Title: Make EfficientNetV2 More Efficient

(Yuchen LI, Hanzhi Zhang, Bingyu Chen)

Abstract

Based on the severe situation of the global coronavirus epidemic, wearing masks plays an important role in blocking the virus and protecting others. The efficient mask recognition model can be used in public places around the world, such as train station face security check, company access control. Our group choose face mask detection as a research task to detect whether people wear masks. Convolutional Neural Network models (CNN), such as VGGNet, ResNet, contain tens of millions to hundreds of millions of parameters, which limits their training and optimization efficiency in practical applications. Therefore we further investigate on this problem to explore how to improve model efficiency without compromising accuracy. Real-World Masked Face Dataset would be utilized to train model, which was the world's largest real-world masked face dataset. EfficientnetV2 with both speed and accuracy using a multi-dimensional hybrid model scaling method will be selected as the baseline model. The exploration method is to use sparse training, channel pruning, model quantization, and low-resolution training based on pre-trained EfficientnetV2 model to reduce the parameters and flops of the model and keep its performance.

1. Introduction

The coronavirus is currently raging around the world, and has continued to endanger human health and names since 2019, and has seriously affected the global economy and situation. More than 400 million people worldwide have been infected with coronavirus and more than 5 million people have died from coronavirus as of February 2022 in worldometer. Masks break the chain of transmission of SARS-CoV-2 by reducing the infectivity of those with sub-clinical viral shedding while providing some protection to susceptible populations (Cheng et al., 2020). Public places require people to wear masks correctly before entering to prevent the spread of the virus. Manually monitoring whether individuals are wearing masks correctly and notifying them in public and crowded areas is a daunting task. The application of mask detection technology can effectively supervise people to wear masks correctly in crowded areas and reduce the workload of staff, especially in railway stations, Airports, shopping malls and other places with high traffic flow.

Most of the models used in mask recognition tasks are based on Convolutional Neural Network. For example, A.Chavada used Multi-Stage CNN Architecture in face mask detection and achieve 99.82% accuracy on NASNET-Mobile (Chavda et al., 2021). And K.Suresh applied Optimistic Convolutional Neural Network and achieve 98.7% accuracy (Suresh et al., 2021). But convolutional neural network models (CNN), such as VGGNet, ResNet, contain tens of millions to hundreds of millions of parameters and flops and cause huge computational and memory overhead. This limits their practical application for practical training, optimization, and memory efficiency (Hasanpour et al., 2016). Therefore, our group is looking for a more efficient CNN model with fewer parameters and flops to be used for mask recognition tasks. Mingxing Tan proposes EfficientNetV2 based on EfficientNet, and verifying that the balance between network depth, width and resolution can improve the performance of CNN (Tan & Le, 2021). And experiments prove that EfficientNet is in ImageNet outperforms other CNN network models, and the efficiency is also guaranteed under the condition of high accuracy. Efficient models in industry can reduce the workload of regularly updating data and ensure high accuracy.

L.Zhuang proposed a model compression method, Network Slimming, which can reduce the model size and runtime memory footprint, and reduce the number of computational operations without affecting the accuracy (Liu et al., 2017). Tensorflow Lite proposes half-precision quantization to convert model weights to 16-bit floating point values which results in model size being reduced by half. GPUs can perform local computations in this reduced-precision algorithm, achieving acceleration over traditional floating-point execution (Abadi et al., 2015). In the experiments, we use channel pruning, half-precision quantization, and low-resolution data to try to reduce the amount of parameters and flops of the pre-trained EfficientNetV2 model, and test the accuracy on the dataset.

Experiments based on Real-World Masked Face Dataset and pre-trained models show that semi-precision quantization and low-resolution data can improve the efficiency of the model from the amount of parameters or flops and ensure high accuracy, and the combination of sparsification and pruning improves efficiency while slightly reducing accuracy. Our contribution is to successfully improve the model efficiency in terms of parameters, flops and training time under the condition of high precision, accuracy and robustness.

2. Data set and task

Our group chose the experimental dataset Real-World Masked Face Dataset (RMFD) and Masked Face Detection Dataset (MFDD) from Github. The samples were crawled by python crawler tools from the web and RMFRD is the world's largest real-world masked face dataset. RMFRD contains 5,000 pictures of 525 people wearing masks and 90,000 images without masks. MFDD contains 24,771 images of masked faces. Each data is a real photo instead of a fake photo generated by simulation.These datasets can be freely downloaded and used by industry and academia, based on which models for masked faces can be trained. And using the data will not cause ethical issues (Wang et al., 2020).

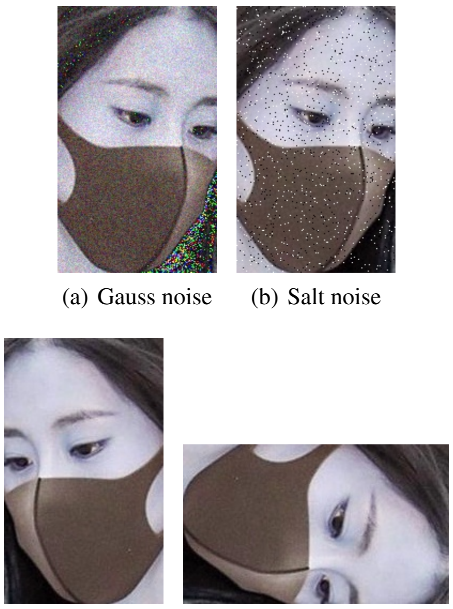
(a) Face (b) Masked

Figure 1. Data example: Face and masked face

The above example is the face and masked face picture of a character.For pre-processing, all pictures are restored in two folders according to the category, face and masked respectively.

The task we selected is to use the model to classify images into two categories: face and masked. We randomly selected 5000 masked images from RMFRD,15000 masked images from MFDD and 20000 unmasked face pictures from RMFR to generate a balanced label dataset with half unmasked and half masked. In the experiment, 80% of the data set is randomly selected as the training set, 10% as the validation set , and 10% as the test set. The model is trained on the training set to learn the image and its features, and the model is adjusted on the validation set. Finally, the model predicts the images on the test set, and the predicted results will be compared and evaluated with the originally marked tags. Precision and recall are the criteria for experimentally evaluating model performance.

In addition, we make a detection robustness dataset through data augmentation to further explore the performance of the model. This dataset consists of 10000 images without masks from RMFRD, and 10000 images masked face generated based on the original dataset by flipping, rotating and adding noise. To avoid image similarity, only one processed image is generated for each face. Figure 2 demonstrates the example of data augmentation methods on one face.


(a) Gauss noise (b) Salt noise

(c) Flipping (d) Rotation

Figure 2. Data example: augmentation

3. Methodology

The target model chosen for this experiment was EfficientNetV2, and various methods were chosen to optimize the model around this original model. The main objective of the optimization is to improve the training efficiency of the model while sacrificing as little recognition accuracy as possible. Channel pruning, semi-precision quantization, Low resolution training and Sparse training were selected to optimize our model. In this Section we describe in detail the technical details of these methods and our approach to evaluating the results of the experiments.

3.1. Model: EfficientNetV2:

In this experiment, we used EfficientNetV2, which is improved version of EfficientNetV1. The main idea of the V1 model is to improve the accuracy by increasing the network scale.(Tan & Le, 2019) There have been many researches on scaling network structure (depth, width, resolution) before, but they all face the same problem, that is, as the size of the model increases, the growth rate of accuracy decreases and eventually reaches a plateau value. Controlled experiments conducted by the team that came up with the model suggest that when one of the network structures (depth, width, resolution) increases, the others should also increase, and the model will perform better. So, they came up with an innovative composite method to scale the network uniformly.

For example, with 2^N times the sum resources, we can increase the network depth to α^N, the width to β^N, and the image resolution to γ^N. Where α, β, γ is the optimal constant found through grid search.

In addition to scaling the network, it was also important to select the efficient Baseline Network, so the team chose ConvNets, and their method works quite well in MobileNets (Sandler et al., 2018) and ResNet (Lin & Jegelka, 2018).

ConvNets can be expressed by the following formula:

$$\begin{aligned} & maxAccuracy(N(d,w,r)) \\ N(d,w,r) = & \bigodot_{i=1...s} \hat{F}_i^{d\cdot\hat{F}_i}(X_{\langle r\cdot\hat{H}_i, r\cdot\hat{W}_i, w\cdot\hat{C}_i\rangle}) \\ & Memory(N) \leq targetMemory \\ & FLOPS(N) \leq targetFlops \end{aligned} \tag{1}$$

F_i is the definition of network layers, L_i is the number of network layers. H_i and W_i are spatial dimensions (resolution), and C_i is channel dimension (width). The purpose of this research can be clearly seen from this formula, that is, to maximize the accuracy of the model by adjusting parameters under a certain target of memory and flops.

The compound scaling of the network structure is mainly reflected by the following formula:

$$\begin{aligned} depth : d = \alpha^{\phi} \\ width : w = \beta^{\phi} \\ resolution : \gamma^{\phi} \\ s.t. \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2 \\ \alpha \geq 1, \beta \geq 1, \gamma \geq 1 \end{aligned} \tag{2}$$

The significance of the compound coefficient ϕ is to scale the depth, width and resolution of the network uniformly. Due to the consideration of the network FLOPS, the model limits the $\alpha \cdot \beta^2 \cdot \gamma^2$ to 2, which means that the network FLOPS are approximately increased by 2^{ϕ}. With the definition of the composite parameters of the scaled-network, the EfficientNet chose MBConv (Tan & Le, 2019) as the main building block of the network, but the EfficientNetV2 replaces MBConv with Fused-MBConv for shallow networks in order to figure out the problem of deep separable convolution leading to slower training, which is illustrated in Figure 3.

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 3. EfficientNetV2-S network architecture (Tan & Le, 2021)

3.2. Pruning

Deep learning network models have a large number of redundant parameters from the convolutional layer to the fully connected layer, with a large number of neurons whose activation values converge to 0. Removing these neurons can exhibit the same expressive power of the model, a situation known as over-parameterisation, and the corresponding technique is known as model pruning. There are a number of approaches to pruning, roughly distinguished as:

Weight pruning, which is a fine-grained pruning, i.e. pruning of connections or neurons , which is the least granular pruning and is also the earlier pruning method.One of the well-known pruning method is(Hassibi & Stork, 1992). This method uses a second-order Taylor expansion to select parameters for pruning and treats pruning as a regular term to improve training and generalisation. The method is a fine-grained pruning with the advantage of retaining model accuracy. The disadvantage is that the method is unstructured pruning, only locally tuned, and relies on a dedicated library of sparse matrix operations and hardware devices.

Structure pruning.This is also known as filter-level and channel pruning. Only the number of filter banks and feature channels in the network is changed.A well-known method is (Molchanov et al., 2016). This method is channel-level pruning with a pruning criterion of selecting a subset of the set of channels and using the difference between the loss corresponding to the calculation of the subset and the loss corresponding to the original set as a measure of channel importance. The method iterates to remove the least important channels based on relevance pruning and fine-tunes them.

As fine-grained pruning relies on hardware and operation libraries1992second, the structured pruning was chosen for this experiment. The pruning strategy references (Liu et al., 2017). The method is a channel-level pruning approach, using a new importance measure that uses the scaling factor in batch normalisation as an importance factor, i.e. the smaller is, the less important the corresponding channel is to be pruned. For example, we remove the 70% channels with a lower scaling factor by being able to choose a percentile threshold of 70% (Figure 4). By doing this, we obtain a more compact network with fewer parameters and operation memory, and fewer computational operations.

3.3. Sparse training

EfficientNet is a large-scale network that requires a large amount of computational resources to train. One solution found in nature to improve the scaling of neural networks is to exploit sparsity:as the brain has more neurons, the fewer connections there are between neurons (Herculano-Houzel et al., 2010). Therefore, for neural networks, a similar strategy can be used to process the model.

Sparse training for this experiment referenced(Han et al., 2015). The core idea of this method is to prune the model through iteration (Figure 5). Unlike traditional training, this method is not learning the final value of the weights, but rather learning which connections are important. The second step is to remove the low weight connections. All connections with weights below a threshold are removed from the network. Also the model uses L1 regularization to penalize non-zero parameters when training connectivity,

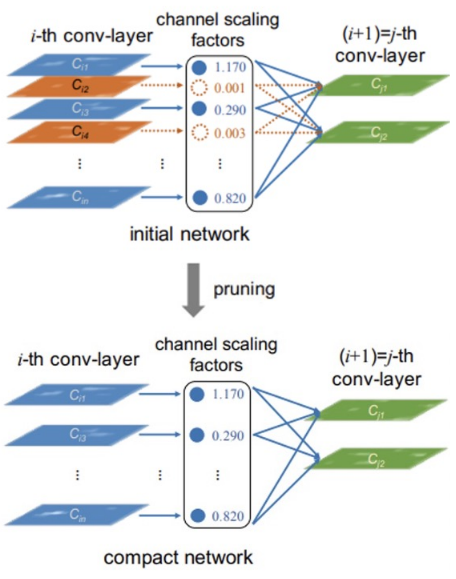


Figure 4. Channel-level pruning (Liu et al., 2017)

resulting in more parameters close to zero. to facilitate the pruning operation in the next step.

Inspired by that method, we also used sparse training to help with correct pruning afterwards. The difference is that instead of using an iterative approach, we perform a round of sparse training prior to pruning. In the sparse training phase, we train the gamma parameters in the BN layer and use them as sparse factors for each layer. After training, we rank the coefficients of each channel and then determine the threshold of gamma coefficients by a pre-set pruning ratio, and then prune the corresponding channels below the threshold.

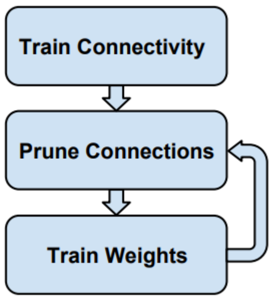


Figure 5. Iteration Training Pipeline (Herculano-Houzel et al., 2010)

3.4. Semi-precision quantization

Uniform quantification was used for this experiment. Uniform quantization is a very intuitive way of quantization and is relatively simple and common. The TensorFlow, Pytorch frameworks currently support quantization (Guo, 2018).

The bit budgets required to train the model can be calculated by the following equation (Jiang, 2020):

$$Bit\_budget = Number\_of\_parameters \cdot n + 2^n \cdot 32 \quad (3)$$

The equation shows that the storage space occupied by the model can be greatly reduced when n is reduced. Also because the model has smaller bit-width parameters, the cache and registers can store more temporary data, reducing the number of accessing RAM, and therefore the training speed will be faster.

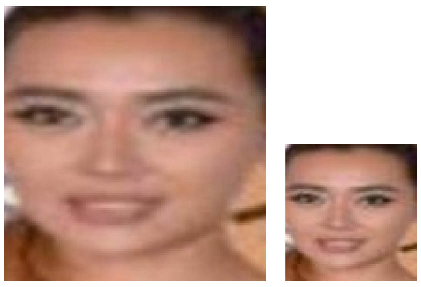
So we convert the precision value of the input data from 32 bits to 16 bits, and we also convert the precision value of parameters, namely the weights, from 32 bits to 16 bits.

3.5. Low resolution training

In the EfficientNet model, the image resolution is an adjustable hyper-parameter. In this way we can train larger networks using images of different sizes. (Touvron et al., 2020) shows that if the final layers of the network are fine-tuned after training using a larger image than the one initially trained. This can achieve significant accuracy improvements. This also means that we can start with a low resolution image to get better training efficiency and gradually increase the size of the image later on in the training process to increase the accuracy of the model. This allows us to reduce the training time while maintaining the accuracy of the model.

We were inspired by this to investigate the impact of using low resolution images on model training. Using smaller images during training allows us to train a given model faster, using less memory or training more epochs in the same amount of time. As the recognition task for our model is mask face recognition, which is a relatively simple task, we believe that low resolution images will not unduly sacrifice the final accuracy.

According to this assumption, we decided to reduce the width and length of the image to 1/2 of the original size, from 140*147 to 70*73 (Figure 6). Finally we trained the model at this resolution and observed its effect.



(a) 140*147 (b) 70*73

Figure 6. Low resolution transformation example

3.6. Evaluation methods

We evaluate the accuracy and efficiency of our models by four main metrics: parameters, FLOPs, accuracy and recall.

Parameters, as part of what a model learns from historical training data, are the key to machine learning algorithms . The process of machine learning is the process of learning better parameters for a model. In a neural network, the parameters are the weights. Therefore more parametric quantities indicate a higher complexity of the model.So this is the main metric for assessing the size of the model.

FLOPs are the most intuitive indicator of the computational intensity required to train a model. Therefore, we mainly look at the FLOPs to determine whether the efficiency of the model has been improved.

In the model performance evaluation phase, we used accuracy and recall. Accuracy reflects the predictive effectiveness of the model, but due to the possibility of sample imbalance, we also need to use recall to evaluate it. For example, when the positive sample size is particularly small, we need to obtain the proportion of correct predictions that are positive as a percentage of all actual positive predictions.

4. Experiments

4.1. Normal Training

We will mostly use the process and results of the model training as our baseline experiments. We fine-tune EfficientnetV2 as the main convolutional neural network in our experiments based on the official release of EfficientnetV2, which is a new series of convolutional neural networks with faster training speed and more outstanding parameter efficiency, and smaller size. We are willing to try it out for our task, face mask recognition. Faster training and higher accuracy will be the two main directions of our extension experiments. The experiments are conducted using a batch size of 8 for 50 epochs, and the initial learning rate is set to 0.01. The initial dropout rate is set to 0.2, and the default activation function is SiLu. The input size of training images is adjusted to 300, and the size of validation images is 384. The initialization weights are the officially published pre-training weights.

The dataset uses 20,000 face images and images of faces wearing masks and with labels as our original dataset. And 2000 of them are randomly selected as the dataset for testing the precision respectively. For face mask recognition, baseline's training results have extremely high precision and accuracy. This makes it difficult to improve its performance and has little practical meaning. Therefore, improving the network efficiency, compressing the model size, and reducing the training time are the ultimate extensions of our experiments.

4.2. Pruning

"Learning Efficient Convolutional Networks through Network Slimming" proposes an effective structural pruning method, i.e., a regularized channel pruning strategy. For pruning the complex weight structure again, the model size can be radically reduced.

The initial settings of the pruning experiments are the same as the baseline experiments except for the model weights. And we pre-set a tensor to calculate and record the number of parameters and Flops(multi-add) after reading in the weights. By comparing the number of parameters and Flops of baseline and pruning experiments, we can find the evidence of model slimming.

We choose the model that has been trained for 50 epochs in the baseline and saved as our original model structure to be pruned. Before pruning, we determine a global threshold and rank all absolute values of the weights. Then the maximum value of the pruning and the pruning ratio is determined. Then regular pruning is performed. For some weights larger than the threshold value, a masking operation is performed and retained. Finally, the number of channels is reduced layer by layer throughout the model, the model volume is compressed, and the amount of parameters is reduced. To explore the effect of pruning on the model structure, three pruning ratios of 30% ,50% and 70% are tried experimentally. We use the two models to re-run 50 epochs of training separately (other initial settings are the same as the baseline) and record data such as the number of parameters, Flops, etc. Finally, the precision is tested in a randomly selected dataset.

We found that the experiments under three different pruning ratios were successful as observed by the model volume and number of parameters alone. There is a noticeable reduction in the number of parameters and computation, as well as a reduction in training time. From the model performance point of view, only the model with 30% pruning ratio can be said to barely meet the requirements of the face mask recognition task, with 92.07% precision. The model with 70% pruning ratio can be said to be a completely failure for the binary classification model. Theoretically, the training time-precision marginal conversion rate of the 30% pruned model is 4.51 min, and the flops-precision marginal transformation rate is 117.41 M. The transformation rates of the remaining two models are 10.51 min, 165.5 M (50% pruned), 2.88 min, and 47.77 M (70% pruned).

4.3. Sparse training

To further improve the model accuracy, we need to improve the pruning measurement. That is, the model is trained sparsely before pruning. The purpose of sparse training is to make most of the learned network weights close to zero, because the relationship between weights and inputs in deep learning networks is multiplicative and additive, if most of the weights in the network model are zero or close to zero, removing this part has little impact on the final network output accuracy.

The specific idea is to constrain the gamma factor of the BN layer. In the previous pruning process, we have not been able to have a normative criterion for the determination of the global threshold. It can be said that we have only ensured that the overall number of channels reduced, and the parameters reduced to the level of the predefined

Model	Parameters	Flops	Training time	Test accuracy	Precision	Robustness	MRT(Flops -precision)
Baseline	20.18M	2.84G	5h9m54s	99.97%	99.82%	98.93%	
Pruned 30%	17.56M (-12.98%)	1.93G (32.04%)	4h34m45s (11.32%)	77.71%	92.07%	66.21%	117.41 M
Pruned 50%	6.02M (-70.18%)	887.08M (-68.76%)	3h5min22s (-40.13%)	79.92%	88.02%	75.33%	165.5 M
Pruned 70%	1.83M (-90.93%)	651.92M (-77.04%)	2h57min32s (-42.71%)	59.42%	54.02%	53.59%	47.77 M
Sparse training	6.02M (-70.18%)	887.08M (-68.76%)	3h2min14s (-47.57%)	95.84%	97.44%	92.29%	820.55M
Low-reso	20.18M	759.17M (-73.26%)	1h56m5s (-62.45%)	99.60%	96.59%	93.24%	657.32M

Table 1. Experiment results of different models

pruning rate. The sparse training is equivalent to providing guidance for pruning. By training the gamma parameter in the BN layer as the sparse factor for each layer, and after training, sorting the coefficients of all channels, and then determining the threshold of gamma coefficients through the pre-set pruning ratio, and then pruning the corresponding channels below the threshold.

The specific sparse training experiment uses sparsification (L1 regularization) of the gamma coefficients of the BN layer, and then the sparse gamma coefficients are used to evaluate the importance of the channels. The initialization settings of the experiment adjust the learning rate to 0.001 (a small learning rate helps the accuracy rebound), the scaling factor is 0.0001 by default, the batch size is adjusted to 10, and 300 epochs are trained. Then the channel pruning is performed according to the scaling factor size. The resulting model is applied to our previous network and dataset and retrained for 50 epochs.

It can be seen that sparse training can restore model accuracy while maintaining the same lightweight model structure. training time-precision marginal transformation rate is 53.36min, Flops-precision marginal transformation rate is 820.55M.Due to the effect of sparse training, the accuracy is substantially rebounded and the pruned model still has strong Identification power for images. The reduction in training time, along with the model file size, can demonstrate the effect of model thinning.

4.4. Low-resolution

By using semi-resolution images for training, we intend to shorten the training time and hope to keep and even improve the final accuracy. Different processing will be used for different requirements of the task, for example, facial expression recognition, fatigue driving, etc., will appear to increase the image resolution to improve the accuracy rate. But for simple binary classification training like face mask recognition, reducing the image resolution to improve efficiency is a better way of processing.

We use the official pre-training weights published by efficientnet and train fifty rounds. We also record the number of parameters and FLOPs (floating point operations). We use the same dataset from baseline and read in the paths image by image. Then reduce the length of each image and to half of the original. After saving the images, we also change the setting on image size for input cropping. we read in the pre-training weights (baseline), train 50 epochs with the learning rate is 0.01, and record the number of parameters and FLOPs (floating point operations). For the task of recognizing whether a face is wearing a mask or not, we use the randomly selected photos to further test the precision of the model. That is, the prediction is performed on a face-only or mask-only face dataset, and the precision is calculated.

In terms of maximizing efficiency in a single pursuit, reducing the image resolution training does reduce the training time. Using the same computer and allocating the same GPU space, we reduce the training time by 62.45% and can see a significant reduction in Floating point operations training time. FLOPs can not only measure the complexity of the model, but also demonstrate the change in the number of operations. Since, this experiment does not change the model structure, the number of parameters of the model does not change.

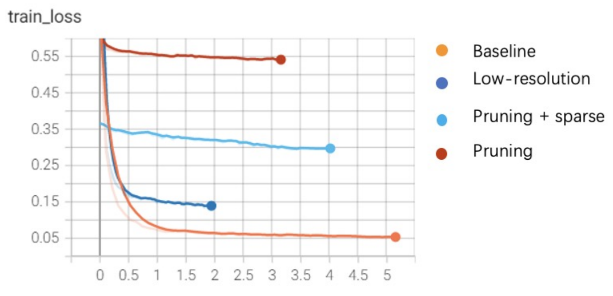
In terms of the accuracy of the experimental results, there is no excessive decrease in accuracy and precision. Compared with the experimental results of baseline, the accuracy decreases by 0.5%, the precision decreases by 3.2%. The actual marginal transform rate (training time - precision) is 60.31min, and the marginal transform rate for Flops-precision is 657M.

Obviously, we sacrifice a small amount of precision and accuracy, but greatly reduce our training time. The low-resolution processing of images is undoubtedly very successful. However, this method of improving efficiency may only be effective for the simple task of face mask recognition. The network does not need to learn too many features in that task, and again the ability of the model to perceive information is not important. However, from the business scenario, it is reasonable to choose the least costly feasible solution.

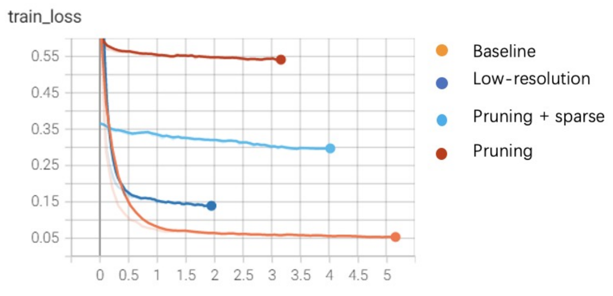
4.5. Quantization

The native torch framework is float32, we can borrow the idea of model quantization to turn it into float16, and pytorch itself defines half-precision tensor. suppose I train a model in the operation of the model directly to half-precision model, perhaps also reduce the efficiency. So we semiprecision the input image of the sparse pruned model with weights and access the GPU for training. The result was found that the change of precision was not high, and the decrease of training time was not obvious, but the use of video memory could be greatly reduced, and the size of weights was also reduced.With the same computing resources, the concurrency can be nearly doubled.

4.6. Result



(a) training loss on relative time



(b) training accuracy on relative time

Figure 7. All model training processes

Measuring the models in terms of model accuracy and training time, the low-resolution and sparsely trained pruned models processed the images best. According to the Pareto trade-off rule, the channel pruning method is better than other methods in terms of marginal variation and has a wide range of applications to other networks and tasks. combining the low-resolution method will improve the training efficiency even further, while the semi-precision method will not improve efficiency and accuracy, but will save memory significantly.

Translated with www.DeepL.com/Translator (free version)

5. Related work

Model efficiency on flops, parameter and training: DenseNet (Huang et al., 2016) and EfficientNet (Tan & Le, 2019) achieve high accuracy with low parameters on parameter efficiency. Several papers achieve to improve training speed instead of reducing parameters such as NFNets (Brock et al., 2021) and BoTNets (Srinivas et al., 2021). EfficientNetV2 (Tan & Le, 2021) proposes an improved incremental learning method that enables the model to be trained faster than previous networks in terms of the number of parameters. This paper aims to try to further improve the efficiency of the model based on EfficientNetV2. We reduce the model parameters, flops, training time and ensure the performance of the model in accuracy, recall and robustness.

Pruning and sparsitying: Pruning the unimportant connections (Han et al., 2015) with small weights can reduce the number of connections by a factor of 9 to 13 without loss of accuracy. Network Slimming (Liu et al., 2017) reduces flops without compromising accuracy However, reducing parameters and flops might reduce model performance on complex models. Pruning filters (Li et al., 2016) proposes to prune channels when training CNNs, and then fine-tune to restore accuracy. Sparsity (Changpinyo et al., 2017) randomly discards channel-wise connections before training, enabling a smoother channel pruning process and a comparatively small loss of accuracy. MicroNet (Li et al., 2021) reduces node connections and relies on improved layer nonlinearity to compensate for the reduced network depth, improving the performance of low-flops models. This paper uses network slimming to reduce parameters and flops, and implements sparsity to improve performance before pruning.

Quantization: Low-precision fixed-point arithmetic (Gupta et al., 2015) uses 16-bit fixed-point representation, which can significantly reduce memory and floating-point operations with little or no loss in classification accuracy. Float16 quantization (Abadi et al., 2015) converts weights to 16-bit floating point values, resulting in a two-fold reduction in model size and hardware acceleration. This paper uses float16 quantization in pytorch to achieve reduced model and training speed.

6. Conclusions

In this paper, we reduce the number of parameters, flops and training time of EfficientNetV2 through combination of network slimming, quantization and low resolution, and improve the performance of the reduced model through sparsity. Experiments show that we can improve model efficiency while ensuring model performance such as accuracy, recall and robustness. We hope this work provides methods to make CNN models more efficient on multiple vision tasks.

References

Abadi, Martín, Agarwal, Ashish, Barham, Paul, Brevdo, Eugene, Chen, Zhifeng, Citro, Craig, Corrado, Greg S., Davis, Andy, Dean, Jeffrey, Devin, Matthieu, Ghemawat, Sanjay, Goodfellow, Ian, Harp, Andrew, Irving, Geoffrey, Isard, Michael, Jia, Yangqing, Jozefowicz, Rafal, Kaiser, Lukasz, Kudlur, Manjunath, Levenberg, Josh, Mané, Dandelion, Monga, Rajat, Moore, Sherry, Murray, Derek, Olah, Chris, Schuster, Mike, Shlens, Jonathon, Steiner, Benoit, Sutskever, Ilya, Talwar, Kunal, Tucker, Paul, Vanhoucke, Vincent, Vasudevan, Vijay, Viégas, Fernanda, Vinyals, Oriol, Warden, Pete, Wattenberg, Martin, Wicke, Martin, Yu, Yuan, and Zheng, Xiaoqiang. TensorFlow Lite.: Post-training float16 quantization, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

Brock, Andrew, De, Soham, Smith, Samuel L., and Simonyan, Karen. High-performance large-scale image recognition without normalization. *CoRR*, abs/2102.06171, 2021. URL https://arxiv.org/abs/2102.06171.

Changpinyo, Soravit, Sandler, Mark, and Zhmoginov, Andrey. The power of sparsity in convolutional neural networks. *CoRR*, abs/1702.06257, 2017. URL http://arxiv.org/abs/1702.06257.

Chavda, Amit, Dsouza, Jason, Badgujar, Sumeet, and Damani, Ankit. Multi-stage cnn architecture for face mask detection. In *2021 6th International Conference for Convergence in Technology (I2CT)*, pp. 1–8, 2021. doi: 10.1109/I2CT51068.2021.9418207.

Cheng, Vincent Chi-Chung, Wong, Shuk-Ching, Chuang, Vivien Wai-Man, So, Simon Yung-Chun, Chen, Jonathan Hon-Kwan, Sridhar, Siddharth, To, Kelvin Kai-Wang, Chan, Jasper Fuk-Woo, Hung, Ivan Fan-Ngai, Ho, Pak-Leung, and Yuen, Kwok-Yung. The role of community-wide wearing of face mask for control of coronavirus disease 2019 (covid-19) epidemic due to sars-cov-2. *Journal of Infection*, 81(1):107–114, 2020. ISSN 0163-4453. doi: https://doi.org/10.1016/j.jinf.2020.04.024. URL https://www.sciencedirect.com/science/article/pii/S0163445320302358.

Guo, Yunhui. A survey on methods and theories of quantized neural networks. *arXiv preprint arXiv:1808.04752*, 2018.

Gupta, Suyog, Agrawal, Ankur, Gopalakrishnan, Kailash, and Narayanan, Pritish. Deep learning with limited numerical precision. *CoRR*, abs/1502.02551, 2015. URL http://arxiv.org/abs/1502.02551.

Han, Song, Pool, Jeff, Tran, John, and Dally, William J. Learning both weights and connections for efficient neural networks. *CoRR*, abs/1506.02626, 2015. URL http://arxiv.org/abs/1506.02626.

Hasanpour, Seyyed Hossein, Rouhani, Mohammad, Fayyaz, Mohsen, and Sabokrou, Mohammad. Lets keep it simple: using simple architectures to outperform deeper and more complex architectures. 08 2016.

Hassibi, Babak and Stork, David. Second order derivatives for network pruning: Optimal brain surgeon. *Advances in neural information processing systems*, 5, 1992.

Herculano-Houzel, Suzana, Mota, Bruno, Wong, Peiyan, and Kaas, Jon H. Connectivity-driven white matter scaling and folding in primate cerebral cortex. *Proceedings of the National Academy of Sciences*, 107(44):19008–19013, 2010.

Huang, Gao, Liu, Zhuang, and Weinberger, Kilian Q. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016. URL http://arxiv.org/abs/1608.06993.

Jiang, Mingchao. *Influence Based Bit-Quantization for Machine Learning: Cost Quality Tradeoffs*. PhD thesis, Rice University, 2020.

Li, Hao, Kadav, Asim, Durdanovic, Igor, Samet, Hanan, and Graf, Hans Peter. Pruning filters for efficient convnets. *CoRR*, abs/1608.08710, 2016. URL http://arxiv.org/abs/1608.08710.

Li, Yunsheng, Chen, Yinpeng, Dai, Xiyang, Chen, Dongdong, Liu, Mengchen, Yuan, Lu, Liu, Zicheng, Zhang, Lei, and Vasconcelos, Nuno. Micronet: Improving image recognition with extremely low flops. *CoRR*, abs/2108.05894, 2021. URL https://arxiv.org/abs/2108.05894.

Lin, Hongzhou and Jegelka, Stefanie. Resnet with one-neuron hidden layers is a universal approximator. *Advances in neural information processing systems*, 31, 2018.

Liu, Zhuang, Li, Jianguo, Shen, Zhiqiang, Huang, Gao, Yan, Shoumeng, and Zhang, Changshui. Learning efficient convolutional networks through network slimming. *CoRR*, abs/1708.06519, 2017. URL http://arxiv.org/abs/1708.06519.

Molchanov, Pavlo, Tyree, Stephen, Karras, Tero, Aila, Timo, and Kautz, Jan. Pruning convolutional neural networks for resource efficient transfer learning. *arXiv preprint arXiv:1611.06440*, 3, 2016.

Sandler, Mark, Howard, Andrew, Zhu, Menglong, Zhmoginov, Andrey, and Chen, Liang-Chieh. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.

Srinivas, Aravind, Lin, Tsung-Yi, Parmar, Niki, Shlens, Jonathon, Abbeel, Pieter, and Vaswani, Ashish. Bottleneck transformers for visual recognition. *CoRR*, abs/2101.11605, 2021. URL https://arxiv.org/abs/2101.11605.

Suresh, K, Palangappa, MB, and Bhuvan, S. Face mask detection by using optimistic convolutional neural network. In *2021 6th International Conference on Inventive Computation Technologies (ICICT)*, pp. 1084–1089, 2021. doi: 10.1109/ICICT50816.2021.9358653.

Tan, Mingxing and Le, Quoc V. Efficientnet: Rethinking model scaling for convolutional neural networks. *CoRR*, abs/1905.11946, 2019. URL http://arxiv.org/abs/1905.11946.

Tan, Mingxing and Le, Quoc V. Efficientnetv2: Smaller models and faster training. 2021. doi: 10.48550/ARXIV.2104.00298. URL https://arxiv.org/abs/2104.00298.

Touvron, Hugo, Vedaldi, Andrea, Douze, Matthijs, and Jégou, Hervé. Fixing the train-test resolution discrepancy: Fixefficientnet. *arXiv preprint arXiv:2003.08237*, 2020.

Wang, Zhongyuan, Wang, Guangcheng, Huang, Baojin, Xiong, Zhangyang, Hong, Qi, Wu, Hao, Yi, Peng, Jiang, Kui, Wang, Nanxi, Pei, Yingjiao, Chen, Heling, Miao, Yu, Huang, Zhibing, and Liang, Jinbi. Masked face recognition dataset and application. *CoRR*, abs/2003.09093, 2020. URL https://arxiv.org/abs/2003.09093.