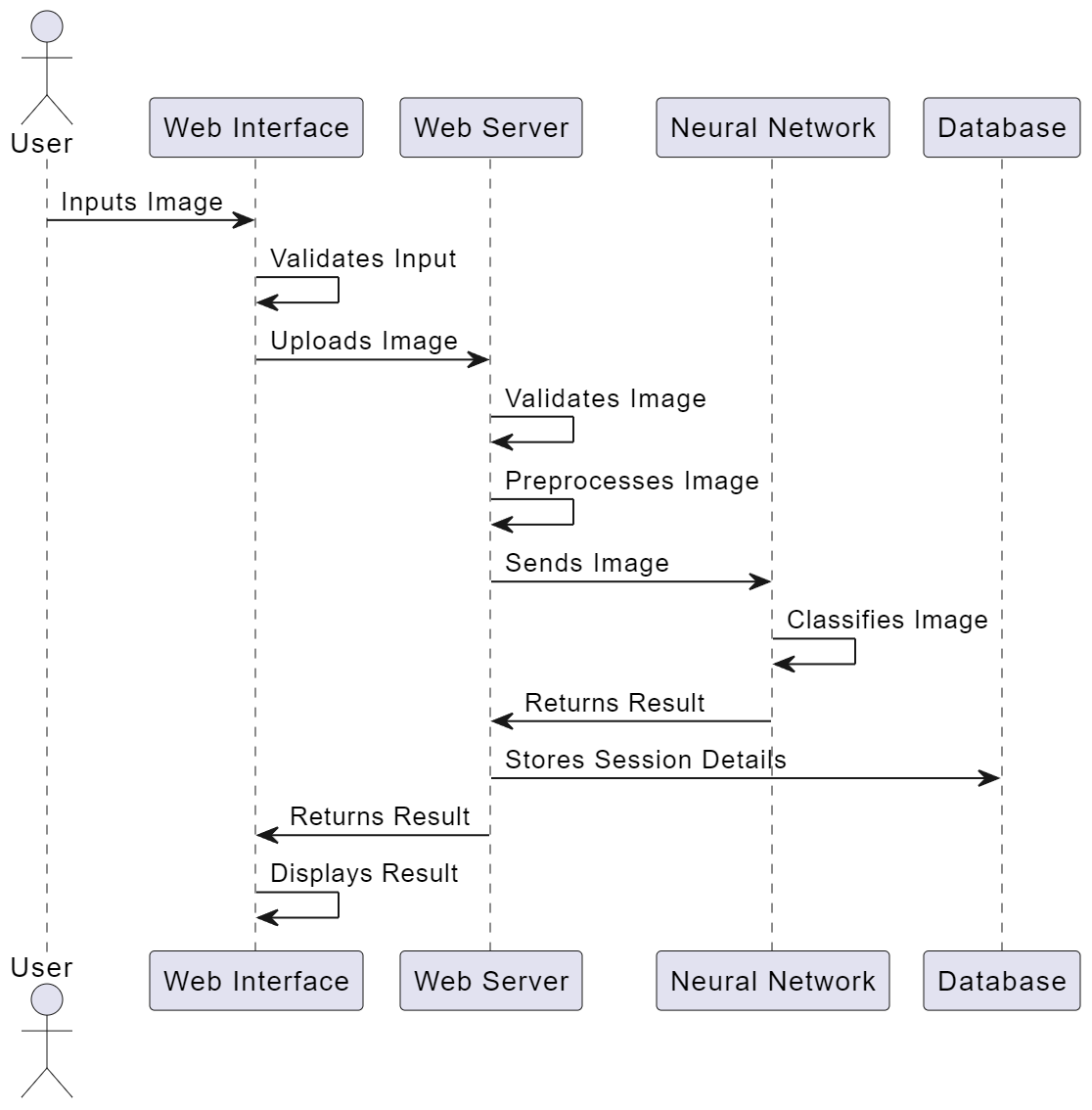


Fig. 3. UML - Sequence Diagram of Fake Image Detection System

# **DEVELOPMENT TOOLS**

This system will be developed using latest technologies such as HTML5, CSS3, JavaScript and the VueJS framework for the client-side/frontend of the web application, and the flask python microframework for the server-side/backend due to its lightweight nature and simplistic integration of machine learning models; these will be programmed on Microsoft’s Visual Studio Code (VS Code). The deep learning model would be implemented using the Keras open-source library that provides a Python interface for artificial neural networks. Keras acts as an interface for the larger Tensorflow library.