1. INTRODUCTION

- Plants play a vital role in our planet's ecosystem, contributing to biodiversity, food production, and environmental health. Accurately identifying plant species is crucial for various applications, including ecological monitoring, conservation efforts, and precision agriculture. Traditionally, plant identification relies on expert knowledge and visual inspection of leaves, flowers, and other botanical features.
- However, this process can be time-consuming, require specialized training, and can be challenging for non-experts or in situations with diverse or unknown plant communities.
- The development of plant identification systems has the potential to revolutionize how we classify and monitor plant populations. Deep learning models offers promising tools for this task, allowing us to analyse plant characteristics and build models that can accurately identify species based on these features.
- This project delves into the exploration o plant species classification using the "One Hundred Plant Species Leaves Data Set" available on the UCI Machine Learning Repository
- https://archive.ics.uci.edu/ml/datasets/Onehundred_plant_species_leaves_data_set.
- This dataset offers a valuable resource for investigating the feasibility of using leaf morphology for plant identification.
- The dataset comprises information on 100 distinct plant species, with 16 leaf samples captured for each. Each sample is accompanied by a set of features extracted from the leaf images, including shape and texture descriptors. This rich dataset provides a foundation for building and evaluating machine learning models for plant species classification.

2. ABSTRACT

- There are half a million species of plant in the world, so the classification and identification of various plant species is one of the important phases. In recent time's computer vision have been successfully applied towards automated systems of plant cataloguing.
- Manual identification requires prior knowledge of species and is a lengthy process, thus the atomization technique helps to speed up the traditional method of plant leaf identification. Plant systematics can be classified and recognized based on their leaves.
- The proposed idea tries to bring an atomization in the process to develop a model which would detect the plant leaf species with the digital image of plant leaf. Convolutional Neural networks is one of the most popular deep learning algorithms for plant leaf classification. The Neural Network would be trained for detection of edge (Shape-based Classification).
- For training, a large number of training samples were produced by UCI Repository. In the experiment, through training and testing on the CNN Techniques under the toolbox of deep learning, the leaf classification finally got the correct rate of ~85% after the optimization of hyper parameter during training.

Through improving the accuracy of image classification and image recognition, the applications of CNN provide a reference value for the field of image processing in botany. This will help the botanists in their study and speed up the process of identifying the species of plant.

3. OBJECTIVE

The main objective of the project is to create a deep learning techniques which is capable of the categorization of plant species using characteristics collected from leaf photos. This ambitious project is motivated by the need to solve the issues provided by the global diversity of plant species and the labour- intensive nature of manual species identification.

1. Image Classification: The project's primary goal is to automate the process of classifying plant species using CNN. The aim is to create a strong classification system capable of properly recognizing plant species by utilizing information derived from leaf photos, such as shape, descriptors ,margin properties.

2. Enhancing Efficiency: Manual species identification from leaf photographs is a time-consuming and labour-intensive job that necessitates knowledge and substantial resources.

3. Improving Accuracy: Classification of images (Leaves) using deep learning models has the potential to increase species identification accuracy over manual techniques. By examining a wide variety of data collected from leaf photos, the pipeline may detect minor changes between plant species that may be invisible to the naked eye.

4. DATASET INFORMATION

The dataset available through the provided link is the "Plant Species Classification Leaves Data Set" from the UCI Machine Learning Repository. This collection includes photos of leaves from 100 plant species, with sixteen samples per species. Each leaf sample has three types of features:

Number of Instances: The dataset contains 1600 instances, with 16 samples for each of the 100 plant species.

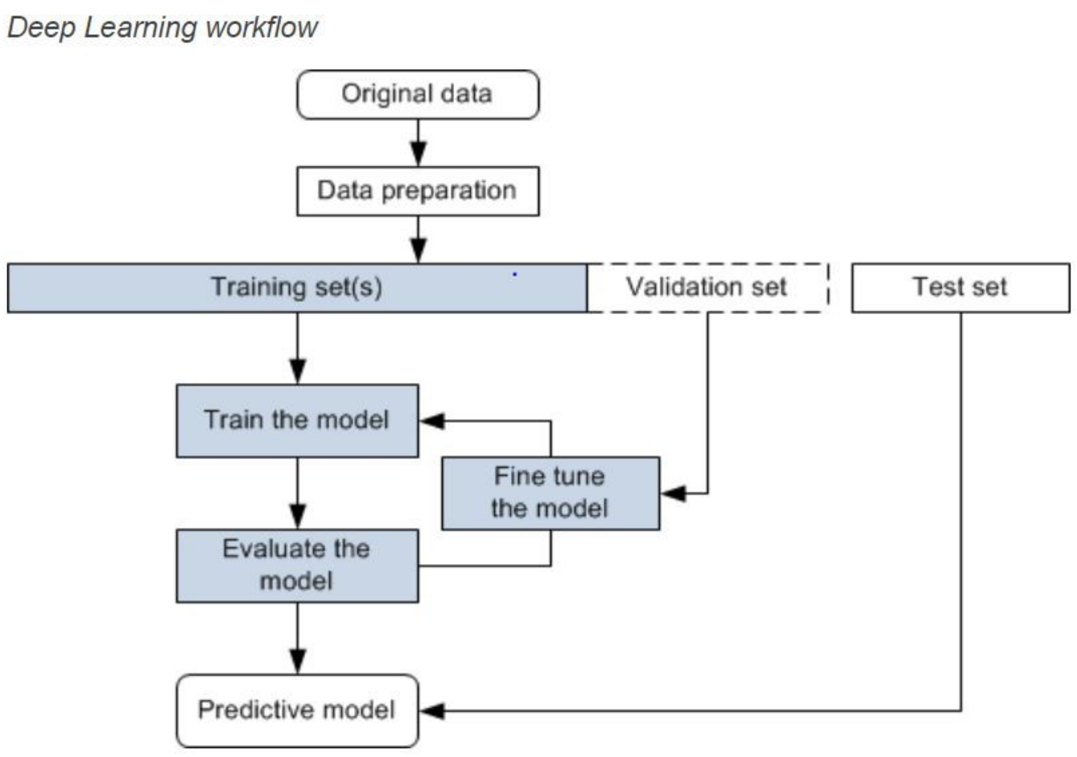
Data Format: The dataset is divided into three files, each holding the feature vectors for a certain collection of characteristics (shape, margin, and texture descriptors). Each entry in the file corresponds to a leaf sample, with the class label (plant species) followed by a 64-element feature vector.

Dataset Characteristics	**Multivariate**
Subject Area	Biology
Associated Tasks	Classification
Feature Type	Real
Number of Instances	1600
Number of Features	64

Link to Dataset:

https://archive.ics.uci.edu/dataset/241/one+hundred+plant+species+leaves+data+set

5. WORK-FLOW

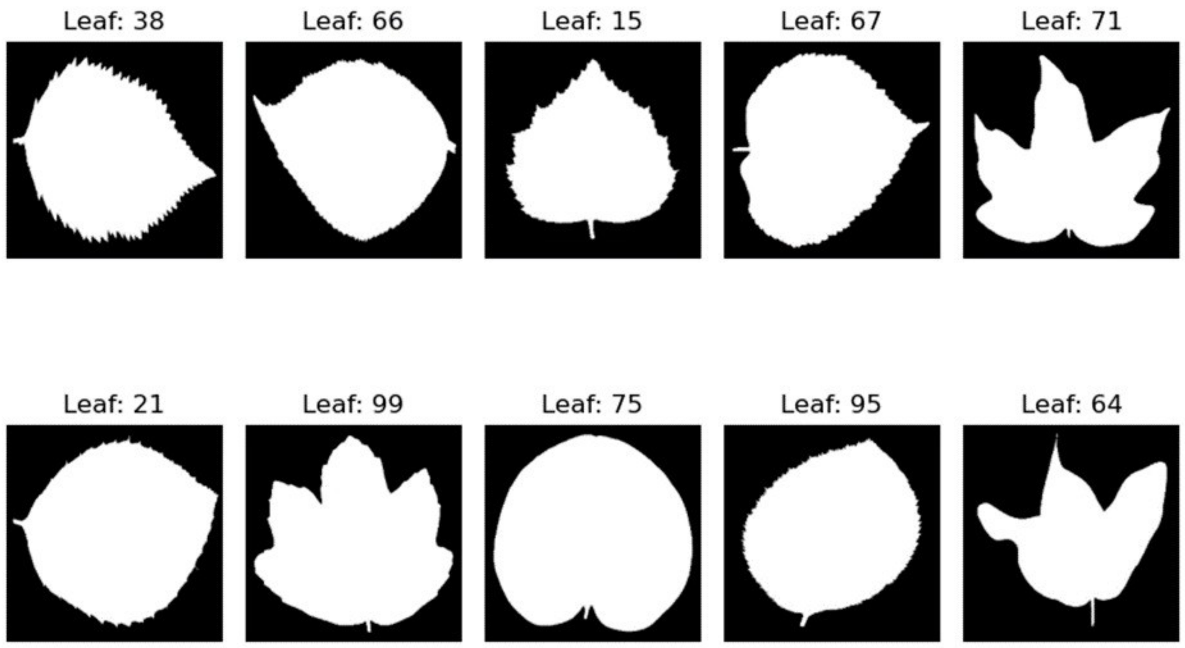


6. EXPLORATORY DATA ANALYSIS

A strategy to evaluating datasets to highlight their key features is called exploratory data analysis (EDA), and it frequently makes use of visual aids. Understanding the data, finding patterns, spotting abnormalities, and testing hypotheses are its main objectives. Before using more formal statistical approaches, EDA is usually carried out at the start of a data analysis project to obtain insights into the underlying structure of the data. EDA involves techniques such as summary statistics, histograms, box plots, scatter plots, and correlation matrices to visually and numerically summarize the dataset's main features. It may also involve data cleaning and preprocessing steps to address missing values, outliers, or inconsistencies in the data.

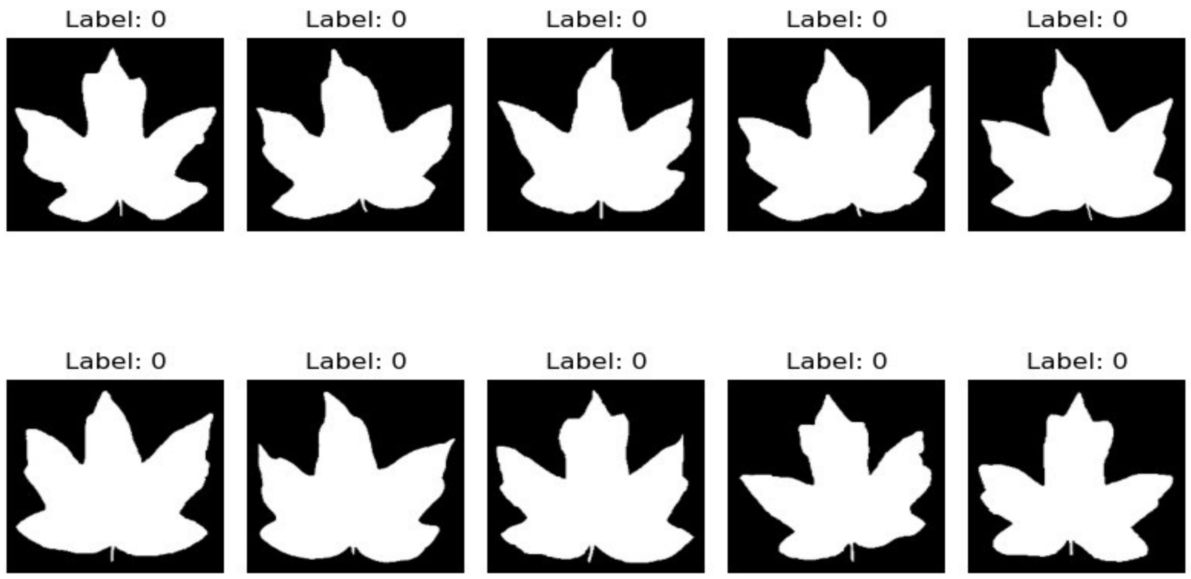
Analysts and data scientists can benefit from EDA by:
1. Examine the data for trends and patterns.
2. Recognize the correlations and distributions of the variables.
3. Find anomalies and outliers.
4. Create theories for additional investigation.
5. Choose which modelling techniques are appropriate to use.

"Data Collection: Gathering Image Paths for Leaf Categories"



The above images are from the dataset used for leaf classification or botanical. Here's a breakdown of what we have observed.

- Leaf Silhouettes: Each image shows a white silhouette of a leaf on a black background. The silhouettes are useful for analysing the shape and structure of the leaves, which are important characteristics for identifying species.
- Numbers: Each leaf silhouette is labelled with a number, such as "Leaf: 38", "Leaf: 66", and so on. These numbers refer classification index and identifiers that link to a specific species.
- Variety of Shapes: The leaves shown have various shapes, margins, and tip styles. For example, some leaves have smooth edges, while others are serrated or lobed. The diversity of leaf shapes suggests a broad representation of species.

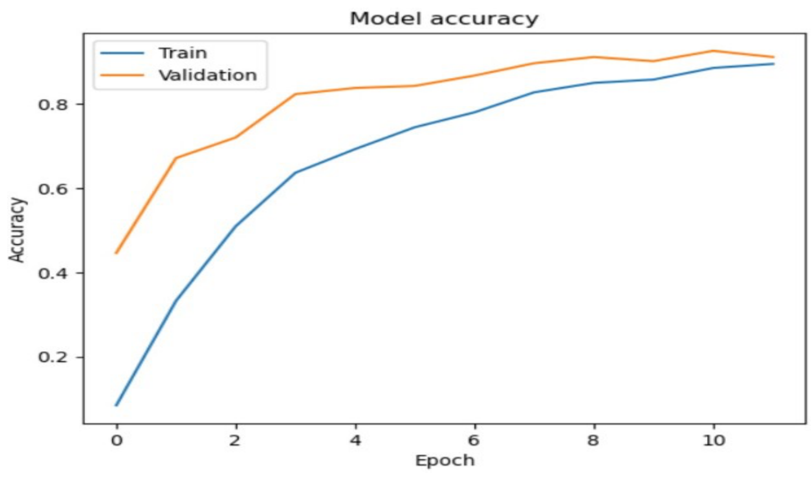


The image provided appears is collection of leaf silhouettes, all uniformly label as "Label: 0."

Dataset Overview:

- **Uniformity in Labelling:** Each leaf silhouette carries the same label, indicating they belong to a singular class or category within the dataset.
- **Purpose of Uniformity:** The identical labelling states that these images are used to train a model to recognize features specific to a certain class, or to evaluate the model's ability to consistently identify similar patterns.
- **Shape and Morphology:** The leaf silhouettes display a complex shape with multiple lobes, a structure that is indicative of a particular species.
- **Consistency in Imagery:** All leaf images are different in shape and size, but having same labels providing a consistent set for analysis.

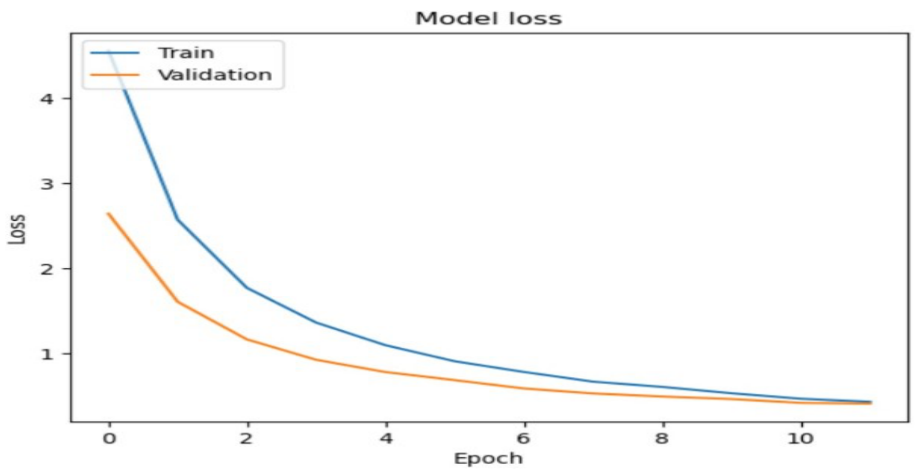
Analysis of Deep Learning Model Training Progress :



The graph presents the evolution of accuracy for deep learning model during its training and validation phases across epochs.

Graph Overview:

- X-Axis (Epochs): The x-axis represents the number of epochs, which are complete passes through the entire training dataset. Here, the graph extends from 0 to 10 epochs.
- Y-Axis (Accuracy): The y-axis measures accuracy, ranging from 0 to 1, where 1 signifies perfect accuracy.
- Analysis of Training Accuracy:
- Training Accuracy (Blue Line): This line shows the model's performance on the training dataset. It starts at around 0.5 accuracy and improves steadily, indicating the model is learning from the training data.
- Trends and Patterns: The smooth upward trajectory suggests that the model is successfully capturing the underlying patterns in the training data without signs of erratic learning or instability.
- Validation Accuracy (Orange Line): This line shows how the model performs on a separate set of data not seen during training. It's crucial for assessing the model's generalization capability.



Graph Explanation:

- X-Axis (Epochs): The x-axis represents epochs, which are iterations over the entire dataset during the training process. The graph shows data for 10 epochs.
- Y-Axis (Loss): The y-axis measures the model's loss, a quantification of the error between the model's predictions and the actual data. Lower loss values indicate better model performance, with 0 being the ideal (no error).
- initial High Loss: At epoch 0, the training loss is highest.
- Rapid Decrease: The training loss decreases sharply between epochs 0 and 2, indicating that the model is quickly learning from the training data.
- Gradual Decrease: As epochs increase, the decrease in training loss becomes more gradual, showing the model's continued but slower improvement.
- Validation Loss (Orange Line):
- Initial Decrease: Similar to the training loss, the validation loss starts high and drops quickly, which means the model is generalizing well initially to unseen data. Convergence Toward Training Loss: The validation loss tends to converge towards the training loss, but doesn't decrease as rapidly after around epoch 4.

7. MODELS USED FOR TRAINING THE DATA

7.1 Convolutional Neural Network(CNN)

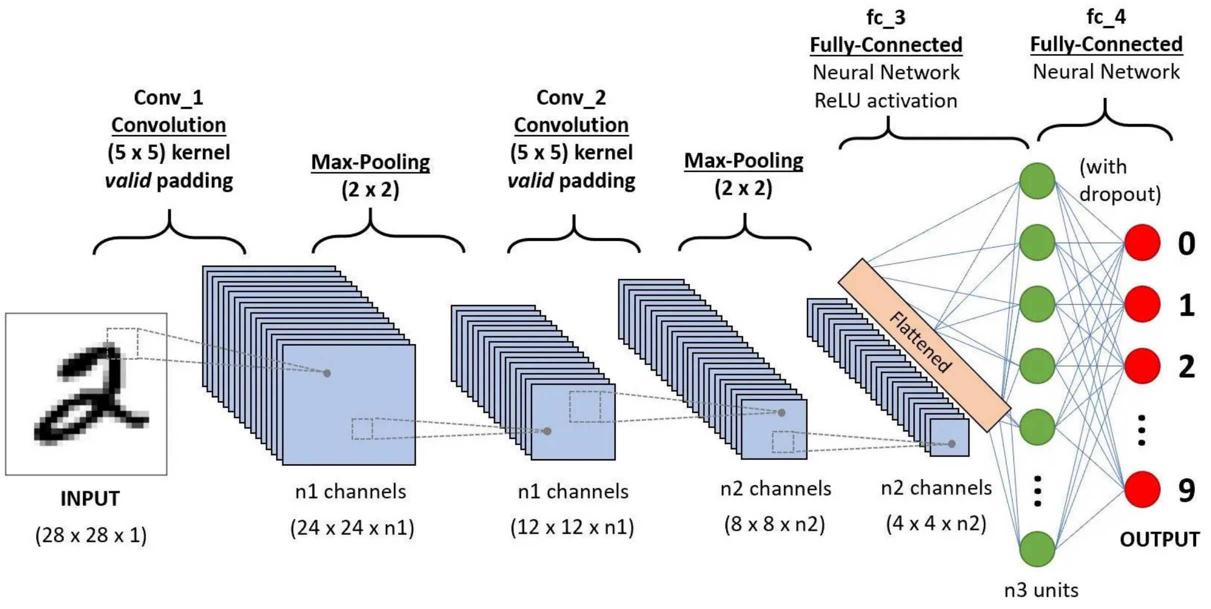
What is CNN?

Convolutional Neural Network is what CNN stands for. This particular kind of deep neural network is useful for tasks like text and time series analysis as well as image recognition and classification.

This is a brief overview of CNNs:

- Convolutional Layers: Convolutional layers are the fundamental components of CNNs. These layers subject input data to a collection of learnable filters, often known as kernels. Certain features, including edges, textures, or patterns, are extracted from the input by each filter. In order to create feature maps, the filters slide (convolve) over the input data and multiply and aggregate each element individually.

- Pooling Layers: To minimize the spatial dimensions of the feature maps without sacrificing significant information, pooling layers are frequently applied after convolutional layers. Choosing the maximum or average value inside a window allows feature maps to be down sampled using popular pooling techniques like max pooling and average pooling.



- Activation Functions: To provide non-linearity to the network and help it learn intricate patterns, non-linear activation functions like as Rectified Linear Unit, or ReLU, are usually applied after convolutional and pooling layers.
- Fully Connected Layers: The high-level features are flattened and processed through one or more fully connected (dense) layers following a number of convolutional and pooling layers. To generate predictions, these layers blend the extracted characteristics. The ultimate dense layer in classification tasks typically features a SoftMax activation function to probability for the output class.
- Training: In order to minimize a loss function, CNNs are trained using optimization methods like gradient descent and backpropagation. Both categorical cross-entropy and sparse categorical cross-entropy are frequently used loss functions in classification applications.

In a variety of tasks, including semantic segmentation, object detection, image classification, and more, CNNs have shown impressive results. They are very good at capturing subtle structures in complex datasets because of their capacity to automatically learn hierarchical patterns from raw data.

Convolutional Neural Networks (CNNs) are indeed extensively utilized in pretrained models for various computer vision tasks. Here's a breakdown of how CNNs are employed in pretrained models:

1. Feature Extraction: CNNs are adept at learning hierarchical representations of visual features from raw image data. In pretrained models, CNNs are typically used as feature extractors. These models are initially trained on large-scale datasets like ImageNet to recognize basic visual patterns such as edges, textures, and shapes. The learned features are then leveraged for various downstream tasks.

2. **Backbone Architecture:** CNNs serve as the backbone architecture of pretrained models. They form the core of the model, responsible for processing input images and extracting meaningful features. These architectures often consist of multiple convolutional layers followed by activation functions (such as ReLU) and pooling layers.

3. **Transfer Learning:** One of the significant advantages of pretrained CNNs is their ability to facilitate transfer learning. Instead of training a CNN from scratch on a new dataset, pretrained models allow us to transfer knowledge learned from a source task (e.g., ImageNet classification) to a target task with potentially fewer labelled examples. By fine-tuning or feature extraction, pretrained CNNs can adapt to new tasks more efficiently.

4. **Architectural Variants:** There is a diverse range of CNN architectures used in pretrained models, each with its own strengths and characteristics. Examples include ResNet, Inception, Mobile Net, among others. These architectures vary in terms of depth, width, skip connections, and other design choices, catering to different requirements and constraints.

5. **Model Zoo:** Many pretrained CNN models are readily available in model zoos provided by deep learning frameworks like TensorFlow, PyTorch, and Keras. These model zoos offer a collection of pretrained models trained on benchmark datasets, enabling researchers and practitioners to easily access and utilize state-of-the-art architectures for their tasks.

Model Architectures

The architecture of two neural network models created for a 100-class classification challenge is covered in this research. In the first model, the hyperparameters are fixed and the architecture is predefined; in the second model, performance is optimized through hyperparameter tuning with Keras Tuner.

7.1.1 Model 1: Predefined MobileNetV1

MobileNetV1 is a convolutional neural network architecture designed for efficient mobile and embedded vision applications. The primary motivation behind MobileNetV1 was to create a model that could achieve high accuracy on image classification tasks while being lightweight and computationally efficient, making it suitable for deployment on resource-constrained devices such as smartphones, tablets, and embedded systems.

Key features of MobileNetV1 include:

- Depth wise Separable Convolutions: MobileNetV1 extensively uses depth wise separable convolutions, which split the standard convolutional operation into two separate layers: depth wise convolution and pointwise convolution. This separation significantly reduces the number of parameters and computations compared to traditional convolutional layers, thus making the model more efficient.
- Width Multiplier and Resolution Multiplier: MobileNetV1 introduces two hyperparameters known as the width multiplier and resolution multiplier. The width multiplier controls the number of channels in each layer, allowing the user to trade-off

between model size and accuracy. The resolution multiplier adjusts the input image resolution, enabling further reduction in computational cost.

- Architecture: MobileNetV1 consists of a series of depth wise separable convolutional layers followed by 1x1 convolutional layers and global average pooling. The final layers typically include a fully connected layer and a SoftMax layer for classification.
- Pre-trained Models: Pre-trained versions of MobileNetV1 are available, trained on large-scale image datasets such as ImageNet. These pre-trained models can be fine-tuned on specific tasks or used as feature extractors for transfer learning.
- MobileNetV1 has been widely adopted in various real-world applications where computational efficiency is crucial, such as image classification, object detection, semantic segmentation, and more. Its success has paved the way for subsequent versions of Mobile Net, each aiming to improve performance while maintaining efficiency.

7.1.2 Pretrained MobileNetV2

The another model used in the project is known as MobileNetV2. Google researchers built the convolutional neural network (CNN) architecture for fast deep learning on mobile and embedded devices. MobileNetV2 expands on the original Mobile Net design, seeking to increase the efficiency and performance of deep neural networks.

Key features of MobileNetV2:

1. **Depth wise separable Convolutions:** MobileNetV2 heavily uses depth wise separable convolutions. Unlike classical convolutions, which work on all input channels simultaneously, depth wise separable convolutions divide the operation into two stages: depth wise convolutions and pointwise convolutions. This decreases processing costs while collecting geographical and channel-specific information effectively.

2. **Inverted Residuals with Linear Bottlenecks:** MobileNetV2 uses inverted residuals and linear bottlenecks to increase feature expression with fewer parameters. This enables for more Making efficient use of computer resources by enlarging the bottleneck layer and then projecting it back to a lower-dimensional space.

3. **Linear bottleneck:** MobileNetV2 uses linear activations in the bottleneck layers rather than nonlinear activations. This reduces the bottleneck layer while preserving expressive power, allowing for rapid feature extraction.

4. **Expansion Layer:** MobileNetV2 uses an expansion layer before depthwise convolution to increase the number of channels. This extension stage allows the network to capture more detailed feature representations.

5. **Improved Shortcut Connections:** Shortcut connections, also known as skip connections, are provided between bottleneck layers to improve gradient flow and help train deeper networks.

Advantages of MobileNetV2:

Effectiveness: MobileNetV2 is meant to be lightweight and efficient, making it appropriate for deployment on resource-constrained devices such as smartphones, tablets, and embedded devices. High Performance: Despite its efficiency, MobileNetV2 performs well on a variety of computer vision tasks, such as picture classification, object identification, and semantic segmentation. Flexibility: MobileNetV2 may be fine-tuned or tailored for individual tasks and datasets, making it suitable for a variety of mobile and embedded vision applications. Overall, MobileNetV2 strikes a good mix between efficiency and performance, making it a popular choice for real-world applications that need deep learning models to operate quickly on smartphones with limited computing resources.

7.1.3 InceptionV3

Inception-v3 is a convolutional neural network (CNN) architecture designed for image classification and recognition tasks. It is part of the Inception family of models developed by Google Research as part of the Inception project, which aims to improve the efficiency and accuracy of deep learning models for computer vision tasks. Here's a detailed explanation of Inception-v3:

1. Architecture: Inception-v3 is a deep neural network architecture composed of multiple layers. It follows the general design principles of CNNs, with convolutional layers for feature extraction and fully connected layers for classification. However, what sets Inception-v3 apart is its unique structure, which includes various types of convolutional layers and advanced architectural features.

2. Inception Modules: The key innovation introduced by the Inception-v3 architecture is the use of "Inception modules." These modules are designed to capture features at multiple spatial scales efficiently. Instead of using a single convolutional layer with a fixed filter size, Inception modules employ a combination of convolutional layers with different filter sizes (1x1, 3x3, 5x5) and pooling operations. By using multiple paths of varying receptive fields, the network can effectively capture both local and global features from the input image.

3. Factorization: Inception-v3 also employs a technique called "factorization" to reduce the computational complexity of the network. This involves decomposing large convolutional filters into smaller ones, which reduces the number of parameters and computations required. For example, instead of using a single 5x5 convolutional filter, the network might use two consecutive 3x3 filters, which results in fewer parameters and allows for more efficient training and inference.

4. Auxiliary Classifiers: Inception-v3 includes auxiliary classifiers at intermediate layers of the network. These classifiers are added to the network to mitigate the vanishing gradient problem during training and provide additional regularization. The outputs of these auxiliary classifiers are combined with the final classification scores during training to improve the overall performance of the model.

5. Applications: Inception-v3 and its variants have been widely used in various computer vision applications, including image classification, object detection, and image segmentation. The model's efficient architecture and high accuracy make it well-suited for tasks where both performance and computational resources are important considerations. We may assess how well hyperparameter adjustment improves the model's prediction power by contrasting the two models' performances.

7.1.4 ResNet50

ResNet, or Residual Network, is a groundbreaking deep learning architecture designed to tackle the challenges of training very deep neural networks. It was introduced in a paper titled "Deep Residual Learning for Image Recognition" in 2015. The main innovation of ResNet is the use of residual blocks, which allow the network to learn residual functions instead of directly learning the underlying mapping. This approach mitigates the degradation problem that occurs when adding more layers to a neural network, improving training convergence and performance. ResNet has become widely adopted due to its remarkable accuracy and efficiency in various computer vision tasks, including image classification, object detection, and segmentation. Its variants, such as ResNet-50 and ResNet-101, are commonly used in both research and practical applications.

Transfer Learning?

Transfer learning is a powerful technique in machine learning where knowledge gained while training a model on one task (source task) is applied to a different but related task (target task). It's like leveraging the knowledge you learned in one class to help you understand a similar subject in another class.

Here's a breakdown of the concept:

Why Transfer Learning?

- Training a machine learning model, especially deep learning models with many parameters, can be computationally expensive and require a massive amount of data. Transfer learning allows you to:
- Reduce Training Time: By using a pre-trained model as a starting point, you can significantly reduce the training time required for the target task. The pre-trained model has already learned low-level features like edges and shapes, which are often generic and transferable across different tasks.
- Improve Performance: In many cases, using a pre-trained model can lead to better performance on the target task compared to training a model from scratch, especially when the amount of data available for the target task is limited.

How Does it Work?

There are two common approaches to transfer learning:

Freeze-and-Fine-tune:

Freeze the weights of the pre-trained model on the source task. These weights represent the low-level features the model learned. Add a new layer (or few layers) on top of the pre-trained model. This new layer is trained specifically for the target task. By fine-tuning only the final layers, you adapt the pre-trained model's knowledge to the specific requirements of your new task.

Full Fine-tuning:

Unfreeze all the weights of the pre-trained model and train them along with the newly added layers for the target task. This approach is useful when the source and target tasks are more distantly related, requiring the model to adapt more significantly.

8. TRAINING AND TESTING DATA

Training and testing data are critical components in the design and assessment of machine learning models, such as neural networks. Here's what they symbolize and why they're used:

1. TrainedData:
- Training data is a subset of the dataset used to train the machine learning model. It is made up of input characteristics (such as photos, text, and numerical data) and their associated target labels.
- During the training phase, the model discovers patterns and correlations in the training data before making predictions or classifications.

2. Testing Data:

- Testing data, also known as validation or evaluation data, is a different piece of the dataset that is not used during model training.

- Once trained, the model is assessed against testing data to determine its performance and generalization capabilities. The testing data helps predict how well the model will perform on previously unknown data.

- Testing data offers an unbiased assessment of the model's performance by mimicking real-world scenarios in which the model encounters previously unknown data. It helps identify overfitting (when the model performs well on training data but badly on unseen data) and advises hyperparameter selection.

In this training phase we uses TensorFlow's `image dataset from directory` function to create training and validation datasets from a directory containing image files. The `data Dir` parameter specifies the directory path where the image files are located. The `validation split` parameter splits the dataset into training and validation subsets, with 80% of the data used for training and 20% for validation. The `image size` parameter sets the size of the images to 224x224 pixels, and the `batch size` parameter specifies the number of images to include in each batch during training and validation. After creating the validation dataset (`val_ds`), the code determines the number of batches in the validation dataset (`Val batches`) and splits it into two equal parts to create the test dataset (`test_ds`). The output indicates that 1600 files belonging to 100 classes were found, with 1280 files used for training and 320 files used for validation.

9. TUNING HYPER PARAMETERS

9.1 What is Keras Hyperband?

- Keras Hyperband is a specific implementation of the Hyperband algorithm, a technique for efficient hyperparameter search. Here's what makes it special:
- Early Stopping: It leverages early stopping to stop training trials that are unlikely to perform well early on. This saves computational resources by focusing on promising configurations.
- Successive Halving: It iteratively reduces the resource allocation (e.g., training time) for trials in subsequent rounds. This prioritizes exploration of diverse hyperparameter combinations initially and then refines the search in later rounds.
- Adaptivity: It dynamically adjusts resource allocation based on the performance of trials, focusing on configurations with the best potential.

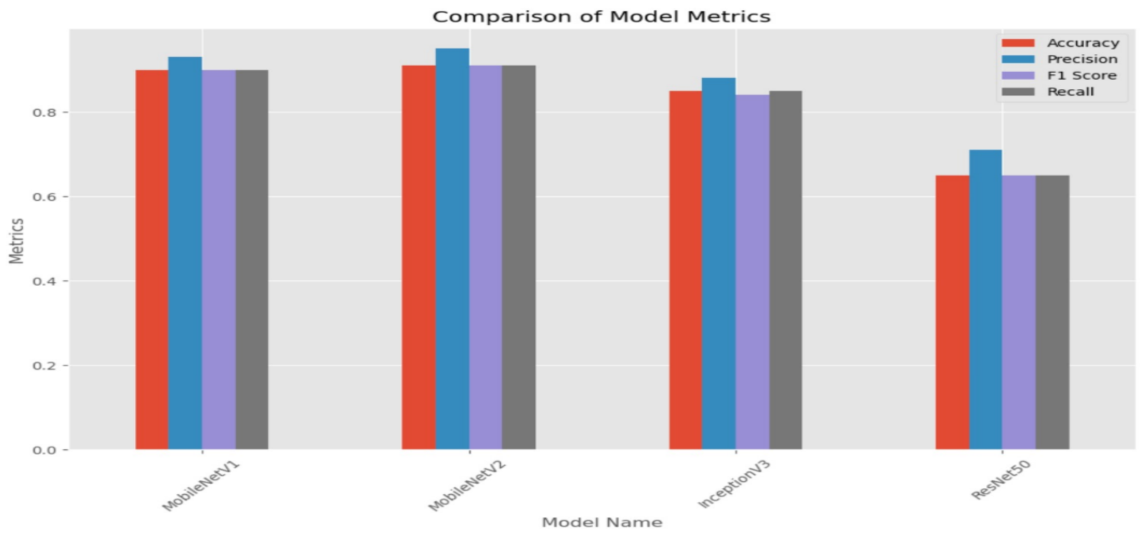
Benefits of Keras Hyperband:

- Efficiency: It can significantly reduce the time and resources required for hyperparameter tuning compared to exhaustive search methods.
- Versatility: It can be used with various machine learning models and can handle a wide range of hyperparameters.
- Ease of Use: Keras Tuner integrates Hyperband seamlessly, allowing you to define your hyperparameter search space and use it with minimal code.

10. RESULTS AND CONCLUSION:

Performance metrics of various models:

Model name	**Accuracy**	**Precision**	**F1 Score**	**Recall**
MobileNetV1	0.90	0.93	0.90	0.90
MobileNetV2	0.91	0.95	0.91	0.91
InceptionV3	0.85	0.88	0.84	0.85
ResNet50	0.65	0.71	0.65	0.65



The bar graph presents a comparison of four different machine learning models — MobileNetV1, MobileNetV2, InceptionV3, and ResNet50 — across four key performance metrics: Accuracy, Precision, F1 Score, and Recall. Here's a detailed summary:

- MobileNetV1: This model exhibits high levels of performance across all metrics, with an Accuracy and Recall both at 0.90, and the Precision and F1 Score very close at 0.93 and 0.90, respectively.
- MobileNetV2: This model slightly outperforms MobileNetV1, achieving the highest metrics among the four models. It has an Accuracy of 0.91 and a Precision of 0.95, which is the highest Precision value. The F1 Score and Recall both match the Accuracy at 0.91.
- InceptionV3: This model shows moderately high performance but lags behind the Mobile Net models. It has an Accuracy of 0.85, Precision at 0.88, an F1 Score of 0.84, and a Recall of 0.85.
- ResNet50: This model has significantly lower metrics compared to the other three models. All the metrics for ResNet50 — Accuracy, Precision, F1 Score, and Recall — are at 0.65, which is substantially lower than its counterparts.

From the graph, we can infer that MobileNetV2 is the top-performing model in this summary, followed closely by MobileNetV1, with InceptionV3 in the middle range, and ResNet50 falling behind significantly in all measured metrics. This visualization makes it easy to compare the models quickly and to identify which models excel or underperform in certain areas.