Exploring New Methods to Improve Existing Deep Neural Networks

Cory Neville
cneville@wpi.edu

Ramesh Doddaiah
rdoddaiah@wpi.edu

Ozan Akylidiz
oakyildiz@wpi.edu

Chinmay Burgul
cmburgul@wpi.edu

***Abstract*—Although steeped in history, neural networks are again a topic of interest for many researchers. Due to several advances in computational ability and increases in parallelization the advent of multi-layer (deep) networks has, in a short time, advanced the state-of-the-art in several domains - particularly image processing. In recent years, researchers have probed into why neural networks work as they do and explored many modifications to the vanilla neural network design architecture in order to push the envelope of recently achieved state-of-the-art. In our work, we present four investigations into deep neural networks that aim to improve neural network testing accuracy using a variety of methods. First, a comparison between shallow and deep ResNets is done. Next, ideas from DenseNet and ResNet are combined to create a novel hybrid Dense-Res block model. In the third approach, transfer learning is performed on existing high performing networks in order to create an ensemble network model. Finally, Deep Supervision, a more direct training of a networks hidden layers, is inspected for its contribution to networks' training time and testing accuracy. We utilize the CIFAR-100 data set to perform these investigations and report our results in the form of test accuracy on this data set.**

I. Introduction

Deep neural networks are modern computational systems that have proven their worth in multiple domains. Machine learning and shallow neural networks have been popular among researchers in the past, however, the current resurgence of popularity gave rise to very deep and complex networks - having possibly billions of trained parameters. Modern research into these networks has exploded into many facets, including analysis into why certain architectures work better than others, exploring network architectures solely to achieve better generalization, and real-time training using online learning in the form of reinforcement learning to name a few.

In our work we explore deep neural networks through four investigations utilizing the CIFAR-100 data set [1] as a common baseline. The rest of the paper is organized as follows. The rest of this section will discuss the contribution of each individual investigation. Section II discusses similar works. Section III further discusses the investigation methodology in detail. Section IV describes the experiment environments. We present out results in section V. We discuss limitations of our works in Section VI. Finally, our conclusions are presented in section VII.

A. Research Contributions

We present four investigations into deep neural networks:

1) Shallow ResNets: Shallow ResNets take the approach of not going deep to get better accuracy by instead expanding the number of neurons in each layer exponentially. With more neurons per layer, the network learns better and without the vanishing and exploding gradient problems seen in deep networks.

2) Dense-Res Blocks: As deeper neural networks were getting more difficult to train due to gradient loss, ResNet [2], DenseNet [3] and other types of skip connections were considered a breakthrough; providing better results for deep networks. Each type of skip connection approach has some advantages and disadvantages which are not yet mathematically proved. So, we tried different approaches to combine this two blocks and study the results.

3) Ensemble Network Model: Ensemble techniques aim to utilize multiple classifiers in parallel to perform as a classifier that is able to make predictions better than any individual classifier making up the ensemble. The ensemble technique presented modifies the CIFAR data input to incorporate the predictions from the classifiers making up the ensemble in the form of additional depth-wise feature planes.

4) Deep Supervision: While the advancing hardware and parallel computing technology made training networks with millions of parameters [4] feasible, a spin-off topic was inspecting the hidden layers, and potential training techniques for them. Deep supervision aims to add secondary goal functions to improve training time, overcome vanishing and exploding gradients and add secondary objectives to the classifier. In this approach we seek to augment VGG16 [5] with deep supervision blocks.

II. Related Work

Much research has been done in order to understand deep neural networks. Recently many modern networks utilize certain architecture wiring [2][3] in order to coarse the network weights to be training efficiently and to avoid common problems such as exploding gradients. Additionally, novel block-style memories [6] and processing modules have been discovered to aid in training [7]. We explore modifications to these architectures. Ensemble methods, on the other hand, are commonly used throughout the data science community. Recently there has been increased interest in exploring utilize this approach for deep networks as well [8]. Deep supervision was initially proposed for improving categorical classification[9][10], but recent methods are used more in feature

extraction such as edge detection [11][12][13] and arguably for attention models.

III. Proposed Method

A. Shallow ResNets

In the first approach, we decided to increase the channel size per convolution layer. Increasing the channel size will increase the number of neurons per layer, which should ideally help the network learn more without increasing the depth of network. In each layer the number of channel sizes was increased exponentially. This should also help solve the vanishing and exploding gradient issues, as they show up with deep networks. In our case, we expand the layer channel size and we were not much concerned with depth of the networks. Huge channel sizes help to learn the correct gradients to improve the network accuracy. The basic network design is as shown in Fig. 1. We also tried to increase the channel size randomly but that did not give good accuracies. It appears that increasing the channel size is the best way to get higher accuracy as per our trials. We also apply dropouts in our networks, unlike the original ResNet and down scale the image by half across the block of layers and increasing the stride size.

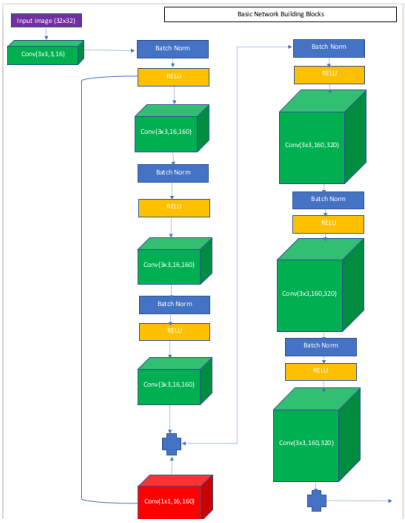


Fig. 1: Shallow ResNet - Basic Network Building Block

In the second approach, we have multiple convolutional layers per layer, original ResNet used short connection between two adjacent convolutional networks. We decided to change it and make it long, such that, the long connections (or circuit) involves N number of convolutional layers. In our case we have long connections across 3 to 4 convolution layers, we continue to use convolutional layer 1-by-1 for connections.

In the final approach we also decided to use dropout unlike in original ResNet. As we all know using drop outs creates additional network paths for better learning, we decided to carefully use dropout only across few layers so that more links or branches could be created when the gradients are high.

B. DenseResNet Blocks

Approach 1: The approaches followed was to combine two types of skip connection models and see if the combined model performs better than the individual model. In approach 1 and 2 ResNet version 1 and DenseNet model were combined and a name was coined for it as DenseResNet. This model architecture is a DenseNet block and ResNet block run in parallel and concatenated at the end instead of concatenating with input and followed by average pooling and the flattening of layers to classify the classes. Typically, a DenseNet model has 5 DenseNet blocks and a ResNet model has 3 ResNet blocks. Each DenseNet block has 12 layers and each ResNet block has 3 ResNet layers i.e 6 layers. In the approach followed, the model architecture has a single DenseNet block and a single ResNet block. This approach is shown in Fig. 2.

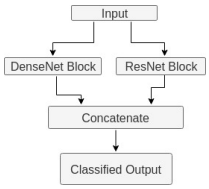


Fig. 2: DenseResNet Approach-1 Block Diagram

Approach 2: In this approach the same the same model was followed, but at the end output of DenseNet block was concatenated with ResNet block and Input in order to check if this might increase the information flow and improve the results. The block diagram is shown in Fig. 3a.

Approach 3: In this approach two DenseNet block were used and each block's output was concatenated with input. The increase in depth this time has decreased the testing accuracy and decreased the over-fitting. The block diagram is shown in Fig. 3b.

C. Ensemble Network Model

The ensemble network injects multiple complex network predictions into a new network which combines these predictions and the CIFAR training data to perform predictions. Although the number of ensemble members has no hard limit, two networks were chosen and are discussed further in the experiments section. In order to separate the training of the ensemble members from training the ensemble itself the CIFAR data set was split 50/50 into two data sets. The first

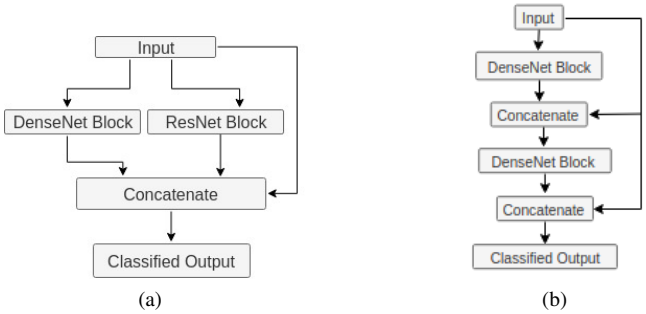


Fig. 3: **(a)** DenseResNet Approach-2 Block Diagram **(b)** DenseResNet Approach-3 Block Diagram

data set was used to train the individual ensemble members, while the second was used to train the ensemble.

After training of the ensemble members each member recorded it's predictions for second data set. These predictions, combined with the raw CIFAR data are used as input to the ensemble network. This is referred to as the modified CIFAR data set. The modified CIFAR data set injects the ensemble predictions via additional depth-wise feature channels that are fully populated with the value of the prediction from that ensemble member. For example, for the two-member ensemble presented in this paper the original 32x32 RGB CIFAR data set is transformed from 32x32x3 to 32x32x5 where the two additional channels are the predictions from the ensemble members. This is shown visually in Fig. 4.

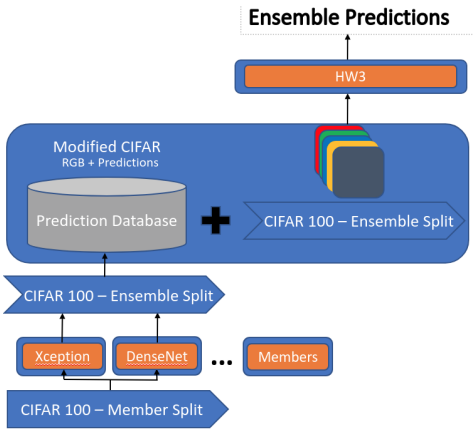


Fig. 4: Architecture of the ensemble network training

D. Deep Supervision

Methods regarding pre-configuring network layers is discussed previously to deep supervision. However, similarly to data augmentation, these are separate and usually time consuming processes[10]. Deep supervision essentially modifies the loss function for a given layer during the training phase. [9] simply proposes a multi-class SVM branching from hidden layers to compare against actual labels. Upon inspection, the implementation consists of inner-product layers, which are Caffe's Fully Connected layers, followed by a squared hinge loss. [10], applies a convolutional block follwed by fully connected layers, which is incorporated into the hidden layer's loss function. These branches are not used during validation.

IV. Experiments

Each investigation has its own experimental setup.

A. Shallow ResNets

We used PyTorch's [14] tools for loading, downloading, and interpreting CIFAR-100 data sets. PyTorch helps to load, normalize and enumerate the CIFAR-100 data sets. PyTorch's basic neural network modules are used as building block to build these shallow ResNets. The first layer takes CIFAR-100's image 32x32x3 size and uses convnets to get 16 channel outputs. This will be the first input data set for the first shallow net layer. Each shallow net layer is made up of multiple convolution networks, which is configurable, long connections are used across multiple convolutions networks, which can also be configurable. Long connections are used across first input data set and 1-by-1 convnets in each group, once the long connection is formed within a single shallow layer, we can further add multiple convnets but without any 1-by-1 convnets. Multiple groups of such shallow networks are also configurable. Ideally we can have N number of convnets within single shallow layer and $M < N$ convents within these single shallow networks can be short circuited, which we call long connection. Across each shallow layers we exponentially increase the neurons within each ConvoNet and also down sample the image linearly. Each layer also adds dropout, which is also configurable. We had 100 epochs across each training and validation data sets. at the end of 100 epochs we apply the learned weights for testing data set to get final accuracy.

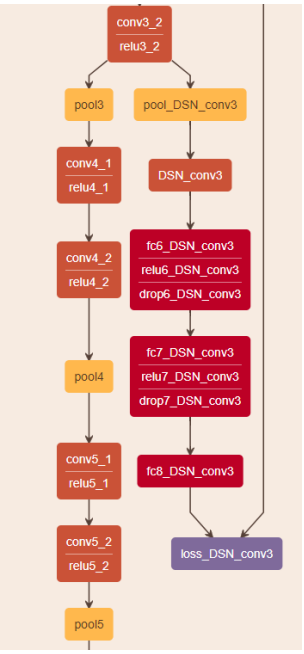
B. Dense-Res Blocks

The combination blocks were designed from existing DenseNet and ResNet models implemented with Keras and imported via Tensorflow. DenseNet maintained the image dimensionality across all the layers and ResNet shrinks the image dimensionality by factor of 2 across each block. DenseNet keeps on increasing the number of channel across each layers and ResNet doubles the number of channels every consecutive block. To combine both of this models we need to concatenate or add the inputs of same image dimensions. In the first approaches mentioned the output of first block of ResNet was concatenated with the output of DenseNet block. Only the first block of ResNet was used because it maintains the image dimensionality. In the second approach input of the model was also concatenated. In third approach, two dense blocks were used and the input of the model was concatenated at the output of the Dense Blocks.

C. Ensemble Network Model

The ensemble network is constructed from existing Keras [15] applications imported via Tensorflow [16]. Two models were chosen: Xception [17] and DenseNet. Xception flips the order of operations for the Inception module by performing a point-wise convolution followed by depth-wise - resulting in a process similar to depth-wise separable kernels. DenseNet blocks provide feed-forward feature maps to each higher layer in the network. These applications are known as the ensemble members.

The ensemble members were loaded with their pre-trained ImageNet [18] weights. Training was split into two passes. First, a new top layer for the CIFAR classifications was added and the rest of the network weights were frozen. This setup was trained for 250 epochs. Afterwards an additional 100 top layer weights were unfrozen and again the network was trained for another 250 epochs. Once fully trained, the remaining CIFAR data set was run through the trained networks and their predictions were recorded in a prediction database for that particular ensemble member.



(a)

(b)

Fig. 5: **(a)** Deep Supervision block visualized using Netscope. https://ethereon.github.io/netscope **(b)** Validation accuracy of 16 layer VGGNet.

The ensemble model consists of a convolution layers, followed by max pooling and dropout. This is repeated twice before flattening into a dense layer and a final dropout before classification. This model was trained twice, once without the addition of the ensemble member feature planes. That is, this network was also trained as-is on the unmodified CIFAR 32x32x3 input data; this unmodified network is called the base network (HW3). This results in four networks: two ensemble members, the base network, and the ensemble network. The base model and the ensemble model were trained for 1000 epochs. All models used categorical cross-entropy loss, a batch size of 128, and a validation split of 20%.

D. Deep Supervision

The proposed methodology consists of adapting Keras' functional API to implement VGGNet16 for CIFAR-100, instead of the original ImageNet challenge with 1000 classes and 254x254 images). Then deep supervision blocks from [9] and [10] is applied to the VGGnet, at each convolutional block. A block here refers to the set of consecutive layers where filter number stays constant followed by 2D max-pooling. Training loss and over same amount of epochs for VGG16, VGG-DSN and VGG-CNDS is used to present the training behaviour, while the validation accuracy **??** for each signifies the performance improvement. More importantly, the experiment aims to provide a framework to test different architectures for on a tested and straightforward yet effective convolutional network.

V. Results

A. Shallow ResNets

Fig.6 shows result of shallow versus original ResNet architecture. We can see that both do good in training around $25 * 100$ epochs; validation too catches up bit later. Both ResNet and shallow networks are doing good but in case of shallow networks, the number of parameters is too low compared the deep ResNets, which will help reduce the time for training and also get similar performances.

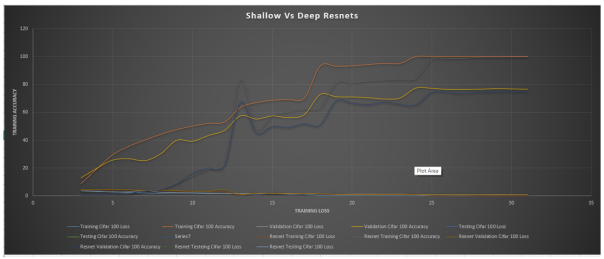


Fig. 6: Shallow Vs Deep ResNets

More clear charts are provided in Fig 7 with final testing accuracy. Our implementation of ResNet provides around 75% accuracy. Current state-of-the-art accuracy for ResNets are 90%+. We got around 67% accuracy with initial shallow ResNet. It appears that most of the lost accuracy was due to dynamic long connections or circuits. Once we decided to use original short connections but still exponentially increase the number of channels per layer we got close to 90% testing accuracy after running it for roughly 24 hours.

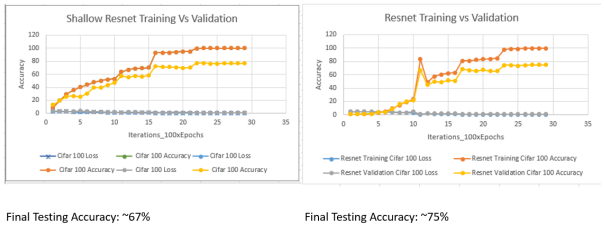


Fig. 7: Testing accuracy shallow network

B. Dense-Res Blocks

The Results of Approach 1 with 75 epochs is Training Accuracy : 96.26%, Testing Accuracy : 77.57%, Validation Accuracy : 77.57% as shown in Fig. 8a

The Results of Approach 2 with 75 epochs is Training Accuracy : 96.91%, Testing Accuracy : 76.13%, Validation Accuracy : 77.36% as shown in Fig. 8b

The Results of Approach 3 with 75 epochs is Training Accuracy : 97.90%, Testing Accuracy : 79.0 %, Validation Accuracy : 80.50% as shown in Fig. 8c

C. Ensemble Network Model

Transfer learning for the ensemble members provided a training accuracy of high 90%. While the base layer achieved low 90% training accuracy and the ensemble only 30%. For test accuracy the Xception model achieved 58%, DenseNet

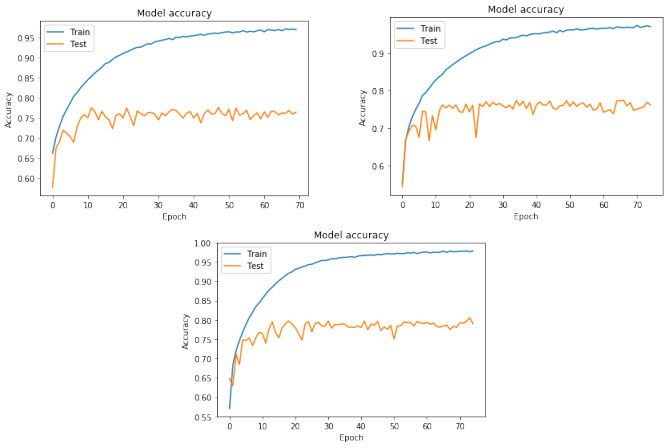


Fig. 8: DenseResNet **(a)** Approach 1 Accuracy **(b)** Approach 2 Accuracy **(c)** Approach 3 Accuracy

	Xception	DenseNet	Base	Ensemble
Correct	58.1%	46.8%	33.8%	4.8%
Incorrect	41.9%	53.2%	66.2%	95.2%

TABLE I: Ensemble Test Accuracy

46%, the base model 33%, and the ensemble network only 5%. The training results are shown in Fig. 9 and the test statistics are shown in Table I. Upon inspection of the ensemble predictions there is a heavy bias towards classes 9 and 99. This is currently unsolved, although it appears the prediction planes and image data of the modified CIFAR data set are as expected.

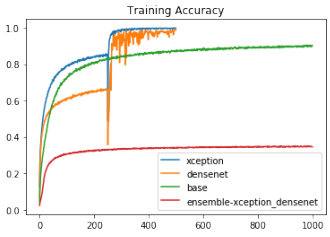


Fig. 9: Ensemble Training Accuracy

D. Deep Supervision

The proposed experiment did not produce a training phase that converged. However, DSN was able to outperform it's competition when it was released in 2014 with 74% accuracy on CIFAR-10. While, interestingly, CDNS is able to achieve 75%, slightly more than DSN, it outperforms the same architecture in Places dataset by 4%.

Methods	top-1 val/test	top-5 val/test
Places-CNN-5 [13]	50.4 / 50.0	80.9 / 81.1
Places-GoogleNet [1]	-/ 56.3	-/ 86.0
Places-CNN-8 (ours)	54.0 / 53.8	83.7 / 83.6
Places-CNDS-8 (ours)	**54.7 / 55.7**	**84.1 / 85.8**

Fig. 10: CNDS performance compared to the DSN on MIT Places dataset

VI. Discussion

Shallow networks definitely help better the accuracy but we have not yet seen the optimal neurons to get good generalization. We need to do some more trial and error to figure out optimal values to generalize the solution across new type of data sets.

The combination of DenseNet and ResNet were studied in using single blocks. Only single blocks were used in the study to simplify and decrease the computational timing. Results of single blocks of DenseNet and ResNets can be compared with the results of DenseResNet blocks.

The expectation of the ensemble model providing better accuracy than the individual ensemble members was not satisfied. This looks to be an issue with the ensemble network itself, judging by the heavy bias on some of the classification labels. Further inspection is necessary to discover why this is. Our immediate suspicion lies with how the prediction data is being injected into the feature plane and will experiment with alternative methods such as postponing the injection to the flattening layer in order to skip the convolutional blocks that may provide limited benefit to a feature plane of a single number.

Existing deep supervision methods for training are based on outdated framework, making replication harder. However, more complicated secondary routes (deep supervision losses) can be beneficial depending on the task . Deep supervision seems to be more promising where the output is also a higher level presentation of features such as 3D points, structure [12] or edges [13].

VII. Conclusion and Future Work

For shallow network parts, we would like to dynamically increase and decrease the neurons across different shallow layers, currently we exponentially increase but we plan to make it dynamic and see if it helps improve the accuracy. Each layer need not increase or decrease the neurons and we need to find optimal neurons per layer. Some more trial and error might help to find optimal hyper parameters. Also, w.r.t long connection part, we need to try to use similar analogy and try to connect across the groups, which might also help. We have not seen anybody else try these new style of networks.

For the DenseResNet blocks the testing accuracy reached to a maximum of 76 %. As this includes only single DenseNet and Single ResNet blocks the results achieved could be compared with single DenseNet and single ResNet blocks. Future works include the depth of DenseResNet can be increased and compared with standard DenseNet and ResNet.

As demonstrated by the previous research, deep supervision can improve accuracy, albeit minimally . Further experiments with conducted on training hidden layers to utilize feature maps (e.g. edges, image gradients, single kernel outputs). Furthermore, intermediate activations and feature maps of trained networks can be used to influence hidden layers where transferring weights is not possible.

References

[1] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," Tech. Rep., 2009.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," 2015. arXiv: 1512.03385. [Online]. Available: http://arxiv.org/abs/1512.03385.

[3] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-January, Institute of Electrical and Electronics Engineers Inc., 2017, pp. 2261–2269, ISBN: 9781538604571. DOI: 10.1109/CVPR.2017.243.

[4] *Applications*. [Online]. Available: https://keras.io/applications/#documentation-for-individual-models.

[5] X. Zhang, J. Zou, K. He, and J. Sun, "Accelerating very deep convolutional networks for classification and detection," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 10, pp. 1943–1955, 2016.

[6] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. DOI: 10.1162/neco.1997.9.8.1735. eprint: https://doi.org/10.1162/neco.1997.9.8.1735. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735.

[7] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 07-12-June-2015, pp. 1–9, 2015, ISSN: 10636919. DOI: 10.1109/CVPR.2015.7298594. arXiv: arXiv:1409.4842v1.

[8] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," 2015. arXiv: 1502.03167. [Online]. Available: http://arxiv.org/abs/1502.03167.

[9] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, "Deeply-Supervised Nets," in *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, G. Lebanon and S. V. N. Vishwanathan, Eds., ser. Proceedings of Machine Learning Research, vol. 38, San Diego, California, USA: PMLR, 2015, pp. 562–570. [Online]. Available: http://proceedings.mlr.press/v38/lee15a.html.

[10] L. Wang, C.-Y. Lee, Z. Tu, and S. Lazebnik, "Training deeper convolutional networks with deep supervision," *arXiv preprint arXiv:1505.02496*, 2015.

[11] S. Xie and Z. Tu, "Holistically-nested edge detection," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1395–1403.

[12] C. Li, M. Z. Zia, Q.-H. Tran, X. Yu, G. D. Hager, and M. Chandraker, "Deep supervision with intermediate concepts," *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[13] Y. Liu, P.-T. Jiang, V. Petrosyan, S.-J. Li, J. Bian, L. Zhang, and M.-M. Cheng, "Del: Deep embedding learning for efficient image segmentation.," in *IJCAI*, 2018, pp. 864–870.

[14] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.

[15] F. Chollet and others, *Keras*, https://keras.io, 2015.

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

[17] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-January, Institute of Electrical and Electronics Engineers Inc., 2017, pp. 1800–1807. DOI: 10.1109/CVPR.2017.195.

[18] J. Deng, W. Dong, R. Socher, L. jia Li, K. Li, and L. Fei-fei, "Imagenet: A large-scale hierarchical image database," in *In CVPR*, 2009.