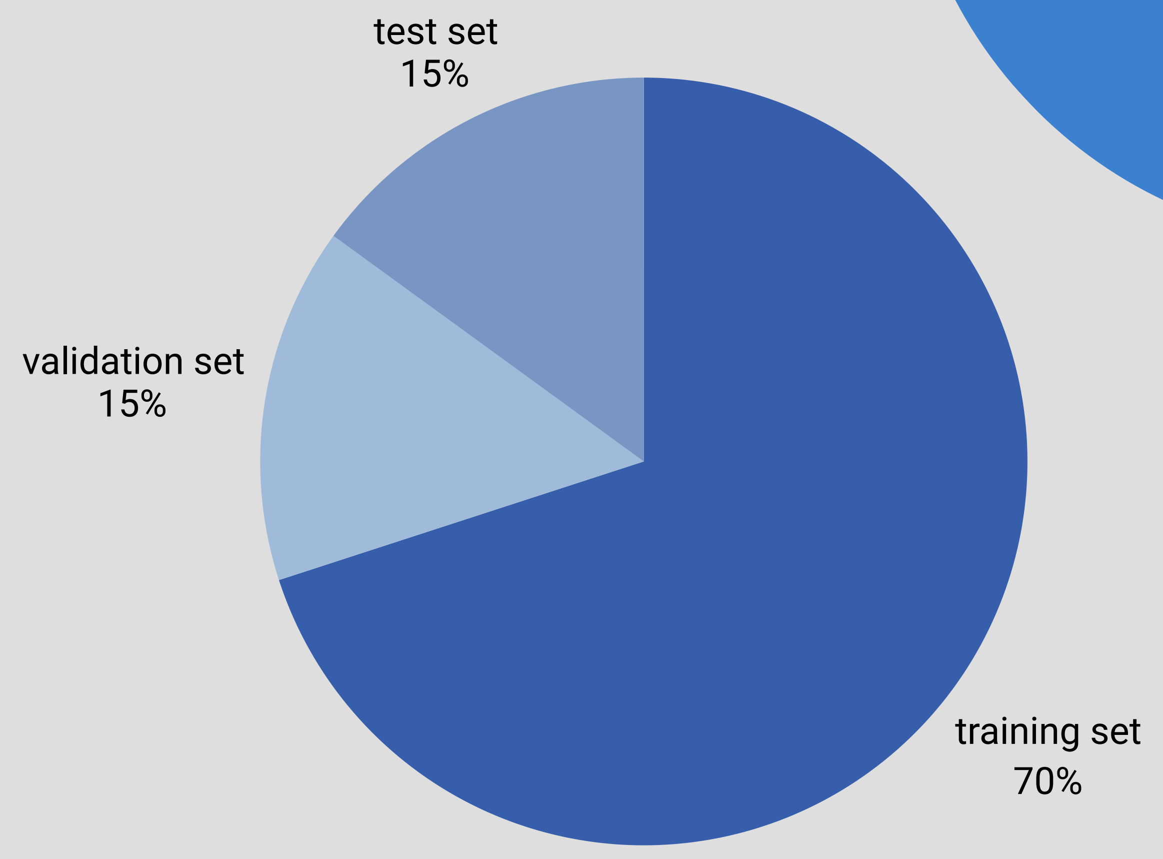
FLOWERS CLASSIFICATION USING DEEP LEARNING

Hadis Forghani (920040)
Semen Sazanov (934361)
Any Das (922710)

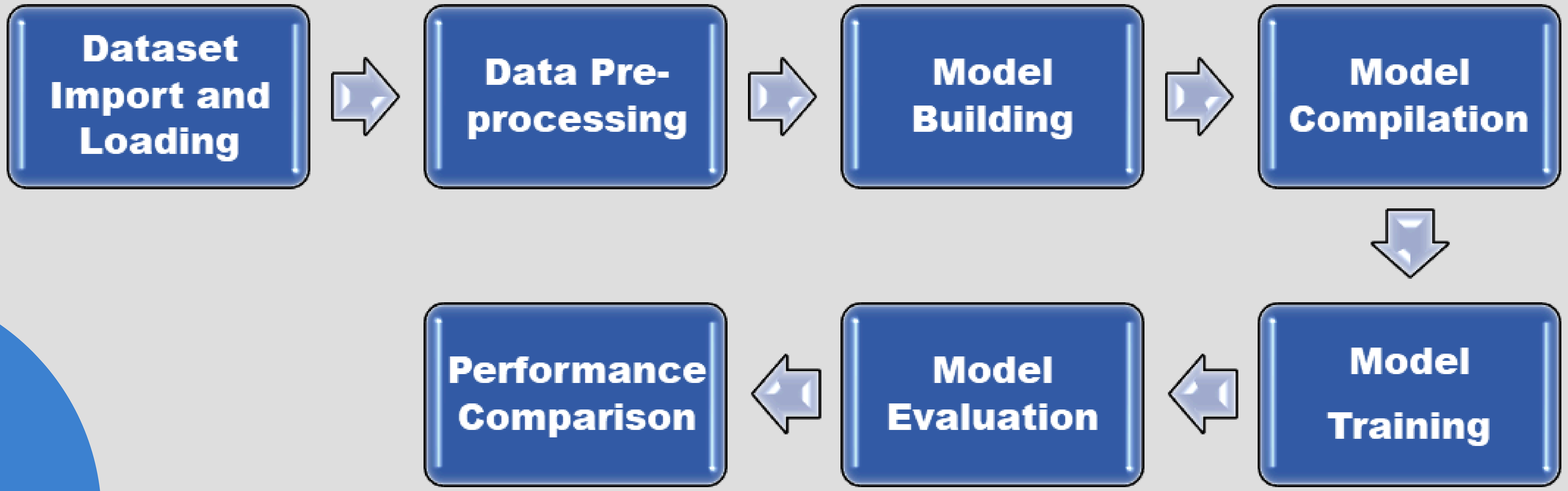
INTRODUCTION & DATASET OVERVIEW

- The main goal of this project is to classify the images contained in the flowers dataset into these 5 categories: Daisy, Dandelion, Roses, Sunflowers and Tulips.
- We used the "flower_photos" dataset provided by TensorFlow.
- The dataset contains 3,670 flower images of varying sizes and orientations. We split the dataset into 70% training, 15% validation and 15% test sets.
- This project shows how deep learning can be applied to solve real-world problems like automated flower recognition.



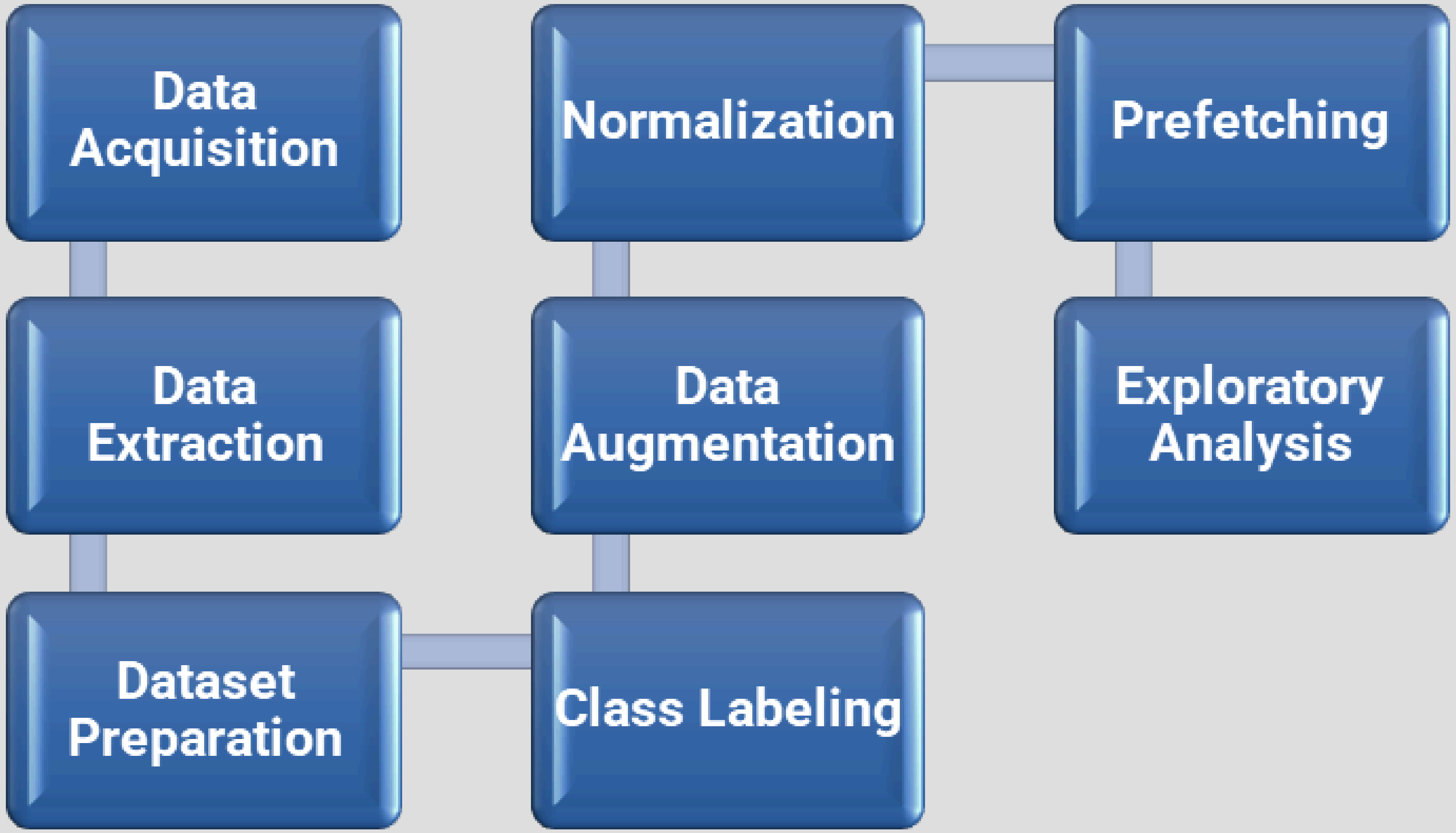
METHODOLOGY OVERVIEW

Overall Working Process of Flower Classification



DATA PREPROCESSING

Work Flow of Data Preprocessing

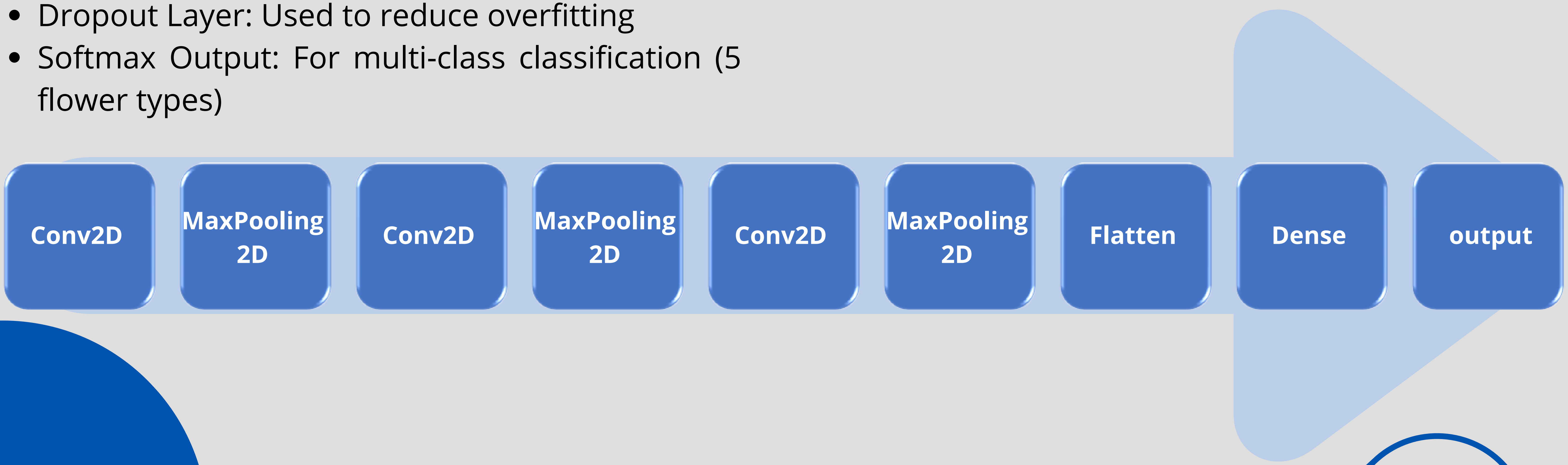


MODEL 1 ARCHITECTURE

We designed a Convolutional Neural Network (CNN) from scratch, consisting of the following layers

Model Summary:

- Parameters: ~6.6 million (most in the Dense layer)
- Dropout Layer: Used to reduce overfitting
- Softmax Output: For multi-class classification (5 flower types)



Compilation & Training:

Loss Function: SparseCategoricalCrossentropy (since labels are integers)

Optimizer: Adam

Callbacks Used:

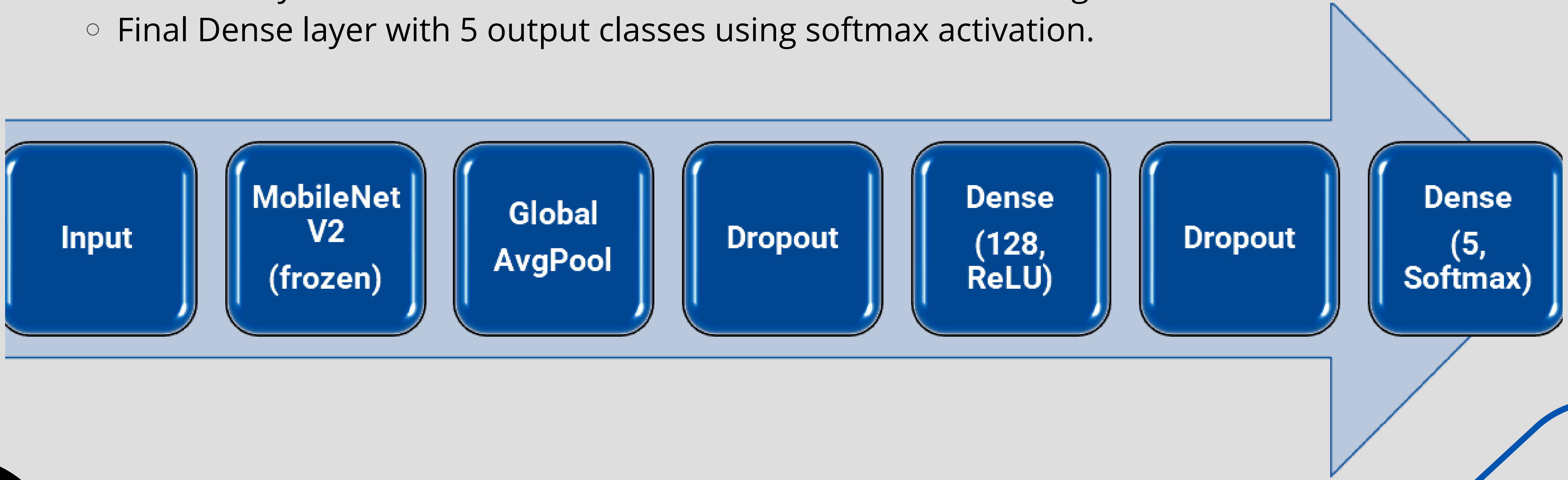
- EarlyStopping: Stops training when validation accuracy stops improving.
- ReduceLROnPlateau: Reduces learning rate on plateau.
- ModelCheckpoint: Saves best model.

Training Results

The model achieved around 60% training accuracy and 55% validation accuracy, indicating that the model has learned a reasonable amount of discriminative features from the flower dataset.
The small gap between training and validation accuracy suggests that the model generalizes reasonably well and is not overfitting severely. The confusion matrix and classification report indicate moderate class-wise performance.

MODEL 2 ARCHITECTURE

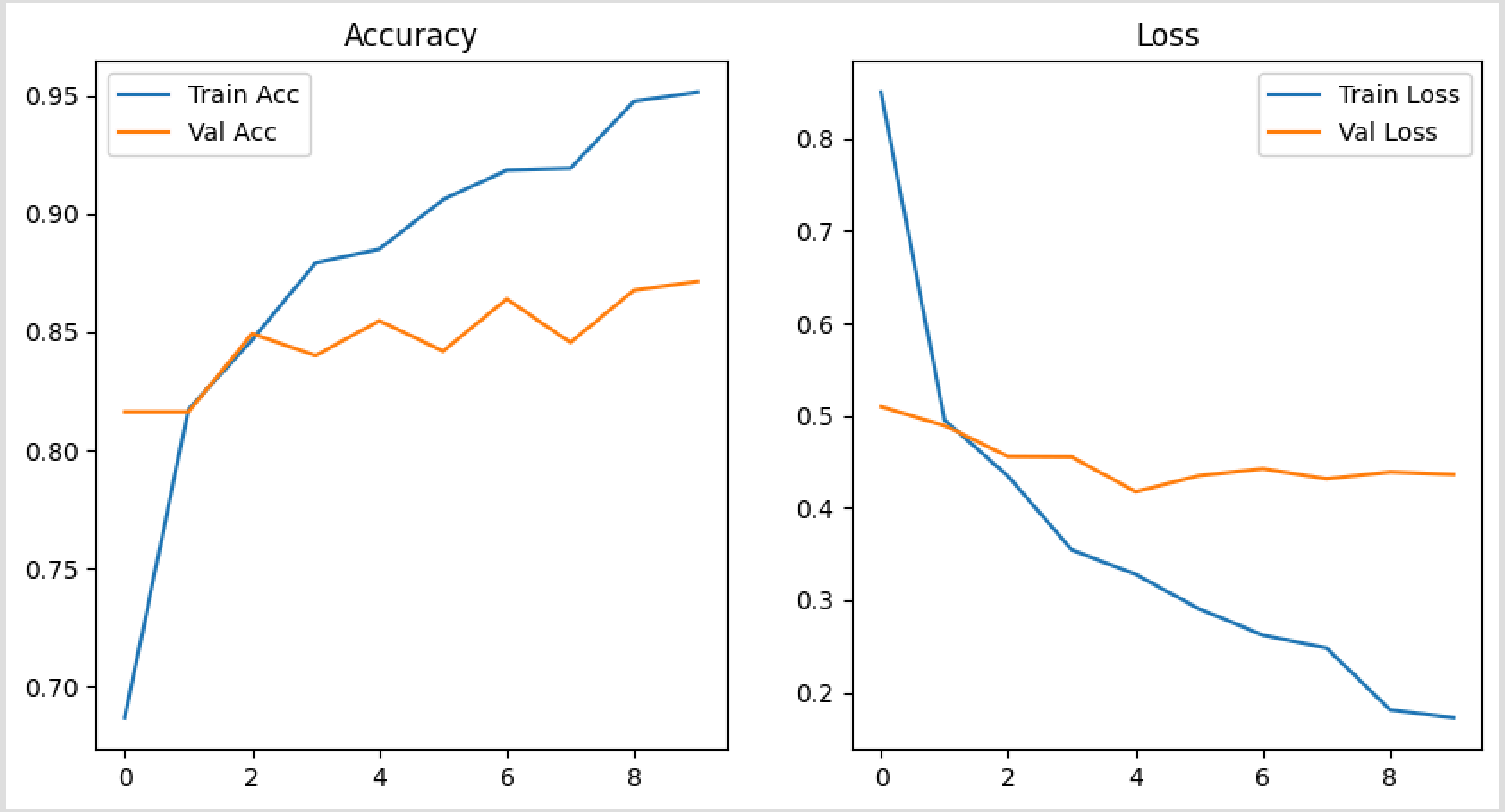
- For the second model, we used MobileNetV2 as the backbone, which is a lightweight and efficient convolutional neural network.
- We froze the base model to use it only for feature extraction.
- We added our own custom classification head:
  - A GlobalAveragePooling2D layer to reduce dimensionality.
  - Dropout layers to prevent overfitting.
  - A Dense layer with 128 neurons and ReLU activation with L2 regularization.
  - Final Dense layer with 5 output classes using softmax activation.



MODEL 2 ARCHITECTURE

Training Process

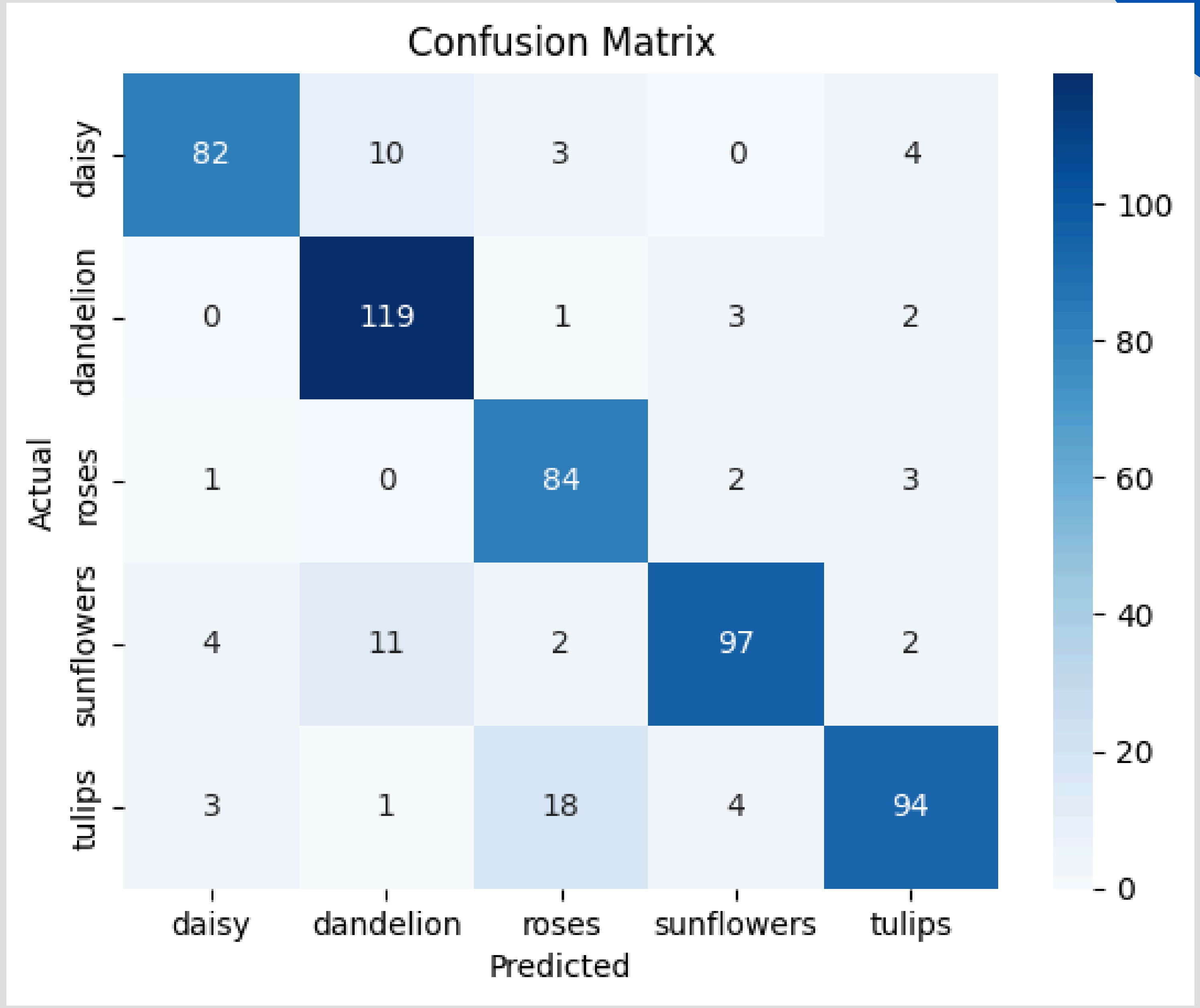
- We trained the model with the Adam optimizer (learning rate 0.001) using sparse categorical crossentropy loss and accuracy as the metric. Dropout layers around the Dense layer and L2 regularization helped prevent overfitting.
- We used three callbacks: EarlyStopping (patience 10, restores best weights), ModelCheckpoint (saves best model), and ReduceLROnPlateau (reduces learning rate when validation loss plateaus).
- Training ran for 10 epochs. The loss curves on the below show smooth convergence and minimal overfitting.



MODEL 2 ARCHITECTURE

Evaluation & Results

Class	Precision	Recall	F1-Score	Support
Daisy	91	83	87	99
Dandelion	84	95	89	125
Roses	78	93	85	90
Sunflowers	92	84	87	116
Tulips	90	78	84	120
Accuracy			**87**	550
Macro Avg	87	87	86	550
Weighted Avg	87	87	86	550

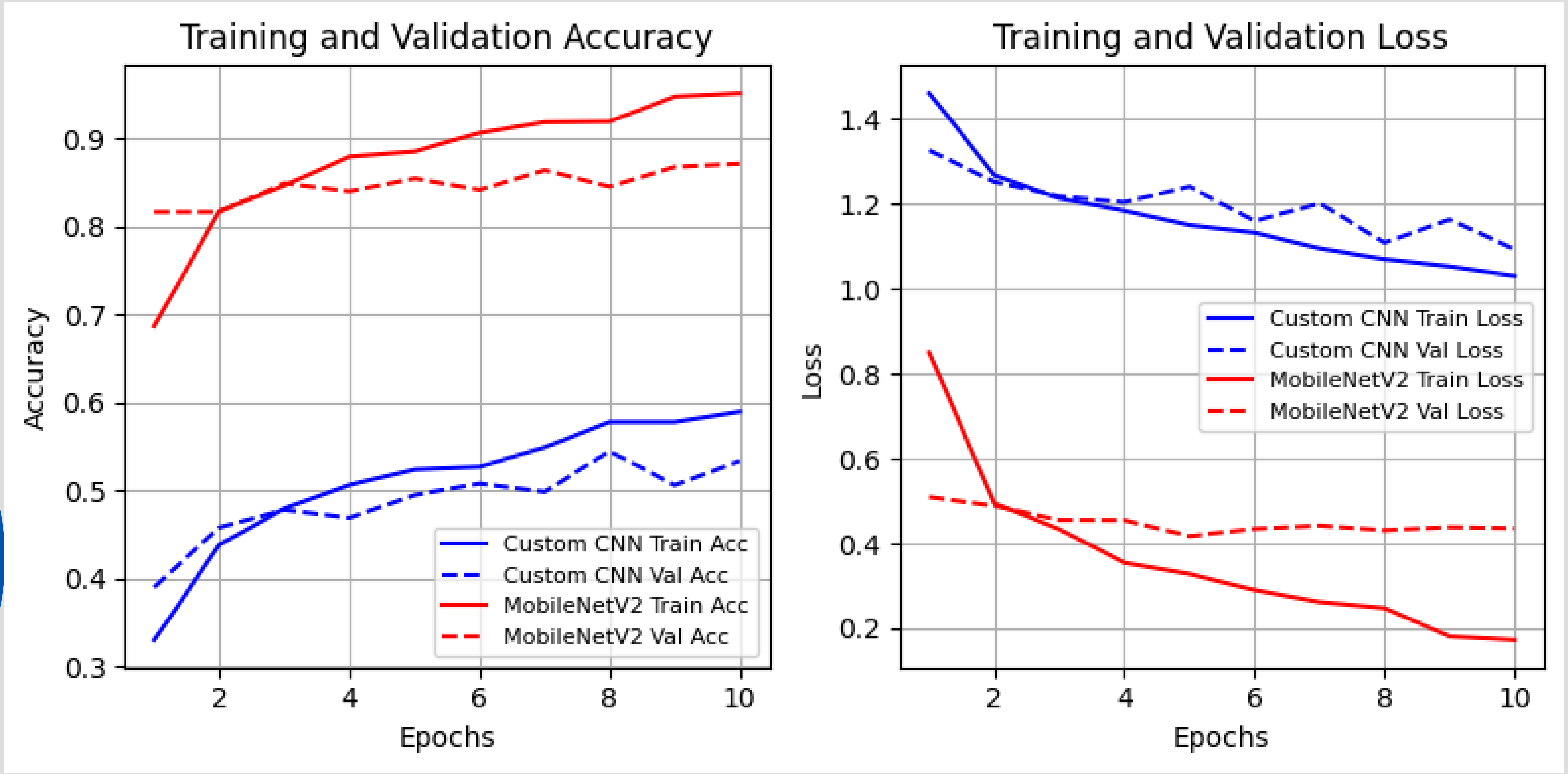


Actual \ Predicted	daisy	dandelion	roses	sunflowers	tulips
daisy	82	10	3	0	4
dandelion	0	119	1	3	2
roses	1	0	84	2	3
sunflowers	4	11	2	97	2
tulips	3	1	18	4	94

COMPARISON BETWEEN MODEL 1 & MODEL 2

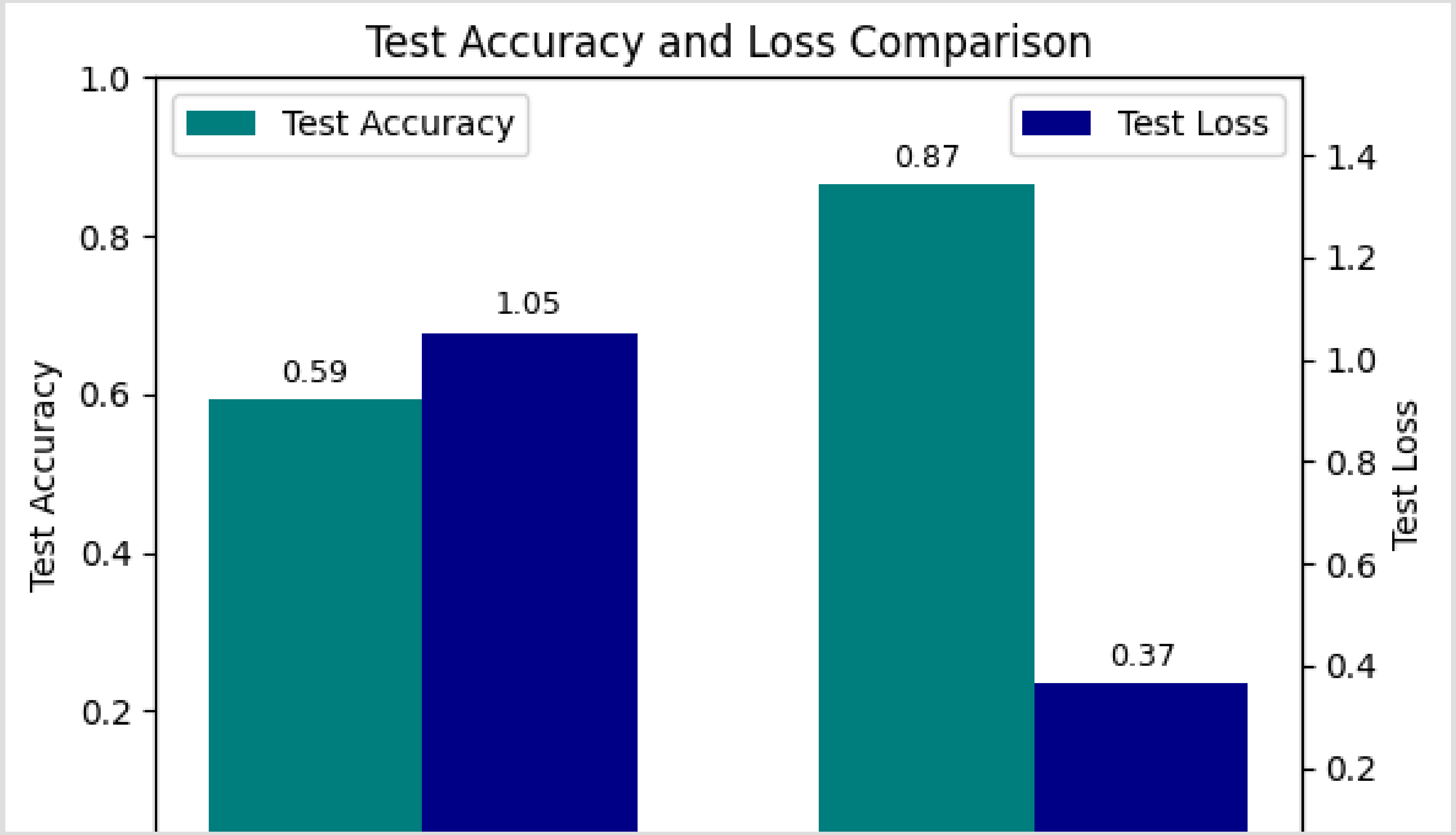
Training and Validation Performance

From the graphs, MobileNetV2 clearly performs better than Custom CNN. It reaches around 94% training accuracy and 88% validation accuracy, while the Custom CNN only gets to about 59% training and 53% validation accuracy. Also, MobileNetV2 keeps a much lower loss across 10 epochs.



Test Accuracy and Loss

On the test set, MobileNetV2 achieves 0.87 accuracy and 0.37 loss, which is much better than Custom CNN's 0.59 accuracy and 1.05 loss. This shows that MobileNetV2 generalizes better and is more reliable for predictions.

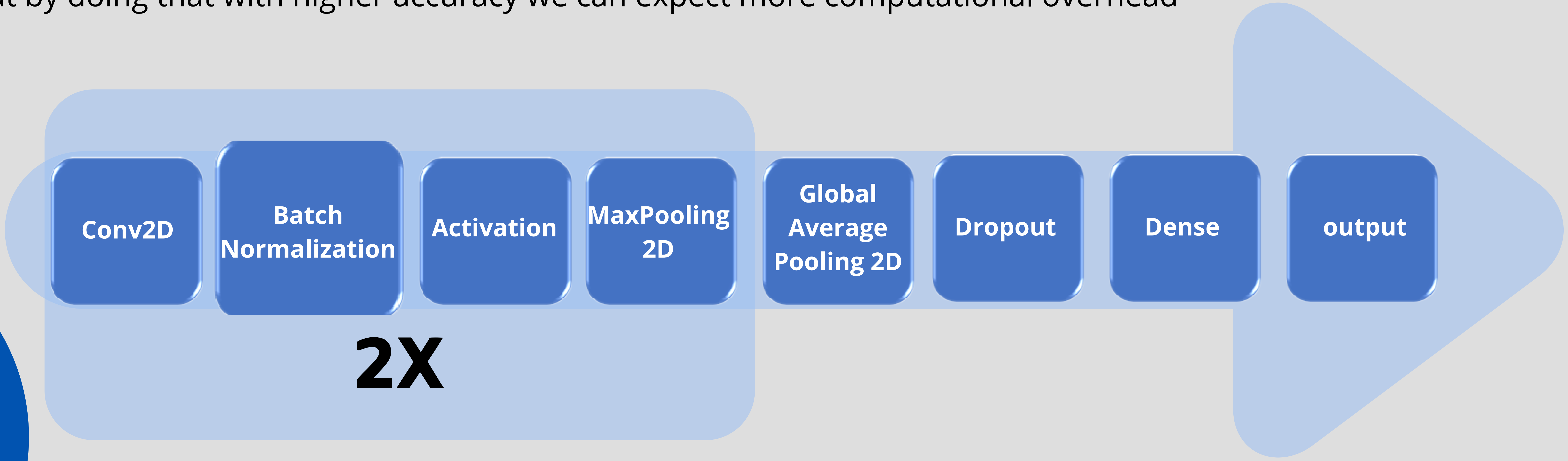


MODEL 3 ARCHITECTURE

The base was Model 1 but to reduce overfitting and use max from data provided we added:

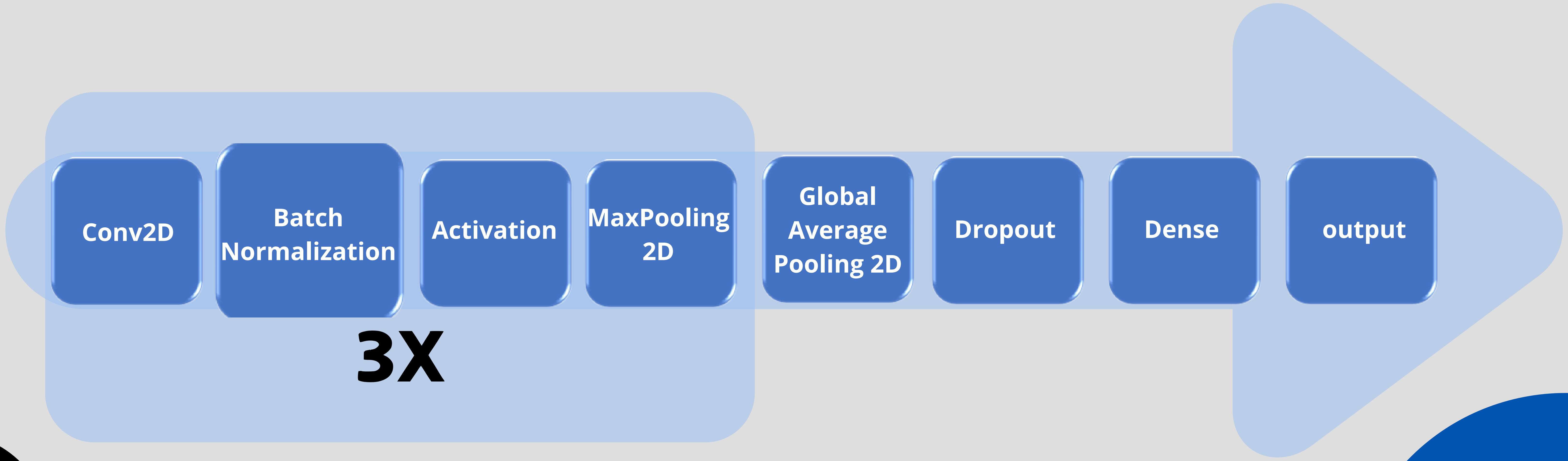
- Normalization
- L2 Regularization
- Another order of layers: Convolution→BN→Activation
- 'same' padding

But by doing that with higher accuracy we can expect more computational overhead

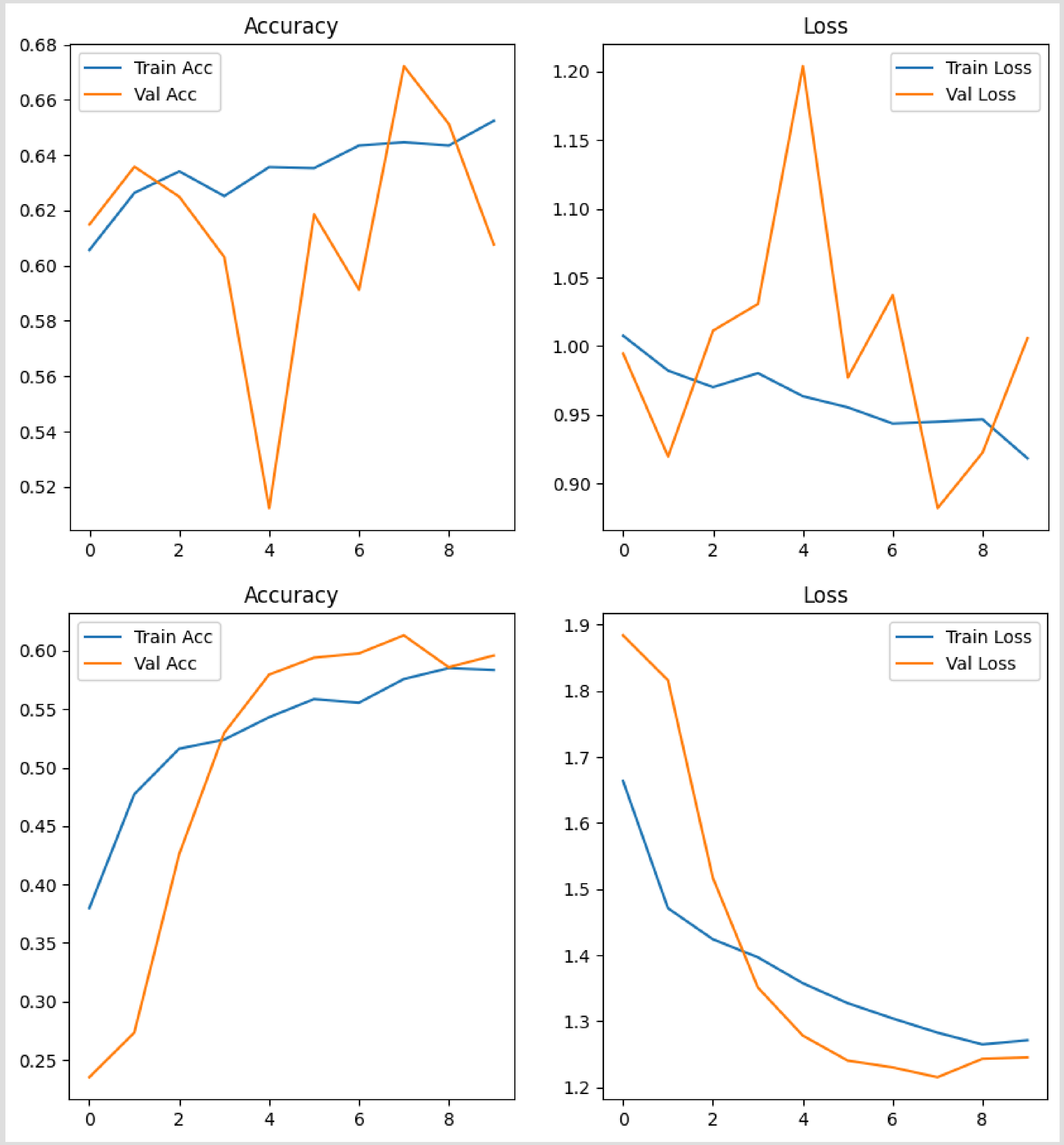


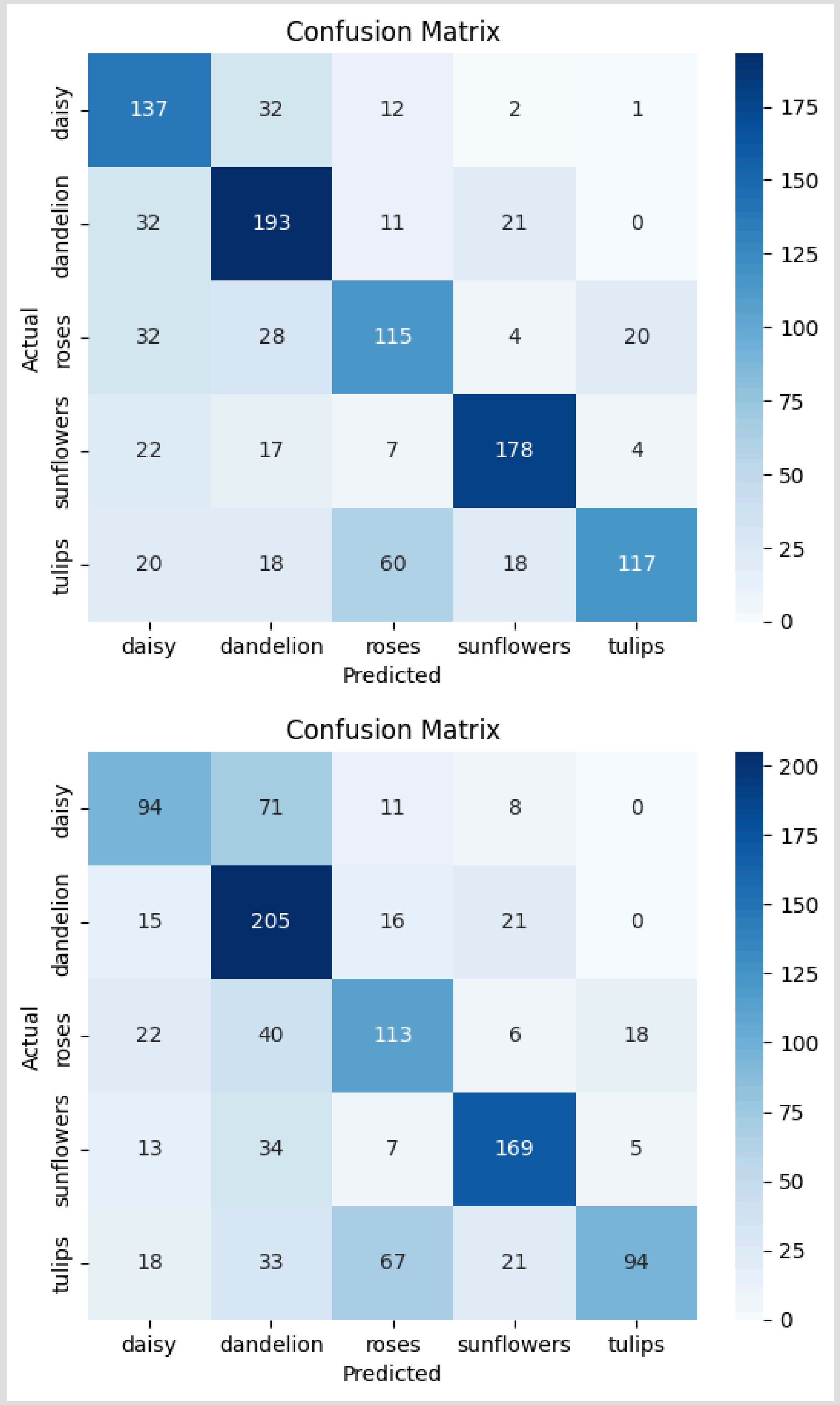
MODEL 4 ARCHITECTURE

The main idea was to improve the Model 3 by tuning main variables (learning rate from 0.001 to 0.0001, dropout rate from 0.3 to 0.4, and raising the L2 regularization factor from 0.0001 to 0.001) and adding one more convolution layer to reduce over and under-regularization. While it will definitely increase the complexity we are expecting the highest performance from data set that we have.



COMPARISON BETWEEN MODEL 3 & MODEL 4





Confusion Matrix

Actual \ Predicted	daisy	dandelion	roses	sunflowers	tulips
daisy	137	32	12	2	1
dandelion	32	193	11	21	0
roses	32	28	115	4	20
sunflowers	22	17	7	178	4
tulips	20	18	60	18	117

Model 3

Confusion Matrix

Actual \ Predicted	daisy	dandelion	roses	sunflowers	tulips
daisy	94	71	11	8	0
dandelion	15	205	16	21	0
roses	22	40	113	6	18
sunflowers	13	34	7	169	5
tulips	18	33	67	21	94

Model 4

CONCLUSION

We obtained the best classification results through the second model (MobileNetV2) with a validation accuracy of 88% and 94% accuracy on the test set.
Our custom CNN model, although less accurate, still showed a significant improvement over the MLP by leveraging spatial information in images. Overall, this project showed us how, even with a limited amount of data, we can improve the performance of the model and how a better pre-trained model is in handling classification problems.

FUTURE IMPROVEMENTS

- We used MobileNetV2 with frozen weights. As a next step, we want to fine-tune the MobileNetV2 model to improve performance.
- Our models used 160×160 images; increasing the input size (like 224×224) could help the network learn finer details.
- We already applied basic augmentation (flip, brightness, contrast), but we could test more advanced ones like zoom or rotation.
- We'd also like to try model ensembling by combining predictions from MobileNetV2 and the tuned CNN to boost accuracy.

THANK YOU