Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning

Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, Alexander A. Alemi
Google Inc.
1600 Amphitheatre Parkway
Mountain View, CA

Abstract

Very deep convolutional networks have been central to the largest advances in image recognition performance in recent years. One example is the Inception architecture that has been shown to achieve very good performance at relatively low computational cost. Recently, the introduction of residual connections in conjunction with a more traditional architecture has yielded state-of-the-art performance in the 2015 ILSVRC challenge; its performance was similar to the latest generation Inception-v3 network. This raises the question: Are there any benefits to combining Inception architectures with residual connections? Here we give clear empirical evidence that training with residual connections accelerates the training of Inception networks significantly. There is also some evidence of residual Inception networks outperforming similarly expensive Inception networks without residual connections by a thin margin. We also present several new streamlined architectures for both residual and non-residual Inception networks. These variations improve the single-frame recognition performance on the ILSVRC 2012 classification task significantly. We further demonstrate how proper activation scaling stabilizes the training of very wide residual Inception networks. With an ensemble of three residual and one Inception-v4 networks, we achieve 3.08% top-5 error on the test set of the ImageNet classification (CLS) challenge.

Introduction

Object recognition is a central task for computer vision and artificial intelligence in general. Strong vision models are key enabling components of AI systems that can process visual inputs. Their applications include computer user interfaces (gesture recognition), web search, OCR systems, autonomous transportation, medical imaging, areal imaging, robotics and image processing. Before 2012, specialized solutions were required for each specific application domain. Since then, deep convolutional neural networks have become mainstream to address these tasks. Convolutional neural networks go back to the 1980s (Fukushima 1980) and (LeCun et al. 1989), but recent good results (Krizhevsky, Sutskever, and Hinton 2012) on the large scale ImageNet image-recognition benchmark ILSVRC (Russakovsky et al. 2014) has lead to a revived interest in their use. The same neural network architecture "AlexNet" (Krizhevsky, Sutskever, and Hinton 2012) has been applied to a large number of application domains with good results.

The same architectures that work well for object detection can be applied successfully to a wide variety of computer vision tasks, including object-detection (Girshick et al. 2014), segmentation (Long, Shelhamer, and Darrell 2015), human pose estimation (Toshev and Szegedy 2014), video classification (Karpathy et al. 2014), object tracking (Wang and Yeung 2013), and super-resolution (Dong et al. 2014). These examples are but a few of the multitude of applications to which deep convolutional networks have been very successfully applied ever since. Moreover it has been demonstrated that architectural improvements of convolutional networks that target recognition performance tend to translate to performance gains for the other tasks as well.

This universal applicability motivates our focus on recognition models for the widely used ILSVRC12 object-recognition benchmark (Russakovsky et al. 2014) on which the task is to classify the images into one (or five) of a thousand different classes. The dataset comprises of 1.2 million training images, 50,000 validation images and 100,000 test images. All of them are divided up equally between the 1000 classes. This benchmark has been a very popular task to measure the quality of object recognition solutions since 2010.

In this work we study the combination of two of the most recent ideas: Residual connections (He et al. 2015) and the latest revised version of the Inception architecture (Szegedy et al. 2015b). In (He et al. 2015), it is argued that residual connections are inherently important for training very deep architectures. Since Inception networks tend to be very deep, it is natural to replace the filter concatenation stage of the Inception architecture with residual connections. This would allow Inception to reap the benefits of the residual approach while retaining its computational efficiency.

Besides a straightforward integration, we have also studied whether Inception without residual connections can be made more efficient by making it deeper and wider. For that purpose, we evaluated a new version named Inception-v4 which has a more uniform simplified architecture and more inception modules than Inception-v3. Historically, Inception-v3 had inherited a lot of the baggage of the earlier incarnations. The technical constraints chiefly came from the need for partitioning the model for distributed training using DistBelief (Dean et al. 2012). Now, after migrating our training setup to TensorFlow (Abadi et al. 2015) these constraints have been lifted, which allowed us to simplify the architecture significantly. The details of that simplified architecture are described in the Architectural Choices Section starting on page 2.

In this paper, we will compare the two pure Inception variants, Inception-v3 and v4, with similarly expensive hybrid Inception-ResNet versions. Admittedly, those models were picked in a somewhat ad hoc manner with the main constraint being that the parameters and computational complexity of the models should be somewhat similar to the cost of the non-residual models. We have tested bigger and wider Inception-ResNet variants and they performed at the level of Inception-ResNet-v2 on the ImageNet classification challenge (Russakovsky et al. 2014) dataset.

Lastly, we report an evaluation of an ensemble of all models described. As it was apparent that both Inception-v4 and Inception-ResNet-v2 performed similarly well, each exceeding state-of-the art single-frame performance on the ImageNet validation dataset, we wanted to see how a combination pushes the state-of-the-art on this well studied dataset. Surprisingly, we found that gains on the single-frame performance do not translate into similarly large gains on ensembled performance. Nonetheless, it still allows us to report 3.1% top-5 error on the validation set with four models ensembled setting a new state-of-the-art, to the best of our knowledge.

Related Work

Convolutional networks have become popular in large scale image recognition tasks after (Krizhevsky, Sutskever, and Hinton 2012). Some of the next important milestones were Network-in-network by (Lin, Chen, and Yan 2013), VGGNet by (Simonyan and Zisserman 2014) and GoogLeNet (Inception-v1) by (Szegedy et al. 2015a).

Residual connections were introduced in (He et al. 2015) in which they give convincing theoretical and practical evidence for the advantages of utilizing additive merging of signals both for image recognition, and especially for object detection. The authors argued that residual connections are inherently necessary for training very deep convolutional models. Our findings do not seem to support this view, at least for image recognition. However it might require more experiments with even deeper networks to fully understand the true benefits of residual connections. In the experimental section we demonstrate that it is not very difficult to train very deep competitive networks without utilizing residual connections. However the use of residual connections seems to improve the training speed greatly, which is alone a great argument for their use.

The Inception deep convolutional architecture was introduced as GoogLeNet in (Szegedy et al. 2015a), here named Inception-v1. Later the Inception architecture was refined in various ways, first by the introduction of batch normalization (Ioffe and Szegedy 2015) (Inception-v2). Later by additional factorization ideas in the third iteration (Szegedy et al. 2015b) which will be referred to as Inception-v3 in this report.

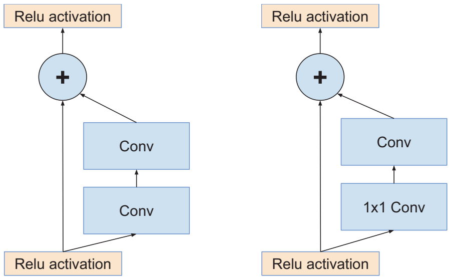

Figure 1: Residual connections as introduced in (He et al. 2015). On the left is the original residual connection. On the right is an optimized version that reduces the computational cost by the use of a 1 × 1 convolution.

Architectural Choices

Pure Inception blocks

Our older Inception models used to be trained in a partitioned manner, where each replica was partitioned into multiple sub-networks in order to be able to fit the whole model in memory. However, the Inception architecture is highly tunable, meaning that there are a lot of possible changes to the number of filters in the various layers that do not affect the quality of the fully trained network. In order to optimize the training speed, we used to tune the layer sizes carefully in order to balance the computation between the various model sub-networks. In contrast, with the introduction of TensorFlow (Abadi et al. 2015), our most recent models can be trained without partitioning the replicas. This is enabled in part by recent memory optimizations to backpropagation, achieved by carefully considering which tensors are needed for gradient computation and structuring the computation to reduce the number of such tensors. Historically, we have been relatively conservative about changing the architecture and restricted our experiments to varying isolated network components while keeping the rest of the network stable. Not simplifying earlier choices resulted in networks that looked more complicated that they needed to be. In our newer experiments, for Inception-v4 we decided to shed this unnecessary baggage and made uniform choices for the Inception blocks for each grid size.

The full configuration of the Inception-v4 network is outlined in Figures 2, which contains the overall schema and stem configuration and Figure 3, which details the construction of the interior modules.

Residual Inception Blocks

For the residual versions of the Inception networks, we use cheaper Inception blocks than the original Inception. Each Inception block is followed by filter-expansion layer (1 × 1 convolution without activation) which is used for scaling up the dimensionality of the filter bank before the residual addition to match the depth of the input. This is needed to compensate for the dimensionality reduction induced by the Inception block.

We tried several versions of the residual version of Inception. Only two of them are detailed here. The first one "Inception-ResNet-v1" has roughly the computational cost of Inception-v3, while "Inception-ResNet-v2" matches the raw cost of the newly introduced Inception-v4 network. However, the step time of Inception-v4 proved to be significantly slower in practice, probably due to the larger number of layers.

Another small technical difference between our residual and non-residual Inception variants is that in our Inception-ResNet experiments, we used batch-normalization only on top of the traditional layers, but not on top of the residual summations. It is reasonable to expect that a thorough use of batch-normalization should be advantageous, but the implementation of batch-normalization in TensorFlow was consuming a lot of memory and we would have needed to decrease the overall number of layers, if batch-normalization had been used everywhere.

The full configuration of the Inception-Resnet-v1 network is outlined in Figure 6 which contains the overall schema and stem configuration and Figure 4 that has the detailed configuration of the interior modules.

The full configuration of the Inception-Resnet-v2 network uses the schema in Figure 6, the stem in Figure 2, and the modules in Figure 5.

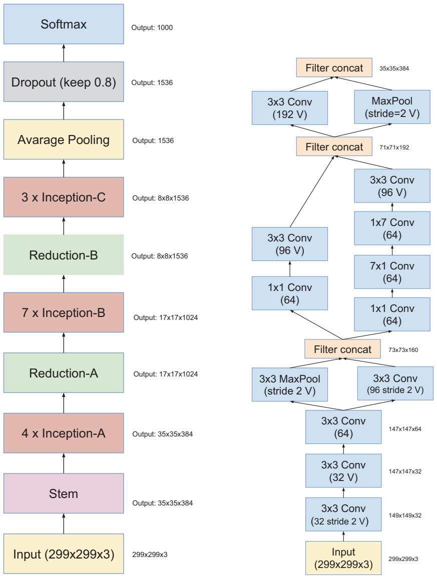

Figure 2: On the left is the overall schema for the pure Inception-v4 network. On the right is the detailed composition of the stem. Note that this stem configuration was also used for the Inception-ResNet-v2 network outlines in Figures 5, 6. V denotes the use of 'Valid' padding, otherwise 'Same' padding was used. Sizes to the side of each layer summarize the shape of the output for that layer.

Scaling of the Residuals

We found that if the number of filters exceeded 1000, the residual variants started to exhibit instabilities and the network just "died" early in the training, meaning that the last layer before the average pooling started to produce only zeros after a few tens of thousands of iterations. This could not be prevented, either by lowering the learning rate, or by adding an extra batch-normalization to this layer.

We found that scaling down the residuals before adding them to the previous layer activation seemed to stabilize the training. In general we picked some scaling factors between 0.1 and 0.3 to scale the residuals before their being added to the accumulated layer activations (cf. Figure 7).

A similar instability was observed by (He et al. 2015) in the case of very deep residual networks and they suggested a two-phase training where the first "warm-up" phase is done with very low learning rate, followed by a second phase with high learning rate. We found that if the number of filters is very high, then even a very low (0.00001) learning rate is not sufficient to cope with the instabilities and the training with high learning rate had a chance to destroy its effects. We found it much more reliable to scale the residuals.

Even where the scaling was not strictly necessary, it never seemed to harmed the final accuracy, but it helped to stabilize the training.

Training Methodology

We have trained our networks with stochastic gradient descent, utilizing the TensorFlow (Abadi et al. 2015) distributed machine learning system using 20 replicas, each running a NVidia Kepler GPU. Our earlier experiments used momentum (Sutskever et al. 2013) with a decay of 0.9, while our best models were achieved using RMSProp (Tieleman and Hinton) with a decay of 0.9 and $\epsilon = 1.0$. We used a learning rate of 0.045, decayed every two epochs using an exponential rate of 0.94. In addition, gradient clipping (Pascanu, Mikolov, and Bengio 2012) was found to be useful to stabilize the training. Model evaluations are performed using a running average of the parameters computed over time.

Experimental Results

First we observe the evolution of the top-1 and top-5 validation-error of the four variants during training. After the experiment was conducted, we have found that our continuous evaluation was conducted on a subset of the validation set which omitted about 1700 blacklisted entities due to poor bounding boxes. It turned out that the omission should have been only performed for the CLSLOC benchmark, but yields somewhat incomparable (more optimistic) numbers when compared to other reports including some earlier reports by our team. The difference is about 0.3% for top-1 error and about 0.15% for the top-5 error. However, since the differences are consistent, we think the comparison between the curves is a fair one.

On the other hand, we have rerun our multi-crop and ensemble results on the complete validation set consisting of 50,000 images. Also the final ensemble result was also performed on the test set and sent to the ILSVRC test server

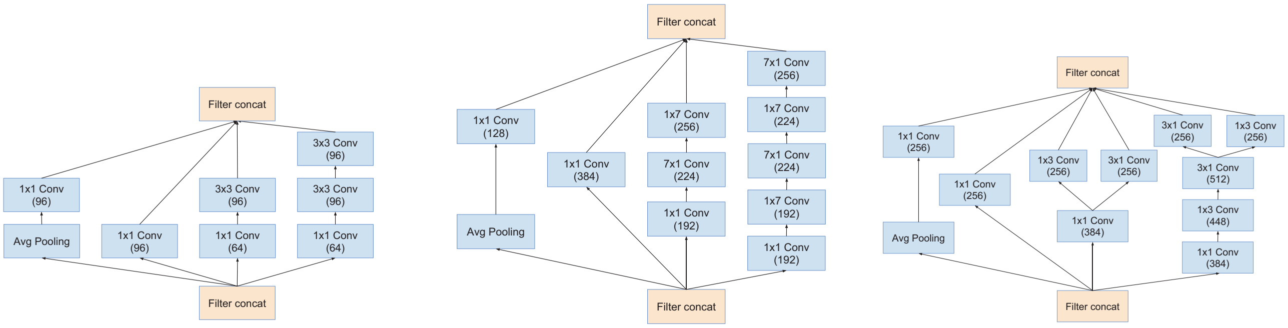

Figure 3: The schema for interior grid modules of the pure Inception-v4 network. The 35 × 35, 17 × 17 and 8 × 8 grid modules are depicted from left to right. These are the Inception-A, Inception-B, and Inception-C blocks of Figure 2 respectfully.

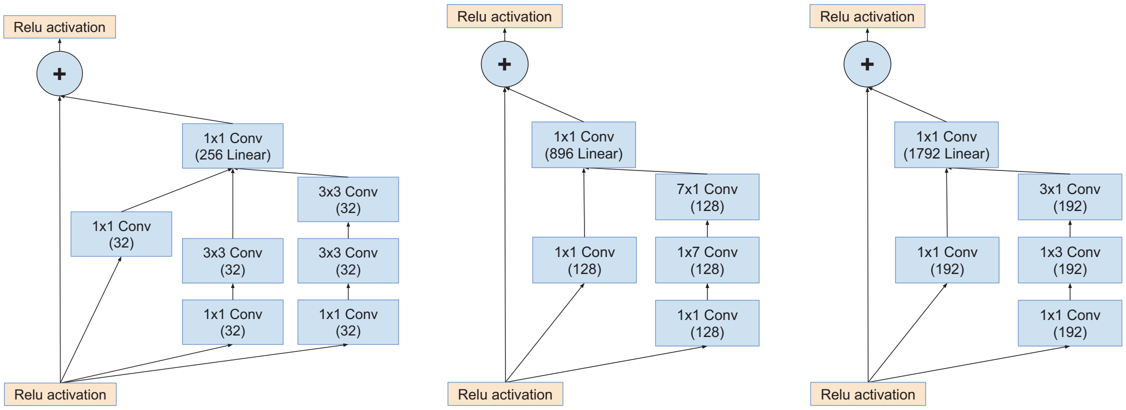

Figure 4: The schema for interior grid modules of the Inception-ResNet-v1 network. The 35 × 35, 17 × 17 and 8 × 8 grid modules are depicted from left to right. These are the Inception-A, Inception-B, and Inception-C blocks of the schema on the left of Figure 6 for the Inception-ResNet-v1 network, respectfully.

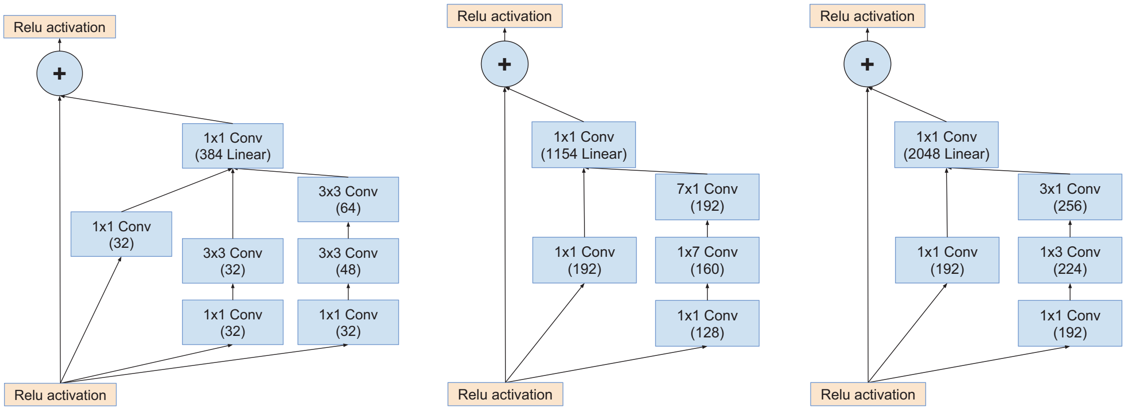

Figure 5: The schema for interior grid modules of the Inception-ResNet-v2 network. The 35 × 35, 17 × 17 and 8 × 8 grid modules are depicted from left to right. These are the Inception-A, Inception-B and Inception-C blocks of the schema on the left of Figure 6 for the Inception-ResNet-v2 network, respectfully.

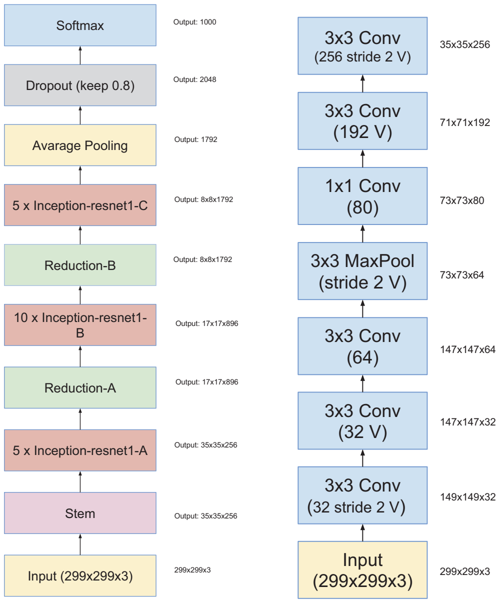

Figure 6: On the left is the overall schema for the Inception-Resnet-v1 and Inception-Resnet-v2 network. While the schema are the same for both networks, the composition of the stem and interior modules differ. The stem of Inception-Resnet-v1 is shown to the right, while the stem of Inception-Resnet-v2 is the same as the pure Inception-v4 network, depicted on the right of Figure 2. The interior modules are denoted in Figure 4 and Figure 5 respectfully. V denotes the use of 'Valid' padding, otherwise 'Same' padding was used. Sizes to the side of each layer summarize the shape of the output for that layer.

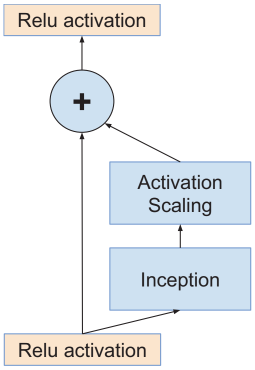

Figure 7: The general schema for scaling combined Inception-ResNet modules. We expect that the same idea is useful in the general ResNet case, where instead of the Inception block an arbitrary subnetwork is used. The scaling block just scales the last linear activations by a suitable constant, typically around 0.1, but we found that deeper networks require lower constants.

for validation to verify that our tuning did not result in overfitting. We would like to stress that this final validation was done only once and we have submitted our results only twice in the last year: once for the BN-Inception paper and later during the ILSVR-2015 CLSLOC competition, so we believe that the test set numbers constitute a true estimate of the generalization capabilities of our model.

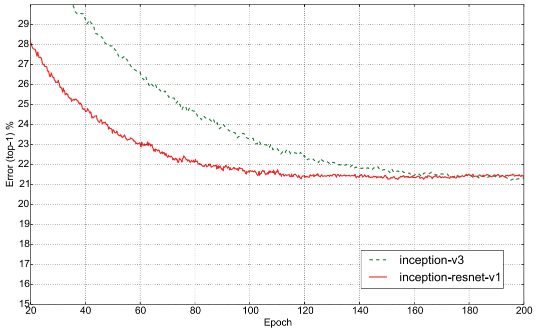

Figure 8: Top-1 error evolution during training of pure Inception-v3 vs a residual network of similar computational cost. The evaluation is measured on a single-crop on the non-blacklist images of the ILSVRC-2012 validation set. The residual model was training much faster, but reached slightly worse final accuracy than the traditional Inception-v3.

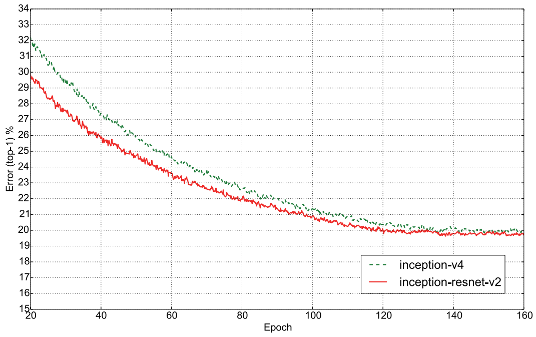

Figure 9: Top-1 error evolution during training of pure Inception-v3 vs a residual Inception of similar computational cost. The evaluation is measured on a single-crop on the non-blacklist images of the ILSVRC-2012 validation set. The residual version was training much faster and reached slightly better final accuracy than the traditional Inception-v4.

Finally, we present some comparisons between various versions of Inception and Inception-ResNet. The models Inception-v3 and Inception-v4 are deep convolutional networks not utilizing residual connections while Inception-ResNet-v1 and Inception-ResNet-v2 are Inception style networks that utilize residual connections instead of filter concatenation.

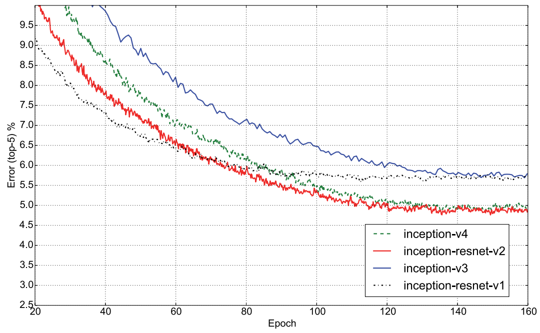

Figure 10: Top-5 error evolution of all four models (single-model, single-crop). This shows the improvement due to larger model size. Although the residual version converges faster, the final accuracy seems to mainly depend on the model size.

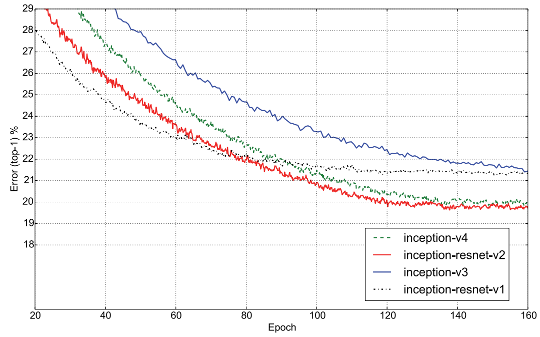

Figure 11: Top-1 error evolution of all four models (single-model, single-crop). This paints a similar picture as the top-5 evaluation in Figure 10.

Network	**Top-1 Error**	**Top-5 Error**
BN-Inception (Ioffe and Szegedy 2015)	25.2%	7.8%
Inception-v3 (Szegedy et al. 2015b)	21.2%	5.6%
Inception-ResNet-v1	21.3%	5.5%
Inception-v4	20.0%	5.0%
Inception-ResNet-v2	19.9%	4.9%

Table 1: Single-crop – single-model experimental results. Reported on the non-blacklisted subset of the validation set of ILSVRC 2012.

Network	Crops	**Top-1 Error**	**Top-5 Error**
ResNet (He et al. 2015)	10	25.2%	7.8%
Inception-v3 (Szegedy et al. 2015b)	12	19.8%	4.6%
Inception-ResNet-v1	12	19.8%	4.6%
Inception-v4	12	18.7%	4.2%
Inception-ResNet-v2	12	18.7%	4.1%

Table 2: 10/12-crop evaluations – single-model experimental results. Reported on all the 50,000 images of the validation set of ILSVRC 2012.

Network	Crops	**Top-1 Error**	**Top-5 Error**
Inception-v3 (Szegedy et al. 2015b)	144	18.9%	4.3%
Inception-ResNet-v1	144	18.8%	4.3%
Inception-v4	144	17.7%	3.8%
Inception-ResNet-v2	144	17.8%	3.7%

Table 3: 144-crop evaluations – single-model experimental results. Reported on the all 50,000 images of the validation set of ILSVRC 2012. In comparison (He et al. 2015) report a Top-1 Error of 19.4% and Top-5 Error of 4.5% for ResNet-151 but with a different 'dense' evaluation strategy.

Network	N	**Top-1 Error**	**Top-5 Error**
ResNet (He et al. 2015)	6	N/A	3.6%
Inception-v3 (Szegedy et al. 2015b)	4	17.3%	3.6%
Inception-v4(+Residual)	4	16.4%	3.1%

Table 4: Ensemble results with 144-crop/dense evaluation. Reported on the all 50,000 images of the validation set of ILSVRC 2012. The second column (N) denotes how many models were ensembled. For Inception-v4(+Residual), the ensemble consists of one pure Inception-v4 and three Inception-ResNet-v2 models and were evaluated both on the validation and on the test-set. The test-set performance was 3.08% top-5 error verifying that we don't over-fit on the validation set.

Table 1 shows the single-model, single-crop top-1 and top-5 error of the various architectures on the validation set.

Table 2 shows the performance of the various models with a small number of crops: 10 crops for ResNet as was reported in (He et al. 2015), for the Inception variants, we have used the 12-crop evaluation as as described in (Szegedy et al. 2015a).

Table 3 shows the single-model performance of the various models using a 144-crop evaluation. These are competitive in comparison to the reported 19.4% top-1 and 4.5% top-5 error rate reported in (He et al. 2015), albeit with a slightly different 'dense' evaluation strategy. For the inception networks, the 144-crop strategy was used as described in (Szegedy et al. 2015a).

Table 4 compares ensemble results. For the pure residual network the 6-model dense evaluation result is reported from (He et al. 2015). For the inception networks 4 models were ensembled using the 144-crop strategy as described in (Szegedy et al. 2015a).

Open source implementations of the Inception-ResNet-v2 and Inception-v4 models in this paper as well as pre-trained weights are available at the TensorFlow Models github page: github.com/tensorflow/models.

Conclusions

We have presented three new network architectures in detail:

- Inception-ResNet-v1: a hybrid Inception version that has a similar computational cost to Inception-v3 from (Szegedy et al. 2015b).
- Inception-ResNet-v2: a costlier hybrid Inception version

with significantly improved recognition performance.

- Inception-v4: a pure Inception variant without residual connections with roughly the same recognition performance as Inception-ResNet-v2.

We studied how the introduction of residual connections leads to dramatically improved training speed for the Inception architecture. Our latest models (with and without residual connections) outperform all our previous networks, just by virtue of the increased model size, while keeping the overall number of parameters and computational cost in check compared to competing approaches.

References

Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G. S.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Goodfellow, I.; Harp, A.; Irving, G.; Isard, M.; Jia, Y.; Jozefowicz, R.; Kaiser, L.; Kudlur, M.; Levenberg, J.; Mané, D.; Monga, R.; Moore, S.; Murray, D.; Olah, C.; Schuster, M.; Shlens, J.; Steiner, B.; Sutskever, I.; Talwar, K.; Tucker, P.; Vanhoucke, V.; Vasudevan, V.; Viégas, F.; Vinyals, O.; Warden, P.; Wattenberg, M.; Wicke, M.; Yu, Y.; and Zheng, X. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Dean, J.; Corrado, G.; Monga, R.; Chen, K.; Devin, M.; Mao, M.; Senior, A.; Tucker, P.; Yang, K.; Le, Q. V.; et al. 2012. Large scale distributed deep networks. In *Advances in Neural Information Processing Systems*, 1223–1231.

Dong, C.; Loy, C. C.; He, K.; and Tang, X. 2014. Learning a deep convolutional network for image super-resolution. In *Computer Vision–ECCV 2014*. Springer. 184–199.

Fukushima, K. 1980. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics* 36(4):193–202.

Girshick, R.; Donahue, J.; Darrell, T.; and Malik, J. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*.

Ioffe, S., and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of The 32nd International Conference on Machine Learning*, 448–456.

Karpathy, A.; Toderici, G.; Shetty, S.; Leung, T.; Sukthankar, R.; and Fei-Fei, L. 2014. Large-scale video classification with convolutional neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, 1725–1732. IEEE.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

LeCun, Y.; Boser, B.; Denker, J. S.; Henderson, D.; Howard, R. E.; Hubbard, W.; and Jackel, L. D. 1989. Backpropagation applied to handwritten zip code recognition. *Neural computation* 1(4):541–551.

Lin, M.; Chen, Q.; and Yan, S. 2013. Network in network. *arXiv preprint arXiv:1312.4400*.

Long, J.; Shelhamer, E.; and Darrell, T. 2015. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3431–3440.

Pascanu, R.; Mikolov, T.; and Bengio, Y. 2012. On the difficulty of training recurrent neural networks. *arXiv preprint arXiv:1211.5063*.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. 2014. Imagenet large scale visual recognition challenge.

Simonyan, K., and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Sutskever, I.; Martens, J.; Dahl, G.; and Hinton, G. 2013. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, volume 28, 1139–1147. JMLR Workshop and Conference Proceedings.

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015a. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1–9.

Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; and Wojna, Z. 2015b. Rethinking the inception architecture for computer vision. *arXiv preprint arXiv:1512.00567*.

Tieleman, T., and Hinton, G. Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 4, 2012. Accessed: 2015-11-05.

Toshev, A., and Szegedy, C. 2014. Deeppose: Human pose estimation via deep neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, 1653–1660. IEEE.

Wang, N., and Yeung, D.-Y. 2013. Learning a deep compact image representation for visual tracking. In *Advances in Neural Information Processing Systems*, 809–817.