DenResNet: Ensembling Dense Networks and Residual Networks

Victor Cheung
Stanford University
Computer Science Department
hoche@stanford.edu

Abstract

We combine various state of the art approaches to training deep convolutional neural networks to achieve the best performance possible on the Tiny ImageNet dataset. We emphasize the depth of the network through residual networks, transfer learning with pretrained models, and ensemble methods. We achieved a final ensemble test error of 25.6%*, which places us at the top of the leaderboard. We further experimented with Bayesian Optimization, ensembling Densely Connected Networks, and ensembling only shallow networks with extensive retraining.*

1. Introduction

Since first introduced as the negcognitron in 1980 by Kunihiko Fukushima [5] in the heydays of neural networks for handwritten character recognition, Convolutional Neural Networks (CNNs) gradually assumed a dominant role in visual machine learning research. The structure of images, despite living in very high dimensional spaces, are exploited in CNNs to enable feasibility of training: networks use only local information (filters) with shared parameters—resistance to noise and further dimensionality reduction is possible through pooling and more recently through larger strides. We now take the representational and inference power of CNNs for granted, fortunately due to the many methods invented to make training CNNs feasible. Furthermore, deep networks have become immensely popular in the machine learning domain for its empirical ability to model complex response functions (surfaces) with data. In computer vision, the reintroduction of convolutional networks as AlexNet and demonstration of its abilities after 2012 have spurred a hotbed of activity and research. Since then, performance has increased on canonical datasets (ImageNet, CIFAR-10, CIFAR-100) year on year [7, 13, 8].

We address the Tiny ImageNet dataset in this paper, where we stand on the shoulder of giants and adapt the many tools now available to us to train a high-performing model, with performance rivaling the state-of-the-art a short time ago. We note that human performance on the ImageNet dataset was benchmarked at $\sim$ 5% by Andrej Karparthy [10] in 2014. Since then, advances in CNNs, counting DenseNet, ResNet, Inception amongst them, have resulted in top-5 error in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) that surpasses human performance in 2015 and 2016[16, 9, 8]. These results are due primarily to the complexity, depth and width of recent CNN architectures, as well as attendant increases in graphical processing units' (GPUs) capabilities.

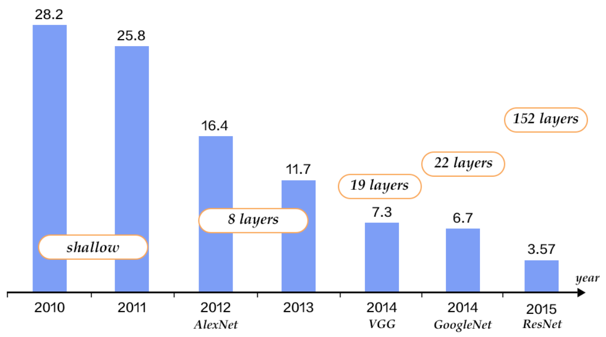

Figure 1. Performance improvement on the ILSVRC ImageNet Challenge has improved as networks have become progressively deeper and more complex in architecture over time [15].

We take advantage of these advances in order to train deep CNNs on the Tiny ImageNet dataset. We note that He et al's residual networks with 18 residual blocks (34 layers) achieved a top-1 error of $\sim$ 24% with 10-crop testing (average of 10 crops, $(4 + 1 \text{ center}) * 2$) on the validation set. Given our reduced number of classes, we expect to hit at least this level of accuracy on our validation set, even with limited training time.

Evaluation of our models will consist of comparisons to state-of-the-art results currently published on ImageNet, as well as peer comparisons on the evaluation severs for the Tiny ImageNet competition. The final evaluation will be based on an unseen and locked away test set on the server.

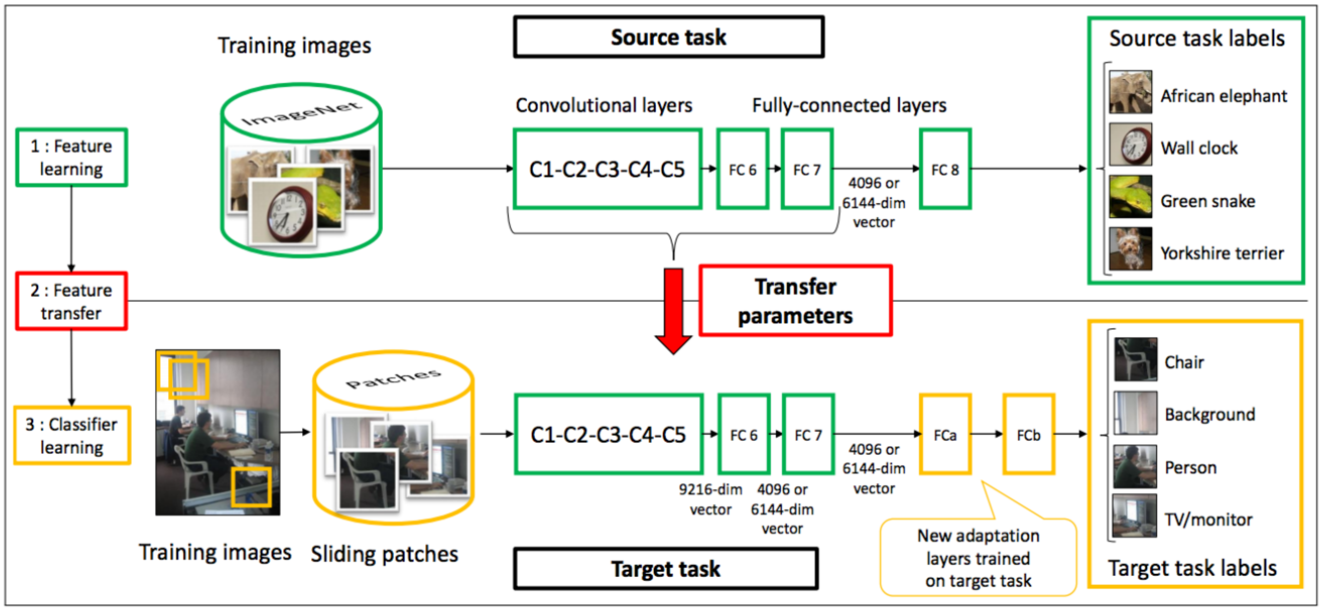

Figure 2. Parameter Transfer from [14]. A 5-layer CNN is trained on the ImageNet dataset; the learned representation the first convolutional layer to the second-to-last fully connected layer are used to initialize an identical network, with two more fully connected layers appended, then trained on the Pascal VOC dataset with different target labels, image characteristics. [14] demonstrated that the learned internal representations are useful even in the presence of dataset drift, different occlusion characteristics, image sizes, etc.

2. Related Work

Our project is motivated in particular by studies done on ImageNet, the use of residual networks and densely networks in ILSVRC, and ensemble models in statistics and machine learning.

2.1. Network Depth

The depth of neural networks is critical to their performance. Very recent work has begun on the theory behind why deep networks are better empirically even though shallow networks can theoretically model any non-linear function [3]. Liang et al. [12] showed that for many piecewise smooth functions, that the number of neurons needed to approximate the function to a certain degree of accuracy is exponentially larger for shallow networks than for deeper nets.

However, the training of deep networks is plagued with practical difficulties. Exploding and vanishing gradients for recurrent nets, gradient saturation for certain nonlinearities, dying ReLu units, and so on, have previously circumscribed the meaning of "deep" in deep networks from 16 to 30 [8]. Much work has been dedicated to resolution of such difficulties, including highway networks to short circuit gradient updates, LSTM and GRU units, and so on. The introduction of residual networks in 2016 works in the same vein; the authors use residual blocks as the basic unit in the network, as opposed to layers, and the gradients are geared towards learning the "residual mapping" as opposed to the actual underlying "mapping". The end results are theoretically the same, but in practice residual learning performs much better. The shortcut connections introduced between blocks thus enable the training of much deeper networks with well-established optimization methods (SGD, etc.). In short, He et al. have made it possible to go deeper still!

2.2. Transfer Learning

Given limited resources and computing time, how best to achieve a high-performing model? In natural language processing, researchers and practitioners alike begin by co-opting pretrained word embeddings for their own purposes. GloVe and Word2Vec trained on massive corpus like Wikipedia Commons and Common Scrapes of the web are amongst the most popular options. Modern deep networks for CNNs easily have millions, tens of millions or hundreds of millions of parameters in the extreme. To train a network from scratch is a data-hungry and computing-heavy process. Therefore, researchers in image recognition tasks need to adopt pretrained weights from existing models like AlexNet, VGGNet and GoogleNet to obviate the need for re-computation of low level features with each new model.

In particular, Oquad et al. found in 2014 that the convolutional layers inside CNNs behave as *generic feature extractors* that transform the images as representations living in a linearly separable space [14]. Oquab et al. demonstrated that CNNs pretrained on ImageNet can be retrained on Pascal VOC to achieve state-of-the-art performance. In essence, they experimentally verified that the learned inter-

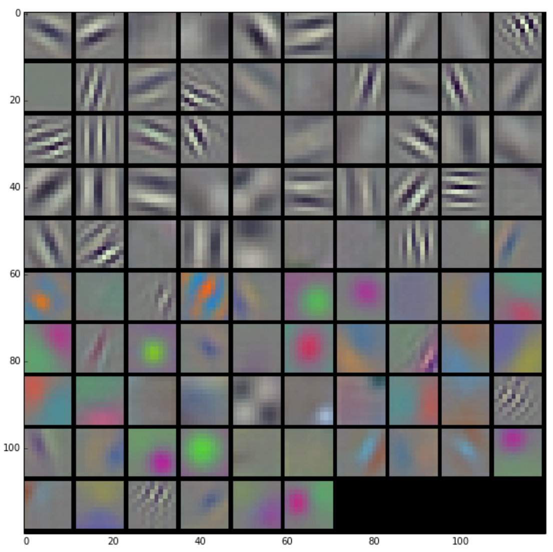
Figure 3. First convolutional layer: filters are edge detectors trained for different orientations, colors, size and frequency. This is usually invariant of the dataset the convolutional network was first trained on. Support for the transfer learning hypothesis advanced by [14] and first suggested in [11] derives thus. Later layers become increasingly more specific to the task and dataset at hand.

nal representations from models trained on a large dataset can be adapted usefully for models trained on smaller datasets. This was made possible since the final fully connected layers (adaptation layers) are specific to the task at hand: it contains numbers of parameters much fewer than the millions and can easily be retrained given even a modest-sized dataset. Furthermore, Oquad et al. found that overlaps in the source and target models has positive effects on training. Certain categories in Tiny ImageNet overlap with those in the considerably larger ImageNet datasets: transfer learning thus stands to benefit from the overlap.

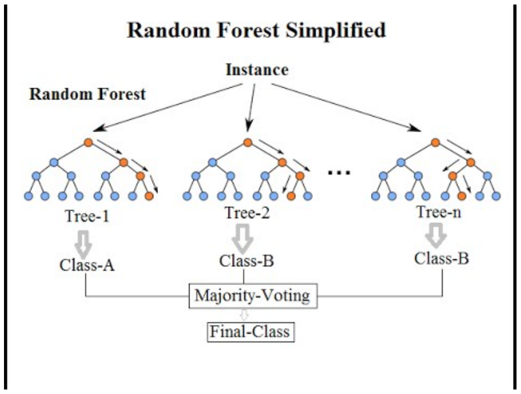

Figure 4. **Random Forests**

2.3. Ensemble Models

Random Forests (RF) are well-known and extensively studied in Statistics for classification and regression [2, 4]. RF is an ensemble or meta algorithm that constructs a large number of decision trees (typically stumps with few layers for computation purposes) via bootstrapped samples (usually with replacement) from a dataset. Each decision tree is also constructed using only a sample (without replacement) of the features or predictors available (similar to dropout). Each resulting decision tree is a high variance and high biased estimator of the outcome. However, we then *average* over the outputs from the many hundreds of thousands of individual estimators to reach a consensus decision, resulting in a meta-estimator or ensemble-estimator with both low variance and low bias [2]. In particular, this approach emphasizes independent construction of estimators. There are other meta-estimators that rely on dependent construction of estimators contingent on the performance of past estimators (boosting or stochastic boosting). We emphasize independently constructed for reasons discussed in the **Methods** section.

3. Methods

3.1. Ensemble Architecture

Independently constructed ensembles are rare in deep networks likely due to the high computational cost of training even individual networks. The performance of deep networks has obviated the need for ensembles of shallower networks; however, in our situation, given the decreasing marginal gains in accuracy with extended epochs of training and limited resources, we would rather train and conduct early-stopping on multiple deep networks than expend all of our resources on a single network. We expect that this ensemble approach will outperform predictions from a single model. Our ensemble model takes the output scores s_c^i from model $i \in I = \{1, \ldots, n\}$ for a class $c \in C$ and normalizes them over the classes and then marginalize over the models via Softmax, as in

$$o_i = P(y = c|x, s_i) = \frac{e^{s_c^i}}{\sum_{c \in C} e^{s_c^i}} \tag{1}$$

$$\arg\max_{c \in C} P(y = c|x) = \frac{e^{o_i}}{\sum_{i \in I} e^{o_i}} \tag{2}$$

where (1) is softmax over class scores and (2) is performing Maximum a posteriori (MAP) inference over model output probabilities to select class labels on the test set. Our loss function is the canonical Cross Entropy loss, where the cross entropy loss over a dataset D of size m is

$$\text{Cross Entropy Loss} = -\frac{1}{m} \sum_{j=1}^{m} \log(f(x_j, y_j)) \tag{3}$$

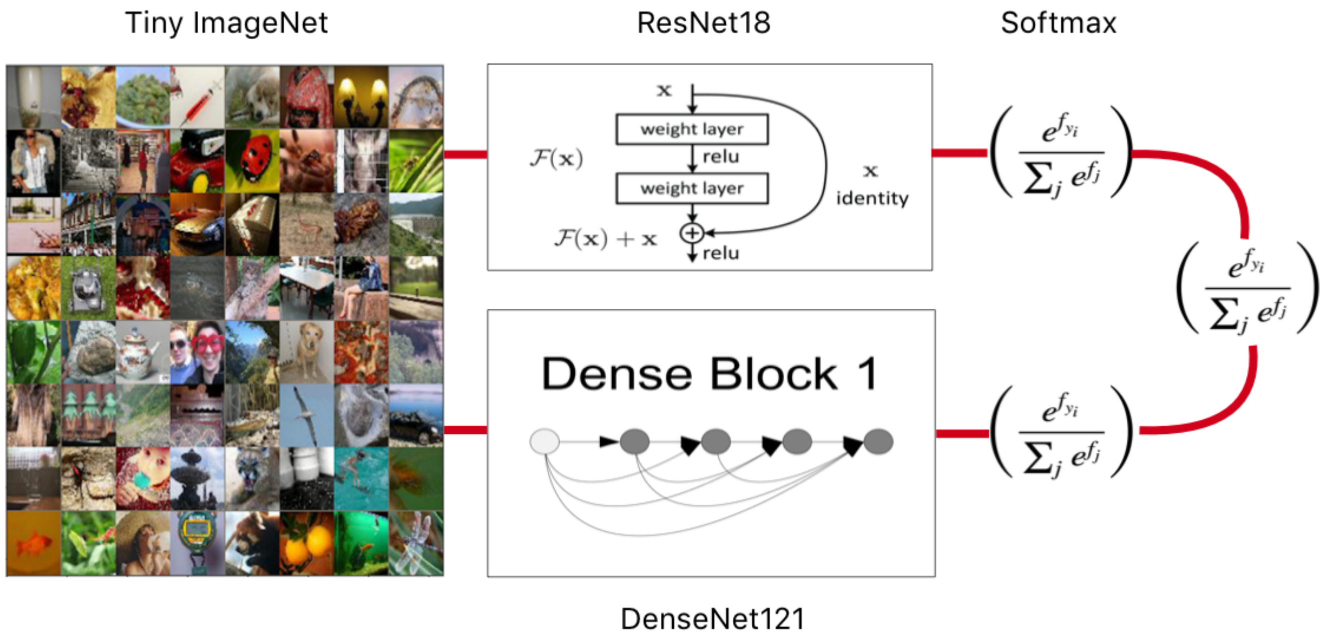
Tiny ImageNet
ResNet18
Softmax
x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu
$\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right)$
Dense Block 1
DenseNet121

Figure 5. **Ensemble Predictions**: Illustration of ensemble with two a residual network and a densely connected network. Scores from last affine layers of different models are not guaranteed to be on same scale. We need to normalize to probability before applying Softmax again to get final ensemble output. Models are trained separately to ensure variance in features emphasized by model. Access to one GPU means that models are trained sequentially with handtuning on hyperparameters.

where f is the softmax function taking the input x_j and the label y_j. Note the final loss is

$$\text{Loss}_D = \text{Cross Entropy Loss} + \sum_l \lambda_l \|\boldsymbol{w}_l\| \tag{4}$$

where l denotes the layer in the network and λ is the regularization parameter.

We further note that past iterations of this competition have seen various forms of ensemble architectures, ranging from simple averaging, ensemble majority voting and weighted averaging with weights determined by the accuracy of the network. We note that naive majority voting is unlikely to be successful in this scenario unless we have a large number of well-trained neural networks—computational and time constraints implies that this is unlikely to be the case (see section under **Ensemble Models** and [2] for more on why majority voting works well in cases with large number of high variance, high bias classifiers). Our approach is thus one of model confidence with equal weights across equally well-trained models. The probability with which a model makes a decision is taken into account — not the final output. In this way we obtain a more nuanced decision criterion for the final MAP inference.

3.2. Framework

We will use PyTorch [1] for training our networks. Empirical comparisons between PyTorch and the many other frameworks for deep learning have shown that PyTorch performs up to 100x faster on training for certain tasks. Atomic units and other building blocks are implemented modularly in PyTorch. Their inclusion of residual blocks, as well as preprocessing tools for data augmentation are particularly attractive, which leaves us free to experiment with architecture and hyperparameter selection as opposed to worrying about implementation details. Given the importance of transfer learning in performing well on a new dataset with limited training data and training time, torchvision's [1] pretrained models make the framework doubly suitable. Furthermore, PyTorch's modularity and design means that bugs are easily isolated to the lines responsible — compared to Tensorflow's error messages, PyTorch makes debugging considerably easier.

3.3. Training

Our models are initialized with pretrained weights from the PyTorch model zoo. We use a batchsize of 128 for ResNet18, 64 for ResNet34 and ResNet50, and 24 for ResNet101 and Resnet152. We train over 50 epochs for ResNet18 and 10 for the rest. DenseNet121 used batch size of 24 due to its intensive usage of memory. We use Stochastic Gradient Descent as in [8] with weight decay (l2-regularization) of 0.0001 and momentum of 0.9.

We emphasize the importance of momentum here. As Goh notes in [6], the story usually told about momentum—

that it is “a heavy ball rolling down a hill, ..., dampening oscillations and causing us to barrel through narrow valleys, small humps and local minima”—is only partially true. In our models, where are several hundred layers deep, training without momentum is likely to end badly. Along with Batch Normalization, momentum is critical to our re-training of deep networks, and provides a quadratic speedup over gradient descent alone [6]. In essence, momentum allows us to at most double the learning rate before losses begin to diverge (as we enter an unstable feedback loop of too large step sizes versus the maximum allowed learning rate [6]).

We further note the potential advantage of having fewer training epochs available to us due to time and resource constraints. Residual Networks trained from random initialization on large datasets like ImageNet are typically done over multiple GPUs at large batch sizes, in a few hundred epochs over the span of a couple weeks. However, our few training epochs amounts to early stopping, which is similar in effect to regularization and leads to better generalization.

Our learning rate started at 0.01 and dropped by 10 every time error plateaus. Losses for deeper networks were more noisy due to lower batch sizes. Images from the dataset are preprocessed by random horizontal flips and then normalized by the ImageNet mean and standard deviation prior to training; images are scaled up and normalized prior to training, validation and testing. Training times for each of the networks differed drastically due to the use of different batch-sizes, learning rates, and number of parameters in the feed forward and backpropagation steps. For shallower networks, 15 minutes was sufficient per epoch. On the deepest networks and the networks with the most parameters, each epoch took more than 120 minutes. All models were trained on a Google Compute Instance with an Nvidia Tesla K80 GPU with 12GB of GDDR5 memory.

4. Dataset

Our dataset, the Tiny ImageNet, is a simpler and smaller clone of the ImageNet, target of the ILSVRC challenges. Our dataset contains an order of magnitude fewer images, from 1.2 million down to 120000. The dataset consists of 200 classes, each with 500 training examples and 50 validation images. The final test will be on 50 images of each class. Training and validation examples are labeled and contain bounding boxes for the object of interest. Each image is of size 64 by 64, smaller than 256 by 256 ImageNet images. We expect that this down-sampling will adversely affect the performance of our networks, especially in cases where the object to be identified is a relative small fraction of the pixels in the image, as is the case when objects and occluded. We note that 500 images per class is also a relative small number, especially against other canonical datasets such as CIFAR10 and MNIST. We are also precluded from pretraining our network on auxiliary datasets, such as by scraping images with the same label as our desired target dataset from search engines, which only emphasizes that the use of pretrained weights is critical, as discussed above.

5. Experimentation and Results

Our current baseline model is a residual network with 18 blocks (2 convolutional layers per block) with manually tuned hyperparameters, step sizes, learning rate decay and regularization. With limited training time and epochs we can achieve ~ 0.65 top-1 error rate on the validation set.

Our final model is an ensemble of three models: ResNet101, ResNet152 and DenseNet121 [8, 9]. We apply the model normalization, marginalization and MAP inference described in the ensemble architecture. We find that the **final top-1 error on the test set is 25.6%**, which as of June 12th, 2017 is first on the leaderboard. We achieved a validation error of 0.200 with the ensemble. Since models are trained independent, training errors for the ensemble are not applicable and not reported here. We note the importance of a correct training regime for each of the models — the criticality of dropping learning rates by an order of magnitude when loss stagnates cannot be overstated.

5.1. Other Experiments

We tried a number of approaches to the obtain the best performing network. The following did not enter into the final model used on the test set, with the exception of the DenseNet, where we used DenseNet121 as part of the ensemble.

5.1.1 Shallow Networks

We first attempted to build an ensemble of only shallow models, with the expectation that longer training epochs on a smaller dataset will in turn compensate; however, we quickly found that our training regime resulted in stagnating accuracy even past 50 epochs. Shallower networks were thus incapable of rapidly learning useful representations of the ImageNet dataset. Experiments on the deeper networks, in particular ResNet101 and ResNet152 demonstrated that, despite the considerably longer time per epoch, that loss quickly converges towards zero and accuracy towards the previous peak by the shallower models, even just after 2 epochs! We thus diverted our attention towards deeper networks and dedicated the rest of our training time towards them.

5.1.2 Bayesian Optimization

The importance of proper hyperparameters for the training and out-of-sample performance of models cannot be understated — however, there is still no real way to systematically and formally select the *best* hyperparameters, esp. given that we are often optimizing w.r.t non-convex functions. Our original focus on Bayesian Optimization—an old idea—seemed promising as a method for automatically selecting hyperparameters as opposed to manual grid or random search. However, more recent research has almost universally used linearly decaying or step decaying learning rates. This was the first warning sign that the applicability of Bayesian Optimization—as ingenious as it is—to image recognition problems is suspect. After days of experimentation with various packages for Bayesian Optimization, the gains over manually tuned hyperparameters proved marginal. This could however be attributed to the relative short training epochs actually possible for us, given limited resources. It could be that Bayesian Optimization would result in considerable gains on a longer time horizon, ex. 50 epochs and onwards, as is seen in [8] where loss suddenly drop days into the training regime. Bayesian Opt. is a consistent and logically pleasing way to teasing out the "right" hyperparameters, and we believe it will assume a greater role in image recognition research and Machine Learning in general once the cost of adoption drops.

5.1.3 Other Network Architectures

DenseNets [9] take the motivation behind residual networks to an extreme, by short circuiting connections within blocks. Every layer within a dense block feeds forward to every layer after it. Feature maps from previous layers are available to later layers, which results in greater variation in subsequent layers, hopefully improving the efficiency with which networks learn. In practice, DenseNets are massively memory intensive, and quickly overwhelms the 12GB of memory available on an Nvidia Tesla K80 Unit. The increased number of connections slowed down backpropagation drastically, resulting in unrealistic time frames for sufficient training of the deeper models. We anticipate that given greater resources and over a long time horizon, DenseNets would have surpassed our ResNet models in performance and contribute meaningfully to the ensemble predictions.

6. Conclusion

Our final ensemble model achieves performance at the top of the leaderboard, at a **final top-1 error on the test set of 25.6%**. We have demonstrated the versatility of the various techniques presented herein: in particular, transfer learning, residual networks, and ensemble modeling have been leveraged to train our model on the Tiny ImageNet dataset with relatively scarce data and few training epochs relative to the models presented in the ILSVRC: nonetheless, we find that we are yet able to rival their top-1 performance on our smaller dataset.

References

[1] Pytorch framework: https://github.com/pytorch/pytorch. Web.

[2] M. A. Arbib. *The handbook of brain theory and neural networks*. MIT press, 2003.

[3] J. Ba and R. Caruana. Do deep nets really need to be deep? In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2654–2662. Curran Associates, Inc., 2014.

[4] R. A. Berk. *Statistical learning from a regression perspective*, volume 14. Springer, 2008.

[5] K. Fukushima. Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural networks*, 1(2):119–130, 1988.

[6] G. Goh. Why momentum really works. *Distill*, 2(4):e6, 2017.

[7] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, and G. Wang. Recent advances in convolutional neural networks. *arXiv preprint arXiv:1512.07108*, 2015.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[9] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten. Densely connected convolutional networks. *arXiv preprint arXiv:1608.06993*, 2016.

[10] A. Karpathy. What i learned from competing against a convnet on imagenet, 2014.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[12] S. Liang and R. Srikant. Why deep neural networks for function approximation? 10 2016.

[13] D. Mishkin, N. Sergievskiy, and J. Matas. Systematic evaluation of cnn advances on the imagenet. *arXiv preprint arXiv:1606.02228*, 2016.

[14] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1717–1724, 2014.

[15] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[16] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. *arXiv preprint arXiv:1602.07261*, 2016.

7. Appendix

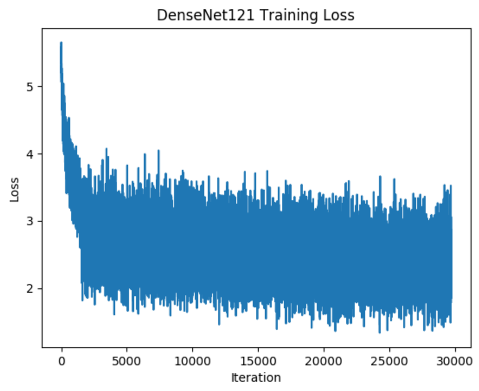

Figure 6.

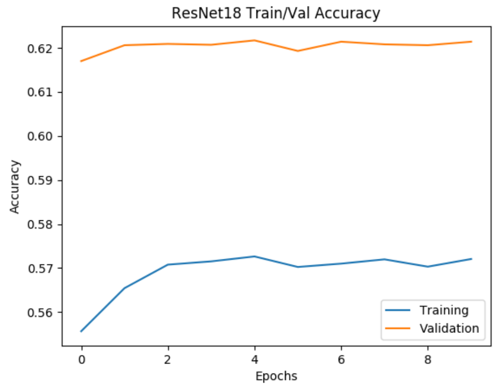

Figure 9.

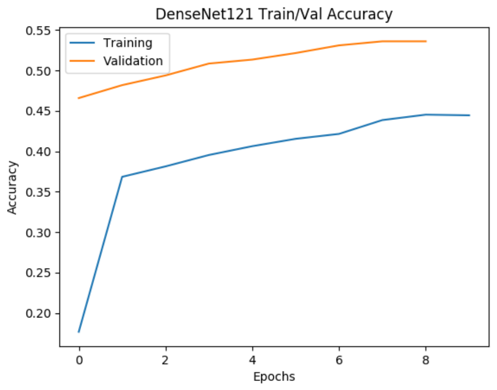

Figure 7.

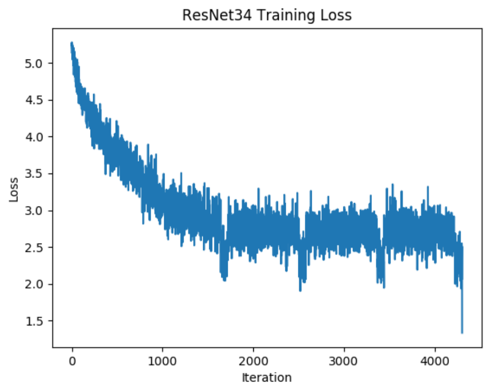

Figure 10.

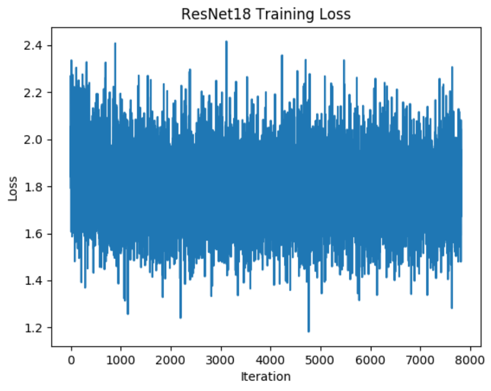

Figure 8.

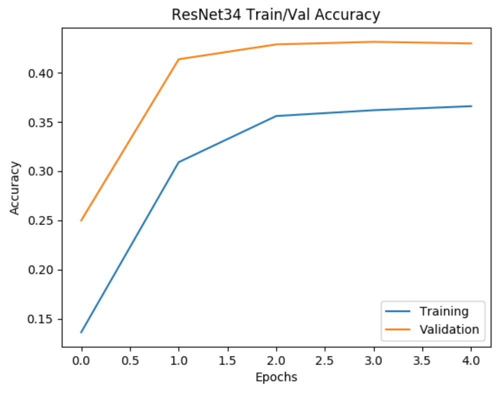

Figure 11.

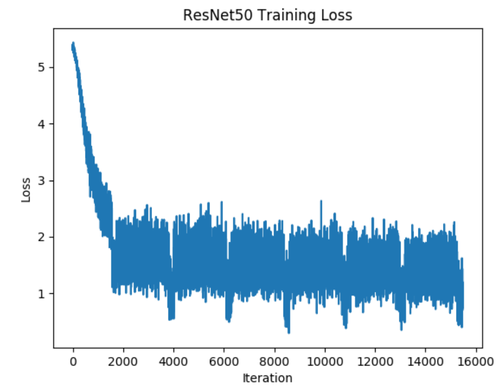

Figure 12.

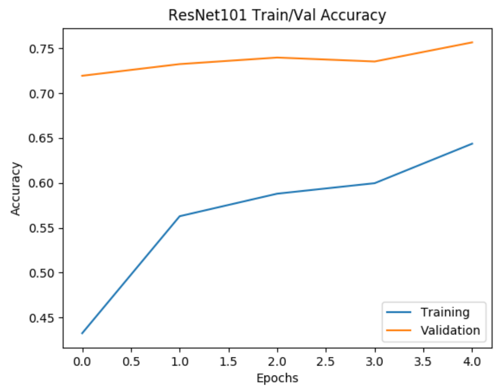

Figure 15.

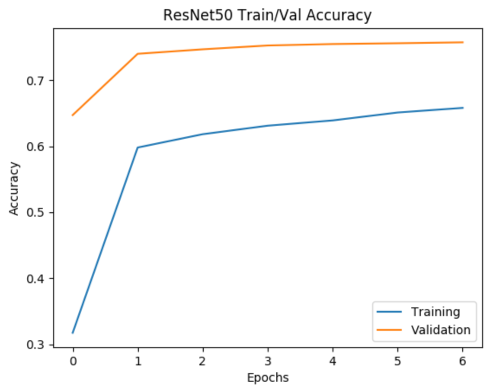

Figure 13.

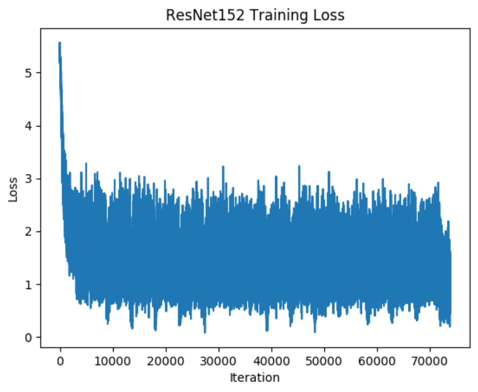

Figure 16.

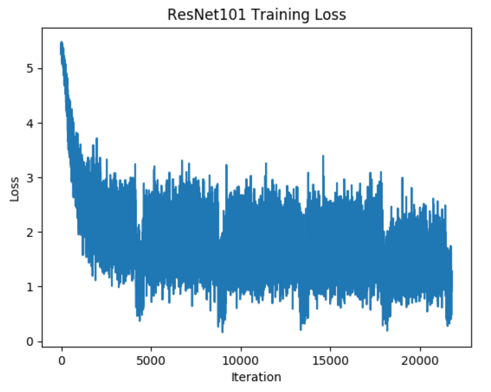

Figure 14.

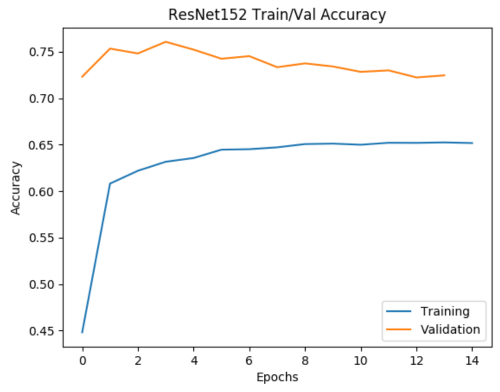

Figure 17.