Imperial College London

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

EfficientNet with RandAuguemnt for Diabetic Retinopathy Detection

Author:
Yurun SONG

Supervisor:
Dr. Amir Alansary

Submitted in partial fulfillment of the requirements for the MSc degree in MSc Advanced Computing of Imperial College London

April 2020

Abstract

Image classification is an important research topic in medical area for decades. Using the Neural Network to classify medical images becomes a convenient and fast way. Many models are proposed for achieving high accuracy with low computational cost. EfficientNets are a range of brand new models that have outstanding performance on transfer learning. Moreover, EfficenNets make full use of limited resources and provide more choices against different datasets. In this project, we will implement EfficentNets to categorize retinal images into different levels associated with image augment methods. Meanwhile, exploration about the dataset and evaluation on the models are performed in the project.

Acknowledgments

I would like to give my sincere gratitude to my supervisor Dr. Amir Alansary, for offering his guidance and support throughout this project, and for always responding to my ideas with enthusiasm and patience.

Contents

Chapter 1

Introduction

Neural Network for image classification has been a significant topic researched for many years. Many works of the published research have been developed for improving the performance of the model with less resource, e.g. Residual Neural Network (ResNet) (He et al., 2016), Mobile Network (MobileNets) (Howard et al., 2017), Neural Architecture Search Network (NasNet) (Ramachandran et al., 2017), Mobile Neural Architecture Search Network (MNasNet) (Zoph et al., 2018). However, these models have a potential defect that there is no systematic method to scale up the model for different datasets. Although model like ResNet can achieve depth scaling by increasing number of layers as in ResNet-101, or width scaling by increasing number of channels as in wide ResNet. Both approaches were manually explored without considering the relationship of depth and width. Additionally, some other models like NasNet, MNasNet can automatically exhaustive search the best model, but it is costly and not scalable.
Recently, a new family of architectures known as EfficientNet (Tan and Le, 2019) is designed for solving this problem by balancing depth, width and resolution scaling within the limited resources. It achieves better performance through uniformly scaling from a baseline to a large model. The EfficientNet family models range from EfficientNet B0 to B8 based on the network's size, and widely used in different kinds of datasets, like Cifar10, Cifar100 (*Cifar*, 2020), and ImageNet (*ImageNet*, 2020).

Imbalance and noise are two common artifacts universally occur in many real-world image datasets. One of the approaches to address the class imbalance is data augmentation, which can artificially expand the size of dataset by creating modified versions of images and potentially enhance the robustness of the model to tackle image noises. In addition, it can perform as a regularization for complex models. More specifically, the ratio of each classes could heavily affects the training of deep models by introducing a bias towards the dominant class. Without manually design policies that can capture prior knowledge in each domain, RandAugment (RA) (Cubuk, Zoph, Shlens and Le, 2019) can automatically generate variaty of images with a little computational cost, and adjust the strength of regularization based on the size of model or dataset. RandAugment are used uniformly across different tasks and datasets, as well as can be combined with EfficientNet models to boost their performance.

In this thesis, we evaluate EfficientNet models on healthcare data. Namely, we choose Diabetic Retinopathy (DR) scanned images as they are very common kinds of diabetic eye diseases. High blood sugar level leads to damage of the small vessels in the retina. DR is the dominant reasons for blindness occurs in many developed countries. According to the NHS 2019, $1,280$ new cases of blindness caused by DR are reported each year in England alone, while a further $4,200$ people in the country are thought to be at risk of retinopathy-related vision loss (Diabetes.co.uk, 2019)
Unlike many other diseases, DR may occur without any symptoms and pain in the early stage, which makes it difficult to prevent in advance. The influence of DR on the vision could come up when the disease starts to deteriorate. Hence, early eye screening and diagnosis are essential measurements to prevent this to happen and reduce the loss of vision.
However, long waiting time and difficulty scheduling appointments make the person less likely to screen. Therefore, deep learning-based approach for detecting DR with high sensitivity and specificity is worthy of developing to resolves the situation.

1.1 Objectives and Challenges

The final objective of this project is to efficiently implement an EfficientNet model to classify scanned retinal images into 5-levels of severity accurately. The first step is to construct an EfficientNet baseline model and restore the experiment in paper (Tan and Le, 2019) on small Cifar datasets . Then, achieving the high performance on the large Diabetic Retinopathy (DR) dataset by making use of transfer learning of the pretrained Efficientnet B5. In order to complete this task efficiently, the AutoML (Alansary, 2020) for DR classification and its previous best performance are used in this project.
There are several challenges in this project. First, the proper training hyperparameters for EfficientNet is much harder to find even though it performs well in different datasets. Some common training techniques are not suitable for the EfficientNet. Besides, the batch size and epochs of original experiment are hard to implement due to the limitation of resources. Moreover, imbalanced classes and noises in the DR dataset are the challenges for the training. According to descriptions of data (*Diabetic Retinopathy Detection*, 2015), images may contain artifacts, be out of focus, underexposed or overexposed. Like any real-world data set, noise in both the images and labels are exists. Developing a robust algorithm that can work in the presence of noise and variation is a challenge for this project.

1.2 Contribution

1. Running extensive experiments for reproducing the reported EfficientNet performance on Cifar data.

2. Applying EfficientNet models on DR datasets and achieving 87.08% in public

test and 85.81% in private test set.

3. Combining RandAugment and BalancedSampler with EfficientNet to solve the imbalance and noise problems.

4. Performing the best model on other DR dataset.

Chapter 2

Background

The chapter explains in detail image classification models and image augmentation. In the first section, we cover some of the recent models that have been used as baselines for evaluation. Whereas, several image augmentation and other improvement techniques are summarised in the second part.

2.1 Models

2.1.1 Residual Neural Network (ResNet)

Blindly increasing the number of layers does not always produce an ideal performance and gradient vanishing problem occurs along with the increment of layers. Therefore, He et al. (2016) have proposed a special architecture based on residual blocks 2.1. These blocks allow to skip certain layers through identity shortcut connection. The shortcut connection provides more possible paths where data can forward and backward through the network. Hence, the network can determine to skip over or pass through the layer. This reduces the relevant backpropagation layers flexibly, and effectively tackles the vanishing gradient problem. Residual block promises the deeper model would not produce a training error higher than its shallower NN.
Additionally, another block architecture is proposed to save the computational time, which contains two 1x1 convolutions with a 3x3 convolution in the middle for a single block, as shown in Fig. 2.2. The short connection is implemented in the EfficientNet to avoid gradient vanishing.

2.1.2 MobileNetV1

MobileNet (Howard et al., 2017) is a convenient and effective architecture designed for the mobile and embedded devices. It results in a lightweight and highly accurate model that also can performs using limited resources. The core concept of the MobileNet is called Depthwise Separable Convolution, which contains depthwise convolution and pointwise convolution.
Depthwise convolution applies the same sized filter with only depth 1. It generates

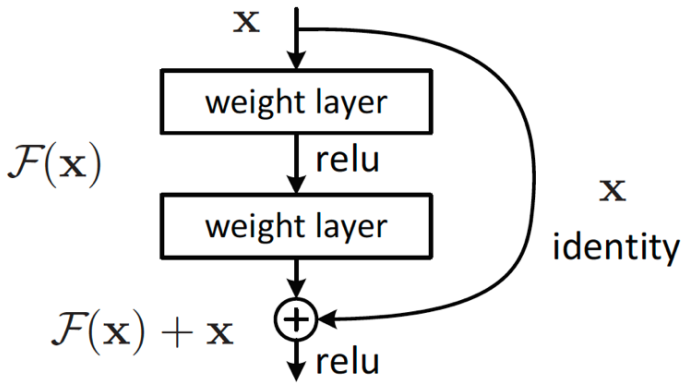


Figure 2.1: A simple diagram of a residual block showing the skip connection (He et al., 2016).

M outputs if the depth of input image is M. Pointwise convolution applies 1x1 filter with depth M, and it generates N outputs, where N is the product of input's width and height. These two simple procedures sacrifice a small amount of accuracy but significantly reduce the computational time.
It also introduces the concept of width multiplier and resolution multiplier, which are used to uniformly thin the network layer and reduce the computational cost of the Neural Network. The trade-off between the width and resolution multiplier is an important part that EfficientNet explored.

2.1.3 MobileNetV2

Extended the outcomes from the MobileNetV1, the second version of MobileNet (MoblieNetV2) (Sandler et al., 2018) proposes an architecture called inverted residual block. It combines a residual bottleneck architecture and depthwise separable convolution. The inverted residual block applies a pointwise (1x1) convolution to convert to high dimensional feature map space with ReLU6, then an expansion ratio is used to generate more channels and widen the layer, see Fig. 2.2. Depthwise (3x3) convolution with a particular stride, can be applied in the next layer and followed by ReLU6 as well. Eventually, a pointwise layer projects back to a low dimension with only linear activation. They present the idea that ReLU is not good at preserve information in the high dimension. Furthermore, the residual connections should be made between two low-dimensional feature maps.
MobileNet2 provides a good concept that the middle layer inside of bottleneck strucutre should expand instead of squeezing, which is exactly what Mnasnet and EfficientNet inherited.

2.1.4 ResNeXt

ResNeXt (Xie et al., 2017), an evolution architecture of the ResNet, introduces a new dimension called cardinality (Except depth and width). It equally splits the channels into different groups and each group is transformed by its own weight. Eventually, the transformations in each group are aggregated by summation. As the Fig. 2.3 illustrates, ResNet and ResNeXt shares the similar residual block architecture, both

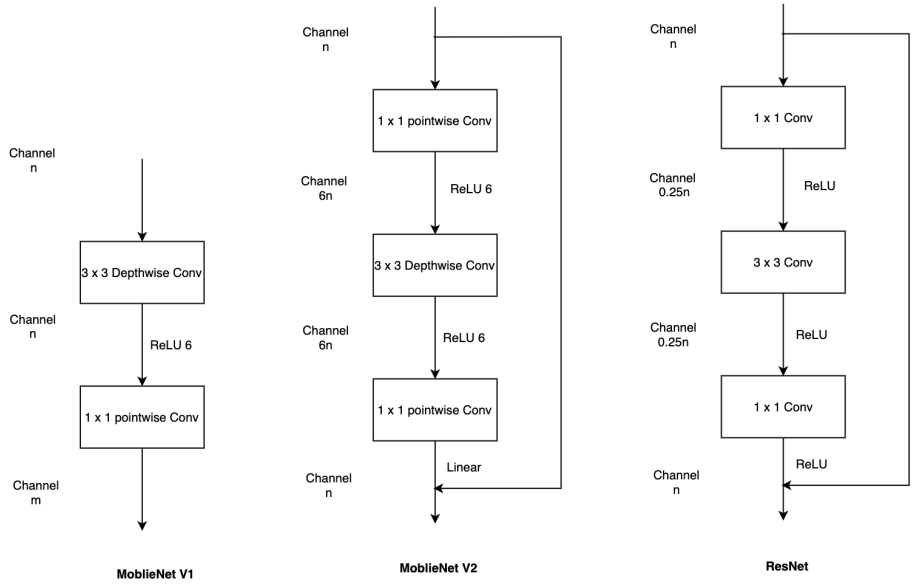


Figure 2.2: Block structure of MobileNet V1, V2 and ResNet.

have 1x1, 3x3 and 1x1 kernel sequentially. The difference is that ResNeXt provides more paths and each path performs a non-linear transformation, which indirectly aggregated to a complex transformation. The new dimension cardinality controls more detail of channel weights and implements more complex transformations. The paper (Xie et al., 2017) shows that the proper combination of cardinality (groups) and width of bottleneck contributes a better performances with the similar cost in number of parameters and floating-point operations (FLOPS).

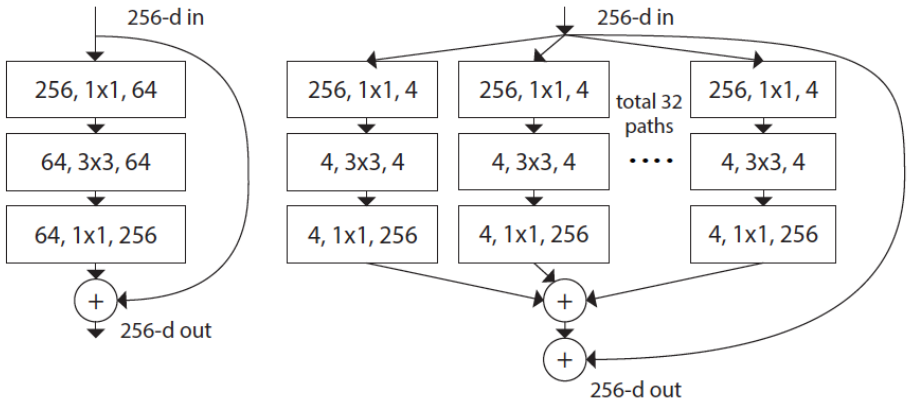


Figure 2.3: ResNet **Left** and ResNeXt **Right** (Xie et al., 2017).

2.1.5 Squeeze and Excitation layer

An issue is proposed in the paper (Hu et al., 2018) that the weight can not assign to the preferred channels. Each channel shares the same weight when generating

the output feature maps. Therefore, based on the Residual block, Squeeze and Excitation (SE) Network raises an adaptive and optimal way to adjust the weighting of each feature map by adding parameters to each channel in the convolutional block. It consists of squeeze, Excitation and scales three stages, see Fig. 2.4.In the squeeze stage, the global average pooling is applied to squeeze the spatial information into channel-wise features. A fully connected layer with ReLU and a fully connected layer with Sigmoid are used successively in the excitation stage. This nonlinear excitation stage fully captures the association of each channel and produces the corresponding weights. Finally, scaling back the original dimensional with additional parameters introduced.
SE layer improves the inter-dependencies between channels without adding too much computational cost. It is widely accepted by lots of deep Neural Network models, especially MNasNet and EfficientNet.

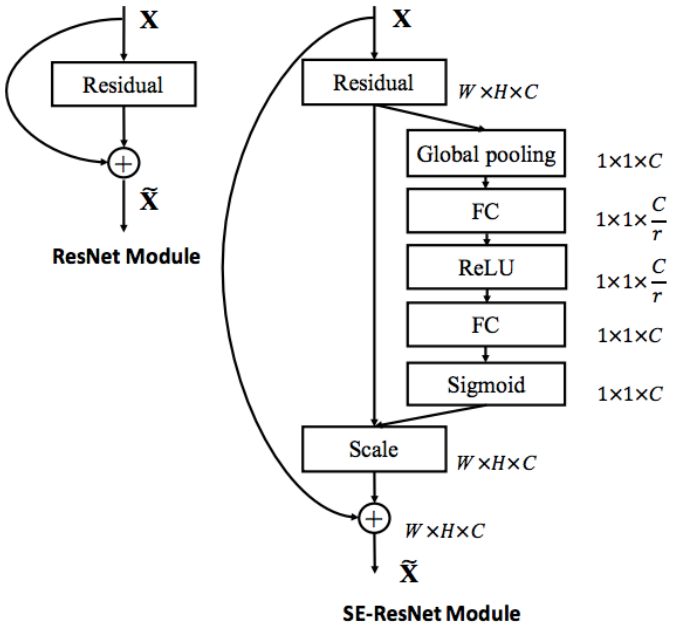


Figure 2.4: Squeeze and Excitation layer on ResNet (Hu et al., 2018).

2.1.6 Neural Architecture Search Network (NasNet)

Neural Architecture Search Network (NasNet) is a systematic searching model. The author (Zoph et al., 2018) searches convolutional cells by using reinforcement learning on the small dataset like Cifar10 and transfer the cells onto a large dataset like ImageNet.
Each dataset has its predefined NasNet framework but they share the same cells searched by reinforcement learning. The convolutional cell, see Fig. 2.5 has two categories: normal cell and reduction cell. The normal cell aims to keep the same dimension of input and output feature maps while the reduction cell generates an output feature map, which is a half size of the input.

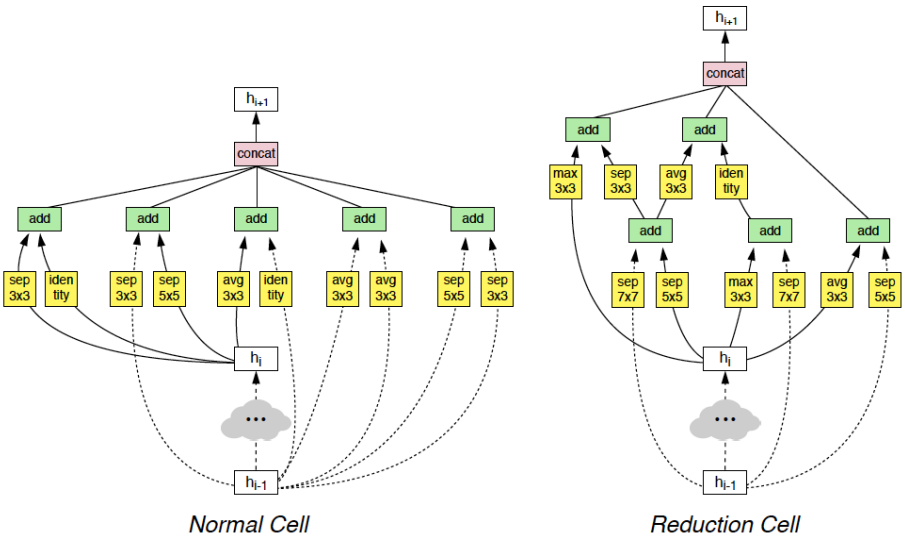


Figure 2.5: NasNet-A (4 blocks) Normal and reduction cells (2 inputs and 1 output) (Zoph et al., 2018).

At the same time, each cell is constructed by a predefined number of blocks. For example, NasNet A has 4 blocks each cell. A block is composed of two hidden (input) layers, two operations like 3x3 depthwise Conv, 5x5 max pooling and a combined method such as addition and concatenate, as the Fig. 2.6 shown. Softmax is applied for each selection of hidden layers, operation and combined method in search space. The resulting hidden state from a block is retained in the set of potential hidden states to be selected in the following blocks.

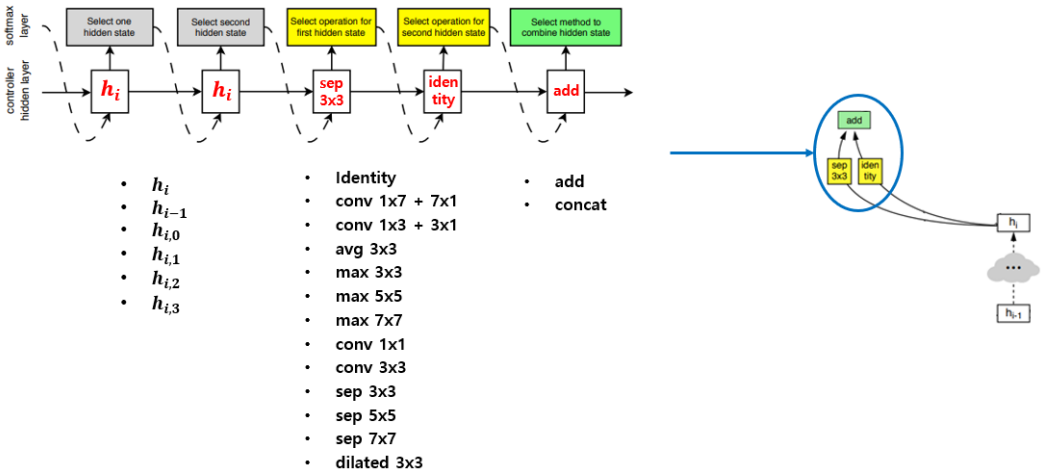


Figure 2.6: Block structure of NasNet (Zoph et al., 2018).

Only normal and reduce cell are searched by the RNN controller, see Figure 2.7. These optimal cells are used in the pre-defined network architectures and fed to

child networks on a small dataset with probability and update the controller using probability and return validation accuracy.

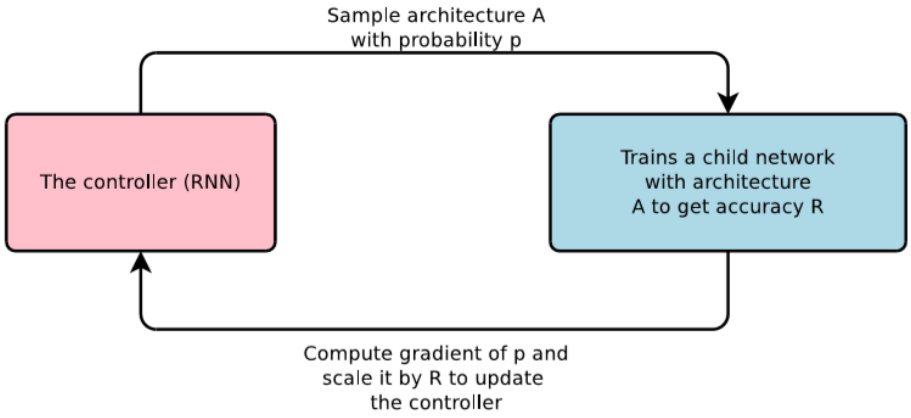


Figure 2.7: An Overview of NasNet and RNN controller (Zoph et al., 2018).

The paper (Zoph et al., 2018) also proposes a method called ScheduledDropPath, which performs well in regularizing NasNet. Drop path stochastically drops out each path in the cell with probability. Moreover, drop path with linearly increasing the probability throughout the whole training significantly improves the final performance, which is the reason why EfficientNet linearly increases drop connect rate every two versions.
NasNet is the basic version of MNasNet, which produce the baseline model of EfficientNet and indirectly contributes to the growth of EfficientNet.

2.1.7 Platform-Aware Neural Architecture Search for Mobile (MNasNet)

NasNet models aim for searching a high accuracy model rather than an efficient one. Therefore, MNasNet (Tan et al., 2019) was developed, which requires not only high accuracy but also low latency in the mobile platform.
In the controller system, see Fig. 2.10, they add a real word mobile phone to measure the inference latency instead of FLOPS and multi-objective rewards containing both latency and accuracy is fed back to the controller.

Considering the same cells are repeated many times in the whole NasNet architecture, MNasNet proposed factorized hierarchical search space to permit the cell diversity, which is critical for achieving high accuracy and lower latency. Factorized hierarchical search space consists of a group of predefined blocks, based on the input resolution and filter size, see Fig. 2.9. Each block includes N number of layers, where N is also a part of search space. A layer can be composed of more detail search spaces like Operations, Kernel sizes, SE ratios, etc. Hence, layers from the different blocks are not the same. This method factorizes a CNN model into unique

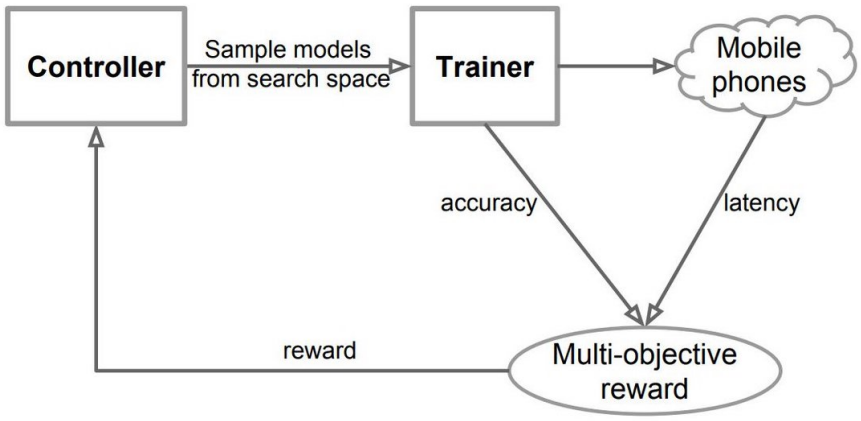


Figure 2.8: An Overview of MNasNet and its RNN controller (Tan et al., 2019).

blocks and then searches for the operations and connections per block separately. The layer diversity achieves a good trade-off between accuracy and latency.

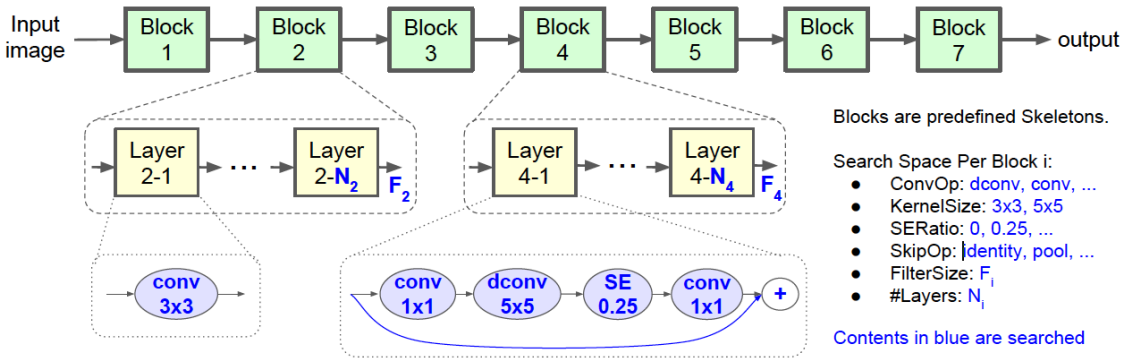


Figure 2.9: Factorized hierarchical search space (Tan et al., 2019).

The baseline model of EfficientNet (EfficientNet-B0) is generated by MNasNet mechanism as well. Beyond that, the paper (Tan et al., 2019) provides Mobile Inverted Bottlenect Conv block (MBConv) as the core used in EfficientNet architectures, see Fig 2.10. MBConv is a block evolved from inverted residual block in MobileNetV2, involving some new techniques like SE layer. MBConv3 and MBConv6 are the searched block from MNasNet with different expansion ratio, and work as the foundation of EfficientNet models.

2.1.8 Swish

Many activation functions were developed to be used in deep neural networks, but they have not shown a consistent gain with the dimension incresing. For example, even ReLU is not good at preserve information in the high dimension (Sandler et al., 2018). In the paper (Ramachandran et al., 2017), they discovered automatic search

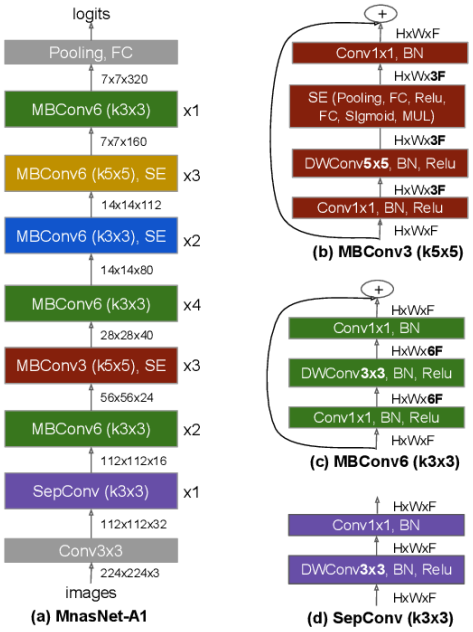


Figure 2.10: NasNet-A and MBConv block (Tan et al., 2019).

techniques to discover new activation functions. Many unary and binary functions as search space, they use exhaustive search and RNN controller for small and large dataset. The core unit in the RNN controller passes two inputs into two unary functions independently and use a binary function to combine and return an output. They get the best performance for Swish, which works better than ReLU both on the small dataset and deeper models.

$$f(x) = x \times \sigma(\beta x) \tag{2.1}$$

Swish function is defined as above, where σ is the sigmoid function and β is either a constant or a trainable parameter.
if $\beta = 1$, swish is equivalent to the Sigmoid-weighted Linear Unit (SiL).
If $\beta = 0$, it becomes the scaled linear function $f(x) = \frac{x}{2}$.
If $\beta \to \infty$, the sigmoid component approaches a 0-1 function and swish turns to the ReLU function.
It shows that Swish can be loosely viewed as a smooth function which non-linearly interpolates between the linear function and the ReLU function. The degree of interpolation can be controlled by the model if β is set as a trainable parameter.
Swish is used in the EfficientNet models and shows great stability than ReLU6 especially for large EfficientNet model B6 and B7.

2.1.9 EfficientNet

The relationship between model size, accuracy and efficiency has been a research topic for a long time. Scaling up the depth and the width of model are most common methods to achieve an optimal accuracy and efficiency. Image resolution is

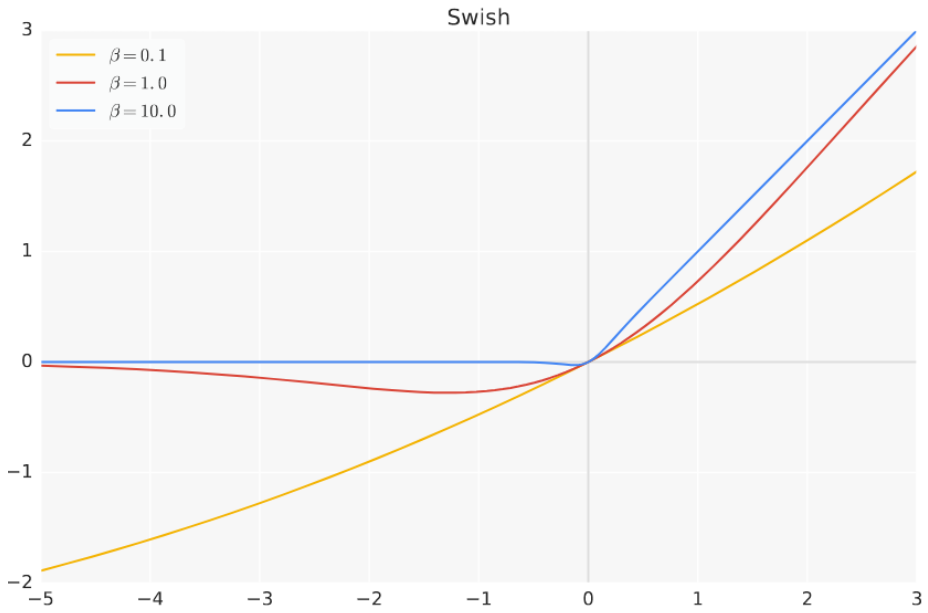


Figure 2.11: Swish activation function (Ramachandran et al., 2017).

gradually accepted as an another important factor that affect the performance. EfficientNet (Tan and Le, 2019) aims to achieve a better performance by systematically scaling the models in depth, width and resolution, see Figure 2.12.
Scaling up width tend to obtain more fine-grained features and easier to train. However, the accuracy is much faster to reach saturation with a wider network. Depth scaling is restricted by the vanishing gradient problem and diminishing accuracy with a deep neural network. In high resolution image, it could potentially capture features in detail, but it also could lead to a decline in the accuracy at the same time. Therefore, a principled method is proposed to uniformly scale all dimensions with a compound coefficients.

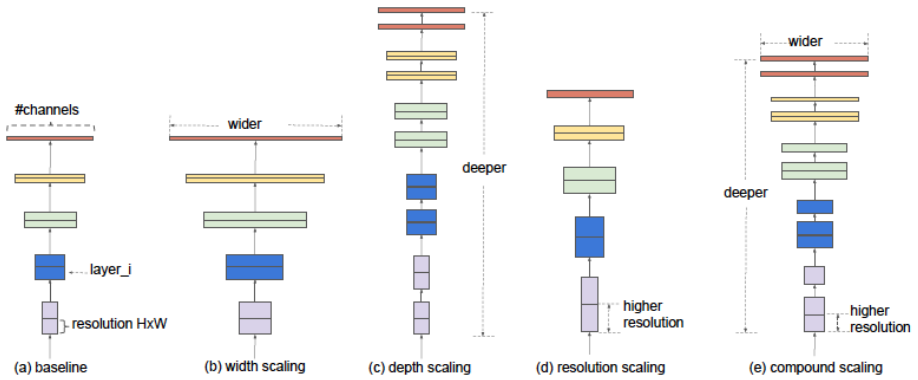


Figure 2.12: Model scaling (Tan and Le, 2019).

A compound coefficient controls how many resources available for the model. It is directly associated to a regular efficiency measurement Floating Point Operations

(FLOPS). FLOPS is almost proportional to the product of depth, the square of width and the square of resolution. By using the fixed FLOPS, assign the resources to the depth, width and resolution individually using the coefficients.
A baseline ConvNet EfficientNet-B0 is developed by performing MnasNet while EfficientNet-B1 to B7 are obtained by scaling up the baseline ConvNet. The main building block in the baseline network is named as mobile inverted bottleneck MBCov. It is mainly composed by Inverted Residual block with squeeze-and-excitation layer.
The performance of the EfficientNet Architecture is measured by accuracy and efficiency. In the high-accuracy area, it surpass the state of art on ImageNet and other five common transfer learning databases. For instances, EfficientNet-B7 reaches state-of-the-art 84.4% top-1 and 97.1% top-5 accuracy on ImageNet. For the efficiency, with an order of magnitude fewer parameters and FLOPS, EfficientNEt-B7 is 8.4x smaller and 6.1x faster on CPU inference than the previous Gpipe (Tan and Le, 2019).

2.2 Image Augment

Image augment is effective way to improve the image accuracy and keeps the robustness of models. At the same time, augment can randomly increases the amount and diversity of the data. This section introduces the most popular image Augment methods and some other techniques.

2.2.1 AutoAugment

Instead of manually searching for the best augmentation method, Cubuk et al. (Cubuk, Zoph, Mane, Vasudevan and Le, 2019) proposed an automatic way to get the optimal augmentation policy for the dataset of interest. Unlike other generators, AutoAugument does not directly generate augmented images. Instead, it generates symbolic transformation operations.
Using RNN Controller like NasNet, AutoAugument also needs a child network to train on the small dataset and transfer the optimal policy to a large dataset. In the searching space, a policy is composed of five sub-policies, which includes two image transformation operations to be applied in sequence, see Figure 2.13. Each operation is associated with two numerical values, magnitude and probability. Magnitude determines how strong the transform applies, and probability decides the occurrence of the operation. The images in the mini-batch randomly select one sub policy for the batch and generate the transformed images for training. In total, there are 16 operations and 10 uniformed magnitudes, 11 probabilities assigned to each operation, see Figure 2.14.

As the paper(Tan and Le, 2019) describes, they have applied fixed AutoAugment policy to training EfficientNets on ImageNet, which implies the pretrained EfficientNets is build on the AutoAugment policy. EfficientNet with AutoAugment shows an improvement in the accuracy and raises the robustness of the original model.

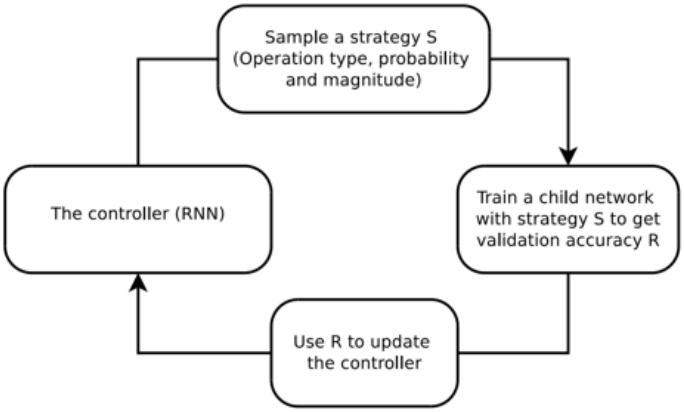


Figure 2.13: RNN controller for AutoAugment(Cubuk, Zoph, Mane, Vasudevan and Le, 2019).

(Rwightman, 2020*a*)

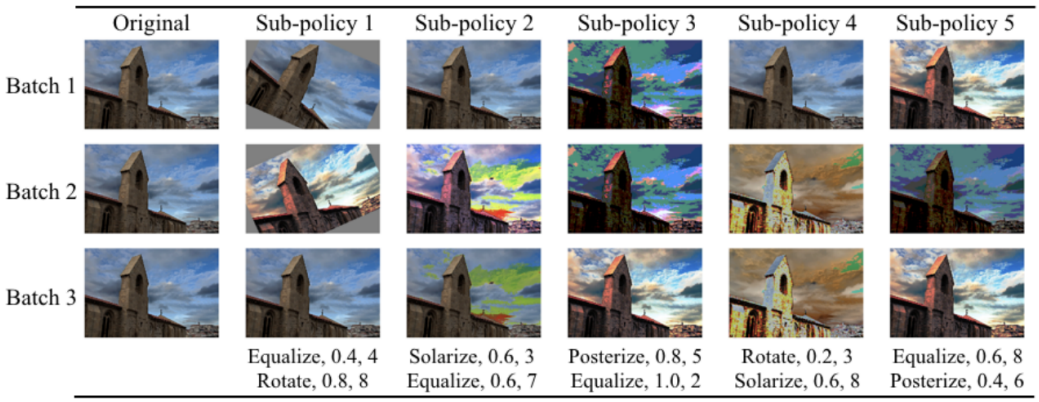


Figure 2.14: AutoAugment policy (2 operations with probability and magnitude under each sub-policy) (Cubuk, Zoph, Mane, Vasudevan and Le, 2019).

2.2.2 RandAugment

A separate search phase in AutoAugument that searching on a small dataset and transferring to large dataset could only produce the sub-optimal policy. Apart from the optimal issue, the search space of AutoAugument is enormous. Cubuk et al. (Cubuk, Zoph, Shlens and Le, 2019) creates a new method called RandAugment, which dramatically reduces the search space without the need for a separate proxy task.
There are only two interpretable hyperparameters, the magnitude value and the number of transformations applies sequentially. Each transformation is selected with a uniform probability.

The paper (Cubuk, Zoph, Shlens and Le, 2019) shows that average performance increases when more transformations are included. Some transformations make great contributions to accuracy like rotation while other operations reduce to the overall performance like posterizing. It also proves the assumption that optimal magnitude value depends on the size of both network and training datasets, and the magnitude value for different operations is not essential. Hence, the search space can be decreased.
RandAugment simply uses a grid search for optimal magnitude value, including the small search for the number of transforms. It provides a similar accuracy with AutoAugment but with a less computational cost. EfficientNet with RandAugment achieves remarkable results, especially for the small models.

2.2.3 Adversarial Propagation

Training adversarial images is a creative approach to increase the robustness of a model, and force the model less sensitive to texture distortions and focus more on shape information. However, too many noises lead to a decline in the accuracy.
Adversarial Propagation (Xie, Tan, Gong, Wang, Yuille and Le, 2019) use an auxiliary batch normalization, see Fig.2.15, to disentangle the mixed distributions between clean and adversarial images. It tries to keep the detachable feature maps into separate domains so that the features are still distinctive in the next layer. These two separate BNs solves the distribution mismatch of two datasets and guarantee the network extracts valuable features from both domains accurately and effectively.
Some other datasets that are hard to recognize are derived from ImageNet, like ImageNet-A, ImageNet-C. They are used to challenge the model from different aspects. Adversarial Propagation achieves extraordinary performance when it is used in EfficientNet architectures and combined with RandAugment(Rwightman, 2020*a*).

2.2.4 Self training with noise student

Self-training with noise student method with the EfficientNet achieves state of the art when training on the ImageNet.
The concept is composed of four steps.(Figure 2.16) First, train a teacher model on the labelled images like EfficientNet-B7. Then use the teacher model to generate pseudo labels on unlabeled images. Next, train an equal or larger student model on the combination of labelled images and pseudo labelled images with some injected input and model noises. Finally, the student model converts to a teacher model and produce new pseudo labels again. The paper also emphasizes the importance of injecting the noise into student model like RandAugument as input noise and dropout and stochastic depth as model noise.

This experiments uses pretrained Efficientnet-B7 as the teacher, training student model a derived EfficientNet-L2, which is much wider and deeper than B7 on unlabelled images. Then, this L2 is used as teacher and training a new L2 as student on the same unlabelled images but larger batch size again.

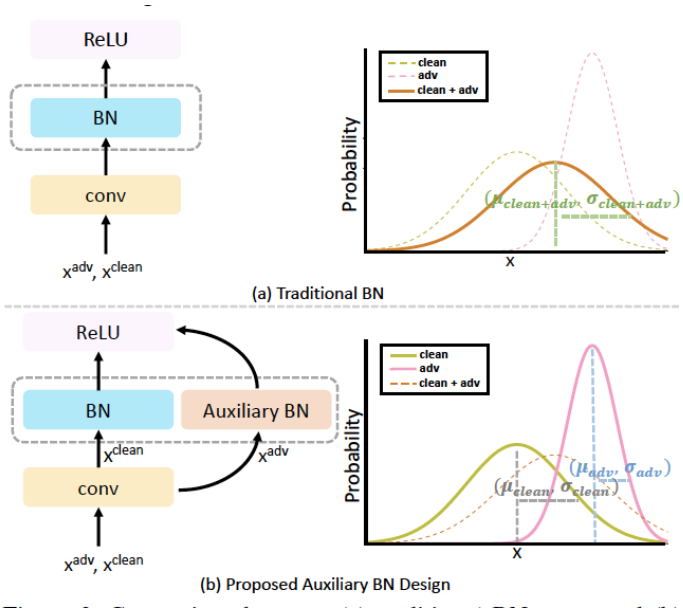


Figure 2.15: Auxiliary batch normalization (Xie, Tan, Gong, Wang, Yuille and Le, 2019).

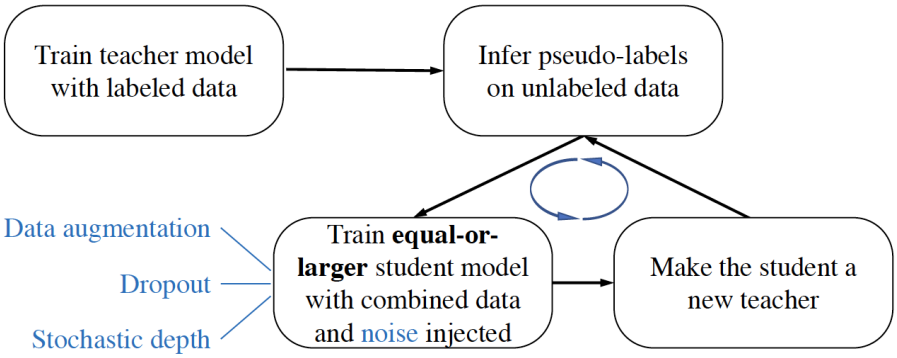


Figure 2.16: Self training with noise student(Xie, Hovy, Luong and Le, 2019).

Apart from the iterative training on L2, the paper (Xie, Hovy, Luong and Le, 2019) also proposes a study which use the same model as both teacher and student. It also leads to a consistent improvement of around 0.8% for all model sizes.
EfficientNets with Noisy Student provide a much better trade-off between model size and accuracy than pure EfficientNets.

Chapter 3

Implementation

3.1 Efficientnet implementation

EfficientNet-B0 is an another searched result from the MNasNet, which targets both accuracy and FLOPS (Float point operations per seconds, a measure for computer performance) rather than accuracy and real-world latency. The search space for Efficientnet-B0 is the same with what used in the MNasNet, which consists of Convolution Operations, Kernel size, Squeeze-and-excitation ratio, Skip Operations, number of filters and number of layers per block. The optimization goal is to maximize the $Accuracy(m) \times [FLOPS(m)/T]^w$, where T is target FLOPS and w is the hyper-parameter to control the trade-off. EfficientNet is slightly bigger than MNasNet that results in larger target FLOPS as well.

Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

Figure 3.1: B0 Baseline

The Fig. 3.1 shows the basic structure of EfficientNets. Stem stage has a regular 3x3 Conv, and head stage contains a 1x1 Conv for increasing dimension, pooling and a fully connected layer for classification. The feature extraction stages in the middle are composed by multiple mobile inverted bottleneck (MBConv Block) with a SE (Squeeze-and-Excitation) layer inside. MBConv block is used to extract the information of features on higher dimension and reduce back to origin channels eventually.

It contains four phases, Expansion phase, Depthwise phase, SE phase and reduction phase. Based on the expand ratio, the Expansion phase uses 1x1 kernel to increase dimension first and information are extracted in multiple channels by corresponding kernel size in Depthwise phase. Then, SE phase is implemented to assign weights to different channels by adaptively pooling the feature size to 1. Finally, pointwise convolution works to reduces the number of output channel. In addition, Batch Normalization and swish activation are appended to each convolutional layer and skip connection is placed for the block.

Once the baseline model is completed, compound scaling method plays an important role for the next scaling. Compound scaling method grid search three best coefficients: width (α), depth (β) and resolution (γ), as well as another compound coefficient (ϕ) to hold the number of FLOPS within the predefined constraints. The scaling in EfficientNets from three aspects width(w), depth(d) and resolution(r), where width coefficient multiples with input channel and output channel of every stage (except stem stage) and depth coefficient implements with the number of repeated layers to increase the depth of the model in feature extraction. For the resolution coefficient, it needs to pre-process before feeding the image into the network.

$$\textbf{Depth} : d = \alpha^{\phi} \tag{3.1}$$

$$\textbf{Width} : w = \beta^{\phi} \tag{3.2}$$

$$\textbf{Resolution} : r = \gamma^{\phi} \tag{3.3}$$

$$\textbf{s.t.} \alpha \times \beta^2 \times \gamma^2 \approx 2 \tag{3.4}$$

$$\alpha \geq 1, \beta \geq 1, \gamma \geq 1 \tag{3.5}$$

$$\textbf{Compound scaling} : (\alpha \times \beta^2 \times \gamma^2)^{\phi} \approx 2^{\phi} \tag{3.6}$$

Based on the equations above, FLOPS of a regular convolution op is proportional to d, w^2, r^2. Simply double the depth of model could double the FLOPS while double the width or resolution increases the FLOPS by four times. Hence, $\alpha \times \beta^2 \times \gamma^2$ is constrained by 2.

3.2 EfficientNet on Cifar10 and Cifar100

EfficientNets achieve outstanding performance on the transfer learning, which is pretrained on the ImageNet dataset and fine-tuning on other small dataset like Cifar10 and Cifar100. However, due to lack of ImageNet dataset, directly training EfficientNet on Cifar data and use the pretrained Efficienents to fine tuning on Cifar data are feasible.

Techniques that work well for other networks, do not always result in a good accuracy using EfficientNet (Rwightman, 2020*b*). Based on my experiments, the configuration of training EfficientNet on ImageNet is not suitable for other datasets when doing transfer learning. It does not get a reasonable results within the limited epochs, therefore I train these models using the general configuration on Cifar

dataset.
The configuration for training Cifar data can find in Appendix section.

3.3 EfficientNets for Diabetic Retinopathy Dataset

Data exploration
Diabetic Retinopathy dataset has five levels, No DR, Mild, Moderate, Severe and Proliferative DR. In total, it contains about 88K samples from both left and right eye of 44K patients. However, the dataset is imbalanced, which is the biggest challenge to solve. Majority of the samples are No DR (74%) and other four categories are also not in the same amounts (14%, 7%, 2% and 2% respectively). These retina images come from the different models and types of cameras, and appearance of both left and right eye. In addition, like any other real world data, noise and variation are inevitable in this dataset. Therefore, a major aim of this project is to develop robust algorithms that can handle these two situations.
The training samples contains 35K, and there are 42k and 11K samples for the public and private test. These sample images are not in the uniform high resolution which needs to resize before feeding into network model. Here are some samples extracting from original images, see Fig. 3.2.

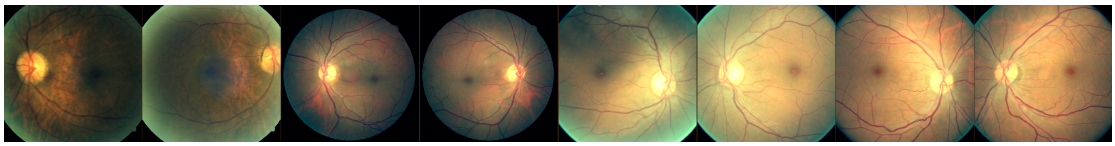
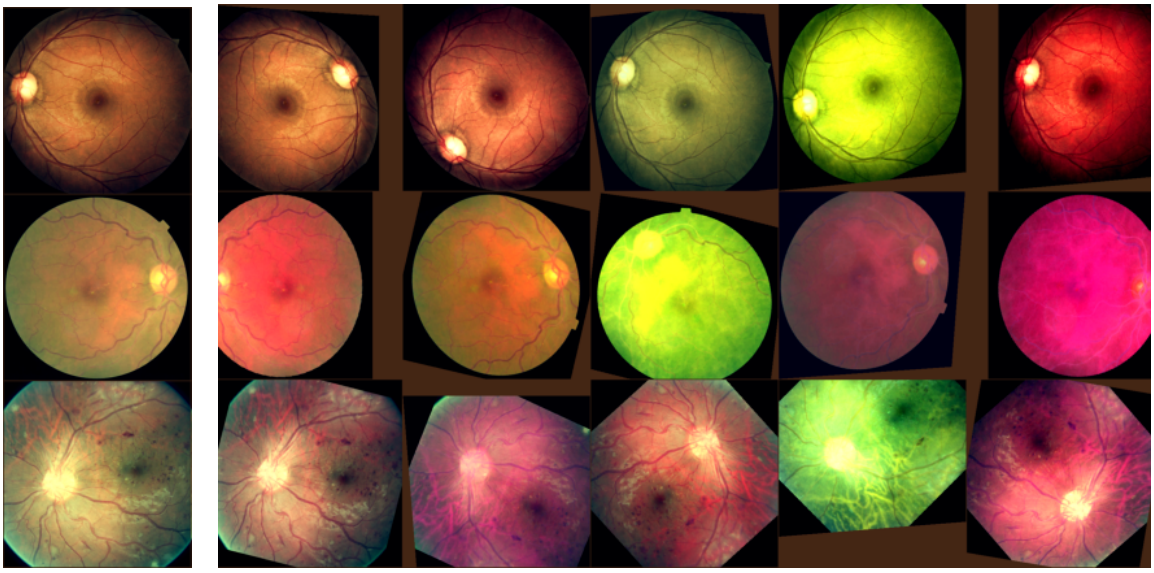


Figure 3.2: Diabetic Retinopathy images (4 pair of eyes).

These images shares similar shape and color, especially the left and right eyes from the same person.

As the Fig. 3.3 shown, there are some samples for original images and its RandAugment (with color injection) images. This RandAugment gives the weight for each augment operations. These weights forces the augment policy to focus more on the shape transformation like shear, translate instead of color changes.

Balanced Sampling
A problem occurs in the image sampling for both training and testing sets. The ratio between each classes in training, public test and private test are similar, which means in three classes sets, No DR is always slight over 70% while both severe and proliferate DR are about 2%. A prediction made by a model should not rely on the imbalance of the dataset. Therefore, the training set needs to resample and a balancedSampling is inevitable.
According to the class weight $(0.272, 2.875, 1.327, 8.047, 9.922)$, generate a new dataframe to store new samples. Directly copying the entire class images N times where N is



(a) Original (b) RandAugment

Figure 3.3: Original Images (**Left**) and RandAugument images (**Right**).

integer multiples, and for the decimal part, samples the class images with replacement. Additionally, 5 multipliers can given to each class weight to control the ratio of each class. Hence, each class is weighted sampled before feeding into RandAugment.
Pretrained EfficientNet models are comes from the resources (Rwightman, 2020*a*).

Chapter 4

Experimental Results

4.1 Cifar10 and Cifar 100

The experiment results for training the EfficientNets from the scratch and transfer Learning on Cifar10 are shown in the Table 4.1.

Model	Batch size	Learning from scratch	Transfer Learning	Results
EfficientNet-B0	32	0.8593	0.9673	0.981
EfficientNet-B3	32	0.8863	0.9769	—
EfficientNet-B5	8	0.8662	0.9754	—

Table 4.1: Cifar10 Performance

The best results comes from the paper (Tan and Le, 2019), which is the outcome of transfer learning from ImageNet and achieves 98.1% performance. Transfer Learning with the same pretrained Efficientnet-B0 gets 96.73%. There is a 1.37% gap between this transfer learning and best results on Efficienet-B0.
For different EfficientNet models, they share the same training configuration in each type of training except batch size, which can be found in the Appendix Section. The training accuracy for B0 and B3 has the improvement with the model size increased while B5 drops the performance due to the batch size.

The experiment results for training the EfficientNets from the scratch and transfer Learning on Cifar100 are shown in the Table 4.2.

Model	Batch size	Learning from scratch	Transfer Learning	Results
EfficientNet-B0	32	0.5704	0.8446	0.8810
EfficientNet-B3	32	0.6146	0.8651	—
EfficientNet-B5	8	0.5294	0.8595	—

Table 4.2: Cifar100 Performance

Same with Cifar10, Cifar100 has about 3.64% error between transfer learning and best outcome from the paper (Tan and Le, 2019) in EfficientNet-B0. Both B0 and B3

have improvement on Cifar10 and Cifar100 while Efficientnet B5 doesn't reach an ideal performance. This is mainly because the batch size.
Due to the limitation of the memory, B0 and B3 have batch size 32 while B5 only uses 8. Other training hyperparameters are the same. The batch size significantly affects the accuracy of model B5, which should exceed B3 if the same batch size is given. Moreover, although the paper (Tan and Le, 2019) doesn't specify their batch size for training Cifar10 and Cifar100 datasets, it gives the batch size 4096 for training on ImageNet.
A Finding for this experiment: large batch size is necessary for EfficientNet. The batch size heavily impacts the accuracy of learning from the scratch than transfer learning.
Therefore, the batch size significantly affects the accuracy for DR dataset as well, which should have a slight improvement if a large batch size id given.

4.2 EfficientNets for Diabetic Retinopathy Dataset

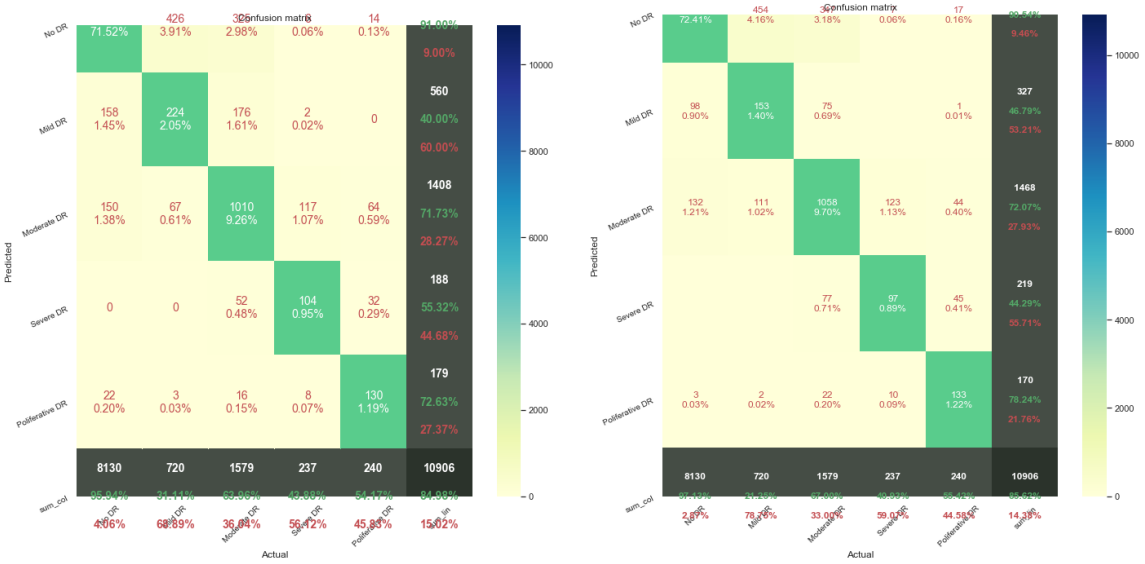
Without BalancedSampler and RandAugment
For the experiment without BalancedSampler and RandAugment, the performance are shown in the Table 4.3.

Model	parameters	Public test	Private test
Kaggle	—	0.8603	0.8496
ResNeXt-50 32x4d	25M	0.8498	0.8404
ResNeXt-101 32x8d	88M	0.8609	0.8519
EfficientNet-B5	30M	0.8562	0.8452
EfficientNet-B5-ap	30M	0.8538	0.8455
EfficientNet-B5-ns	30M	0.8708	0.8581

Table 4.3: DR Performance

There are two kinds of testing outcomes for different models. Public test is used for validation and private test is the holdout set. In the above table, A competition from Kaggle (*Diabetic Retinopathy Detection*, 2015) displays their best performance for this challenge but it doesn't declare the type and size of model. ResNeXt is the previous best model for this dataset collected from (Alansary, 2020). For the ResNeXt-50, although it just has about 25M parameters, it reaches 84.98% and 84.04% for the public and private sets.
Compare with ResNeXt-50, ResNeXt-101 has 1.11% and 1.15% improvement in the public and private datasets. However, the size of ResNeXt-101 is more than three times larger than ResNeXt-50.

There are three type of pretrained EfficientNet-B5 models are used in the experiment. EfficientNet B5, EfficientNet with adversarial propagation (EfficientNet-B5 with ap) and EfficeinetnNet-B5 with noise student (EfficientNet-B5 with ns). With



(a) ResNeXt-50 (b) EfficientNet B5

Figure 4.1: ResNeXt-50 and EfficientNet B5 public test confusion matrix.

regard to overall accuracy, they all show great performance in comparison to ResNeXt-50. They are all have the same model size and close to the size of ResNeXt-50.

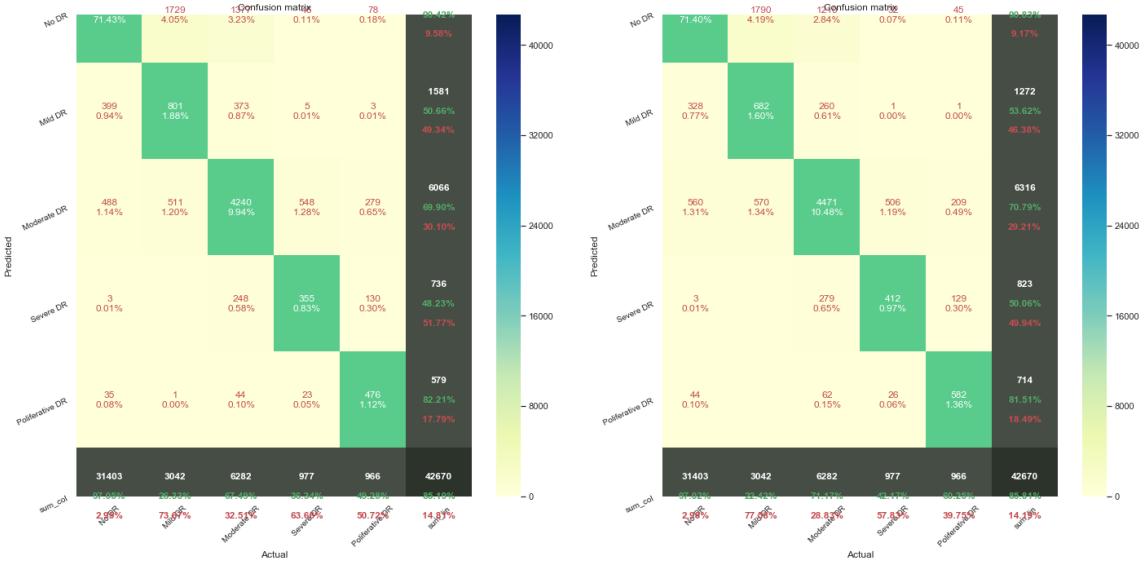
In addition to overall accuracy, it is inevitable to use confusion matrix to check the distribution of each class. The robustness of the model can be reflected from the confusion matrix. Compared with confusion matrix of ResNeXt-50 and EfficientNet-B5, see Fig. 4.1, EfficientNet-B5 has advantage in all five classes about precision and has better prediction in three of them regarding sensitivity. The worst case is sensitivity of Mild DR, where EfficientNet-B5 is far away from ResNeXt-50.

For EfficientNet-B5-ns, it achieves the 87.08% and 85.81% as the best performance of all models. The confusion matrix shows that the private test results works well in all classes except Mild DR in comparison to ResNeXt-101 private test results, see Fig. 4.2.

ROC, precision and recall curve

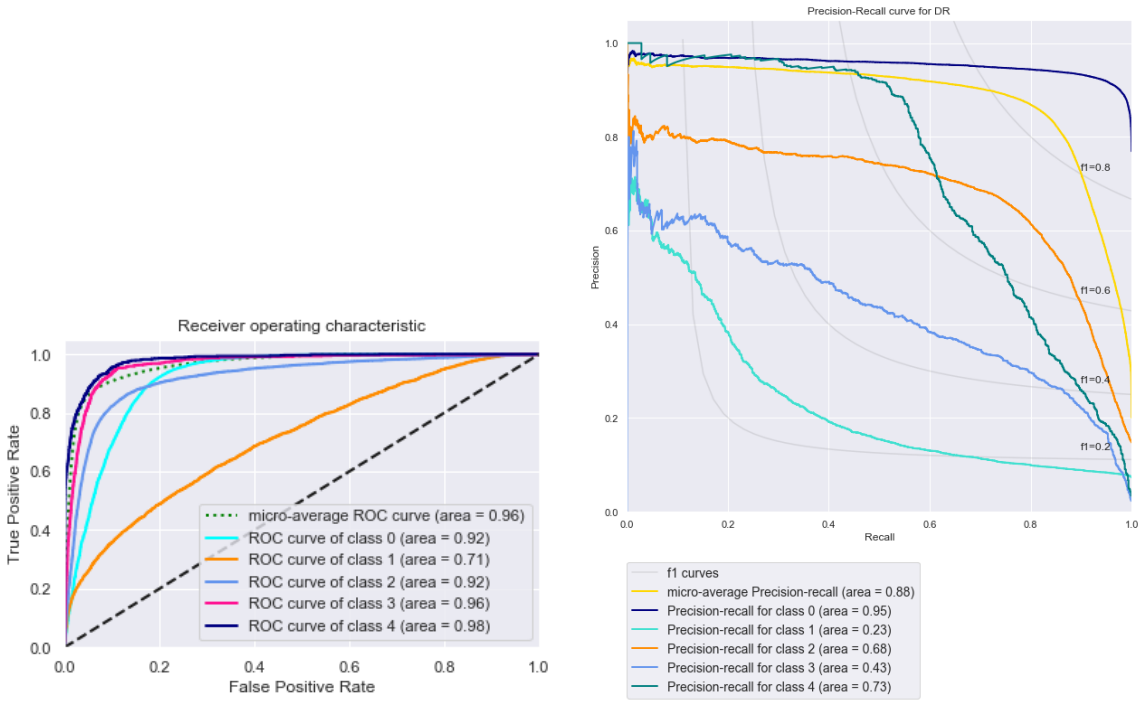
In the Fig. 4.3, ROC curve shows the situation of all five classes. A ROC curve for a class expresses the capability of a model distinguishes a class with others based on the overall accuracy. Class 1 (Mild DR) performaces much worse than other 4 classes. Therefore, the model is not able to distinguish Mild DR with other labels precisely.

For imbalanced multi-classes dataset, precision and recall curve represents more accurate than ROC curve because both precision and recall determinate the performance of multi-classes dataset instead of overall accuracy only. The Fig. 4.3 illustrates the same problem that class 1 (Mild DR) is not learned very well.



(a) ResNeXt-101 (b) EfficientNet-B5-ns

Figure 4.2: ResNeXt-101 and EfficientNet-B5-ns private test confusion matrix.



(a) ROC (b) precision and recall

Figure 4.3: ROC (a), precision and recall curve (b) for EfficientNet-B5-ns private test.

With BalancedSampler and RandAugment

For the experiment with BalancedSampler and RandAugment, the performance are

shown in the Table 4.4.

Model	Public test	Private test
EfficientNet-B5	0.7949	0.7847
EfficientNet-B5 with ap	0.8175	0.8057
EfficientNet-B5 with ns	0.8284	0.8160

Table 4.4: EfficientNet-B5 with RandAugment and BalancedSampler Performance.

The performance of EfficientNet-B5 with RandAugment and BalancedSampler is not as good as without using RandAugment and BalancedSampler, see Fig 4.4.

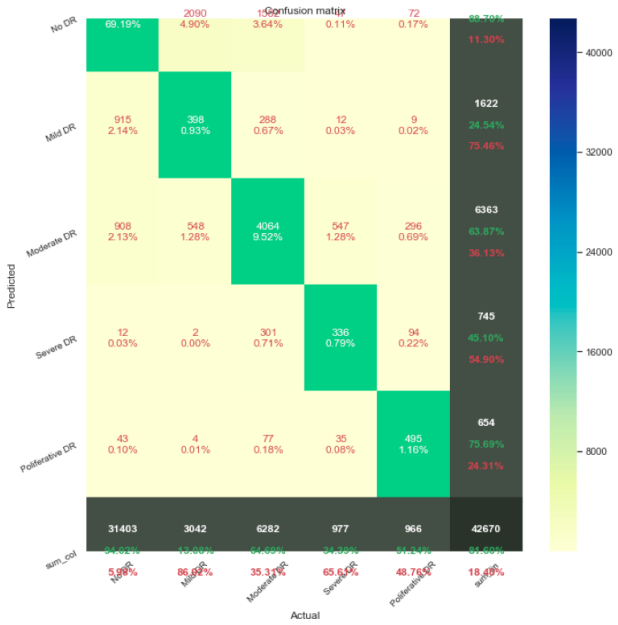


Figure 4.4: EfficientNet-B5 with NS confusion matrix

In conclusion, EfficientNet-B5-ns achieves a better performance than ResNeXt-101 in four of five classes, and the overall accuracy is the best. However, the RandAugment doesn't performs well with BalancedSampler.

4.3 EfficientNets for other Diabetic Retinopathy Dataset

APTOS (Asia Pacific Tele-Ophthalmology Society) (*APTOS 2019 Blindness Detection*, 2019) is a dataset for detecting diabetic retinopathy with the same five labels. The following Table 4.5 is the performance directly measured by the EfficientNet-B5 models. The training datasets is much smaller, which contains only 3662 samples. This performance is achieved directly testing on the APTOS training set. Same with DR dataset, APTOS training set is also imbalanced, which has No DR (49.29%),

Mild DR (10.10%), Moderate DR (27.28%) Severe DR (5.27%) and Poliferative DR (8.06%). Therefore, this small dataset can be used to better represent the performance of our model in real world.

Model	**performance**
EfficientNet-B5	0.7073
EfficientNet-B5-ap	0.7261
EfficientNet-B5-ns	0.7425

Table 4.5: EfficientNet B5 Performance in APTOS dataset

Although the overall performance of EfficientNet B5 models are not ideal, the confusion matrix from Fig. 4.5 shows where the problem is. For EfficientNet-B5-ns, majority of Mild DR are misclassified as Moderate DR. The sensitivity of detecting Mild DR for this model is only 0.01%, which is not reasonable. For other four classes, they reach the similar sensitivity with Fig. 4.2.

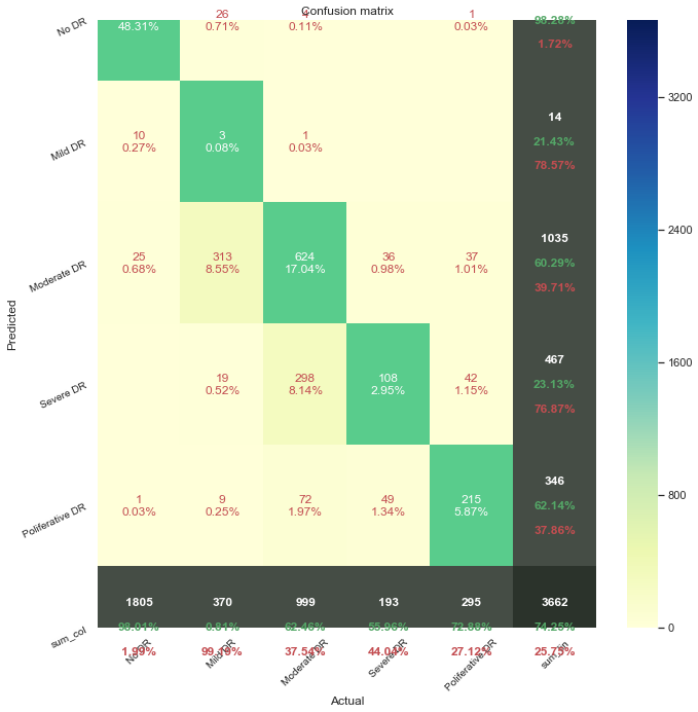


Figure 4.5: EfficientNet-B5 with NS confusion matrix for Aptos.

Consequently, there is no standard criterion to measure Mild and Moderate DR. The labels for Mild and Moderate DR could contains a large number of noises. The gold truth for Mild and Moderate DR is not reliable and need to rectify or re-label..

Chapter 5

Conclusion

EfficientNets provides a systematic way to scaling a model in both accuracy and efficiency. Compound scaling method can be applied to many other models to make full use of the limited resource. EfficientNets can provide the different levels of models to fit different datasets. It also performs well when applying transfer learning to small datasets. Many other techiques can be used with EfficientNets like RandAugment, Adversarial Propagation and self training with noise student. These methods generate a pipeline to serve EfficientNets family and enhance capability of EfficientNet.
In this project, many EfficientNet models are used to classify the retinal images. EfficientNet-B5-ns achieves the best performance, which has 87.07% for public test and 85.81% for private test. Some techniques like RandAugment are used in order to improve the performance and handle the imbalance of dataset.

Future work

1. Compound Scaling method can be applied to other models like ResNeXt-50 to check if these scaled models perform better in the DR dataset.

2. Give a larger batch size for EfficientNet-B5 and try some larger models like B6 and B7.

3. Re-considering about the label of Mild DR classes and try to use unlabeled data to improve performance like self training with noise student.

Bibliography

Alansary, A. (2020), 'Dr-detection'.
URL: *https://github.com/amiralansary/DR-Detection* pages 2, 22

APTOS 2019 Blindness Detection (2019).
URL: *https://www.kaggle.com/c/aptos2019-blindness-detection/overview* pages 25

Cifar (2020).
URL: *http://www.cs.toronto.edu/ kriz/cifar.html* pages 1

Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V. and Le, Q. V. (2019), Autoaugment: Learning augmentation strategies from data, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 113–123. pages 13, 14

Cubuk, E. D., Zoph, B., Shlens, J. and Le, Q. V. (2019), 'Randaugment: Practical automated data augmentation with a reduced search space', *arXiv preprint arXiv:1909.13719* . pages 1, 14, 15

Diabetes.co.uk (2019), 'Diabetic retinopathy'.
URL: *https://www.diabetes.co.uk/diabetes-complications/diabetic-retinopathy.html* pages 2

Diabetic Retinopathy Detection (2015).
URL: *https://www.kaggle.com/c/diabetic-retinopathy-detection* pages 2, 22

He, K., Zhang, X., Ren, S. and Sun, J. (2016), Deep residual learning for image recognition, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 770–778. pages 1, 4, 5

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* . pages 1, 4

Hu, J., Shen, L. and Sun, G. (2018), Squeeze-and-excitation networks, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 7132–7141. pages 6, 7

ImageNet (2020).
URL: *http://www.image-net.org/* pages 1

Ramachandran, P., Zoph, B. and Le, Q. V. (2017), 'Searching for activation functions', *arXiv preprint arXiv:1710.05941* . pages 1, 10, 12

Rwightman (2020*a*), 'rwightman/pytorch-image-models'.
URL: *https://github.com/rwightman/pytorch-image-models* pages 14, 15, 20

Rwightman (2020*b*), 'rwightman/pytorch-image-models'.
URL: *https://github.com/rwightman/pytorch-image-models/issues/11* pages 18

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A. and Chen, L.-C. (2018), Mobilenetv2: Inverted residuals and linear bottlenecks, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 4510–4520. pages 5, 10

Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A. and Le, Q. V. (2019), Mnasnet: Platform-aware neural architecture search for mobile, *in* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition', pp. 2820–2828. pages 9, 10, 11

Tan, M. and Le, Q. V. (2019), 'Efficientnet: Rethinking model scaling for convolutional neural networks', *arXiv preprint arXiv:1905.11946* . pages 1, 2, 12, 13, 21, 22, 30

Xie, C., Tan, M., Gong, B., Wang, J., Yuille, A. and Le, Q. V. (2019), 'Adversarial examples improve image recognition', *arXiv preprint arXiv:1911.09665* . pages 15, 16

Xie, Q., Hovy, E., Luong, M.-T. and Le, Q. V. (2019), 'Self-training with noisy student improves imagenet classification', *arXiv preprint arXiv:1911.04252* . pages 16

Xie, S., Girshick, R., Dollár, P., Tu, Z. and He, K. (2017), Aggregated residual transformations for deep neural networks, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 1492–1500. pages 5, 6

Zoph, B., Vasudevan, V., Shlens, J. and Le, Q. V. (2018), Learning transferable architectures for scalable image recognition, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 8697–8710. pages 1, 7, 8, 9

Appendix

5.1 Training configuration

ImageNet
Training process for efficientnet on ImageNet closes to the MNasNet, which uses RMSProp optimizer with 0.9 momentum, Batch Normalization 0.99 with weight decay 1e-5, initial learning rate 0.256 that decays by 0.97 every 2.4 epochs and linearly increased dropout from B0 to B7. Additionaly, fixed AutoAugment policy and stochastic depth are also used to perform regularization. MNasNet use batch size 4K and doesn't specify the number of epoch.(Tan and Le, 2019)
Cifar10 and Cifar100
Training Efficientnet B0 from scratch
Batch size 32, Adam optimizer with weight decay 1e-5, learning rate 0.001, Reduce Learning scheduler factored a half by 2 patience, 40 epochs.

For pretrained Efficientnet B0
Batch size 32, SGD optimizer with weight decay 1e-5, learning rate 0.001, momentum 0.99, Reduce Learning scheduler factored a half by 1 patience, 40 epochs.
Image normalization for Cifar10 mean:(0.4914, 0.4822, 0.4465) and std: (0.247, 0.243, 0.261)
Image normalization for Cifar100 mean:(0.5071, 0.4867, 0.4408) and std: (0.2675, 0.2565, 0.2761)

EfficientNet-B5-ns
Fine tuning
Batch size 8, Adam optimizer with weight decay 1e-5, learning rate 0.0001, Step learning scheduler with step size 0.5 for every 2 epochs. 40 epochs.
Image normalization for DR dataset mean: (0.42, 0.22, 0.075), std: (0.27, 0.15, 0.081).

5.2 EfficientNet

Code for EfficientNets on Cifar: https://github.com/Rain9876/EfficientNet2D
Code for DR-Detection: https://github.com/amiralansary/DR-Detection