IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

CogniCapture: Visual Insight Engine

[1]Vaishnavi Jaywant Kurunde, [2]Nitish Kumar

[1]Undergraduate Student, [2]Assistance Professor

[1] School of Engineering ,

[1]Ajeenkya DY Patil University, Pune, India

Abstract: Deep learning and neural networks have had a dramatic impact on many fields, notably with the widespread use of Convolutional Neural Network (CNN) models. These models have proved indispensable in a wide range of applications, including image recognition and segmentation. The introduction of CNNs has ushered in an era of extraordinary accuracy and efficiency in activities that were previously considered difficult for standard algorithms.

One of the most recent developments in this field is the release of "CogniCapture: Visual Insight Engine," a cutting-edge application that uses CNN models. This utility uses numerous critical components to improve its performance. To efficiently handle visual data, it uses the Max pooling function in conjunction with two crucial activation functions, ReLU and Simoid. Furthermore, to successfully fine-tune the model parameters, the Adam optimizer is used, with categorical_crossentropy as the loss function. Furthermore, the model's cornerstone is its pre-training with the widely regarded EfficientNetB1 architecture, which ensures resilience and adaptability acrossa wide range of datasets. The integration of these components within CogniCapture demonstrates a comprehensive approach to visual data analysis and insight development. CogniCapture improves accuracy and speed in picture classification and segmentation by leveraging the power of CNNs and optimising their design and training procedures. This technology offers a big step forward in using deep learning for practical applications, with transformational capabilities in domains ranging from healthcare and autonomous systems to marketing and security.

Key words: Convolutional Neural Network (CNN), EfficientNet Activation Function, MaxPooling2D, Adam Optimizer, Categorical Crossentropy

I. INTRODUCTION

"CogniCapture" refers to a technology or system designed to gather and analyse visual data in an intellectually sophisticated manner. The prefix "Cogni" represents intelligence, whereas "capture" refers to the collection or acquisition of visual data.Convolutional Neural Networks (CNN) have made amazing advances in computer vision as machine learning and deep learning have evolved in recent years. Deep learning, an emerging field of study, employs the building of neural networks with multiple layers and the use of large amounts of training data to identify primary features, thereby improving classification or prediction accuracy [1]. The increase in popularity is primarily due to the current desire for rapid access to information. Models such as CNN have gained popularity owing to their capacity to effectively modify images. In my research, they focused on image analysis, which involves scanning images and using existing database information to synthesise or anticipate image details. This prediction method is made easier by enhancing the image data. After scanning, the image is converted into a visual representation, which helps users understand the underlying data more efficiently.In our study report, they used a Convolutional Neural Network (CNN) model, also called a sequential model. We used MaxPooling2D and combined two popular activation functions, ReLU and Sigmoid, with the Adam optimizer and category cross-entropy loss function. they also implemented the early stopping callback function. In addition, they assessed the model's performance using classification

metrics such as F1 score, precision, and recall obtained from confusion matrices in the classification report. They improved my frontend design by using Flask and CSS, which increased appeal and inventiveness. Using CSS styling, they created a heatmap overlay that shows major changes between image regions for classification.

1. Preliminary Work:

2.1 Basic Structure

What is a convolutional Neural Network? A Convolutional Neural Network (CNN) is a form of neural network designed specifically for image processing. CNNs are widely used in applications where computers recognise objects inside images, but they can also be used for natural language processing tasks. Their adaptability makes them essential to advances in deep learning and artificial intelligence. To comprehend CNNs, they'll start with a standard neural network, which has input, hidden, and output layers. In this configuration, the input layer receives various input features, the hidden layers do computations, and the output layer returns the final results. Neurons in each layer are linked to neurons in the previous layer, with each neuron having a distinct weight. This architecture avoids making any assumptions about the incoming data. When dealing with images or words, CNNs behave differently, perceiving the data as spatial. Instead of connecting to every neuron in the preceding layer, neurons selectively link to neighbouring neurons with the same weights. This method preserves the dataset's spatial organisation, ensuring that the network does not view components like eyes as spread across the entire image. The term "convolutional" in CNN refers to the filtering process that takes place within the network. CNNs reduce complex images, allowing for more effective data processing and understanding. A Convolutional Neural Network (CNN) is made up of multiple layers, each of which contributes uniquely to the network's functionality. Among these layers are the convolutional layer and the pooling layer, which are critical to the architecture. Additionally, CNNs, like other neural networks, have a ReLU layer for activation and a fully connected layer for classification. The ReLU layer acts as an activation function, introducing nonlinearity as data moves through the network's layers. This non-linearity is required to maintain the dimensionality of the data, which is critical for effective feature extraction. Without ReLU, the data risks losing its dimensional properties. On the other hand, the fully connected layer makes classification tasks on the dataset easier, allowing the network to make predictions or choices based on the extracted features. However, the convolutional layer stands out as the foundation of the CNN. It works by applying a filter to an array of picture pixels, resulting in a convolved feature map. This procedure is critical for finding patterns and characteristics in the input data, making it an essential step in image analysis and identification tasks

2.2 Activation Function

2.2.1 ReLU Activation Function

As of 2017, the rectifier was the most often used activation function for deep neural networks. This function is widely used in units known as rectified linear units (ReLUs).[2]

The main reason ReLU wasn’t used until more recently is because it was not differentiable at the point zero. Researchers tended to use differentiable activation functions like sigmoid and tanh. However, it’s now determined that ReLU is the best activation function for deep learning.[2] In computational processing, negative values are set to zero, and positive values are evaluated based on their highest magnitude. When backpropagating in neural networks, the differentiation of the Rectified Linear Unit (ReLU) function is relatively simple. They assume that the derivative at zero is also zero, which is a common convention that produces reliable results. The derivative represents the function's slope; for negative values, it is 0.0, while for positive ones, it is 1.0.[2] The ReLU activation function provides several significant advantages: Convolutional Layers and Deep Learning: ReLU is commonly used to train convolutional layers and deep learning models because to its efficacy. Computationa simplicity: Implementing the rectifier function is simple, using only the max () function to improve computational efficiency. Representational Sparsity: One key feature of ReLU is its capacity to generate real zero outputs, which promotes representational sparsity in the network. Linear behaviour: ReLU's linear or near-linear behaviour simplifies optimisation procedures within neural networks, making them easier to train effectively.

2.2.2 Sigmod

The sigmoid function follows an S-shaped curve. It begins around zero when \(x \) is very large negative, rises sharply as \ (x \) increases, and eventually converges to 1 when \(x \) becomes very large positive. Similarly, as \ (x \) grows extremely negative, the function converges to zero. Sigmoid functions are widely utilised in

machine learning methods, especially logistic regression and artificial neural networks. In logistic regression, the sigmoid function is used to convert the linear model's output to a probability value between 0 and 1, expressing the likelihood of a specific result. In neural networks, sigmoid functions are often utilised in the activation functions of hidden layers. They introduce non-linearity to the model, allowing it to learn complicated patterns in the data.However, it is important to note that sigmoid functions have several downsides, such as the vanishing gradient problem, particularly in deep neural networks. As a result of this issue, many recent neural network topologies now use alternate activation functions such as ReLU (Rectified Linear Unit).

2.3 Optimizer and Loss Function

2.3.1 Adam Optimizer:

Adam (Adaptive Moment Estimation) is a widely used adaptive optimisation approach for training deep neural networks. It is an adaptation of the stochastic gradient descent (SGD) technique that aims to overcome some of its drawbacks.

- Adam maintains adaptive learning rates for each parameter by computing the first and second moments of the gradients. It adjusts the learning rates for each parameter based on the estimated moments.
- First and Second Moments: - Adam calculates the first moment (mean) of the gradients, indicating the average gradient. It also calculates the gradients' second moment (variance), which is the average of the squared gradients.
- Parameter Update: Adam mixes the first and second moments to update the parameters. It uses the first moment accounts for the gradient descent's direction, whereas the second moment scales the learning rate for each parameter.
- Combining Advantages: Adam combines the benefits of AdaGrad and RMSProp, two prominent optimisation methods. AdaGrad uses per-parameter learning rates, which can result in aggressive updates for uncommon parameters. However, it may reduce the learning rate excessively over time. RMSProp scales the learning rate by taking a moving average of squared gradients. It addresses AdaGrad's aggressive learning rate decline but does not include adjustable learning rates for specific parameters.

2.3.2 Categorical Crossentropy:

Categorical Crossentropy is a loss function often used in multi-class classification problems with one-hot encoding. It assesses the disparity between the true distribution (ground truth) and the projected probability distribution for the classes.Categorical Crossentropy is an important loss function in multi-class classification problems, particularly when the output is represented using one-hot encoding. Let's dig deeper into how categorical crossentropy works:True Distribution (Ground Truth): In multi-class classification, each input sample is allocated to a specific class label. One-hot encoding is commonly used to encode the real distribution (yy). Create a vector with length N N, where N represents the number of classes. This vector is completely filled with zeros, with the exception of a single 1 at the index corresponding to the true class label. The model estimates the probability distribution across classes based on input samples. The expected distribution is represented by p p, a vector of length N.N encapsulates the predicted probability for each class.Categorical crossentropy calculates the difference between the true distribution (y) and the expected distribution (p). It achieves this by comparing the genuine probability (1 for the proper class and 0 for the rest) to the anticipated probabilities for each class.The objective is to minimise categorical crossentropy loss, resulting in high probabilities for valid classes and low probabilities for incorrect ones. This loss function acts as the training objective, allowing the model to improve its ability to appropriately categorise input samples into their various classes. In essence, categorical crossentropy is an important component in training neural networks for multi-class classification tasks, allowing the quantification of differences between the true and predicted probability distributions.

3 Proposed Method

3.1 MaxPooling2D:

- Max pooling is a key function in convolutional neural networks (CNNs) that down samples input along spatial dimensions (height and breadth). Here's a detailed description of how the max pooling procedure works with 2D spatial data:
- Operation Overview: Max pooling works by sliding a 2D window (determined by the pool size argument) over the input data and calculating the maximum value within each frame. This procedure significantly decreases the spatial dimensions of the input while keeping the most important elements. [4]
- Arguments: - pool size: specifies the size of the pooling window. It can be an integer or a pair of integers (dim1, dim2). If only one integer is provided, the same window size is used for both dimensions. [4]
- Strides: Sets the stride of the pooling process. It may be an integer, a tuple of two integers, or None. If None, the stride is set to the pool size. If only one integer is supplied, the same stride size is utilised for both dimensions, just as it is for pool size.
- Padding: Determines the padding strategy. It can be "valid" (no padding) or "same" (padding to keep the output dimensions).[4]
- Data format: Specifies the order of dimensions in the input data. It might be either "channels last" or "channels first". The default value is usually determined from the Keras configuration file.[4]
- Input shape: If data format="channels last", expect a 4D tensor with the following shape: batch size, height, width, and channels. If data format "channels first": Expects a 4D tensor with the following shape: batch size, channels, height, and width.[4]
- Output shape: If data format "channels last": Generates a 4D tensor with the shape (batch size, pooled height, pooled width, channels). If data format "channels first": Generates a 4D tensor with the shape (batch size, channels, pooled height, pooled width). [4]

3.2 EfficientNet:

EfficientNet is a convolutional neural network architecture meant to balance model size, accuracy, and processing economy. It presents the concept of "compound scaling" to efficiently scale the three critical dimensions of a neural network: width, depth, and resolution.

- Width Scaling: - Increase or decrease the number of channels (or neurons) in each layer of the neural network.
- Increasing the width allows the model to capture more complicated patterns and features, perhaps leading to increased accuracy.
- Reducing the width, on the other hand, produces a more lightweight model that is more suited for deployment in low-resource contexts or on mobile devices.
- Depth Scaling: - This entails altering the number of layers in the neural network. Deeper models can capture more complicated data representations, allowing for feature extraction at a higher level.However, deeper models demand more computer resources and are more prone to overfitting, especially if the dataset is insufficiently large.
- Resolution Scaling: - Adjusts the size of the input image.Higher-resolution photos provide more precise information, which could lead to improved performance, particularly for applications that require fine-grained features. However, processing higher-resolution photos requires more memory and computational resources. Lower-resolution photos use fewer resources but may forfeit detail and precision in the supplied data.
- EfficientNet provides an ideal balance between these three dimensions by systematically scaling width, depth, and resolution. This compound scaling strategy ensures that the model's capacity grows efficiently, resulting in better performance while not considerably increasing computational cost or model size.
- EfficientNet has achieved cutting-edge performance on a variety of computer vision tasks while remaining highly efficient in terms of both accuracy and resource utilisation. It has become a popular alternative for a variety of applications, including image classification, object recognition, and segmentation, especially in resource-constrained situations like mobile and edge devices.[6]

3.3 Flask

Flask is a Python micro web framework that is widely used to create online applications, APIs (Application Programming Interfaces), and other web-based utilities. It gives developers a versatile and simple approach to build web apps and services with Python.
Flask is well-known for its simple style and ease of use, making it a popular choice for developing online apps. Unlike full-stack frameworks such as Django, Flask does not have built-in facilities for database operations, authentication, or form validation. Instead, it offers a lightweight framework that enables developers to select the components they require and tailor their applications accordingly.

4 Experiments

4.1: Dataset: This dataset contains 4,242 photos of different flowers that were methodically gathered from numerous sources such as Flickr, Google photos, and Yandex Images. The dataset is a significant resource for training and assessing machine learning models for recognising plants in pictures. The collection is divided into five categories, each representing a unique type of flower: Chamomile, Tulip, Rose, Sunflower, and Dandelion. Each class contains roughly 800 photos. The photos are of middling resolution, with an average of 320x240 pixels. Notably, the photos 'aspect ratios differ, which adds to the dataset's diversity. Potential applications:

- Botanical Research: Researchers can use this dataset to create algorithms for automated flower identification, which will help with botanical surveys and ecological studies.
- Educational Tools: The dataset can be used as a significant resource in education, facilitating projects and assignments on plant recognition and classification.
- Precision agriculture uses automated flower recognition systems to monitor crop health, identify invasive species, and optimise pollination tactics.
- Image Processing and Computer Vision: The dataset serves as a standard for assessing the performance of image classification systems, helping to further research in computer vision and pattern recognition.

4.2Training Model: According on the code provided, we appear to be utilising a convolutional neural network (CNN) architecture to train the model. CNNs are appropriate for image classification problems because they can record spatial hierarchies of features inside images. Your training model could have numerous convolutional layers followed by pooling layers that extract relevant information from the input flower photos. These convolutional layers can be followed by fully connected layers for classification using the retrieved features. Consider topologies like as LeNet, VGG, or a custom CNN architecture designed to meet your specific requirements. Because the photos in our dataset are low resolution and have different proportions, it is critical to preprocess them properly before feeding them into the network. Resizing photos to a consistent size, normalising pixel values, and using data augmentation techniques like rotation, flipping, and scaling can all assist improve the model's performance and generalisation ability.

RESEARCH METHODOLOGY

- Accuracy: Accuracy is a key parameter for evaluating the model's overall performance. It calculates the fraction of correctly categorised flower photos among the total number of photographs in the collection.
- Precision: Precision assesses the model's ability to make accurate positive predictions. In flower recognition, accuracy is the percentage of properly detected flowers out of all the blooms predicted by the model.
- Recall: Recall, also known as sensitivity, evaluates the model's ability to properly identify all relevant instances. In the context of flower recognition, recall refers to the percentage of properly recognised flowers among all flowers in the dataset.
- F1 Score: The F1 score is the harmonic mean of precision and recall, which provides a fair assessment of the model's performance. It is especially handy when there is an imbalance in the number of instances between classes.

- Confusion Matrix: The confusion matrix contains a detailed breakdown of the model's predictions, including the number of true positives, true negatives, false positives, and false negatives for each flower class. It provides insights on the model's strengths and flaws in various classes.

RESULTS AND DISCUSSION

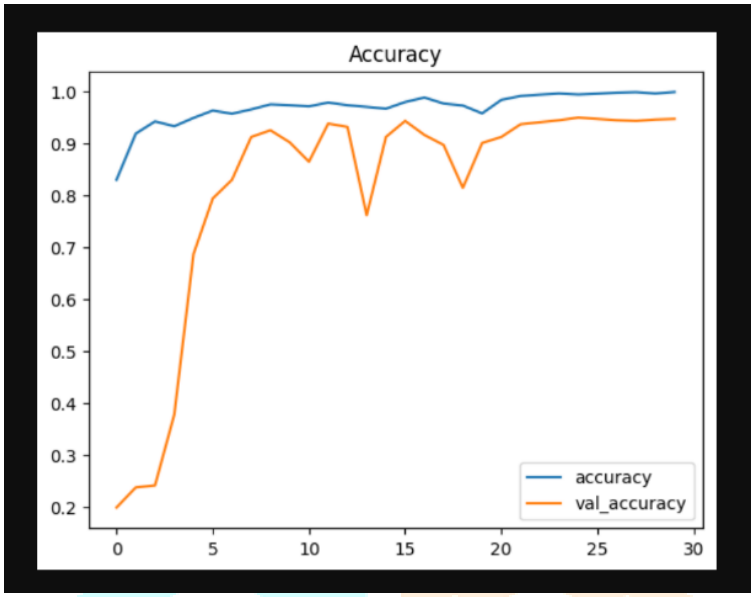

Fig.1 [Evaluation of accuracy and val accuracy (y-axis is Epoch and x-axis is accuracy)]

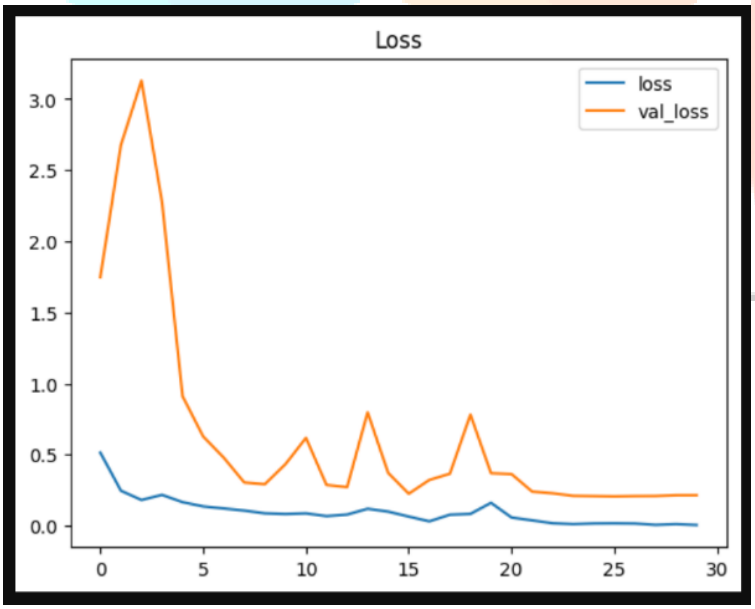

Fig.2 [Evaluation of Loss and val loss (y-axis is Epoch and x-axis is loss)]

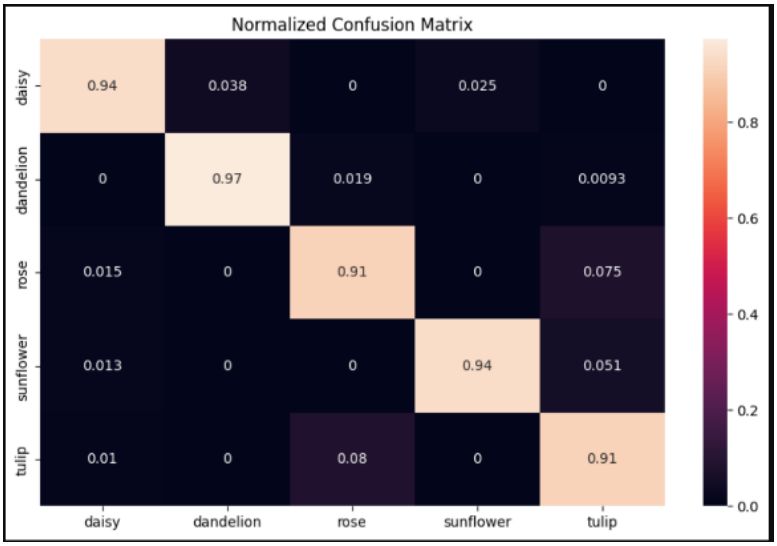

Fig.3 [Evaluation of Confusion Matrix and Flowers dataset (Difference in heatmap colour is showing the matrix of different groups

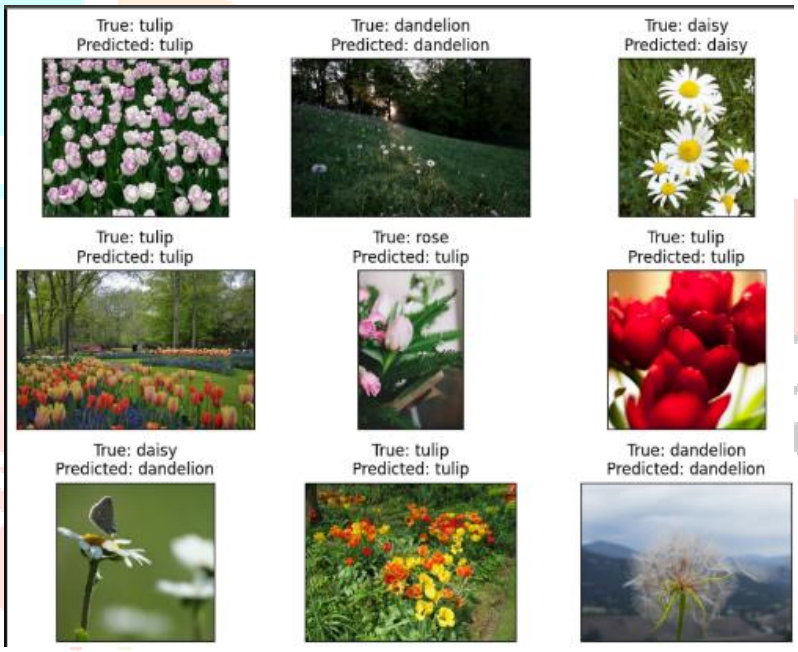

Fig.4 [Displaying the result after training and pre-model]

- Fig.1 shows an evaluation of accuracy and validation accuracy throughout different epochs. It offers information on the performance of both the CNN model and the predefined EfficientNet model. The graphic shows how the models' accuracy changes during training, as they'll as their performance on unseen validation data. By comparing these two curves, we can evaluate the models' capacity to generalise to new data and detect any potential overfitting or underfitting.
- Fig.2 displays the loss and validation loss throughout training epochs. Loss is the difference between the expected outputs of the model and the actual ground truth labels. The graphic depicts how the loss diminishes over time, reflecting the models' increasing ability to generate correct predictions. In contrast, validation loss assesses the models' performance using previously unreported validation data. They may examine the models' convergence and generalisation performance by looking at loss and validation loss trends.
- Fig.3 shows a heatmap of the confusion matrix, displaying precision, F1 score, and recall for each class in the dataset. The confusion matrix is an effective tool for understanding the models' classification

performance because it visualises the number of correct and wrong predictions across classes. The heatmap shows areas of high precision, suggesting the models' ability to correctly classify instances of each class, as well as areas for improvement, where misclassifications occur.

- Fig.4 shows the output comparison of trained models.It compares the output generated by the trained CNN model to the pre-defined EfficientNet model. By comparing the outputs of both models, they may determine their relative strengths and shortcomings in terms of classification accuracy and robustness. This comparison allows us to make informed decisions regarding which models are best suited to the specific task at hand, taking into account considerations like as computational efficiency, model size, and performance indicators.

Conclusion:

The EfficientNetB1 model trained on the flower dataset performs they'll in categorising floral photos, with high accuracy on both training and validation sets.The programme accurately predicts flower classifications based on previously unseen test data. The confusion matrix and classification report provide detailed information on the model's performance across many floral categories, showing areas of accurate classification and probable is classifications. Overall, the EfficientNetB1 model is a reliable and efficient solution for flower classification tasks, demonstrating the use of deep learning models in picture recognition applications.This code uses the EfficientNetB1 model to do a thorough examination of a flower categorization task. Here's a breakdown of the main steps and results:

- Data Preparation: - They load the flower dataset from a provided directory, extract file paths and labels, and organise the data using pandas Data Frames. The dataset is divided into training and testing sets, with 90% being utilised for training and 10% for testing.
- Data Augmentation and Image Generators: - They created image data generators for training, validation, and testing, using preprocessing routines and resizing photos to the desired size (224x224).
- Data augmentation techniques such as random flips and rotations are used on training images to boost diversity and improve model generalisation.
- Model Definition and Training: -They create a customised CNN model using the EfficientNetB1 architecture, adding dense layers for classification.The model is built with the Adam optimizer and the categorical cross-entropy loss function, with accuracy and AUC serving as assessment metrics. Training lasts 30 epochs, with early stopping and learning rate decrease callbacks to avoid overfitting and improve convergence.
- Model Evaluation: - They visualise the training history with plots that illustrate changes in accuracy and loss over epochs for training and validation sets.
  The trained model is evaluated on the test set to determine the final test loss and accuracy scores. A classification report and confusion matrix are created to evaluate the model's performance, which includes precision, recall, and F1-score for each class.
- Results Visualization: - They present photos from the test set with true and expected labels to visually check the model's predictions.

Reference:

[1] (Alam et al., 2021)Convolutional Neural Network for the Semantic Segmentation of Remote Sensing Images.

[2]BharathKrishnamurthy,Feb26,2024,AnIntroduction to the ReLU Activation Function(https://builtin.com/machine-learning/relu-activation-function)

[3]ShipraSaxena,26Oct,2023,IntroductiontoSoftmaxforNeuralNetwork(https://www.analyticsvidhya.com/blog/2021/04/introduction-to-softmax-for-neural-network/)

[4] MaxPooling2D layer(https://keras.io/api/layers/pooling_layers/max_pooling2d/)

[5]Ashi108,07,Feb,2024,UnderstandingFlaskFramework:Installation,features&ExpertInsights(https://www.analyticsvidhya.com/blog/2021/10/flask-python/)

[6] Petru Potrimba AUG 9, 2023 What is EfficientNet? The Ultimate Guide. (https://blog.roboflow.com/what-is-efficientnet/)