Novel CNN Architecture with Residual Learning and Deep Supervision for Large-Scale Scene Image Categorization

Hussein A. Al-Barazanchi, Hussam Qassim, and Abhishek Verma
Department of Computer Science
California State University
Fullerton, California 92834
Email: {hussein_albarazanchi, hualkassam}(at)csu.fullerton.edu, averma(at)fullerton.edu

***Abstract*— One of the most investigated methods to increase the accuracy of convolutional neural networks (CNN) is by increasing its depth. However, increasing the depth also increases the number of parameters, which makes convergence of back-propagation very slow and prone to overfitting. Convolutional networks with deep supervision (CNDS) add auxiliary branch to addresses the problem of slower convergence and overfitting. However, CNDS does not resolve the issue of degradation, which can be addressed by residual learning. In order to effectively train deep neural networks, in this paper, we propose Residual-CNDS network, which adds residual learning to CNDS. Residual connections are parameter free and add only negligible amount of computation, thereby it has very little impact over complexity of the network. Results of our experiments on very large-scale MIT Places 205 scene dataset support our hypothesis that adding the residual connections to the CNDS will enhance the accuracy of the network. Our experiments show that the proposed network improves upon other recently introduced state of the art networks both in terms of top-1 and top-5 classification accuracy.**

***Keywords*— Convolutional Neural Networks; Convolutional Networks with Deep Supervision; Residual Learning; Residual-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 competition [2] and also in other image classification tasks [3, 4]. CNN layers learn the images' low, mid, and high level features and classifies [5] 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 [6, 7]. This shows that the depth of the network is of critical importance. Top results obtained in [6-9] all use very deep convolutional neural networks models on the ImageNet dataset [10]. The benefits of very deep models can be extended from regular image classification tasks to other significant recognition challenges such as object detection and segmentation [11-15]. 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 also prone to overfitting. Furthermore, increasing the depth makes the gradients vulnerable to the issue of vanishing/exploding of gradients [16, 17].

Using the pre-trained weights of shallower networks to initialize the weight of deeper networks was proposed by Simonyan and Zisserman [6]. The proposed solution is to solve the problem of slower convergence and overfitting. Nevertheless, using this technique to train multiple networks of incremental depth is computationally expensive and may result in difficulty of tuning the parameters. Another technique to overcome this challenge is proposed by Szegedy et al. [7]. Where they used subsidiary branches attached to the middle layers. These subsidiary branches are auxiliary classifiers. The goal of Szegedy et al. [7] of using these classifiers is to increase the gradients to propagate back through layers of the deep neural network structure. Also, 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 are summed 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] with very deep networks.

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 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 [20, 21] when extending the network depth by adding more layers. Moreover,

978-1-5090-1496-5/16/$31.00 ©2016 IEEE

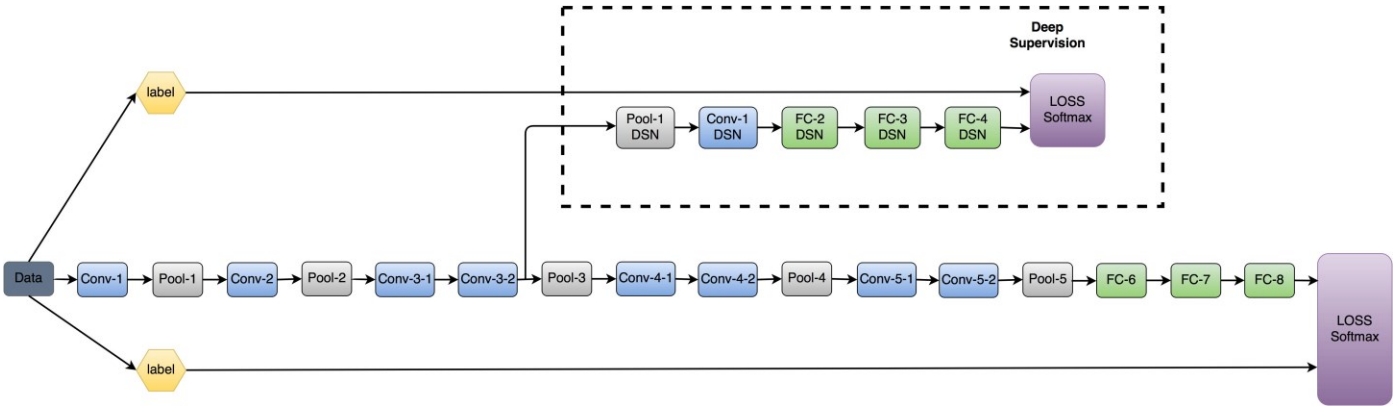

Fig 1. The architecture of convolutional networks with deep supervision (CNDS) [19].

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 our paper, we address issues of overfitting and degradation together by combining the CNDS network with residual learning. We add residual connections [22] into the basic CNDS [19] eight layers structure. Experimental results on our proposed Residual-CNDS network design shows the benefits of combining both structures as it enhances the accuracy upon CNDS network.

The rest of this paper is organized as follows. In section II, we give a brief background of the CNDS network and residual learning. We discuss the details of our proposed Residual-CNDS method in section III. In section IV, we present the details of very large-scale MIT Places 205 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

The idea behind using very deep neural networks first gained prominence in ILSVRC 2014 contest [1] in context of computer vision tasks. As a result, the research on how to effectively train such networks has more recently shown signs of advancement. In this section we explore the architecture of CNDS network and how their authors used vanishing gradients to locate the best position to add the auxiliary branch. Next, we discuss the residual learning technique in section B. The discussion on CNDS and residual learning covers the essential background and serves as a good basis for our proposed Residual-CNDS method, which is explained later in this paper.

A. CNDS Network

Adding auxiliary classifiers that connect to layers in the middle was introduced by Szegedy et al. [7]. These extra classifiers help the generalization of the network by allowing extra supervision in the training phase. However, the procedure discussed in Szegedy et al. [7] does not provide any rules as to where the auxiliary classifiers should be connected and what might be their depth. Thorough analysis of where to insert auxiliary classifier branches is provided in Lee et al. [18]. In their network named deeply-supervised nets (DSN), they insert SVM classifier at the output of every hidden layer. They use this technique only during the training phase. Their optimization is on the sum of the output layer's loss and the losses of auxiliary classifiers.

Wang et al. [19] addressed the issue of where to insert the auxiliary branches in their convolutional neural networks with deep supervision (CNDS). CNDS network [19] has some major differences from the Lee et al. [18] network. The first difference is Lee et al. [18] inserts auxiliary classifier after each hidden layer, whereas Wang et al. [19] use gradient-based heuristic rule to decide where they should connect the supervision branches. The second difference is Wang et al. [19] model uses a small neural network as an auxiliary classifier. This small network consists of convolutional layer, several fully connected layers, and a final softmax layer where it is very similar to [7]. Whereas Lee et al. [16] used SVM classifiers that are connected to the outputs of each hidden layer.

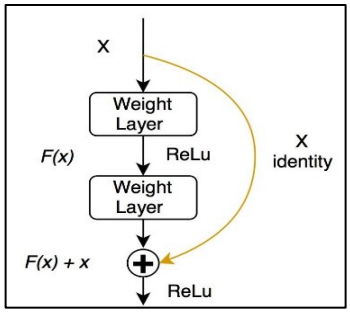

Fig. 2. Residual connection [22].

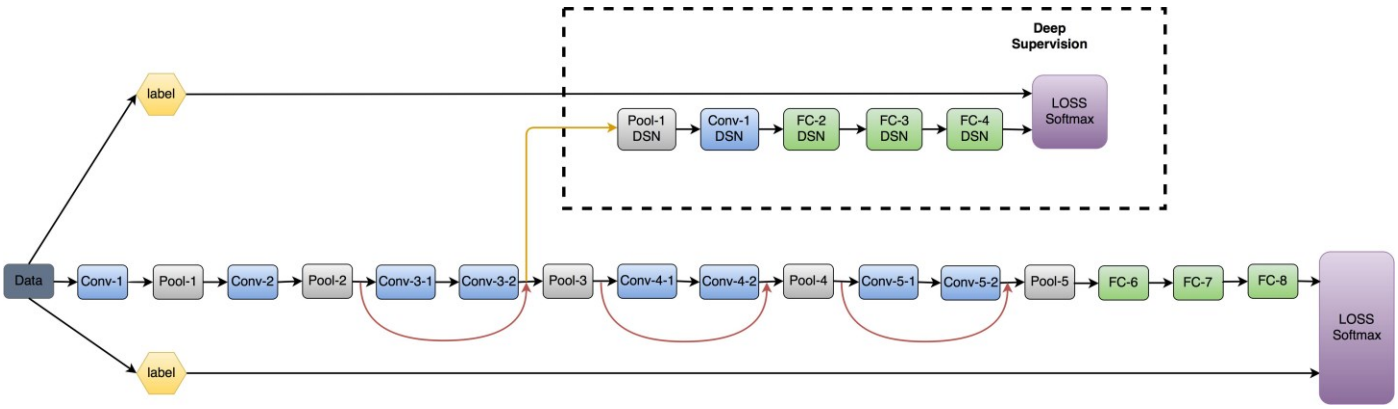

Fig. 3. Proposed CNN architecture with residual learning and deep supervision (Residual-CNDS). Dashed rectangle shows the deep supervision branch and residual connections are labeled as 1, 2, and 3.

Wang et al. [19] built their decision of where to put the auxiliary supervision classifiers (branches) based on vanishing of gradients. First, they designed the network without any branches. The weights were initialized from Gaussian distribution with zero mean, std=0.01, and bias=0. After that, they execute several backpropagation epochs in the range of 10-50 and they monitored the average gradient values of middle layers by plotting them. Then they added the auxiliary classifier branch where the average gradient value degrades, i.e., falls below the threshold of 10^{-7}. In their design, in the fourth convolutional layer the mean gradient falls below the specified threshold. As a result, they added the auxiliary branch after the fourth layer. Fig. 1 shows Wang et al. [19] network design. In order for us to make the comparison easier with our proposed design, in Fig. 1, we follow the naming convention of convolutional layers similar to [6]. The fourth layer is named conv3_2 in Fig. 1.

B. *Residual Learning*

Not all convolutional neural networks are easy to optimize because of the degradation problem. Increasing the depth of the network supposedly increases the resulting accuracy. However, experimental results show a different story; deeper networks produce error higher than their counterpart shallower networks. He et al. [22], solved the issue of degradation by proposing residual learning. They let each of the few stacked layers to fit the *residual mapping*, where degradation prevents the layers to fit the desired underlying mapping. They change the underlying mapping to be as in equation 2 instead of as in equation 1. They hypothesize that it is more difficult to optimize the original mapping than to optimize the residual mapping.

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

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

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

In feedforward neural networks, shortcut connections can be represented as equation 3 [22]. A shortcut connection is the operation of skipping one or more layers in the network [23-25]. Fig. 2 shows how the shortcut connection can be implemented in CNN. As shown in Fig. 2, He et al. [22] used the shortcut connections to implement identity mapping [22]. The output that results from these shortcut connections is summed with the output resulting from the stacked layers as illustrated in Fig. 2, which is represented by equation 3. The benefits of identity shortcut connections are that they are parameter free and add only negligible amount of computation for the summation operation. This is in contrast to "highway networks" [21], where shortcut connections are introduced with gating functions [26] and these gates have parameters. The other benefit of identity shortcut connections in [22] is that they can be optimized by stochastic gradient descent (SGD) algorithm in an end to end manner. Also, they are easy to implement using available deep learning libraries such as [27-30].

III. Proposed Residual-CNDS Network Architecture

The Residual-CNDS network is composed of eight convolutional layers in the main branch. The kernel size of the first layer is 7*7 with a stride of two. While the kernel size for the rest of layers are 3*3 with a stride of one. Based on the rule in Wang et al. [19], the auxiliary supervision classifier is added after the convolutional layer that suffers from the issue of vanishing gradients, which is conv-3-2 in Fig. 3. Feature maps generated from lower convolutional layers are noisy and it is necessary to reduce this noise before feeding it to the classifiers. To reduce the noise, Wang et al. [19] reduced the dimensionality of the feature maps and passed them in non-linear functions before inserting them into the classifiers. Based on that, the auxiliary branch begins with pooling layer (average

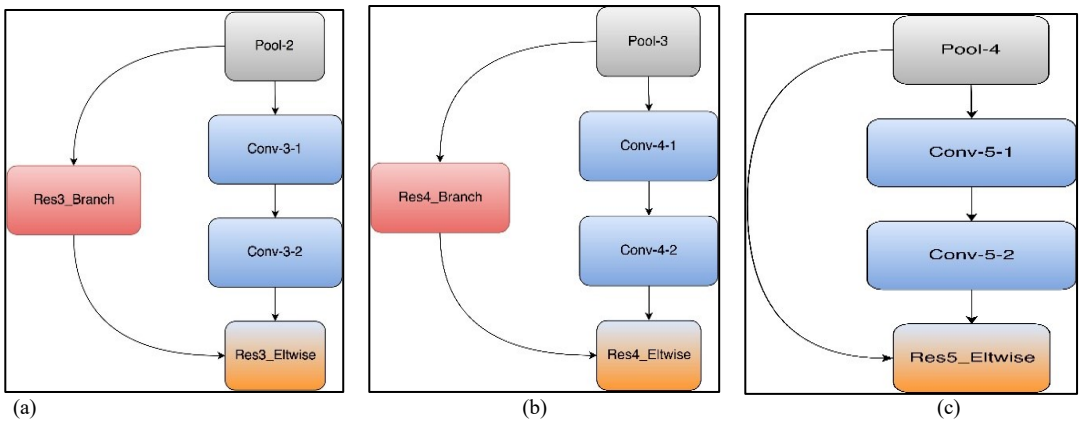

Fig. 4. Details of the residual connections in the proposed Residual-CNDS network.

pooling layer) of size five with a stride of two. After that, there is a convolutional layer of kernel size one and stride one. Next in sequence there are two fully connected layers each of size 1,024 and followed by 0.5 dropout ratio. The main branch contains two fully connected layers of size 4,096, which is followed by 0.5 dropout ratio. The main branch and the auxiliary branch have their own output layer, which is based on softmax layer to compute the probability of classes.

$$W\_main = (W1, \ldots, W11) \quad (4)$$

$$W\_branch = (Ws5, \ldots, Ws8) \quad (5)$$

Naming of weights in main branch is as shown in equation 4 [19]. These weights correspond to the eight convolutional layers and three fully connected layers. On the other hand, the auxiliary branch weights are as shown in equation 5 [19], where the weights correspond to first convolutional layer and three fully connected layers. Firstly, if we consider the feature map produced from output layer in main branch to be X11 then computing the probability using the softmax function for the labels k =1, ..., K is given by equation 6 [19]. Secondly, if features produced from output layer in the auxiliary branch is S8 then computing the response is given by equation 7 [19].

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

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

Computing the loss for the main branch using the probabilities generated from the softmax function is shown in equation 8 [19]. Computing the loss for the auxiliary branch is shown in equation 9 [19]. Computing the loss for the auxiliary branch includes the weights from the auxiliary branch and the weights from earlier convolutional layers in the main branch.

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

$$L_s(W\_main, W\_branch) = -\sum_{k=1}^{K} yk \ln psk \quad (9)$$

The loss from the two branches: main and auxiliary, is combined using the formula shown in equation 10 [19]. This equation computes weighted sum where the main branch is given more weight than the auxiliary branch. The term α_t is used to control the importance of the auxiliary branch as a regularization parameter. α_t decays with successive epochs as shown in equation 11 [19], where N is the total number of epochs.

Fig. 5. Example images from MIT Places 205 scene dataset [31].

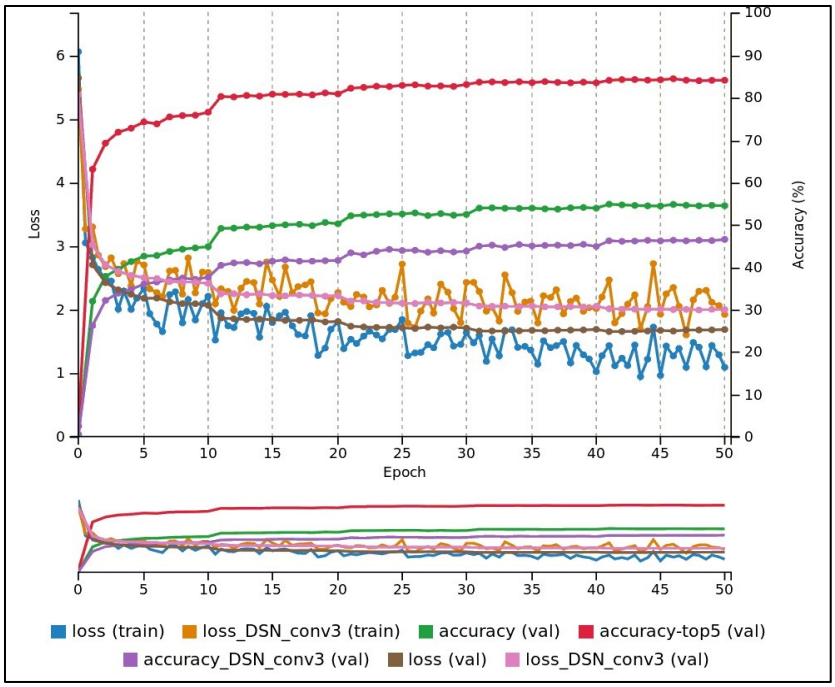

Fig. 6. Training and validation loss. Top 1 and 5 validation accuracy on MIT Places 205 dataset using proposed Residual-CNDS Network.

$$L_s(W\_main, W_{\_branch}) = L_0(W\_main) + \alpha_t L_s(W\_main, W_{\_branch}) \quad (10)$$

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

The shortcut connections from residual learning [22] are used in our model as shown in equation 12 [22].

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

After analyzing the CNDS network, we decided to add the residual learning connections [23] only to the main branch. The places where we can add the residual connections are those positions that have consecutive convolutional layers without pooling layers in between. So, for this reason we cannot add residual connections to the auxiliary branch because it has only one convolutional layer. Fig. 3 shows the resulting architecture after the addition of residual connections to the main branch. Conv., Pool, and FC stand for convolutional, pooling, and fully connected layers respectively. The first residual connection connects the input to the Conv3-1 with the output from Conv3-2 where the element-wise addition connects the output of Pool-2 with output of Conv3-2. The number of kernels in Conv-2 is 128 while the number of kernels in Conv3-2 is 256. To make the kernels' output equal, we perform element-wise addition. We add a convolution layer of 256 kernels between Pool-2 and the element-wise addition layers. This is shown in Figure 4(a); Res3_Branch represents the convolutional layer that is added.

The second residual connection is added after Pool-3 layer and the shortcut connection passes over two convolutional layers. Thus, the residual connection is between the output of Pool-3 and the output of convolutional layer Conv4-2. The number of kernels of Conv3-2 is 256 while the number of kernels of Conv4-2 is 512. Therefore, we add convolutional layer of 512 kernels between Pool-3 and before the element-wise addition layer to make the number of kernels of Pool-3 and Conv4-2 as equal. Fig. 4(b) shows this addition. The auxiliary branch is inserted after the merging operation between the output of Pool-3 and Conv4-2. Finally, the third and the last residual link connects the output of Pool-4 with the last convolutional layer, i.e., Conv5-2. We did not need to add a convolutional layer between Pool-4 and the element-wise addition because the number of kernels of Conv4-2 and Conv5-2 is equal, which is 512 kernels for both of those.

IV. Image Dataset Description

We conduct our experiments on MIT Places 205 [31] dataset. Since, the results using CNDS network are reported on this dataset in [31], it allows us to compare our results using Residual-CNDS with the CNDS network. MIT Places [31] is a very large-scale dataset created by MIT Computer Science and Artificial Intelligence Laboratory, which is bigger than scene challenge competition of ILSVRC [1] and the SUN dataset [32]. The MIT Places is also called Places 205 because it contains 205 scene categories. Places 205 is scene-centric database, which is provided to help the academic experiments and education objectives in the area of computer vision. This dataset contains 2.4 million training images. The number of images for the 205 classes is in the range of 5,000-15,000 per category. The validation set contains 100 images per category

TABLE I
COMPARISON OF THE TOP 1 & 5 VALIDATION AND TEST CLASSIFICATION ACCURACY (%) WITH OTHER PEER REVIEWED PUBLISHED METHODS ON THE MIT PLACES 205 DATASET

Method	Top-1 Validation/Test	Top-5 Validation/Test
AlexNet [31]	- / 50.0	- / 81.1
GoogLeNet [31]	- / 55.5	- / 85.7
CNDS [19]	54.7 / 55.7	84.1 / 85.8
Proposed Method: Residual-CNDS	**54.8 / 56.3**	**84.5 / 86.0**

Note: Data not available is marked as '-'

and the total number of images for the validation set is 20,500. The test set contains 200 images per category and the total number is 41,000 images for testing. Fig. 5 shows four random samples for a few categories from MIT Places dataset.

V. EXPERIMENTAL ENVIRONMENT AND APPROACH

In this work, we trained our model, which contains eight convolutional layers in the main branch with three residual connections and one convolutional layer in the auxiliary branch from scratch, i.e., we did not use pre-trained weights. We use Caffe [28], which is an open source deep learning software framework developed by the Berkley Vision and Learning Center. Caffe plugs in into the NVIDIA DIGITS platform [33], which is a Deep Learning GPU Training System. NVIDIA DIGITS [33] is an open source project that enables the users to design and test their neural networks for image category classification and object detection with real-time visualization. The hardware configuration of our system is four NVIDIA GeForce GTX TITAN X GPUs. The system has two Intel Xeon processors with a total of 48/24 logical/physical cores and 256 GB of main memory.

All images in the training, validation, and testing set are resized to 256*256. The only pre-processing we implemented is subtracting the mean pixel for each color channel of RGB color space. We set the batch size for training to be 256 and for validation to be 128. We set the number of epochs to be 50 and the initial learning rate as 0.01. The learning rate is decreased five times during the training process after every 10th epoch. We set the decay of the learning rate to half of the previous value. Images are cropped to 227*227 from random points before feeding them to the first convolutional layer. All layers' weights are initialized from Gaussian distribution with 0.01 as standard deviation. The only image augmentation that we used is image reflection for the training data.

We trained our Residual-CNDS network on the Places 205 with 2.4 million training images from 205 scene classes, with 5,000-15,000 images per class. We validated our model with 100 images per class. The training process using the aforementioned setup took two days and 14 hours. Fig. 6 shows the loss and the accuracy of our model. We report top-1 and top-5 accuracy on the validation and test dataset. We selected the epoch with highest validation accuracy to be used with the test set. In our case epoch 46 gave the highest validation accuracy. Also, the auxiliary branch is removed during testing and it is only used during the training phase. For testing, we used the average of 10-crops technique, which is shown to improve performance upon other testing techniques [2]. For the test set, the class labels are not available. Hence, we submitted our predictions to the MIT Places server to obtain the test results, which are discussed in section VI.

VI. RESULTS AND DISCUSSION

In this paper, we have worked on merging two powerful ideas, which are convolutional neural networks with deep supervision [19] and residual learning [22] to train deep neural networks. Purpose of adding residual connections to the CNDS network was to investigate if these shortcut connections could enhance the performance of the original network. Residual

TABLE II
TOP 1 CLASSIFICATION ACCURACY FOR 50 BEST CATEGORIES ON THE MIT PLACES 205 DATASET USING PROPOSED RESIDUAL-CNDS

Class	%	Class	%	Class	%	Class	%	Class	%
outdoor	96	pulpit	88	nursery	83	alley	79	outdoor	76
runway	95	raft	88	igloo	82	rock_arch	79	amphitheater	75
cockpit	94	phone_booth	87	coral_reef	82	aquarium	78	corridor	75
wind_farm	94	ballroom	86	bookstore	81	closet	78	gas_station	75
bus_interior	92	music_studio	86	bowling_alley	81	crosswalk	78	auditorium	73
fire_escape	92	playground	86	campsite	81	dam	78	cemetery	73
iceberg	91	shower	86	laundromat	81	water_tower	78	engine_room	73
lighthouse	91	bamboo_forest	85	rice_paddy	81	airport_terminal	77	sea_cliff	73
football	91	racecourse	85	fire_station	80	hotel_room	77	shoe_shop	73
boxing_ring	89	badlands	83	pagoda	80	martial_arts_gym	77	supermarket	73

connections are parameter free and add only negligible amount of computation for the summation operation and thereby have a very little impact over complexity of the network. Experiments on the MIT Places 205 dataset support our hypothesis that adding the residual connections to the CNDS will enhance the accuracy of the network.

Table I shows that top-1 results on the proposed Residual-CNDS network surpass the CNDS [18] by 0.1% and 0.6% on validation and test set respectively. Furthermore, our top-5 results surpass the CNDS [18] by 0.4% and 0.2% on validation and test set respectively. Additionally, our model improves upon GoogLeNet [7] and AlexNet [2] that was tested by the MIT team [31] on Places 205. Aforementioned gains are valuable when considering the big challenge that this dataset poses. Table II, shows the top-1 accuracy for 50 best categories using Residual-CNDS. Mean average accuracy for 50 best categories comes to 82.3%.

VII. Conclusion and Future Work

In this paper we proposed Residual-CNDS network, which adds residual learning to CNDS network. Our experiments using very large-scale image dataset show that the proposed network improves upon other recently proposed state of the art networks both in terms of top-1 and top-5 accuracy. For future work, we plan to investigate the effect of residual connections on other popular networks such as VGG16 and AlexNet.

References

[1] J. Deng et al., "Imagenet large scale visual recognition competition 2012 (ilsvrc2012)," 2012.

[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. "Overfeat: 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 preprint 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 preprint arXiv:1409.5185, 2014.

[19] L. Wang et al., "Training deeper convolutional networks with deep supervision," arXiv preprint 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 preprint 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 preprint arXiv:1603.04467, 2016.

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

[29] R. Collobert, K. Kavukcuoglu, and C. Farabet, "Torch7: A MATLAB-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. (2015). Retrieved April 23, 2016, from https: //developer.nvidia.com/digits.