Artificial Neural Networks

3rd Assignment

Saeid Cheshmi - 98222027

Exercise 1

Common ConvNet architectures are developed for fixed resource cost. We can scale up these networks to achieve more accuracy. Depth-wise scaling can be done by adding more layers to baseline model. For example, we can scale ResNet-18 to ResNet-200 by adding more layers. But almost there is no conventional way for scaling up, some models are scaled depth-wise, some are scaled widthwise. Some models use higher resolution inputs. Authors of **EfficientNet** provided a much more moral way for scaling up ConvNets to get better accuracy.

EfficientNet uses compound coefficient to scale up models in effective manner. Compound scaling uniformly each dimension with a certain fixed set of compound coefficients. Using this method, authors of **EfficientNet** developed seven models which got state-of-the-art accuracy on some benchmarking datasets at the time.

Compound scaling method

For developing the method of compound scaling, authors studied the impacts of each scaling technique has on model's performance. They found out each single dimension scaling improve the model performance. After that they proposed a method to scale all dimensions considering the variable available resources.

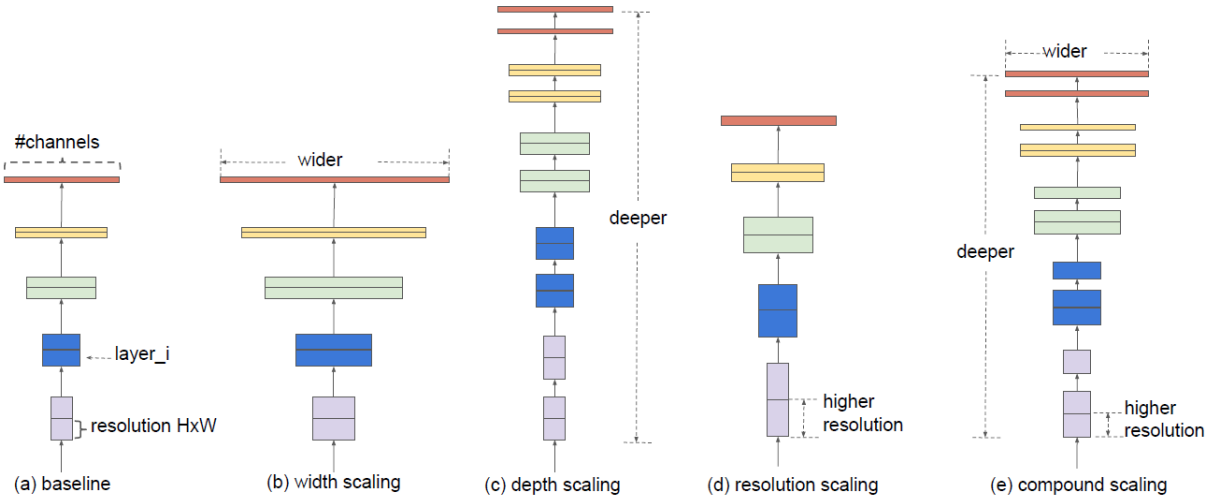

Figure 1. Model Scaling

Compound scaling method is based on balancing all dimensions by scaling with constant ratio. Below equations show how it is achieved:

$$\text{Depth } d = \alpha^{\emptyset}, \text{Width } w = \beta^{\emptyset}, \text{Resolution } r = \gamma^{\emptyset},$$

$$\text{such that } \alpha . \beta^2 . \gamma^2 \approx 2$$

$$\alpha \geq 1, \beta \geq 1, \gamma \geq 1$$

The value of α, β, γ are determined by grid search algorithm. …. Is a user-defined determines the increases in computational resources to the network.

When … is set to 1, and assuming twice more resources are available, doing a small grid search gives the value of α = 1.2, β = 1.1 and γ = 1.15. this is EfficientNet-B0 model. Then, the values of …. Are fixed and the baseline network is scaled up.

This shows that if input becomes bigger, the network needs more layers to increase receptive field and more channels to capture more patterns. Compound scaling method also can be used for scaling up **ResNet** and **MobileNet**.

Architecture

The two main contributions of this paper are first designing a baseline model like **EfficientNet-B0**, then providing compound scaling method for scaling the model.

The main building block of baseline model is **MBConv** plus squeeze-and-excitation optimization. **MBConv** is similar to the inverted residual blocks used in **MobileNet** v2. These form a shortcut connection between the beginning and end of a convolutional block. The shortcut connections connect the narrow layers whilst the wider layers are present between the skip connections. This structure helps in decreasing the overall number of operations required as well as the model size.

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 2. Architecture of baseline model

Exercise 2

Inception-ResNet

The **Inception-ResNet** network is a hybrid network inspired both by inception and the performance of **ResNet**. This hybrid has two versions, **Inception-ResNet v1** and v2. Although their working principles are the same, **Inception-ResNet v2** is more accurate, but has a higher computational cost than the previous **Inception-ResNet** v1 network. The **Inception-ResNet** like the Inception v4 incorporates the use of 3 different stem modules and reduction blocks. However, as this network is a hybrid between the **Inception v4** and **ResNet**, the key functionality of the **Inception-ResNet** is that the output of the inception module is added to the input.

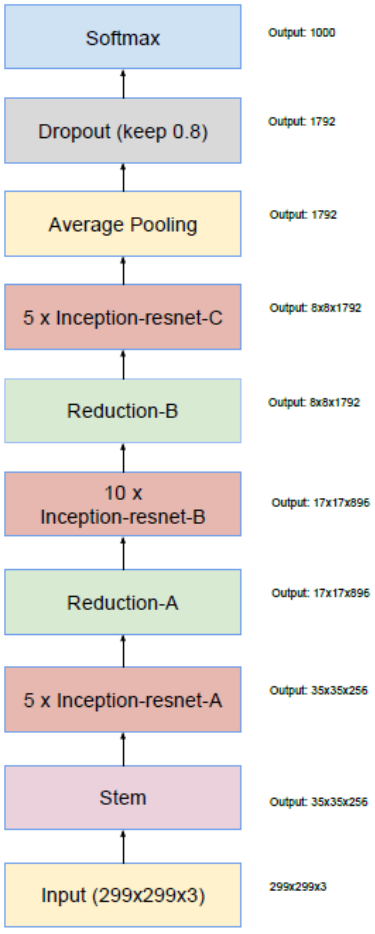

Figure 3. Schema for Inception-ResNet

Res-Next

ResNeXt inherited from **ResNet**, **VGG**, and **Inception**, **ResNeXt** includes shortcuts from the previous block to next block, stacking layers and adapting split-transform-merge strategy.

ResNet: Introducing a shortcut from the previous layer to the next layer

VGG: Leveraging repeating layers to build a deep architecture model

Inception: Following split-transform-merge practice to split the input to multiple blocks and merging blocks later on.

ResNeXt: The principle is stacking the same topology blocks. Within the residual block, hyper-parameters (width and filter sizes) are shared.

Cardinality is introduced in this paper. It means the size of the set of transformations. As you can see in below figure, the architecture includes 32 same topology blocks, so the value of cardinality is 32. Because of using the same topology, fewer parameters are required while more layers are added into this architecture.

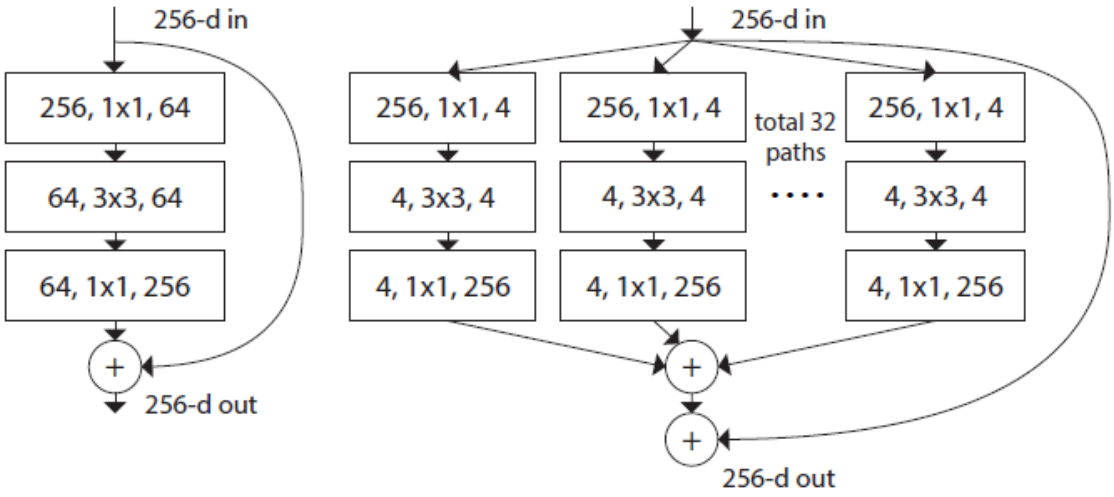

Figure 4. A block of ResNeXt

Relationship between Inception-ResNet and ResNeXt

- **ResNeXt Block**
  - For each path, Conv1×1–Conv3×3–Conv1×1 are done at each convolution path. This is the bottleneck design in ResNet block. The internal dimension for each path is denoted as d (d=4). The number of paths is the cardinality C (C=32). If we sum up the dimension of each Conv3×3, it is also the dimensions of 128.
  - The dimension is increased directly from 4 to 256, and then added together, and also added with the skip connection path.
  - Compared with Inception-ResNet that it needs to increase the dimension from 4 to 128 then to 256, ResNeXt requires minimal extra effort designing each path.
  - Unlike ResNet, in ResNeXt, the neurons at one path will not connected to the neurons at other paths.
- **Inception-ResNet Block**
  - or each convolution path, Conv1×1–Conv3×3 are done first. When added together (i.e. 4×32), the Conv3×3 has the dimension of 128.
  - Then the outputs are concatenated together with dimension of 128. And Conv1×1 is used to restore the dimensions from 128 to 256.
  - Finally the output is added with the skip connection path.
  - The main difference is that they have an early concatenation.

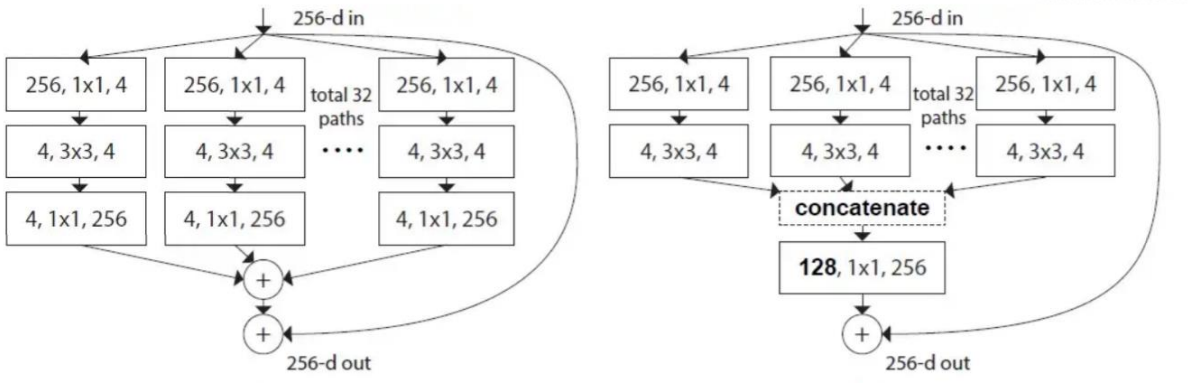

Figure 5. ResNeXt block and Inception-ResNet block

As mentioned in ResNeXt paper, with larger images for input, ResNeXt-101 outperforms ResNet and Inception-ResNet-v2 in ImageNet-1K classification.

Exercise 3

In this exercise we are going to solve an imagery classification problem on Intel Image classification dataset which consists of 25k samples belongs to 6 classes. For this purpose, we use transfer learning technique and use different architectures.

- **Data loading and preprocessing**
  Intel Image classification dataset consists of 3 parts, train, test, prediction. We just used first two parts of dataset. We used batch size of 32 for data loader and applied some augmentations on images such as resizing to 224x224, random horizontal flip, and normalization according to their means and stds.
- **Model training**
  1) **ResNet-50**
     As first model, we used ResNet-50, we changed last fully connected with a linear layer with 6 neurons. We loaded the model with pretrained weights on ImageNet dataset, then frozen all layers except last layer and train our model. We trained model for 10 epochs with Adam optimizer with learning rate of 0.001. we used test dataset as validation data during the training. You can see train and validation loss in below figure. We got 91.67% accuracy on test set.

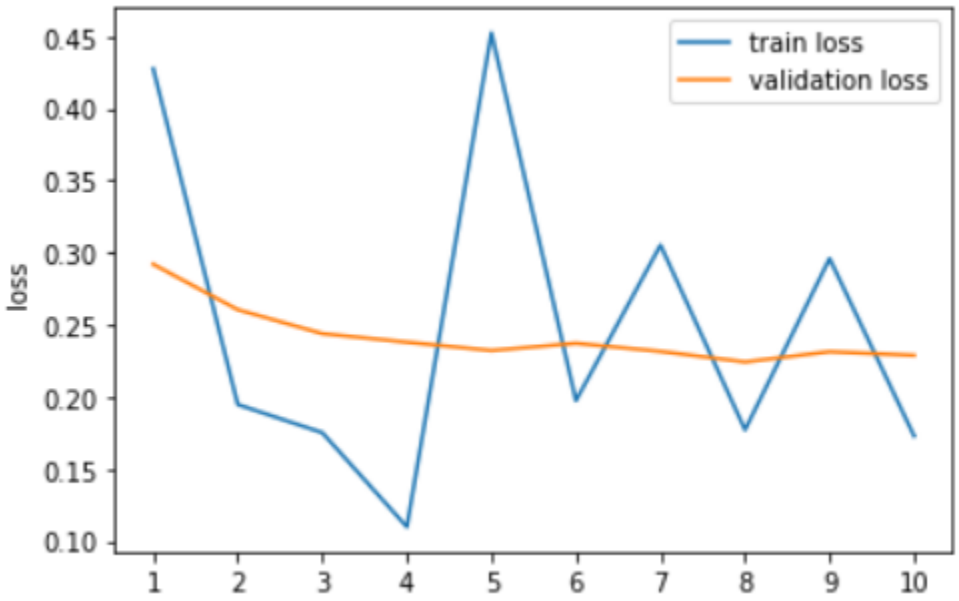

Figure 6. Train and validation loss of ResNet-50

2) **Finetuning ResNet-50**

We can finetune our base model as we freeze all blocks of **ResNet** except last block, then we will train the model. As mentioned above, we replaced last fully connected layer with proper number of neurons, then trained the model with same config as above for 5 epochs. We got accuracy of 92.87% on test set.

3) **Finetuning DenseNet- 201**

We used **DenseNet** as base model, too. Same as above we frozen all blocks of **DenseNet- 201** except last one, then trained the model. We trained the model with same config as above. After trained the model we got accuracy of 93.13% on test set. Below figure shows train and validation loss.

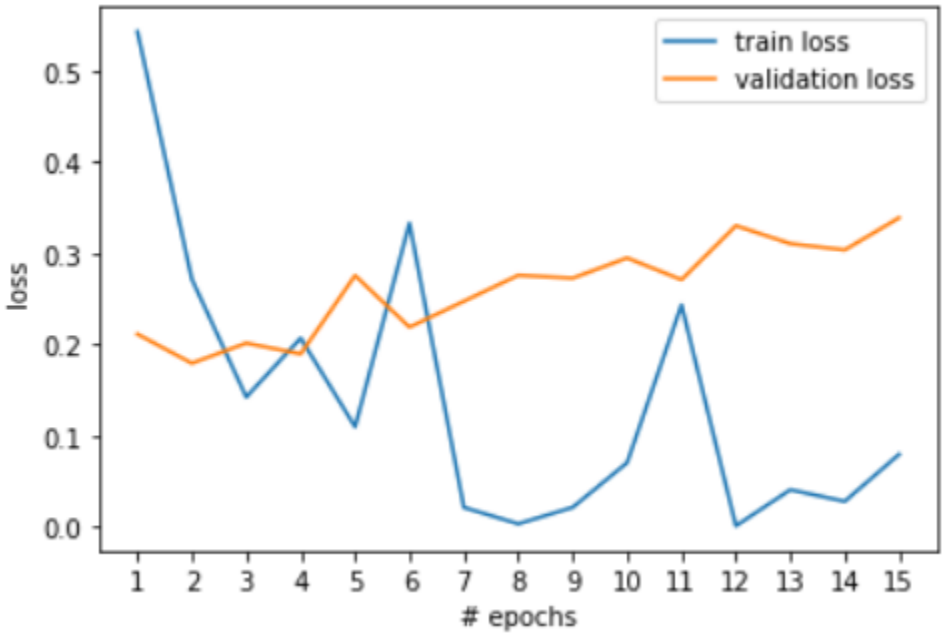

Figure 7.Train and validation loss of DenseNet-201

4) **Finetuning ResNeXt- 50**

We also used **ResNeXt** as base model. We frozen all blocks of model except last block, then trained model. After training the model for 10 epochs we got 92.77% accuracy on test set. You can see train and validation loss on below figure.

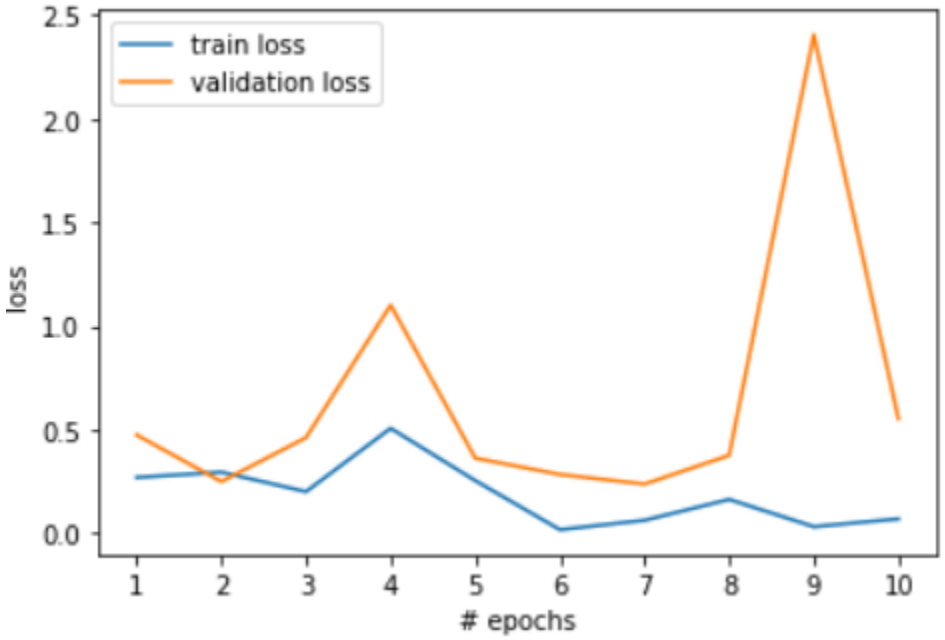

Figure 8. Train and validation loss of ResNeXt-50

5) **Vision Transformer**

As last model, we used **ViT** as base model which has got some state-of-the-art in various classification problems. We added an MLP on top of **ViT** and trained the model for 5 epochs. As expected, we got highest accuracy so far, 94.90% on test set.

- **Result**

After training 5 models with 4 different base model with/without finetuning we got highest accuracy by **ViT** with accuracy of 94.90%.

References

- https://arxiv.org/abs/1905.11946
- https://arxiv.org/abs/1611.05431
- https://arxiv.org/abs/1602.07261
- https://medium.com/dataseries/enhancing-resnet-to-resnext-for-image-classification-3449f62a774c
- https://medium.datadriveninvestor.com/resnext-explained-part-1-17a6b510fb0a