Supplementary Materials

Diabetic Foot Ulcer Ischemia and Infection Recognition using EfficientNet Deep Learning Models

Ziyang Liu, Josvin John, Emmanuel Agu

I. MATERIALS AND METHODS

A. Inverted Residual Block

The residual block module (Figure 1a) was first proposed as part of the ResNet [1] architecture, while the inverted residual block (Figure 1b) was first introduced in the MobileNet architecture [2] [3]. Shortcuts can improve the gradient's ability to propagate across multiplier layers. The inverted residual block is more memory efficient and works slightly better than the residual block [3]. MobileNet applies depthwise separable convolutions inside the residual block with depthwise convolution being applied before pointwise convolution. This architecture decreases the number of trainable parameters significantly. The light-colored blocks in Figure 1a and Figure 1b indicate the beginning of the next block. The thickness of each block indicates the number of channels it has. The classic residual connections connect layers with a high number of channels and they connect the beginning and end of a convolutional block with a skip connection. However, the inverted residuals connect the bottlenecks, which are the leftmost layer and rightmost layer in the inverted residual block in Figure 1b. These two layers do not contain non-linearities. There exists a natural separation between the linear bottleneck layers and the layer transformation, which is the nonlinear function that maps the input to the output. Here, the bottleneck shows the capacity of the network at each layer, while the layer transformation shows the expressiveness.

Non-linear activation functions in neural networks such as the ReLU function discard values that are below 0, which ultimately is a loss of information. However, this can be handled by increasing the number of channels to increase the capacity of the network. The inverted residual blocks do the opposite by squeezing the layers that are linked by the skip connections. The linear bottleneck allows the last convolution of a residual block to have a linear output before it is added to the initial activation.

$F(x)$ is a bottleneck block operator shown in Figure 1b. It can be expressed as a composition of three operators $F(x) = [A \circ N \circ B]x$. Here A is a linear transformation $A : R^{s \times s \times k} \rightarrow R^{s \times s \times n}$. N is a non-linear per-channel transformation $N : R^{s \times s \times n} \rightarrow R^{s' \times s' \times n}$ and here $N = ReLU6 \circ dwise \circ ReLU6$. B is a linear transformation to the output domain $B : R^{s' \times s' \times n} \rightarrow R^{s' \times s' \times k'}$. The inner tensor I can be represented as concatenation of t tensors with size n/t so that the function can be represented as $F(x) = \sum_{i=1}^{t} (A_i \circ N \circ B_i)(x)$.

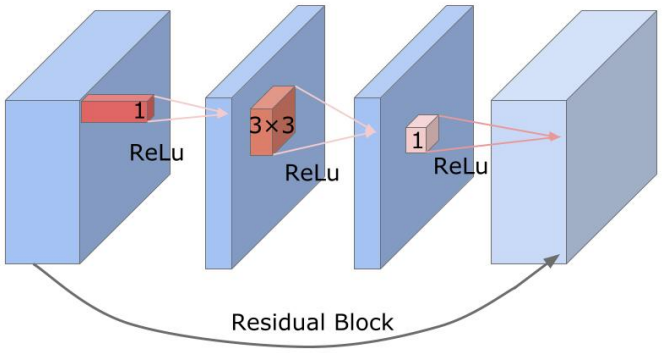
B. SE Block and Swish Activation in EfficientNet

a) Squeeze and Excitation Block: Squeeze-and-excitation (SE) block is used extensively throughout EfficientNet. A Squeeze-and-Excitation (SE) block [4] is a computational unit that works as a transformation F_{tr} that maps an input $X \in R^{H' \times W' \times C'}$ to feature maps $U \in R^{H \times W \times C}$. The transformation F_{tr} is a convolutional operator and a corresponding SE block is constructed to perform feature recalibration for it. The features U are input into a squeeze operation and it produces a channel descriptor with the shape $1 \times 1 \times C$ by using average pooling to squeeze each channel to a single numeric value. This descriptor provides the global distribution of channel-wise feature responses so that all network layers can use the information from the global receptive field of the network. The next step is the excitation operation that uses a simple self-gating mechanism. It takes the descriptor as input and outputs the modulation weights with the shape $1 \times 1 \times C$ for each channel. These weights are applied to the feature maps U to generate the SE block's output, which can be utilized by subsequent layers of the network. Traditional neural networks weight each of the channels of the output feature maps equally. However, SE networks add a single parameter to each channel in order to weight them differently. The SE network can learn the weight of each feature map adaptively, which scales each channel according to its importance. A residual network module that uses the SE block is shown in Figure 1c.

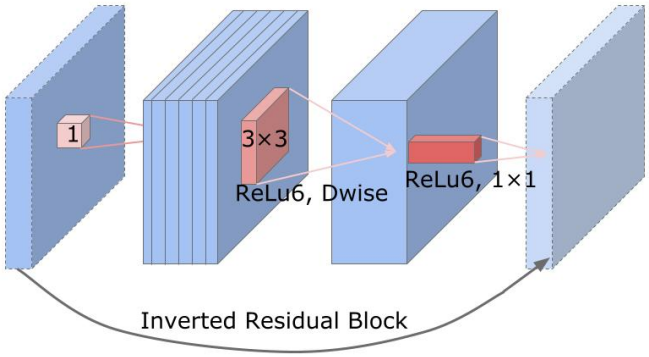
b) Swish Activation: Swish Activation is a more recent activation function and is used in EfficientNet. The specific activation function selected can affect the training of the deep neural network and its performance. The Rectified Linear Unit (ReLU), $f(x) = max(0,x)$, is one of the most widely used in many deep neural networks. However, ReLU has the problem that its derivative is 0 for half of the values of the input, which results in their parameters never being updated during the Gradient Descent algorithm. Consequently, the Google Brain Team proposed a new activation function called Swish [5], which is simply $f(x) = x \cdot sigmoid(x)$ as shown in Figure 1d. Unlike the ReLU function, Swish is a smooth function that does not change direction suddenly near x=0 and does not remain 0 or move in one direction. Google Brain Team's experiments showed that Swish worked better than ReLU on deeper models with different datasets. For example, Mobile NASNetA [6] top-1 classification accuracy on ImageNet increased by 0.9% and Inception-ResNet-v2 [7] increased by 0.6% after replacing ReLUs with Swish.

REFERENCES

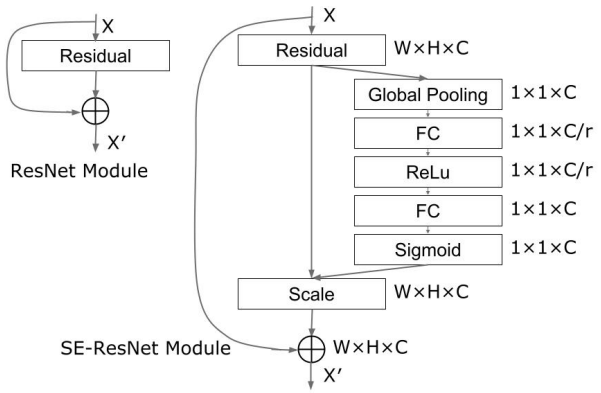
[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision*



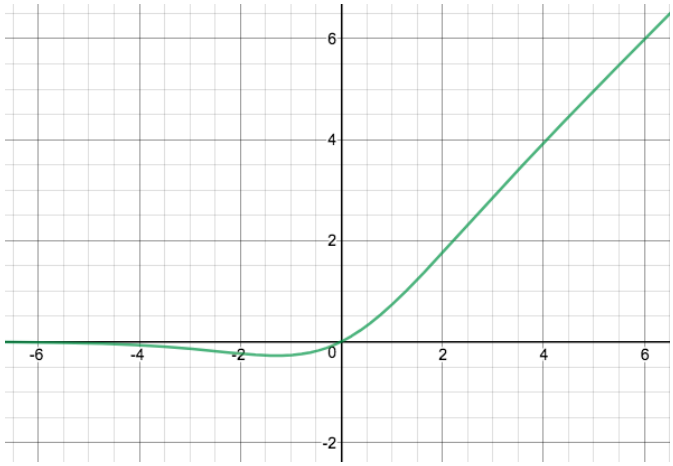
(a) Residual Block



(b) Inverted Residual Block



(c) SE-ResNet Module



(d) Swish Activation

Fig. 1. Residual Block, Inverted Residual Block, SE-ResNet Module and Swish Activation

and pattern recognition, 2016, pp. 770–778.

[2] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[3] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[4] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.

[5] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *arXiv preprint arXiv:1710.05941*, 2017.

[6] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8697–8710.

[7] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Thirty-first AAAI conference on artificial intelligence*, 2017.