Machine Learning

Semester A, 2023

Assignment 2 - Group Project

Team T1-G02: Flower Classification & Recommendation System

Group members:	Vo Thanh Luan, s3822042
	Nguyen Xuan Thanh, s3915468
	Nguyen Vi Phi Long, s3904632
	Vo Ngoc Diem Tien, s3911365

Lecturer:	Dr. Nguyen Thien Bao
Submission Date:	May 19th, 2023

I. Introduction

Due to rapid expansion, online flower sellers are competing more. This industry strives to create a user-friendly search engine to assist customers discover their flowers. Customers unfamiliar with flower names may not find text-based search useful or accurate. Thus, the objective is to develop a machine learning model for an image-based search tool that improves client experience and search result accuracy.provides an image-based search tool to improve consumer experience and search results.

The solution comprises floral categorization and related flower suggestions. First, create a classification system that accurately identifies flower kinds from the input picture. The goal is to train the model to classify input photos into eight flower classes from the dataset. Recommending dataset flower photos is the second job. The model is trained to create 10 floral pictures that match the input image. The methodology improves user experience and flower discovery by offering relevant and aesthetically comparable choices.

II. Data preparation:

Dataset:

A dataset consisting of 4621 flower photographs has been made available to us. This dataset will help us identify flowers in photographs. There are eight different flower kinds, giving us eight distinct categories to divide them into. With reference to Appendix [1], it is evident that the resolution of each image differs. Additionally, Appendix [2] presents the overall count of flowers by category.

There is an imbalance in the distribution of flower categories in this data set. Some categories contain more images than others; Babi, Chrysanthemum, and Lisianthus account for 56.18% of the dataset. This imbalance has the potential to impact the classification model's performance, as it may become skewed toward the majority categories and struggle to accurately classify the minority categories.

The algorithm may recall duplicate images in the training dataset instead of recognizing patterns and attributes that apply to unseen data. Overfitting happens when the model performs well on training data but fails to generalise to new images. Duplicate images may potentially bias the model. This may affect class representation and degrade the model's ability to classify whole dataset images. Finally, dataset duplication may limit model training images. Diverse images teach the model resilient traits and generalise to new images. If duplicates predominate, the model may not be trained on all data variations and complexity.

Normalisation minimises model input data variance sensitivity and enhances training optimization problem conditioning. Scaling features to a consistent range improves model resilience to input magnitude or units. Therefore, it positively impacts the model's generalisation. If input features have different scales, the optimizer may struggle to identify an optimal solution. Normalisation equalises learning and avoids dominance and optimize the condition of the dataset. Without normalisation, larger characteristics may dominate learning and skew relevance.

Data Cleaning

To illustrate the various stages of the data cleaning process, we employed a sample image as a reference point for the pre-and post-processing stages. Removing image duplication and normalisation in the dataset might significantly affect the training model. The normalisation process involves modifying the range of pixel intensity values in order to bring an image into a range that is more appropriate for perception. Furthermore, it is imperative to verify and eliminate duplications after normalising images during the training process, as this can enhance the accuracy and speed of learning progress.

III. Model Training:

Task 1: Classification

For this task, we employed the CNN, VGG19, and Resnet50V2 models. The selected architectures were chosen based on their established success in analogous image classification tasks, as well as their appropriateness for transfer learning. The use of Convolutional Neural Network (CNN) models is widespread in image classification, employing deep learning techniques. The architectural design contains convolutional, pooling, and fully connected layers to extract image features at different levels of abstraction. Furthermore, ResNet and VGG, specifically ResNet50V2 and VGG19, are established convolutional neural network architectures that have demonstrated exceptional accuracy in various image classification benchmarks.

The assessment posed a difficulty by prohibiting the use of pre-trained weights, necessitating the design of our own CNN architecture. We discovered how to determine the number of convolutional and max-pooling layers, apply dropout to prevent overfitting, and add an output layer with dense softmax activation for classification since the dataset has multiple classes. Understanding the function and order of each layer was crucial, such as using max-pooling layers to reduce spatial dimensions after each convolutional layer and situating the dense softmax layer for classification at the end.

We also investigated the influence of sample size and the number of epochs on the training procedure, recognizing their importance in obtaining desirable results. The initial model's overfitting prompted us to investigate methods for preventing it further. We discovered the efficacy of early stopping to prevent overfitting and a learning rate reduction strategy.

We employ data augmentation techniques such as random image twisting, moving, rotating, and magnifying. This strategy seeks to improve model performance by reducing overfitting by creating modified copies of original images, thereby increasing the size of the training dataset.

We had the opportunity to integrate existing models, ResNet50V2 and VGG19, to enhance the efficacy. Recognizing the intricacy of these models, we chose to enable certain layers in order to simplify and adapt them to the task at hand. This procedure required a more in-depth understanding of the architecture of these base models and their adaptation to the classification issue. Eventually, ResNet50V2 became the most accurate model.

- ***ResNet50V2***

The model is built based on the ResNet50V2 with an input size of 256*256 and 3 colour channels, excluding the output layer. We freeze the first 170 layers of ResNet50V2 to simplify our model. The layers of the base model are presented in Appendix [3]. Next, the Global Average Pooling (GAP) layer averages the feature maps from the base model and produces a feature vector summarising the image. This vector is then passed through a dense layer with 512 units and the ReLu activation function. A dropout layer is then added to prevent overfitting by randomly setting some of the output values of the previous layer to zero during training. The output layer is a dense layer with 8 units and a softmax activation function that outputs a probability distribution over the possible classes because our task is a multi-label classification problem. The model's architecture is presented in Appendix [4].

We have opted to use the categories-cross-entropy function as our designated loss function. The metric has been chosen which is accuracy. The metric was appropriate due to its simplicity and clarity in assessing the accuracy of the classification model, expressed as a percentage. In addition, we employ the technique of data augmentation, as previously discussed, which has the potential to generate additional data and enhance the efficacy of our models. Our team employs the Adam optimization algorithm, an extension of stochastic gradient descent (SGD), to train neural networks. This algorithm uses adaptive learning rates for each parameter and is a widely recognized optimization technique in the field.

Task 2: Recommendation

Data Augmentation

The images are resized to a range of 0 to 1, sheared, zoomed, flipped horizontally, and shifted in width and height. Additionally, the images are randomly rotated by up to 20 degrees. These transformations introduce diversity into the training data, allowing the model to generalise better and improve its performance. The augmented dataset is split into training and validation subsets, with 80% used for training and 20% for validation.

- ***VGG16:***

The VGG16 model architecture you provided is a modified version of the original VGG16 network, which was introduced by the Visual Geometry Group at the University of Oxford. The model is designed for image classification tasks and consists of several convolutional and pooling layers followed by fully connected layers. Our model begins with the input layer, which expects images of size 224x224 pixels with 3 channels (RGB). The first block, Block 1, includes a convolutional layer with 32 filters of size 3x3, followed by a ReLU activation function to introduce non-linearity. Max pooling is applied with a pool size of 2x2 to reduce the spatial dimensions.

Block 2 follows a similar pattern, with a convolutional layer of 64 filters and a pool size of 2x2. Block 3 increases the number of filters to 128, and Block 4 further increases it to 256. In each block, the convolutional layers are followed by ReLU activation and max pooling. After the convolutional blocks, a flatten layer is added to convert the 3D feature maps into a 1D feature vector. This is followed by fully connected layers: FC1, FC2, and the output layer. FC1 has 128 units, FC2 has 64 units, and the output layer has 8 units, corresponding to the number of classes in your flower dataset. The activation function for the output layer is softmax, which outputs probabilities for each class.

The model is compiled with the categorical cross-entropy loss function, which is suitable for multi-class classification problems. The Adam optimizer is used to optimise the model parameters. During training, the model will be fitted using the `train_generator` and validated using the `validation_generator`, which generates batches of augmented images from the specified data folder. The training process will iterate over the data for a specified number of epochs, adjusting the model's parameters to minimise the loss function.

IV. Result

Task 1: Classification

The ResNet50v2 model was trained for only 14 of 70 epochs with a batch size of 64 due to premature callback function termination. Accuracy in the training set was at 96.67% , whereas accuracy in validation set was 79.14%. The model showed signs of mild overfitting after the 10th epoch, as seen by an increasing discrepancy between the training accuracy and loss and the validation accuracy and loss. As a result, the measures suffer a marginal decline in comparison to the validation phase. (According to appendix [5] [6].)

- **Accuracy**: The ResNet50v2 model attained an accuracy score of 0.7712.
- **Precision**: The precision score of 0.7795 indicates an accuracy of around 77.95% in predicting.
- **Recall**: The ResNet50v2 model can correctly recognize 77.12% of flower categories.
- **F1**: ResNet50v2's F1-score of 0.7713 suggests consistent accuracy and recall.
- **Confusion Matrix**: The ResNet50v2 model exhibits improved performance according to its confusion matrix when compared to CNN and VGG19. Notably, categories 1 (Babi), 3 (Chrysanthemum), and 5 (Lisianthus) manifest a higher number of precise predictions, as evidenced by elevated values along the diagonal. Furthermore, the ResNet50v2 model demonstrates better distribution of accurate predictions across various flower categories.

ResNet50v2 outperforms CNN and VGG19 in accuracy, precision, recall, and F1-score. Flower species are better classified with greater accuracy and precision ratings. The ResNet50v2 model also reliably detects actual occurrences of each category, improving recall. The confusion matrix shows improved performance, supporting the findings. However, it is worth noting that the model currently shows slight signs of overfitting.

Task 2: Recommendation

The VGG16 model achieved an accuracy of 82.02% on the training data, which means that it correctly classified approximately 82.02% of the training samples. Moving on to the validation accuracy, the model achieved an accuracy of 65.91% on the validation set. This metric measures the model's ability to generalize to unseen data. That means the model had overfitted a little bit.

When comparing the accuracy of the VGG16 model and the Siamese model, it is evident that the VGG16 model outperforms the Siamese model. The VGG16 model achieves an accuracy of 0.8202 during training and 0.6591 during validation, whereas the Siamese model achieves an accuracy of 0.6768 during training and 0.6160 during validation.

In conclusion, the VGG16 model outperforms the Siamese model in terms of precision, making it the best candidate for the flower recommendation assignment.

V. Independance Evaluation

To provide a comprehensive evaluation of our Machine Learning model for flower classification, it is necessary to compare and contrast our results with those of previously published works with similar objectives. To conduct this evaluation, we researched and analysed a variety of flower classification-related works [8][9][10]. The target is searching for studies, research papers, and initiatives employing comparable datasets and classification tasks. By analysing these external works, we intended to evaluate our model's strengths and shortcomings in comparison to other methodologies and establish a performance benchmark.

To fully understand flower classification, we examined model architecture, training methods, data augmentation strategies, and optimization algorithms from prior research [11][12]. Appendix [9], our model has a higher accuracy (0.9672) than the other two training set models studied. On the validation set, our model is more accurate than the second researched model (0.7894), but less accurate than the first researched model (0.9062). Our model has a lower loss (0.0929) on the training set than the other two researched models. Our model has a greater loss (0.8644) than the other two models on the validation set. Overall, our model outperforms the investigated models in terms of accuracy on the training set. On the validation set, it performs better than one researched model but less accurately than the other. In addition, your model has a greater loss on the validation set than the studied models.

Based on the research, here are some areas that may be considered to improve our model:

- **Model Architecture**: Try the base model with pre-trained weights to profit from a huge dataset. Apply fine-tuning to balance pre-trained features, change the basic model's frozen layers. Consider ensemble methods, where multiple models are combined to execute the predictions.
- **Data Augmentation**: Explore additional augmentation techniques such as translation, vertical flip and brightness adjustments.
- **Regularisation**: To prevent overfitting and enhance the model's performance on unseen data, investigate regularisation techniques such as L1 and L2 regularisation.
- **Hyperparameter Tuning**: Fine-tuning hyperparameters: learning rate, batch size, and number of epochs. Explore different combinations to find the optimal set of hyperparameters.
- **Optimization**: Explore different optimization algorithms (eg., SGD with momentum).
- **Callbacks**: Use ModelCheckpoint callbacks to store validation-based model weights. Use LearningRateScheduler to dynamically alter training speed depending on a timetable.

Due to hardware constraints like a shortage of computational units, training and evaluating the model can be a time-consuming process. This issue is exacerbated by the computational complexity such as the ResNet50V2 or VGG19 model, with its deep architecture and numerous parameters. As a result, fine-tuning layers or hyperparameters becomes a time-consuming task that restricts the scope of experimentation. With limited computing capacity, it is difficult to investigate a broad variety of hyperparameter settings, conduct extensive grid searches, or conduct extensive experiments. Consequently, the model may not attain the required level of precision or optimal performance within the time constraints available.

VI. Ultimate Judgment

Task 1: Classification

Based on the model description and evaluation, the flower classification accuracy of the ResNet50V2 model is quite good. The ResNet50V2 model has a slight tendency toward overfitting, indicating that its performance may not generalise well to unobserved data. Using techniques such as regularisation or expanding the size and diversity of the training dataset can circumvent this limitation.

The capacity of the ResNet50V2 model to achieve high accuracy when trained on massive datasets is a significant advantage. The architecture of the model makes use of depth and skip connections, allowing it to recognize intricate patterns and acquire complex representations from numerous training examples.

Transfer learning is an additional advantage of ResNet50V2, as it can be utilised as a pre-trained model on massive datasets like ImageNet. This pre-training facilitates transfer learning, enabling the model to be fine-tuned on a reduced task-specific dataset, leading to faster convergence and improved performance.

However, it is essential to evaluate the computational complexity of ResNet50V2. Due to the deep and complex of the model, it requires more robust hardware and longer training periods than simpler architectures. Additionally, the deep architecture of ResNet50V2 results in a large memory footprint, especially during training. This may limit its applicability in environments with limited resources or on devices with limited memory.

In conclusion, despite the fact that the ResNet50V2 model offers the advantage of high accuracy when trained on large datasets and the benefits of transfer learning, it is essential to address its overfitting behaviour and consider the availability of copious training data. In addition, computational complexity and memory requirements must be taken into account when deploying the model in real-world scenarios.

Task 2: Recommendation

One of the reasons for choosing the VGG16 model for a visually similar image system is its excellent performance in image recognition tasks. VGG16 is a deep convolutional neural network architecture that has achieved remarkable success in various computer vision challenges. It is known for its ability to capture and understand intricate details in images, making it particularly suitable for tasks that require identifying visually similar images.

One of the significant advantages of the VGG16 model is its depth. With 16 layers, it can learn complex hierarchical representations of images, enabling it to extract both low-level and high-level features. This depth allows the model to capture fine-grained details and intricate patterns, making it well-suited for tasks like identifying visually similar images. Additionally, VGG16's architecture consists of small 3x3 convolutional filters, which help in preserving spatial information and retaining the fine details of the images.

Another advantage of VGG16 is its availability and ease of use. The model can be trained on large-scale image classification datasets like ImageNet, which contain millions of labeled images from various categories. This pre-training allows the model to learn general features from a wide range of images and helps in achieving good performance even with limited training data. Moreover, many deep learning frameworks provide pre-trained versions of VGG16, making it easily accessible and enabling rapid development of visually similar image systems.

However, there are a few disadvantages to consider in the real-world perspective. One of them is the computational complexity of VGG16. The model has a large number of parameters, which can make training and inference computationally expensive, especially when dealing with large datasets or real-time applications. Additionally, the memory requirements for storing the model can be significant. These factors may pose challenges in deploying VGG16 on resource-constrained devices or in scenarios where real-time performance is crucial.

To improve the VGG16 model for visually similar image systems, one possible approach is to incorporate attention mechanisms. Attention mechanisms allow the model to focus on relevant regions of an image, providing a better understanding of important visual cues. By adapting the VGG16 architecture to include attention modules, the model could prioritise specific features or regions that are more informative for identifying visually similar images. This could potentially enhance the model's performance and improve its efficiency by reducing the computational load.

Another improvement could involve incorporating transfer learning techniques. Instead of using the entire VGG16 model, which may be over-parameterized for the specific task of visually similar image retrieval, one could fine-tune only certain layers or use them as feature extractors. By leveraging the pre-trained knowledge of the model on large-scale datasets, and then adapting it to the specific task at hand, we can achieve better performance with less computational overhead.

In conclusion, the VGG16 model is a powerful choice for visually similar image systems due to its strong recognition capabilities and availability of pre-trained models. However, its computational complexity and memory requirements may pose challenges in real-world applications. By incorporating attention mechanisms and leveraging transfer learning techniques, we can address these limitations and enhance the model's performance, making it more efficient and effective for visually similar image retrieval tasks.

VII. Reference

[1] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, et al., "Gradient-based learning applied to document recognition," Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, 1998.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Advances in Neural Information Processing Systems, 2012, pp. 1097-1105.

[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," CoRR, vol. abs/1409.1556, 2014.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770-778.

[4] Tammina, S., 2019. Transfer learning using vgg-16 with deep convolutional neural network for classifying images. International Journal of Scientific and Research Publications (IJSRP), 9(10), pp.143-150.

[5] Tiwari, V., Pandey, C., Dwivedi, A. and Yadav, V., 2020, December. Image classification using deep neural network. In *2020* 2nd International Conference on Advances in Computing, Communication Control and Networking *(ICACCCN)* (pp. 730-733). IEEE.

[6] Mascarenhas, S. and Agarwal, M., 2021, November. A comparison between VGG16, VGG19 and ResNet50 architecture frameworks for Image Classification. In 2021 International Conference on Disruptive Technologies for Multi-Disciplinary Research and Applications (CENTCON) (Vol. 1, pp. 96-99). IEEE.

[7] S. Mascarenhas and M. Agarwal, "A comparison between VGG16, VGG19 and ResNet50 architecture frameworks for Image Classification," 2021 International Conference on Disruptive Technologies for Multi-Disciplinary Research and Applications (CENTCON), Bengaluru, India, 2021, pp. 96-99, doi: 10.1109/CENTCON52345.2021.9687944

[8] G. Azzopardi, N. Petkov, and M. M. van Gerven, "The Dutch Belgian Iris Database," in Machine Learning and Knowledge Discovery in Databases, 2014. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Iris

[9] Bo Chen, Jun Liu, Jiong Liu and Jiacheng Sun, "Flowers Classification via Deep Learning", noiselab.ucsd.edu. [Online]. Available: https://arxiv.org/abs/1610.02391

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in Proceedings of the 25th International Conference on Neural Information Processing Systems, 2012, pp. 1097–1105. [Online]. Available: https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[11] Yongjie Laiacee, "FlowerRecoProjectResNet50", kaggle.com, 2023, [Online]. Available:

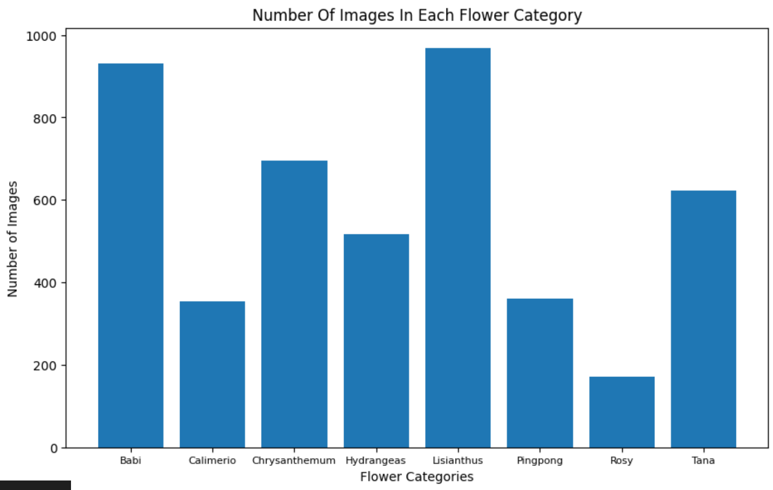
https://www.kaggle.com/code/yongjielaiacee/flowerrecoprojectresnet50

[12] Hasan Emre Bagriyank, "Flowers Recognition With CNN ", kaggle.com, 2023, [Online]. Available: https://www.kaggle.com/code/hasanemrebaryank/flowers-recognition-with-cnn

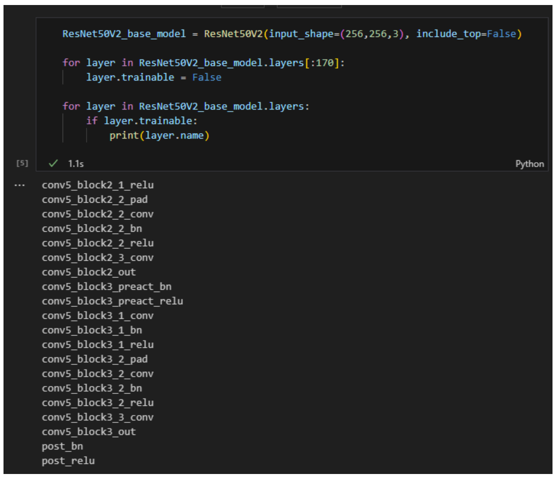
VIII. Appendix

[1]

[2]



[3]



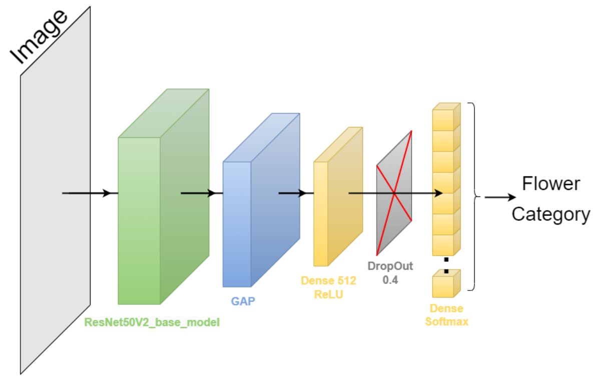
```
ResNet50V2_base_model = ResNet50V2(input_shape=(256,256,3), include_top=False)

for layer in ResNet50V2_base_model.layers[:170]:
    layer.trainable = False

for layer in ResNet50V2_base_model.layers:
    if layer.trainable:
        print(layer.name)
```

```
conv5_block2_1_relu
conv5_block2_2_pad
conv5_block2_2_conv
conv5_block2_2_bn
conv5_block2_2_relu
conv5_block2_3_conv
conv5_block2_out
conv5_block3_preact_bn
conv5_block3_preact_relu
conv5_block3_1_conv
conv5_block3_1_bn
conv5_block3_1_relu
conv5_block3_2_pad
conv5_block3_2_conv
conv5_block3_2_bn
conv5_block3_2_relu
conv5_block3_3_conv
conv5_block3_out
post_bn
post_relu
```

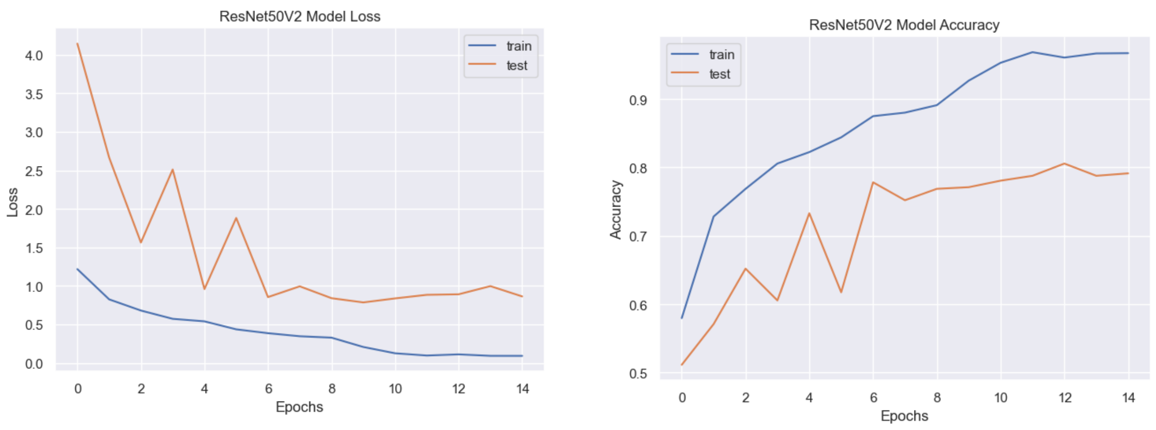
[4]



[5]

Model	Accuracy	Precision	Recall	F1-Score
CNN	0.6758	0.6792	0.6758	0.6769
ResNet50V2	0.7712	0.7795	0.7712	0.7713
VGG19	0.7402	0.7459	0.7402	0.7409

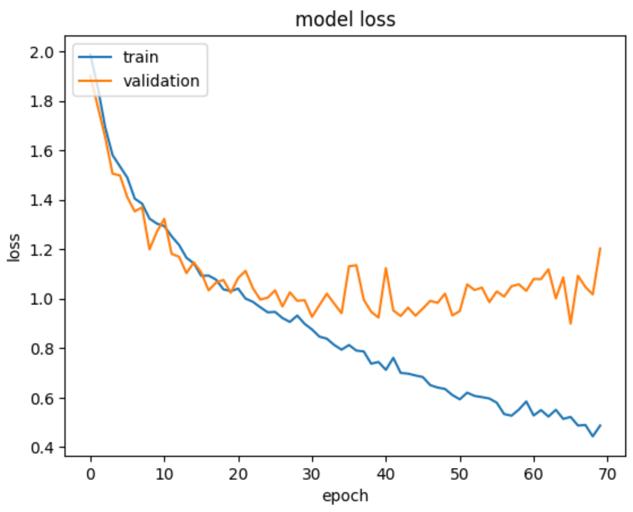
[6]

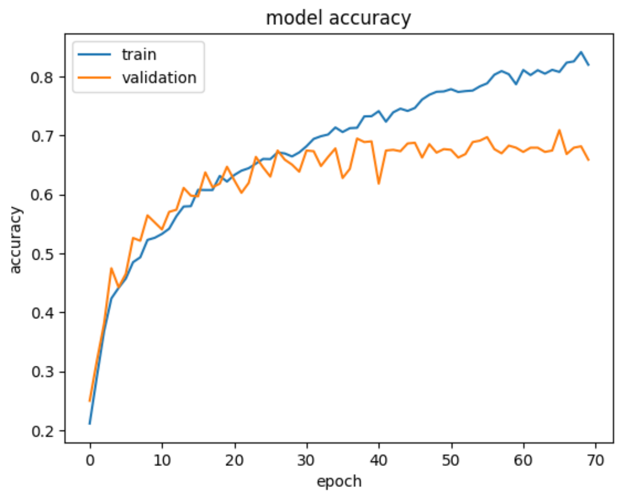


[7]

Model	Accuracy	Recall	Precision	F1-Score
VGG16	0.6591	0.6339	0.7016	0.6655
Siamese	0.6160	0.5145	0.6783	0.5838

[8]





[9]

Model	Our Model	Researched Model[11]	Researched Model[12]
Architecture	ResNet50V2 base model (freeze 170 first layers) without weights	ResNet50 base model (freeze all layers) with weights	CNN (no base model)
Training and Optimization	Batch size: 64 Epoch: 70 Optimizer: Adam	Batch size: 64 Epoch: 10 Optimizer: not provided	Batch size: 32 Epoch: 50 Optimizer: Adam(learning_rate=0.001)
Data Augmentation	Rotation: 10 degrees Zoom: 0.1 Horizontal shift: 0.2 Vertical shift: 0.2 Horizontal flip	Shear range: 0.2 Zoom: 0.2 Horizontal flip	Rotation: 10 degrees Zoom: 0.1 Horizontal shift: 0.2 Vertical shift: 0.2 Horizontal flip
Callbacks	Stop Early + ReduceLR	None	None
Metrics	Loss: 0.0929 Accuracy: 0.9672 Val_loss: 0.8644 Val_accuracy: 0.7914	Loss: 0.2146 Accuracy: 0.9271 Val_loss: 0.2737 Val_accuracy: 0.9062	Loss: 0.2599 Accuracy: 0.9035 Val_loss: 0.7932 Val_accuracy: 0.7894