SnapNLearn

CS3244 Group 2

Chu Qinghao (A0158644W), Maria Salazar (A0179899R), Ong Jing Yin (A0131465L)

Tan Zheng Wei (A0150739B), Shen Yichen (A0091173J), Zi Ying Fan (A0179993B)

Abstract

Our aim is to improve the current lack of knowledge and appreciation for biodiversity in Singapore. We plan to do this by educating users on select plant species, helping them to identify the plants through a mobile application capable of image recognition using machine learning techniques. We will discuss our choice of machine learning technique, Convolutional Neural Network (CNN), and how it fits the requirements of our application. This report contrasts the use and results of three CNN models namely: MobileNet, DenseNet121 and InceptionResnetV2 in the context of our application requirements. We achieved good performance for all 3 models in our testing.

1. The Application

1.1 Context

Singapore is known as the garden city and hosts a wide variety of plant life. According to NParks, there are more than 2000 species of plants in Singapore! However, most Singaporeans do not know much about them and pay little attention to nature. We might appreciate the landscaped look of the plants, but do not know much about the individual species. We have all been to the famous botanic gardens where every plant is labeled and helpful informational signs are available, but these are unavailable elsewhere. If someone saw an interesting native plant on the street or in a nature reserve, they have no way of accurately identifying it without the help of an expert.

To address this problem and increase the appreciation for the biodiversity around us, we propose an interactive, ML-driven mobile application where by simply snapping a picture, the application identifies the plant species and gives the user information about the plant. Our application is designed as a learning tool for all ages which takes the informational signs that one would find in the botanic gardens and brings them all over Singapore.

Because of the huge number of species available, we decided to focus on flowers as it is expected that users will be most interested in flowering plants. Moreover, plant species are more easily identifiable by their flowers, as opposed to branches or leaves.

We expect that the application would be of most interest to schools and government agencies such as NParks. Our application would be an engaging tool for field trips, especially for younger students. Ultimately, the hope is that the application will promote a closer relationship between Singaporeans and our natural environment, so that we can coexist in a continually-urbanizing cityscape.

1.2 User interface

Users of our application can use the in-application camera to snap a photo of a flower they see, and press the "Identify!" button to upload the image to our model. Once our model has predicted the species, the application will display a page with its scientific name at the top. On the screen, there will also be a "Learn More" section which we will populate with interesting facts about the plant, such as what conditions they thrive under, etc. This information will be pre-loaded onto the application.

Near the bottom half of the screen, there will be a map with clusters or dots of where other occurrences of the plant have been discovered, based on our user-aggregated data. This can be easily implemented by adding a geotagging feature to associate each user-uploaded image with a location. This crowd-sourced map of the biodiversity in Singapore will help users learn more about different ecologies depending on where a plant is commonly found or inspire them to visit an area if a flower that they particularly like is found there, promoting more active engagement with nature in our city. This crowd-sourced map feature opens up potential avenues for collaboration with National Parks Board's `trees.sg` project, which maps the physical location of tree species in Singapore.

1.3 Machine Learning Model

Convolutional neural networks (CNNs) are the widely recognized standard model for image classification problems. In fact, state-of-the-art CNNs have achieved better-than-human results on the ImageNet Large Scale Visual Recognition Competition (Russakovsky et al. 2015). CNNs use supervised feature learning, which is helpful for image classification because it is extremely difficult to engineer features, and naive features would certainly not work due to the wide variety of images. However, CNN architecture is still an ongoing area of research. Hence, we will experiment with several architectures to determine their suitability for our application.

1.4 Application Requirements

Our application has minimal requirements for the model as the interface is straightforward. Specifically, we only require fast real-time prediction.

Thus, to balance speed and accuracy, our application will choose an architecture depending on the resources available. If Internet access is present, we will send the input to a remote server which can make a fast, accurate prediction with a large, complex model. If it is not available, we will use a lightweight model specifically designed for mobile devices, which will give worse but still acceptable results.

2. Technical Details

2.1 Convolutional Neural Network

CNNs extract features from images and use these features to classify the images. Features refer to sections of an image, represented as 2D arrays of bits, which are common across images from the same category. CNNs are repeated stacked processes of convolution, ReLU, pooling and fully connected layers.

The purpose of convolution layers is to identify features in an image. This is done through the use of a filters, which essentially work as a sliding window. As this sliding window moves around the image, the match between the feature and the filtered section is calculated. The match is a value from range [-1, 1], where 1 denotes an exact match and -1 denotes an exact mismatch. At the end of this process, we get a filtered image, which represents how closely each section of the image matches with the feature. This process is then repeated with the remaining features.

After the convolutional layer, a pooling layer is used to reduce images to a quarter of their original size without losing important information. This process involves a sliding window which keeps only the maximum value in each window.

Rectified Linear Units (ReLU) remove all negative values by replacing them with 0.

Fully connected layers are attached at the end of the CNN process. In the fully connected layers, weighted votes are used to assign probabilities of how likely the image belongs to each category. The final output, i.e. the prediction of what category the image belongs in, will be the category assigned with the highest probability.

The CNN is trained via back-propagation. Initial values are set randomly and a set of training data with labeled images is fed to the CNN. The CNN makes a prediction of each imageâĂŹs category and the features / weights are adjusted accordingly for every correct or wrong prediction.

2.2 Architecture

We identified three CNN architectures of interest for this project: MobileNet, DenseNet121 and InceptionResnetV2. We deliberately chose these models based on varying size and complexity. Comparisons of these are shown in Table 1.

Name	Parameters	Depth	Multi-Adds
MobileNet	4,253,864	88	569M
DenseNet121	8,062,504	121	—
InceptionResnetV2	55,873,736	572	13.2B

Table 1: Comparison of basic characteristics of the three models. (Mult-Adds for DenseNet121 could not be found)

2.2.1 DenseNet121 DenseNet121 derives its name from the dense connections between its layers (Huang et al. 2017). DenseNet121 has shorter connections between input layers and output layers and due to the way it concatenates features when passing them between layers, has more connections between layers than there would be in a traditional architecture. DenseNets have many advantages including their speed, accuracy, decreased tendency to overfit the data and ease of training. Due to these advantages, DenseNet121 consistently outperforms other existing models, making it a strong candidate for our application.

2.2.2 MobileNet MobileNet, released by Google in 2017, is a small, efficient, mobile-first neural network model (Howard et al. 2017). The architecture makes use of depth-wise separable convolution, which separates the standard one-step filtering and combining in convolution into two steps of depth-wise and point-wise convolution, thus removing inter-dependencies between certain terms and achieving both lower computation costs and a smaller size for the model. Computation efficiency is further enhanced through its use of 1×1 convolutions, for which highly optimized implementations exist. The ability of using the neural network directly on a mobile device is enticing for our application, since it means we would no longer need Internet connectivity to classify images on our server. In addition, should we investigate further into how to optimize the application requirements for particular mobile platforms, MobileNet's design allows us to easily adjust the trade-off between latency and accuracy by adjusting the width multiplier and resolution multiplier.

2.2.3 Inception-Resnet-V2 Inception-Resnet-V2 combines residual connections with Inception models, which improves training speed greatly as well as performance (Szegedy, Ioffe, and Vanhoucke 2016). For models without residual connections, there is a limit to the depth of the neural network, beyond which the accuracy will degrade and there will be a higher training error. This degradation is not due to over-fitting. Residual connections work as shortcuts in the network through the use of identity mapping. This allows deeper neural networks to be trained without suffering from the accuracy degradation, and as a result gives a better image recognition performance than Inception-v3 and Inception-v4.

3. Experiments

The code, as well as dataset used for our experiments can be found at https://github.com/qinghao1/cs3244.

Common Name	Species Name	No. training images
Lipstick Plant	Aeschynanthus parvifolius	100
Bougainvillea	Bougainvillea glabra	100
Ginger Lily	Hedychium gardnerianum	100
Hibiscus	Hibiscus rosa sinensis	100
Jasmine	Jasminum sambac	100
Vanda Miss Joaquim	Papilionanthe 'Miss Joaquim'	100
Red Button Ginger	Costus woodsonii maas	82
Frangipani	Plumeria	78
Ixora	Ixora congesta	71
Spider Lily	Lycoris radiata	76

Table 2: Species and distribution of images.

3.1 Dataset

As a proof of concept, we chose to train our model with ten of the most commonly found flowering plants in Singapore. We scraped pictures from Google Images and manually verified the images to impose a set of criteria that reduced noise due to reasons such as images that were photo collages or had prominent watermarks. We ended up with a total of 907 images, with the species and distributions as described in Table 2.

We made sure to achieve a roughly-uniform distribution of images among the species, in order to avoid skewing the predictions as a result of the neural net biasly fitting against particular categories.

The images were split into training and test sets in the ratio of 9:1. The images from the training set were further augmented with the Image Preprocessing module from Keras, which applies various transformations such as rotation, flipping, zooming, shearing, and scaling to the base images, using the parameters reproduced below in Listing 1. By performing such augmentation at random during training, this increases the effective dataset size and helps to prevent the model from overfitting to specific images in the training data.

3.2 Training

All three models of interest have existing implementations in Keras, a high-level framework built on top of TensorFlow. This made it straightforward to train the models on our dataset.

We trained the models using Stochastic Gradient Descent, and we used categorical cross-entropy loss. We used a "Nvidia K80" GPU with 12GB of memory, and picked the mini-batch size to be the maximum that could fit into memory, which was 8 images. Training time depended on model but generally took a day until the loss stopped decreasing, at which point we stopped training.

```
datagen = ImageDataGenerator(
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    rescale=1./255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    vertical_flip=True,
    fill_mode='wrap'
)
```

Listing 1: Parameters used for image augmentation.

4. Results

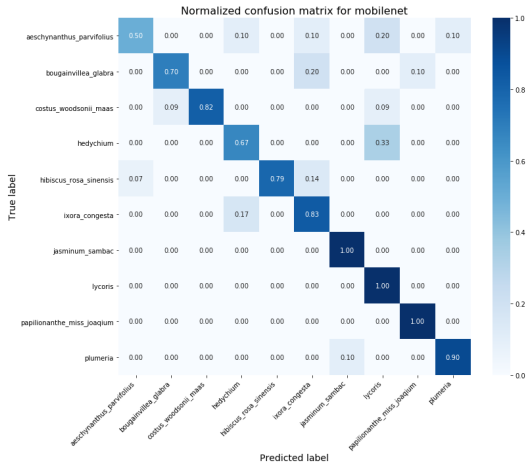
4.1 Analysis

Model	Accuracy	Precision	Recall	F1-Score
MobileNet	0.80	0.85	0.80	0.81
DenseNet121	0.84	0.86	0.84	0.84
InceptionResNetV2	0.88	0.89	0.88	0.88

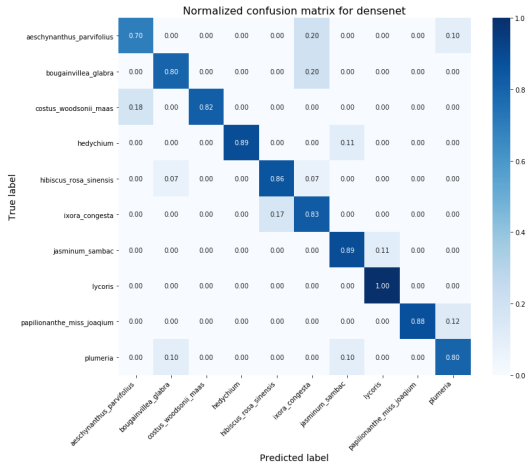
Table 3: Abbreviated Results

The results from our tests are shown in Table 3 as confusion matrices and detailed metrics are in Figure 2. We use 4 evaluation metrics here: accuracy, precision, recall and F1-score which is the harmonic mean of precision and recall. The precision, recall and F1-scores are the prevalence-weighted macro-average scores from the 10 classes.

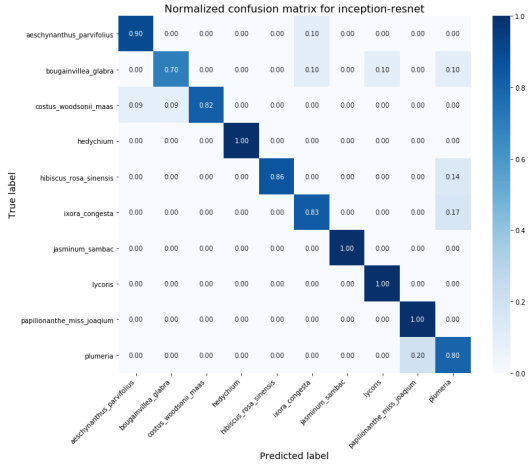
It is clear from the results that the performance of models over all metrics is in the order Mobilenet < DenseNet < Inception-ResNet. For example, both F1-Score and Accuracy for Inception-ResNet are about 10% better than Mo-



(a) Confusion matrix for our ten species based on the trained MobileNet model.



(b) Confusion matrix for our ten species based on the trained DenseNet model.



(c) Confusion matrix for our ten species based on the trained Inception-ResNetV2 model.

Figure 1: Confusion matrices for our three models.

	precision	recall	f1-score	support
aeschynanthus_parvifolius	0.83	0.50	0.62	10
bougainvillea_glabra	0.88	0.70	0.78	10
costus_woodsonii_maas	1.00	0.82	0.90	11
hedychium	0.75	0.67	0.71	9
hibiscus_rosa_sinensis	1.00	0.79	0.88	14
ixora_congesta	0.50	0.83	0.62	6
jasminum_sambac	0.90	1.00	0.95	9
lycoris	0.40	1.00	0.57	4
papilionanthe_miss_joaqium	0.89	1.00	0.94	8
plumeria	0.90	0.90	0.90	10
avg / total	0.85	0.80	0.81	91

(a) Metrics for the MobileNet model.

	precision	recall	f1-score	support
aeschynanthus_parvifolius	0.78	0.70	0.74	10
bougainvillea_glabra	0.80	0.80	0.80	10
costus_woodsonii_maas	1.00	0.82	0.90	11
hedychium	1.00	0.89	0.94	9
hibiscus_rosa_sinensis	0.92	0.86	0.89	14
ixora_congesta	0.50	0.83	0.62	6
jasminum_sambac	0.80	0.89	0.84	9
lycoris	0.80	1.00	0.89	4
papilionanthe_miss_joaqium	1.00	0.88	0.93	8
plumeria	0.80	0.80	0.80	10
avg / total	0.86	0.84	0.84	91

(b) Metrics for the DenseNet model.

	precision	recall	f1-score	support
aeschynanthus_parvifolius	0.90	0.90	0.90	10
bougainvillea_glabra	0.88	0.70	0.78	10
costus_woodsonii_maas	1.00	0.82	0.90	11
hedychium	1.00	1.00	1.00	9
hibiscus_rosa_sinensis	1.00	0.86	0.92	14
ixora_congesta	0.71	0.83	0.77	6
jasminum_sambac	1.00	1.00	1.00	9
lycoris	0.80	1.00	0.89	4
papilionanthe_miss_joaqium	0.80	1.00	0.89	8
plumeria	0.67	0.80	0.73	10
avg / total	0.89	0.88	0.88	91

(c) Metrics for the Inception-ResnetV2 model.

Figure 2: Performance Metrics for our three models.

bilenet. We believe that this is due to the increase in filter size and number of layers in bigger CNNs which allow them to use more image features at more levels.

Nevertheless, even the smallest model gives good results, with 0.80 accuracy and 0.81 F1-score. Moreover, Inception-Resnet is roughly 10 times larger in size (by number of parameters) and takes 20 times as many multiplication-accumulator operations. This means that prediction times will be an order of magnitude slower for Inception-Resnet compared to Mobilenet. Even though our own testing showed that running a prediction using the biggest model on a batch of 16 images only takes a fraction of a second, given that our application is mobile-based and that mobile AI processors are already being seen in high-end devices, MobileNet might actually be a better choice for offline predictions, despite its lower performance.

However, usage of the two are not mutually exclusive. We can always host a large Inception-Resnet model on a server, and include a small MobileNet model in the application itself. The application can then make use of the best model it can based on network conditions and have

the best of both worlds.

Interestingly, while some individual classes have better (or worse) performance across all the models, other classes perform distinctly better (or worse) in one model compared to the rest, even though the training and test datasets are identical. We can intuitively understand that some classes are more "distinctive" than others and thus easier to classify for all models, but it is not clear why some classes perform differently in one model.

For example, an obvious trend across all three models is that Aeschynanthus parvifolius and Bougainvillea glabra tend to be classified as Ixora congesta. We attribute this higher variance in the appearance of Ixora itself. Figure 3 shows two of the images of Ixora within our dataset. It is rather obvious that they are somewhat different. This variance in the feature of the flower itself might cause the trained model to treat it more "generally", thus causing a bias towards selecting it as the most probably prediction. An example of a misclassified Bougainvillea image is shown in Figure 4.

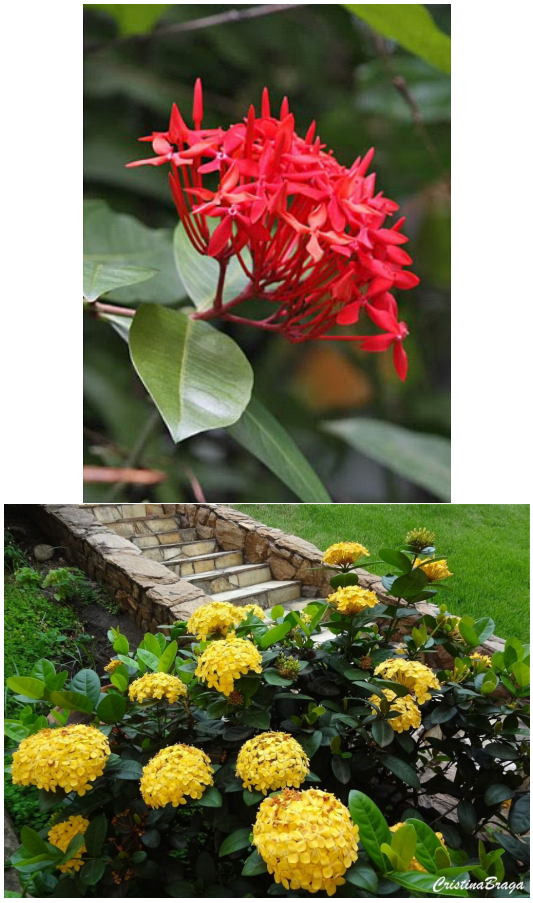


Figure 3: Two photos of Ixora congesta from our dataset. As we can see, there are variants of this plant that are quite different in terms of physical features.

Figure 4: A photo of Bougainvillea glabra that was classified as Ixora congesta.

4.2 Discussion

Our experiments have several factors that limit their relevance to real-world image classification. Firstly, we used a roughly equal distribution of classes which might not be representative of the real world. Secondly, we chose professional-shot and well-lit images which may also not be representative of the actual photos that users will upload to the application. Lastly, we did not account for irrelevant images that users may upload and that do not correspond to any of our classes.

We could also have tried some other methods to improve our accuracy, such as image preprocessing to standardize the images before feeding them into our model. Nevertheless, we felt the results demonstrate that the simple approach is already strongly viable without modifications.

5. Conclusion

Although prior research seemed to indicate that all three models as described above could be appropriate for our proposed application, through the training and experiments, we determined that for our application a combination of MobileNet and Inception-Resnet is workable and will provide the best user experience. In this project, we did not focus on brainstorming novel improvements to the chosen models since their performance was sufficient for the complexity of our proposed application.

Overall, the project was a valuable learning experience for our team since this was the first time delving into a machine learning project for almost all of us. Although we divided up the main responsibilities such as pulling data, coding, analysis, and writing the report according to each team member's strengths in order to make best usage of our manpower, we were all able to learn from the intricacies of this process. Specifically, we realized that a machine learning problem is not just about the training, but moreover, gauging the performance on a test set and choosing the most meaningful metrics to analyze our results is also very important. In addition, this project pushed us to research on not only how CNNs work, but also how the most successful CNN models from the last decade, from AlexNet to recent developments such as Resnet-Inception and MobileNet, differ in their architecture and capabilities.

6. Roles and Reflections

Qinghao - I focused on the technical aspects of the project. It was a challenge learning Python and Keras, but I was pleasantly surprised by how well the CNNs tackled the problem. I learnt that modern ML image classification techniques are impressively fast and accurate.

Maria - I was involved in pulling and checking the images we used and also helped write the report. Throughout this process, I really enjoyed gaining a more in-depth understanding of neural networks and getting to compare the techniques, improvements and results of the three different CNN models that we used.

Jing Yin - I was involved in image scraping and cleaning, and helped to write the report. It was interesting to learn how convolutional neural networks work, and how using residual connections enable deeper networks to be trained, thereby improving the accuracy.

Zheng Wei - Besides the general discussions of the project, I was mainly involved in the writing of the report. Despite having learnt the basic concepts of neural networks, this project has been an eye-opener as we are required to go a step further, not just in learning about how convolutional neural networks work, but also in terms of the design of the procedure to train our models. Specifically, the considerations behind choosing a model and how to evaluate the chosen model by choosing the right metrics.

Yichen - For the most part, I took part in discussions, worked on the report, and read up on the different models of CNN that are relevant to this project. To be honest before the project I had no idea what convolutional neural networks were all about. Through the project I was at least able to understand the basics of what make them tick. Since CNNs are a subset of the perceptron classifier taught in class, it was also very exciting to see what was learned in the lectures being applied directly to a real world scenario.

Zi Ying - I was involved in image scraping and cleaning. Apart from this, my greatest contribution was in researching and comparing the various CNN models (2012 onwards.) Therefore, I helped make the decision of which models to try in our project. It was really interesting to read about how models improved upon each othersâĂŹ architecture, for example using average pooling instead of fully connected layers to reduce the number of parameters needed (GoogLeNet vs AlexNet.) I had also looked into how to do transfer learning and fine-tuning on a pre-trained model, although we did not use this in the end. Finally, I helped with writing the report.

References

Alemi, A. 2016. Improving inception and image classification in tensorflow. *Google Research Blog*. Available at https://research.googleblog.com/2016/08/improving-inception-and-image.html.

Deshpande, A. 2016. A beginner's guide to understanding convolutional neural networks. Available at https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/.

Howard, A. G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; and Adam, H. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

Huang, G.; Liu, Z.; Weinberger, K. Q.; and van derMaaten, L. 2017. Densely connected convolutional networks. *Proceedings of the IEEE conference on computer vision and pattern recognition* 1(2):3.

Keras Contributors. 2018a. *Keras Applications*. Available at https://keras.io/applications/.

Keras Contributors. 2018b. *Keras: Deep Learning for humans*. Available at https://github.com/keras-team/keras/.

Rohrer, B. 2016. *How do Convolutional Neural Networks work?* Available at https://brohrer.github.io/how_convolutional_neural_networks_work.html.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. 2015. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision* 115(3):211–252.

Szegedy, C.; Ioffe, S.; and Vanhoucke, V. 2016. Inception-v4, inception-resnet and the impact of residual connections on learning. *Computing Research Repository* abs/1602.07261.