Bonus Section: Efficient Net

Group D:
Aniket Chandak
Ashraf Saber
David Bui
Deanne Kshipra Charan
Hardik Kumar
Siddartha Thentu

Efficient Net

Scope

This paper studies the impact of model scaling, adjusting depth, width and resolution on performance. The paper proposes a scaling method that scales all dimensions uniformly. This scaling uses a highly effective compound coefficient. The paper studies the effectiveness of this method on MobileNets and ResNets [1].

Efficient Nets use neural network architecture search in order to design the baseline network then scale it up. Scaling up is essential as it creates a family of models that have a higher accuracy than the known Convolutional Neural Networks (ConvNets).

The EfficientNets presented in the paper reach very good results when applied to major datasets, they also deliver good results in transfer learning. The EfficientNet families reached state of the art results on CIFAR-100 WITH 91.7% accuracy. It was tested on Flowers data set and reached results of 98.8%. Three other datasets were

One of the Efficient Nets, *Efficient Net B7* has the state of the art results of 84.4% on ImageNet. It is 6.1x faster in regards to inference and it is 8.4x smaller than the best ConvNet.

ConvNet Scaling

One of the common methods used to improve the accuracy of convolutional networks is scaling up the networks. However, there is no standard method followed in scaling. Usually ConvNets are scaled up by increasing one of the three dimensions: depth, width and image resolution. The EfficientNet Paper studies scaling ConvNets efficiently for highest results. The authors argue that *compound scaling:* which is scaling all dimensions by a constant ratio yields the best results. Image resolution, width and depth are all scaled with a uniform value.

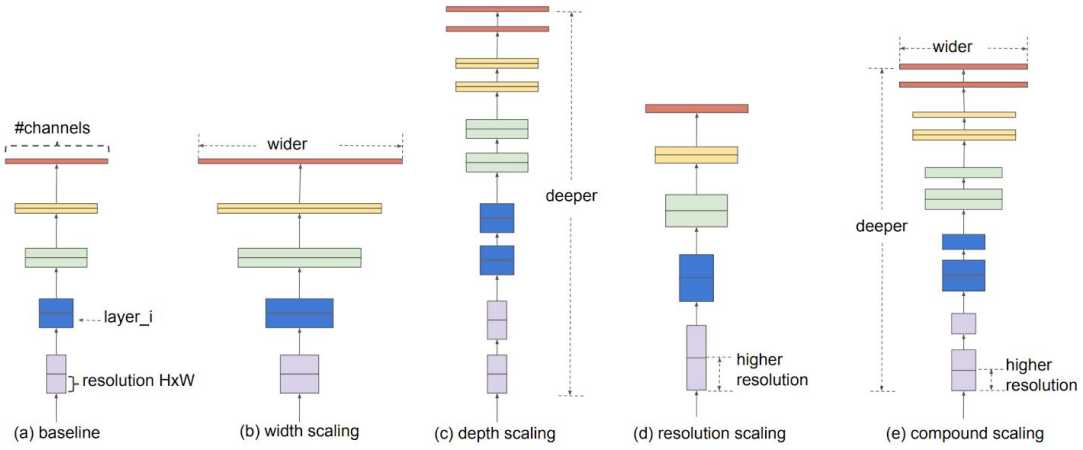

Figure 1. Model Scaling

Goal

The goal of model scaling is to maximize the accuracy. This is done by altering the length, width and resolution without changing the layer architecture in the baseline network. Scaling enables a model to operate with lower resources. It is critical that scaling occurs with a uniform value for all parameters.

$$\max_{d,w,r} \quad Accuracy\big(\mathcal{N}(d,w,r)\big)$$

$$s.t. \quad \mathcal{N}(d,w,r) = \bigodot_{i=1\ldots s} \hat{\mathcal{F}}_i^{d\cdot\hat{L}_i}\left(X_{\langle r\cdot\hat{H}_i, r\cdot\hat{W}_i, w\cdot\hat{C}_i\rangle}\right)$$

$$\text{Memory}(\mathcal{N}) \leq \text{target\_memory}$$

$$\text{FLOPS}(\mathcal{N}) \leq \text{target\_flops}$$

where w, d, r are coefficients for scaling network width, depth, and resolution; $\hat{\mathcal{F}}_i, \hat{L}_i, \hat{H}_i, \hat{W}_i, \hat{C}_i$ are predefined parameters in baseline network

Figure 2. Goal of Scaling

Architecture

The baseline network is developed and is called EfficientNet. It leverages multi-objective neural architecture search. Its goal is to optimize accuracy and FLOPs results. The output baseline network is named EfficientNet-B0. Compound scaling is applied to it in two steps. The first step includes fixing $\Phi = 1$ and doing a small grid search of the three parameters (α, β and γ). In the second step, α, β and γ are fixed and different values for Φ are used to scale up the network. Doing so, we get EfficientNet-B1 - B7 [1].

Our Attempt

To perform transfer learning on CIFAR100 dataset, we selected the EfficientNet B0 model, which is relatively smaller, requires less time to train and demands less computational resources than other models.

CIFAR100 data has 60,000 images of size 32x32 from 100 classes with 600 images for each class. For our experiment, we took the first 50,000 images for training and the rest 10,000 images for validation.

Furthermore, since EfficientNet is trained on ImageNet data with 1000 classes, the ultimate softmax classifier has 1000 classes. Therefore, we performed transfer learning on the CIFAR100 dataset which has only 100 classes, we replaced the last layer of B0 model with a softmax layer with 100 output classes.

Our transfer learning model's architecture is presented below.

Layer (type)	Output Shape	Param #
efficientnet-b0 (Model)	(None, 5, 5, 1280)	4049564
gap (GlobalMaxPooling2D)	(None, 1280)	0
dropout_out (Dropout)	(None, 1280)	0
fc_out (Dense)	(None, 100)	128100

To perform transfer learning, we first download the pre-trained weights for EfficientNetB0 from https://github.com/tensorflow/tpu/tree/master/models/official/efficientnet and load them into our model. There are two approaches to perform transfer learning, we can either freeze the initial layer and fine tune the final layers or fine tune all the layers. Since it was computationally feasible (and to experiment with different results), we chose to freeze the earlier layers and fine tune the last 4 layers of the network. Also, since the input layer of EfficientNet was designed to take the input samples of size 224x224, the CIFAR100 data samples were first upsampled from 32x32 to 224x224 before performing the training.

The experimental parameters are discussed below:

The learning rate was set to 0.0064 with the decay set to 1e-6. We chose the optimizer RMSProp which was the same optimizer used in the original paper. To prevent any overfitting, the training data was augmented by normalizing, horizontal flip, rotating and zooming. The batch size was set to 128 and the model was run for 20 epochs.

The experimental results are discussed below.

Results

```
Epoch 8/20
232/232 [==============================] - 371s 2s/step - loss: 3.1316 - acc: 0.3506 - val_loss: 4.8667 - val_acc: 0.0835
Epoch 9/20
232/232 [==============================] - 368s 2s/step - loss: 3.1160 - acc: 0.3504 - val_loss: 4.8147 - val_acc: 0.0937
Epoch 10/20
232/232 [==============================] - 367s 2s/step - loss: 3.0700 - acc: 0.3563 - val_loss: 5.0772 - val_acc: 0.0801
Epoch 11/20
232/232 [==============================] - 369s 2s/step - loss: 3.0588 - acc: 0.3564 - val_loss: 4.7762 - val_acc: 0.0973
Epoch 12/20
```

As we can see in the results, although the training accuracies were reaching 40%, the validation accuracies were very poor. This clearly tell us that the model was overfitting.

Since, the validation accuracies are really poor when training only the final layers, for future attempts, we are planning to fine tune the entire model.

References

[1] Tan, Mingxing, and Quoc V. Le. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks." *arXiv preprint arXiv:1905.11946* (2019).