Image Classification on Oxford 17 Flowers Dataset Using Convolutional Neural Network

YUNG Yiu Yin
BSc(Statistics)
u3568933@connect.hku.hk
3035689336

KWOK Ka Ho
BASc(Applied AI)
ivankwok@connect.hku.hk
3035689142

LO Sai Kwan
BASc(Applied AI)
u3568618@connect.hku.hk
3035686188

CHEUNG Hiu Tung
BASc(Applied AI)
irenecht@connect.hku.hk
3035694525

HO Tik Hang
BEng(CS)
ivan0901@connect.hku.hk
3035688502

1. Introduction

Image classification has been one of the most popular topics in computer vision. It is the study of classifying objects in an image. Convolutional Neural network (CNN), invented in recent years, has achieved major success in accurately classifying images. Composed of multiple convolutional layers, fully connected layers, and pooling layers, a CNN can capture the details of an image, then recognize and classify them. Many CNN architectures are capable of recognizing objects with high accuracy [11]. While LeNet-5 was introduced to classify handwritten digits in 1998, architectures such as VGG [10], AlexNet [12], ResNet [1], DenseNet [3] and NASNet [12] were built to further improve the robustness of CNNs in recent years [2, 7]. This project attempts to utilize different CNN models to classify the type of flowers in a set of images.

1.1 Dataset Description and Problem Definition

In this project, we would like to maximize the accuracy in classifying the flowers in an image. The training dataset is based on the *Oxford 17 Flowers Dataset* with a total of 1360 images, comprising 80 images in each of the 17 categories of flowers [7]. We will split the dataset into the training set, validation set, and test set with a ratio of 8:1:1, training set would be used for model training, the validation set would be used for hyperparameter tuning and the final performance is measured with the test set. It is believed that such an arrangement would help our model achieve better performance and avoid overfitting.

2. Related Work

2.1 Application of CNN

Healthcare is one of the areas where CNN can be applied. Image recognition on MRI and X-ray can replace human eye detection, with the potential of higher accuracy. The detection of abnormalities requires a large dataset, to identify the differences between them. It can potentially save lives, which is the main purpose of the healthcare industry.

Another application of CNN is facial recognition. Facial expressions, features, and poses can be recognized. In social media, facial recognition can help create different filters. For example, filters can generate accessories or apply makeup on human faces. In addition, face recognition can be used to verify the identity of a person. More security protocol involves face recognition, such as approval of bank transaction.

3. Methodologies

This section discusses our modifications to the flower dataset and the architecture of our models. These changes are made accordingly to fit the specification of our project.

3.1 Data Preprocessing

To reduce overfitting and increase the quantity of data, we augmented the dataset through a series of augmentation techniques. In each of the classes, the 64 original training images resized into (224, 224) will be used, while we expanded the dataset to two times the original amount of data by creating 64 more augmented images with random augmentation methods, which includes flipping (vertically and horizontally), cropping, translating, and rotating. Sample augmented images can be referenced from fig. 1. This expands our training set size to 2176 with 128 images in each class. In order to keep the integrity of the data, the scale of translation and cropping will be adjusted to make sure the object is still identifiable. For validation and test data, we only resized the images to (224, 224), all images are also normalized.

Figure 1. Augmented Training Image

To standardize the storing format of the flower dataset, we maintain the data directory structure in the following manner:

```
flower
  |--train
     |--1_daffodil
          |--image_xxxx.jpg
     …
     |--17_pansy
          |--image_xxxx.jpg
  |--val
     |--1_daffodil
          |--image_xxxx.jpg
     …
     |--17_pansy
          |--image_xxxx.jpg
  |--test
     |--1_daffodil
          |--image_xxxx.jpg
     …
     |--17_pansy
          |--image_xxxx.jpg
```

3.2 Model Architecture

We will discuss the rationale behind our model choices in this project.

3.2.1 Baseline Models

We will use the augmented data to train for baseline models, namely VGG, ResNet, and DenseNet. These models are chosen because they have proven to deliver fantastic results in image classification tasks. EfficientNet, with 7 versions, is released in 2019 will also be selected when compared with the aforementioned baseline models. In specific, we will implement ResNet18, ResNet34, VGG16, VGG19, DenseNet121, EfficientNet_b0, EfficientNet_b1, and EfficientNet_b2. The structures, number of parameters, and FLOP of each of the models are listed in table 1 below.

3.2.2 EfficientNet

EfficientNet is a SOTA CNN architecture that makes use of the Compound Scaling method to uniformly scale the depth, resolution, and width of the network, which means that the complexity of the layers can be adjusted according to the dimension of the input image. As illustrated by Figure 2, Compound Scaling expands the depth, width, and resolution in scale.

With the proposed scaling method, EfficientNet achieves tremendous accuracy on CIFAR-100, Flowers, and 3 other transfer learning datasets with fewer parameters than the baseline models. Research has shown that EfficientNet has a much higher classification accuracy in the ImageNet dataset with a similar number of parameters and FLOPs. As shown in Figure 3, all versions of EfficientNet have outstanding accuracy given similar computational resources, for instance, EfficientNet-B1 is 7.6x fewer in the number of parameters compared to ResNet-152, but still gives 1.3% higher top-1 accuracy on the Imagenet dataset [13]. In light of these reasons, this SOTA model will be investigated to see if it could perform well in the flower classification task.

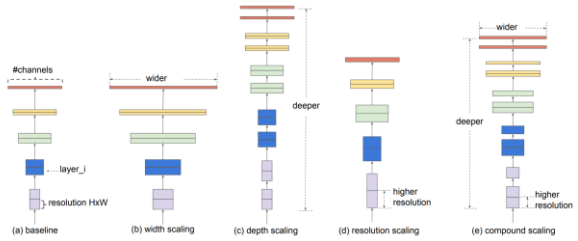

Figure 2. Model scaling in EfficientNet, (a) baseline network, (b)-(d) conventional scaling methods, (e) compound scaling in EfficientNet [13]

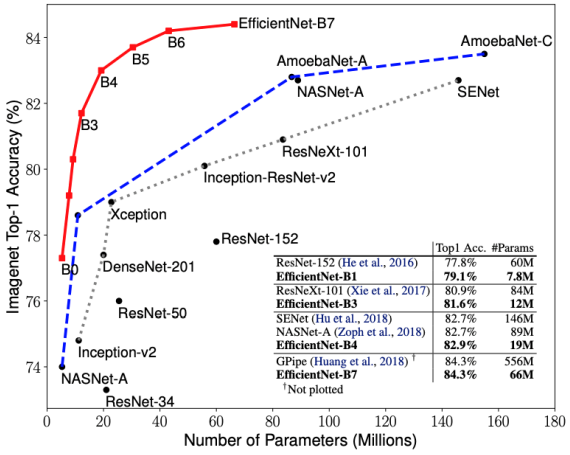

Figure 3. Model Size vs. ImageNet Accuracy

Model names	ResNet-18	ResNet-34	VGG16	VGG19	DenseNet121	EfficientNet b0	EfficientNet b1	EfficientNet b2
	7×7, 64, stride 2		[3×3, 64] ×2 2×2 max pool, stride 2		7×7, 64, stride 2	3×3, 32, stride 2	3×3, 32, stride 2	3×3, 35, stride 2
	3×3 max pool, stride 2		[3x3, 128] ×2 2×2 max pool, stride 2		3×3 max pool, stride 2	3×3, 16 MBConv1	[3×3, 16 MBConv1] ×2	3×3, 118 MBConv1
	$\begin{bmatrix} 3x3,\ 64 \\ 3x3,\ 64 \end{bmatrix}$ ×2	$\begin{bmatrix} 3x3,\ 64 \\ 3x3,\ 64 \end{bmatrix}$ ×3	[3×3, 256]×3 2×2 max pool, stride 2	[3×3, 256]×4 2×2 max pool, stride 2	$\begin{bmatrix} 1x1,\ 256 \\ 3x3,\ 256 \end{bmatrix}$ ×6 1×1, 128 2×2 average pool, stride 2	[3×3, 24 MBConv6]×2	[3×3, 24 MBConv6]×3	[3×3, 26 MBConv6]×2
	$\begin{bmatrix} 3x3,\ 128 \\ 3x3,\ 128 \end{bmatrix}$ ×2	$\begin{bmatrix} 3x3,\ 128 \\ 3x3,\ 128 \end{bmatrix}$ ×4	[3×3, 512]×3 2×2 max pool, stride 2	[3×3, 512]×4 2×2 max pool, stride 2	$\begin{bmatrix} 1x1,\ 512 \\ 3x3,\ 512 \end{bmatrix}$ ×12 1×1, 256 2×2 average pool, stride 2	[5×5, 40 MBConv6]×2 [3×3, 80 MBConv6] ×3	[5×5, 40 MBConv6]×3 [3×3, 80 MBConv6] ×4	[5×5, 44 MBConv6]×2 [3×3, 88 MBConv6] ×4
	$\begin{bmatrix} 3x3,\ 256 \\ 3x3,\ 256 \end{bmatrix}$ ×2	$\begin{bmatrix} 3x3,\ 256 \\ 3x3,\ 256 \end{bmatrix}$ ×6	[3×3, 512]×3 2×2 max pool, stride 2	[3×3, 512]×4 2×2 max pool, stride 2	$\begin{bmatrix} 1x1,\ 1024 \\ 3x3,\ 1024 \end{bmatrix}$ ×24 1×1, 512 2×2 average pool, stride 2	[5×5, 112 MBConv6] ×3 [5×5, 192 MBConv6]×4	[5×5, 112 MBConv6]×4 [5×5, 192 MBConv6]×5	[5×5, 123 MBConv6]×4 [5×5, 211 MBConv6]×5
	$\begin{bmatrix} 3x3,\ 512 \\ 3x3,\ 512 \end{bmatrix}$ ×2	$\begin{bmatrix} 3x3,\ 512 \\ 3x3,\ 512 \end{bmatrix}$ ×3	[4096-d fc]×2 17-d fc softmax		$\begin{bmatrix} 1x1,\ 1024 \\ 3x3,\ 1024 \end{bmatrix}$ ×16	3×3, 320 MBConv6	[3×3, 320 MBConv6] ×2	3×3, 352 MBConv6
	Average pool, 17-d fully connected layer (fc), softmax					1×1 Conv & average pool, 17-d fc, softmax		
FLOPs	7.2×10^9	1.5×10^{10}	6.2×10^{10}	7.9×10^{10}	1.2×10^{10}	7.0×10^9	1.2×10^{10}	1.3×10^{10}
#params	1.2×10^7	2.1×10^7	1.3×10^8	1.4×10^8	7.0×10^6	1.4×10^7	2.7×10^7	2.9×10^7

Table 1. Baseline network structures

3.3 Configurations

We intend to further improve the model capability by exploring different configurations and components of our model.

3.3.1 Batch Normalization

There are numerous normalization methods, and Batch Normalization is one of the most popular ones. While the loss surface can be smoothened by Batch Norm [9], it may not accurately represent the distribution when the batch size is small. We will investigate whether using Batch Norm could improve the accuracy.

3.3.2 Hyperparameter Tuning

To optimize the training process, we performed hyperparameter tuning to each of the baseline models. Grid Searching is applied with the learning rate grid $[10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}]$, and batch size grid $[4, 32, 64]$. The training goes for 50 epochs and Adam Optimizer is adapted in the VGG, DenseNet, and EfficientNet training, while SGD with momentum of 0.9 is adapted in ResNet training. To reduce overfitting, lr scheduler is used throughout all the model training, the learning rate reduces for every 13 epochs with the gamma set to 0.1.

3.3.3 L1/L2 Regularization

L1 and L2 regularization are common techniques used to lower the complexity of model, and hence avoid overfitting. The regularization term of L1 is the sum of absolute values of the weight matrix, while that of L2 is the sum of squared values of the weight matrix. The regularization term is then multiplied by alpha and added to the loss function. By adding regularization terms, models of higher complexity are penalized. Noise in the training data is less emphasized.

4. Evaluation

Since there are only 17 categories, the top 1 accuracy will be used to evaluate the performance of the different models. It measures the proportion of correct top-1 predicted labels.

4.1 Hyperparameters Effect

We aim to compare the impact of learning rates and batch sizes on the baseline models.

4.1.1 Effect of Learning Rate

To achieve the best learning rate for our baseline models, we evaluated the model performance over different learning rates, which vary over $[10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}]$.

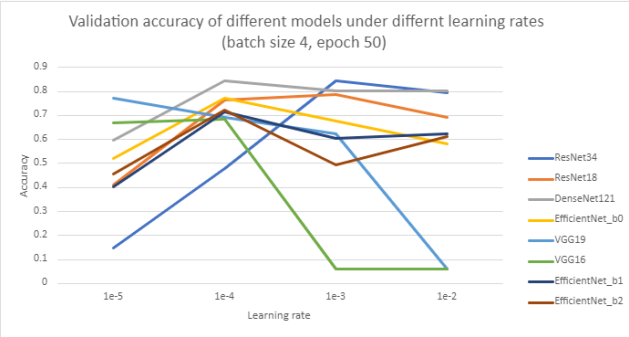

Figure 4. Comparison of performance of different models under different learning

From Figure 4, we could observe that for DenseNet and EfficientNet have the best learning rate at around 10^{-4}. However, for ResNet18 and ResNet34, their best learning rate might lie at around 10^{-3}. This means that a relatively larger learning rate is needed for ResNet. For VGG16 and VGG19, their optimal learning rate are at around 10^{-5}, and their performance deteriorated a lot when the learning rate increased, indicate that a smaller learning rate is needed for VGG.

4.1.2 Effect of Batch Size

We have trained the models with 3 different batch sizes $[4, 32, 64]$, under the same controlling condition, we adopted the learning rate that gives the best performance on each of the models, and based on the learning rate, we trained the models with different batch size to observe the effect of it.

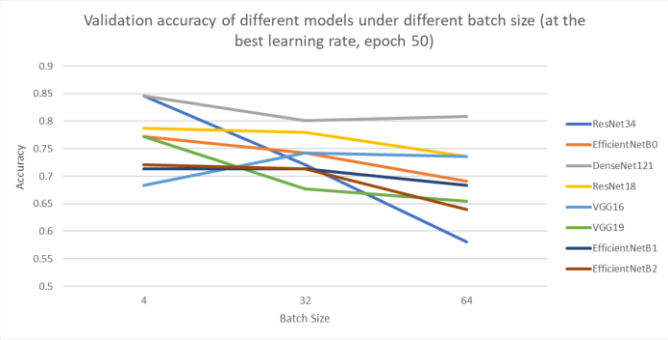

Figure 5. Comparison of performance under different batch sizes

From figure 5, we could notice that the performance of ResNet18, EfficientNet_b0, EfficientNet_b1, EfficientNet_b2, VGG19, and ResNet34 decreases over batch size. On the other hand, the performance of VGG16 is optimal at batch size 32, while a larger batch size does not deteriorate a lot. DenseNet121 has the highest performance at batch size 4. The performance decreases a lot when batch size increase to 32, while it improved slightly when we further tune up the batch size to 64. More experiments could be conducted to test the model sensitivity over batch size.

4.2 Results of Baseline Models

Four convolutional neural networks, ResNet, DenseNet, VGG, and EfficientNet are used as the baseline models. The Preliminary results are generated for each model with their best configuration after hyperparameter tuning with 50 epochs. Their top 1 validation accuracies are shown and compared below. Further improvements will be discussed in Sec. 5.

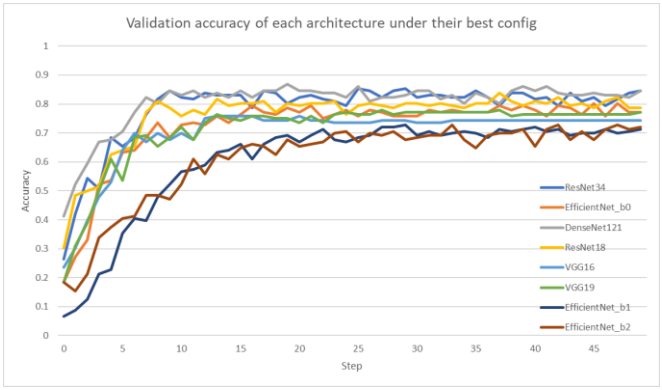

Figure 6. Comparison of the best performance of different architectures

From Figure 6, we could observe most of the final validation accuracies of the models lies between 0.75 – 0.85, while ResNet34 and DenseNet121 are the top-performing models, with **84.56%** and **86.03%** validation accuracy. All of the models converged after 50 epochs, meaning that 50 epochs are adequate and suitable for our training.

Out of our expectation, although EfficientNet has tremendous performance on ImageNet classification tasks, its capability of flower classification is below average when compared to the other baseline models. Though EfficientNetb0 achieved **80.15%** accuracy, EfficientNetb1 and b2 have the lowest accuracy among all the models, with **72.79%** accuracy. More discussion of EfficientNet will be introduced in Section 5.3.

5. Improvements and future work

5.1 Effect of Batch Normalization

As mentioned in section 3.3, we have investigated the effect of batch normalization on the model performance. We selected VGG16 as our candidate of investigation and implemented VGG16 and the same model with batch normalization, setting both models with the learning rate 10^{-4} and batch size of 4 to observe the effect.

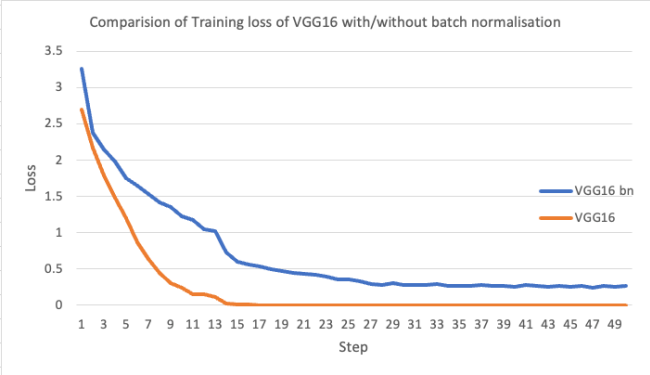

Figure 7. Training Loss of VGG16 and VGG16_bn

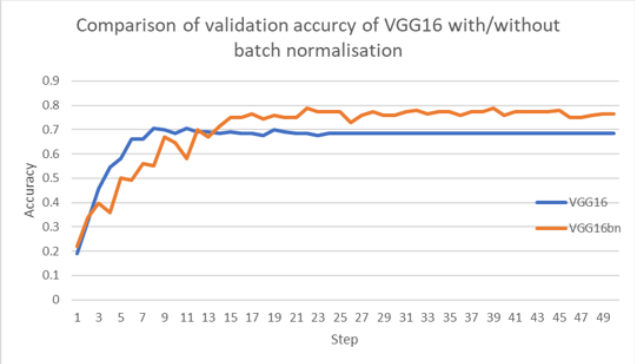

Figure 8. Validation Accuracy of VGG16 and VGG16_bn

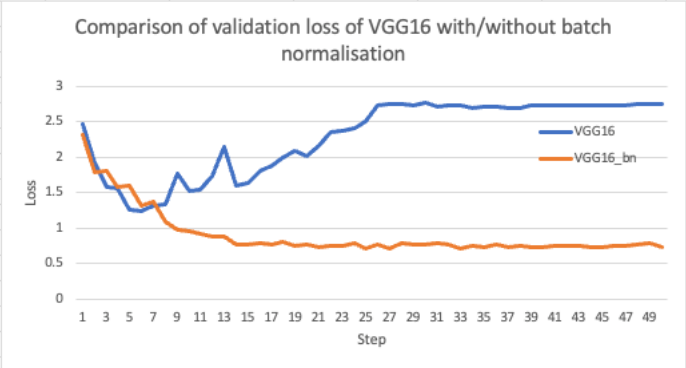

Figure 9. Validation Loss of VGG16 and VGG16_bn

In figure 7, the problem of overfitting is observed in VGG16, where the training loss approaches 0 rapidly in the first 15 epochs and sticks to 0 after that. From figure 9, we observe that batch normalization has significantly reduced the overfitting issue existing in the VGG16 setting, the validation loss of VGG16_bn reduces smoothly, while VGG16 first goes down but rises again rapidly. It shows that batch normalization has greatly stabilized the training performance and reduced the number of epochs for the loss to converge, which could be validated in figure 8, that VGG16_bn consistently outperformed VGG16 in accuracy.

5.2 L1/L2 Regularization

L1 and L2 Regularization can alleviate the problem of overfitting. Models of higher complexity are discouraged with regularization. While these are two commonly used techniques to penalize the large values in the weight matrix [6], we add L1 or L2 regularization to the best model among the baseline models, which is DenseNet121, and analyze the effects of the two types of regularization. 1e-4 is being used as the λ value of L1 and L2 regularization.

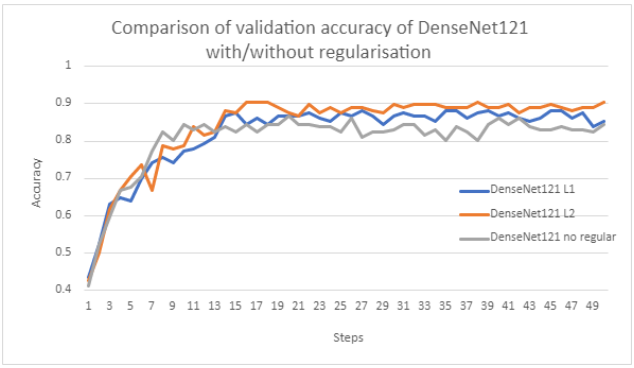

Figure 10. Validation Accuracy of DenseNet121 under regularization effect

Figure 10 shows that the validation accuracy of DenseNet121 is higher with regularization, although it converges slightly faster without regularization. In particular, DenseNet121 with L2

regularization reached **90.44%** accuracy, followed by DenseNet121 with L1 regularization with **88.24%** accuracy, which outperformed the base DenseNet121 with only **86.02%** accuracy, this matches our expectation. Since L1 regularization turns some weights to 0 and L2 does not, fewer parameters in the neural network may result in underfitting.

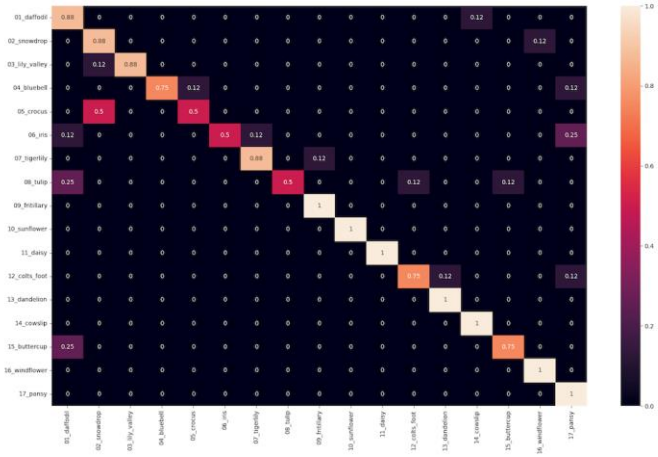

Figure 11. Confusion Matrix of the best performing model – DenseNet121 with L2 regularization

We then generate the confusion matrix of our best-performing model among all experiments – DenseNet121 with L2 regularization. From Figure 11, we can observe that the model could accurately classify most of the flower categories, except Crocus, Iris, and Tulip, only half of them are correctly classified by our model. Noticeably, half of the Crocus images are misclassified as Snowdrop, which is worth noticing and could be further investigated on how to improve our model performance on these categories.

We also tried to run regularization on ResNet34. The result is quite interesting that at the beginning, adding regularization (both L1 and L2) deteriorates the validation accuracy. After tuning the lambda, the L2 regularization can give a higher accuracy, but L1 still performs poorly. The results match the fact that L2 regularization gives better accuracy than L1, and L1 sometimes works worse than the one without regularization. One possible reason may be the L1 regularization sparse some important parameters to 0. This problem is especially significant when the number of data is small, especially in this project. The performance of L1 is unstable when we do not have enough data. In general, L2 regularization can improve accuracy but L1 may not.

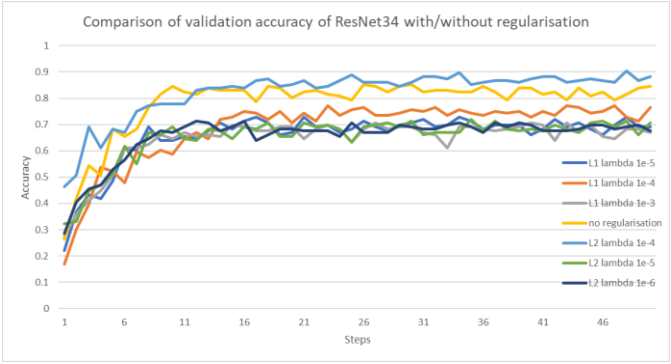

Figure 12. Validation Accuracy of ResNet34 under regularization effect

5.3 Discussion on EfficientNet

We investigated the possible reasons of EfficientNet's inferior performance on the flower classification task. One possible reason could be the noise existing in our small dataset. Though we have performed data augmentation to expand the number of training images to double, the number of training images is still small when compared to ImageNet1K, with over 12 million training images and 50000 validation images [14], which is incomparable to our data with 2176 training images along with 136 validation images. So, the small dataset might contain more noise which interevent the EfficentNet model training. What's more, the small validation set could easily pose a fluctuating prediction on a model's accuracy. In our dataset, we only have 136 validation data, with 8 images per class, thus a tiny misclassification of images could already pose a significant drop in validation accuracy, for instance, misclassifying 3 images pose a 2.2% drop in validation accuracy. Thus, the result might be more accurate when we add more data for training and validation. A further collection of data with proper data preprocessing could be conducted in the future to improve the representativeness of the flower data.

6. Conclusion

In this project, we performed training on different models and subsequent improvements for image classification tasks on the Oxford 17 Flowers dataset. We begin with several baseline models VGG, DenseNet, ResNet, and EfficientNet using grid search methods to tune the hyperparameters (learning rate and batch size). The best baseline model in our search is DenseNet121 with batch size 4 and learning rate 1e-4. Then we explore how to further improve this model by adding batch normalization and L1, and L2 regularization and the top-1 accuracy further improved from **86.02%** to **90.44%**, which is the best result among our experiments. We also added batch normalization on VGG16 and proved the ability to address the overfitting issue. Considering EfficientNet, the performance is

not as powerful as we thought, more explorations on this model shall be conducted.

Contribution

Kwok Ka Ho
- Built the infrastructure of the model training scripts
- Data Augmentation and Data Preprocessing
- Training EfficientNetB2, DenseNet121, VGG16
- Experiment on the effect of Batch Normalization with VGG16 and VGG16_bn
- Implemented Tensorboard for storing training results

Yung Yiu Yin
- Data augmentation and preprocessing
- Train ResNet34 and EfficientNetB1
- Plot loss and accuracy graphs
- Calculate FLOP and no. of params

Ho Tik Hang
- L1, L2 Regularization Implementation
- Training EfficientNetB0

Lo Sai Kwan
- Experiment on DenseNet121 preliminary results
- Training DenseNet121, VGG19

Cheung Hiu Tung
- Training ResNet18, VGG16
- Presentation Powerpoint Design

References

[1] K. He, X. Zhang, S. Ren, J. Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770-778, 2016

[2] T. He, Z., Zhang, H., Zhang, Z., Zhang, J., Xie, M. Li. Bag of tricks for image classification with convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 558-567

[3] G. Huang, Z. Liu, L. Van Der Maaten, K. Q. Weinberger. Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4700-4708, 2017

[4] A., Krizhevsky, I., Sutskever, G. E., Hinton. "Imagenet classification with deep convolutional neural networks." *Communications of the ACM*, vol *60*(6), pp. 84-90, 2017

[5] B. J., Lei, J. R. Kiros, G. E. Hinton. "Layer normalization." *arXiv preprint arXiv:1607.06450*, 2016.

[6] S. Mazil, J. Iria, "L1 vs. L2 Regularization in Text Classification when Learning from Labeled Features," *2011 10th International Conference on Machine Learning and Applications and Workshops*, 2011, pp. 166-171, doi: 10.1109/ICMLA.2011.85.

[7] M. E. Nilsback, A. Zisserman. *17 Category Flower Dataset*. Available: https://www.robots.ox.ac.uk/~vgg/data/flowers/17/index.html

[8] T. Salimans, D. P., Kingma. "Weight normalization: A simple reparameterization to accelerate training of deep neural networks." *Advances in neural information processing systems*, *29*, 2016.

[9] S., Santurkar, D., Tsipras, A., Ilyas, A., Madry. "How does batch normalization help optimization?". *Advances in neural information processing systems*, *31*, 2018

[10] K. Simonyan, A., Zisserman. Very deep convolutional networks for large-scale image recognition, pp. 1409-1556, 2014

[11] F. Sultana, A., Sufian, P. Dutta. Advancements in image classification using convolutional neural network. In *2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*, 2018, pp. 122-129

[12] B. Zoph, V. Vasudevan, J. Shlens, Q. V. Le. Learning transferable architectures for scalable image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 8697-8710, 2018

[13] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for Convolutional Neural Networks," arXiv.org, 28-May-2019. [Online]. Available: https://arxiv.org/abs/1905.11946.

[14] O. Russakovsky et. al. , Dataset: imagenet-1k, [Online]. Available: https://huggingface.co/datasets/imagenet-1k

Appendix: GitHub Repository and TensorBoard

We maintain a Github repository of the project here and the baseline models used are stored in the model directory. We provide our results with the Tensorboard which could be run on local host