**2.4.1 GoogLeNet (Ziyu Guo)**

GoogLeNet, the winner of ILSVRC 2014, is an inception architecture (Szegedy et al.,2015a). It combines the multi-scale idea and dimension reduction layers based on the Hebbian principle and embedding learning (Mehdipour Ghazi et al., 2017). There are 4 versions of GoogLeNet. In this project, we used Inception V3 model to do fine tuning experiment.

Inception V3 model has 42 layers and about 7 million parameters[[1]](#footnote-1). The most important feature of Inception V3 is that it uses factorizing convolutions. The factorization is aimed at reducing the number of parameters without reducing the efficiency of the neural network.

In Inception V3, the original 1 layer of 5x5 convolution layer is replaced by two 3x3 convolution layer (Figure 2.4.1.1). In this way, one 5x5 convolution layer with 25 parameters has been reduced to two 3x3 convolution layers with a total of 9+9=18 parameters. With the parameter reduction, the network will be less likely to be overfitting. It also allows the network to go deeper.

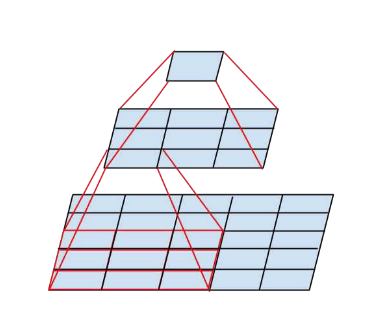


Figure 2.4.1.1 ﻿Mini-network replacing the 5 × 5 convolutions

*Figure source: Szegedy et al. 2015b*

Figure 2.4.1.2 is an architecture to describe the Inception V3 network. The model is made up of symmetric and asymmetric building blocks. There are six types of layers in this model: convolutions layer, average pooling layer, max pooling layer, concats layer, dropouts layer, and fully connected layer. The loss is computed using softmax[[2]](#footnote-2). It is much more efficient than VGGNet in terms of computation cost.

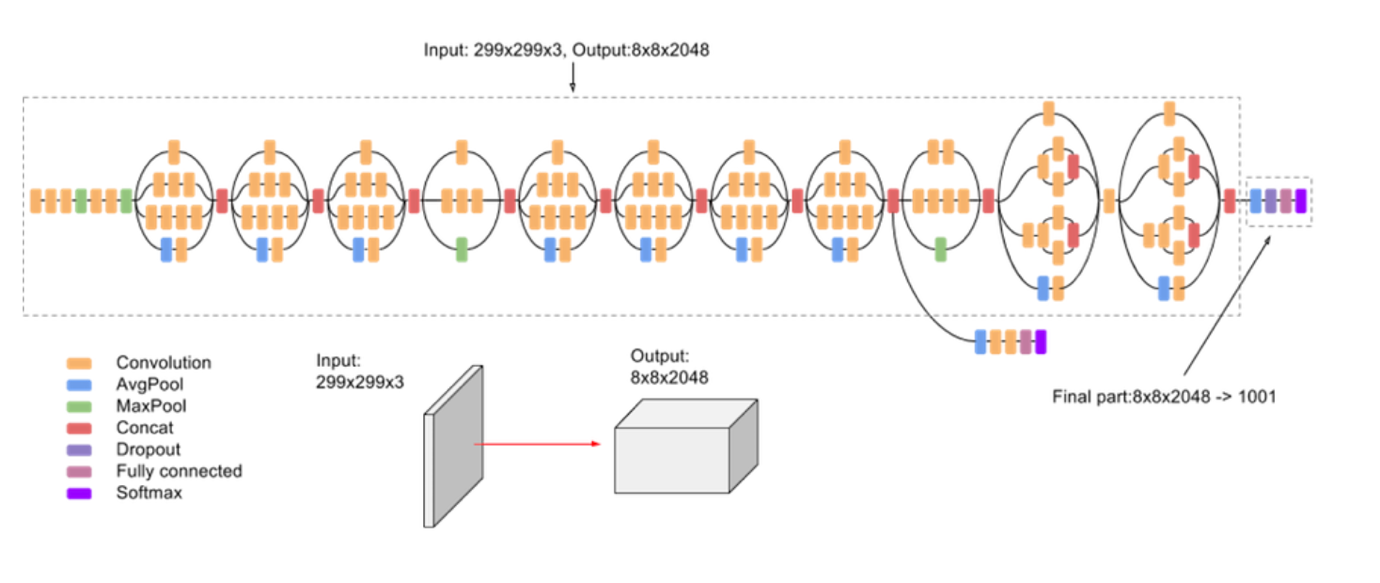


Figure 2.4.1.2 ﻿ Inception V3 network archetechture

*Figure source: https://cloud.google.com/tpu/docs/inception-v3-advanced*

**4.3.1 GoogLeNet (Ziyu Guo)**

**1. Experiment Design**

There are 8 groups of experiments with different sample size as show in table 4.3.1.1. For each sample size, we use 80% of data for training and 20% of data for testing. In each experiment, we try to training 9 different set of parameters to see how is the number of trainable parameters can affect the testing accuracy. In the experiments, we use batch size = 40 for all samples sizes. We choose 20 epochs and 0.0004 as our learning rate.

|  |  |
| --- | --- |
| Table 4.3.1.1 Sample size and Average experiment time | |
| **Sample size** | **Average experiment time (seconds/20 Epochs)** |
| **400** | 330 |
| **1200** | 970 |
| **2000** | 1640 |
| **2800** | 2120 |
| **3600** | 2950 |
| **4400** | 3780 |
| **5200** | 4320 |
| **6000** | 5150 |

When sample size is bigger than 2000, the experiment become very time consuming. Therefore, I only train 5 different set of parameters for sample size = 2800, 3600, 4400, 5200 and 6000, to see how is the number of trainable parameters can affect the testing accuracy.

|  |  |
| --- | --- |
| Table 4.3.1.2 Added layer type and number of parameter | |
| **Layer type** | **Number of parameters** |
| **Dense Layer 1** | 262272 |
| **Dense Layer 2** | 16512 |
| **Output Layer** | 516 |

In order to do fine tuning with Inception V3 model, we exclude the output layer in the original model and add 3 layers as shown in table 4.3.1.2. With the change of the model, the output will be 4 dimensional vector.

|  |  |  |  |
| --- | --- | --- | --- |
| Table 4.3.1.3 Sublayers from 283-310 in original Inception V3 model | | | |
| **Sublayer number** | **Sublayer name** | **Sublayer number** | **Sublayer name** |
| **283** | conv2d\_87 | **297** | batch\_normalization\_92 |
| **284** | conv2d\_91 | **298** | batch\_normalization\_93 |
| **285** | batch\_normalization\_87 | **299** | conv2d\_94 |
| **286** | batch\_normalization\_91 | **300** | batch\_normalization\_86 |
| **287** | activation\_87 | **301** | activation\_88 |
| **288** | activation\_91 | **302** | activation\_89 |
| **289** | conv2d\_88 | **303** | activation\_92 |
| **290** | conv2d\_89 | **304** | activation\_93 |
| **291** | conv2d\_92 | **305** | batch\_normalization\_94 |
| **292** | conv2d\_93 | **306** | activation\_86 |
| **293** | average\_pooling2d\_9 | **307** | mixed9\_1 |
| **294** | conv2d\_86 | **308** | concatenate\_2 |
| **295** | batch\_normalization\_88 | **309** | activation\_94 |
| **296** | batch\_normalization\_89 | **310** | mixed10 |

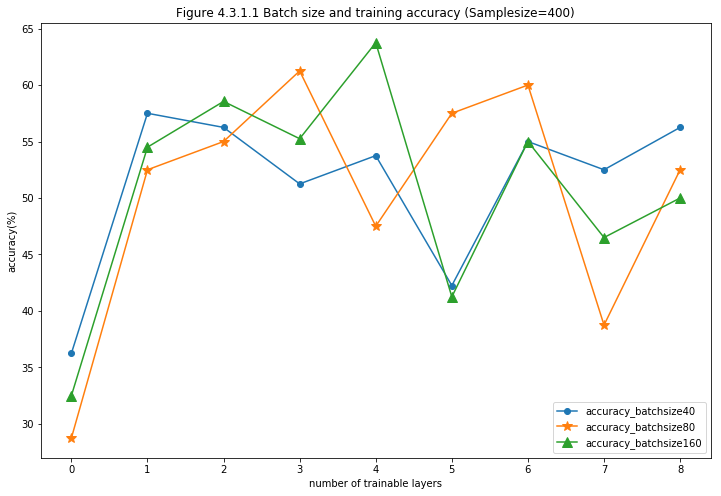
Inception V3 model is very complicated and contains a total of 42 layers which consisting 310 sublayers[[3]](#footnote-3). Many sublayers do not have any trainable parameters. Therefore, I only count the sublayers as one sublayer when its type named ‘Conv2D’(see table 4.3.1.3). The number of trainable layers and number of trainable parameters are given in table 4.3.1.4. Sublayers 310-312 refer to the 3 new added layers.

|  |  |  |  |
| --- | --- | --- | --- |
| Table 4.3.1.4 Trainable number of layers and trainable parameters | | | |
| **Number of trainable sublayers** | **Corresponding**  **sublayer**  **number** | **Number of trainable parameters** | **Number of nontrainable parameters** |
| **0** | 310-312 | 279,300 | 21,802,784 |
| **1** | 299-312 | 673,028 | 21,409,056 |
| **2** | 294-312 | 1,329,924 | 20,752,160 |
| **3** | 292-312 | 1,772,292 | 20,309,792 |
| **4** | 291-312 | 2,214,660 | 19,867,424 |
| **5** | 290-312 | 2,657,028 | 19,425,056 |
| **6** | 289-312 | 3,099,396 | 18,982,688 |
| **7** | 284-312 | 4,648,452 | 17,433,632 |
| **8** | 283-312 | 5,434,884 | 16,647,200 |

**2. Experiment results**

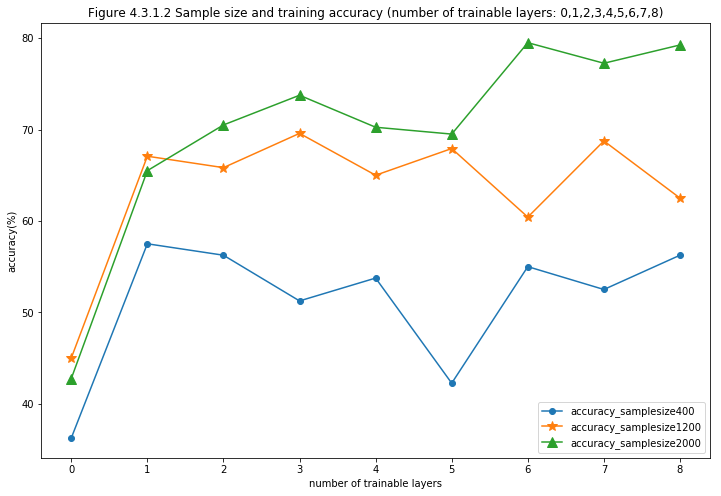
In this part, we show the experimental results. First, in order to see if the batch size affect the accuracy, I test different batch sizes on the sample size = 400. Table 4.3.1.5 show the results and figure 4.3.1.1 plots these results.

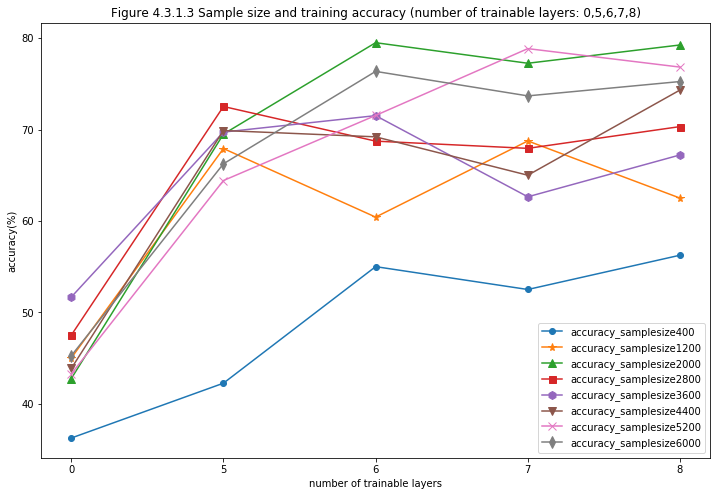
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table 4.3.1.5 Batch size and testing accuracy(sample size = 400) | | | | | | | | | | |
| **Number of trainable layers**  **Batch**  **size** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| **40** | 36.25% | 57.5% | 56.25% | 51.25% | 53.75% | 42.25% | 55.00% | 52.50% | 56.25% |
| **80** | 28.75% | 52.50% | 55.00% | 61.25% | 47.50% | 57.50% | 60.00% | 38.75% | 52.50% |
| **160** | 32.50% | 54.50% | 58.55% | 55.25% | 63.75% | 41.25% | 55.00% | 46.50% | 50.00% |

****

In table 4.3.1.6, we show the experiment results for different sample sizes with different set of number of trainable sublayers. Figure 4.3.1.2 shows the results for sample size = 400, 1200 and 2000 with 9 different sets of trainable layers. Figure 4.3.1.3 shows the results for sample size =400, 1200, 2000, 2800, 3600, 4400, 5200 and 6000 with 5 different sets of trainable layers.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table 4.3.1.6 Sample size and testing accuracy | | | | | | | | | |
| **Number of trainable layers**  **Sample**  **size** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| **400** | 36.25% | 57.5% | 56.25% | 51.25% | 53.75% | 42.25% | 55.00% | 52.5% | 56.25% |
| **1200** | 45.00% | 67.08% | 65.83% | 69.58% | 65.00% | 67.92% | 60.42% | 68.75% | 62.50% |
| **2000** | 42.75% | 65.50% | 70.50% | 73.75% | 70.25% | 69.50% | 79.50% | 77.25% | 79.25% |
| **2800** | 47.50% | N/A | N/A | N/A | N/A | 72.52% | 68.73% | 67.94% | 70.32% |
| **3600** | 51.67% | N/A | N/A | N/A | N/A | 69.72% | 71.52% | 62.64% | 67.22% |
| **4400** | 43.86% | N/A | N/A | N/A | N/A | 69.89% | 69.20% | 65.00% | 74.32% |
| **5200** | 43.27% | N/A | N/A | N/A | N/A | 64.42% | 71.54% | 78.85% | 76.83% |
| **6000** | 45.25% | N/A | N/A | N/A | N/A | 66.25% | 76.36% | 73.68% | 75.25% |

****

****

**3. Discussion of results**

From table 4.3.1.5 and figure 4.3.1.1, we can see different batch sizes do not significantly affect the testing accuracy. With different batch sizes, the average testing accuracy with sample size = 400 is around 35% for layer=0 and 55% for layers between 1-8. The reason for that is most likely because the batch size is still reasonably small. When it increase to bigger than 1000 and with only 20 epochs, we do expect a lower testing accuracy.

From table 4.3.1.6 and figure 4.3.1.2, we can see the sample size affects the testing accuracy. When sample size = 400, the testing accuracy is around 35% for layer =0 and 52% for layers between 1-8. When sample size=1200, the testing accuracy is around 45% for layer =0 and 65% for layers between 1-8. When sample size= 2000, the testing accuracy is around 42% for layer =0 and 70% for layers between 1-8. Therefore, larger sample size will have higher testing accuracy. However, according to figure 4.3.1.3, the testing accuracy did not have significant improvement when sample size increase from 2000 to 6000. This is probably because there is no big difference for such a large neural nets to train on the level of ‘thousand’ number of pictures. It is probably that when the sample size increase to more than ten thousand, the accuracy will change.

Another interesting results is that when the number of trainable layer =0, i.e. when the number of trainable parameters are small, the testing accuracy keeps around 45% when sample size is bigger than 1200. But when the trainable layers is larger than 0, the testing accuracy is around 70% when sample size is bigger than 1200. This means that only fine tuning a small set of trainable parameters in Inception V3 model cannot generate good classification performance. It also confirms that with factorization convolution, Inception V3 model can deal with overfitting problem really well. Even with 5 million trainable parameters, the performance is still good. When compare with the results from the VGGNets, we can see the benefit of the Inception V3 model more clearly.

Reference from Ziyu Guo

[1] C. Szegedy *et al.*, “Going deeper with convolutions,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 07-12-June-2015, pp. 1–9, 2015a

[2] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the Inception Architecture for Computer Vision,” 2015b.

[3] M. Mehdipour Ghazi, B. Yanikoglu, and E. Aptoula, “Plant identification using deep neural networks via optimization of transfer learning parameters,” *Neurocomputing*, vol. 235, no. August 2016, pp. 228–235, 2017.

[4] https://cloud.google.com/tpu/docs/inception-v3-advanced

[5] https://medium.com/@sh.tsang/review-inception-v3-1st-runner-up-image-classificat- ion-in-ilsvrc-2015-17915421f77c

1. Some layers contain different kind of parallel sublayers, and the total number of sublayers is 310. [↑](#footnote-ref-1)
2. https://cloud.google.com/tpu/docs/inception-v3-advanced [↑](#footnote-ref-2)
3. Sublayers form into layers. [↑](#footnote-ref-3)