Residual Squeeze CNDS Deep Learning CNN Model for Very Large Scale Places Image Recognition

Abhishek Verma
Department of Computer Science
New Jersey City University
Jersey City, NJ 07305
av56(at)njit.edu

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

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

***Abstract*— Deep convolutional neural network models have achieved great success in the recent years. However, the optimization of size and the time needed to train a deep network is a research area that needs much improvement. In this paper, we address the issue of speed and size by proposing a compressed convolutional neural network model namely Residual Squeeze CNDS. Proposed models compresses the earlier very successful Residual-CNDS 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 Residual-CNDS the proposed model is 87.64% smaller in size and 13.33% faster in the training time. This supports our claim that the proposed model inherits the best aspects of Residual-CNDS model and further improves upon it.**

Moreover, we present our attempt at a more disciplined approach to searching the design space for novel CNN architectures. 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.

***Keywords*— Convolutional Neural Networks; Convolutional Networks with Deep Supervision; Residual Learning; Residual-CNDS; Squeeze Neural Networks; Residual Squeeze CNDS; scene classification.**

I. INTRODUCTION

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [1] is the current test bed for computer vision algorithms. Convolutional neural networks (CNNs) have achieved breakthroughs in this series of annual competition [4] and also in other image classification tasks [5, 6]. CNN layers learn the images' low, mid, and high level features and classifies [7] in an end-to-end framework. The quality of features' levels can be boosted by the number of layers used in the network. In the ILSVRC contest, it was revealed that the convolutional neural network's accuracy could be improved by increasing the network depth, i.e., number of layers [8, 9]. This shows that the depth of the network is of critical importance. Top results obtained in [8-11] all use very deep convolutional neural networks models on the previous or current version of the ImageNet dataset [1]. The benefits of very deep models extends from regular image classification tasks to other significant recognition challenges such as object detection and segmentation [13-17]. On the other hand, increasing the depth of the network by adding more layers increases the number of parameters, which makes the convergence of back-propagation very slow and prone to overfitting. Furthermore, increasing the depth makes the gradients vulnerable to the issue of vanishing/exploding of gradients [18, 19].

Using the pre-trained weights of shallower networks to initialize the weight of deeper networks was proposed by Simonyan and Zisserman [6]. Szegedy et al. [9] use subsidiary branches attached to the middle layers. These subsidiary branches are auxiliary classifiers. The goal of Szegedy et al. [9] of using these classifiers is to increase the gradients to propagate back through layers of the deep neural network structure. Furthermore, the branches are used to motivate feature maps in the shallower layers to anticipate the labels used at the final layer. However, they did not specify a method that can determine the location of where to add these branches or how to add them. Lee et al. [18] follow similar idea by proposing to add the subsidiary branches after each intermediate layer. The losses from these branches get added with the loss of the final layer. This technique showed an enhancement in the rate of convergence. However, they did not explore the deeply supervised networks (DSN) [18] on very deep networks with many more convolutional layers.

Wang et al. [19] suggested convolutional neural networks with deep supervision (CNDS). They addressed the issue of where to add the auxiliary branches. They explored the issue of vanishing gradients in deep networks to determine which intermediate layer needs to have an auxiliary branch. Adding auxiliary branch addresses the problem of slower convergence and overfitting. Even though the network is now able to start converging; another challenge surfaces, which is the degradation problem. As the depth of the network increases, the degradation problem increases in deeper networks. Degradation issue starts to saturate the accuracy of the

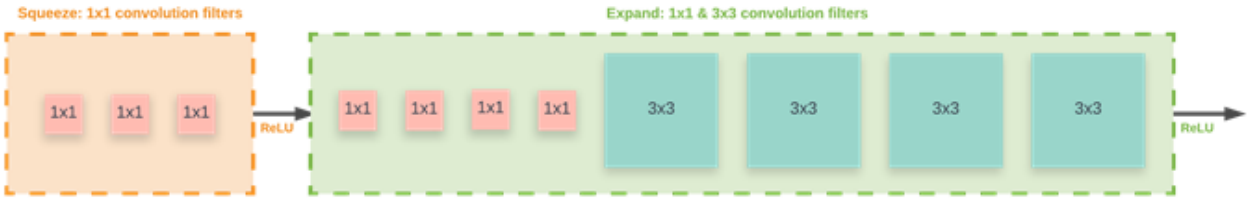

Figure 1: Fire module architecture illustrates the convolution filters in the Fire module. In this figure, s1x1 = 3, e1x1 = 4, and e3x3 = 4. (the convolution filters are presented without the activations)

network and forces it to quickly break down. Surprisingly, overfitting is not the reason behind the degradation problem. Degradation leads to higher training error as reported in [23, 24] when extending the network depth by adding more layers. Moreover, the degradation that happens to the accuracy during the training phase shows that different neural network models are not equivalently easy to optimize. Residual learning [22] is a recently developed technique that solves the issue of degradation. In previous research, the issues of slower convergence, overfitting, and degradation were simultaneously addressed by combining the CNDS network with residual learning [45]. Residual connections were added [22] into the basic CNDS [19] eight layers' structure. Experimental results on the Residual-CNDS network [45] design shows the benefits of combining both structures as it enhances the accuracy upon CNDS network.

Much of the recent research on deep convolutional neural networks (CNNs) has focused on increasing accuracy on computer vision datasets. For a given accuracy level, there typically exist multiple CNN architectures, which achieve that accuracy level. Given equivalent accuracy, a CNN architecture with fewer parameters has several advantages:

- More efficient distributed training. Communication among servers is the limiting factor to the scalability of distributed CNN training. For distributed data-parallel training, communication overhead is directly proportional to the number of parameters in the model [36]. In short, small models train faster due to requiring less communication.
- Less overhead when exporting new models to the clients. For autonomous driving, companies such as Tesla periodically copy new models from their servers to customers' cars. This practice is often referred to as an over-the-air update. Consumer Reports has found that the safety of Tesla's autopilot semi-autonomous driving functionality has incrementally improved with recent over-the-air updates [37]. However, over-the-air updates of today's typical CNN/DNN models can require large data transfers. With AlexNet [38], this would require 240 megabytes of communication from the server to the car. Smaller models require less communication, making frequent updates more feasible.
- Feasible FPGA and embedded deployment. Field-Programmable Gate Arrays (FPGAs) is the state of the art hardware technology for fast processing. FPGA boards often have less than 10MB of on-chip memory and no off-chip memory or storage. For inference, a sufficiently small model could be stored directly on the FPGA instead of being bottlenecked by memory bandwidth [39].

As it can be seen, there are many advantages of smaller CNN architectures. With this in mind, we focus directly on the problem of developing a new CNN architecture with fewer parameters but equivalent accuracy compared to previous model Residual-CNDS [45]. We propose such an architecture, which we call Residual Squeeze CNDS. In addition, we present our attempt at a more disciplined approach to searching the design space for novel CNN architectures. The proposed Residual Squeeze CNDS is a very advanced model, which inherits all the features from the previous Residual-CNDS model [45]. Furthermore, the Residual Squeeze CNDS model is smaller in size and faster than the previous model.

Additionally, in this paper, we show our state of the art technique to add the residual learning to the compressed model of the CNDS. 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. [40] 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.

The remainder of this paper is organized as follows. In section II, we give a brief background of the CNDS network, residual learning and SqueezeNet. We discuss the details of our proposed Residual Squeeze CNDS method in section III. In section IV, we present the details of very large-scale MIT Places365-Standard scene dataset used in our experiments. Section V presents our experimental approach. The discussion

of the results is given in section VI. We conclude the paper and suggest future work in section VII.

II. Background

Ever since ILSVRC 2014 [47], the idea to use very deep artificial neural networks have been recognized as very important. In recent times, progress has been made on finding efficient ways to train very deep neural networks. Part (A) of this section outlines a CNDS network structure and the ways in which their respective authors utilized vanishing gradients in deciding appropriate locations for auxiliary branch placement. Part (B) explores the intricacies of a residual learning mechanism. Part (C) contains a discussion of SqueezeNet [40], which is an attempt at compressing convolutional network namely Alexnet. The following overview and discussion of the CNDS network structure, residual learning mechanism and SqueezeNet [40] should provide a detailed enough description to constitute a good basis for understanding the Residual Squeeze CNDS that we aim to outline in this paper.

A. CNDS Network

Adding auxiliary classifiers which provide further supervision in the training stage improves the generalization of neural networks. Szegedy et al. [7] first proposed this idea through adding subsidiary classifiers, which links to middle layers. However, Szegedy et al. [7] did not explore best location in the network and depth of where to add subsidiary classifiers. Lee et al. [18] proposed an improvement namely Deeply-Supervised Nets (DSN) in which a support vector machine classifier is connected to the output of each hidden layer in a network. From utilizing this method in training, Lee et al. [18] achieved improvements in the sum of the output layer's loss and subsidiary classifiers' losses.

Wang et al. [19] clarified where exactly to add auxiliary classifiers. Deep supervision networks such as the ones proposed by Wang et al. [19] have major distinctions from those proposed by Lee et al [18]. Lee et al. [18] connect the branch classifier in every hidden layer rather than utilizing a gradient focused heuristic in determining where to add an auxiliary classifier. Additionally, Wang et al. [19] implements a small artificial neural network in subsidiary supervision classification. This small network contains one convolutional layer, a small grouping of fully connected layers and one Softmax layer which highly resembles a design introduced by Szegedy et al. [7]. In contrast, Lee et al. [18] use SVM classifiers linked to the outputs of all hidden layers.

To decide where to add auxiliary supervision branches, Wang et al. [19] measured the strength of gradients and the points in the network where it starts to vanish. Wang et al. [19] built the neural network while foregoing the use of supervision classifiers. Weights for this network where adapted from the Gaussian pattern, a mean of zero, standard deviation set to 0.01 and bias set to zero. Wang et al. [19] performed between ten and fifty back-propagation epochs, the mean gradient amount of the shallower layers was controlled by plotting subsidiary supervision classifiers whenever a mean gradient rate would drop under a certain threshold, such as 10^{-7} for instance. Where an average gradient dropped under a predetermined threshold the auxiliary classifier is added to that layer in the network.

B. Residual Learning

Degradation decays optimization in deep convolutional neural networks where as rising depth should always increase accuracy. Additionally, the error from deeper convolutional neural networks is often higher when compared to that of equivalent superficial neural networks. Nonetheless, He at al. [22] proposed a design with a solution to degradation. In this design, He et al. [22] allowed every few stacked layers to qualify the residual mapping whereas degradation stops layers to fit a required subsidiary mapping. To do this subsidiary mapping formula resembles (2) as opposed to formula (1). He et al. [22] assumed it harder to optimize a primary mapping then a residual one.

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

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

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

A shortcut connection is the process in which one or more layers of a convolutional neural network are passed up [23-25]. A shortcut link can be expressed by formula (3) [22]. He et al. [22] use the idea of shortcut connections in order to perform identity mapping. Shortcut connection output is combined with output from the stacked layers. An advantage of shortcut connections is that they remain parameter free and only attach trivial numbers for computation operations, thereby the model complexity in terms of hyper parameters does not increase. Highway networks [21] have shown differences as a result of using shortcut connections in combination with gating functions with parameters [26]. Another advantage of shortcut connections of the type proposed by He et al. [22] is that they can be optimized through stochastic gradient descent (SGD). Finally, identity shortcut connections are easily implemented through open deep learning libraries [27-30].

C. SqueezeNet

Neural network architectures (including those of deep and convolutional denominations) leave a lot of room for choosing different options such as micro/macro architectures, solvers and additional hyper parameters. Consequently, a good amount of work has been concentrated around designing automated ways for creating neural network architectures with

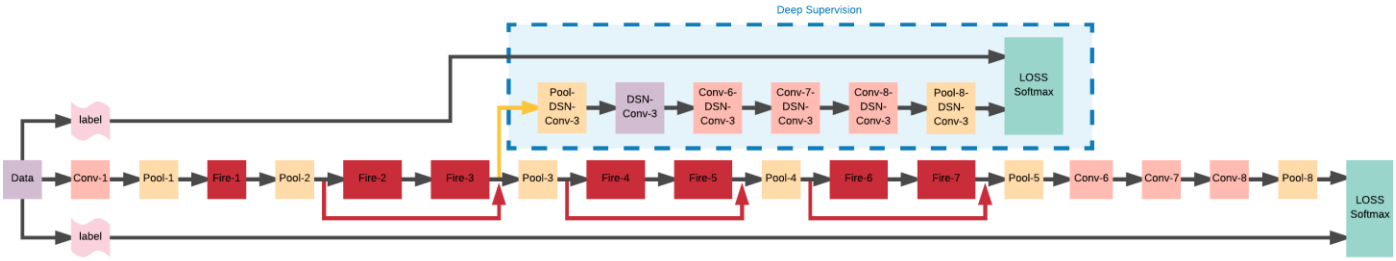

Figure 3: The architecture of proposed Residual Squeeze CNDS network. Fire modules and residual connections are shown in red

a high level of accuracy. Well known automated approaches include Bayesian optimization [41], simulated annealing [42], randomized search [42], and genetic algorithms [44]. Each of these approaches has achieved higher accuracy upon their respective baseline.

The objective of the SqueezeNet proposed by Iandola et al. [40] is to highlight CNN architectures with a small number of parameters and competitive accuracy. The Iandola et al. [40] follow three main strategies in designing CNN architectures:

1. Replace 3x3 filters with 1x1 filters.
2. Decrease the number of input channels to 3x3 filters.
3. Downsample late in the network to give convolutional layers large activation maps.

One and two decrease the number of parameters in CNN while maintaining accuracy. Three would help to maximize accuracy when working with a limited amount of parameters.

III. Proposed Residual Squeeze CNDS Network Architecture

Our proposed Residual Squeeze CNDS contains seven fire modules [40] and four convolutional neural layers in the main branch. Fig. 2 shows the proposed architecture. We attach a scale layer to all fire modules and the first convolutional layer in the main branch. We assign a stride of two and a size of 3x3 to the kernel in layer one. We replace the second convolutional layer of the Residual-CNDS [45] with one fire module [40]. We choose to do this because the fire module [40] has 9x fewer parameters than a 3x3 filter counterpart. Additionally, we reduce input channels to only 3x3 filters. The number of parameters in the fire module is the number of input channels multiplied by the number of filters. Therefore, to maintain a small amount of parameters in a CNN architecture it is important to decrease the number of filters and the number of input channels. We set all max pooling layers a kernel size of 3x3 following the idea of the third strategy [40] in that we downsample late in the network so that convolutional layers can have large activation maps. A convolutional layer will produce an output activation map with a spatial resolution of at least 1x1, but often much larger. The resulting height and width of these activation maps are controlled by two factors: size of the input data and the choice of layers where downsampling will occur. Downsampling has been implemented into CNN architectures by applying a stride greater than one to some convolutional or pooling layers (e.g. (Szegedy et al. [9]; Simonyan & Zisserman [6]; Krizhevsky et al. [2])). When early layers have large stride parameters, as a result, most layers will have small activation maps. However, if most layers have a stride of one and those layers are towards the end of a network, many layers will have large activation map. We observe that large activation maps lead to increased classification accuracy. After He & Sun [46] conducted delayed downsampling tests on four unique CNN architectures they observed higher classification accuracy in each result.

In our Residual Squeeze CNDS model we utilize the fire module first proposed in Iandola et al. [40]. The fire module [40] is a composition of squeeze convolutional layer (with 1 x 1 filters), which is then fed into an expanded layer comprised of a mix of 1x1 and 3x3 convolution filters. The fire module is illustrated in Fig. 1. A fire module has three adjustable dimensions: s_{1*1}, e_{1*1}, and e_{3*3} [40]. s_{1*1} [40] represents the number of 1x1 filters in the squeeze layer. e_{1*1} [40] represents the number of 1x1 filters in the expand layer. e_{3*3} [40] is the number of 3x3 filters in the expand layer. In order to limit the amount of input channels to the 3x3 filters we set s_{1*1} [40] to be less than (e_{1*1} + e_{3*3}) [40] as shown in Table 1.

As observed by Wang et al. [19], the subsidiary branch, which contains the supervision classifier, follows the convolutional neural layer, which experiences the problem of vanishing gradients (Fire3) as shown in the Fig. 2. Characteristic maps that the shallower layers create are noisy and it is very important to minimize this noise in the convolutional layers before it reaches the classifiers. In order to minimize the noise, we decrease the dimensionality of the characteristic maps as in Wang et al. [19] paper, and then pass them into non-linear functions before placing them into classifiers. This results in the subsidiary classifier with an average pooling layer of kernel size 5x5 and a stride of two. Furthermore, a convolutional layer follows the average pooling layer with a kernel of size one and a stride of one, we add a scale layer to it. Then we add two additional convolutional layers in place of fully connected layers, each

TABLE I

RESIDUAL SQUEEZE CNDS MAIN BRANCH ARCHITECTURAL DIMENSIONS

Layer name/type	s1x1 (#1x1 squeeze)	e1x1 (#1x1 expand)	e3x3 (#3x3 expand)
Fire1	16	64	64
Fire2 and 3	32	128	128
Fire4 to 7	64	256	256

with a size of 512 and a kernel of size 3x3, connected by a 0.5 dropout ratio. The master and subsidiary branch have their own output convolutional layer with an output that resembles the amount of classes in the dataset, a kernel of size one, an average pooling layer and a softmax layer for classification.

$$W_{main} = (W_1, \ldots, W_{11}) \quad (4)$$

$$W_{branch} = (W_{s5}, \ldots, W_{s8}) \quad (5)$$

Weights in master branch names are illustrated in formula (4) [19]. These weights match with the eight convolutional layers and three fully connected layers, which resemble those of the original Residual-CNDS model [45]. The eight convolutional layers in [45] are replaced by one convolutional layer and seven fire modules in the proposed Residual Squeeze CNDS. Each fire module (three convolutional layers) replaces one convolutional layer in the original Residual-CNDS [45]. Additionally, the auxiliary classifier's weight computation in formula (5) [19], is matched to the four convolutional layers in the auxiliary branch. If we consider the characteristic map generated from the output layer in the master branch at the beginning to be X_{11} then we are able to calculate likelihood by using the softmax function from the labels k =1, ..., K, illustrated by formula (6) [19]. We can calculate the reply through formula (7) [19] if the characteristic map is S_8, generated from the output layer in the subsidiary branch.

$$pk = \frac{exp(X_{11(k)})}{\sum_k exp(X_{11(k)})} \quad (6)$$

$$psk = \frac{exp(S_{8(k)})}{\sum_k exp(S_{8(k)})} \quad (7)$$

Formula (8) [19] illustrates the loss calculated by the master branch by computing from the probabilities initialized in the softmax. The auxiliary branch loss is calculated using formula (9) [19]. This loss includes weights from the auxiliary branch and the early convolutional layers from the master branch.

$$L_0 (W_{main}) = - \sum_{k=1}^{K} yk \ ln \ pk \quad (8)$$

$$L_s (W_{main}, W_{branch}) = - \sum_{k=1}^{K} yk \ ln \ psk \quad (9)$$

Loss from master and auxiliary branches can be calculated using formula (10) [19]. Formula (10) calculates a weighted sum as the master branch is exposed to additional weight than the subsidiary branch. In order to manage the value of the subsidiary branch as a regularization parameter we use the parameter α_t. This parameter degenerates over sequential iterations as illustrated in formula (11) [19].

$$L_s (W_{main}, W_{branch}) = L_0 (W_{main}) + \alpha_t L_s (W_{main}, W_{branch}) \quad (10)$$

$$\alpha_t = \alpha_t * (1 - t/N) \quad (11)$$

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

$$y = F(x, \{W_i\}) + x \quad (12)$$

Following a deep study of the Squeeze-CNDS neural network we decided to attach residual learning connections [23] to only the master branch. The residual connections are attached to places with sequences of convolutional layers and no pooling in between. Fig. 2 illustrates our architecture showing residual connections in the main branch. The residual connection links input of the Fire2 to output of Fire3. The kernel of Fire1 is 128 and kernel of Fire3 is 256. In order to make the kernels' output equal we use element-wise addition. We also connect a convolutional layer of kernel size 256 between Pool2 and the element-wise addition layers. An element-wise addition links the output of Pool2 to output of (Fire3).

The second residual connection connects to Pool3 and crosses over two Fire module layers. This results in the residual connection being connected between the output of Pool3 and the output of Fire5. The Fire3 has a kernel of size 256 and Fire5 has a kernel of size 512. In order to adjust the size of kernels of Pool3 and Fire5, we insert a convolutional layer with a kernel of size 512 after Pool3 but before the element-wise addition layer. We add the subsidiary branch after the integration process between the output of Pool2 and Fire3. The last third residual connection links the output of Pool4 to the Fire7. Finally, we add a convolutional layer with a kernel of size 512 after Pool4 but before the element-wise addition layer in order to boost the feature mapping at the end of the CNN architecture.

IV. MIT PLACES365-STANDARD IMAGE DATASET DESCRIPTION

MIT Places365-Standard [34] is a very large-scale dataset and perhaps the largest publicly available image dataset. It is created and maintained by MIT Computer Science and Artificial Intelligence Laboratory. It is bigger than ImageNet (ILSVRC2016) [35] and SUN dataset [32]. MIT Places365-Standard [34] dataset has 365 places categories, in total 1,803,460 training images, each class contains anywhere from 3,068 to 5,000 images. It has 50 images per class as validation set and 900 images per class as test set. This dataset is scene

based, meaning it includes images labeled with a scene or place 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.

V. Experimental Environment and Approach

The Residual-CNDS [45] and proposed Residual Squeeze CNDS are trained from scratch. We compare performance of these two networks on the MIT Places dataset. Residual-CNDS [45] contains eight convolutional layers with three residual connections in the main branch and one convolutional layer in the subsidiary branch in contrast the Residual Squeeze CNDS, which has four convolutional layers and seven Fire modules with three residual connections in the main branch, and four convolutional layers in the subsidiary branch. We use Caffe [28], an open source deep learning framework from the Berkeley Vision and Learning Center. In conjunction with Caffe, we use NVIDIA DIGITS an open source deep learning GPU training system [33], which allows users to build and examine their artificial neural networks for object detection and image classification with real-time visualization. As for physical hardware, we operate on four NVIDIA GeForce GTX TITAN X GPUs and two Intel Xeon processors allowing us a total of 48/24 logical/physical cores and 256 GB of main memory.

We resize all images to 256x256 for training, validation and testing. Preprocessing step on each image subtracts from the pixel value the average of the pixel values for each color channel in RGB color space. We set the batch size for the training phase to 256, while we set the batch size of the validation to 128. We set our epoch count to 50, and we set the learning rate to 0.01. Our learning rate degrades 5x during training after every 10 epochs and the decay in the learning rate is half of its previous values. Images are cropped to 227x227 in random areas before being fed into the first convolutional layer. Next, the weights of all layers are adapted from the Xavier distribution with a 0.01 standard deviation. The final convolutional layer, which acts as our output layer has its weight adapted from the Gaussian distribution with a 0.01 standard deviation as well. Data augmentation is performed by reflection.

VI. Results and Discussion

This paper ensembles approaches from three popular methods: convolutional neural networks with deep supervision [19], residual learning [22] and the Squeeze technique [40]. We set out to examine whether residual connections can boost CNDS [19] network effectiveness while simultaneously making the network smaller and faster. To do this we adapted and modified the Fire module concept [40] and added residual connections. We observe that the residual connections are parameter free, even after a trivial amount of computation for the collection process; the networks complexity does not see much increase. Additionally, the fire modules [40] help in the reduction of our network size, training time and complexity of our network with only a small Top-1 and Top-5 accuracy loss. Table II shows that after training from scratch, Residual Squeeze CNDS Top-1 outcome is 51.32% whereas the original Residual-CNDS [45] Top-1 outcome was almost a similar 51.98% on the validation set in the MIT Places 365-Standard dataset [34], a difference of mere 0.66 %. Our Residual Squeeze CNDS Top-5 result is 81.34% very close to the performance of Residual-CNDS [45].

TABLE II

Comparison of the Top 1 & 5 Validation Classification Accuracy (%), Duration and Size between Residual-CNDS [45] and Proposed Residual Squeeze CNDS on the MIT Places 365-Standard Dataset [34]

Network	Top-1 Validation %	Top-5 Validation %	Duration	Size
Residual-CNDS	51.98	82.11	1 Day 21 Hour	14 GB
Proposed Residual Squeeze CNDS	51.32	81.34	1 Day 15 Hour	1.73 GB

Residual-CNDS model [45] took one day and 21 hours to converge with a total size of (14 gigabyte). In comparison, the new Residual Squeeze CNDS took only one day and fifteen hours and a total size of (1.73 gigabyte). This means, our proposed Residual Squeeze CNDS model is 13.33% faster and 87.64% smaller in size that the original Residual-CNDS [45]. This is excellent saving in terms of space and time with minimal impact on accuracy.

VII. Conclusion

Compressed CNN architecture leads to more efficient distributed training, less network overhead when exporting new models to the clients, and feasible FPGA and embedded deployment. This paper proposed a Residual Squeeze CNDS network to address the issue of speed and size. Proposed models compresses the earlier very successful Residual-CNDS 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. Broader impact of our work could improve the performance in specialized tasks such as self-driving cars, video-based surveillance, and mobile GPU applications.

REFERENCES

[1] O. Russakovsky et al., "Imagenet large scale visual recognition challenge," *Int. J. of Computer Vision*, vol. 115, no. 3, pp. 2011-252, 2015.

[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] Y. LeCun et al., "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541-551, 1989.

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

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

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

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

[8] K. He et al., "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *Int. Conf. on Computer Vision*, Santiago, Chile, 2015.

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

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

[11] R. Girshick et al., "Rich feature hierarchies for accurate object detection and semantic segmentation," *Conf. on Computer Vision and Pattern Recognition*, Columbus, OH, 2014.

[12] K. He et al., "Spatial pyramid pooling in deep convolutional networks for visual recognition," *European Conf. on Computer Vision*, Zurich, Switzerland, 2014.

[13] R. Girshick, "Fast R-CNN," *Int. Conf. on Computer Vision*, Santiago, Chile, 2015.

[14] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *Neural Information Processing Systems*, Montreal, Canada, 2015.

[15] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *Conf. on Computer Vision and Pattern Recognition*, Boston, MA, 2015.

[16] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[17] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," *Int. Conf. on Artificial Intelligence and Statistics*, Sardinia, Italy, 2010.

[18] C.-Y. Lee et al., "Deeply supervised nets," *arXiv:1409.5185*, 2014.

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

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

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

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

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

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

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

[26] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *J. of Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

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

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

[29] 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.

[30] F. Chollet. "Keras". GitHub repository, https: //github.com/fchollet/keras, 2015.

[31] B. Zhou et al., "Learning deep features for scene recognition using places database," *Conf. on Neural Information Processing Systems*, Montreal, Canada, 2014.

[32] 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.

[33] NVIDIA DIGITS Software. (2016). Retrieved June 19, 2016, from https: //developer.nvidia.com/digits.

[34] B. Zhou et al., "Places: an image database for deep scene understanding," arXiv, 2016.

[35] O. Russakovsky et al., "ImageNet large scale visual recognition challenge," CoRR, abs/1409.0575, 2014.

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

[37] Consumer Reports. Teslas new autopilot: better but still needs improvement (2016). Retrieved Feb 20, 2016, from http://www.consumerreports.org/tesla/tesla-new-autopilot-better-but-needs-improvement.

[38] 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.

[39] J. Qiu et al., "Going deeper with embedded FPGA platform for convolutional neural network," *Proc. ACM/SIGDA Int. Symposium on Field-Programmable Gate Arrays*, 2016, pp. 26-35.

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

[41] J. Snoek, H. Larochelle, and R.P. Adams, "Practical bayesian optimization of machine learning algorithms," *Neural Information Processing Systems*, Lake Tahoe, CA, 2012.

[42] T.B. Ludermir, A. Yamazaki, and C. Zanchettin, "An optimization methodology for neural network weights and architectures," *IEEE Trans. On Neural Networks*, vol. 17, no. 6, pp. 1452-1459, 2006.

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

[44] K.O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary Computation*, vol. 10, no. 2, pp. 99-127, 2002.

[45] 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.

[46] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on Imagenet classification," *Int. Conf. on Computer Vision*, Santiago, Chile, 2015.