type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	double #3×3 reduce	double #3×3	Pool +proj
convolution*	7×7/2	112×112×64	1						
max pool	3×3/2	56×56×64	0						
convolution	3×3/1	56×56×192	1		64	192			
max pool	3×3/2	28×28×192	0						
inception (3a)		28×28×256	3	64	64	64	64	96	avg + 32
inception (3b)		28×28×320	3	64	64	96	64	96	avg + 64
inception (3c)	stride 2	28×28×576	3	0	128	160	64	96	max + pass through
inception (4a)		14×14×576	3	224	64	96	96	128	avg + 128
inception (4b)		14×14×576	3	192	96	128	96	128	avg + 128
inception (4c)		14×14×576	3	160	128	160	128	160	avg + 96
inception (4d)		14×14×576	3	96	128	192	160	192	avg + 128
inception (4e)	stride 2	14×14×1024	3	0	128	192	192	256	max + pass through
inception (5a)		7×7×1024	3	352	192	320	160	224	avg + 128
inception (5b)		7×7×1024	3	352	192	320	192	224	max + 128
avg pool	7×7/1	1×1×1024	0						

Figure 1: Inception architecture

Variant of the Inception Model Used

Figure 1 documents the changes that were performed compared to the architecture with respect to the GoogleNet archictecture. For the interpretation of this table, please consult [1]. The notable architecture changes compared to the GoogLeNet model include:

- The 5×5 convolutional layers are replaced by two consecutive 3×3 convolutional layers. This increases the maximum depth of the network by 9 weight layers. Also it increases the number of parameters by 25% and the computational cost is increased by about 30%.
- The number 28×28 inception modules is increased from 2 to 3.
- Inside the modules, sometimes average, sometimes maximum-pooling is employed. This is indicated in the entries corresponding to the pooling layers of the table.
- There are no across the board pooling layers between any two Inception modules, but stride-2 convolution/pooling layers are employed before the filter concatenation in the modules 3c, 4e.
- There is a single side tower on top of the module 4c: it is followed by a 5×5 average pooling with 128 filters with stride 3. This is followed by a 768 nodes bottleneck layer before the 1000 way linear classifier with softmax loss. The weight of this classifier in the final objective is 0.4. The weight of the main classifier is 1.

Our model employed separable convolution with depth multiplier 8 on the first convolutional layer. This reduces the computational cost while increasing the memory consumption at training time.

References

[1] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014. 1