

VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY  
UNIVERSITY OF TECHNOLOGY  
DIVE INTO CODE MACHINE LEARNING ENGINEER COURSE



DIVE INTO CODE

Graduation Assignment

---

# Image Detection with EfficientNetV2 models

---

**Lecturer:** CEO NORO Hiroyoshi

**Student:** Doan Anh Tien - 1852789

HO CHI MINH CITY, 26TH FEBRUARY 2022

## Contents

<b>List of Figures</b>	<b>2</b>
<b>1 Introduction</b>	<b>2</b>
1.1 Student self-introduction . . . . .	2
1.2 Competition overview . . . . .	3
<b>2 Background &amp; Methodologies</b>	<b>4</b>
2.1 EfficientNet . . . . .	4
2.1.1 Optimization Problem . . . . .	5
2.1.2 Scaling Dimensions . . . . .	7
2.1.3 Compound Scaling . . . . .	7
2.1.4 Model architecture . . . . .	8
2.1.5 Performance experiments . . . . .	9
2.2 EfficientNetV2 . . . . .	9
2.2.1 NAS & Model architecture . . . . .	9
2.2.2 Progressive Learning . . . . .	11
2.2.3 ImageNet1k and ImageNet21k . . . . .	11
2.2.4 EfficientNetV2 Performance . . . . .	12
<b>3 Approach overview</b>	<b>12</b>
3.1 First Approach . . . . .	12
3.2 Second Approach . . . . .	14
<b>4 Data Understanding</b>	<b>16</b>
<b>5 Pre-processing</b>	<b>18</b>
<b>6 Modeling</b>	<b>19</b>
6.1 EfficientNetV2 variant selection . . . . .	19
6.2 Model Architecture . . . . .	19
<b>7 Discussion</b>	<b>21</b>
<b>8 Conclusion</b>	<b>22</b>
<b>9 Acknowledgements</b>	<b>22</b>
<b>Terms</b>	<b>23</b>

## List of Figures

1	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 the proposed compound scaling method that uniformly scales all three dimensions with a fixed ratio by creators of EfficientNet [TL20] . . . . .	4
2	Formula with coefficients for optimization . . . . .	6
3	Architecture expansion using compound scaling method . . . . .	8
4	Comparison of MBConv and fused-MBConv structure . . . . .	10
5	First approach pipeline . . . . .	13
6	Second approach pipeline . . . . .	15
7	5K Compliance dataset . . . . .	16
8	Missing values from the 5K datasets . . . . .	17
9	Load image function . . . . .	18
10	Tensorflow Dataset creation . . . . .	18
11	Model architecture . . . . .	20
12	Full Model architecture . . . . .	21
13	WandB - Weights Biases . . . . .	21

## 1 Introduction

### 1.1 Student self-introduction

Machine Learning has been being adapted to the latest technologies in the recent years and has oriented many fields to make it as a part of business operations, chain production, automation and reality practical problem solutions. Arouse as an advanced technique combined with such aspects like Data Mining, Data Analyzing, Machine Learning has set many useful applications, ranging from simple tasks like forecasting, classification, anomaly detection, to complex procedures like deep learning, object detection, natural language processing.

There is no doubt that machine learning is becoming well-known these days not only to developer, but also common businesses due to its simplicity, all thanks to the efforts of the creator to make the libraries, tools and framework to be more concise and user-friendly.

As as student that major in Computer Science and Engineering, my interests in Data Science and ML has been charmed with the appearance of the Machine Learning Engineer course by DIVE INTO CODE. Being a learner for more than one year, the course has convey a lot of useful theories and practical problem to brush my skills. This report is conducted to demonstrate the final assignment for my graduation at DIVE INTO CODE, and everything here is served to describe my work in details.

For the project, I chose to participate to a contest called **Zalo AI Challenge 2021**, the fourth-year of an annual online competition for Vietnam's AI engineers to explore AI technologies and impact life in exciting new ways. In 2021, the competition consist of 3 main problems, and my work is related to one of them, the **5K Compliance**.

## 1.2 Competition overview

In this section, I will briefly explain about the meaning of the name, the description and rules of the challenge, and the dataset that need to be interpreted on.

During the Covid-19 outbreak, the Vietnamese government pushed the "5K" public health safety message. In the message, masking and keeping a safe distance are two key rules that have been shown to be extremely successful in preventing people from contracting or spreading the virus. Enforcing these principles on a large scale is where technology may help. In this challenge, you will create algorithm to detect whether or not a person or group of individuals in a picture adhere to the "mask" and "distance" standards.

### Basic rules

We are given the dataset contains images of people either wearing mask or not and they are standing either close of far from each other. Our mission is to predict whether the formation of these people adhere the 5k standard.

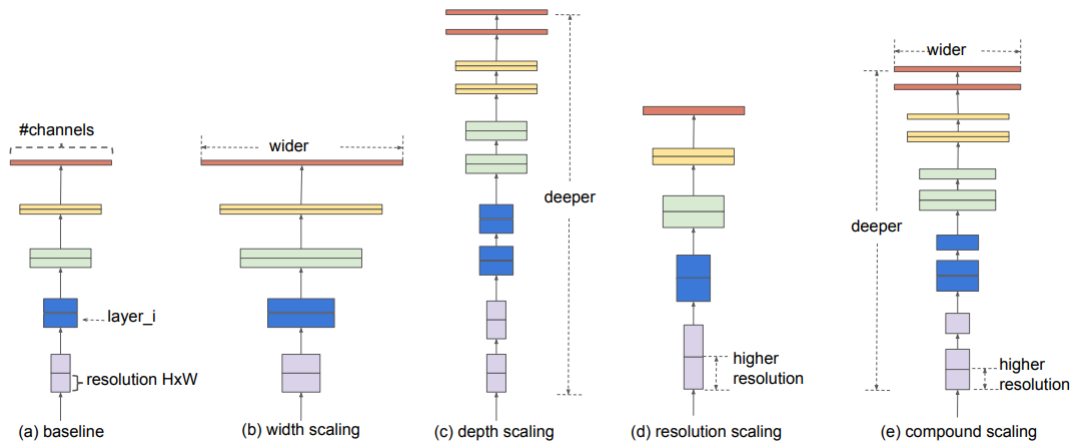
The 5k standard is also based on the two conditions, mask (0 = not wearing, 1 = wearing) and distancing (0 = too close, 1 = far enough). People that adhere the 5k standard will not likely to expose the virus to each other in case they did caught it before, and it is to prevent the spread of the COVID-19 pandemic through people interactions.

## 2 Background & Methodologies

In this section, we will have a discussion about the methodologies, models that our project has implemented and applied. This will enable reader to understand the mechanism behind them and how useful they are in this specific image detection field. The first part will mainly focus on the EfficientNet baseline model first.

### 2.1 EfficientNet

Convolutional Neural Networks (ConvNets) are frequently constructed with a limited resource budget and then scaled up for improved accuracy as more resources become available. Creators of EfficientNet [TL20] has conducted a research paper, in which they study model scaling and learn that better accuracy or resources efficiency can be reach if we can balance the network depth, width, and resolution. They proposed a new scaling method uses a simple but very effective *compound scaling method* to scale all depth/width/resolution dimensions equally. The researchers also design a family of models to evaluate its performance and size. The dataset used in their scope is **ImageNet** which is a large database of annotated photographs intended for computer vision research. It has more than 14 million images that capture different objects and indicate what they are, and in at least one million of the images, bounding boxes are also provided.



**Figure 1:** 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 the proposed compound scaling method that uniformly scales all three dimensions with a fixed ratio by creators of EfficientNet [TL20]

When training model with more computational resources, we can either increase the network, width or resolution. The factors (coefficients) can be determined by a small grid search on the original smaller model. Figure 1 demonstrate the difference between the compound scaling of the creators and other convetional methods.

### 2.1.1 Optimization Problem

As the model scaling factor is quite complex yet still be one of the interesting of these researches, so I had tried my best to make the explanation simpler.

First, the ConvNet Layer  $i$  can be defined with a function:

$$Y_i = F_i(X_i) \quad (1)$$

where:

$F_i$  = can be seen as the activation function used in that layer  
 $X_i$  = input tensor with shape  $\langle H_i, W_i, C_i \rangle$   
 $Y_i$  = output tensor  
 $H_i, W_i$  = spatial dimensions  
 $C_i$  = channel dimension

Then, a whole ConvNet  $\eta$  is composed of layers with Hadamard Product (component-wise multiplication for matrices):

$$\eta = F_k \odot \dots \odot F_2 \odot F_1(X_1) = \odot_{j=1\dots k} F_j(X_1) \quad (2)$$

To explain the notation of equation (2), the hadamard product can be described as the product of two 3x3 matrices below:

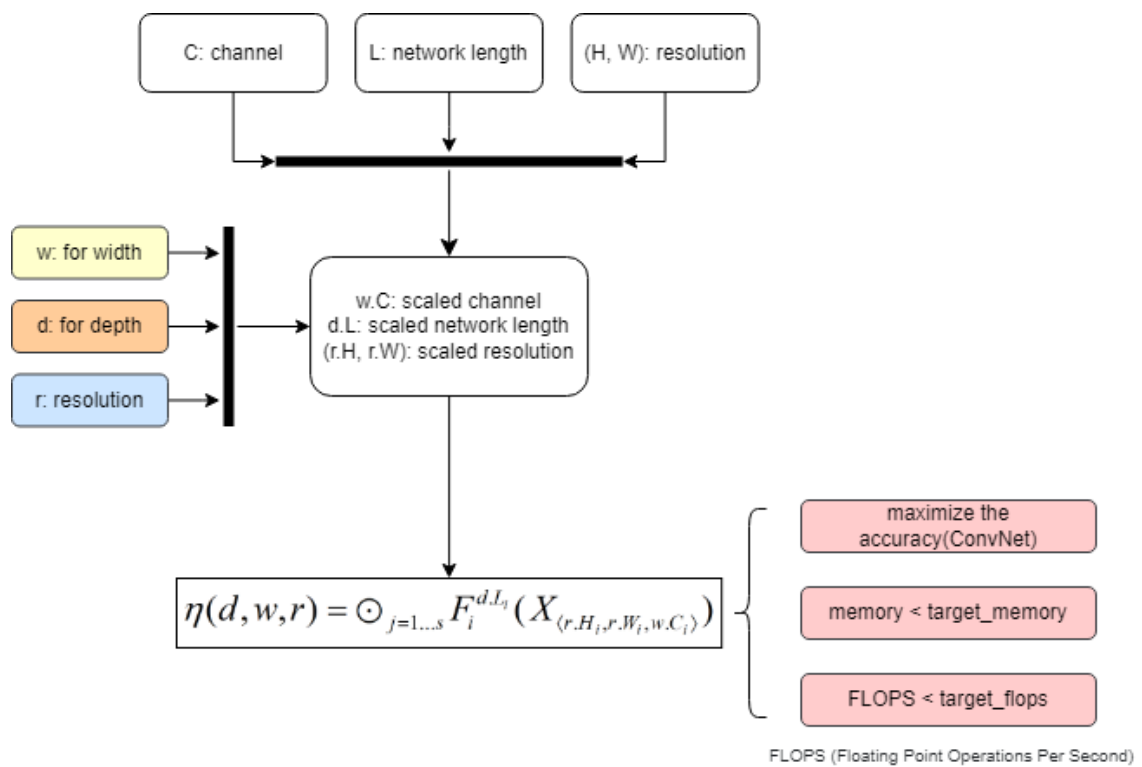
$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \odot \begin{bmatrix} j & k & l \\ m & n & o \\ p & q & r \end{bmatrix} = \begin{bmatrix} aj & bk & cl \\ dm & ne & fo \\ gp & hp & ir \end{bmatrix}$$

In a network, the operation of layers are often repeated  $L_i$  times in each stage, thus the general formula of  $\eta$ :

$$\eta = \odot_{j=1\dots s} F_i^{L_i}(X_{\langle H_i, W_i, C_i \rangle}) \quad (3)$$

So far, we have know the formula that describe the architecture  $F_i$ . Most regular ConvNet designs focus on finding the best  $F_i$ , meanwhile, the model scaling opt to change the network length ( $L_i$ ), network width ( $L_i$ ), and/or resolution ( $H_i, W_i$ ) without changing  $F_i$  predefined in the baseline network. However, by changing these factors, it lead to the fact that a large design space will occurs in order to investigate different  $L_i, C_i, H_i, W_i$  for each layer.

Creators of EfficientNet came up with an idea that restrict all layers to be scaled up uniformly by a constant ratio. Their target is to maximize the model accuracy for any given resources constraints, which can be described as follow:



**Figure 2:** Formula with coefficients for optimization

The yellow, orange and blue rectangles are the scaling constants for the method, the white box will then take all required parameters to form the formula, and the red rectangles represent for their target by applying the model scaling. In next section, we will discuss some problems of this optimization problem.

### 2.1.2 Scaling Dimensions

As we have know that the formula from Figure 2, choosing the optimal parameters  $d$ ,  $w$ ,  $r$  is quite tricky since they depend on each other, and in some different resource constraints these values may change. Therefore, the conventional method like Figure 1 (a), (b), (c), (d) from mostly scale ConvNets in one of these dimensions, and here is the table that sum-up the performance of each way:

	Pros	Cons
<b>Depth (d)</b>	- Capture richer & more complex features	- Face vanishing gradient problem
<b>Width (w)</b>	- Capture more fine-grained features - Easier to train	- Hard to capture higher level feature (wide but shallow network) - Face vanishing gradient problem
<b>Resolution (r)</b>	- Capture more fine-grained patterns - Higher resolution -> higher accuracy, capable of achieves state-of-the-art	- Face vanishing gradient problem

Table 1: Model conventional scaling methods

From their observation, the researchers conclude that "scaling up any dimension of network width, depth, or resolution improves accuracy, but the accuracy gain diminishes for bigger models" ([TL20], 2020, p.4).

### 2.1.3 Compound Scaling

In the later experiment, the researchers pointed out that balancing different dimensions scaling ratio would be better than conventional single-dimension scaling. And to validate this point, they use different network depths and resolutions, altogether with the width scaling. Results show that, by using width scaling without changing depth ( $d$ ) and resolution ( $r$ ), the accuracy will be diminished. However, with deeper  $d$  and higher resolution  $r$ , width scaling method achieves better accuracy while maintaining the same Floating Point Operations Per Second (FLOPS) cost. As a result, they gave a second point of view that "In order to pursue better accuracy and efficiency, it is critical to balance all dimensions of network width, depth, and resolution during ConvNet scaling." ([TL20], 2020, p.4).

As said, the creators of EfficientNet proposed a new **compound scaling method**, which use a coefficient  $\phi$  to uniformly scales network width, depth and resolution:



$$\begin{aligned}
 \text{depth} : d &= \alpha^\phi \\
 \text{width} : w &= \beta^\phi \\
 \text{resolution} : r &= \gamma^\phi
 \end{aligned} \tag{4}$$

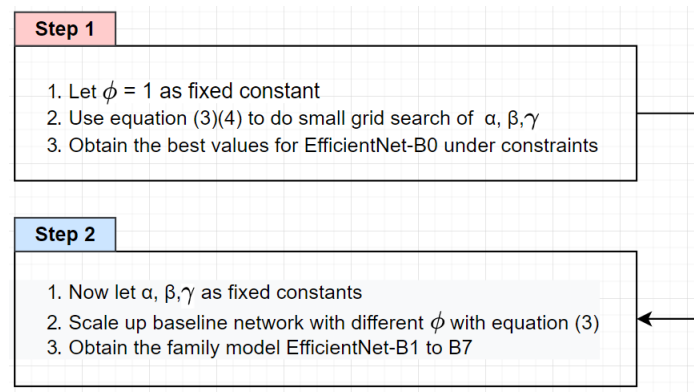
The coefficient  $\phi$  will determine how many more resources that will be used for model scaling operation and it is specified by user. Meanwhile,  $\alpha, \beta, \gamma$  will specify how to assign these extra resources to the network width, depth and resolution.

What is more notable is that when increase  $d, w, r$  in convolution operation, the FLOPS also increases (proportion to those factors). Specifically, when double the depth, FLOPS will proportional to  $d$  and is doubled. And when we double the width or resolution will cause the FLOPS to be increased four times since it is proportional to  $w^2, r^2$ . As convolution operation is a major part in ConvNets, scaling the model with equation 4 will increase the FLOPS by  $(\alpha.\beta^2.\gamma^2)^\phi$ . The researchers opted to limit this increase by putting a constraint  $\alpha.\beta^2.\gamma^2 \approx 2$ , thus the total FLOPS will approximately increase by  $2^\phi$  for any given  $\phi$ .

#### 2.1.4 Model architecture

The researchers decided to design a new mobile-size baseline called EfficientNet to evaluate their scaling method. They adapt and use the multi-objective neural architecture search inspired by [Tan+19] to develop the baseline that optimizes both accuracy and FLOPS. The researchers come up with sort of optimization formula that involve accuracy, FLOPS, the target FLOPS and a hyperparameter for controlling the trade-off between accuracy and FLOPS.

From the beginning, the researchers completed the baseline EfficientNet-B0, and they applied the proposed compound scaling method to scale it up with two steps:



**Figure 3:** Architecture expansion using compound scaling method

### 2.1.5 Performance experiments

The background section is quite long at this moment, so I just summarize the main points instead of providing the original proofs in form tables (figures) from the research.

First, the researchers evaluate their scaling method on the existing ConvNets such as MobileNets and ResNet, along with the ImageNet database. The results showed that their compound scaling method, comparing to other single-dimension scaling method, has *improved the accuracy* on all evaluated models and prove its effectiveness.

Second, they train the EfficientNet models on the ImageNet database using an RMSProp optimizer with the configured parameters as that of [Tan+19], and other customized settings. The EfficientNet achieved much better accuracy than GPipe, ConvNets, ResNets while being dramatically smaller in size and computationally cheaper than mentioned models (fewer FLOPS). One example is that the EfficientNet-B3 achieves higher accuracy than ResNeXt101 [Xie+17] using 18x fewer FLOPS.

**Conclusion:** The creators of EfficientNet and compound scaling method has studied and proposed their solution that can balance the network width, depth and resolution in a more principled way for any given resources constraints. Based on this factor, they develop a mobile-size EfficientNet model that can be scaled effectively to different sizes, with a much better state-of-the-art accuracy and fewer parameters and FLOPS. Their work has been evaluated on the ImageNet and other five commonly used datasets and proved its effectiveness and influence.

## 2.2 EfficientNetV2

A year later, the same creators Mingxing Tan and Quoc V.Le also proposed EfficientNetV2 [TL21], a new family of convolutional networks and also an upgraded version of EfficientNet, which has faster training speed and better parameter efficiency compared to the previous models.

The researchers use a combination of training-aware Neural Architecture Search (NAS) and scaling. They indicated that the training process can be sped up by increasing the image size, but it may results in the drop of accuracy. Therefore, they proposed a new method of progressive learning which adaptively adjusts regularization along with image size. The EfficientNetV2 achieved a better accuracy than previous models, performed significantly faster while using the same computational resources (FLOPS).

### 2.2.1 NAS & Model architecture

Before heading to the explanation of progressive learning, we will have a look at the NAS and the model structure of EfficientNetV2.

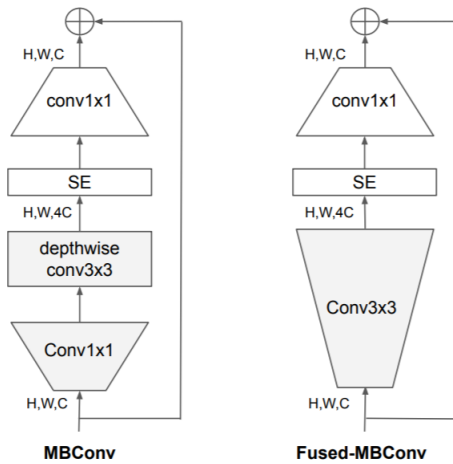
The researchers develop a training-aware NAS framework with EfficientNet as backbone that optimize accuracy, parameter efficiency, and training efficiency. The NAS search adapt an mechanism which is a search space that consist of multiple design choices for convolutional operation types, number of layers, kernel size, expansion ratio.

*In short, the researchers will use the proposed training-aware NAS to search for the best combinations of architecture design in order to improve the training speed.*

As mentioned, the search space includes many options, the researchers managed to remove unnecessary search options like pooling skip operation since it was not used in the EfficientNet. This lead to the smaller search space so they can add reinforcement learning into it. And in addition to that, they reuse the same channel sizes from the backbone which has been used in [TL20].

The researchers come up with the first version EfficientNetV2-S (S stands for Small). There are major differences between the EfficientNetV2 and the backbone EfficientNet:

1. Use both MBConv [San+19][TL20] and the fused-MBConv [GA20]



**Figure 4:** Comparison of MBConv and fused-MBConv structure

2. EfficientNetV2 prefers smaller expansion ratio for MBConv since smaller expansion ratios tend to have less memory access overhead
3. EfficientNetV2 prefers smaller  $3 \times 3$  kernel sizes, but it adds more layers to compensate the reduced receptive field resulted from the smaller kernel size

4. EfficientNetV2 completely removes the last stride-1 stage in the original EfficientNet

Similarly to their previous work [TL20], they also scale up the EfficientNetV2-S to obtain the expanded EfficientNetV2-M/L by using the compound scaling method. A small experiment has been conducted to compare the training speed of EfficientNetV2 and other models in the scenario: without **progressive learning**. With the training-aware NAS and scaling, the EfficientNetV2 model witnessed a **better training speed** than the other recent models.

### 2.2.2 Progressive Learning

As the term progressive learning has been indicated once in Section 2.2, here we will investigate further the effect of this proposed method.

Basically, some works dynamically change image sizes, which may cause in drop of accuracy. This maybe come from the unbalance regularization. Therefore, the researchers consider that instead of having fixed regularization, we should adjust it accordingly to the image size changes.

Since the procedure of progressive learning is hard to described in a detailed mathematically way, I will re-use a statement from their paper that can briefly demonstrate its mechanism: "...in the early training epochs, we train the network with smaller images and weak regularization, such that the network can learn simple representations easily and fast. Then, we gradually increase image size but also making learning more difficult by adding stronger regularization." Some of the types of regularization are Dropout, RandAugment, Mixup.

### 2.2.3 ImageNet1k and ImageNet21k

Later in the Approach Section, the report will mention about the ImageNet1k or ImageNet21k along with the model architecture, so I manage to explain what are they in this small section.

Technically, the ImageNet1k contains about 1.28M training images and 50,000 validation images with 1000 classes. Meanwhile, the ImageNet21k (Full ImageNet, Fall 2011 release) contains about 13M training images with 21,841 classes. The researchers did use the ImageNet21k to pretrain the EfficientNetV2, following by fine-tuning it on the ImageNet1k using the cosine learning rate decay. In the end, the EfficientNetV2 that is trained on imagenet-21k and fine-tuned on ImageNet1K has improved the accuracy, used 2.5 times fewer parameters and 3.6 times fewer FLOPS, while running 6-7 times faster.

### 2.2.4 EfficientNetV2 Performance

In conclusion, the EfficientNetV2 surpasses earlier models while being more quicker and more efficient in parameters, thanks to training-aware NAS and model scaling. The researchers presented an enhanced approach of progressive learning that raises picture size and regularization simultaneously during training to speed up the process even more. Extensive testing shows that their EfficientNetV2 performs well on ImageNet and CIFAR/Flowers/Cars. In addition, EfficientNetV2 trains up to 11 times quicker while being 6.8 times smaller than EfficientNet and other recent research. They also scale up the baseline EfficientNetV2-S into larger size model like EfficientV2-M/L, while also scale down into different smaller size like EfficientNetV2-B0/B1/B2/B3 to compare with original EfficientNet variants, along with some pretrained and fine-tuned version.

## 3 Approach overview

During the interpretation of models and process testing, I did come up with an idea that quite basic and brute force as I ignore the missing values in the provided dataset. After completing the first approach, I thought about my work again and decided to recreate the pipeline with another longer yet better process.

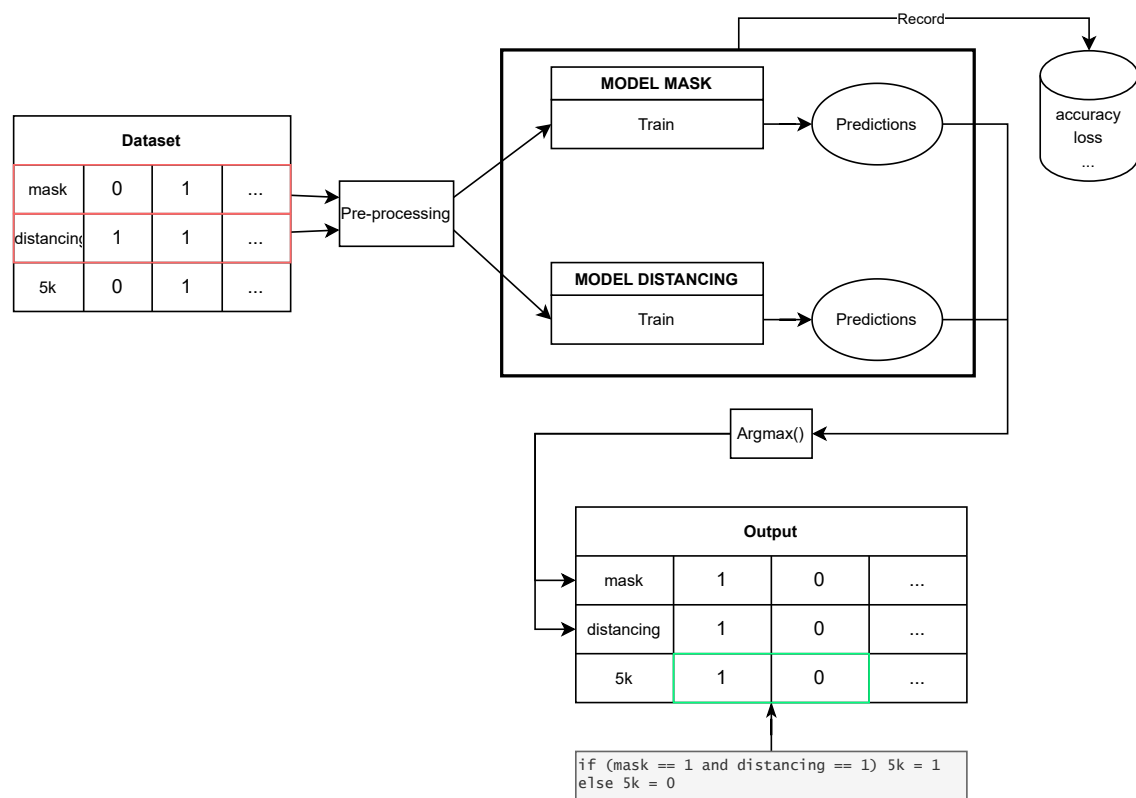
**However**, the metrics and results obtained from both approaches should not be considered as comparable values since I did change the training data, the hyper-parameters or other factors that maybe small but still affect the whole process. So, they should be used as a references and as a proof of my work, and should not be used as an appropriate validation.

### 3.1 First Approach

As the report is quite long now, for the approach overview, I decided to summarize it in form of pipeline diagrams. Some details or specific comments/discussions can be seen in my Notebooks and I hope you will spend time reading it.

Basically, in the first approach we will pre-process data by skipping all rows that have missing values. In the Pre-processing block, the training data and validation data will be converted into `tf.data.Dataset` datasets, which will then map into a function that load image and resize into specific dimensions, and be returned as new transformed datasets

Since the modeling part is quite similar for this approach and second one, so I will put the explanation and discussion in the Section 6.2.

**Figure 5:** First approach pipeline

### 3.2 Second Approach

In the second approach, some specific parts like Dataset creation, model structure or activation functions are similar to the first approach. Specifically, we witnessed too many missing values of the dataset. This approach represents the pseudo-labeling to compensate this problem in different stages.

First, we train the mask detection model only, then use it to predict the original datasets to fill out the missing mask values (mask labels). Afterwards, we use the pseudo-labeling logic to fill the missing distancing values as much as possible.

Second, we train the distancing detection model only, then use it to predict the updated datasets (with filled missing mask values) to fill out the missing distancing values (distancing labels). And similarly to step 1, we also use the pseudo-labeling logic to fill the 5k values, which is our final target labels. Note that the first and second step involve models that detect either mask or distancing, just like the first approach. The only difference is that we add a third model that predict the 5k label, which also finalize our pipeline.

Third, with the missing 5k values being filled, we can then use the 5k detection model to predict the final results (notably it only based on the 5k values, which is more appropriate for the training process).

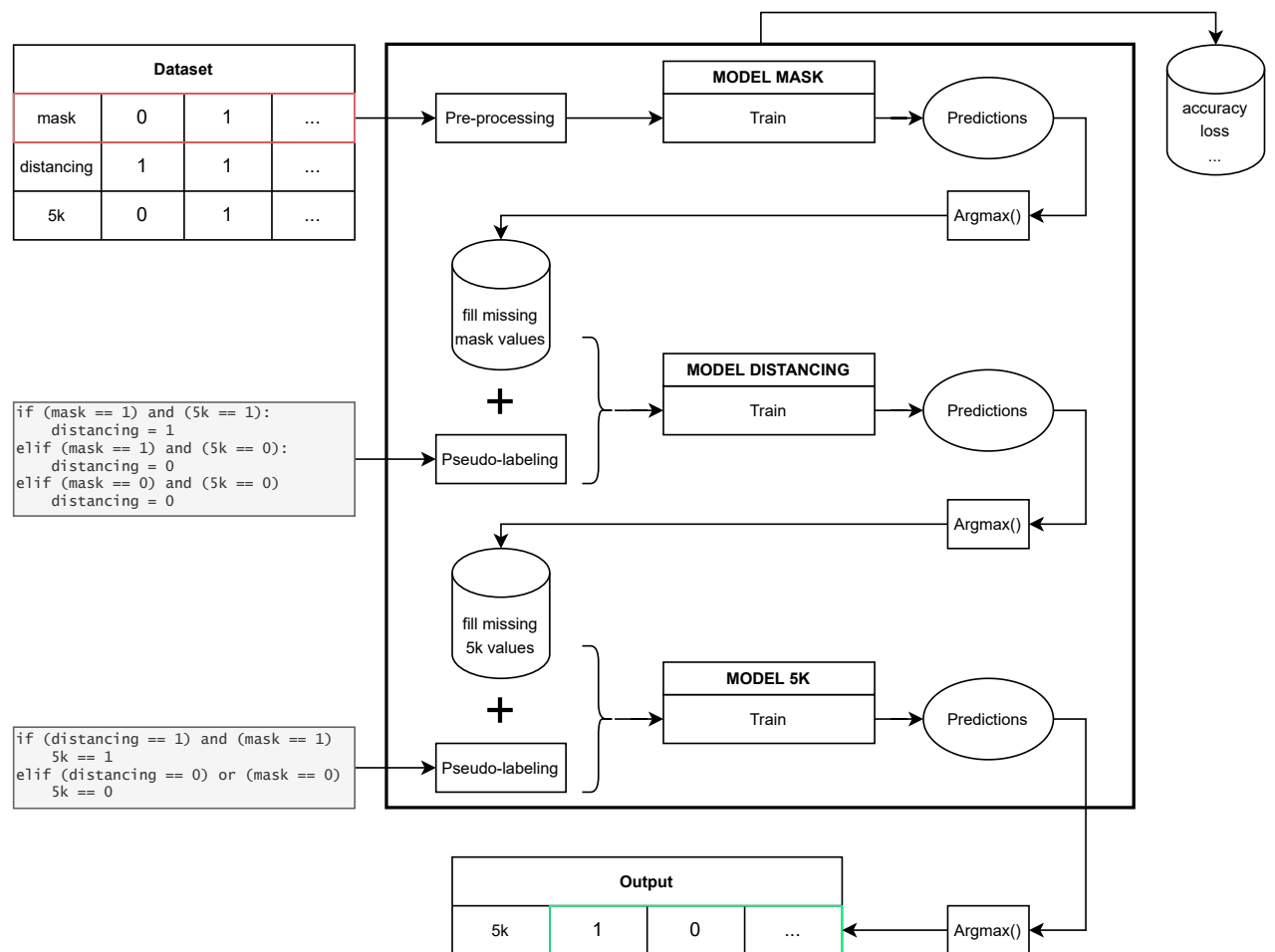
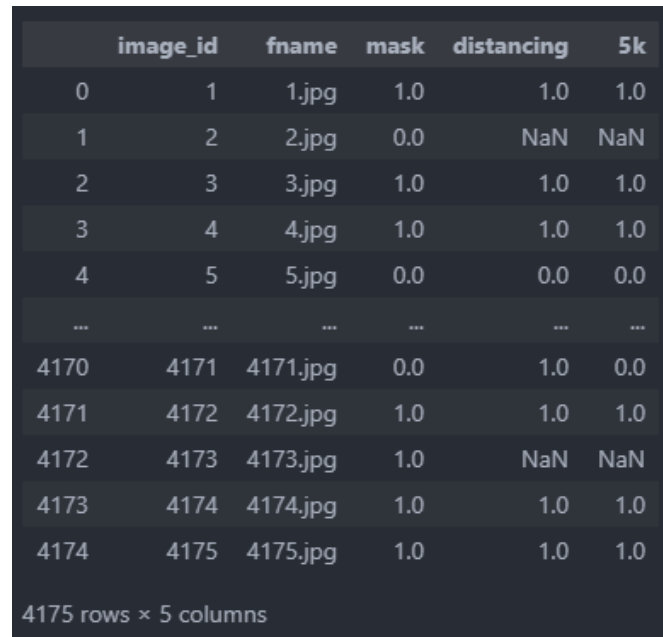


Figure 6: Second approach pipeline



## 4 Data Understanding

Here is a capture image of the dataframe. The dataset has 4175 records and 5 features:



	image_id	fname	mask	distancing	5k
0	1	1.jpg	1.0	1.0	1.0
1	2	2.jpg	0.0	NaN	NaN
2	3	3.jpg	1.0	1.0	1.0
3	4	4.jpg	1.0	1.0	1.0
4	5	5.jpg	0.0	0.0	0.0
...	...	...	...	...	...
4170	4171	4171.jpg	0.0	1.0	0.0
4171	4172	4172.jpg	1.0	1.0	1.0
4172	4173	4173.jpg	1.0	NaN	NaN
4173	4174	4174.jpg	1.0	1.0	1.0
4174	4175	4175.jpg	1.0	1.0	1.0

4175 rows x 5 columns

**Figure 7:** 5K Compliance dataset

where:

- **image\_id:** the id of the image
- **fname:** the name of the image, which also used as a path
- **mask:** the binary label indicate whether people are wearing mask
- **distancing:** the binary label indicate whether people standing far enough from each other
- **5k:** the binary label indicate whether people adhere both standards

And the image showing the number of missing values in multiple scenarios:

```
Num. missing mask 734  
Num. missing distancing 1273  
Num. missing 5k 2007  
Num. missing mask and distancing: 0  
Num. missing mask and 5k: 734  
Num. missing distancing and 5k: 1273  
Num. missing all three attributes: 0
```

**Figure 8:** Missing values from the 5K datasets

Apparently, the missing values are occurs as either missing one of three attribute, or a pair of attributes respectively (except for mask and distancing). None of row missing all three attributes.

To get the 5k value, we should have know the mask and distancing value first. Luckily, none of row miss these two variables. Therefore, we can fill the missing values with our own logic and partition the process into different stages as described in Section 3.2.

Notably, the number of missing value of mask labels is 734, while for distancing it is 1273, nearly double the figure of mask labels. Thus, it is possible for the distancing detection model to have a worse performance in term of accuracy. To be fair, if we choose the distancing detection model to be the first model in second approach's pipeline, it may affect the whole procedure since pseudo-labeling heavily relies on the predictions of the previous models.

## 5 Pre-processing

For the pre-processing step, I manage to work on two things: the `load_image` function and the `tf.data.Dataset` creation.

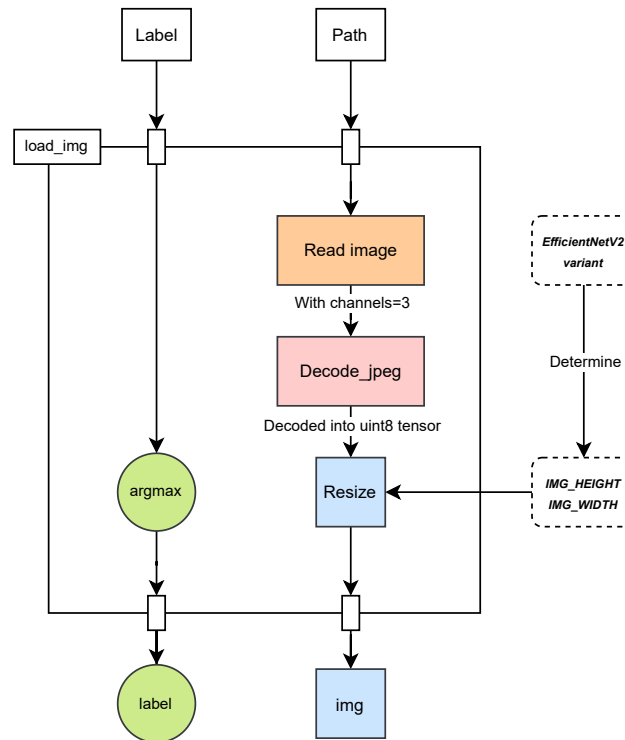


Figure 9: Load image function

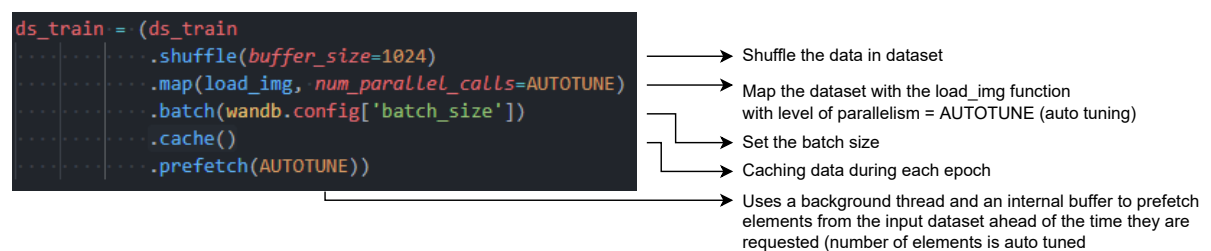


Figure 10: Tensorflow Dataset creation

## 6 Modeling

### 6.1 EfficientNetV2 variant selection

At first, I decided to use the EfficientNetV2-M-21k-ft1k (trained on ImageNet21k, fine-tuned on ImageNet1k) that will take the input size of 480x480 pixels. However, the parameter size is so large for the IDE (Google Colab/Kaggle) that I used, which also lead to larger FLOPS and the run-out-of-memory incident always occurs during the training process. I then managed to used the smaller version which is EfficientNetV2-S-21k-ft1 (Small), the implementation is doable but still face the memory problem in the middle of the process and 'black out'. Finally, I decided to choose EfficientNetV2-B3-21k-ft1, a scaled version with a purpose to compare the performance with the original EfficientNet. This model take the input size of 300x300 pixels, with a proper FLOPS that is appropriate for the GPU to handle.

For comparison, here is the table provide the characteristics of each models:

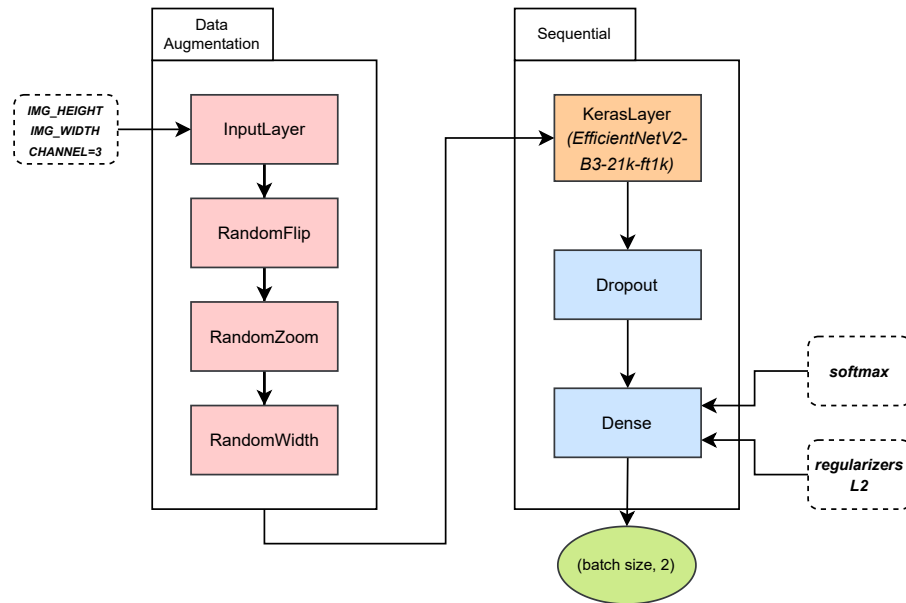
	EfficientNetV2-B3 (ImageNet 21k - FT 1K)	EfficientNetV2-M (ImageNet 21k - FT 1K)	EfficientNetV2-L (ImageNet 21k - FT 1K)
Params	<b>54.1M</b>	<b>21.5M</b>	<b>14.4M</b>
FLOPS	<b>24.7B</b>	<b>8.4B</b>	<b>3.0B</b>

Table 2: EfficientNetV2 variants comparison

### 6.2 Model Architecture

The model first use a sequential of data augmentation layers which include an Input-Layer of shape referenced from EfficientNetV2 model, followed by some image transformation layer like RandomFlip, RandomZoom and RandomWidth (or Random Rotation).

The augmentation block will then connect with the KerasLayer, which is the EfficientNetV2-B3-21k-ft1k, then followed by a Dropout layer, and a Dense layer with softmax activation function and L2 regularizers.



**Figure 11:** Model architecture

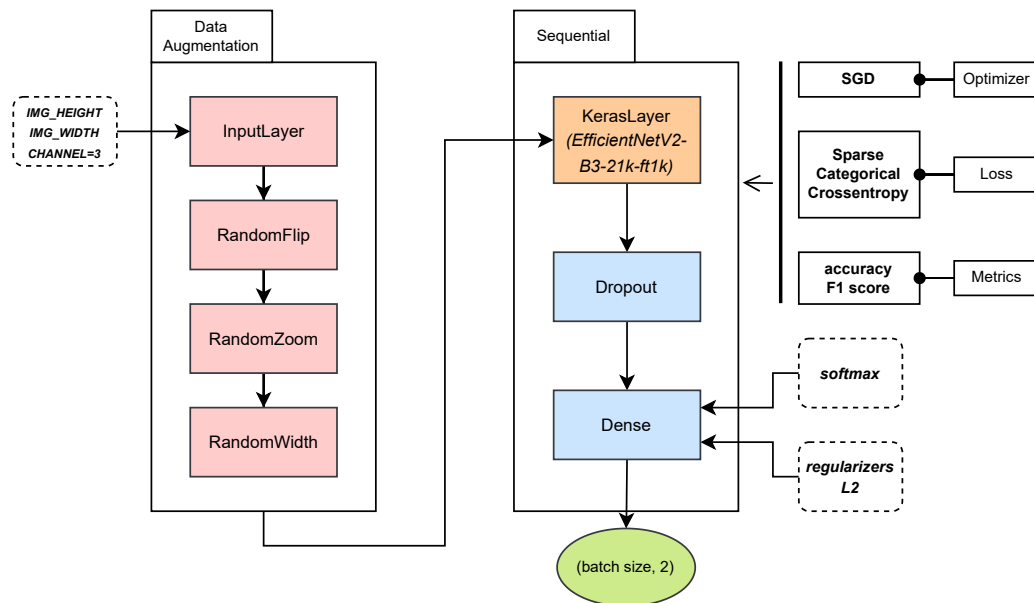
For the optimizer, I used **Stochastic Gradient Descent (SGD)** with momentum. When we set the momentum for SGD, it will use the new update rule for the weight as follows. The former rule:

$$w = w - learning\_rate * g \quad (5)$$

The new rule with momentum parameter:

$$\begin{aligned} velocity &= momentum * velocity - learning\_rate * g \\ w &= w + velocity \end{aligned}$$

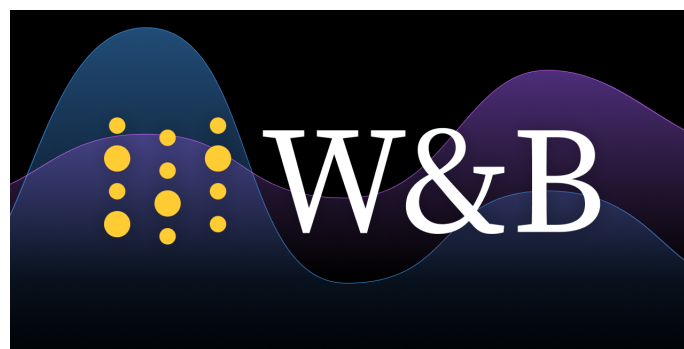
The loss function used is **SparseCategoricalCrossentropy** from the Keras library, which computes the cross-entropy loss between the labels and predictions. The metrics will be **accuracy** and **F1 scores** for all training and testing step.



**Figure 12:** Full Model architecture

## 7 Discussion

The results and experiment data will be put described on a separate platform since it support generating report with interactive graphs. The platform is **WandB** (stands for Weights Biases), a central dashboard to keep track of model's hyperparameters, system metrics, and predictions so we can compare models live, and share our findings. All of my notebooks also implement the WandB project to record the history of each epoch run and logging the data in form of line charts.



**Figure 13:** WandB - Weights Biases

The link to the discussion report: [Comparison of metrics](#).

## 8 Conclusion

As for myself, the problem and competition is very interesting and give me a lot of experience as well as practices. I have learned a lot from carry out the graduation assignment:

- capable of comprehending the research papers
- investigate how some specific models systematically work
- examine and convey the concept and theories based on the original work
- working on a bigger pipeline and procedure instead of single model
- integrate the notebook with visualize tool/dashboard like WandB
- implement the model architecture, train and predict
- get used with the image detection problem
- analyze the experimental data and metrics

The EfficientNet and EfficientNetV2 are indeed interesting architecture with impressive mechanism and methods integrated with them. As said, it would be a good experience or even a good choice using them in some specific problem where you need faster training time, decent accuracy while having less computational resources under constraints.

## 9 Acknowledgements

I deeply thank Quan Thanh Tho, Noro Hiroyoshi, Mouhamed Diop, Jules Ntaganda, Iradukunda Peter Yves, Cedrick Justin, other mentors and staffs of DIVE INTO CODE for offering this course and their support throughout the lessons.

## Terms

**ML** ..... Machine Learning

**AI** ..... Artificial Intelligence

**FLOPS** ..... Floating Point Operations Per Second

**5K** ..... An alliteration represent the COVID-19 prevention standard

**NAS** ..... Neural Architecture Search

**WB** ..... Weights and Biases

**FT** ..... Fine-tuned

**21k** ..... ImageNet 21k dataset

**1k** ..... ImageNet 1k dataset



## References

- [Xie+17] Saining Xie et al. *Aggregated Residual Transformations for Deep Neural Networks*. 2017. arXiv: [1611.05431](#) [[cs.CV](#)].
- [San+19] Mark Sandler et al. *MobileNetV2: Inverted Residuals and Linear Bottlenecks*. 2019. arXiv: [1801.04381](#) [[cs.CV](#)].
- [Tan+19] Mingxing Tan et al. *MnasNet: Platform-Aware Neural Architecture Search for Mobile*. 2019. arXiv: [1807.11626](#) [[cs.CV](#)].
- [GA20] Suyog Gupta and Berkin Akin. *Accelerator-aware Neural Network Design using AutoML*. 2020. arXiv: [2003.02838](#) [[eess.SP](#)].
- [TL20] Mingxing Tan and Quoc V. Le. *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. 2020. arXiv: [1905.11946](#) [[cs.LG](#)].
- [TL21] Mingxing Tan and Quoc V. Le. *EfficientNetV2: Smaller Models and Faster Training*. 2021. arXiv: [2104.00298](#) [[cs.CV](#)].