Implementing and benchmarking an EfficientNetV2 style network

Kurt Perez (N9693441)
MXN442 Assignment 2

This report provides a high-level summary of the *EfficientNetV2: Smaller Models and Faster Training* paper and explores some of the findings within the paper through experiments.

The Github repository containing my experiments is located here:
https://github.com/kurtperez123/MXN442-Assignment

Some of the models I've built are larger than the 100mb limit that Github has for each file. These models are downloadable in this Google Drive:
https://drive.google.com/drive/folders/1AtxXnANCRgR6-5x4hOjaYNMOkqrlLanR?usp=sharing

Note that due to the 7-page constraint, this report excludes some of the intermediary findings presented in the paper.

1.0 Project Background

EfficientNetV2 is a family of convolutional neural networks (CNN) [1]. This family of CNNs are introduced in a paper published in 2021 titled *EfficientNetV2: Smaller Models and Faster Training. The* paper's authors are Mingxing Tan and Quoc V.Le.

Note that the EfficientNet is a family of models published by the same authors [2] and is reviewed in the EfficientNetV2 paper.

1.1 The Research Question

The paper is an attempt to improve training and parameter efficiency in the context of deep learning. Authors emphasise the importance of efficiency by citing models such as GPT-3 [3] stating that while models are becoming more complex and showing remarkable results, training or retraining these models can take a long amount of time. This motivates the authors to identify strategies aimed at improving training time and parameter efficiency.

1.2 The proposed solution

Authors initially examine prior literature related to training and parameter efficiency, progressive training and neural architecture search to understand bottlenecks and identify techniques which could be explored.

The authors identify new strategies to improve model efficiencies:

1. **Utilising a convolutional block known as Fused-MBConv (Fused – Mobile Inverted Bottleneck Convolution).**

   After examining bottlenecks in the original EfficientNet models, the authors determined that:

   - Training with large images is slow.
   - Depthwise convolutions are slow in early layers but are effective in later stages.

   With these in mind, the authors intended to replace some of the MBConv blocks in the original EfficientNet architecture with Fused-MBConv blocks. However, the challenge then becomes finding the right combination of MBConv and Fused-MBConv blocks as replacing all MBConv blocks does not yield optimal efficiency.

   MBConv and Fused-MBConv blocks are shown in [1, Fig.1]. The Depthwise cov3x3 and Conv1x1 in MBConv are replaced a single Conv3x3 layer.

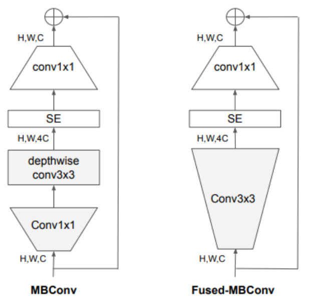


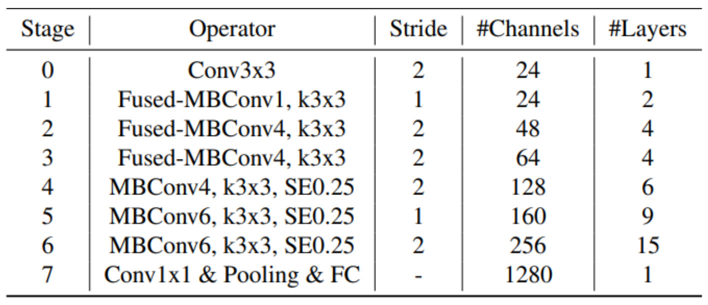
Figure 1: MBConv and Fused-MBConv blocks

   The authors then use a neural architecture search to identify potential models.

2. **Using training-aware neural architecture search (NAS) and scaling.**

   A neural architecture search is an automated technique to find useful neural network architectures that meet specific objectives or constraints (e.g. finding an architecture that is optimised for accuracy) [4].

   The NAS search in this paper aimed to optimise accuracy, parameter efficiency, and training efficiency. This search resulted in EfficientNetV2-S. The architecture for EfficientNetV2-S is shown in [1, Fig 2].



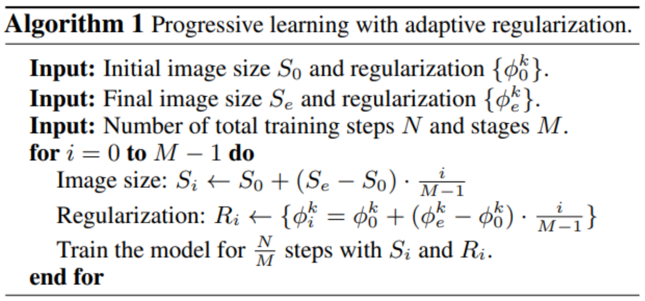
Stage	Operator	Stride	#Channels	#Layers
0	Conv3x3	2	24	1
1	Fused-MBConv1, k3x3	1	24	2
2	Fused-MBConv4, k3x3	2	48	4
3	Fused-MBConv4, k3x3	2	64	4
4	MBConv4, k3x3, SE0.25	2	128	6
5	MBConv6, k3x3, SE0.25	1	160	9
6	MBConv6, k3x3, SE0.25	2	256	15
7	Conv1x1 & Pooling & FC	-	1280	1

Figure 2: EfficientNetV2-S

To scale up EfficientNetV2-S into M/L models, compound scaling as described in [4] is used with some small adaptations since it was found that equally scaling up every stage within the network is suboptimal.

3. **Progressive learning with adaptive regularisation.**

   Prior literature had shown that varying image size during training resulted in better training efficiency with the caveat that a drop in accuracy can occur. This drop in accuracy is hypothesised to be due to uneven regularisation and so it was decided that adaptive regularisation should be used. The algorithm for this training is described below in [1, Fig.3] and a sample image showing adaptive regularisation is shown in [1. Fig. 4].



Algorithm 1 Progressive learning with adaptive regularization.

```
Input: Initial image size $S_0$ and regularization $\{\phi_0^k\}$.
Input: Final image size $S_e$ and regularization $\{\phi_e^k\}$.
Input: Number of total training steps $N$ and stages $M$.
for $i = 0$ to $M - 1$ do
    Image size: $S_i \leftarrow S_0 + (S_e - S_0) \cdot \frac{i}{M-1}$
    Regularization: $R_i \leftarrow \{\phi_i^k = \phi_0^k + (\phi_e^k - \phi_0^k) \cdot \frac{i}{M-1}\}$
    Train the model for $\frac{N}{M}$ steps with $S_i$ and $R_i$.
end for
```

Figure 3:Progressive learning and regularisation algorithm.

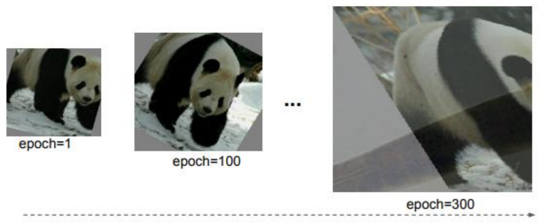


Figure 4: A sampled training image showing Adaptive Regularisation.

Incorporating all these techniques resulted in a family of models (EfficientNetV2).

1.3 Main findings and Results

Benchmarking EfficientNetV2 models generated some interesting results:

Better accuracy compared to other models

EfficientNetV2-S achieves 83.9% Top-1 accuracy on Imagenet. EfficientNetV2 models demonstrate higher Top-1 Accuracy and shorter training times as shown in [1, Figure 5.].

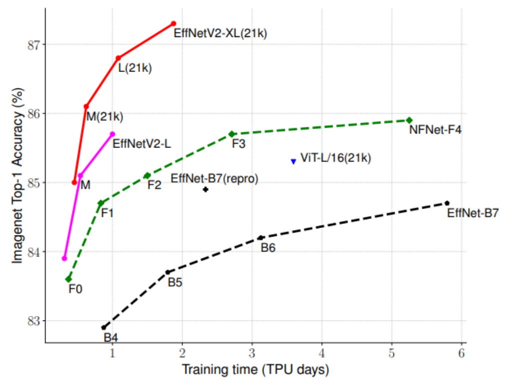


Figure 5: Comparison of accuracy and training times.

A table showing an extensive comparison with other image classifiers can be found in [1].

Improved training times

EfficientNetV2 models are demonstrated to train 5x-11x faster using similar resources compared to other image classification models.

Improved parameter efficiency.

EfficientNetV2-S can be shown to achieve comparable accuracy to models with much more parameters as shown in [1, fig. 6 and Tab. 1].

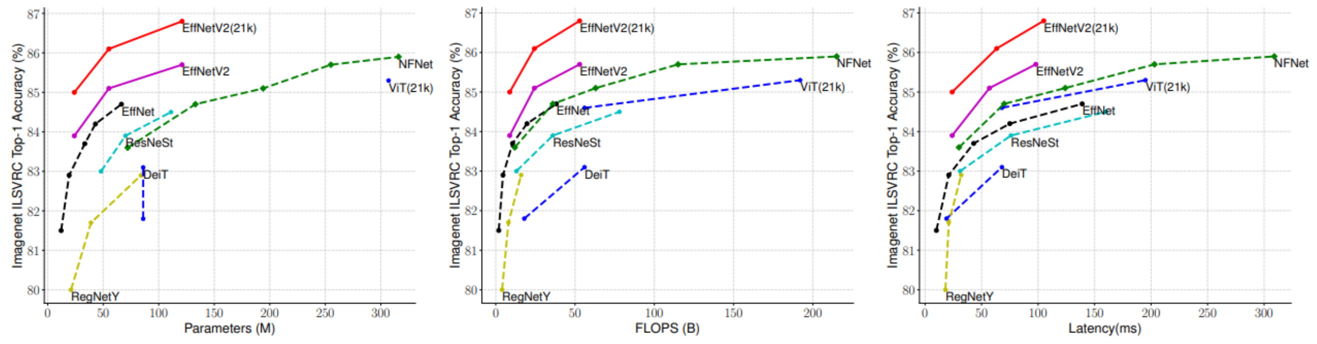
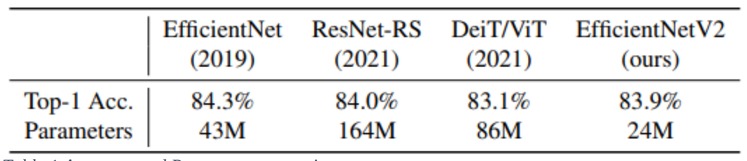


Figure 6 Parameter efficiency comparison



	EfficientNet (2019)	ResNet-RS (2021)	DeiT/ViT (2021)	EfficientNetV2 (ours)
Top-1 Acc.	84.3%	84.0%	83.1%	83.9%
Parameters	43M	164M	86M	24M

Table 1 Accuracy and Parameter comparison.

1.4 Significance

This paper displays multiple strategies to improve model efficiency suggesting that while making a model more complex may yield better performance, it is also warranted to put emphasis on efficiency.

The paper also showcases a family of models that perform exceptionally well when compared to larger models. These models are publicly available and can be trained or adapted for feature learning.

2.0 Replicating the study results

The remaining section of this report explores some of the concepts within the EfficientNetV2 paper. The following experiments are conducted:

Experiment 1.
Compare the training times of EfficientNetV2 with a similar sized model. EfficientNetV2 should train much faster than other models. The purpose of this is to compare efficiency against other similar sized CNNs as the EfficientNetV2 paper highlights these models to be more time efficient and parameter efficient than other similar-sized models.

Experiment 2.
Benchmark the performance of EfficientNetV2 with other models explored in MXN442 namely support vector machines and a random forest classifier. The purpose of this is to understand if we even need to use CNNs for simple image classification tasks. In other words, is it possible to use traditional machine learning methods and avoid deep-learning?

2.1 Methodology

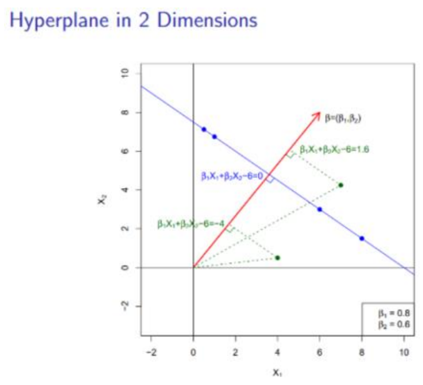
Experiments are conducted using the Extended MNIST Letters (EMNIST-Letters) dataset. EMNIST is a more challenging dataset compared to MNIST [5]. Google Colab Pro is used to perform the experiments as A100 GPUs are available on the platform. Note that EfficientNetV2 models are originally trained on an ImageNet dataset. However, this dataset is extremely large, and training is difficult on a Google Colab environment.

Models are available from torchvision.models as well as scikit-learn. These experiments instantiate models from these sources.

Experiment 1.
EfficientNetV2-S and EfficientNet-B4 are trained and evaluated on the EMNIST dataset. These models have a similar number of parameters (EfficientNet-B4 having 19million parameters and EfficientNetV2-S having 22million). These models are trained for only 10 epochs as the objective is to compare efficiency and not necessarily to converge on an optimal model.

Experiment 2.
SVMs and Decision Trees are trained on EMNIST data. The objective is to identify if non-deep learning methods have utility when it comes to simpler image classification. This also serves to act as an additional benchmark for the EfficientNetV2-S network.

As discussed in MXN442 Week 7, an SVM algorithm finds a hyperplane that separates classes in the feature space. [6, fig 7.] Images in the form of tensors can be collapsed essentially producing large arrays. These arrays can then be passed onto an SVM for classification. It would be interesting to see how an SVM algorithm deals with such high dimensional data.



Similarly, it is possible for a tree-based method such as a Random Forest classifier to separate classes.

2.2 Results and Discussion

Experiment 1.
When comparing EfficientNetV2-S and EfficientNet-B4, training time for 10 epochs was much shorter at 66 minutes for EfficientNetV2-S compared to 80 minutes for EfficientNet-B4. This seems to suggest that EfficientNetV2-S is quicker to train compared to prior EfficientNet models.

Accuracy after 10 epochs was 95% for both models. While EfficientNetV2 models achieved higher accuracy in the paper's test sets, this is not evident in this experiment. A large reason for this could be due to the lack of complexity in the test dataset. The test images from EMNIST are grayscale 28x28 sized images. In contrast, the EfficientNetV2-S model from PyTorch has an input layer of coloured 224x224 images. The dataset chosen could lack the complexity required to discern classification performance between EfficientNetV2-S and EfficientNet-B4.

Model	Training Time (minutes) (10 epochs)	Accuracy
EfficientNetV2-S	66	95%
EfficientNet-B4	80	95%

In future, it may be a good idea to train on a smaller dataset with coloured images and a higher resolution as it may allow for a larger difference in performance.

Experiment 2.
SVMs and the random forest classifier tend to perform well on the EMNIST-Letters dataset. The random forest classifier was able to achieve an accuracy of 88% with the SVM achieving 78%.

Note that it took the random forest classifier 2-minutes to complete training on the training set and predict the test set. This is much more time efficient compared to either of the CNNs explored in #1 or #2. The SVM model took 45 minutes to train and test. This may be because finding a hyperplane on large dimensional data (such as when an image is flattened) for 26 classes is difficult.

Model	Training & Evaluation Time (minutes)	Accuracy
Random Forest Classifier	2	88
Support Vector Machine (SVM)	45	78

The random forest classifier shows promise in handling simple image classification. It is not a bad idea to use a random forest classifier over a CNN such as EfficientNetV2 when images are small (28x28) and grayscale. It would be interesting to see how a random forest classifier performs on larger images with colour as this would add complexity.

For the given dataset, the random forest classifier performed exceptionally well even compared to the CNNs. While the level of accuracy wasn't on par with the CNNs, the fact that a random forest classifier can require much fewer resources and still perform well is interesting.

3.0 Reproducibility

Code is available and experiments can be replicated here: https://github.com/kurtperez123/MXN442-Assignment

Please follow the README instructions as it contains file structures and overviews of what each file does. Some models are already saved and so it is possible to load these models and simply evaluate them.

I have used Google Colab Pro and utilised an A100 GPU to train and evaluate my models. Performing the same experiments on a CPU may take a long time. Additionally, RAM and GPU memory can be a constraint. It may be necessary to subset the dataset or train/evaluate in smaller batches if you are experiencing any system issues.

4.0 Reference List

[1] Tan, Mingxing, and Quoc V. Le. "EfficientNetV2: Smaller Models and Faster Training." 2021. arXiv preprint arXiv:2104.00298.
Available at :https://arxiv.org/abs/2104.00298.

[2] M. Tan and Q. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," *proceedings.mlr.press*, May 24, 2019. https://proceedings.mlr.press/v97/tan19a.html

[3] T. Brown *et al.*, "Language Models are Few-Shot Learners," Jul. 2020. Available: https://arxiv.org/pdf/2005.14165

[4] B. Zoph and Q. V. Le, "Neural Architecture Search with Reinforcement Learning," *arXiv:1611.01578 [cs]*, Feb. 2017, Available: https://arxiv.org/abs/1611.01578

[5]
G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, "EMNIST: an extension of MNIST to handwritten letters," *arXiv:1702.05373 [cs]*, Mar. 2017, Available: https://arxiv.org/abs/1702.05373

[6]
M. Abolghasemi (2024). MXN442 Week 7 SVM Lecture [PowerPoint slides]. Accesible on Canvas.