Question 2 Report

Project Members:

- Shaaf Salman (ID: 21L6083)
- Abdul Hadi (ID: 21L6077)
- Haider Khan (ID: 21L6067)

Course Details

- Applied Artificial Intelligence
- Professor : Kashif Zafar
- Assignment 2
- Question 2

Image Classification Report: Convolutional Neural Network (CNN) Approach

1- Introduction

We were basically tasked with solving a "Weed Detection and Classification" problem. For this purpose, we proposed developing a neural system capable of distinguishing between weed and crop images, considering the limitations of the dataset. Furthermore, we used CNN and had a success rate of approximately 86 %.

2- Dataset

The dataset was obtained from a GitHub repository link (https://github.com/ravirajsinh45/Crop_and_weed_detection.git) and comprised approximately 1300 images, each followed by an annotation file. These images were categorized into two classes: crop and weed (labelled as 0 and 1, respectively). The annotations also included additional relevant image information such as size, although this was not utilized in the classification process.

2.1 -Breakdown of the Dataset

A class file, along with image and annotation lists, was created for data handling.

```
Number of images: 1300
Number of annotations: 1300
Sample image path: data\agri_0_1009.jpeg
Sample annotation: 1 0.608398 0.498047 0.541016 0.531250
```

2.2- Pre-processing of Images

During pre-processing, the following tasks were performed:

- Conversion of images from BGR to RGB. as by default cv converter has BGR format.
- Resizing images to 224×224 for improved space complexity.
- Normalization of pixel values to the range of 0 to 1 to remove any order relevance.
- Conversion of normal lists to NumPy arrays for improved space complexity.

2.3- Split of Dataset

The dataset was split using the default test size of 0.2, resulting in an 80-20 split for training and validation, respectively.

```
Training images shape: (1040, 224, 224, 3)
Training labels shape: (1040,)
Validation images shape: (260, 224, 224, 3)
Validation labels shape: (260,)
```

3- Model Architecture

The CNN architecture utilized for image classification comprised multiple layers:

1. **Input Layer**: Accepting input images of fixed dimensions (e.g., 224×224 pixels).
2. **Convolutional Layers**: A stack of three convolutional layers with increasing depth (32, 64, and 128) to capture spatial features, utilizing a 3×3 kernel size and ReLU activation function.
3. **Max Pooling Layers**: Down sampling layers using a 2×2 matrix to reduce spatial dimensions and extract key features.
4. **Flatten Layer**: Flattening the output from the convolutional layers into a 1D array.
5. **Dense Layers**: Fully connected layers for classification, consisting of 128 nodes with ReLU activation.
6. **Output Layer**: The final layer with SoftMax activation for multi-class classification.

Layer (type)	Output Shape	Param #
conv2d_33 (Conv2D)	(None, 222, 222, 32)	896
max_pooling2d_33 (MaxPooling2D)	(None, 111, 111, 32)	0
conv2d_34 (Conv2D)	(None, 109, 109, 64)	18,496
max_pooling2d_34 (MaxPooling2D)	(None, 54, 54, 64)	0
conv2d_35 (Conv2D)	(None, 52, 52, 128)	73,856
max_pooling2d_35 (MaxPooling2D)	(None, 26, 26, 128)	0
flatten_13 (Flatten)	(None, 86528)	0
dense_26 (Dense)	(None, 128)	11,075,712
dense_27 (Dense)	(None, 2)	258

Total params: 11,169,218 (42.61 MB)
Trainable params: 11,169,218 (42.61 MB)
Non-trainable params: 0 (0.00 B)

4- Model Training

The CNN model was trained on the training set using the following configurations:

- **Optimizer:** We used a method called Adam to make the model learn efficiently. It helped the model find the best settings for its 'brain' by adjusting how much it learns from each example. we used it as for default and kept most of the training parameters as it as they were in other models.
- **Learning Rate:** We set how much the model can learn from each example to a value of 0.001. This was like finding a balance between learning quickly and not making mistakes.
- **Loss Function:** We used a method called Sparse Categorical Cross-Entropy to see how much the model's guesses differed from the actual answers. This helped the model get better at picking the right answers.
- **Epochs:** The model looked passed the dataset 10 times to learn from it.

```python
model.compile(optimizer='adam',
              # Use sparse categorical crossentropy for integer labels
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

```
[74]: # Train the model
history = model.fit(X_train, y_train, epochs=10, validation_data=(X_val, y_val), verbose=1)

Epoch 1/10
33/33 ━━━━━━━━━━━━━━━━━━━━ 21s 600ms/step - accuracy: 0.5397 - loss: 1.1516 - val_accuracy: 0.5923 - val_loss: 0.7822
Epoch 2/10
33/33 ━━━━━━━━━━━━━━━━━━━━ 20s 609ms/step - accuracy: 0.7684 - loss: 0.5465 - val_accuracy: 0.8385 - val_loss: 0.4891
Epoch 3/10
33/33 ━━━━━━━━━━━━━━━━━━━━ 19s 563ms/step - accuracy: 0.8652 - loss: 0.3725 - val_accuracy: 0.8731 - val_loss: 0.4302
Epoch 4/10
33/33 ━━━━━━━━━━━━━━━━━━━━ 18s 560ms/step - accuracy: 0.8374 - loss: 0.3957 - val_accuracy: 0.8654 - val_loss: 0.3806
Epoch 5/10
33/33 ━━━━━━━━━━━━━━━━━━━━ 18s 557ms/step - accuracy: 0.8677 - loss: 0.3769 - val_accuracy: 0.8885 - val_loss: 0.3795
Epoch 6/10
33/33 ━━━━━━━━━━━━━━━━━━━━ 19s 567ms/step - accuracy: 0.8923 - loss: 0.3102 - val_accuracy: 0.8808 - val_loss: 0.3435
Epoch 7/10
33/33 ━━━━━━━━━━━━━━━━━━━━ 19s 575ms/step - accuracy: 0.8863 - loss: 0.3174 - val_accuracy: 0.8769 - val_loss: 0.3612
Epoch 8/10
33/33 ━━━━━━━━━━━━━━━━━━━━ 20s 592ms/step - accuracy: 0.9034 - loss: 0.2743 - val_accuracy: 0.8885 - val_loss: 0.3101
Epoch 9/10
33/33 ━━━━━━━━━━━━━━━━━━━━ 20s 616ms/step - accuracy: 0.9134 - loss: 0.2353 - val_accuracy: 0.9000 - val_loss: 0.2957
Epoch 10/10
33/33 ━━━━━━━━━━━━━━━━━━━━ 20s 608ms/step - accuracy: 0.9255 - loss: 0.2037 - val_accuracy: 0.8923 - val_loss: 0.3060
```

The model's performance was monitored using the validation set to prevent overfitting, with early stopping based on validation loss.

5- Evaluation

Following training, the CNN model was evaluated on the test set to assess its generalization performance.

Step 9: Evaluate the Model

: This cell evaluates the trained CNN model on the validation data.

Explanation:

- The model is evaluated on the validation data using the `evaluate` function.
- Validation loss and accuracy are printed to assess the model's performance.

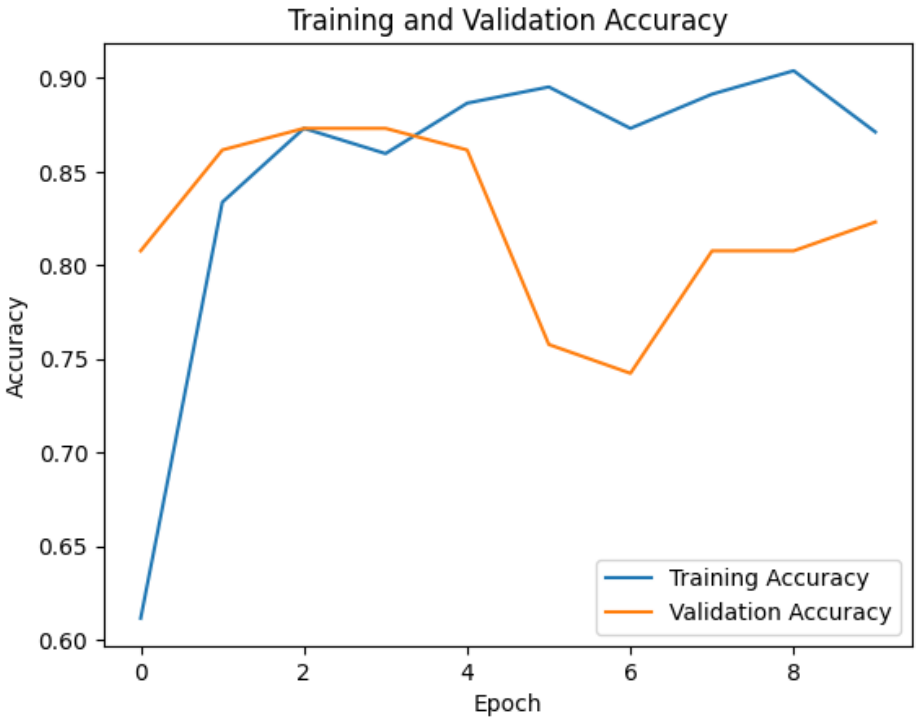
```
[81]: # Evaluate the model on validation data
val_loss, val_accuracy = model.evaluate(X_val, y_val, verbose=0)
print("Validation Loss:", val_loss)
print("Validation Accuracy:", val_accuracy)

Validation Loss: 0.3059946894645691
Validation Accuracy: 0.892307698726654
```

5.1 Results

The CNN model demonstrated promising performance in classifying images of weeds and crops. The following results were obtained:

- **Validation Accuracy**: 89%
- **Validation Accuracy**: 30%



6- Further Testing

After training, the model was tested on high-quality images sourced from Pinterest and Google. These new testing data files had different Environment and resolution as compared to the default dataset. Our Model showed favourable results with a few exceptions discussed later.

6.1- Analysis

it was noticed that the model was differing between the crop and weed on the basis of the cutout of the leaves as per my observation. If a normal crop has very deteriorated cut-out of leaves it will distinguish it as a weed but for most of the approximation images it detects the weed successfully.

Conclusion

In conclusion, the CNN approach proved highly effective for the task of weed and crop classification in agricultural images. The model's performance on both the validation and test sets demonstrates its capability to generalize well to unseen data.