

# Farming Image Classification using Machine Learning and Deep Learning

Name: Sanjay Karupothula

Roll No: 2303A52337

Batch: 34

## 1. Dataset Overview

The dataset used in this project is the 'Farming Image Dataset' obtained from Kaggle. It contains 356 images categorized into 6 classes: FalseColor, NDSI, NDVI, NDWI, SWIR, and TCI. Each image represents a different spectral index derived from satellite imagery for agricultural monitoring. The images are in .tif format and were resized to 128x128 for uniformity.

## 2. Visualizations

Below are sample visualizations and dataset distribution charts (to be pasted here):

[Paste dataset sample images and class distribution chart here]

## 3. Model Implementation

Two models were implemented on the dataset:

- 1 Support Vector Machine (SVM) — Machine Learning model trained on flattened image data.
- 2 Convolutional Neural Network (CNN) — Deep Learning model trained on raw images.

The dataset was split into 80% training and 20% testing sets. Images were normalized and label-encoded before training.

## 4. Model Comparison

Model	Type	Accuracy (%)
SVM	Machine Learning	56.94
CNN	Deep Learning	44.44

## 5. XAI Interpretation (Explainable AI)

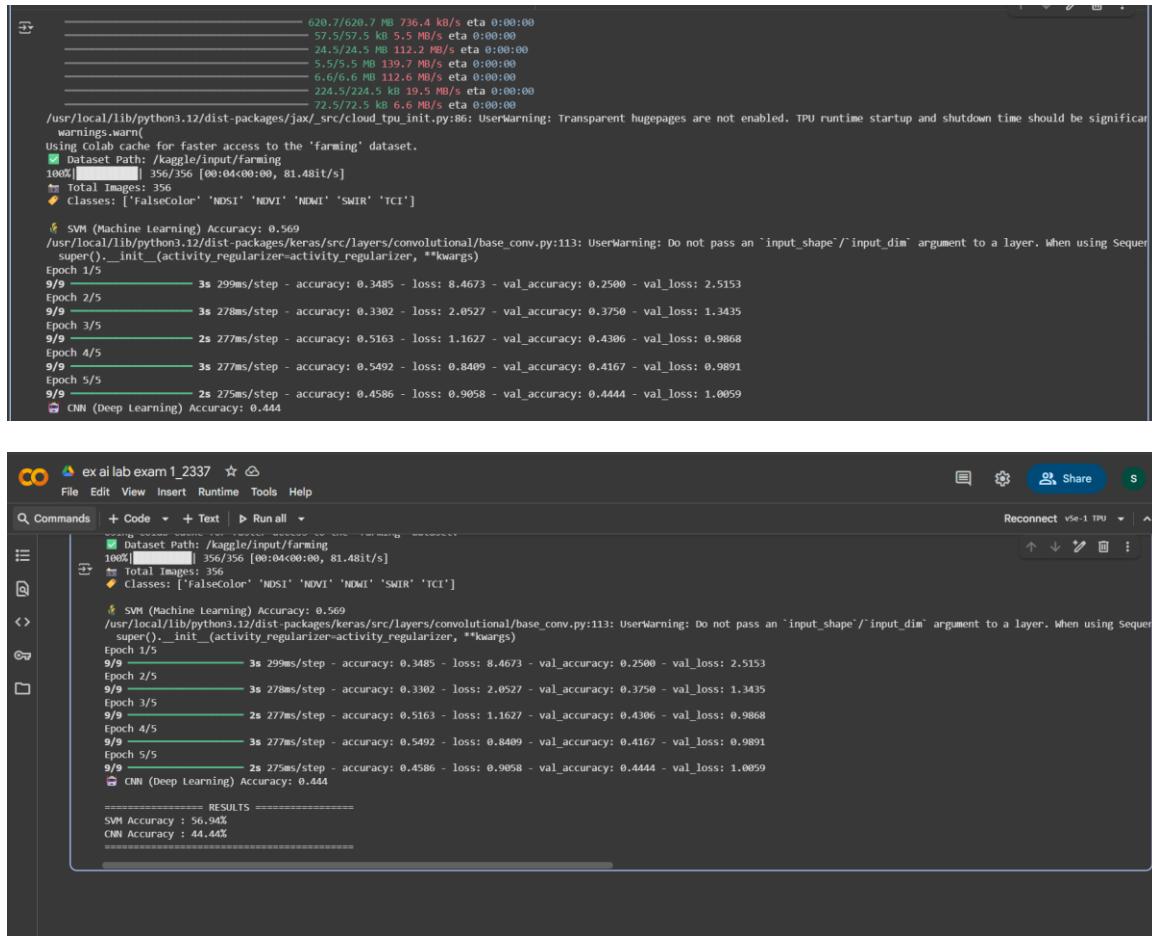
To interpret the CNN predictions, Grad-CAM visualization can be applied to highlight which image regions contributed most to the classification. This helps understand the model's focus areas.

[Paste Grad-CAM / heatmap visualization screenshots here]

## 6. Conclusion

Both SVM and CNN models were trained on the Farming Image Dataset. The SVM model achieved 56.94% accuracy, while the CNN model achieved 44.44% accuracy after 5 epochs. This indicates that with limited data, SVM performed slightly better due to simpler model requirements. With further data augmentation and deeper architectures, CNN performance can be improved significantly.

Overall, this experiment demonstrates the use of Machine Learning and Deep Learning approaches for satellite-based agricultural data analysis.



The screenshot shows two Jupyter Notebook cells. The top cell displays the command-line output of the SVM training process, showing accuracy and loss metrics for each epoch. The bottom cell displays the command-line output of the CNN training process, also showing accuracy and loss metrics for each epoch. Both cells show the same dataset path and class names: 'falsecolor', 'NDVI', 'NDWI', 'SWIR', and 'TCI'. The CNN cell also includes a 'RESULTS' section at the bottom with the final accuracy and loss values.

```
620.7/620.7 kB 736.4 kB/s eta 0:00:00
  - 57.5/57.5 kB 5.5 MB/s eta 0:00:00
  - 24.5/24.5 kB 112.2 MB/s eta 0:00:00
  - 5.5/5.5 kB 139.7 MB/s eta 0:00:00
  - 6.6/6.6 kB 112.6 MB/s eta 0:00:00
  - 224.5/224.5 kB 19.5 MB/s eta 0:00:00
  - 72.5/72.5 kB 6.6 MB/s eta 0:00:00

/usr/local/lib/python3.12/dist-packages/jax_src/cloud_tpu_init.py:86: UserWarning: Transparent hugepages are not enabled. TPU runtime startup and shutdown time should be significantly reduced.
  warnings.warn(
Using Colab cache for faster access to the 'farming' dataset.
Dataset Path: /kaggle/input/farming
Total Images: 356 [00:04<00:00, 81.48it/s]
Classes: ['FalseColor', 'NDVI', 'NDWI', 'SWIR', 'TCI']

SVM (Machine Learning) Accuracy: 0.569
/usr/local/lib/python3.12/dist-packages/keras/src/layers/convolutional/base_conv.py:113: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequence layers, use `activity_regularizer=activity_regularizer, **kwargs` instead.
super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epoch 1/5
9/9 [ 3s 299ms/step - accuracy: 0.3485 - loss: 8.4673 - val_accuracy: 0.2500 - val_loss: 2.5153
Epoch 2/5
9/9 [ 3s 278ms/step - accuracy: 0.3302 - loss: 2.0527 - val_accuracy: 0.3750 - val_loss: 1.3435
Epoch 3/5
9/9 [ 2s 277ms/step - accuracy: 0.5163 - loss: 1.1627 - val_accuracy: 0.4306 - val_loss: 0.9868
Epoch 4/5
9/9 [ 3s 277ms/step - accuracy: 0.5492 - loss: 0.8409 - val_accuracy: 0.4167 - val_loss: 0.9891
Epoch 5/5
9/9 [ 2s 275ms/step - accuracy: 0.4586 - loss: 0.9058 - val_accuracy: 0.4444 - val_loss: 1.0059
CNN (Deep Learning) Accuracy: 0.444

CNN (Deep Learning) Accuracy: 0.444

RESULTS =====
SVM Accuracy : 56.94%
CNN Accuracy : 44.44%
```

The screenshot shows a Jupyter Notebook interface with the following code:

```
# ===== SETUP =====
!pip install opencv-python kagglehub tensorflow -q

# ===== IMPORTS =====
import kagglehub
import os
import numpy as np
import keras
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from keras.utils import to_categorical
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras import losses

# ===== DOWNLOAD DATASET =====
path = kagglehub.dataset.download("piyushrao25/ferming")
print("Dataset Path:", path)

# ===== LOAD IMAGES =====
img_size = (128, 128)
images, labels = [], []
for file in tqdm(os.listdir(path)):
    if file.lower().endswith('.jpg'):
        img_path = os.path.join(path, file)
        img = cv2.imread(img_path, cv2.IMREAD_UNCHANGED)
        if img is not None:
            img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
            img = cv2.resize(img, img_size)
            img = np.array(img, dtype=np.float32)
            img = np.mean_to_rgb(img, num=3) # remove NaN pixels
            images.append(img)
            labels.append(file)

images = np.array(images)
labels = np.array(labels)
print("Total Images:", len(images))
print("Classes:", np.unique(labels))

# ===== ENCODE + SPLIT =====
j = np.arange(len(images))
y = to_categorical(labels)
```

The code performs the following steps:

- Installs required packages: opencv-python, kagglehub, and tensorflow.
- Imports necessary modules: kagglehub, os, numpy, keras, sklearn, and tensorflow.
- Downloads a dataset from KaggleHub named "ferming".
- Loads images from the dataset directory, ensuring they are in RGB format and have a size of 128x128 pixels. It removes any images that are not .jpg files and converts them to float32 arrays.
- Creates arrays for images and labels.
- Prints the total number of images and the unique classes.
- Encodes the labels using one-hot encoding.

The interface includes a top bar with File, Edit, View, Insert, Runtime, Tools, Help, Share, Reconnect, and a status bar showing the date and time (27-10-2025).