CZ4042 Neural Networks and Deep Learning

Group Project

Group Members:

Name	**Matriculation Number**
Chan Hui De, Elliot	U2021098D
Ong Xin Rui Celestine	U2022737G

Table of Contents

1. Introduction

Our project aims to increase the performance of well-established models by incorporating relatively recent advancements in model architecture and training techniques. Since the introduction of these popular models in the early 2010s, there have been significant strides in Deep Learning, such as the development of attention mechanisms and advanced data augmentation. Thus, by applying these modern innovations to older models, our group seeks to explore whether such integration can lead to improved performances of the older models.

Our group focused on image classification and used The Oxford Flowers 102 dataset (Project F: Flowers Recognition) (PyTorch, n.d.). The three main model groups that we have chosen to explore are VGG Very Deep Convolutional Networks (VGGNet), Residual neural network (ResNet), and EfficientNetV2. After obtaining the model that gave us the best performance, we then made modifications to the model including adding more advanced techniques from deeper models and also incorporating attention mechanisms. Additionally, we studied the effects of data augmentation techniques and using fewer training images on the original model chosen.

2. Review of Existing Techniques

2.1. ResNet

ResNet is a Convolutional Neural Network (CNN) architecture that supports a large number of layers which previous CNN models did not allow for. This is because ResNet is able to address the problem of vanishing gradients by using "skip connections”, meaning the output of a layer is added to the output of a few layers deeper in the network (Datagen, n.d.). These skip connections enable the training of very deep networks by allowing gradients to flow through the network during backpropagation directly.

2.2. VGGNet

VGGNet stacks multiple convolutional layers with small receptive fields of 3x3 filters, followed by max pooling layers, and fully connected layers (Deepchecks, n.d.). The use of smaller filters in consecutive convolutional layers allows the model to have a substantial number of weight layers, and it can learn more complex features at each layer. Furthermore, stacking multiple of such layers deepens the network, enabling it to capture a wide range of features from the input images.

2.3. EfficientNetV2

EfficientNetV2 is a variant of the EfficentNet architecture, whereby it focuses on optimising training efficiency and model scalability (Ried, 2022). It introduces a concept called compound scaling, which uniformly scales the network's width, depth, and input image resolution, while maintaining a balance between these dimensions as the network grows. This allows the model to obtain higher levels of accuracy without the need to compromise accuracy. Compared to EfficientNet, new optimised convolutional blocks such as Fused-MBConv, which fuses the convolutions in the MBConv block for higher efficiency. Additionally, progressive learning is also used, where sizes of the images used are progressively increased. This speeds up the training process and improves the model's performance.

3. Methods Used

We aim to be able to increase the performance of the models in one of two metrics, namely accuracy on test set, or average training time for 1 epoch, while keeping the other as static as possible. For each of our experiments, we only change 1 aspect of the model or perform 1 different training or augmentation technique in order to keep the results fair and accurate. We also performed each test multiple times and took the average in order to minimise any discrepancies that may have resulted due to external factors. For all of our training, we employ the early stopping mechanism based on the accuracy on the evaluation set, this allows us to prevent overfitting while also decreasing the time taken for each experiment.

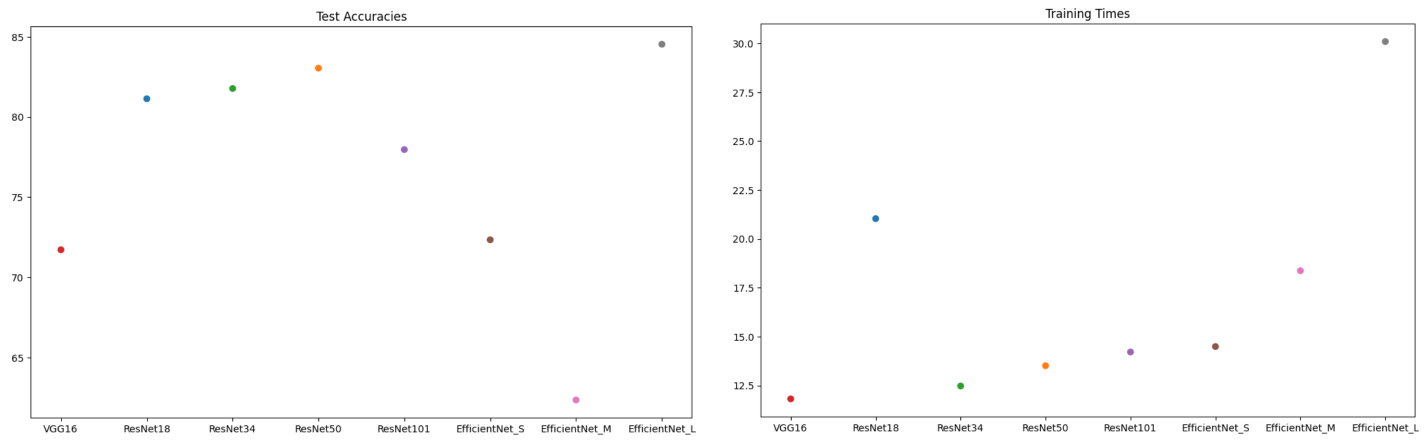


Figure 1: Test Accuracies and Training Times for 8 Models

In our exploratory analysis, we performed transfer learning on 8 models where we only trained a classifier layer (Figure 1). We discovered that ResNet models had generally good performance across the board with low training times. Shallow ResNet models were even able to outperform deeper models like EfficientNetV2_m. Thus, we decided to focus on ResNet34 as it had great performance in our preliminary exploration, with minimal training time and relatively high accuracy as seen from the figure above, which we wanted to explore if it could be further optimised.

Using the ResNet34 model, we performed a total of 3 experiments:

1. Modifying the model

2. Trying advanced data augmentations and transformation
3. Trying different training techniques

3.1. Modifying the Model

We made 2 different modifications to the model. Firstly, we attempted to transplant the bottleneck layer from more complex ResNet models into ResNet34 (Modified Model). We limited this to only replacing the last layer of the model. The bottleneck layer aims to reduce the number of parameters and computations before performing more expensive computations like convolution, in order to maintain or improve the networks' representational capacity (Ruiz, 2018). The bottleneck layer added is almost identical to the one found in ResNet50. The main differences however are due to the different dimensions of the data from layer3, as well as the kernel size. Layer3 in ResNet34 outputs are of 256 dimension with a kernel of (3, 3) while ResNet50 has an output of 1024 dimension and kernel of (1, 1). In order to circumvent this issue, we opted to replace the first layer in our new BottleNeck layer with the initial Conv2d layer in layer4 of ResNet34, specifically a Conv2d layer with an input of 256, output of 512, and a kernel size of (3, 3). By making this adjustment, we will be able to minimise differences between the original BottleNeck layer and our new last layer.

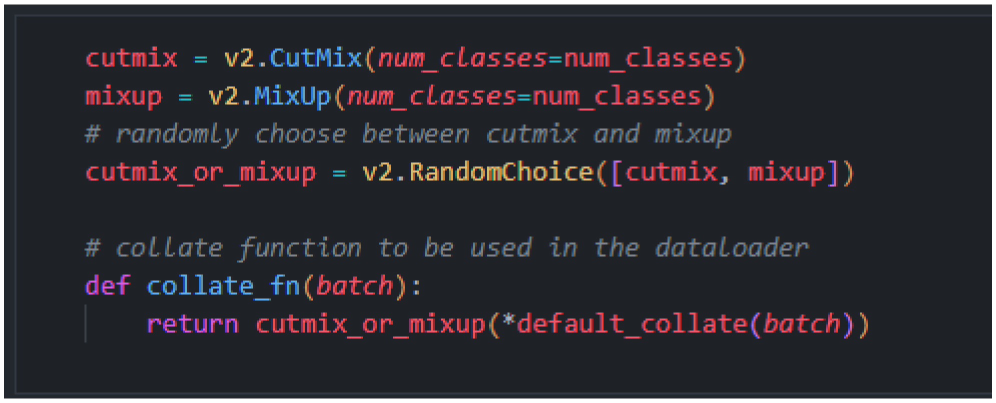
By doing this, our group hoped to allow the ResNet34 Model to handle more complex feature extraction in a less computationally expensive way. Also, the bottleneck design can help maintain a balance between the depth of the network and computational efficiency. By adding these layers to ResNet34, the network could potentially learn more intricate details from the data without a significant increase in computational cost.

Secondly, we incorporated more modern and advanced techniques into the ResNet34 model. We did this by adding attention via Squeeze and Excitation. Squeeze and Excitation is a technique introduced in the paper "Squeeze-and-Excitation Networks" by Jie Hu, Li Shen, and Gang Sun in 2018. Its purpose is to adaptively adjust the weighting of each feature map produced by the convolutional layer, and place more attention on the important features, and less on less important features (paperswithcode, n.d.). Squeezing involves aggregating the global spatial information with a global average pool which reduces spatial dimensions of feature maps to a single value per channel. Excitation involves training a small fully connected network with a sigmoid or ReLU activation function to learn a set of channel-wise weights which represents the importance of each channel to then rescale the original feature maps to emphasise important channels and suppress less important ones. We added Squeeze and Excitation blocks to the end of the first and last layer of the model (SE Model).

By adding SE blocks to the ResNet34 Model, our group hoped that the blocks could complement the skip connections by enabling the network to focus more on important features within each block, enhancing and balancing overall representation of data in the model. Furthermore, SE blocks introduce a mechanism to adaptively recalibrate channel-wise feature responses. This means that the network learns to emphasise more informative features and suppress less useful ones dynamically. When integrated into the model, this could lead to a more focused and effective feature extraction process.

3.2. Advanced data augmentations and transformation

We experimented with applying CutMix and MixUp to our data in order to introduce complexity to the training data. CutMix does this by randomly cutting a portion of an image and combining it with another image by replacing the pixels in the corresponding region (Nath, 2021). On the other hand, Mixup uses a more subtle approach by combining 2 images linearly using a random interpolation factor which determines the proportion of each pixel (Paul, 2021). By using these techniques, we aim to provide regularisation to the training data and introduce additional diversity and variations, which theoretically leads to better generalisation performance.



```
cutmix = v2.CutMix(num_classes=num_classes)
mixup = v2.MixUp(num_classes=num_classes)
# randomly choose between cutmix and mixup
cutmix_or_mixup = v2.RandomChoice([cutmix, mixup])

# collate function to be used in the dataloader
def collate_fn(batch):
    return cutmix_or_mixup(*default_collate(batch))
```

Figure 2: CutMix and MixUp Function Code

For our code in Figure 2, we create a collate function which randomly selects either CutMix or MixUp to use. This collate function will be used by the dataloader and applied to each batch of data. This allows us to introduce even more regularisation via randomness, which can potentially lead to even better generalisation performance (PyTorch, n.d.).

3.3 Using different training techniques

Lastly, we applied a simplified implementation of few shot learning on a pretrained ResNet34 to induce better performance from the model. Few shot learning aims to train the model with a small amount of images per class (Syed, 2018). This can be done with the help of a pretrained model like ResNet where we use a new classifier layer for a specific task. In our experiment, we varied the amount of sample data used for training and compared the test accuracy results for each sample data size. For our experiment, we performed testing on sample sizes 20, 50, 100, 200, 500, 1020. By comparing the test accuracies across the various training sizes, we aimed to uncover insights into the model's adaptability and learning efficiency under varying data constraints. This experiment not only tested the limits of few-shot learning, but also gave us a better understanding of how pretrained models can be effectively utilised and fine-tuned for specific tasks with limited data.

4. Experimentation and Analysis of Results

4.1. Modifying the Model

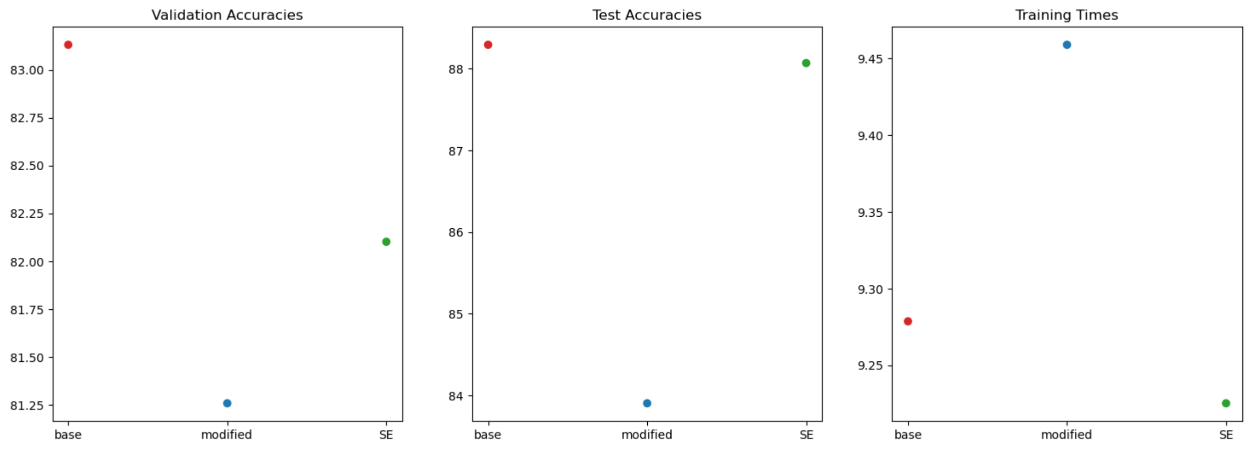
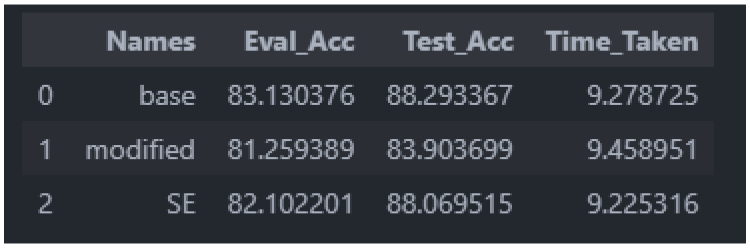


Figure 3: Graphs Showing Accuracies and Training Times for Our 2 Modified Models



	Names	Eval_Acc	Test_Acc	Time_Taken
0	base	83.130376	88.293367	9.278725
1	modified	81.259389	83.903699	9.458951
2	SE	82.102201	88.069515	9.225316

Figure 4: Specific Values of Accuracies and Training Times for Our 2 Modified Models

From our results in Figures 3 and 4, we can see that the modified model did not perform well in either of the 2 categories compared to the base model, while the SE model was able to perform similarly in terms of test accuracy while having an improvement in training times.

We believe that the reason why the modified model had worse performance was due to the increase in model depth and complexity, which might have led to potential overfitting, and also the increase in training time. Another possible reason was due to replacing layer4 completely, we were unable to make use of the pretrained weights for that layer. This could have led to the drop in test accuracy as the pretrained weights are more robust due to being trained on the ImageNET1K dataset which has 1000 classes and 1.2 million training images.

We believe that the SE model was able to improve on the time taken due to the focusing the model on specific areas which it deems to be more important. By reducing the spatial information of feature maps using global average pooling, the model will have a lower computational cost and will be able to reduce the time taken for training.

4.2. Advanced data augmentations and transformation

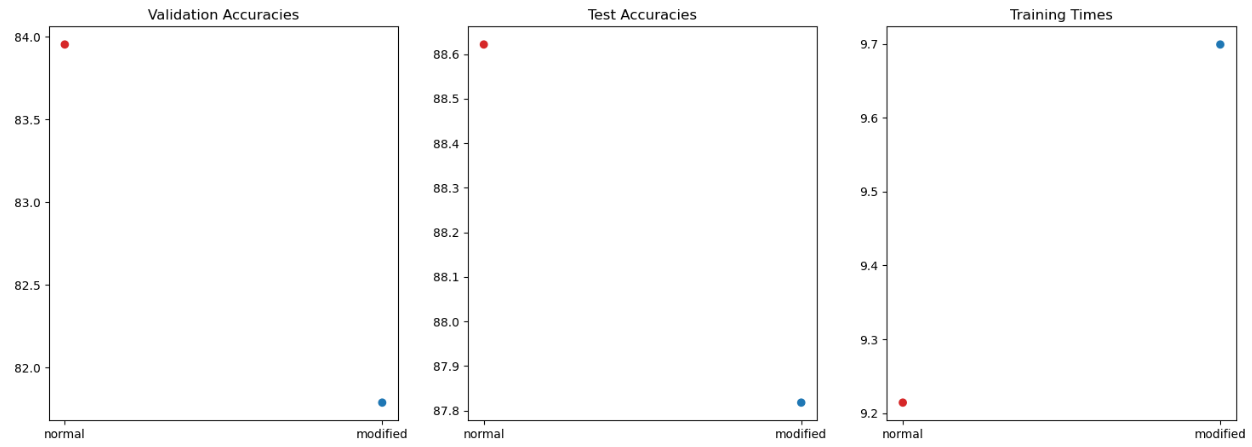


Figure 5: Graphs Showing Accuracies and Training Times for our Base Model versus Model with Data Augmentation

We can see from the results in Figure 5 that performance has dropped for both accuracy and training time when using the CutMix and MixUp technique. This decline could be attributed to the added noise and complexity these methods introduce, which may not align well with the characteristics of the Flowers102 dataset. The Flowers102 dataset has a rather small training set of only roughly 12% of the total data, compared to other datasets like ImageNET1K which has approximately 90% of its data as training data. Regularising the data in our dataset may cause the model to be unable to capture the full diversity and complexity of the data, and could potentially limit the model's ability to discern and learn from the variations and essential features present in the dataset.

4.3 Using different training techniques

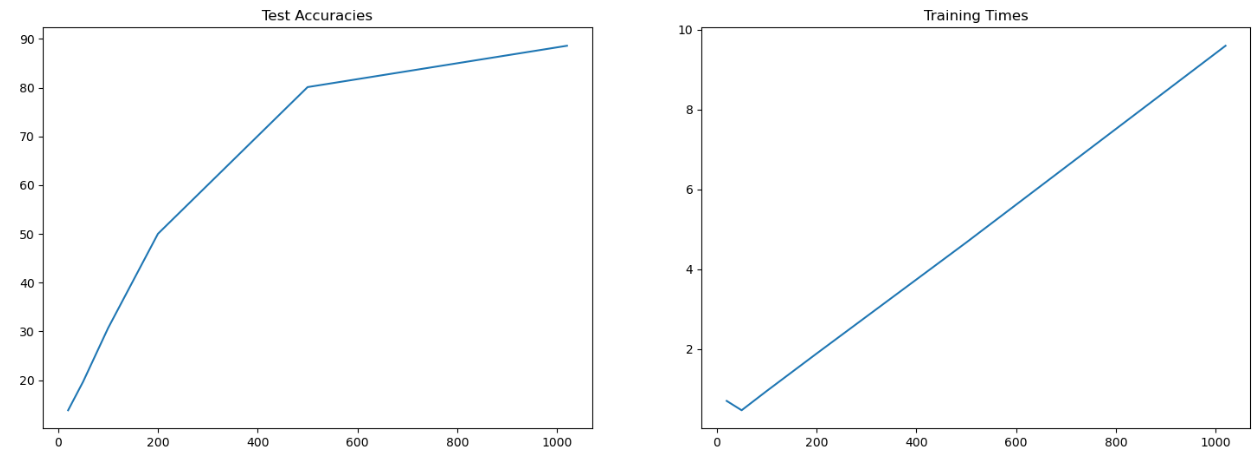


Figure 6: Graph of Accuracies and Training Times for Different Sample Sizes

From the results in Figure 6, we can see that accuracy on the test set increases as the sample size increases, but encounters the problem of diminishing returns where each increment in sample size does not yield the proportionate increase in test accuracy. For instance, increasing the sample size from 200 to 500 yields a much larger jump in test accuracy as compared to the jump from 500 to 1020. This suggests that beyond a certain point, adding more data gives progressively smaller gains in performance, which shows an efficiency threshold in data usage for model training.

Additionally, our group also noted that the training times are directly proportional to the sample sizes. Although our attempts at few-shot learning did not meet our initial expectations in terms of performance, the results still offer valuable insights. In particular, the example where using less than half the sample data (500 versus 1020) results in slightly lower accuracy but half the training time, suggests a potential trade-off between resource allocation and performance. This is useful in scenarios where computational resources or time are constraints, highlighting the value of a balanced approach when training our model. This understanding is crucial, especially in real-world applications where resources are often limited, and decisions must be made to maximise the effectiveness of the available data.

5. Conclusion

In our experiments, we attempted to implement different architectures, data augmentation techniques or training methods to our model in order to increase the performance of our models. Squeeze and Excitation was able to bring about performance improvements in the runtime of the training process by leveraging on newer concepts like attention to improve upon tried-and-tested models. However, apart from Squeeze and Excitation, many of these experiments have led to less favourable results. We believe that this may be due to the dataset used. The Flowers102 dataset is a small sized dataset compared to more modern datasets which tends to have a few million data points to work with (e.g. ImageNET1K: 1.2mil, MovieLens 20M: 20mil). Many of the techniques we employed have been built and optimised for much larger datasets. This could lead to varying performance when applied on a small dataset like the Flowers102 dataset, which might not provide enough information or variation for these advanced methods to exhibit their full potential. All in all, we have learnt the critical importance of aligning model enhancement strategies with the chosen dataset's characteristics. Techniques that show promising results in one context, particularly with large-scale datasets, may not necessarily translate effectively to smaller datasets.

6. Reference List

Datagen. (n.d.). *ResNet: The Basics and 3 ResNet Extensions*. Datagen. Retrieved November 10, 2023, from https://datagen.tech/guides/computer-vision/resnet/#

Deepchecks. (n.d.). *What is VGGNet*. Deepchecks. Retrieved November 11, 2023, from https://deepchecks.com/glossary/vggnet/

Nath, S. (2021, June 8). *CutMix data augmentation for image classification*. Keras. Retrieved November 11, 2023, from https://keras.io/examples/vision/cutmix/

paperswithcode. (n.d.). *Papers with Code - Squeeze-and-Excitation Block Explained*. Papers With Code. Retrieved November 11, 2023, from https://paperswithcode.com/method/squeeze-and-excitation-block

Paul, S. (2021, March 6). *MixUp augmentation for image classification*. Keras. Retrieved November 11, 2023, from https://keras.io/examples/vision/mixup/

PyTorch. (n.d.). *EfficientNetV2 — Torchvision main documentation*. PyTorch. Retrieved November 11, 2023, from https://pytorch.org/vision/main/models/efficientnetv2.html

PyTorch. (n.d.). *Flowers102 — Torchvision main documentation*. PyTorch. Retrieved November 11, 2023, from https://pytorch.org/vision/main/generated/torchvision.datasets.Flowers102.html

PyTorch. (n.d.). *How to use CutMix and MixUp — Torchvision main documentation*. PyTorch. Retrieved November 11, 2023, from https://pytorch.org/vision/master/auto_examples/transforms/plot_cutmix_mixup.html

PyTorch. (n.d.). *ResNet — Torchvision main documentation*. PyTorch. Retrieved November 11, 2023, from https://pytorch.org/vision/main/models/resnet.html

PyTorch. (n.d.). *SqueezeExcitation — Torchvision main documentation*. PyTorch. Retrieved November 11, 2023, from https://pytorch.org/vision/main/generated/torchvision.ops.SqueezeExcitation.html

PyTorch. (n.d.). *vgg16 — Torchvision main documentation*. PyTorch. Retrieved November 11, 2023, from https://pytorch.org/vision/main/models/generated/torchvision.models.vgg16.html

Ried, C. (2022, October 7). *EfficientNetV2: Faster, Smaller, and Higher Accuracy than Vision Transformers | by Arjun Sarkar*. Towards Data Science. Retrieved November 11, 2023, from https://towardsdatascience.com/efficientnetv2-faster-smaller-and-higher-accuracy-than-vision-transformers-98e23587bf04

Ruiz, P. (2018, October 8). *Understanding and visualizing ResNets* | *by Pablo Ruiz*. Towards Data Science. Retrieved November 11, 2023, from https://towardsdatascience.com/understanding-and-visualizing-resnets-442284831be8

Syed, A. H. (2018, April). *Transfer learning and few-shot learning: Improving generalization across diverse tasks and domains*. Abis Hussain Syed. Retrieved November 11, 2023, from https://syedabis98.medium.com/transfer-learning-and-few-shot-learning-improving-generalization-across-diverse-tasks-and-domains-a743781ee357