Assessment 3: Convolutional Neural Network

MA5852: Data Science Master Class 2

Name: Rijo Kuruvilla

Student ID: 14171011

I. Abstract

Soil, as a cornerstone of terrestrial ecosystems, sustains life and maintains environmental balance through its multifaceted functions. In addition to being a vital source of food and medicine, soil is also a major factor in climate change. Soil plays a vital role in sustaining natural gases like carbon and nitrogen by reducing flooding, cleaning, and filtering water bodies, and regulating the atmosphere. This report delves into the classification of six distinct soil types utilizing Convolutional Neural Networks (CNNs), aiming to address the need for improved soil classification methodologies. By leveraging CNNs, the study aims to streamline the classification process and enhance its accuracy, thereby facilitating more informed decision-making in various applications. The primary objective of this report is to analyze diverse soil types, apply data transformations, and construct and train a CNN model effective in classifying soil varieties. The methodological approach involves evaluating the various soil types, producing training and testing datasets for developing models, and using data augmentation to generate additional data. Furthermore, the CNN model is enhanced using techniques such as dropout, early stopping, and batch normalization to bolster performance and generalization. Preliminary findings reveal promising results, with the CNN model with tuned parameters achieving a classification accuracy of 64.10% when employing a dropout rate of 0.2. Additionally, the model exhibits a test loss of 0.9087, indicative of its robust performance with potential for further refinement. These outcomes underline the potential of CNNs to significantly enhance soil classification problems, offering superior efficiency, accuracy, and scalability compared to conventional machine learning approaches. The findings also highlight the utility of CNNs in soil classification, paving the way for more effective soil management and environmental stewardship. The developed model aims to provide significant benefits to soil scientists, agronomists, and environmentalists, ultimately leading to better soil management practices and environmental sustainability.

II. Introduction

The Earth's soil, a critical element within terrestrial ecosystems, stands as a vital pillar supporting life and ecological equilibrium. Soil is essential to many environmental processes because of its wide range of applications which include promoting agricultural productivity and serving as a repository for biodiversity (Zurich, 2023) . Its complex makeup and dynamic biological interactions highlight how essential it is to maintain the health of our planet and the welfare of its inhabitants (FAO, 2015, pp. 137,259) Soil is home to a wide variety of microorganisms, plants, and invertebrates that are essential for ecosystem functioning. This is in addition to its obvious benefits to agricultural productivity and ecosystem stability (Bardgett & van der Putten, 2014). Moreover, soil plays a pivotal role in global bio-geochemical cycles, influencing the distribution and cycling of essential nutrients such as carbon, nitrogen, and phosphorus (Wu & Yu, 2023, as cited in Bertolet et al., 2018, p. 139). Consequently, alterations in soil properties and processes can have far-reaching consequences for ecosystem resilience, agricultural sustainability, and climate regulation.

Although soil classification is of utmost importance, methods have historically depended on human labor and subjective criteria, which has resulted in limitations in accuracy and efficiency as well as inconsistencies. (Isbell, 2016). A promising path to increasing soil classification accuracy and expediting the process is provided by the development of sophisticated computational techniques, especially machine learning algorithms. This report aims to investigate the potential of CNN in soil classification as a response to these challenges. The goal is to improve soil classification accuracy and efficiency by utilizing deep learning algorithms, which will help make better decisions about agricultural and environmental management. The ultimate goal is to advance soil science and sustainability by creating a reliable CNN model that can distinguish between different types of soil through thorough exploration and analysis.

The key objectives of this report include analyzing different types of soil using visual data and utilizing deep learning with neural networks for soil classification. By leveraging the power of CNNs, this study aims to streamline the classification process and enhance its accuracy, ultimately facilitating more informed decision-making in various applications.

This study sets the stage for the investigative efforts detailed in this report, highlighting the significance of soil classification, and providing an explanation of the rationale behind CNN adoption as a potential remedy for current issues. The intention is to improve soil classification techniques for the benefit of the environment and make a significant contribution to the field of agricultural science through comprehensive research.

III. Data Preparation

The data under consideration is the 'Soil Types' dataset, which contains 144 images of six different soil types: ‘Alluvial’, ‘Clayey’, ‘Laterite’, ‘Loamy’, ‘Sandy loam’, and ‘Sandy soil’. The dataset was obtained from Kaggle (Matshidiso, n.d). Using exploratory data analysis (EDA) the soil images were analyzed. As depicted in Figure 1, these images are formatted in RGBA, indicating the presence of four channels: Red, Green, Blue, and Alpha. The Alpha channel dictates the opacity of the color, presenting an additional layer of complexity to the images (Geeks for Geeks, 2023). Notably, the varying image dimensions can also be observed across the dataset in Figure 1. To standardize the analysis, all images were rescaled to a uniform size of 256 x 256 pixels and converted to RGB format, facilitating streamlined processing and comparison.

Figure 1

Six different types of Soil

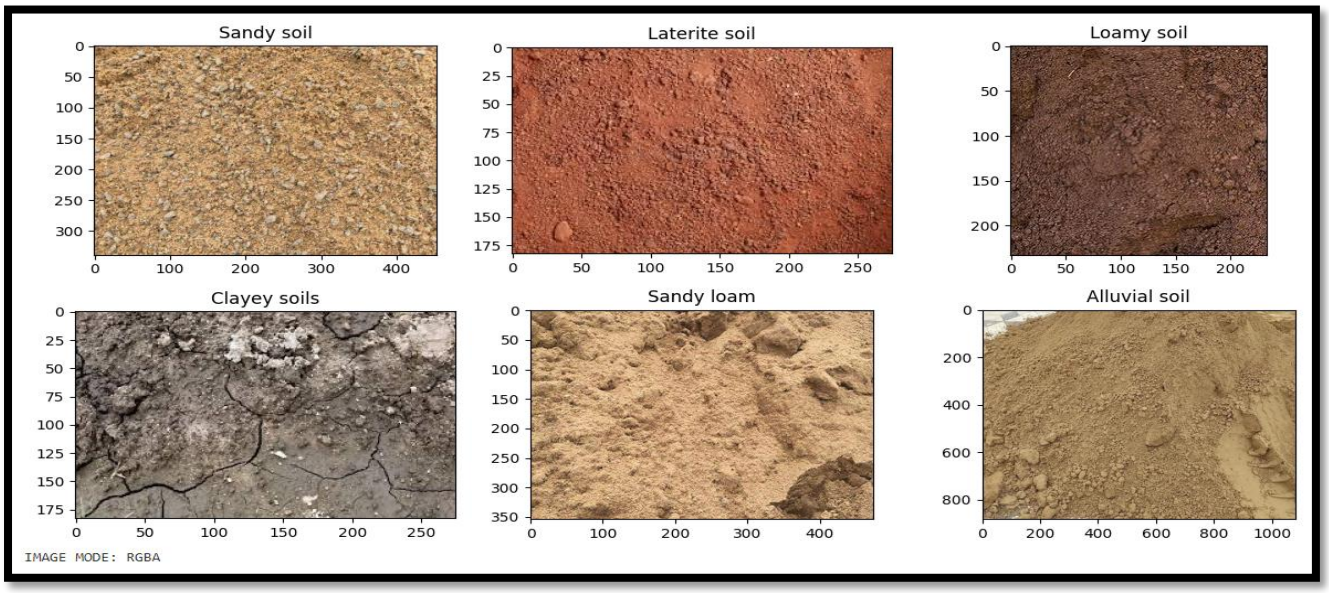
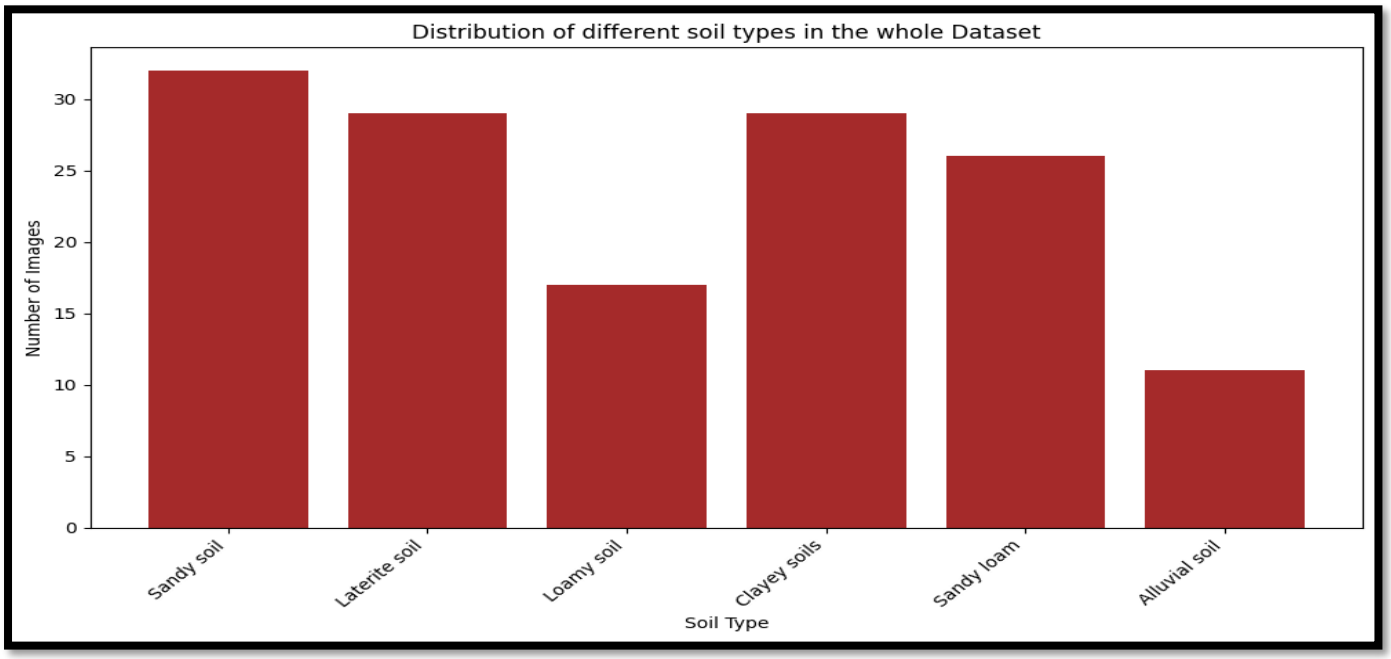


Figure 2 illustrates the bar chart displaying the frequency distribution of the different types of soil. Sandy Soil predominates with 32 images, closely followed by Laterite and Clayey Soil, each comprising 29 images. Meanwhile, Sandy Loam and Loamy Soil contain 26 and 17 images, respectively, with Alluvial soil containing the fewest images at 11. This distribution provides valuable context for understanding the dataset's composition and informs subsequent analytical endeavors.

Figure 2

Soil distribution by soil types



Data Partitioning and Data Augmentation

After conducting EDA, the dataset was divided into distinct subsets for training and testing. To facilitate this, six separate folders were created, each designated for a specific soil type: 'Alluvial', 'Clayey', 'Laterite', 'Loamy', 'Sandy loam', and 'Sandy soil'. This segregation ensured that the model was trained and evaluated on a diverse set of samples representative of each soil type. The partitioning used was a 75:25 split ratio, with 75% of the data allocated for model training and 25% for testing, ensuring a balance between training efficacy and model evaluation robustness. After splitting the data there were 105 images allocated for training whereas 39 images for testing. The standard procedure is to split the data into training, validation, and test sets; however, due to the small size of the data set, only a train-test split was used.

As mentioned earlier in the data preparation section, the images were rescaled to have a height and width of 256 pixel each. A batch size of 64 was also initialized to ensure consistency. This standardization simplifies data processing and ensures that the model can effectively analyze and classify images regardless of their original size or aspect ratio.

After splitting the data, data augmentation was applied on the training dataset. Data augmentation is a technique of artificially increasing the training set by creating modified copies of a dataset using existing data (Ali Awan, 2022) . It includes making minor changes to the dataset or generating new data. Data Augmentation's primary goal is to diversify the dataset by applying a variety of transformations to existing images, simulating real-world scenarios, and improving the model's ability to generalize to previously unseen data (test data). For the classification of soils, the following data augmentation techniques were used:

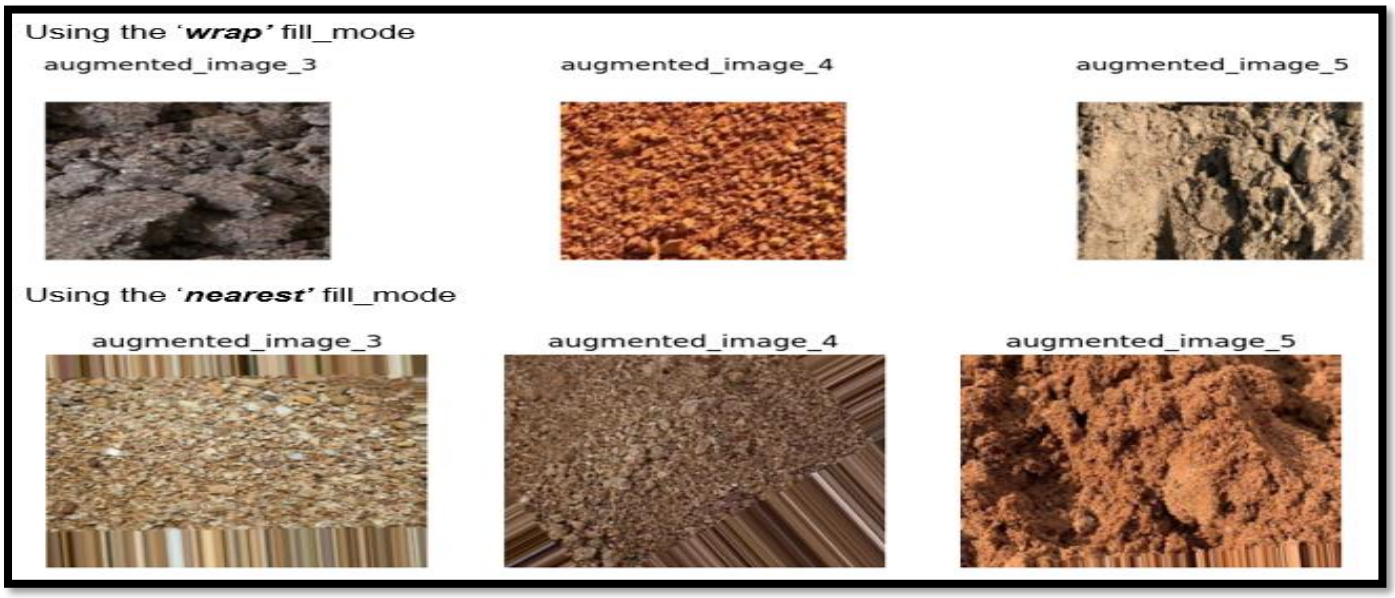
- *Rescaling* : The pixel values of the images were rescaled by a factor of 1/255.0. This rescaling operation ensures that the pixel values fall within the range of 0 to 1, facilitating numerical stability and convergence during model training.
- *Rotation* : Images were subjected to a rotation of 35 degrees. This ensures that the model can accurately classify images regardless of their rotational alignment.
- *Width and Height Shifting :* The images width and height were randomly shifted by up to 20%. This manipulation creates spatial variations that mimic potential changes in camera angles

or distances during image acquisition. By exposing the model to such variations, it learns to recognize and classify soil samples in a variety of spatial configurations.

- *Shear*: Shear is a geometric augmentation that modifies an image's shape along a particular axis to produce an alternative perception angle (Cloud Factory, n.d.) . This augmentation technique simulates potential deformations in soil samples caused by environmental factors or imaging artifacts, ensuring that the model remains robust to such variations.
- *Zoom*: Random zooming of images by up to 40% was used, allowing the model to detect soil characteristics at various scales. This augmentation technique improves the model's ability to capture fine-grained details within soil samples, resulting in higher overall classification accuracy across a wide range of resolutions.
- *Horizontal Flipping*: Horizontal flipping ensures that the images are randomly flipped horizontally effectively mirroring the soil samples along the vertical axis. This augmentation technique introduces changes in image orientation, which helps the model learn invariant features and reduces the risk of overfitting to specific orientations in the training data.
- *Fill Mode*: 'fill_mode' is the strategy used for filling in newly created pixels, which can appear after a rotation or a width/height shift (Chollet, 2016). For the soil classification, both 'wrap and 'nearest modes were tested. However, 'wrap' was used over 'nearest' for further analysis as it provided better results in the preservation of image continuity and minimizing abrupt discontinuities at the borders as displayed in Figure 3 , thereby facilitating smoother transitions and more visually consistent augmented images.

Figure 3

*Comparison of the Augmented Images using '**wrap'** and '**nearest'** fill_modes*



In contrast to the training data, the test data underwent only one augmentation technique, which involved rescaling the images to maintain consistency with the training data. This rescaling process ensures that the pixel values in the test images are in the same range as those in the training set, allowing the model to process the data uniformly during the evaluation phase. After preparing the test data to match the model's training conditions, a CNN model was built to train and classify the distinct soil types.

IV. Build, train, and deploy a neural network.

a) Propose a CNN for the classification. Justify your choice.

A Convolutional Neural Network (CNN) is a deep learning architecture used to process visual data such as images. It works by passing the input image through a series of layers, including convolutional, pooling, and fully connected layers. Filters or kernels are used in convolutional layers to extract features from images, which are then activated using activation functions like Rectified Linear Unit (ReLu) to introduce nonlinearity. Pooling layers reduce spatial dimensions while preserving important information. The fully connected layers perform classification based on extracted features, frequently using the SoftMax function to generate class probabilities. The network learns to recognize patterns and features through iterative training on labeled data, which allows for accurate image classification and object detection.

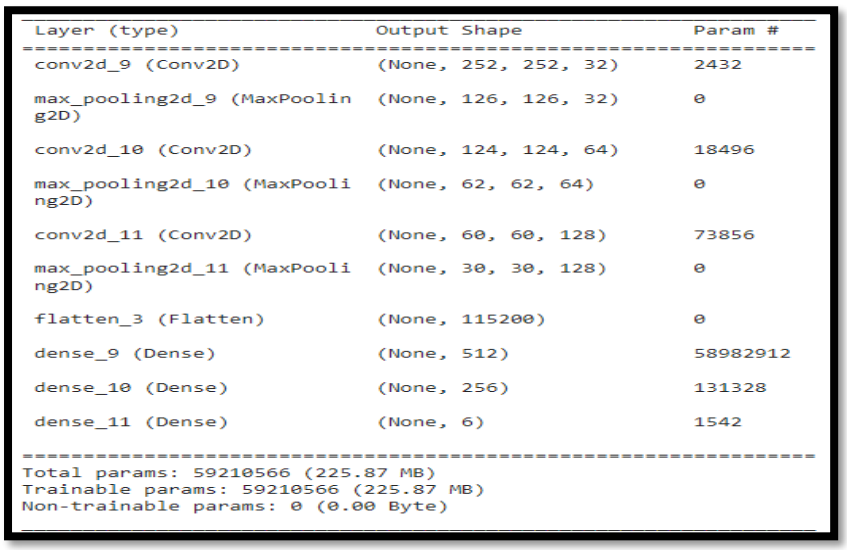
For soil type classification, CNN is an appropriate choice due to its effectiveness in handling image data. CNNs are especially well-suited for the subtle complexities found in soil imagery because they are specifically designed to perform exceptionally well in tasks involving image recognition and classification. These networks are skilled at extracting and learning complex patterns and features found in soil samples, such as textures, shapes, and spatial arrangements, thanks to their hierarchical architecture made up of convolutional layers. Furthermore, CNNs demonstrate outstanding flexibility in response to changes in soil images caused by variables such as illumination, camera angles, and soil type, guaranteeing strong classification results in a variety of environmental settings. The ability of CNNs to distinguish minute differences between soil types can be leveraged to improve the soil type classification efforts, ultimately producing more accurate and dependable classification results. Figure 4 displays the base CNN architecture proposed for classifying different types of soil.

Before building the model, an evaluation of class imbalance was conducted to ascertain the distribution of soil types within the dataset and to address any potential biases that may affect model performance. As previously shown in Figure 2, the distribution of soil types revealed an imbalance, with some classes having significantly fewer samples. For example, there are 32 samples of “Sandy soil” and only 11 samples of “Alluvial soil”. To address this issue and ensure fair learning across all classes, class weights were computed. By assigning appropriate weights to each class based on their frequency in the dataset, the model training process is optimized to account for class imbalance. This method improves the model's ability to generalize well across all soil types and make accurate predictions, regardless of class distribution.

The first layer is the input layer, which has been configured to receive images with 256 x 256-pixel dimensions and three RGB color channels. Three convolutional layers are then used to extract features with hierarchies from the input images. The first convolutional layer uses the Rectified Linear Unit (ReLU) activation function and has 32 filters of size (5, 5). A max-pooling layer with a pool size of (2, 2) is used for down sampling. In a similar manner, the next convolutional layers are made up of 64 and 128 filters, respectively, and are each followed by max-pooling and ReLU activation processes to extract and down sample additional features. A flatten layer is used after the convolutional layers to transform the 2D feature maps into a 1D vector. Two fully connected layers, one with 512 neurons and the other with 256 neurons follow this, each with dense neurons with ReLU activation functions. To enable multi-class classification, the output layer, which contains six neurons, one for each of the six soil types, employs a SoftMax activation function. The model consists of 59,210,566 trainable parameters.

Figure 4

Proposed CNN architecture for soil classification



Layer (type)	Output Shape	Param #
conv2d_9 (Conv2D)	(None, 252, 252, 32)	2432
max_pooling2d_9 (MaxPooling2D)	(None, 126, 126, 32)	0
conv2d_10 (Conv2D)	(None, 124, 124, 64)	18496
max_pooling2d_10 (MaxPooling2D)	(None, 62, 62, 64)	0
conv2d_11 (Conv2D)	(None, 60, 60, 128)	73856
max_pooling2d_11 (MaxPooling2D)	(None, 30, 30, 128)	0
flatten_3 (Flatten)	(None, 115200)	0
dense_9 (Dense)	(None, 512)	58982912
dense_10 (Dense)	(None, 256)	131328
dense_11 (Dense)	(None, 6)	1542

Total params: 59210566 (225.87 MB)
Trainable params: 59210566 (225.87 MB)
Non-trainable params: 0 (0.00 Byte)

Justifications for the proposed model:

- *Ability to Capture Spatial Hierarchies*: The proposed CNN architecture's convolutional layers are made to capture the spatial hierarchies of the features found in soil images. Through the application of filters with different sizes and their hierarchical stacking, the model can extract and learn intricate spatial patterns and textures found in various types of soil (Yamashita et al., 2018).

- *Translational Invariance*: CNNs can identify features regardless of where they are in the image because they are translationally invariant by nature (Sue, 2023). This characteristic is essential for classifying soil types because different images may have different soil sample positions and orientations. By down sampling and aggregating features over different spatial regions, the max-pooling layers improve translation invariance even more.

- *Parameter sharing*: The proposed model facilitates parameter sharing using convolutional layers. In convolutional layers, a set of filters (also known as kernels) is applied to the input image to extract features. These filters slide across the input image, computing dot products between their weights and the values of the input pixels at each position. Parameter sharing occurs because the same set of filter weights is used across the entire input image (Bano, 2023). This means that the learned parameters (weights) of the filters are shared across different spatial locations of the input image. As a result, the model adapts to be computationally efficient and effective when dealing with similar features across the entire image, making it well-suited for soil classification.

- *Adaptability to Image Variance*: With its convolutional and pooling layers, the suggested CNN architecture is well-suited to handle variations in soil images brought about by variables like camera angles, lighting, and soil texture. The model's ability to learn robust representations invariant to such variations is improved by extracting hierarchical features and down sampling spatial dimensions.

- *Scalability*: The proposed model can be readily expanded to suit more profound architectures, facilitating the extraction of progressively more ethereal characteristics. The scalability of the proposed architecture allows it to learn discriminative features such as depth, width and resolution that are specific to various soil types, even when there are minute variations between them. For tricky classification tasks, such as soil type classification, where subtle differences between soil types might necessitate learning deeper representations, this scalability can prove to beneficial.

b) Report and discuss the performance of the proposed CNN models.

The performance of the proposed CNN models was evaluated through training and testing processes. The model was compiled using the Adam optimizer and categorical cross-entropy loss function, with accuracy as the evaluation metric. It was trained over 30 epochs using the training

generator, with steps per epoch set to the length of the training generator. The test data were evaluated using the test generator, and class weights were applied to account for class imbalance during training.

Upon evaluation, the model achieved a test accuracy of approximately 64.10%. Further analysis was conducted by predicting classes for the test data. The predicted classes were compared with the actual class labels, revealing a range of predictions across different soil types. Figure 5 shows the confusion matrix heatmap, which provides a comprehensive overview of the model's performance across different soil types. Figure 6 compares predicted and actual images, highlighting some of the model's correct and incorrect predictions. Overall, the model demonstrates reasonable performance, with varying levels of accuracy across different soil types. The model accurately predicts most samples of 'Sandy Soil' (7 out of 8) and 'Laterite Soil' (5 out of 8), achieving high precision and recall. However, there are instances of misclassifications, such as misclassifying 3 instances of 'Clayey Soil' with 'Sandy Soil' and 1 instance 'Sandy Loam Soil' with 'Alluvial Soil'.

Figure 5

Heatmap displaying the proposed model's classification performance

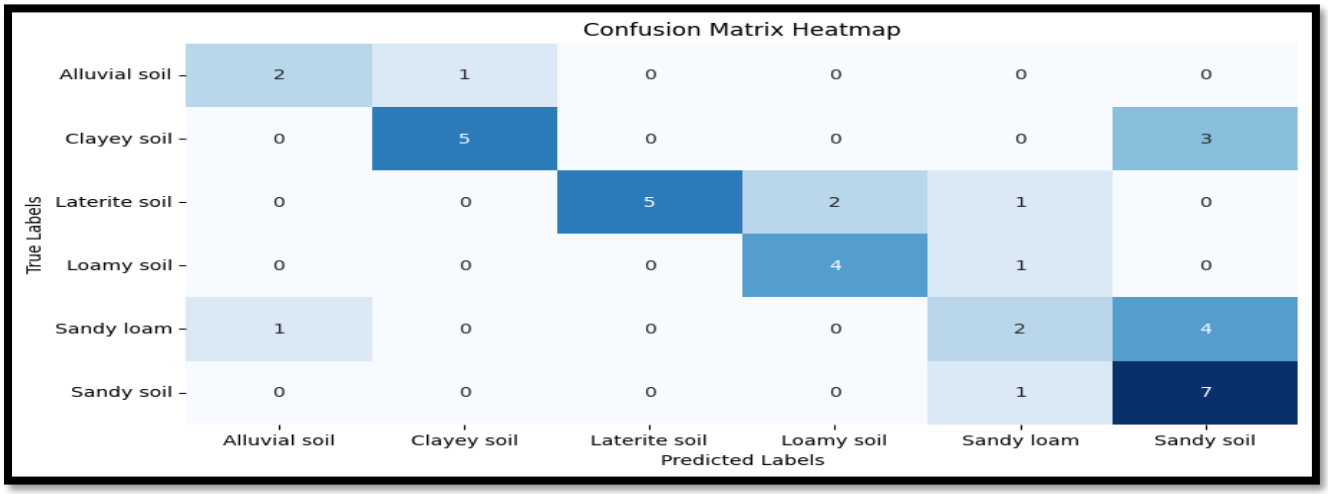
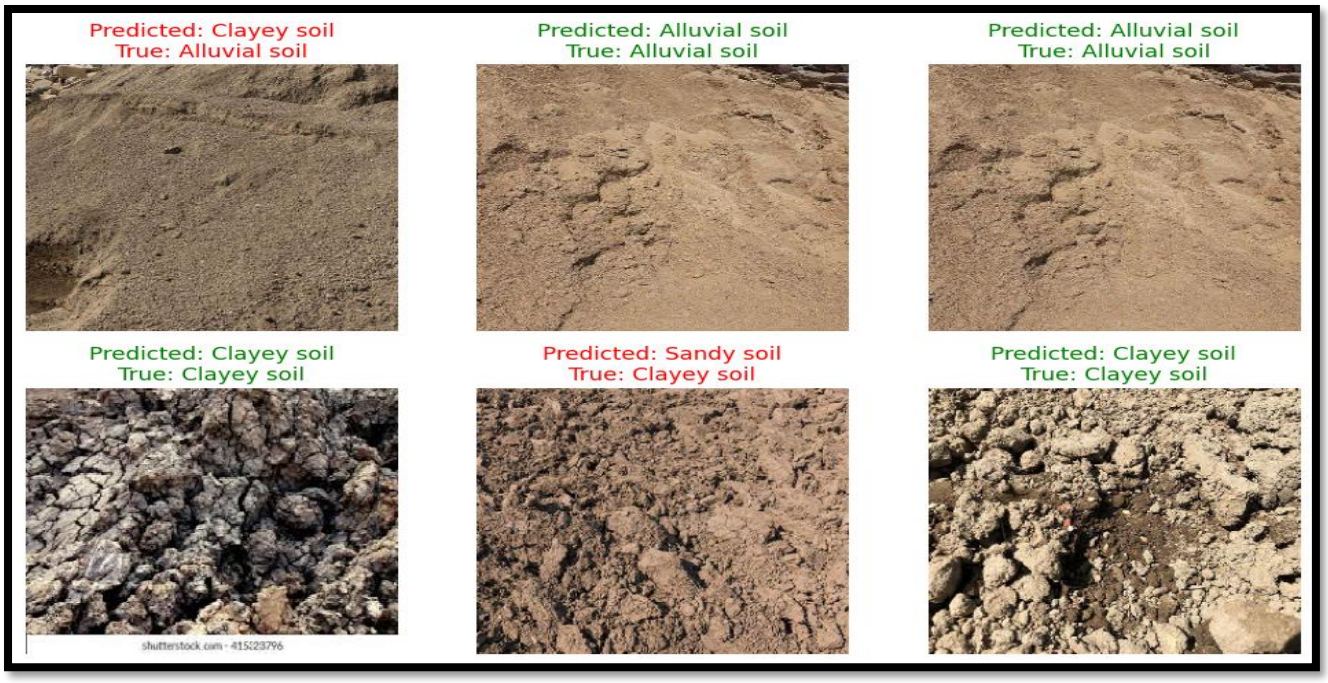


Figure 6

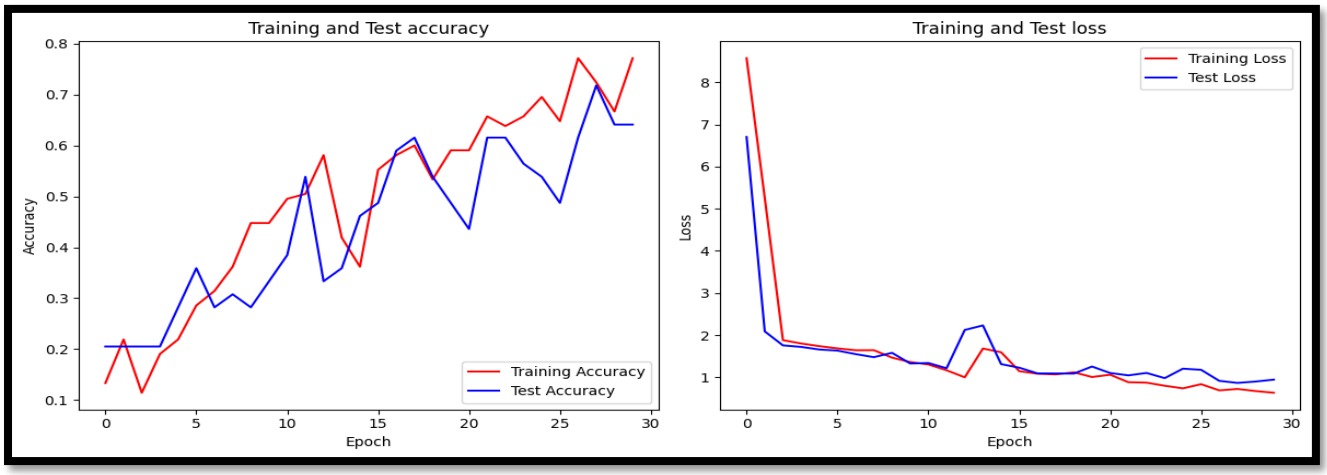
Comparison of Predicted Images by the Proposed Model with Actual Images



The line charts in Figure 7 provide insights into the training and test performance of the proposed CNN model over successive epochs. While the blue line, which represents test accuracy, initially hovers around 20.51% before stabilizing around 64.10%, the red line, which represents training accuracy, starts at roughly 13.33% and gradually rises to around 77.14% by the end of training. Simultaneously, the line chart on the right shows that the training loss (represented by the red line) gradually decreases from 8.57 to approximately 0.63, while the test loss (represented by the blue line) fluctuates before settling around 0.94. These patterns indicate that the model learns well from the training set of data, and the test metrics demonstrate that the model generalizes well to new sets of data. Differences in test metrics, however, might call for additional research to rule out overfitting or unstable models. While the model demonstrated reasonable performance across various soil types, further refinement using techniques such as Batch Normalization, Dropout, and Early Stop was used to enhance accuracy, which is discussed in the next section.

Figure 7

Comparing the accuracy and loss of training and test of the proposed model



c) Apply techniques such as dropout, early stop, batch normalization to the CNN (in (b)) and investigate their impacts on the performance on the CNN.

Building on the base model, Batch Normalization, Dropout, and Early Stop were used to improve model performance. Apart from applying the aforementioned techniques, a *Stride* of one and the 'same' *Padding* technique was applied to each convolutional layer. Strides control the step size of the filter's movement during convolution, influencing the reduction of spatial dimensions in output feature maps, whereas 'padding' ensures output feature maps match input dimensions, facilitating seamless integration with subsequent network layers. Strides and padding collectively optimize feature extraction and spatial information preservation in convolutional layers, enhancing overall network performance.

Batch Normalization: Batch Normalization is a technique used to improve the training speed, stability, and performance of neural networks by normalizing the activations of each layer. Batch Normalization is a two-step process where first the inputs are normalized and later re-scaled and offset using the learned parameters (Saxena, 2024) . In addition to stabilizing the training process and enabling higher learning rates, batch normalization aids in addressing the internal covariate shift issue and promotes faster convergence and improved generalization.

Dropout: Dropout is a computationally inexpensive and powerful method for avoiding overfitting. Dropout avoids overfitting by employing an ensemble approach. Dropout works with a substructure of the network at each iteration and combines the results of all iterations for the final output, as opposed to training the entire network (James Cook University, n.d.).

Early Stop: Early Stop, like Dropout is a regularization technique used to prevent overfitting by monitoring the model's performance on a test set during training and stopping the training process when the performance no longer improves (James Cook University, n.d.). Early Stopping strikes the

ideal balance between bias and variance, which helps keep the model from memorizing the training set and improves its ability to generalize to new data.

The process followed in this report involves applying Batch Normalization to the proposed model with 30 epochs first, then testing Batch Normalization with Early Stop with 100 epochs. Similarly, Dropout was applied to the proposed model for 30 epochs, followed by Dropout with Early Stop for 100 epochs.

Also in this model, Batch Normalization and Dropout were not combined because doing so would create inconsistency between the two techniques (Kim, 2021). Dropout alters the variance of individual neural units when transitioning from training to testing phases, whereas Batch Normalization maintains statistical variance accumulated during training into the testing phase. This inconsistency, known as "variance shift," results in unstable numerical behavior during inference and leads to more inaccurate predictions (Li et al., 2018). Additionally, the decision to avoid combining these techniques was influenced by the small size of the dataset.

Table 1 provides a comparison of the performance metrics for different combinations of Batch Normalization, Dropout, and Early Stopping techniques. The model's training accuracy was 93.33% with a corresponding loss of 1.68 when Batch Normalization was applied after each Convolutional Layer; after 30 epochs, the test accuracy was 23.08% with a loss of 145.15. After training for 100 epochs and stopping at epoch 11, Early Stopping with Batch Normalization led to lower training accuracy (82.86%) and higher loss (9.35), but improved test accuracy (38.46%). Similarly, Dropout alone, with a dropout rate of 0.2, produced a training accuracy of 64.76% and a loss of 0.83, with a test accuracy of 69.23% and a loss of 0.71 after 30 epochs. Combining Dropout with Early Stopping yielded a higher training accuracy (79.05%) and lower loss (0.45), while the test accuracy increased to 84.62%, with a loss of 0.46 after training for 100 epochs, stopping at epoch 58.

The results of applying each technique indicate their respective influences on the model's training and generalization capabilities. When batch normalization is used exclusively, the model's training accuracy increases dramatically, indicating improved convergence during training. The difference in test and training accuracies, however, suggests that there may be a risk of overfitting because the model might not generalize well to new data. Introducing Early Stopping with Batch Normalization mitigates this risk by halting training when the model starts to overfit, resulting in a more balanced performance between training and test accuracies. By randomly removing a portion of neurons during training, Dropout, on the other hand, acts as a regularization technique that effectively lowers overfitting and enhances the model's capacity for generalization. The regularization effect is further enhanced when Dropout and Early Stopping are combined. This results in better test accuracy and lower loss, which points to a more reliable and generalized model (See Appendix for the Line Charts for each technique).

To further improve the model performance, it would be beneficial to explore the impact of adjusting hyperparameters such as the dropout rate, batch size, and learning rate on model performance, which is discussed in the next section. Further research into more complex architectures and ensemble techniques may also result in better model performance. Moreover, investigating methods to expand the dataset or utilizing transfer learning from previously trained models may lead to improved model generalization and robustness.

Table 1

Performance Comparison of Batch Normalization, Dropout, and Early Stopping Technique and the AlexNet Model

Techniques	Description	Total Parameters	Epochs	Epoch stopped at	Train Accuracy	Train Loss	Test Accuracy	Test Loss
Batch Normalization	One batch after each Convolutional Layer	Trainable: 67335942 Non-trainable: 448	30	-	93.33	1.68	23.08	145.15
Batch Normalization + Early Stop	One batch after each Convolutional Layer	Trainable: 1,237,126 Non-trainable params: 1152	100	11	82.86	9.35	38.46	4.61
Dropout	One Dropout Layer placed before the Flatten layer, Dropout rate = 0.2	Trainable: 26,866,3622 Non-trainable params: -	30	-	64.76	0.83	69.23	0.71
Dropout + Early Stop	One Dropout Layer placed before the Flatten layer, Dropout rate = 0.2	Trainable: 67,337,030 Non-trainable params: -	100	58	79.05	0.45	84.62	0.46
AlexNet model		Trainable: 71,940,166 Non-trainable params: - 2752	100	11	65.71	3.1955	20.51	639.6170

d) Apply hyperparameter tuning to the CNN

Hyperparameter tuning aims to systematically search the hyperparameter space to find the optimal configuration that maximizes model accuracy and generalization. Since the model with Dropout combined with Early Stopping generated the best model performance, hyperparameter tuning using the Keras Tuner, a library by Keras, was utilized on that model to obtain the optimal hyperparameters for CNN .

The Hyperband tuner, a component of Keras Tuner, was utilized to efficiently explore the hyperparameter space, allocating resources to promising configurations while discarding fewer effective ones (Singh, 2021). This algorithmic approach is guided by the objective of maximizing validation accuracy. During the hyperparameter search, various hyperparameter configurations were evaluated using the provided training and test data. Early stopping was implemented to prevent overfitting and restore the best weights when the test loss ceased to improve, ensuring optimal convergence. Once the hyperparameter search concluded, the best set of hyperparameters was extracted from the tuner, and the optimal model architecture was constructed accordingly. Table 2 provides an overview of the tuned hyperparameter and the optimal parameters after tuning.

The model architecture encompasses three convolutional layers followed by max-pooling layers, facilitating feature extraction and spatial down sampling. Hyperparameters such as the number of filters (ranging from 32 to 128), kernel sizes, and strides in these convolutional layers are tunable. Additionally, two dense layers, along with a dropout layer to mitigate overfitting, perform classification based on the extracted features. The number of units in these dense layers (ranging from 128 to 512 for the first layer and 64 to 256 for the second layer) and the dropout rate (set at 0.2) are adjustable. The output layer, employing a SoftMax activation function, produces the final classification output for the six soil types. Moreover, hyperparameters like the learning rate (chosen from 0.001 or 0.1) and optimization algorithm (Adam) are specified during model compilation, providing further flexibility. Through hyperparameter tuning using Keras Tuner, the model dynamically adjusts its architecture and learning parameters to optimize accuracy and generalization for soil classification tasks.

Table 2

An overview of the Hyperparameters tuned to obtain the optimal parameters

Layer Type	Hyperparameters	Value / Range	Best Parameters Obtained
Convolutional	Number of Layers	3	'conv1_units': 96, 'conv2_units': 32, 'conv3_units': 32
	Number of Filters	32 to 128 (Step size = 32)	
Dense	Number of Layers	2	'dense1_units': 128, 'dense2_units': 192
	Number of Dense Units	128 to 512 (Step Size = 128) , 64 to 256 (Step Size = 64)	
Optimization	Learning Rate	0.001 to 0.1	'learning_rate': 0.001
	Optimization Algorithm	Adam	-
Callbacks	Early Stopping	5	-
Drop out	Dropout Rate	0.2	-
-	Maximum Epochs	10	-
-	Factor	3	-

Analyzing the performance of the Tuned Model

The classification report in Figure 8 is utilized to assess the model's performance for each soil type, and included precision, recall, and F1-score metrics. The model achieves an accuracy of 64.10% like the base model. However, the range of hyperparameters was lowered compared to the base model due to RAM and GPU limitations in Google Colaboratory (Colab) . While executing the model , the Google Colab session crashed stopping model execution when the GPU reached its limits in Google Colab. The model took approximately 1hr and 46 minutes to execute over 30 trials coupled with Colab session crashing two times. (See Appendix Image 9 for the Image displaying the Crash Message).

Precision measures the accuracy of positive predictions, recall measures the proportion of actual positives that were correctly identified, and F1-score is the harmonic mean of precision and recall. The report highlights variations in performance across different soil types, indicating areas where the model excelled or could be improved. For example, considering the class 'Clayey soil', the precision of 0.78 suggests that when the model predicts a sample as 'Clayey soil', it is correct 78% of the time. The recall of 0.88 indicates that the model identifies 88% of all 'Clayey soil' samples correctly. The F1-score, which is the harmonic mean of precision and recall, is 0.82. These metrics provide insights into the model's performance for each class.

The confusion matrix inf Figure 9 complements the classification report by visualizing the model's predictions against the actual class labels. For instance, in the row corresponding to 'Clayey soil', the second column shows that 7 instances of 'Clayey soil' were correctly predicted, while one instance of 'Clayey soil' was mistakenly classified as 'Sandy soil' . Meanwhile, Sandy soil was poorly classified. Only 1 out 8 instances were correctly predicted, while 4 of them being incorrectly classified as ‘Loamy Soil’ and 2 being misclassified as ‘Sandy Loam soil’ (See Appendix Image 7 for a comparison of the predicted and actual images).

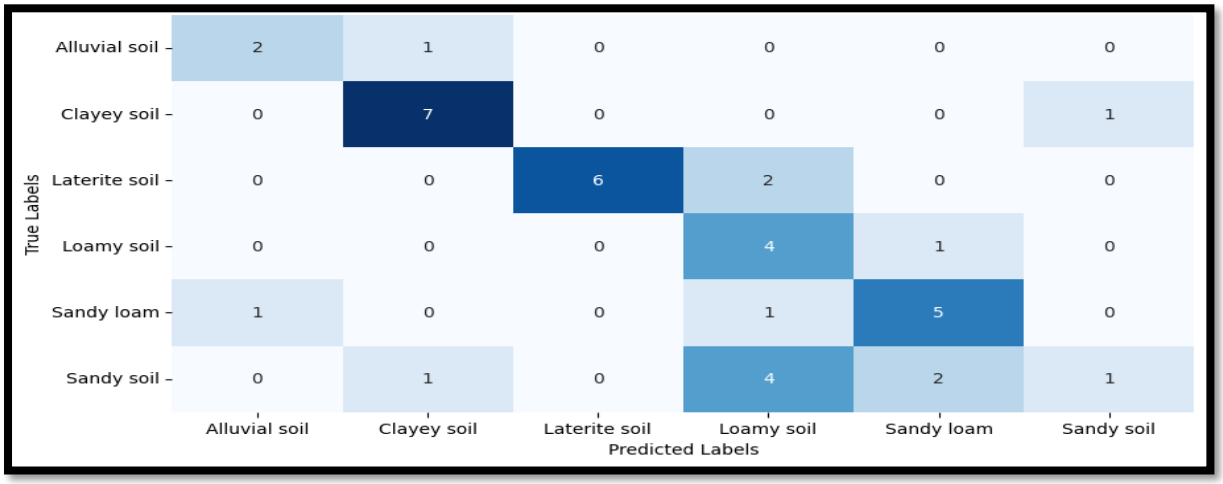
Figure 8

Classification Report for the Tuned Model

	precision	recall	f1-score	support
Alluvial soil	0.67	0.67	0.67	3
Clayey soil	0.78	0.88	0.82	8
Laterite soil	1.00	0.75	0.86	8
Loamy soil	0.36	0.80	0.50	5
Sandy loam	0.62	0.71	0.67	7
Sandy soil	0.50	0.12	0.20	8
accuracy			0.64	39
macro avg	0.66	0.66	0.62	39
weighted avg	0.68	0.64	0.62	39

Figure 9

Heatmap displaying the tuned model’s classification performance



A line chart (Figure 10) was plotted to show the training and test performance of the tuned CNN model over a series of epochs, just like the base model. Using the Early Stop technique, the model stopped training in the 19th epoch when it reached a predetermined validation loss threshold.

Test accuracy is represented by the blue line, which starts off at about 20% and then steeply declines to about 35% around the 17th epoch before stabilizing at 64.10%. The training accuracy is represented by the red line, which begins at about 17.14% and progressively increases to 65.71% by the 13th epoch.

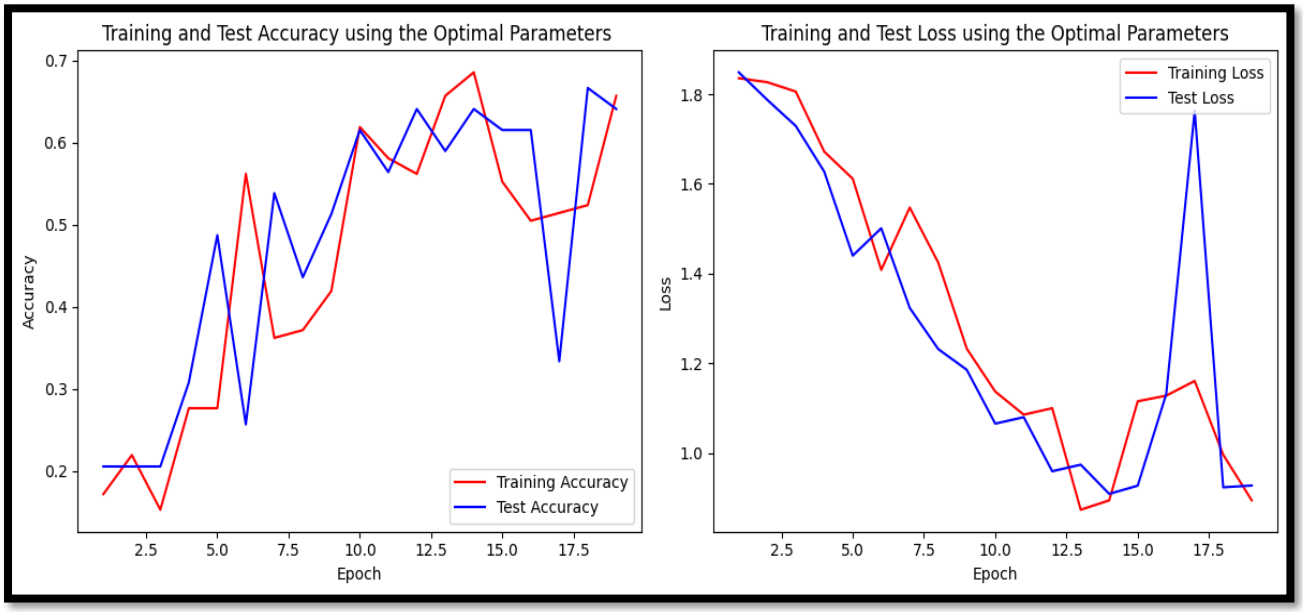
The training loss (represented by the red line) shows a gradual decrease from 1.83 to 0.89 in the line chart on the right, while the test loss (represented by the blue line) fluctuates before settling around 0.927. The test metrics show that the model generalizes well to new sets of data, and these patterns show that the model learns well from the training set of data.

Variations in the test metrics, however, might call for additional research to rule out overfitting or unstable models. Deeper neural network training may be used to address this. Although deeper networks have the potential to perform better, their increased computational complexity makes them more likely to crash apps like Google Colab. As a result, deeper networks also demand better computational resources. It's also important to note that deeper networks might be more prone to overfitting, which would need to be mitigated with careful regularization techniques.

As an additional test, AlexNet, a well-known deep learning model, was also evaluated. However, it resulted in poor accuracy of 23.91% and a test loss of 1.88, indicating potential challenges in adapting deep architectures to this specific dataset. Further investigation and experimentation are warranted to explore the feasibility and effectiveness of using deeper neural networks for this task. (See Appendix Image 8 for model summary).

Figure 10

Comparing the accuracy and loss of training and test of the tuned model



e) Discuss limitations of the proposed model for the classification task.

The proposed CNN model has several limitations that should be considered:

- Firstly, the relatively small dataset used for training and evaluation may limit the model's ability to generalize to unseen data. Small datasets can lead to overfitting, in which the model memorizes training data rather than learning meaningful patterns, resulting in poor performance on new data. Furthermore, the effectiveness of hyperparameter tuning and regularization techniques such as dropout and early stopping can be limited by dataset size and complexity. While these techniques can reduce overfitting to some extent, they may not fully address the challenges posed by small dataset sizes and variability.

- Another limitation lies in the model's reliance on a relatively shallow architecture compared to more complex CNN architectures. Although logical considering the size of data, while simpler architectures offer computational advantages and faster training times, they may limit the model's capacity to learn intricate features and patterns in the data, especially for tasks with high complexity or variability.

- Furthermore, the computational resources required to train and evaluate more complex models may have practical limitations. Limited access to high-performance computing infrastructure may limit the exploration of complex architectures or large hyperparameter search spaces which was the case in this dataset.

To address these limitations, more data must be sourced and techniques like transfer learning can be explored to leverage pretrained models. Furthermore, experimenting with more advanced CNN architectures or ensemble methods could improve model performance. Careful experimental design and interpretation of results are crucial to ensure robust and reliable performance in real-world applications.

V. Conclusion

Soil is critical to the long-term viability of our planet. The classification task performed with convolutional neural networks (CNNs) reveals both successes and limitations. While regularization techniques and hyperparameter tuning improve model performance, issues such as overfitting remain due to the small dataset size. The importance of careful model design and resource allocation is clear, especially given computational constraints. Despite these limitations, the task provides valuable insights into CNN applications for soil classification, as demonstrated by a thorough analysis of performance metrics. Moving forward, overcoming these limitations will necessitate a multifaceted approach that includes dataset augmentation, experimentation with advanced architectures, and resource optimization. Through iterative refinement of methodology and model design, researchers can advance soil classification methods and contribute to broader applications in environmental science and agriculture.

References

[1] Ali Awan, A. (2022, November). *A Complete Guide to Data Augmentation*. Learn Data Science and AI Online | DataCamp. https://www.datacamp.com/tutorial/complete-guide-data-augmentation

[2] Bano, R. (2023, May 30). *Convolution neural networks: All you need to know*. Medium. https://medium.com/@rukaiya.rk24/convolution-neural-networks-all-you-need-to-know-a71fde27e498#:~:text=Parameter%20sharing%20refers%20to%20the,the%20output%20of%20a%20layer

[3] Bardgett, R. D., & van der Putten, W. H. (2014, November 26). *Belowground biodiversity and ecosystem functioning*. https://www.nature.com/articles/nature13855#article-info. https://doi.org/10.1038/nature13855

[4] Bertolet, B. L., Corman, J. R., Casson, N. J., Sebestyen, S. D., Kolka, R. K., & Stanley, E. H. (2018). Influence of soil temperature and moisture on the dissolved carbon, nitrogen, and phosphorus in organic matter entering lake ecosystems. *Biogeochemistry*, *139*(3), 293-305. https://doi.org/10.1007/s10533-018-0469-3

[5] Chollet, F. (2016, June 5). Building powerful image classification models using very little data. *The Keras Blog*. https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

[6] The Class of AI. (2021, March 29). *Multi-class image classification using Keras (Python) on weather dataset (Kaggle)*. YouTube. https://youtu.be/VCHNh3cMsRE

[7] The Class of AI. (2021). *Multiclass_Image_Classification/README.md at main · theclassofai/Multiclass_Image_Classification*. GitHub. https://github.com/theclassofai/Multiclass_Image_Classification/blob/main/README.md

[8] Cloud Factory. (n.d.). *Shear*. CloudFactory Computer Vision Wiki. Retrieved February 28, 2024, from https://wiki.cloudfactory.com/docs/mp-wiki/augmentations/shear

[9] FAO. (2015). *Status of the world's soil resources*. Food and Agriculture Organization of the United Nations. https://www.fao.org/policy-support/tools-and-publications/resources-details/en/c/435200/

[10] Geeks for Geeks. (2023, May 8). *Difference between RGB vs RGBA color format*. GeeksforGeeks. Retrieved February 28, 2024, from https://www.geeksforgeeks.org/difference-between-rgb-vs-rgba-color-format/

[11] Isbell, R. (2016). *The Australian soil classification* (3rd ed.). CSIRO PUBLISHING. https://www.soilscienceaustralia.org.au/asc/soiusing.htm

[12] James Cook University. (n.d.). *Regularization and Optimization Algorthims*. LearnJCU. Retrieved February 28, 2024, from https://learn.jcu.edu.au/

[13] Kayastha, A. M. (2017). *Soil texture based on different composition*. Research Gate. https://www.researchgate.net/figure/A-Morphological-outlook-of-soils-sandy-loamy-and-clay-B-Soil-texture-based-on_fig1_313351187

[14] Kim, B. J., Choi, H., Lee, D., & Kim, S. W. (2023, February 13). *How to Use Dropout Correctly on Residual Networks with Batch Normalization*. arXiv.org e-Print archive. https://arxiv.org/pdf/2302.06112.pdf

[15] Kim, S. (2021, October 11). *Demystifying batch normalization vs drop out*. Medium. https://medium.com/mlearning-ai/demystifying-batch-normalization-vs-drop-out-1c8310d9b516

[16] Li, X., Chen, S., Yang, J., & Hu, X. (2018, January 16). *Understanding the Disharmony between Dropout and Batch Normalization by Variance Shift*. arXiv.org e-Print archive. https://arxiv.org/pdf/1801.05134.pdf

[17] Matshidiso. (n.d.). *Soil types*. Kaggle: Your Machine Learning and Data Science Community. https://www.kaggle.com/datasets/matshidiso/soil-types/data

[18] Saxena, S. (2024, February 20). *Introduction to batch normalization*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/03/introduction-to-batch-normalization/

[19] Singh, A. (2021, August 5). *Hyperparameter tuning of neural networks using Keras tuner*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/08/hyperparameter-tuning-of-neural-networks-using-keras-tuner/

[20] Sue. (2023, March 25). *All you should know about translation equivariance/invariance in CNN*. Medium. https://medium.com/mlearning-ai/all-you-should-know-about-translation-equivariance-invariance-in-cnn-cbf2a2ad33cd#:~:text=Translation%20invariance%20means%20that%20a,or%20translated%20in%20any%20direction

[21] Wu, Y., & Yu, Y. (2023, February 15). *Soil carbon, nitrogen and phosphorus fractions and response to microorganisms and mineral elements in zanthoxylum planispinum 'Dintanensis' plantations at different altitudes*. MDPI. https://www.mdpi.com/2073-4395/13/2/558#:~:text=1.,growth%2C%20development%2C%20and%20metabolism

[22] Yamashita, R., Nishio, M., Gian Do, R. K., & Togashi, K. (2018, June 22). *Convolutional neural networks: An overview and application in radiology*. SpringerOpen. https://insightsimaging.springeropen.com/articles/10.1007/s13244-018-0639-9

[23] Zurich. (2023, December 5). Why soil is important to life on Earth – and helps fight climate change. *Biodiversity*. https://www.zurich.com/en/media/magazine/2021/how-soil-supports-life-on-earth-and-could-help-win-the-fight-against-climate-change#:~:text=Soil%20helps%20produce%20our%20food,stores%20vast%20amounts%20of%20carbon

Appendix

A. Images

Figure 1

Visual Representation of Image Dimensions for Various Soil Types

Figure 2

Comparing the accuracy and loss of training and test using Batch Normalization

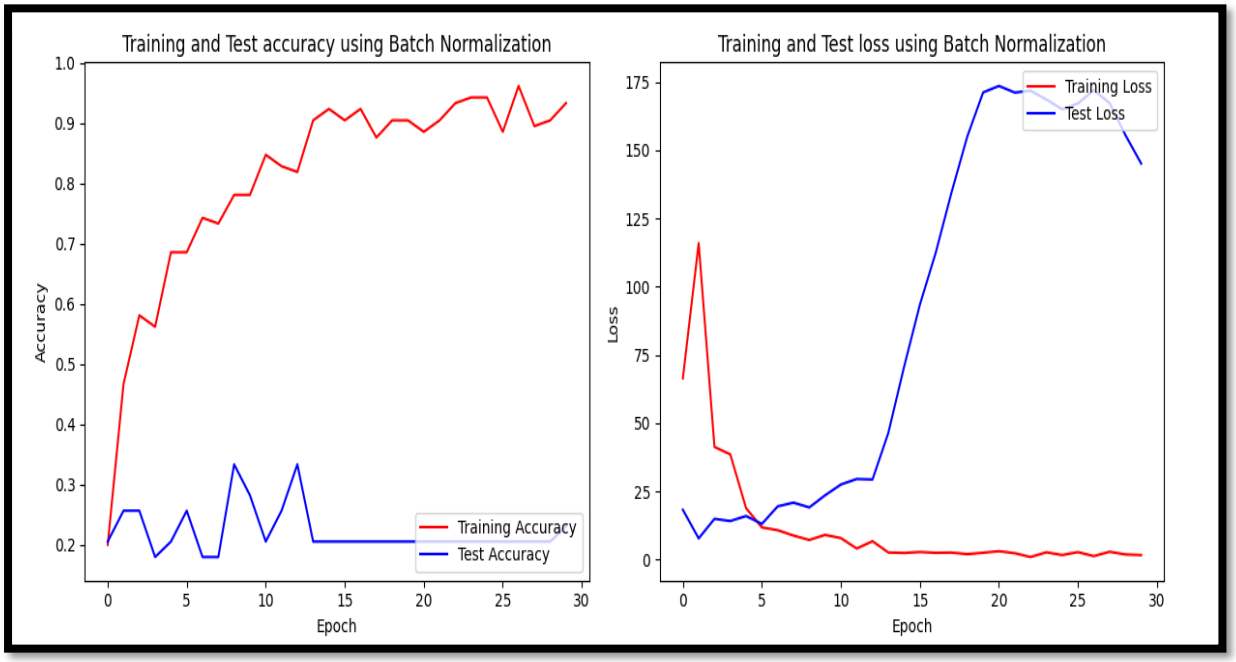


Figure 3

Comparing the accuracy and loss of training and test using Batch Normalization + Early Stop

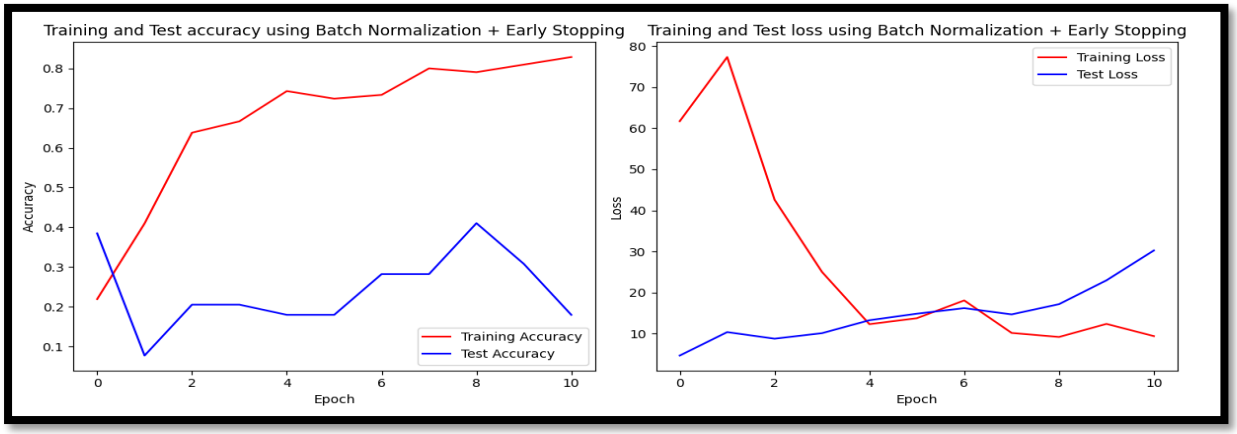


Figure 4

Comparing the accuracy and loss of training and test using Dropout

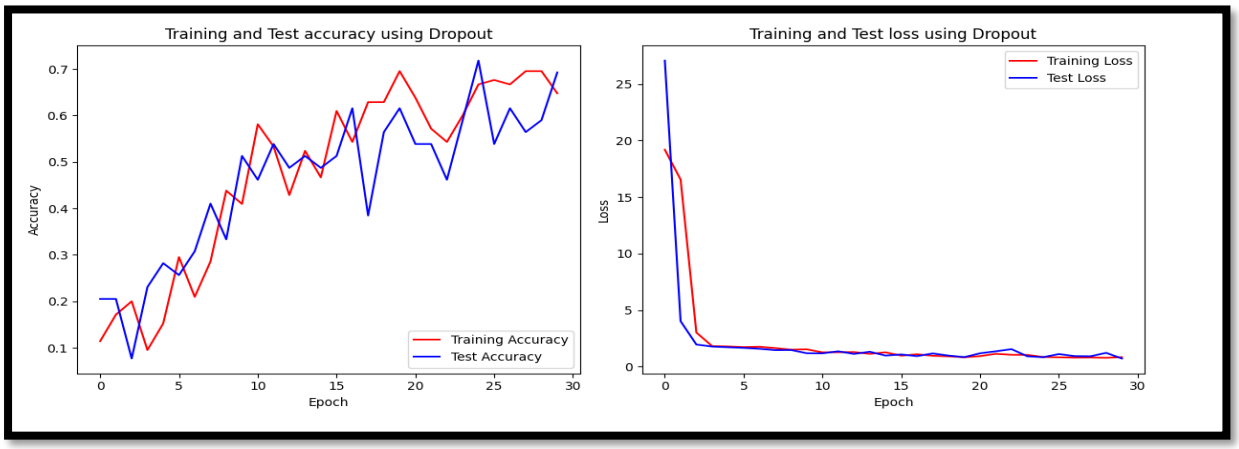


Figure 5

Comparing the accuracy and loss of training and test using Dropout and Early Stop

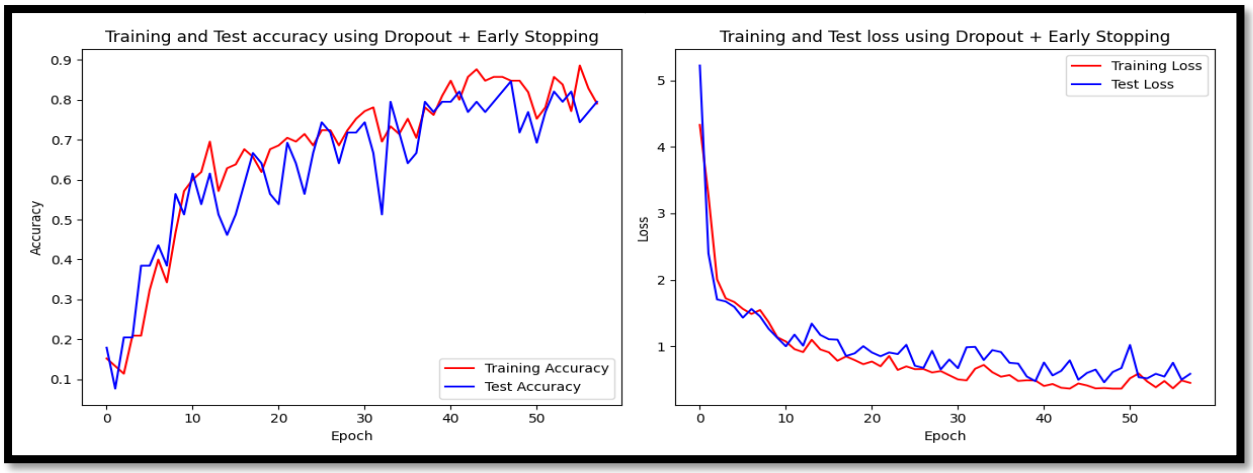


Figure 6

Comparing AlexNet's accuracy and loss for training and test data

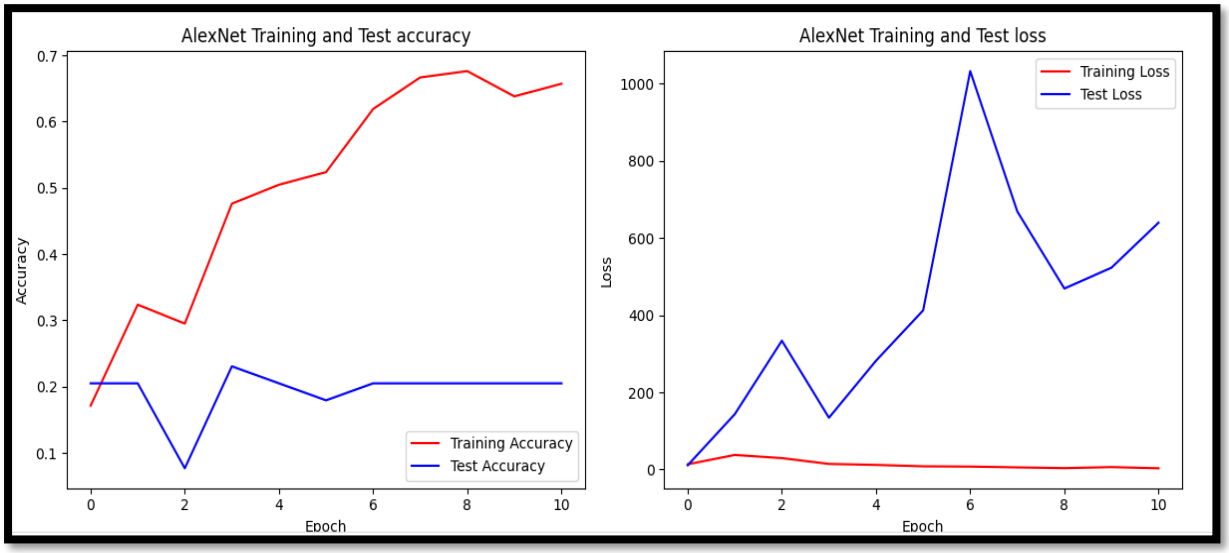
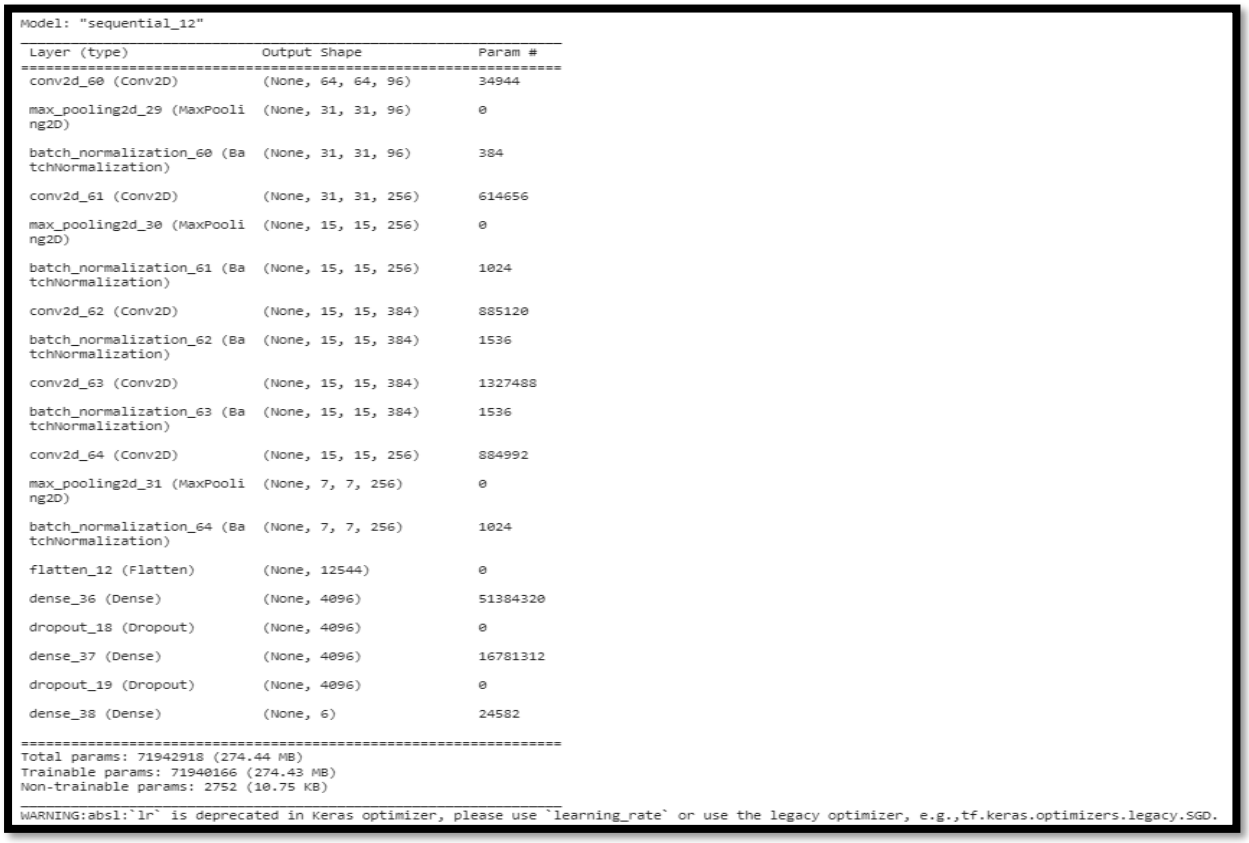


Figure 7

Comparing the Predicted vs Actual Images using the tuned hyperparameters

Figure 8

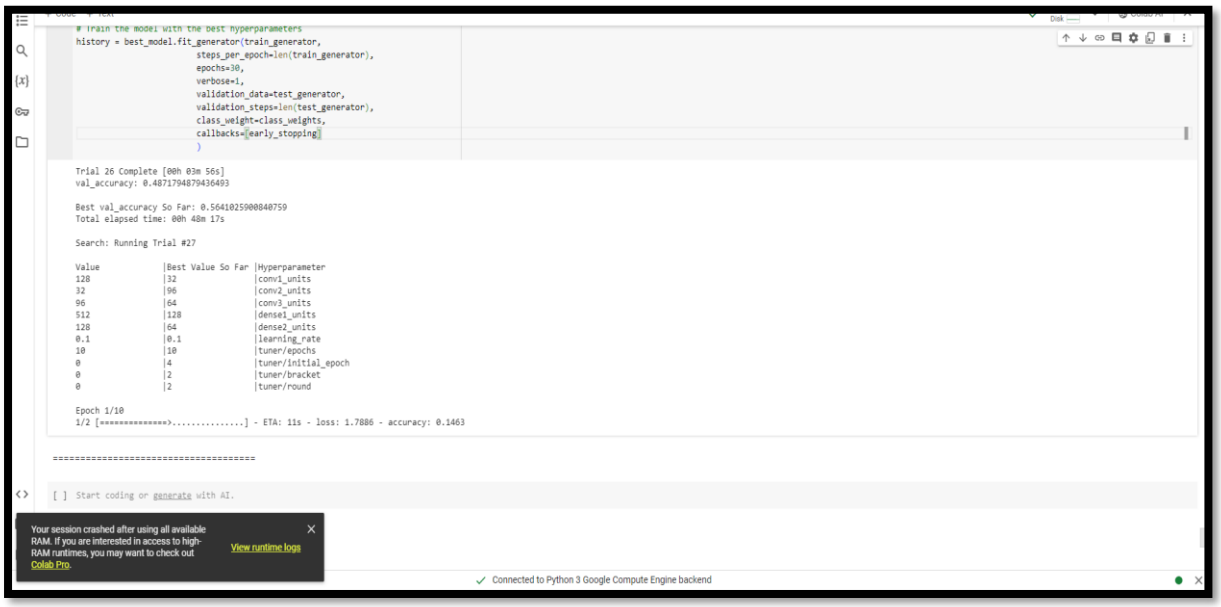
AlexNet model summary



```
Model: "sequential_12"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_60 (Conv2D)          (None, 64, 64, 96)        34944

 max_pooling2d_29 (MaxPooli  (None, 31, 31, 96)        0
 ng2D)

 batch_normalization_60 (Ba  (None, 31, 31, 96)        384
 tchNormalization)

 conv2d_61 (Conv2D)          (None, 31, 31, 256)       614656

 max_pooling2d_30 (MaxPooli  (None, 15, 15, 256)       0
 ng2D)

 batch_normalization_61 (Ba  (None, 15, 15, 256)       1024
 tchNormalization)

 conv2d_62 (Conv2D)          (None, 15, 15, 384)       885120

 batch_normalization_62 (Ba  (None, 15, 15, 384)       1536
 tchNormalization)

 conv2d_63 (Conv2D)          (None, 15, 15, 384)       1327488

 batch_normalization_63 (Ba  (None, 15, 15, 384)       1536
 tchNormalization)

 conv2d_64 (Conv2D)          (None, 15, 15, 256)       884992

 max_pooling2d_31 (MaxPooli  (None, 7, 7, 256)         0
 ng2D)

 batch_normalization_64 (Ba  (None, 7, 7, 256)         1024
 tchNormalization)

 flatten_12 (Flatten)        (None, 12544)             0

 dense_36 (Dense)            (None, 4096)              51384320

 dropout_18 (Dropout)        (None, 4096)              0

 dense_37 (Dense)            (None, 4096)              16781312

 dropout_19 (Dropout)        (None, 4096)              0

 dense_38 (Dense)            (None, 6)                 24582

=================================================================
Total params: 71942918 (274.44 MB)
Trainable params: 71940166 (274.43 MB)
Non-trainable params: 2752 (10.75 KB)
_________________________________________________________________
WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning_rate` or use the legacy optimizer, e.g.,tf.keras.optimizers.legacy.SGD.
```

Figure 9

Session crash after utilizing all available RAM



```
# Train the model with the best hyperparameters
history = best_model.fit_generator(train_generator,
                        steps_per_epoch=len(train_generator),
                        epochs=30,
                        verbose=1,
                        validation_data=test_generator,
                        validation_steps=len(test_generator),
                        class_weight=class_weights,
                        callbacks=[early_stopping]
                        )

Trial 26 Complete [00h 03m 56s]
val_accuracy: 0.4871794879436493

Best val_accuracy So Far: 0.5641025900840759
Total elapsed time: 00h 48m 17s

Search: Running Trial #27

Value             |Best Value So Far |Hyperparameter
128               |32                |conv1_units
32                |96                |conv2_units
96                |64                |conv3_units
512               |128               |dense1_units
128               |64                |dense2_units
0.1               |0.1               |learning_rate
10                |10                |tuner/epochs
0                 |4                 |tuner/initial_epoch
0                 |2                 |tuner/bracket
0                 |2                 |tuner/round

Epoch 1/10
1/2 [==============>.............] - ETA: 11s - loss: 1.7886 - accuracy: 0.1463

====================================

[ ] Start coding or generate with AI.

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.
View runtime logs

Connected to Python 3 Google Compute Engine backend
```

Figure 10

Output summary for the tuned model

```
Trial 30 Complete [00h 12m 37s]
val_accuracy: 0.43589743971824646

Best val_accuracy So Far: 0.5641025900840759
Total elapsed time: 01h 46m 05s
Epoch 1/30
2/2 [==============================] - 35s 15s/step - loss: 1.8357 - accuracy: 0.1714 - val_loss: 1.8488 - val_accuracy: 0.2051
Epoch 2/30
2/2 [==============================] - 33s 20s/step - loss: 1.8269 - accuracy: 0.2190 - val_loss: 1.7875 - val_accuracy: 0.2051
Epoch 3/30
2/2 [==============================] - 32s 15s/step - loss: 1.8060 - accuracy: 0.1524 - val_loss: 1.7292 - val_accuracy: 0.2051
Epoch 4/30
2/2 [==============================] - 32s 19s/step - loss: 1.6720 - accuracy: 0.2762 - val_loss: 1.6273 - val_accuracy: 0.3077
Epoch 5/30
2/2 [==============================] - 33s 14s/step - loss: 1.6114 - accuracy: 0.2762 - val_loss: 1.4401 - val_accuracy: 0.4872
Epoch 6/30
2/2 [==============================] - 32s 20s/step - loss: 1.4082 - accuracy: 0.5619 - val_loss: 1.5011 - val_accuracy: 0.2564
Epoch 7/30
2/2 [==============================] - 32s 20s/step - loss: 1.5473 - accuracy: 0.3619 - val_loss: 1.3235 - val_accuracy: 0.5385
Epoch 8/30
2/2 [==============================] - 32s 20s/step - loss: 1.4245 - accuracy: 0.3714 - val_loss: 1.2317 - val_accuracy: 0.4359
Epoch 9/30
2/2 [==============================] - 30s 12s/step - loss: 1.2328 - accuracy: 0.4190 - val_loss: 1.1855 - val_accuracy: 0.5128
Epoch 10/30
2/2 [==============================] - 32s 19s/step - loss: 1.1370 - accuracy: 0.6190 - val_loss: 1.0654 - val_accuracy: 0.6154
Epoch 11/30
2/2 [==============================] - 30s 12s/step - loss: 1.0854 - accuracy: 0.5810 - val_loss: 1.0798 - val_accuracy: 0.5641
Epoch 12/30
2/2 [==============================] - 32s 15s/step - loss: 1.0999 - accuracy: 0.5619 - val_loss: 0.9592 - val_accuracy: 0.6410
Epoch 13/30
2/2 [==============================] - 30s 12s/step - loss: 0.8734 - accuracy: 0.6571 - val_loss: 0.9739 - val_accuracy: 0.5897
Epoch 14/30
2/2 [==============================] - 32s 20s/step - loss: 0.8943 - accuracy: 0.6857 - val_loss: 0.9087 - val_accuracy: 0.6410
Epoch 15/30
2/2 [==============================] - 33s 20s/step - loss: 1.1152 - accuracy: 0.5524 - val_loss: 0.9270 - val_accuracy: 0.6154
Epoch 16/30
2/2 [==============================] - 32s 20s/step - loss: 1.1278 - accuracy: 0.5048 - val_loss: 1.1311 - val_accuracy: 0.6154
Epoch 17/30
2/2 [==============================] - 34s 12s/step - loss: 1.1605 - accuracy: 0.5143 - val_loss: 1.7632 - val_accuracy: 0.3333
Epoch 18/30
2/2 [==============================] - 32s 19s/step - loss: 0.9961 - accuracy: 0.5238 - val_loss: 0.9234 - val_accuracy: 0.6667
Epoch 19/30
2/2 [==============================] - 32s 19s/step - loss: 0.8945 - accuracy: 0.6571 - val_loss: 0.9273 - val_accuracy: 0.6410
```