A Deeper Look into Dense Shortcut Nets

Seyed Ahmad Abdollahpouri Hosseini
Department of Computer Science
University of Toronto
ahmadph@cs.toronto.edu

Maryam Ebrahimi
Department of Computer Science
University of Toronto
mary.ebrahimi@mail.utoronto.ca

Teerapat Chaiwachirasak
Department of Computer Science
University of Toronto
teerapat.chaiwachirasak@mail.utoronto.ca

Abstract

DenseNets have demonstrated superior performance on vision tasks, but have a much higher memory footprint than ResNets. DSNets claim to have solved this trade-off. In this paper, we will take a deeper look at this architecture and compare it side-by-side to the other two in multiple experiments. We find that DSNets are indeed signficantly better than ResNets, but only if the model size is large enough. We also test the effect of different hyperparameters and provide a guideline on when to use each model based on memory and time requirements.

1 Introduction

Deep convolutional neural networks (CNNs) (Krizhevsky et al., 2012) have demonstrated state-of-the-art performance on a variety of computer vision tasks (He et al., 2015), (Faster, 2015),(Girshick et al., 2014). They integrate features from different levels of abstraction (Zeiler and Fergus, 2014), which are enriched by increasing the depth of the network. However, as depth increases, the problem of vanishing gradients becomes more pronounced. Two popular solutions are the ResNet (He et al., 2015) and DenseNet (Huang et al., 2016) architectures. ResNets reuse features from previous layers by adding identity shortcuts, while DenseNets concatenate features of all preceding layers with the current one. DenseNets are more expressive and demonstrate better performance but have heavier memory requirements due to the concatenation operation. This trade-off creates a dilemma in choosing between ResNets and DenseNets in practical applications.

To address this dilemma, (Zhang et al., 2020) introduces the DSNet architecture, utilizing dense, normalized shortcut connections that are summed instead of concatenated. This is a middle-point between ResNet and DenseNet, performing better than Resnets but with much lighter memory requirements than DenseNets. However, the DSNet paper does not provide a comprehensive, side-by-side comparison of ResNet, DenseNet, and DSNet models on similar settings (all the comparisons are mainly with ResNet). This is why we aim to take a deeper look at this architecture with more experiments. You can find our code here: https://github.com/teerapat-ch/image_nn

Our contributions include:

- Implementing the DSNet architecture using the ResNet backbone for the CIFAR dataset, as this backbone was not explored in the original paper.
- Modifying the architectures to remove all the differences not essential for the different types of shortcut connections in order to isolate their effect.
- Studying the effects of different hyperparameters on the performance of the models.
- Studying whether the reported improvements in DSNet are statistically significant through statistical tests.
- Providing a guideline on when to use each model based on memory and time requirements.

Preprint. Under review.

2 Related Work

The remarkable success of deep CNN networks has turned the design of such networks into a hot research topic. This success is due to numerous methods that classified as micro-module design and macro-architecture design. Micro module design techniques are those that can be inserted into existing neural network architectures to improve their performance, such as normalization modules (Ioffe and Szegedy, 2015), attention modules (Hu et al., 2017), and group convolutions (Zhang et al., 2017). Normalization modules are perhaps the most widely used among this group. In the literature, normalization has been mainly used in the residual paths of architectures like ResNets, but the method we investigate uses it in the dense shortcut path.

On the other hand, macro architecture design aims to design effective backbones architectures usable in many different applications. Famous examples include architectures like AlexNet (Krizhevsky et al., 2012), VGGNet (Simonyan and Zisserman, 2014), GoogleNet (Szegedy et al., 2015). Among these architectures, ResNet (He et al., 2015) and DenseNet (Huang et al., 2016) are two popular ones due to their effectiveness despite their simplicity. Compared to ResNets, DenseNets have a more complex architecture which leads to higher memory usage but yields better performance. DSNet is among the many approaches that combine these two ideas, and we will be investigating it further.

3 Method

Deep models suffer from the vanishing gradient problem. ResNet (He et al., 2015) uses the simple solution of adding residual connections between earlier and deeper layers to allow access to information from previous layers. Figure 1a shows the architecture of "ResNet block"s. ResNet does summation for combining the residual connection and the block output. Thus the size of the feature maps stays the same.

DenseNet (Huang et al., 2016) tries to improve the performance of ResNet by connecting each layer to the feature maps of all preceding layers, which are concatenated together, resulting in higher memory usage. Figure 1c shows the architecture of a "Dense Block". More information flow in the DenseNet model improves the training process. So even though DenseNet requires more training resources, it offers better performance.

DSNet (Zhang et al., 2020) combines the advantages of ResNet and DenseNet and achieves a middle ground performance with less required resource than DenseNet. First, to control the memory consumption, DSNet uses summation instead of concatenation to combine the residual connections. And Second, DSNet suggests using a weighted, normalized version of these connections (1b). Normalization is needed to avoid any feature dominating others. Also, the weighted summation allows the network to learn proper weights for each normalized feature map depending on its significance.

The following equations describe ResNet, DenseNet and DSNet respectively. where f_l is the feature map of layer l, H_l is the convolution weight for that layer, "*" and "/" are the convolution and concatenation operations respectively. X_i represents each of the preceding inputs. And finaly $DS()$ indicates dense shortcuts, referring to normalization and channel-wise weights.

$$f_l = H_l * (X_0 + X_1 + ... + X_l) \tag{ResNet}$$

$$f_l = H_l * (X_0/X_1/.../X_l) \tag{DSNet}$$

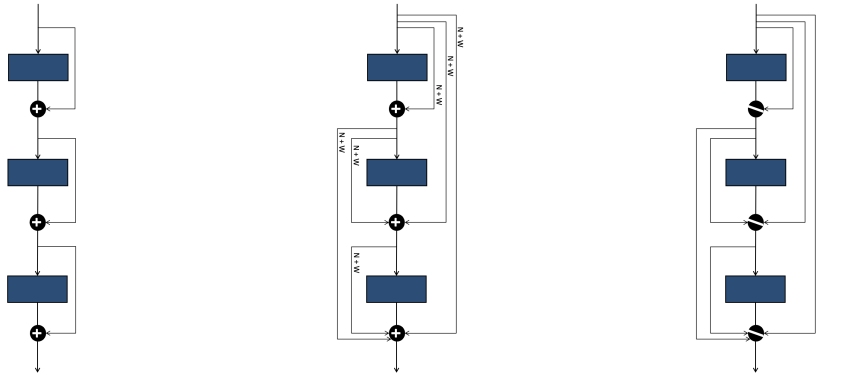
$$f_l = H_l * (DS_l^0(X_0) + DS_l^1(X_1) + ... + DS_l^{l-1}(X_{l-1}) + X_l) \tag{DenseNet}$$

4 Experiments

In this section, we compare ResNet, DSNet, and DenseNet models in different categories in our experiments. We use CIFAR-10 and CIFAR-100 for our experiments. This dataset contains 60,000 32x32 images split into 50,000, 5000 and 5000 sets for training, validation, and testing, respectively. For the data augmentation, we adopt random cropping with 4-pixel padding and horizontal flipping. The images are normalized using the training dataset's mean and standard deviation.

Since we're using CIFAR, we chose the CIFAR-specific architecture of ResNet, which uses smaller and fewer convolutional layers than the original ResNet, which designed for the ImageNet dataset. This architecture is more suitable for the CIFAR dataset and suits our limited resources better.

We tried to keep our implementations as close as possible to the ResNet backbone for CIFAR and only change the parts necessary to accommodate the different types of connections being studied.



(a) ResNet block (b) DSNet block (c) DenseNet block

Figure 1: Three used structures for models

4.1 Hyperparameter analysis

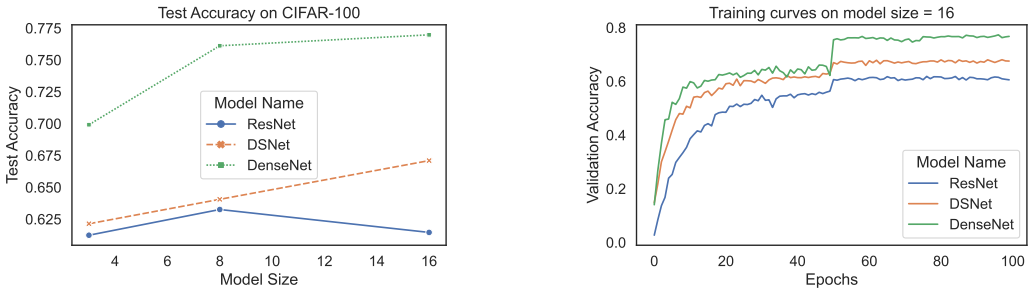
We tried DSNet with CIFAR100 with different learning rates and batch sizes to choose the best configuration for our experiments. The results are summarized in table 1. As expected, smaller batch sizes result in higher test accuracies. Also, the best initial learning rate is 0.1. Note that just like ResNet and DSNet paper suggest, we divide the learning rate by ten at $0.5 \times$ #epochs and $0.75 \times$ #epochs. The experiments in the further sections are all done with the best configuration found here (batch size = 32 and learning rate = 0.1) unless stated otherwise.

Table 1: Test accuracy of DSNet on different hyperparameters

Model size	**3**		**8**	
Batch size	**LR=0.01**	**LR=0.1**	**LR=0.01**	**LR=0.1**
128	0.6232	0.66	0.6518	0.6952
32	0.6624	0.6584	0.6746	0.6984

4.2 Effect of different models

In this part, we investigates the effect of model type and model size on test performance. We test three models, ResNet, DSNet, and DenseNet, with different numbers of basic blocks {3, 8, 16} (See Figure 1). In figure 2a we see that the test accuracy for DenseNet is higher than DSNet and DSNet performs slightly better than ResNet. As expected, DSNet acts as a middle ground between two other models. Our second observation is that these differences are even more visible with deeper models (higher model size). So the deeper the model gets, the more connection is needed to prevent the gradient vanishing problem; as a result, using DenseNet and DSNet becomes more necessary. In figure 2b as an example, we also included training curves for model size = 16, to show the differences between three models in the training process.



(a) Test accuracy of models with different sizes (b) Training Curves of models with size 16

Figure 2: Effect of model type and model size

4.3 Significance test

To test whether the observed differences are statistically significant, we conducted repeated experiments with different random seeds 5 times to collect the means and standard deviation of the test set accuracy of each model (Table 2). We used the hyperparameters from section 4.1 except for batch size which was set to 128. Using the collected samples, we also performed one-sided Welch's t-test with $\alpha = 0.01$ to test statistical significance (See table 3). As evident in the table, the improvement of DSNet over Resnet and DenseNet over DSNet are all statistically significant even with the strictest significance levels. The only exception is the improvement of DSNet over ResNet in model size = 3. This confirms our previous experiments that DSNet only shows improvement on bigger model sizes.

Table 2: Mean and standard deviation of test accuracy. Each experiment was repeated 5 times.

Model Size	**ResNet**	**DSNet**	**DenseNet**
3	0.6298 ± 0.0074	0.6272 ± 0.0058	0.662 ± 0.0098
8	0.6317 ± 0.0062	0.6542 ± 0.0036	0.7071 ± 0.0048

Table 3: T-test results. Each model is compared with the next-best model in a one-sided t-test to see whether improvements are statistically significant. R=Resnet, DS=DSNet, and D=DenseNet

Model size	**3**		**8**	
Null-hypothesis	**R >= DS**	**DS >= D**	**R >= DS**	**DS >= D**
P-value	0.69764	0.00032	0.00029	0.00029
Null-hypothesis rejected?	No	Yes	Yes	Yes

4.4 Effect of different datasets

The test accuracy on CIFAR-10 (See Table 4) follows the same trend as in CIFAR-100, where DSNet's improvement only becomes prominent on bigger model sizes. As CIFAR-10 is relatively an easier dataset, the improvements from DSNet and DenseNet are relatively lower than the increase seen in CIFAR-100.

Table 4: Test accuracy on CIFAR-10 vs CIFAR-100

	CIFAR-10		**CIFAR-100**	
Model size	**3**	**8**	**3**	**8**
ResNet	0.913	0.9186	0.6554	0.6850
DSNet	0.907	0.9210	0.6584	0.6984
DenseNet	0.919	0.9442	0.6992	0.7612

4.5 Memory and time analysis

In this section we detail the number of parameters for all three models with different model sizes, their memory use, and the time it takes to train them for 100 epochs (See Table 5). The models have all been trained on a single NVIDIA T4 GPU card on the CIFAR-100 dataset with batch size of 128. As can be seen, the memory use increases when we move from ResNet to DSNet and then to DenseNet. This is expected since the DSNet introduces extra normalization and weight parameters on top of the ResNet model, and the DenseNet model uses dense, concatenated connections which naturally leads to a larger computation graph. This trend repeats itself on the training time of the models, which is, again, expected since we have more operations to perform when we move from left to right in the table.

Table 5: Memory usage and parameters for models of different sizes.

Model size	**3**			**8**		
Model name	**ResNet**	**DSNet**	**DenseNet**	**ResNet**	**DSNet**	**DenseNet**
Memory Use	0.19 GB	0.29 GB	1.05 GB	0.49 GB	1.02 GB	4.96 GB
#Parameters	278,916	280,932	175,228	766,116	778,212	771,228
Time (hh:mm)	00:43	00:36	00:56	00:58	01:36	04:00

5 Discussion and Conclusion

In summary, we find that DSNets do outperform ResNets while having a much lower memory footprint than DenseNets. However, this only happens if the model size is large enough. Also, the original paper claims competitive performance with DenseNets, which is an effect that we could not reproduce. Based on our time and memory analysis, if you have strict memory / train time constraints, then ResNets are the best choice. If you want the best performance and aren't restricted by memory, then choose DenseNets. And if you want a middle-ground, go for DSNets, unless your model is already very small, in which case ResNets and DSNets perform the same.

References

RCNN Faster. 2015. Towards real-time object detection with region proposal networks. *Advances in neural information processing systems* 9199, 10.5555 (2015), 2969239–2969250.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 580–587.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. *CoRR* (2015).

Jie Hu, Li Shen, and Gang Sun. 2017. Squeeze-and-Excitation Networks. *CoRR* (2017).

Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. 2016. Densely Connected Convolutional Networks. *CoRR* (2016).

Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *CoRR* (2015).

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems* 25 (2012).

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1–9.

Matthew D Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *European conference on computer vision*. Springer, 818–833.

Chaoning Zhang, Philipp Benz, Dawit Mureja Argaw, Seokju Lee, Junsik Kim, Francois Rameau, Jean-Charles Bazin, and In So Kweon. 2020. ResNet or DenseNet? Introducing Dense Shortcuts to ResNet. *CoRR* (2020).

Ting Zhang, Guo-Jun Qi, Bin Xiao, and Jingdong Wang. 2017. Interleaved Group Convolutions for Deep Neural Networks. *CoRR* (2017).