State of The Art

EfficientNetV2 with Image Classification Benchmarks

Wong Zhao Wu

Abstract

Improving the performance of a vision model is product of the following three different sections (Cited from Revisiting ResNets Paper).

1. Model Architecture
    - EfficientNetV2 (NAS, Squeeze & Excitation, Fused-MBConv, MBConv, Skip Connections)
2. Training Procedure
    - Progressive Learning (Progressive Resizing, RandAugment, Dropout, Stochastic Depth, Weight Decay)
3. Scaling Method *(Not Enough Computation Power to Explore)*

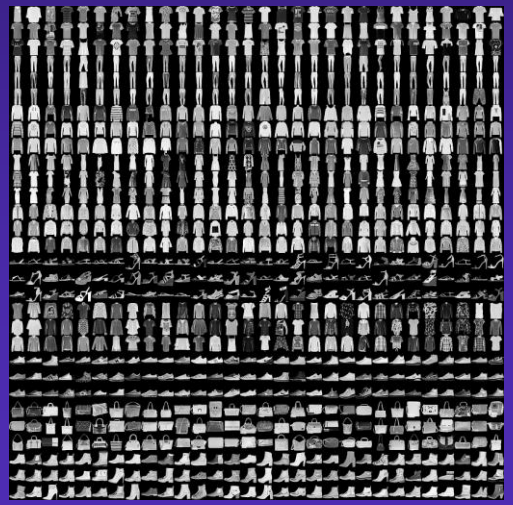
Modelling Objective

In CA1, the goal is to implement state of the art **model architecture** and **training procedure** without compromising the **generalization capability** and **performance** of the model.

Benchmark Datasets

Fashion-MNIST

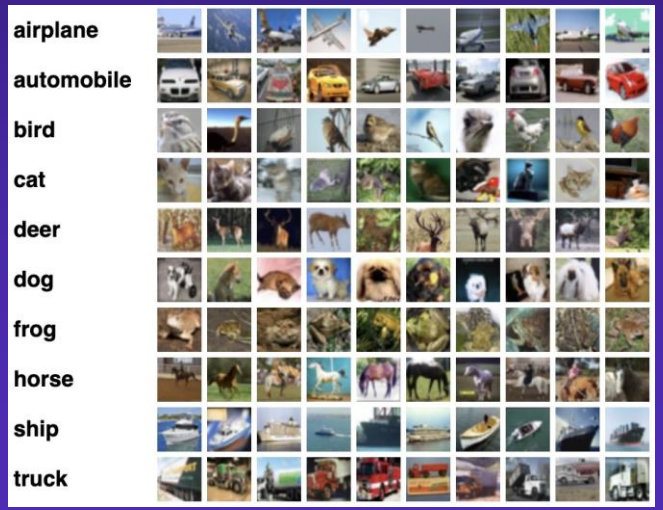
28×28 grayscale images of 70,000 fashion products from 10 categories, with 7,000 images per category. The training set has 60,000 images and the test set has 10,000 images.



0 T-shirt/top
1 Trouser
2 Pullover
3 Dress
4 Coat
5 Sandal
6 Shirt
7 Sneaker
8 Bag
9 Ankle boot

CIFAR-10

Canadian Institute for Advanced Research consist of 60000 32x32 color images from 10 categories. There are 6000 images per class with 5000 training and 1000 testing images per class.



Modelling Assumption

Since both tasks involved Image Classification with exactly 10 labels and,
CIFAR-10 is a harder task than Fashion-MNIST, thus, Model that performs well in CIFAR-10 will perform as good in Fashion-MNIST.

Based on this assumption, earlier Modelling Experiments are done with CIFAR-10 dataset, to save computational resource, before readapting and finetune the experiment for Fashion-MNIST accordingly.

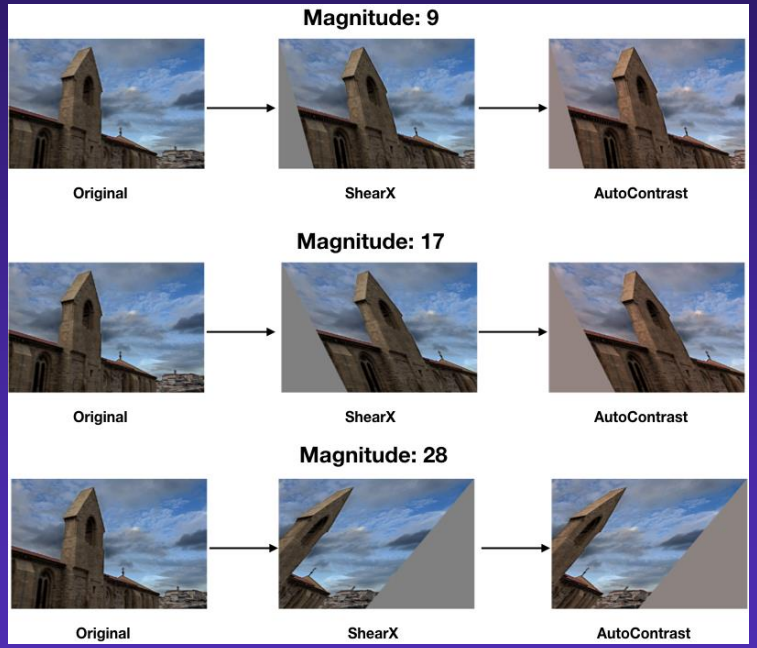
Feature Engineering and Image Augmentation: Progressive Resizing (Howard, 2018), FastAI

- Gradually using larger and larger images as you train.
- Motivations
  1. Faster Training on Smaller Image Resolution.
  2. Use large image to fine tune the model to further boost accuracy.
  3. Kinds of features that are learned by convolutional neural networks are not in any way specific to the size of the image
  4. Another form of data augmentation to increase generalization ability of model

Feature Engineering and Image Augmentation: RandAugment (Cubuk et al., 2020), Google Brain

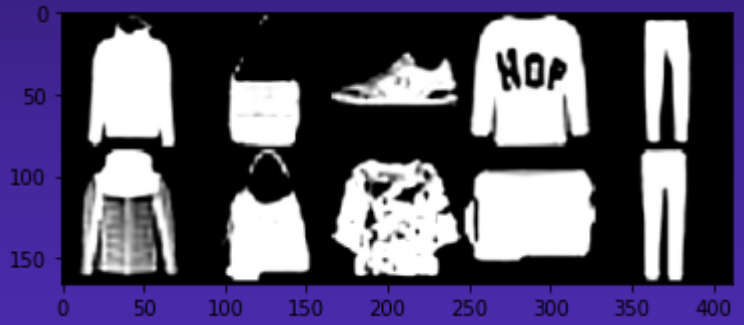
- Sets of basic data augmentation techniques/policies with predefined (e.g. Identity, ShearX, ShearY, TranslateX, TranslateY, Rotate, Brightness, Color, Contrast, Sharpness, Posterize, Solarize, AutoContrast and Equalize)

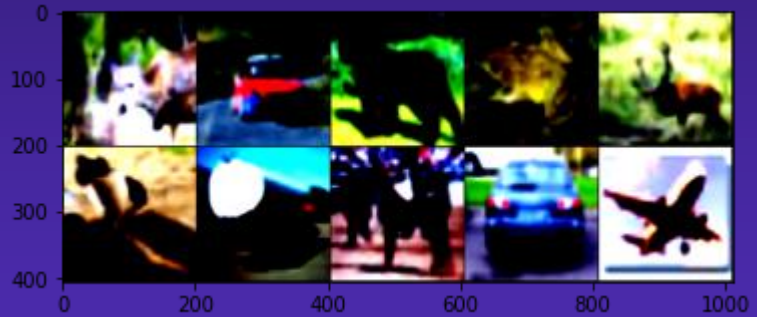
- Motivations
  1. Single Magnitude parameter to adjust the magnitude of augmentation for all policies



Feature Engineering and Image Augmentation:
Image Rescaling

- Basic rescaling of means and standard deviation of image to standardize the pixel values of input image





Early Modelling Experiments

Early Modelling Experiments:
Resnet-50

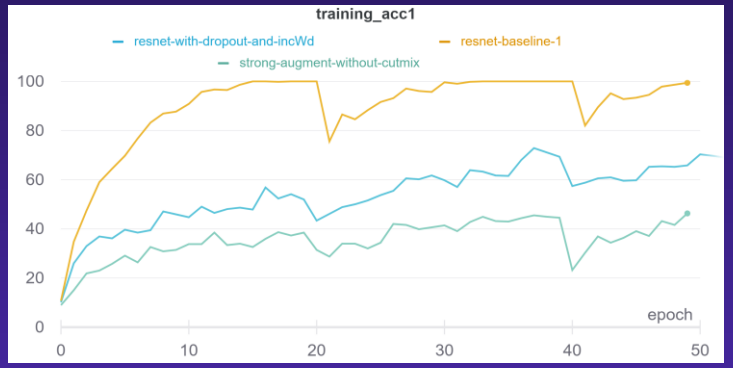
1. **Resnet-50 as baseline**
   Encounters heavy overfitting

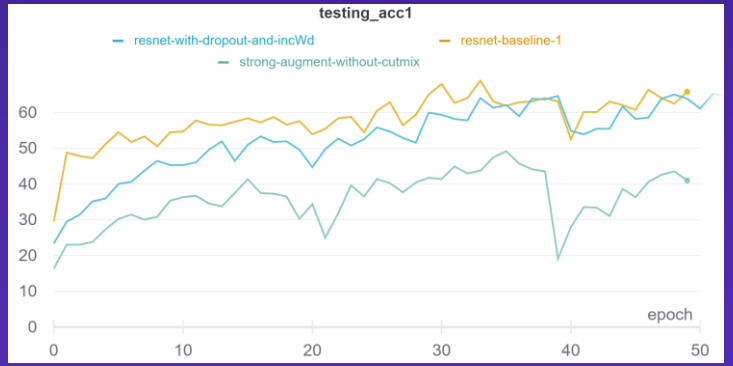
2. **Resnet-50 with strong Augmentation**
   (Dropout, Weight Decay, Strong Affine Transformation),
   Encounters heavy underfitting

3. **Resnet-50 with Moderate Augmentation**
   Promising Performance without suffering from overfitting, but Validation Accuracy capped at 80.

Possible Explanation:

1. Model not complex enough (i.e. Underfitting)
2. Not enough training data





Early Modelling Experiments: Wide-Resnet-101

1. **Wide-Resnet-101 with CosineAnnealingLRScheduler (50 epoch)**
2. **Wide-Resnet-101 with CosineAnnealingLRScheduler (100 epoch)**

After training both models for long epochs, the model still stuck at 80 validation accuracy, despite the change of scheduler hyperparameters.

Thus, we can rule out the option that Resnet is not complex enough for the task as it did managed to overfit for ~100% accuracy without regularization.

Hence, better model architecture and training procedure is required.

EfficientNet V2

Background History: EfficientNetV1

1. EfficientNetV1 by Tan et al., 2020 is introduced by Google Brain to study on the most efficient way to scale network through Compound Scaling.

2. Through performing Neural Architecture Search the team proposed a small baseline EffcientNet-B0 model to find the optimal scaling coefficients by optimizing FLOPs and performance.

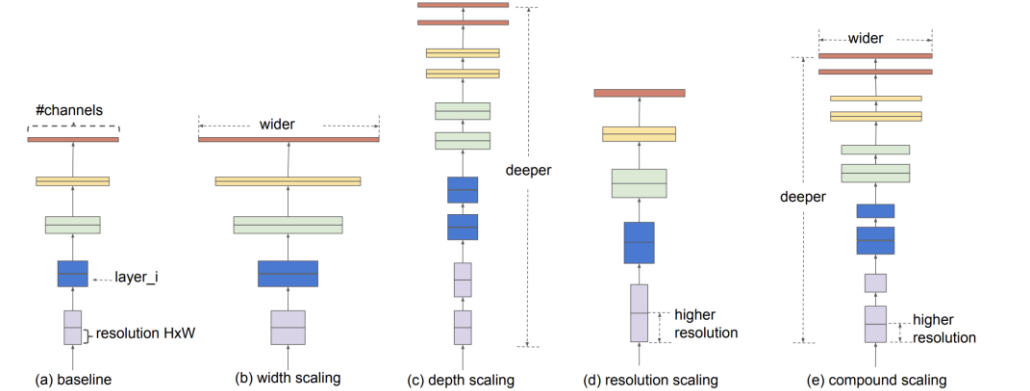


Figure 2. **Model Scaling.** (a) is a baseline network example; (b)-(d) are conventional scaling that only increases one dimension of network width, depth, or resolution. (e) is our proposed compound scaling method that uniformly scales all three dimensions with a fixed ratio.

Background History:
EfficientNetV2

1. EfficientNetV2 by Tan et al., 2021 is then introduced to include training speed as part of the optimizing objective and proposed Progressive Learning which adaptively adjust regularization along with image size speed up training speed and outperforms previous models on ImageNet and CIFAR/Cars/Flowers dataset on performance and parameter efficiency.

2. Both of the EfficientNetV2 models make used of novel architecture techniques like Skip Connections, Squeeze and Excitation, Inverted Residual Block/Mb-Conv and has shown to be powerful and parameter efficient than other SOTA models.

Table 4. EfficientNetV2-S architecture – MBConv and Fused-MBConv blocks are described in Figure 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

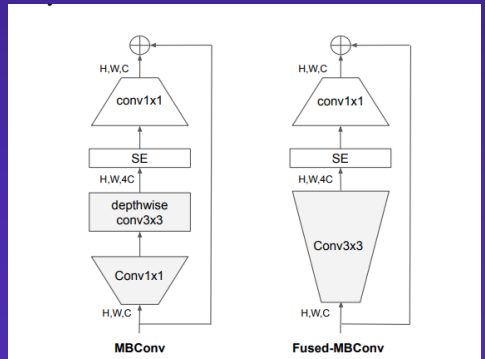


Figure 2. Structure of MBConv and Fused-MBConv.

Training Summary

Progressive Learning

1. For CIFAR-10, EfficientNetV2-M is used while for Fashion-MNIST, EfficientV2-S model is used.

2. Since the paper only shows the training hyperparameters used for ImageNet dataset, which is much harder than CIFAR10 or FashionMNIST, I have to make some adjustment to suit the training procedure into different dataset.

3. The following tables shows the exact hyperparameters used in both dataset after running a few iterations of experiments on both dataset. However, since most of the codes are overlapping and only the hyperparameters are fine tuned, and sometimes I have to eyeball the model training performance to determine when to switch over to next training phase, and hence, I did not left over the history of tuning to showcase.

Epochs	Image Size	RandAugment Magnitude	Dropout Rate	Stochastic Depth(Drop Path Rate)	Weight Decay
1-20	28	5	0.2	0	0
21-50	50	10	0.3	0.1	3e-4
51-100	80	20	0.4	0.2	3e-3

Fashion-MNIST

Epochs	Image Size	RandAugment Magnitude	Dropout Rate	Stochastic Depth(Drop Path Rate)	Weight Decay
1-20	128	10	0.2	0	1e-4
21-40	150	15	0.3	0.1	3e-4
41-60	200	20	0.4	0.2	3e-3

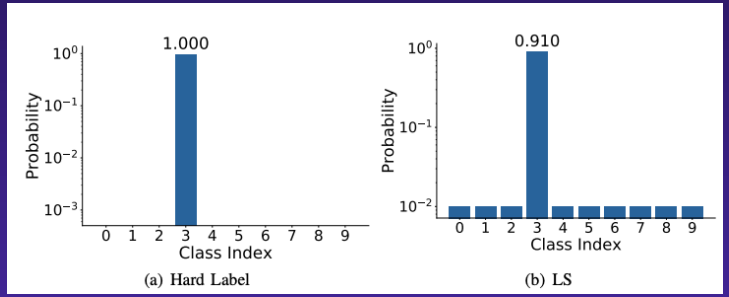
CIFAR-10

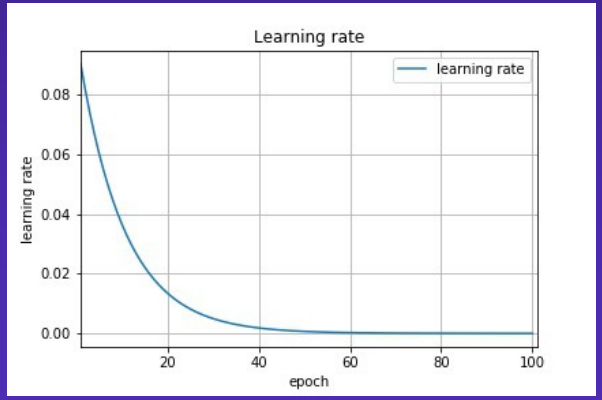
Training Summary

Static Hyperparameters

Other training hyperparameters that I make constant across all experiments are as followed:

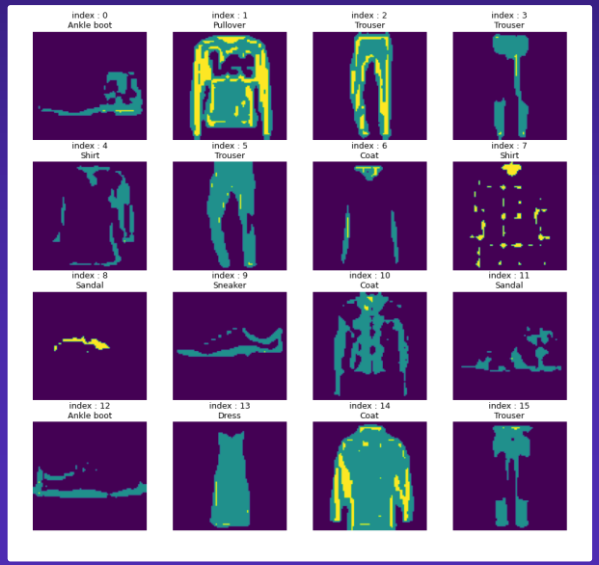
- Loss:
  Categorical Cross Entropy with Label Smoothing of 0.1
- Optimizer:
  Adam with initial learning rate of 1e-3
- Learning Rate Scheduler:
  Exponential Decay Learning Rate with Gamma 0.99

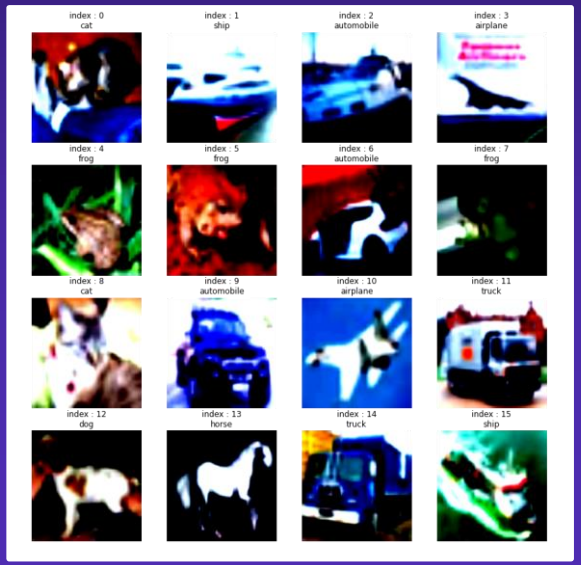




Model Evaluation

1. After finalizing the model performance through the validation dataset, the hold-out test set is used to evaluate the model performance.
    - Fashion-MNIST : 93.93%
    - CIFAR-10 : 92.55%





Thanks!

Personal Learning Reflection

Although I did not manage to achieve the same performance as the leaderboard with proper training-validation-test split convention, I managed to achieve accuracy of 93.93% on 10,000 test set and gain practical experience in training and building vision-based CNN models in Pytorch.