

Going Deeper with Convolutions

Christian Szegedy1, Wei Liu2, Yangqing Jia1, Pierre Sermanet1, Scott Reed3, Dragomir Anguelov1, Dumitru Erhan1, Vincent Vanhoucke1, Andrew Rabinovich4 1Google Inc. 2University of North Carolina, Chapel Hill 3University of Michigan, Ann Arbor 4Magic Leap Inc.

1fszegedy,jiayq,sermanet,dragomir,dumitru,vanhouckeg@google.com

2wliu@cs.unc.edu, 3reedscott@umich.edu, 4arabinovich@magicleap.com

Abstract

We propose a deep convolutional neural network ar-chitecture codenamed Inception that achieves the new state of the art for classification and detection in the Im-ageNet Large-Scale Visual Recognition Challenge 2014 (ILSVRC14). The main hallmark of this architecture is the improved utilization of the computing resources inside the network. By a carefully crafted design, we increased the depth and width of the network while keeping the compu-tational budget constant. To optimize quality, the architec-tural decisions were based on the Hebbian principle and the intuition of multi-scale processing. One particular in-carnation used in our submission for ILSVRC14 is called GoogLeNet, a 22 layers deep network, the quality of which is assessed in the context of classification and detection.

1. Introduction

In the last three years, our object classification and de-tection capabilities have dramatically improved due to ad-vances in deep learning and convolutional networks [10]. One encouraging news is that most of this progress is not just the result of more powerful hardware, larger datasets and bigger models, but mainly a consequence of new ideas, algorithms and improved network architectures. No new data sources were used, for example, by the top entries in the ILSVRC 2014 competition besides the classification dataset of the same competition for detection purposes. Our GoogLeNet submission to ILSVRC 2014 actually uses 12 times fewer parameters than the winning architecture of Krizhevsky et al [9] from two years ago, while being sig-nificantly more accurate. On the object detection front, the biggest gains have not come from naive application of bigger and bigger deep networks, but from the synergy of deep architectures and classical computer vision, like the R-CNN algorithm by Girshick et al [6].

Another notable factor is that with the ongoing traction of mobile and embedded computing, the efficiency of our algorithms – especially their power and memory use – gains importance. It is noteworthy that the considerations leading to the design of the deep architecture presented in this paper included this factor rather than having a sheer fixation on accuracy numbers. For most of the experiments, the models were designed to keep a computational budget of 1:5 billion multiply-adds at inference time, so that the they do not end up to be a purely academic curiosity, but could be put to real world use, even on large datasets, at a reasonable cost.

In this paper, we will focus on an efficient deep neural network architecture for computer vision, codenamed In-ception, which derives its name from the Network in net-work paper by Lin et al [12] in conjunction with the famous “we need to go deeper” internet meme [1]. In our case, the word “deep” is used in two different meanings: first of all, in the sense that we introduce a new level of organization in the form of the “Inception module” and also in the more direct sense of increased network depth. In general, one can view the Inception model as a logical culmination of [12] while taking inspiration and guidance from the theoretical work by Arora et al [2]. The benefits of the architecture are experimentally verified on the ILSVRC 2014 classification and detection challenges, where it significantly outperforms the current state of the art.

2. Related Work

Starting with LeNet-5 [10], convolutional neural net-works (CNN) have typically had a standard structure – stacked convolutional layers (optionally followed by contrast normalization and max-pooling) are followed by one or more fully-connected layers. Variants of this basic design are prevalent in the image classification literature and have yielded the best results to-date on MNIST, CIFAR and most notably on the ImageNet classification challenge [9, 21]. For larger datasets such as Imagenet, the recent trend has been to increase the number of layers [12] and layer size [21, 14], while using dropout [7] to address the problem of overfitting.

Despite concerns that max-pooling layers result in loss of accurate spatial information, the same convolutional net-work architecture as [9] has also been successfully em-ployed for localization [9, 14], object detection [6, 14, 18, 5] and human pose estimation [19].

Inspired by a neuroscience model of the primate visual cortex, Serre et al. [15] used a series of fixed Gabor filters of different sizes to handle multiple scales. We use a similar strategy here. However, contrary to the fixed 2-layer deep model of [15], all filters in the Inception architecture are learned. Furthermore, Inception layers are repeated many times, leading to a 22-layer deep model in the case of the GoogLeNet model.

Network-in-Network is an approach proposed by Lin et al. [12] in order to increase the representational power of neural networks. In their model, additional 1 1 convolu-tional layers are added to the network, increasing its depth. We use this approach heavily in our architecture. However, in our setting, 1 1 convolutions have dual purpose: most critically, they are used mainly as dimension reduction mod-ules to remove computational bottlenecks, that would oth-erwise limit the size of our networks. This allows for not just increasing the depth, but also the width of our networks without a significant performance penalty.

Finally, the current state of the art for object detection is the Regions with Convolutional Neural Networks (R-CNN) method by Girshick et al. [6]. R-CNN decomposes the over-all detection problem into two subproblems: utilizing low-level cues such as color and texture in order to generate ob-ject location proposals in a category-agnostic fashion and using CNN classifiers to identify object categories at those locations. Such a two stage approach leverages the accu-racy of bounding box segmentation with low-level cues, as well as the highly powerful classification power of state-of-the-art CNNs. We adopted a similar pipeline in our detec-tion submissions, but have explored enhancements in both stages, such as multi-box [5] prediction for higher object bounding box recall, and ensemble approaches for better categorization of bounding box proposals.

3. Motivation and High Level Considerations

The most straightforward way of improving the perfor-mance of deep neural networks is by increasing their size. This includes both increasing the depth – the number of net

work levels – as well as its width: the number of units at each level. This is an easy and safe way of training higher quality models, especially given the availability of a large amount of labeled training data. However, this simple solu-tion comes with two major drawbacks.

Bigger size typically means a larger number of parame-ters, which makes the enlarged network more prone to over-fitting, especially if the number of labeled examples in the training set is limited. This is a major bottleneck as strongly labeled datasets are laborious and expensive to obtain, often requiring expert human raters to distinguish between vari-ous fine-grained visual categories such as those in ImageNet (even in the 1000-class ILSVRC subset) as shown in Fig-ure 1.

The other drawback of uniformly increased network size is the dramatically increased use of computational re-sources. For example, in a deep vision network, if two convolutional layers are chained, any uniform increase in the number of their filters results in a quadratic increase of computation. If the added capacity is used inefficiently (for example, if most weights end up to be close to zero), then much of the computation is wasted. As the computational budget is always finite, an efficient distribution of comput-ing resources is preferred to an indiscriminate increase of size, even when the main objective is to increase the quality of performance.

A fundamental way of solving both of these issues would be to introduce sparsity and replace the fully connected lay-ers by the sparse ones, even inside the convolutions. Be-sides mimicking biological systems, this would also have the advantage of firmer theoretical underpinnings due to the groundbreaking work of Arora et al. [2]. Their main re-sult states that if the probability distribution of the dataset is representable by a large, very sparse deep neural network, then the optimal network topology can be constructed layer after layer by analyzing the correlation statistics of the pre-ceding layer activations and clustering neurons with highly correlated outputs. Although the strict mathematical proof requires very strong conditions, the fact that this statement resonates with the well known Hebbian principle – neurons that fire together, wire together – suggests that the under-lying idea is applicable even under less strict conditions, in practice.

Unfortunately, today’s computing infrastructures are very inefficient when it comes to numerical calculation on non-uniform sparse data structures. Even if the number of arithmetic operations is reduced by 100 , the overhead of lookups and cache misses would dominate: switching to sparse matrices might not pay off. The gap is widened yet further by the use of steadily improving and highly tuned numerical libraries that allow for extremely fast dense ma-trix multiplication, exploiting the minute details of the un-derlying CPU or GPU hardware [16, 9]. Also, non-uniform sparse models require more sophisticated engineering and computing infrastructure. Most current vision oriented ma-chine learning systems utilize sparsity in the spatial domain just by the virtue of employing convolutions. However, con-volutions are implemented as collections of dense connec-tions to the patches in the earlier layer. ConvNets have tra-ditionally used random and sparse connection tables in the feature dimensions since [11] in order to break the sym-metry and improve learning, yet the trend changed back to full connections with [9] in order to further optimize par-allel computation. Current state-of-the-art architectures for computer vision have uniform structure. The large number of filters and greater batch size allows for the efficient use of dense computation.

This raises the question of whether there is any hope for a next, intermediate step: an architecture that makes use of filter-level sparsity, as suggested by the theory, but ex-ploits our current hardware by utilizing computations on dense matrices. The vast literature on sparse matrix com-putations (e.g. [3]) suggests that clustering sparse matrices into relatively dense submatrices tends to give competitive performance for sparse matrix multiplication. It does not seem far-fetched to think that similar methods would be uti-lized for the automated construction of non-uniform deep-learning architectures in the near future.

The Inception architecture started out as a case study for assessing the hypothetical output of a sophisticated network topology construction algorithm that tries to approximate a sparse structure implied by [2] for vision networks and cov-ering the hypothesized outcome by dense, readily available components. Despite being a highly speculative undertak-ing, modest gains were observed early on when compared with reference networks based on [12]. With a bit of tun-ing the gap widened and Inception proved to be especially useful in the context of localization and object detection as the base network for [6] and [5]. Interestingly, while most of the original architectural choices have been questioned and tested thoroughly in separation, they turned out to be close to optimal locally. One must be cautious though: al-though the Inception architecture has become a success for computer vision, it is still questionable whether this can be attributed to the guiding principles that have lead to its con-struction. Making sure of this would require a much more thorough analysis and verification.

4. Architectural Details

The main idea of the Inception architecture is to consider how an optimal local sparse structure of a convolutional vi-sion network can be approximated and covered by readily available dense components. Note that assuming translation invariance means that our network will be built from convo-lutional building blocks. All we need is to find the optimal local construction and to repeat it spatially. Arora et al. [2] suggests a layer-by layer construction where one should an-alyze the correlation statistics of the last layer and cluster them into groups of units with high correlation. These clus-ters form the units of the next layer and are connected to the units in the previous layer. We assume that each unit from an earlier layer corresponds to some region of the in-put image and these units are grouped into filter banks. In the lower layers (the ones close to the input) correlated units would concentrate in local regions. Thus, we would end up with a lot of clusters concentrated in a single region and they can be covered by a layer of 1 1 convolutions in the next layer, as suggested in [12]. However, one can also expect that there will be a smaller number of more spatially spread out clusters that can be covered by convolutions over larger patches, and there will be a decreasing number of patches over larger and larger regions. In order to avoid patch-alignment issues, current incarnations of the Incep-tion architecture are restricted to filter sizes 1 1, 3 3 and 5 5; this decision was based more on convenience rather than necessity. It also means that the suggested architecture is a combination of all those layers with their output filter banks concatenated into a single output vector forming the input of the next stage. Additionally, since pooling opera-tions have been essential for the success of current convo-lutional networks, it suggests that adding an alternative par-allel pooling path in each such stage should have additional beneficial effect, too (see Figure 2(a)).

As these “Inception modules” are stacked on top of each other, their output correlation statistics are bound to vary: as features of higher abstraction are captured by higher lay-ers, their spatial concentration is expected to decrease. This suggests that the ratio of 3 3 and 5 5 convolutions should increase as we move to higher layers.

One big problem with the above modules, at least in this na¨ıve form, is that even a modest number of 5 5 convo-lutions can be prohibitively expensive on top of a convolu-tional layer with a large number of filters. This problem be-comes even more pronounced once pooling units are added to the mix: the number of output filters equals to the number of filters in the previous stage. The merging of output of the pooling layer with outputs of the convolutional lay-ers would lead to an inevitable increase in the number of outputs from stage to stage. While this architecture might cover the optimal sparse structure, it would do it very inef-ficiently, leading to a computational blow up within a few stages.

This leads to the second idea of the Inception architec-ture: judiciously reducing dimension wherever the compu-tational requirements would increase too much otherwise. This is based on the success of embeddings: even low di-mensional embeddings might contain a lot of information about a relatively large image patch. However, embed-dings represent information in a dense, compressed form and compressed information is harder to process. The rep-resentation should be kept sparse at most places (as required by the conditions of [2]) and compress the signals only whenever they have to be aggregated en masse. That is, 1 1 convolutions are used to compute reductions before the expensive 3 3 and 5 5 convolutions. Besides being used as reductions, they also include the use of rectified lin-ear activation making them dual-purpose. The final result is depicted in Figure 2(b).

In general, an Inception network is a network consist-ing of modules of the above type stacked upon each other, with occasional max-pooling layers with stride 2 to halve the resolution of the grid. For technical reasons (memory efficiency during training), it seemed beneficial to start us-ing Inception modules only at higher layers while keeping the lower layers in traditional convolutional fashion. This is not strictly necessary, simply reflecting some infrastructural inefficiencies in our current implementation.

A useful aspect of this architecture is that it allows for increasing the number of units at each stage significantly without an uncontrolled blow-up in computational com-plexity at later stages. This is achieved by the ubiquitous use of dimensionality reduction prior to expensive convolu-tions with larger patch sizes. Furthermore, the design fol-lows the practical intuition that visual information should be processed at various scales and then aggregated so that the next stage can abstract features from the different scales simultaneously.

The improved use of computational resources allows for increasing both the width of each stage as well as the num-ber of stages without getting into computational difficulties. One can utilize the Inception architecture to create slightly inferior, but computationally cheaper versions of it. We have found that all the available knobs and levers allow for a controlled balancing of computational resources resulting in networks that are 3 10 faster than similarly perform-ing networks with non-Inception architecture, however this requires careful manual design at this point.

5. GoogLeNet

By the “GoogLeNet” name we refer to the particular in-carnation of the Inception architecture used in our submis-sion for the ILSVRC 2014 competition. We also used one deeper and wider Inception network with slightly superior quality, but adding it to the ensemble seemed to improve the results only marginally. We omit the details of that network, as empirical evidence suggests that the influence of the ex-act architectural parameters is relatively minor. Table 1 il-lustrates the most common instance of Inception used in the competition. This network (trained with different image-patch sampling methods) was used for 6 out of the 7 models in our ensemble.

All the convolutions, including those inside the Incep-tion modules, use rectified linear activation. The size of the receptive field in our network is 224 224 in the RGB color space with zero mean. “#3 3 reduce” and “#5 5 reduce” stands for the number of 1 1 filters in the reduction layer used before the 3 3 and 5 5 convolutions. One can see the number of 1 1 filters in the projection layer after the built-in max-pooling in the pool proj column. All these re-duction/projection layers use rectified linear activation as well The network is 22 layers deep when counting only layers with parameters (or 27 layers if we also count pooling). The overall number of layers (independent building blocks) used for the construction of the network is about 100. The exact number depends on how layers are counted by the machine learning infrastructure. The use of average pooling before the classifier is based on [12], although our implementation has an additional linear layer. The linear layer enables us to easily adapt our networks to other label sets, however it is used mostly for convenience and we do not expect it to have a major effect. We found that a move from fully connected layers to average pooling improved the top-1 accuracy by about 0.6%, however the use of dropout remained essential even after removing the fully connected layers.

Given relatively large depth of the network, the ability to propagate gradients back through all the layers in an effective manner was a concern. The strong performance of shallower networks on this task suggests that the fea-tures produced by the layers in the middle of the network should be very discriminative. By adding auxiliary classi-fiers connected to these intermediate layers, discrimination in the lower stages in the classifier was expected. This was thought to combat the vanishing gradient problem while providing regularization. These classifiers take the form of smaller convolutional networks put on top of the out-put of the Inception (4a) and (4d) modules. During train-ing, their loss gets added to the total loss of the network with a discount weight (the losses of the auxiliary classi-fiers were weighted by 0.3). At inference time, these auxil-iary networks are discarded. Later control experiments have shown that the effect of the auxiliary networks is relatively minor (around 0.5%) and that it required only one of them to achieve the same effect.

The exact structure of the extra network on the side, in-cluding the auxiliary classifier, is as follows:

* An average pooling layer with 5 5 filter size and stride 3, resulting in an 4 4 512 output for the (4a), and 4 4 528 for the (4d) stage.
* A 1 1 convolution with 128 filters for dimension re-duction and rectified linear activation.
* A fully connected layer with 1024 units and rectified linear activation.
* A dropout layer with 70% ratio of dropped outputs.
* A linear layer with softmax loss as the classifier (pre-dicting the same 1000 classes as the main classifier, but removed at inference time).

A schematic view of the resulting network is depicted in Figure 3.

6. Training Methodology

GoogLeNet networks were trained using the DistBelief [4] distributed machine learning system using mod-est amount of model and data-parallelism. Although we used a CPU based implementation only, a rough estimate suggests that the GoogLeNet network could be trained to convergence using few high-end GPUs within a week, the main limitation being the memory usage. Our training used asynchronous stochastic gradient descent with 0.9 momen-tum [17], fixed learning rate schedule (decreasing the learn-ing rate by 4% every 8 epochs). Polyak averaging [13] was used to create the final model used at inference time.

Image sampling methods have changed substantially over the months leading to the competition, and already converged models were trained on with other options, some-times in conjunction with changed hyperparameters, such as dropout and the learning rate. Therefore, it is hard to give a definitive guidance to the most effective single way to train these networks. To complicate matters further, some of the models were mainly trained on smaller relative crops, others on larger ones, inspired by [8]. Still, one prescription that was verified to work very well after the competition, includes sampling of various sized patches of the im-age whose size is distributed evenly between 8% and 100% of the image area with aspect ratio constrained to the interval . Also, we found that the photometric distortions of Andrew Howard [8] were useful to combat overfitting to the imaging conditions of training data.

1. ILSVRC 2014 Classification Challenge Setup and Results

The ILSVRC 2014 classification challenge involves the task of classifying the image into one of 1000 leaf-node cat-egories in the Imagenet hierarchy. There are about 1.2 mil-lion images for training, 50,000 for validation and 100,000 images for testing. Each image is associated with one ground truth category, and performance is measured based on the highest scoring classifier predictions. Two num-bers are usually reported: the top-1 accuracy rate, which compares the ground truth against the first predicted class, and the top-5 error rate, which compares the ground truth against the first 5 predicted classes: an image is deemed correctly classified if the ground truth is among the top-5, regardless of its rank in them. The challenge uses the top-5 error rate for ranking purposes.

We participated in the challenge with no external data used for training. In addition to the training techniques aforementioned in this paper, we adopted a set of techniques during testing to obtain a higher performance, which we de-scribe next.

1. We independently trained 7 versions of the same GoogLeNet model (including one wider version), and performed ensemble prediction with them. These models were trained with the same initialization (even with the same initial weights, due to an oversight) and learning rate policies. They differed only in sampling methodologies and the randomized input image order.
2. During testing, we adopted a more aggressive cropping approach than that of Krizhevsky et al. [9]. Specif-ically, we resized the image to 4 scales where the shorter dimension (height or width) is 256, 288, 320 and 352 respectively, take the left, center and right square of these resized images (in the case of portrait images, we take the top, center and bottom squares). For each square, we then take the 4 corners and the center 224 224 crop as well as the square resized to 224 224, and their mirrored versions. This leads to 4 3 6 2 = 144 crops per image. A similar ap-proach was used by Andrew Howard [8] in the pre-vious year’s entry, which we empirically verified to perform slightly worse than the proposed scheme. We note that such aggressive cropping may not be neces-sary in real applications, as the benefit of more crops becomes marginal after a reasonable number of crops are present (as we will show later on).
3. The softmax probabilities are averaged over multiple crops and over all the individual classifiers to obtain the final prediction. In our experiments we analyzed alternative approaches on the validation data, such as max pooling over crops and averaging over classifiers, but they lead to inferior performance than the simple averaging.

In the remainder of this paper, we analyze the multiple factors that contribute to the overall performance of the final submission.

Our final submission to the challenge obtains a top-5 er-ror of 6.67% on both the validation and testing data, ranking the first among other participants. This is a 56.5% relative reduction compared to the SuperVision approach in 2012, and about 40% relative reduction compared to the previous year’s best approach (Clarifai), both of which used external data for training the classifiers. Table 2 shows the statistics of some of the top-performing approaches over the past 3 years.

We also analyze and report the performance of multiple testing choices, by varying the number of models and the number of crops used when predicting an image in Table 3. When we use one model, we chose the one with the lowest top-1 error rate on the validation data. All numbers are re-ported on the validation dataset in order to not overfit to the testing data statistics.

1. ILSVRC 2014 Detection Challenge Setup and Results

The ILSVRC detection task is to produce bounding boxes around objects in images among 200 possible classes. Detected objects count as correct if they match the class of the groundtruth and their bounding boxes overlap by at least 50% (using the Jaccard index). Extraneous detections count as false positives and are penalized. Contrary to the classification task, each image may contain many objects or none, and their scale may vary. Results are reported using the mean average precision (mAP). The approach taken by GoogLeNet for detection is similar to the R-CNN by [6], but is augmented with the Inception model as the region classi-fier. Additionally, the region proposal step is improved by combining the selective search [20] approach with multi-box [5] predictions for higher object bounding box recall. In order to reduce the number of false positives, the super-pixel size was increased by 2 . This halves the proposals coming from the selective search algorithm. We added back 200 region proposals coming from multi-box [5] resulting, in total, in about 60% of the proposals used by [6], while increasing the coverage from 92% to 93%. The overall ef-fect of cutting the number of proposals with increased cov-erage is a 1% improvement of the mean average precision for the single model case. Finally, we use an ensemble of 6 GoogLeNets when classifying each region. This leads to an increase in accuracy from 40% to 43.9%. Note that con-trary to R-CNN, we did not use bounding box regression due to lack of time.

We first report the top detection results and show the progress since the first edition of the detection task. Com-pared to the 2013 result, the accuracy has almost doubled. The top performing teams all use convolutional networks. We report the official scores in Table 4 and common strate-gies for each team: the use of external data, ensemble mod-els or contextual models. The external data is typically the ILSVRC12 classification data for pre-training a model that is later refined on the detection data. Some teams also men-tion the use of the localization data. Since a good portion of the localization task bounding boxes are not included in the detection dataset, one can pre-train a general bounding box regressor with this data the same way classification is used for pre-training. The GoogLeNet entry did not use the localization data for pretraining.

In Table 5, we compare results using a single model only. The top performing model is by Deep Insight and surpris-ingly only improves by 0.3 points with an ensemble of 3 models while the GoogLeNet obtains significantly stronger results with the ensemble.

9. Conclusions

Our results yield a solid evidence that approximating the expected optimal sparse structure by readily available dense building blocks is a viable method for improving neural net-works for computer vision. The main advantage of this method is a significant quality gain at a modest increase of computational requirements compared to shallower and narrower architectures.

Our object detection work was competitive despite not utilizing context nor performing bounding box regression, suggesting yet further evidence of the strengths of the Inception architecture. For both classification and detection, it is expected that similar quality of result can be achieved by much more ex-pensive non-Inception-type networks of similar depth and width. Still, our approach yields solid evidence that mov-ing to sparser architectures is feasible and useful idea in general. This suggest future work towards creating sparser and more refined structures in automated ways on the basis of [2], as well as on applying the insights of the Inception architecture to other domains.

References

[1] Know your meme: We need to go deeper.

http://knowyourmeme.com/memes/we-need-to-go-deeper.

Accessed: 2014-09-15.

1. S. Arora, A. Bhaskara, R. Ge, and T. Ma. Provable bounds for learning some deep representations. CoRR, abs/1310.6343, 2013.
2. U. V. C¸atalyurek,¨ C. Aykanat, and B. Uc¸ar. On two-dimensional sparse matrix partitioning: Mod-els, methods, and a recipe. SIAM J. Sci. Comput., 32(2):656–683, Feb. 2010.
3. J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, Q. V. Le, and A. Y. Ng. Large scale distributed deep networks. In P. Bartlett, F. Pereira, C. Burges, L. Bot-tou, and K. Weinberger, editors, NIPS, pages 1232– 1240. 2012.
4. D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov. Scalable object detection using deep neural networks. In CVPR, 2014.
5. R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In Computer Vision and Pattern Recognition, 2014. CVPR 2014. IEEE Confer-ence on, 2014.

[7] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature de-tectors. CoRR, abs/1207.0580, 2012.

1. A. G. Howard. Some improvements on deep con-volutional neural network based image classification. CoRR, abs/1312.5402, 2013.
2. A. Krizhevsky, I. Sutskever, and G. Hinton. Im-agenet classification with deep convolutional neural networks. In Advances in Neural Information Process-ing Systems 25, pages 1106–1114, 2012.
3. Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backprop-agation applied to handwritten zip code recognition. Neural Comput., 1(4):541–551, Dec. 1989.
4. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recog-nition. Proceedings of the IEEE, 86(11):2278–2324, 1998.
5. M. Lin, Q. Chen, and S. Yan. Network in network. CoRR, abs/1312.4400, 2013.
6. B. T. Polyak and A. B. Juditsky. Acceleration of stochastic approximation by averaging. SIAM J. Con-trol Optim., 30(4):838–855, July 1992.
7. P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fer-gus, and Y. LeCun. Overfeat: Integrated recogni-tion, localization and detection using convolutional networks. CoRR, abs/1312.6229, 2013.
8. T. Serre, L. Wolf, S. M. Bileschi, M. Riesenhuber, and T. Poggio. Robust object recognition with cortex-like mechanisms. IEEE Trans. Pattern Anal. Mach. Intell., 29(3):411–426, 2007.
9. F. Song and J. Dongarra. Scaling up matrix computa-tions on shared-memory manycore systems with 1000 cpu cores. In Proceedings of the 28th ACM Interna-tional Conference on Supercomputing, ICS ’14, pages 333–342, New York, NY, USA, 2014. ACM.
10. I. Sutskever, J. Martens, G. E. Dahl, and G. E. Hinton. On the importance of initialization and momentum in deep learning. In ICML, volume 28 of JMLR Proceed-ings, pages 1139–1147. JMLR.org, 2013.
11. C. Szegedy, A. Toshev, and D. Erhan. Deep neural net-works for object detection. In C. J. C. Burges, L. Bot-tou, Z. Ghahramani, and K. Q. Weinberger, editors, NIPS, pages 2553–2561, 2013.
12. A. Toshev and C. Szegedy. Deeppose: Human pose estimation via deep neural networks. CoRR, abs/1312.4659, 2013.
13. K. E. A. van de Sande, J. R. R. Uijlings, T. Gevers, and A. W. M. Smeulders. Segmentation as selective search for object recognition. In Proceedings of the 2011 International Conference on Computer Vision, ICCV ’11, pages 1879–1886, Washington, DC, USA, 2011. IEEE Computer Society.
14. M. D. Zeiler and R. Fergus. Visualizing and under-standing convolutional networks. In D. J. Fleet, T. Pa-jdla, B. Schiele, and T. Tuytelaars, editors, ECCV, volume 8689 of Lecture Notes in Computer Science, pages 818–833. Springer, 2014.

