SECTION 2

A Scientific report to identify the species of flowers from a photograph using CNN models and hyperparameters.

ABSTRACT

The purpose of this report is to develop a CNN model that will identify the species of flowers from different photographs using the dataset from TensorFlow.

Keywords: CNN, Models, Neural networks, Normalisation, Keras, Metrics, Kernel

1. INTRODUCTION

Flower recognition with CNN is an emerging field in computer vision. Its aim is to create a model that will be able accurately identify various types of flowers, based on their distinctive visual features. This is done by providing the CNN model with a huge set of labelled images showing various flower species so that it can be able to learn patterns and features characteristic only for these flowers. This report also aims to classify flower species using various CNN techniques and hyperparameter to improve its performance and classification accuracy.

2. RELATED WORKS

The identification of flowers via CNN models is attracting much attention as many challenges are faced in flower classification because Image recognition vectorization faces issues that possess a high variety of shapes colour distribution, lighting conditions or deformation of exposure (Seeland et al., 2017; Hanafiah et al., 2022). Different studies indicate the strong capability of deep convolutional neural networks (CNN) to determine plant species from flower images automatically (Le et al., 2016; Hiary et al., 2018). The literature review of different Artificial Neural Network (ANN) classifiers that have been used to categorize flowers will offer knowledge on the possibilities of CNNs and their contribution towards fostering computer vision technology (BOZKURT, 2022). Using CNN models to identify flowers is an exciting and fast-growing area of research in computer vision, as the amount of available large datasets that are diverse grows and as deep learning models increase their power sophistication (BOZKURT, 2022).

3.0 METHODOLOGY

3.1 Data Description

The dataset used for this report was gotten from the keras library and contains 3670 colour photographs of flowers, consisting of five different species: Daisy, Dandelion, Roses, Sunflowers, Tulips. The data was then loaded and pulled from a folder directory and then split into train, test and validation.

3.2 CNN ARCHITECTURE

The architecture of the CNN models is broken down into these categories:

3.2.1 Data Pre-Processing and Data Augmentation

The dataset has an input shape of (256,256,3) showing the height, width and the rgb colour type (3). The test and trained data were converted categorically using One Hot Encoding then normalized between 0 and 1. The data was Augmented by use of rotation range, flipping range, zoom, shear range and validation split. Image was then fitted into the trained dataset.

3.2.2 Hyper Parameter Metrics

The parameter metrics used in this model are the: 3 Convolutional layers (32, 64, 128), 3 Pooling Layer (2 x 2), 1 Flatten Layer, 1 Dense Layer (64) with 'relu' activation mode, 50% Dropout rate and 1 Output Dense Layer (5) with 'softmax' activation mode. Max Pooling2D, AveragePooling2D was also used so the model can focus on the most significant features in the images. A batch size of 32 and 128 was used to train this model with different parameters to help improve the performance of the model. 20 epochs were used to help the training process of the model. Adam optimiser of 0.001 was used for all models as the learning rate as this could enhance a better model performance. Stride length of (2,2), padding, and two kernel sizes of (3,3) and (5,5) were to check the performance levels of the models.

3.2.3 Regularization Method

Dropout rate: A dropout rate of (0.5)50% was added to manage overfitting ensuring the model performs well towards a new image input which helped in the accuracy of the model.

Batch Normalization: this was applied the pooling layers to normalize the output. This normalization improved the accuracy of the CNN model.

3.3 Visualization

Training, validation loss and accuracy graphs were plotted to observe and visualise overfitting, A confusion metrics was also plotted to see the accurate prediction correlation between the flowers for each model to see each model performance.

3.4 Evaluation Metrics

The accuracy was primarily used to evaluate each model performance using the classification report technique.

4.0 RESULTS

4.1 CNN Architecture

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 254, 254, 32)	896
max_pooling2d (MaxPooling2D)	(None, 127, 127, 32)	0
conv2d_1 (Conv2D)	(None, 125, 125, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 62, 62, 64)	0
conv2d_2 (Conv2D)	(None, 60, 60, 128)	73856
max_pooling2d_2 (MaxPooling2D)	(None, 30, 30, 128)	0
flatten (Flatten)	(None, 115200)	0
dense (Dense)	(None, 64)	7372864
dropout (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 5)	325

Total params: 7466437 (28.48 MB)
Trainable params: 7466437 (28.48 MB)
Non-trainable params: 0 (0.00 Byte)

Figure 1.0 CNN Architecture (Model 1)

Model: "sequential_2"

Layer (type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 128, 128, 32)	2432
max_pooling2d_6 (MaxPooling2D)	(None, 64, 64, 32)	0
conv2d_7 (Conv2D)	(None, 32, 32, 64)	51264
max_pooling2d_7 (MaxPooling2D)	(None, 16, 16, 64)	0
conv2d_8 (Conv2D)	(None, 8, 8, 128)	204928
max_pooling2d_8 (MaxPooling2D)	(None, 4, 4, 128)	0
flatten_2 (Flatten)	(None, 2048)	0
dense_4 (Dense)	(None, 64)	131136
dropout_2 (Dropout)	(None, 64)	0
dense_5 (Dense)	(None, 5)	325

Total params: 390085 (1.49 MB)
Trainable params: 390085 (1.49 MB)
Non-trainable params: 0 (0.00 Byte)

Figure 1.2 CNN Architecture (Model 2)

Model: "sequential_3"

Layer (type)	Output Shape	Param #
conv2d_9 (Conv2D)	(None, 128, 128, 32)	896
average_pooling2d (AveragePooling2D)	(None, 64, 64, 32)	0
conv2d_10 (Conv2D)	(None, 32, 32, 64)	18496
average_pooling2d_1 (AveragePooling2D)	(None, 16, 16, 64)	0
conv2d_11 (Conv2D)	(None, 8, 8, 128)	73856
average_pooling2d_2 (AveragePooling2D)	(None, 4, 4, 128)	0
flatten_3 (Flatten)	(None, 2048)	0
dense_6 (Dense)	(None, 64)	131136
dropout_3 (Dropout)	(None, 64)	0
dense_7 (Dense)	(None, 5)	325

Total params: 224709 (877.77 KB)
Trainable params: 224709 (877.77 KB)
Non-trainable params: 0 (0.00 Byte)

Figure 1.3 CNN Architecture (Model 3)

Model: "sequential_4"

Layer (type)	Output Shape	Param #
conv2d_12 (Conv2D)	(None, 128, 128, 32)	896
average_pooling2d_3 (AveragePooling2D)	(None, 64, 64, 32)	0
batch_normalization (BatchNormalization)	(None, 64, 64, 32)	128
conv2d_13 (Conv2D)	(None, 32, 32, 64)	18496
average_pooling2d_4 (AveragePooling2D)	(None, 16, 16, 64)	0
batch_normalization_1 (BatchNormalization)	(None, 16, 16, 64)	256
conv2d_14 (Conv2D)	(None, 8, 8, 128)	73856
average_pooling2d_5 (AveragePooling2D)	(None, 4, 4, 128)	0
batch_normalization_2 (BatchNormalization)	(None, 4, 4, 128)	512
flatten_4 (Flatten)	(None, 2048)	0
dense_8 (Dense)	(None, 64)	131136
dropout_4 (Dropout)	(None, 64)	0
dense_9 (Dense)	(None, 5)	325

Total params: 225605 (881.27 KB)
Trainable params: 225157 (879.52 KB)
Non-trainable params: 448 (1.75 KB)

Figure 1.4 CNN Architecture (Model 4)

Model: "sequential_5"

Layer (type)	Output Shape	Param #
conv2d_15 (Conv2D)	(None, 252, 252, 32)	2432
max_pooling2d_9 (MaxPooling2D)	(None, 126, 126, 32)	0
conv2d_16 (Conv2D)	(None, 122, 122, 64)	51264
max_pooling2d_10 (MaxPooling2D)	(None, 61, 61, 64)	0
conv2d_17 (Conv2D)	(None, 57, 57, 128)	204928
max_pooling2d_11 (MaxPooling2D)	(None, 28, 28, 128)	0
flatten_5 (Flatten)	(None, 100352)	0
dense_10 (Dense)	(None, 64)	6422592
dropout_5 (Dropout)	(None, 64)	0
dense_11 (Dense)	(None, 5)	325

Total params: 6681541 (25.49 MB)
Trainable params: 6681541 (25.49 MB)
Non-trainable params: 0 (0.00 Byte)

Figure 1.5 CNN Architecture (model 5)

Model: "sequential_6"

Layer (type)	Output Shape	Param #
conv2d_18 (Conv2D)	(None, 254, 254, 32)	896
max_pooling2d_12 (MaxPooling2D)	(None, 127, 127, 32)	0
batch_normalization_3 (BatchNormalization)	(None, 127, 127, 32)	128
conv2d_19 (Conv2D)	(None, 125, 125, 64)	18496
max_pooling2d_13 (MaxPooling2D)	(None, 62, 62, 64)	0
batch_normalization_4 (BatchNormalization)	(None, 62, 62, 64)	256
conv2d_20 (Conv2D)	(None, 60, 60, 128)	73856
max_pooling2d_14 (MaxPooling2D)	(None, 30, 30, 128)	0
batch_normalization_5 (BatchNormalization)	(None, 30, 30, 128)	512
flatten_6 (Flatten)	(None, 115200)	0
dense_12 (Dense)	(None, 64)	7372864
dropout_6 (Dropout)	(None, 64)	0
dense_13 (Dense)	(None, 5)	325

Total params: 7467333 (28.49 MB)
Trainable params: 7466885 (28.48 MB)
Non-trainable params: 448 (1.75 KB)

Figure 1.6 CNN Architecture (model 6)

4.2 CNN Model

4.2.1 Model 1(Normal Architecture)

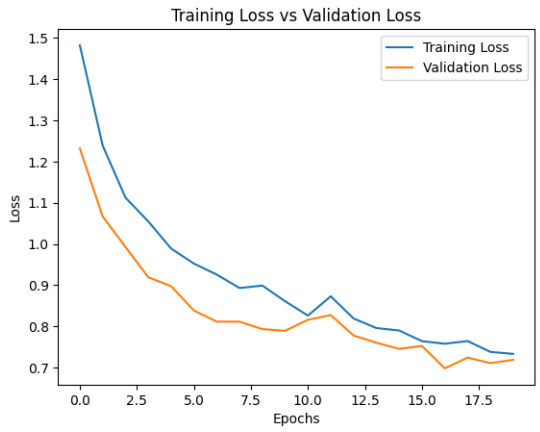

Figure 1.7 CNN Loss Plot (Model 1)

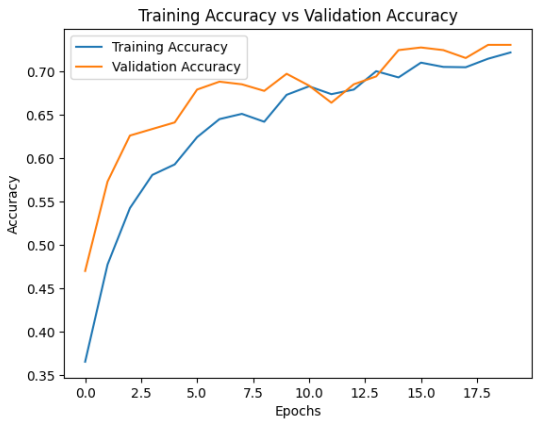

Figure 1.8 CNN Accuracy Plot (Model 1)

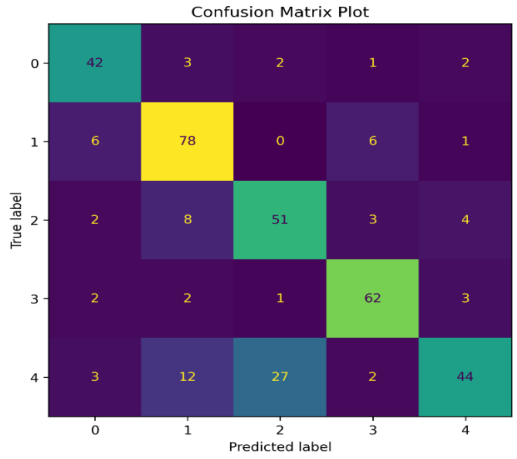

Figure 1.9 Confusion Metrics (Model 1)

	precision	recall	f1-score	support
0	0.76	0.84	0.80	50
1	0.76	0.86	0.80	91
2	0.63	0.75	0.68	68
3	0.84	0.89	0.86	70
4	0.81	0.50	0.62	88
accuracy			0.75	367
macro avg	0.76	0.77	0.75	367
weighted avg	0.76	0.75	0.75	367

Key: 0 = Daisy, 1 = Dandelion, 2 = Roses

3 = Sunflowers, 4 = Tulips

Figure 2.0 Classification Report (Model 1)

4.2.2 Model 2(Batch 128)

Figure 2.1 CNN Loss Plot (Model 1)

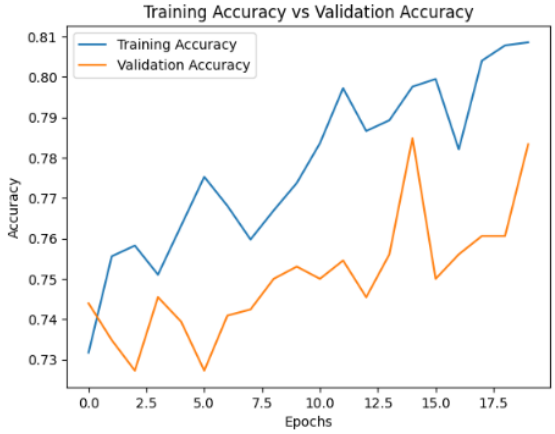

Figure 2.2 CNN Accuracy Plot (Model 1)

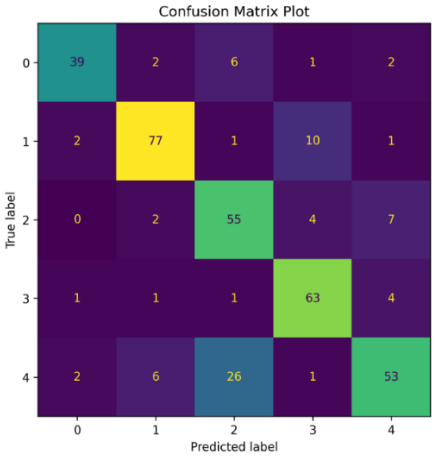

Figure 2.3 Confusion Metrics (Model 2)

	precision	recall	f1-score	support
0	0.89	0.78	0.83	50
1	0.88	0.85	0.86	91
2	0.62	0.81	0.70	68
3	0.80	0.90	0.85	70
4	0.79	0.60	0.68	88
accuracy			0.78	367
macro avg	0.79	0.79	0.78	367
weighted avg	0.79	0.78	0.78	367

Key: 0 = Daisy, 1 = Dandelion, 2 = Roses

3 = Sunflowers, 4 = Tulips

Figure 2.4 Classification Report (Model 2)

4.2.3 Model 3(Stride 2)

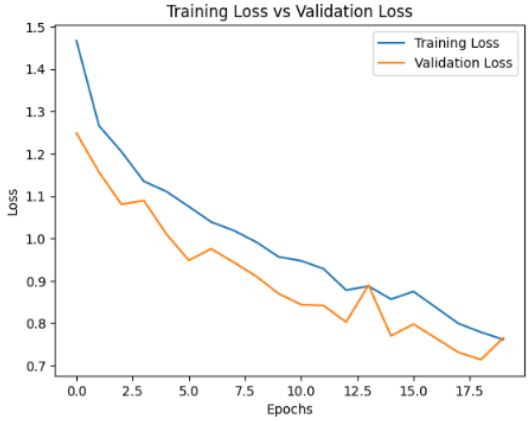

Figure 2.5 CNN Loss Plot (Model 3)

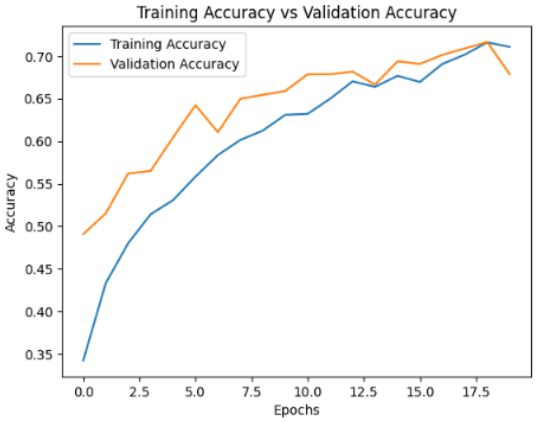

Figure 2.6 CNN Accuracy Plot (Model 3)

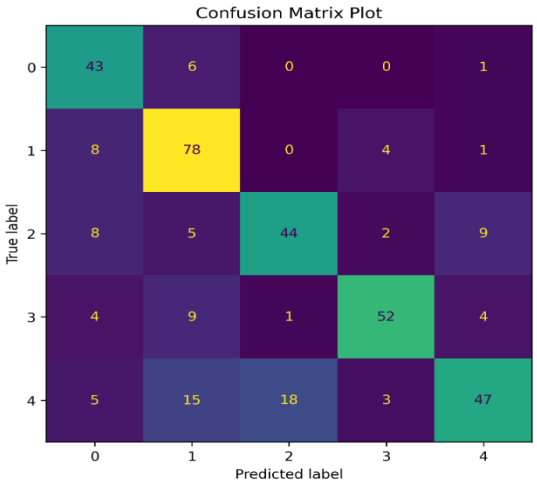

Figure 2.7 Confusion Metrics (Model 3)

	precision	recall	f1-score	support
0	0.63	0.86	0.73	50
1	0.69	0.86	0.76	91
2	0.70	0.65	0.67	68
3	0.85	0.74	0.79	70
4	0.76	0.53	0.63	88
accuracy			0.72	367
macro avg	0.73	0.73	0.72	367
weighted avg	0.73	0.72	0.72	367

Key: 0 = Daisy, 1 = Dandelion, 2 = Roses

3 = Sunflowers, 4 = Tulips

Figure 2.8 Classification Report (Model 3)

4.2.4 Model 4(Kenel 5X5)

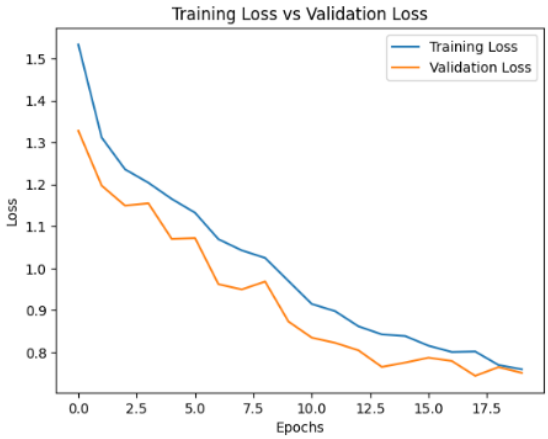

Figure 2.9 CNN Loss Plot (Model 4)

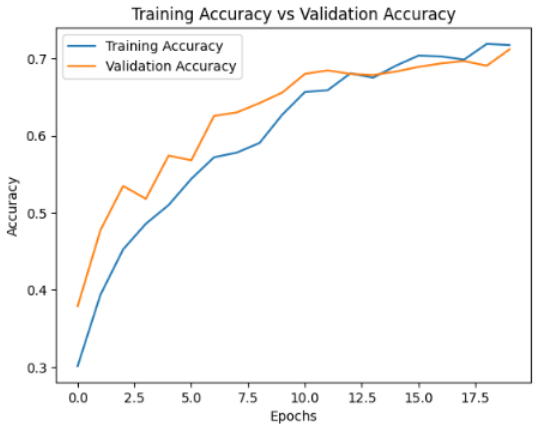

Figure 3.0 CNN Accuracy Plot (Model 4)

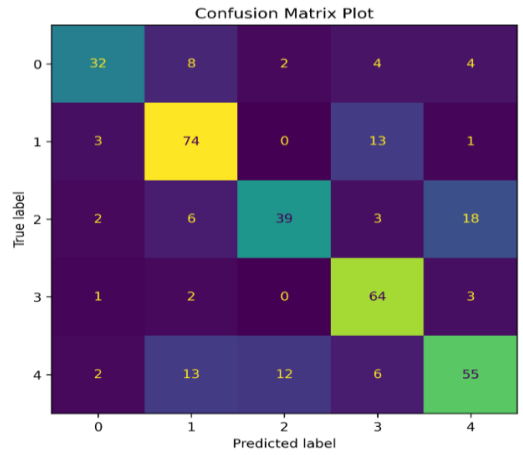

Figure 3.1 Confusion Metrics (Model 4)

	precision	recall	f1-score	support
0	0.80	0.64	0.71	50
1	0.72	0.81	0.76	91
2	0.74	0.57	0.64	68
3	0.71	0.91	0.80	70
4	0.68	0.62	0.65	88
accuracy			0.72	367
macro avg	0.73	0.71	0.71	367
weighted avg	0.72	0.72	0.71	367

Key: 0 = Daisy, 1 = Dandelion, 2 = Roses

3 = Sunflowers, 4 = Tulips

Figure 3.2 Classification Report (Model 4)

4.2.5 Model 5(Average pooling)

Figure 3.3 CNN Loss Plot (Model 5)

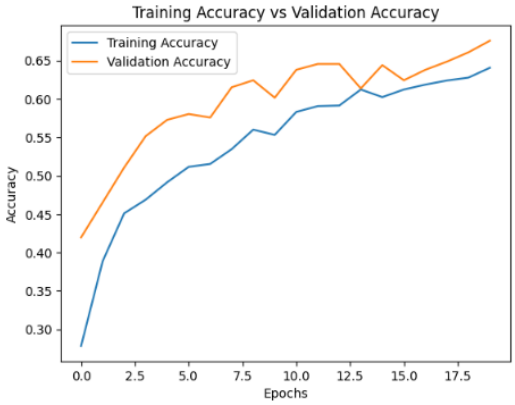

Figure 3.4 CNN Accuracy Plot (Model 5)

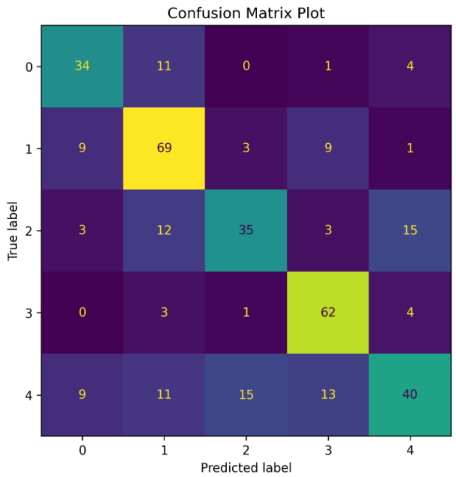

Figure 3.5 Confusion Metrics (Model 5)

	precision	recall	f1-score	support
0	0.62	0.68	0.65	50
1	0.65	0.76	0.70	91
2	0.65	0.51	0.57	68
3	0.70	0.89	0.78	70
4	0.62	0.45	0.53	88
accuracy			0.65	367
macro avg	0.65	0.66	0.65	367
weighted avg	0.65	0.65	0.64	367

Key: 0 = Daisy, 1 = Dandelion, 2 = Roses

3 = Sunflowers, 4 = Tulips

Figure 3.6 Classification Report (Model 5)

4.2.6 Model 6(Batch normalisation and pooling)

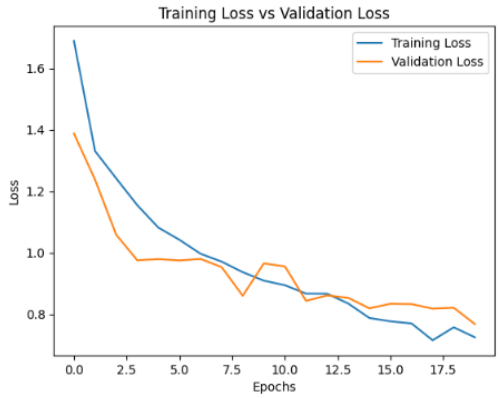

Figure 3.7 CNN Loss Plot (Model 6)

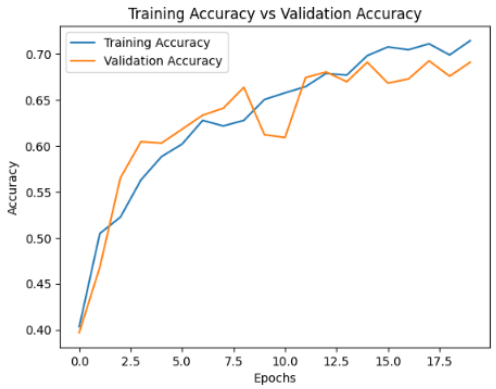

Figure 3.8 CNN Accuracy Plot (Model 6)

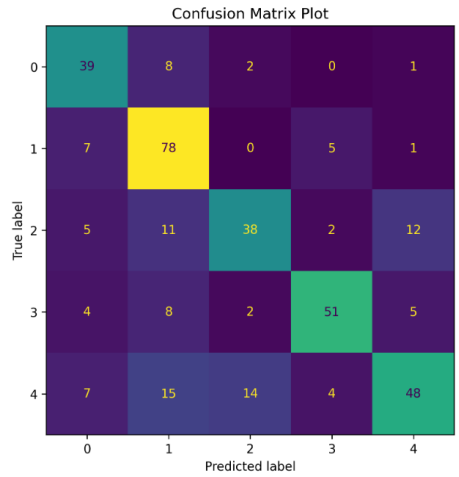

Figure 3.9 Confusion Metrics (Model 5)

	precision	recall	f1-score	support
0	0.63	0.78	0.70	50
1	0.65	0.86	0.74	91
2	0.68	0.56	0.61	68
3	0.82	0.73	0.77	70
4	0.72	0.55	0.62	88
accuracy			0.69	367
macro avg	0.70	0.69	0.69	367
weighted avg	0.70	0.69	0.69	367

Key: 0 = Daisy, 1 = Dandelion, 2 = Roses

3 = Sunflowers, 4 = Tulips

Figure 4.0 Classification Report (Model 5)

4.2.7 Model 7(kernel 5 and batch size 128)

Figure 4.1 CNN Loss Plot (Model 7)

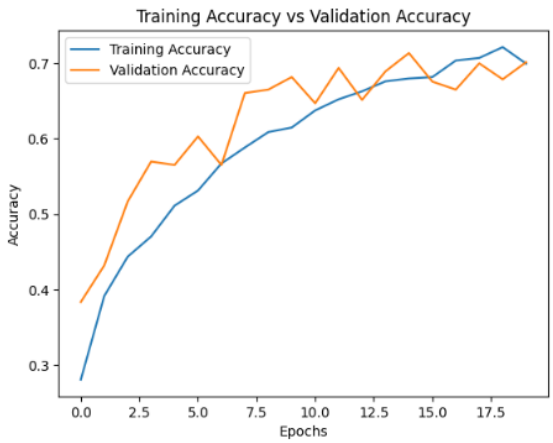

Figure 4.2 CNN Accuracy Plot (Model 7)

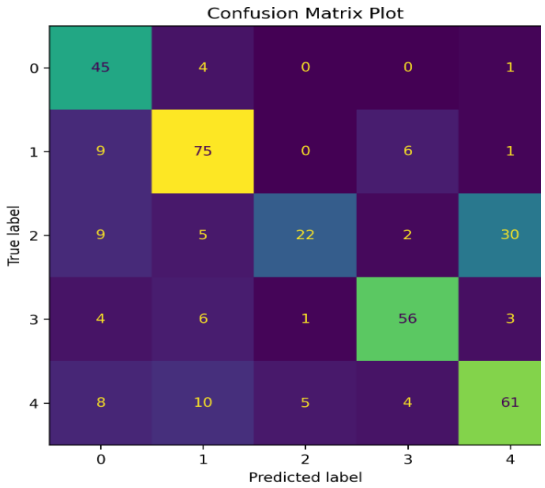

Figure 4.3 Confusion Metrics (Model 5)

	precision	recall	f1-score	support
0	0.60	0.90	0.72	50
1	0.75	0.82	0.79	91
2	0.79	0.32	0.46	68
3	0.82	0.80	0.81	70
4	0.64	0.69	0.66	88
accuracy			0.71	367
macro avg	0.72	0.71	0.69	367
weighted avg	0.72	0.71	0.69	367

Key: 0 = Daisy, 1 = Dandelion, 2 = Roses

3 = Sunflowers, 4 = Tulips

Figure 4.4 Classification Report (Model 5)

4.2.8 Model 8(Kernel 3 and batch normalization)

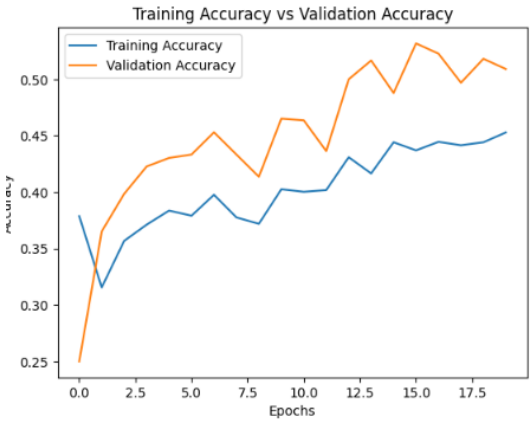

Figure 4.5 CNN Loss Plot (Model 8)

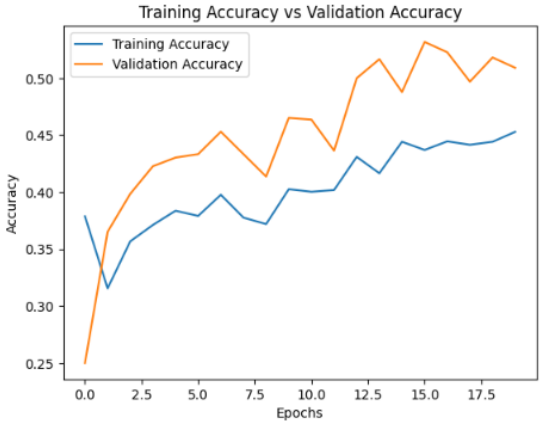

Figure 4.6 CNN Accuracy Plot (Model 8)

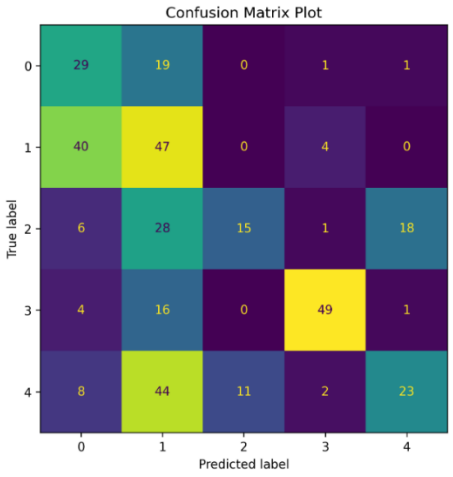

Figure 4.7 Confusion Metrics (Model 5)

	precision	recall	f1-score	support
0	0.33	0.58	0.42	50
1	0.31	0.52	0.38	91
2	0.58	0.22	0.32	68
3	0.86	0.70	0.77	70
4	0.53	0.26	0.35	88
accuracy			0.44	367
macro avg	0.52	0.46	0.45	367
weighted avg	0.52	0.44	0.44	367

Key: 0 = Daisy, 1 = Dandelion, 2 = Roses

3 = Sunflowers, 4 = Tulips

Figure 4.8 Classification Report (Model 5)

5.0 DISCUSSION

The report contains 8 different CNN models from the results figures above shows the different accuracy and precision measures for each of the models used for the image classification. All CNN model used a uniform constructor stage parameters with tunings adjustments like Batch Normalization, Average Pooling to see the improvement and performance of the model as shown in Table 1.0, all the CNN models had an over-fitting which can be seen in the evaluation accuracy and loss plots in the training/validation figures above. The below table shows the accuracy outcome of the various 8 CNN model variations:

Table 1.0 Table showing results of all 8 CNN models.

CNN Models	**Accuracy**
Model 1A	**75%**
Model 1A(Batch size 128)	**78%**
Model 2(Stride 2 x 2)	**72%**
Model 3(Kernel 5 X 5)	**72%**
Model 4(3 Average Pooling 2D)	**65%**
Model 5(3 Batch normalization and 3 Average Pooling2D)	**69%**
Model 5A(3 Batch normalization, Batch size = 128, Kernel size 5 x 5)	**71%**
Model 6 (3 Batch normalization and Kernel size 3 x 3)	**44%**

From all the 8 CNN Models, Model 1A using a batch size of 128 is the best in the flower specie recognition with a 78% accuracy, followed by Model 1 with an accuracy of 75% also using a batch size of 32. This means the original structure of the model is better in predicting flower specie than the other tunned models.

6.0 CONCLUSIONS

Overall, the image recognition report shows the variety of models and their various hyperparameter tunning features used for the image classification. The Batch Normalization alongside a lager batch size (128) shows a better accuracy in their various classification of flower species from the photographs used to test and validate the models, we can infer that a large batch size, 2D convolutional and a MaxPooling2D technique strongly affects the model accuracy.

REFERENCES

1. BOZKURT, F. (2022) *A Study on CNN Based Transfer Learning for Recognition of Flower Species. European Journal of Science and Technology [Preprint].* Available online: https://doi.org/10.31590/ejosat.1039632.

2. E Meghala (2023) *FLOWER RECOGNITION. International Research Journal of Modernization in Engineering Technology and Science [Preprint].* Available online: https://doi.org/10.56726/irjmets37390.

3. Hanafiah, M., Adnan, M.A., Abdul-Rahman, S., Mutalib, S., Malik, A.M.A. & Shamsuddin, M.R. (2022*) Flower Recognition using Deep Convolutional Neural Networks. In IOP Conference Series: Earth and Environmental Science. Institute of Physics.* Available online: https://doi.org/10.1088/1755-1315/1019/1/012021.

4. Hiary, H., Saadeh, H., Saadeh, M. & Yaqub, M. (2018) *Flower classification using deep convolutional neural networks.* IET Computer Vision, 12(6), 855–862. Available online: https://doi.org/10.1049/iet-cvi.2017.0155.

5. Le, T.L., Hai, V., Yagi, Y., Thanh, T., Nguyen, N., Le, V.T., Vu, H. & Pantuwong, N. (2016*) Flower species identification using deep convolutional neural networks.* Available online: https://www.researchgate.net/publication/308322586.

6. Seeland, M., Rzanny, M., Alaqraa, N., Wäldchen, J. & Mäder, P. (2017) *Plant species classification using flower images - A comparative study of local feature representations. PLoS ONE, 12(2).* Available online: https://doi.org/10.1371/journal.pone.0170629.