Compressed Residual-VGG16 CNN Model for Big Data Places Image Recognition

Hussam Qassim
Dept. of Computer Science
California State University
Fullerton, California 92831
hualkassam(at)csu.fullerton.edu

Abhishek Verma
Dept. of Computer Science
New Jersey City University
Jersey City, NJ 07305
averma(at)njcu.edu

David Feinzimer
Dept. of Computer Science
California State University
Fullerton, California 92831
dfeinzimer(at)csu.fullerton.edu

Abstract

Deep learning has given way to a new era of machine learning, apart from computer vision. Convolutional neural networks have been implemented in image classification, segmentation and object detection. Despite recent advancements, we are still in the very early stages and have yet to settle on best practices for network architecture in terms of deep design, small in size and a short training time. In this paper, we address the issue of speed and size by proposing a compressed convolutional neural network model namely Residual Squeeze VGG16. Proposed model compresses the earlier very successful VGG16 network and further improves on following aspects: (1) small model size, (2) faster speed, (3) uses residual learning for faster convergence, better generalization, and solves the issue of degradation, (4) matches the recognition accuracy of the non-compressed model on the very large-scale grand challenge MIT Places 365-Standard scene dataset.

In comparison to VGG16 the proposed model is 88.4% smaller in size and 23.86% faster in the training time. This supports our claim that the proposed model inherits the best aspects of VGG16 and further improves upon it. In comparison to SqueezeNet our proposed framework can be more easily adapted and fully integrated with the residual learning for compressing various other contemporary deep learning convolutional neural network models Broader impact of our work could improve the performance in specialized tasks such as video-based surveillance, self-driving cars, and mobile GPU applications.

Keywords— Convolutional Neural Networks; VGG16; Residual Learning; Squeeze Neural Networks; scene classification.

1 Introduction

Due to recent advancements in high-performing computing systems, GPUs and large distributed clusters [16] along with the availability of large public image repositories like ImageNet, Deng et al. [17] Convolutional networks have seen a lot of research and development interest as of late (Krizhevsky et al. [2]; Zeiler & Fergus [29]; Sermanet et al. [33]; Simonyan & Zisserman [25]). The ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) [32] has served as a platform for multi-generational large scale image classification systems leading to many advancements in deep visual recognition architectures. ILSVRC has seen everything from high-dimensional shallow feature encodings (Perronnin et al. [11]) (winner of ILSVRC-2011) to deep ConvNets (Krizhevsky et al. [2]) (winner of ILSVRC-2012). Since 2012, deep ConvNets have become a focus of the computer vision field with numerous attempts to improve upon the architecture of Krizhevsky et al. [2] to achieve higher accuracy. Top submissions to ILSVRC-2013 (Zeiler & Fergus [29]; Sermanet et al. [33]) called for a smaller receptive window size and smaller stride in the first convolutional layer. Other areas of improvement have been concerned with the training and testing of dense networks over an entire image and on multiple scales (Sermanet et al. [33]; Howard [1]). Simonyan and Zisserman [25] addressed depth in ConvNet architectural design by adding additional convolutional layers, made possible by the use of very small (3 x 3) convolutional filters in all layers, as shown in the Figure 1. As a result, Simonyan and Zisserman [25] developed a significantly more accurate ConvNet architecture which achieved record-breaking results on ILSVRC classification and localization tasks and similar achievements on other image recognition datasets and tasks such as linear SVM feature classification without the benefit of fine-tuning.

With increased network depth, come more problems stemming from degradation. Degradation begins to saturate network accuracy leading to an early failure. Surprisingly, this is not a result of overfitting. Degradation has been shown to be a cause of high training error in [3], [6] when network depth was extended with the addition of more layers. Degradation shows that all neural network models aren't easily and equivalently optimized. Residual learning [22] is a recently developed solution to degradation. Previous work addressed slow convergence, overfitting and degradation by fusing the CNDS network [27] and residual learning connections with shortcuts [22] to build the Residual-CNDS

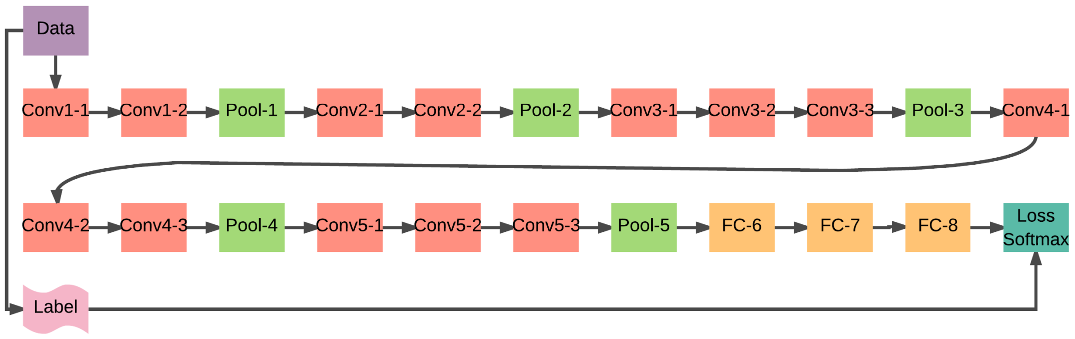

Figure 1: The architecture of VGG16 [26].

[13]. In [13] residual connections are integrated to the basic CNDS [27] eight-layer structure. Experiments [13] showed that a combination of both structures enhances the accuracy of the CNDS network [27].

Late research on deep convolutional neural networks (CNNs) focuses on increasing accuracy on computer vision datasets. Multiple CNN architectures exist that attain any given accuracy level. With a given equivalent accuracy, CNN architectures with a smaller number of parameters may have several advantages:

• *Deployment on FPGA and embedded systems becomes feasible*. Since FPGAs commonly contain 10MB or less of local memory and no remote memory or storage, size is a definite issue. However, a small model can be stored and ran directly on the FPGA rather than being streamed and constrained by bandwidth in real-time [16]. Similarly, on Application-Specific Integrated Circuits (ASICs), a small CNN model can be stored onboard, enabling the ASIC for placement on a smaller die.

• *There is less overhead when exporting new models to client devices in production environments*. In the field of autonomous driving, companies such as Tesla will often distribute updated models from their servers to customers' cars, a method referred to as an over-the-air update. With this, Consumer Reports has noted that the safety and reliability of Tesla's Autopilot semi-autonomous driving features have seen incremental improvements with recent updates [7]. Unfortunately, these over-the-air updates of current CNN/DNN models may require large data transfers. With larger models, such as AlexNet [2], 240MB of data would need to be sent from the server to the car. A smaller model would require less communication, allowing for more frequent update cycles.

• *Compressed models also benefit from more efficient distribution*. Communication between servers limits the scaling of distributed CNN training. In distributed data-parallel training, communication overhead is directly connected to parameter count in the model [21]. Smaller models would complete a distributed training faster.

Therefore, compressed CNN architectures come with several benefits. This brings us to the task of finding a CNN architecture with a reduced parameter count but accuracy equivalent to that of Simonyan and Zisserman's previous model, VGG16 [26], as shown in Figure 1. We propose such an architecture: Residual Squeeze VGG16. We also present a further refined approach to searching for novel CNN architectures. This new model brings many advancements such as being smaller and faster than VGG16 [26].

Additionally, in this paper, we show our state of the art technique to add the residual learning to the compressed model of the VGG16. This prevents the degradation problem from occurring due to compression. Furthermore, proposed model shows excellent optimization in terms of the size and time. Moreover, our adaptable compression method surpasses Iandola et al. [10] both in terms of improving the generalization performance and removing the performance degradation issue, which make us very confident that our framework can be easily adapted for compressing various other contemporary deep learning convolutional neural network models.

Our paper is organized as follows: Section 2 contains a brief background of the VGG network, residual learning and SqueezeNet. Section 3 contains the details of the proposed Residual Squeeze VGG16 model. Section 4 presents details of the large-scale MIT Places365-Standard scene dataset, which we used in our experiments. Section 5 presents our experimental approach. Section 6 contains a discussion of our results and section 7 summarizes our work and provides a brief insight into our planned future work.

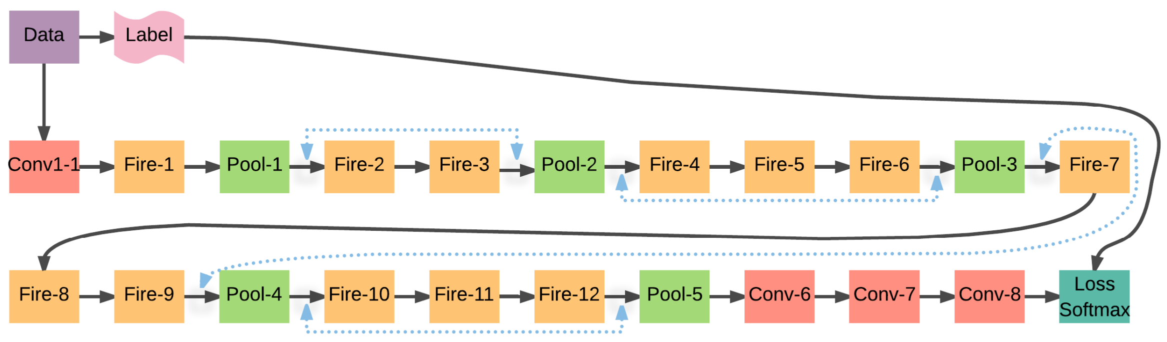

Figure 2: The architecture of proposed Residual Squeeze VGG16 network. Fire modules are shown as orange box and residual connections are shown as dashed blue line.

2 Background

2.1 VGG

The ConvNet architecture of Simonyan and Zisserman [26] contain several differences from the ones in high-performing entries from the ILSVRC-2012 [3] and ILSVRC-2013 (Zeiler & Fergus [29]; Sermanet et al. [32]) competitions. For comparison with our proposed model we show the Simonyan and Zisserman [26] model in Figure 1. First, Simonyan and Zisserman [26] have a very small (3 x 3) receptive field size throughout the entire net, convolved with input at every pixel (with a stride of 1) rather than large receptive fields in the first convolutional layers (e.g. 11 x 11 with a stride of 4 [3] or (7 x 7) with a stride of 2 (Zeiler & Fergus [29]; Sermanet et al. [32]). A stack of two, (3 x 3) convolutional layers (without any intermixed spatial pooling) have an effective receptive field size of (5 x 5). Three of these layers would have a (7 x 7) effective receptive field. This leads to the question of what Simonyan and Zisserman [26] gained through using a stack of (3 x 3) convolutional layers rather than a single (7 x 7) layer. First, Simonyan and Zisserman [26] used three non-linear rectification layers as opposed to one, rendering the resulting decision more discriminative. Second, parameter count was decreased. Assume both the input and output of a three-layer (3 x 3) convolution stack has C channels. The stack is therefore parameterized by $3(3^2C^2) = 27C^2$ weights, and a single (7×7) conv. layer would require $7^2C^2 = 49C^2$ parameters, an increase of 81%. This can be interpreted as imposing a regularization on the (7×7) convolutional filters, resulting in a decomposition through the 3×3 filters (with non-linearity inserted in between). The use of (1 x 1) convolutional layers increases the nonlinearity of the decision function while avoiding a change to the receptive fields of the convolutional layers. In the Simonyan and Zisserman's [26] model, the rectification function introduces an extra non-linearity even though the (1 x 1) convolution is akin to a linear projection onto the same-sized space. It is important to highlight that (1 x 1) convolutional layers were recently used by Lin et al. [30] in "Network in Network" architecture. Ciresan et al. [5] used small-size convolution filters, however, their nets are much shallower than Simonyan and Zisserman [26], and additionally they did not run any test on the large scale ILSVRC dataset. Interestingly, Goodfellow et al. [14] used deep ConvNets with 11 weight layers to recognize street numbers and their results showed a relationship between increased depth and better performance. Another top performer from ILSVRC-2014, GoogLeNet [8], was created independently from Simonyan and Zisserman's [26] work but has similarities because it is based on very deep ConvNets (22 weight layers) and small convolution filters. The network topology of GoogLeNet from Szegedy et al. [8] was more complex than that of Simonyan and Zisserman [26], and spatial resolution is reduced in early layers to decrease computation. Nonetheless, Simonyan and Zisserman's [26] model outperforms GoogLeNet [8] in single-network classification accuracy.

2.2 Residual Learning

Where depth should always result in improved accuracy, degradation will decay optimization. Moreover, error in deeper convolutional neural networks is regularly higher when compared to results of superficial neural networks. He et al. [22] has proposed a degradation resolution which allows a portion of stacked layers to approve the current residual mapping where degradation normally stops layers to fit a required subsidiary mapping. This subsidiary mapping

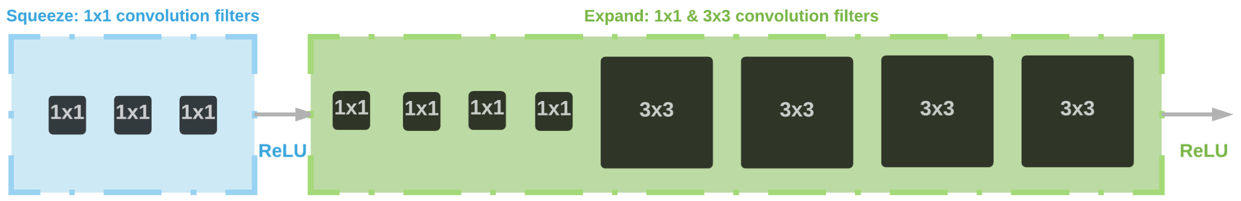
Squeeze: 1x1 convolution filters
Expand: 1x1 & 3x3 convolution filters
1x1
1x1
1x1
ReLU
1x1
1x1
1x1
1x1
3x3
3x3
3x3
3x3
ReLU

Figure 3: Fire module architecture illustrates the convolution filters in the Fire module. In this figure, s1x1 = 3, e1x1 = 4, and e3x3 = 4. (The convolution filters are presented without the activations)

will follow formula (2) instead of formula (1). He et al. [22] found it to be easier to optimize a residual mapping more than a primary mapping.

$$F(x) = H(x) \qquad (1)$$

$$F(x) = H(x) - x \qquad (2)$$

$$F(x) = H(x) + x \qquad (3)$$

In shortcut connections, several layers in a convolutional neural network are skipped [3], [6], [39]. Shortcut links are depicted by formula (3) [22]. He et al. [22] use shortcut connections to perform identity mappings. Output sent by shortcut connections is fused with output sent by stacked layers. Shortcut connections have the advantage of remaining parameter free, only attaching trivial numbers for computation. Shortcut connections combined with gating functions with parameters [36] have been coined, Highway networks [35]. Another advantage of He et al. [22] shortcut connections is the option of optimization through stochastic gradient descent (SGD). Identity shortcut connections can be easily implemented through open source deep learning libraries [1], [28], [34], [40].

2.3 SqueezeNet

Neural network architectures of deep and convolutional backgrounds often leave space for differing arrangements including choices between micro or macro architectures, solvers and an array of hyperparameters. A healthy amount of research work has centered on the development of automated processes for generating network architectures with high levels of accuracy. Some of the more popular processes include Bayesian optimization [18], simulated annealing [38], randomized search [15], and genetic algorithms [24]. These processes achieved improved accuracy when compared with respective baselines. SqueezeNet by Iandola et al. [10] aims to highlight CNN architectures that have a small number of parameters paired with high accuracy. Iandola et al. [1] follow three criteria when generating CNN architectures:

1. (1 x 1) filters will take the place of former (3 x 3) filters
2. The amount of input channels to (3 x 3) filters will see a reduction
3. Downsampling should come later in the network, to provide for large activation maps

Both criteria one and two serve to decrease parameter count while three will ensure maximum accuracy even while working with a limited parameter count.

3 Proposed Residual Squeeze VGG16 Network Architecture

Proposed Residual Squeeze VGG16 contains twelve fire modules [10] and four convolutional layers as shown in figure 2 in comparison to the thirteen convolutional and three fully connected layers of VGG16 [26] as shown in figure 1. We attach a scale layer to all fire modules and the first convolutional layer. The kernel in layer is assigned a stride of two and a size of (3 x 3). Next we replace the second convolutional layer of VGG16 with one fire module. This is done since the fire module has 9x fewer parameters than its (3 x 3) filter equivalent. Input channels are reduced to only (3 x 3) filters. To find the number of parameters in the fire module, we multiply the number of input channels by the number of filters. In order to build CNN architecture with a small parameter count it is important to decrease filter and input channel count. Each Max Pooling layer is assigned a (3 x 3) kernel size following the strategy of downsampling late in a network to give convolution layers' large activation maps [16]. Convolutional layers produce activation maps with spatial resolutions of at least (1 x 1) but often much larger. Height and width of activation maps is determined by a set of two factors: size of the input data and the different choices of layers where downsampling will occur. Szegedy et al. [37]; Simonyan & Zisserman [26]; Krizhevsky et al. [2] have all implemented downsampling in CNN architectures by applying a stride greater than one to a selection of convolution or pooling layers. It has been found that when early layers are given large stride parameters, more layers will have small activation maps. He & Sun [23] observed higher classification accuracy after testing downsampling implemented into four unique CNN architectures.

We use the fire module proposed by Iandola et al. [10] in our: Residual Squeeze VGG16. This module is composed of a squeeze convolution layer with (1 x 1) filters fed to an expanded layer consisting of (1 x 1) and (3 x 3) convolution filters. Figure 3 depicts a typical fire module which contains three tunable dimensions: s1*1, e1*1 and e3*3 [10]. s1*1 [10] represents the count of (1 x 1) filters held in the squeeze layer. e1*1 [10] represents the count of (1 x 1) filters held in the expand layer. e3*3 [10] represents the count of (3 x 3) filters held in the expand layer. We set s1*1 [10] to be less than (e1*1 + e3*3) [10] to limit the count of input channels to the (3 x 3) filters as shown in Table 1.

Formula (12) [22] uses shortcut connections from the residual learning [22] in Residual Squeeze VGG16.

$$y = F(x, \{Wi\}) + x \qquad (4)$$

After an in-depth review of the Squeeze VGG16 neural network, we decided to attach residual learning connections [23] in four locations. Residual connections are attached to locations composed of convolutional layer sequences without any intermediate pooling. Figure 2 shows the described architectures, including residual connections. Element-wise addition links output from Pool1 to output of Fire3. Fire1 has

Table 1: Residual Squeeze VGG16 architectural dimensions.

Layer name/type	s1x1 (#1x1 squeeze)	e1x1 (#1x1 expand)	e3x3 (#3x3 expand)
Fire1	8	32	32
Fire2	16	64	64
Fire3	16	64	64
Fire4	32	128	128
Fire5	32	128	128
Fire6	32	128	128
Fire7	64	256	256
Fire8	64	256	256
Fire9	64	256	256
Fire10	64	256	256
Fire11	64	256	256
Fire12	64	256	256

64 output channels and Fire3 has 128 output channels. We connect a convolutional layer with a 128 size kernel between Pool1 and the element-wise addition layers as a way to equalize the number of output channels.

Next, the second residual connection Pool2 is connected and the shortcut connection crosses over three fire module layers. As a result, the residual connection is attached to the output of Pool2 and Fire6. Fire3 has 128 output channels while Fire6 has 256. A new convolution layer with 256 output channels is added following Pool2, preceding the element-wise addition layer to adjust the output channels of Pool2 and Fire6. The third residual connection connects the output of Pool3 and Fire9. Fire6 has 256 output channels and Fire9 512. A new convolutional layer with 512 output channels is added following Pool3, preceding the element-wise addition layer to adjust the output channels of Pool3 and Fire9. Lastly, the final residual connection fuses the output of Pool4 and Fire12. Fire9 and Fire12 both have 512 output channels. No new convolutional layer is added following Pool4 as the number of output channels between Fire9 and Fire12 is already equalized at 512 each.

4 Image Dataset Description

MIT Computer Science and Artificial Intelligence Laboratory created and maintain the large-scale MIT Places365-Standard [4] image dataset. This dataset outsizes ImageNet (ILSVRC 2016) [32] and SUN dataset [20]. There are 1,803,460 total training images in MIT Places365-Standard [4] dataset with 50 validation classes and 900 test classes of sizes ranging from a low of 3,068 to a high of 5,000. MIT Places365-Standard [4] dataset has classes composed of different scenes, which are images labeled with a place or name. The purpose of this dataset is to assist in the development of innovative computer vision and machine learning techniques that can excel in real world image recognition, scalability, parallelism, and deeper understanding of a very diverse problem domain. Broader impact of designing solutions on this dataset could improve the recognition in specialized tasks such as self-driving cars, medical imaging, video-based surveillance, etc. We benchmark our algorithm on this dataset.

5 Experimental Environment and Approach

During training phase, our proposed network: Residual Squeeze VGG16 was trained from scratch: It is composed of twelve fire modules [10] and four convolution neural layers with four residual connections as opposed to VGG16 [26] with thirteen convolutions and three fully-connected layers.

Table 2: Comparison of the Top 1 & 5 validation classification accuracy (%), duration, size, and number of training epochs between the VGG16 [26] and Proposed Residual Squeeze VGG16 on the MIT Places 365-Standard Dataset [4]

Network	Top-1 Validation %	Top-5 Validation %	Duration	Size	No. of Epoch
VGG16	54	84.3	3 Days 16 Hours	10.6 GB	20
Proposed Residual Squeeze VGG16	51.68	82.04	2 Days 19 Hours	1.23 GB	50

We fined tuned VGG16 on the MIT Places365-Standard dataset [10] from pre-trained model on ImageNet ILSVRC-2014 [32].

To conduct training, we utilize Berkeley Vision and Learning Center's open source deep learning framework, Caffe [40]. We pair Caffe and an open source deep learning GPU training system, NVIDIA DIGITS [31], which allows users to build and examine artificial neural networks with real-time graphical representations. Physical hardware consists of four NVIDIA GeForce GTX TITAN X GPUs and two Intel Xeon processors with 48/24 logical/physical cores and 256GB of main memory. All images in the training and validation set are resized to (256 x 256). Batch size for training is 128 and validation is 64. The epoch attribute is set to 50 and learning rate to 0.01. Every 10 epochs, the learning rate degrades 5x and average decay reduces to ½ of the previous value. In VGG16 [26] the epoch attribute is set to 20 and learning rate to 0.001. After the completion of every 4 epochs, the learning rate degrades 5x and learning average decays to ½ of the previous value. A randomized crop of size (227 x 227) is applied before introduction to the first convolutional layer. We adapt the weights of all layers from the Xavier distribution with a standard deviation of 0.01 as opposed to VGG16 [26] which uses a Gaussian distribution with a 0.01 standard deviation for the weights of each layer. The final convolutional layer of Residual Squeeze VGG16 serves as an output layer with weight adapted from the Gaussian distribution with a 0.01 standard deviation. Data augmentation is performed via reflection.

6 Results and Discussion

This paper ensembles approaches from three popular methods: VGG16 [26], residual learning [22] and the Squeeze technique [10]. We set out to examine whether residual connections can boost VGG16 [26] network effectiveness while simultaneously making the network smaller and faster. We modify the fire module concept [10] and added residual connections at places in the network where they would be most effective. Since the residual connections are parameter free, the network does not see an increase in complexity except for a small amount of computation for the collection process. Furthermore, our network size, training time and complexity was reduced with the help of the fire modules [10] and sees only a marginal top-1 and top-5 accuracy loss. Table 2 compares results between our new network: Residual Squeeze VGG16 and the original VGG16 [26] on top-1 outcome based on the MIT Places 365-Standard dataset [4]. Our new network got a result of 51.68% compared to VGG16's [26] result of 54% (after fine-tuning from ImageNet ILSVRC-2014 [32]), a difference of only 2.32% in top-1 accuracy. Top-5 results are comparable as well: Residual Squeeze VGG16 at 82.04% and VGG16 [26] at 84.3%, a difference of only 2.26%.

VGG16 model [26] took three days and 16 hours to converge with a total size of (10.6 gigabytes). In comparison, the new Residual Squeeze VGG16 took only two days and nineteen hours and a total size of (1.23 gigabytes). This means, our proposed Residual Squeeze VGG16 model is 23.86% faster and 88.4% smaller in size than the original VGG16 [26]. This is excellent saving in terms of space and time with minimal impact on accuracy.

7 Conclusion and Future Work

This paper proposed a Residual Squeeze VGG16 network to address the issue of speed and size. Proposed models compresses the earlier very successful VGG16 network and further improves on following aspects: small model size; faster speed; uses residual learning for faster convergence, better generalization, and solves the issue of degradation; matches the recognition accuracy of the non-compressed model. In comparison to SQUEEZENET our proposed framework can be more easily adapted and fully integrated with the residual learning for compressing various other deep learning convolutional neural network models.

Future work will focus on the application of the techniques we have outlined in this paper to compress other highly-regarded networks including but not limited to ResNet [22] and Densely Connected Convolutional Networks [12]. We hope to achieve similar reductions in size and complexity and match the recognition accuracy.

References

[1] A. G. Howard. Some improvements on deep convolutional neural network based image classification. CoRR, abs/1312.5402, 2013.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, Lake Tahoe, NV, 2012, pp. 1097-1105.

[3] B. D. Ripley, *Pattern Recognition and Neural Networks*. Cambridge university press, 1996.

[4] B. Zhou, A. Khosla, A. Lapedriza, A. Torralba, and A. Oliva. Places: An image database for deep scene understanding. arXiv preprint arXiv:1610.02055, 2016.

[5] D. C. Ciresan et al. Flexible, high performance convolutional neural networks for image classification. *Int. Joint Conf. on Artificial Intelligence*, Barcelona, Spain, 2011.

[6] C. M. Bishop, *Neural Networks for Pattern Recognition. Oxford university press*, 1995.

[7] Consumer Reports. Teslas new autopilot: Better but still needs improvement. http: //www.consumerreports.org/tesla/tesla-new-autopilot-better-but-needs-improvement, 2016.

[8] C. Szegedy et al.. Going deeper with convolutions. *Conf. on Computer Vision and Pattern Recognition*, Boston, MA, 2015.

[9] F. Chollet. “Keras”. GitHub repository, https: //github.com/fchollet/keras, 2015.

[10] F. N. Iandola et al. Squeezenet: alexnet-level accuracy with 50x fewer parameters and << 1mb model size. arXiv preprint arXiv:1602.07360, 2016.

[11] F. Perronnin, J. Sanchez, and T. Mensink. Improving the Fisher kernel for large-scale image classification. In Proc. ECCV, 2010.

[12] G. Huang, Z. Liu, and K. Q. Weinberger. Densely connected convolutional networks. arXiv preprint arXiv:1608.06993, 2016.

[13] H. A. Al-Barazanchi, H. Qassim and A. Verma. Novel CNN architecture with residual learning and deep supervision for large-scale scene image categorization. IEEE 7th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), New York, NY, 2016, pp. 1-7. doi: 10.1109/UEMCON.2016.7777858.

[14] I. J. Goodfellow et al. Multi-digit number recognition from street view imagery using deep convolutional neural networks. arXiv preprint arXiv:1312.6082, 2013.

[15] J. Bergstra and Y. Bengio. An optimization methodology for neural network weights and architectures. JMLR, 2012.

[16] J. Dean et al. Large Scale Distributed Deep Networks. NIPS, 2012.

[17] J. Deng et al.. ImageNet: A Large-Scale Hierarchical Image Database. In CVPR, 2009.

[18] J. Snoek, H. Larochelle, and R.P. Adams. Practical bayesian optimization of machine learning algorithms. In NIPS, 2012.

[19] J. Qiu et al. Going deeper with embedded fpga platform for convolutional neural network. In FPGA, 2016.

[20] J. Xiao et al.. SUN Database: Large-scale Scene Recognition from Abbey to Zoo. Conf. on Computer Vision and Pattern Recognition, San Francisco, CA, 2010.

[21] K. Ashraf et al. Shallow networks for high-accuracy road object-detection. arXiv:1606.01561, 2016.

[22] K. He et al. Deep residual learning for image recognition. *arXiv:1512.03385*, 2015.

[23] K. He and J. Sun. Convolutional neural networks at constrained time cost. *Conf. on Computer Vision and Pattern Recognition*, Boston, MA, 2015.

[24] K.O. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Neurocomputing*, 2002.

[25] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. arXiv preprint arXiv:1406.2199, 2014.

[26] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *Int. Conf. on Learning Representations*, San Diego, CA, 2015.

[27] L. Wang et al.. Training deeper convolutional networks with deep supervision. *arXiv:1505.02496*, 2015.

[28] M. Abadi et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv:1603.04467*, 2016.

[29] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional neural networks. *European Conf. on Computer Vision*, Zurich, Switzerland, 2014.

[30] M. Lin, Q. Chen, and S. Yan. Network in network. CoRR, abs/1312.4400, 2013.

[31] NVIDIA DIGITS Software. (2017). Retrieved April 28, 2017, from https: //developer.nvidia.com/digits.

[32] O. Russakovsky et al. Imagenet large scale visual recognition challenge. *arXiv:1409.0575*, 2014.

[33] P. Sermanet et al. Overfelt: Integrated recognition, localization, and detection using convolutional networks. *Int. Conf. on Learning Representations*, Banff, Canada, 2014.

[34] R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A MATAB-like environment for machine learning. *Conf. on Neural Information Processing Systems: BigLearn Workshop*, Granada, Spain, 2011.

[35] R. K. Srivastava, K. Greff, and J. Schmidhuber. Highway networks. *arXiv:1505.00387*, 2015.

[36] S. Hochreiter and J. Schmidhuber. Long short-term memory. *J. of Neural computation*, 9(8):1735–1780, 1997.

[37] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deepnetwork training by reducing internal covariate shift. *Int. Conf. on Machine Learning*, Lille, France, 2015.

[38] T.B. Ludermir, A. Yamazaki, and C. Zanchettin. An optimization methodology for neural network weights and architectures. *IEEE Trans. Neural Networks*, 17(6):1452-1469, 2006.

[39] W. Venables and B. Ripley, *Modern Applied Statistics with S-Plus*. Springer-Verlag New York, 2002.

[40] Y. Jia et al.. Caffe: Convolutional architecture for fast feature embedding. *arXiv:1408.5093*, 2014.