Beyond Fine Tuning: Adding capacity to leverage few labels

Nathan O. Hodas[†] **Kyle Shaffer**[*] **Artem Yankov**[*] **& Courtney D. Corley**[†]
Pacific Northwest National Laboratory
Washington, USA
{kyle.shaffer,artem.yankov,court,nathan.hodas}@pnnl.gov

Aryk Anderson[‡]
Eastern Washington University
Cheney, Washington
aryk.anderson@eagles.ewu.edu

Abstract

In this paper we present a technique to train neural network models on small amounts of data. Current methods for training neural networks on small amounts of rich data typically rely on strategies such as fine-tuning a pre-trained neural network or the use of domain-specific hand-engineered features. Here we take the approach of treating network layers, or entire networks, as modules and combine pre-trained modules with untrained modules, to learn the shift in distributions between data sets. The central impact of using a modular approach comes from adding new representations to a network, as opposed to replacing representations via fine-tuning. Using this technique, we are able surpass results using standard fine-tuning transfer learning approaches, and we are also able to significantly increase performance over such approaches when using smaller amounts of data.

1 Introduction

Training generalizable models using only a small amount of data has proved a significant challenge in the field of machine learning since its inception. This is especially true when using artificial neural networks, with millions or billions of parameters. Conventional wisdom gleaned from the surge in popularity of neural network models indicates that extremely large quantities of data are required for these models to be effectively trained. Indeed the work of Kriszhevsky (8) has commonly been cited as only being possible through the development of ImageNet (16). As practitioners and researchers continue to explore neural networks in more specialized domains, the volume of available labeled data also narrows. Although training methods have improved, it is still difficult to train deep learning models on small quantities of data, such as only tens or hundreds of examples.

The current paradigm for solving this problem has come through the use of pre-trained neural networks. (1) were able to show that transfer of knowledge in networks could be achieved by first training a neural network on a domain for which there is a large amount of data and then retraining that network on a related but different domain via fine-tuning its weights. Though this approach demonstrated promising results on small data, these models do not retain the ability to function as previously trained. That is, these models end up fine tuning their weights to the new learning task, forgetting many of the important features learned from the previous domain.

[*] Seattle, WA
[†] Richland, WA; AA and NOH contributed equally

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

In this paper we present our neuro-modular approach to fine-tuning. The first contribution is demonstrating a novel topological approach to fine-tuning. We quantitatively compare traditional fine-tuning with our modular approach, showing that our approach is more accurate on small amounts of data (<100 examples per class). We also demonstrate how to improve classification in a number of experiments, including CIFAR-100, text classification, and fine-grained image classification, all with limited data.

2 Related Work

Transferring knowledge from a source domain to a target domain is an important challenge in machine learning research. Many shallow methods have been published, those that learn feature invariant representations or by approximating a value without using an instance's label (14; 19; 15; 24; 22; 3). More recent deep transfer learning methods enable identification of variational factors in the data and align them to disparate domain distributions (20; 9; 2; 21). Mesnil et al. (11) presents the Unsupervised and Transfer Learning Challenge and discusses the important advances that are needed for representation learning, and the importance of deep learning in transfer learning. Oquab et al. (13) applied these techniques to mid-level image representations using CNNs. Specifically, they showed that image representations learned in visual recognition tasks (ImageNet) can be transferred to other visual recognition tasks (Pascal VOC) efficiently. Further study regarding the transferability of features by (23) showed surprising results that features from distant tasks perform better than random features and that difficulties arise when optimizing splitting networks between co-adapted neurons. We build on these results by leveraging existing representations to transfer to target domains without overwriting the pre-trained models through standard fine-tuning approaches.

The progressive network architecture of (17) shares a number of qualities with our work. Both the results we present here and the progressive networks allow neural networks to extend their knowledge without forgetting previous information. In addition, Montone et al. (12) discusses a semi-modular approach. Montone et al. also froze the weights of the original network, although it did not demonstrate success on the small data regime. Our approach provides the user with a novel fine-tuning approach for learning on small data. Our modular approach detailed here allows small data to adapt to different domains. Our architecture also complements existing network building strategies, such as downloading pre-trained neural networks to then be fine-tuned for domain adaptation. The modular approach presents a particular advantage when large volumes of training data for the base network may not be available, i.e. when you can not retrain an entire network from scratch. Our approach addresses a key need for large deep learning pre-trained models to be adapted to new tasks using only small amounts of data.

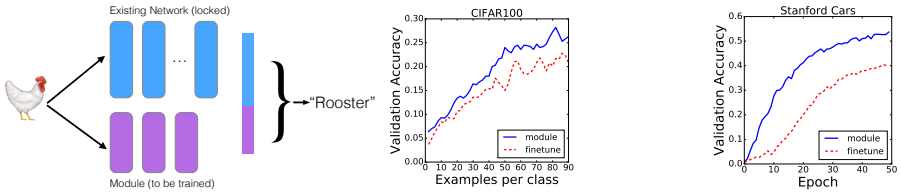
3 Results

Generically, modular neural networks are directed graphs of pre-trained networks linked together with auxiliary, untrained networks. As depicted in Fig. 1a, only the new components of the resulting network is trained. The architecture could take the form of simply placing two networks in parallel (the two-towers approach), shown in Fig. 1a.

More formally, we can describe any neural network as a differentiable function G taking input $\mathbf{x}$, and parameterized by weight matrix θ, $G(\mathbf{x};\theta)$. In our modular architecture, we denote at least one additional network $G'(\mathbf{x};\theta')$. It is crucial to note that the overall architecture of G' need not be identical to that of G, and this allows considerable flexibility for the resulting architecture to learn complementary representations from each sub-network. Each of these networks, G and G' are treated as feature extractors, lacking classifier layers for making final predictions. Our final modular architecture combines the output of the base network G and the modular network, G', via tensor concatenation, which is then fed to a classifier layer for final predictions, as in Equation 1:

$$H = f\big(G(\mathbf{x};\theta) \oplus G'(\mathbf{x};\theta');\theta_f\big). \tag{1}$$

Tuning H (and minimizing the resulting loss) can be broken down into error due to $\frac{\partial G}{\partial \theta}$, $\frac{\partial G'}{\partial \theta'}$, and $\frac{\partial f}{\partial \theta_f}$. If the domain of the new data closely aligns with the original domain of the fully trained G, then we may presume $|\frac{\partial G}{\partial \theta}| \ll |\frac{\partial G'}{\partial \theta'}|$. Thus, for computational expedience, for our modular approach



(a) (b) (c)

Figure 1: (a) An example module setup. By explicitly preserving the original representation learned on pre-trained net, the module is able to learn more robust features using fewer examples than naive fine-tuning. (b) Comparison of modular approach vs. fine-tuning based on amount of training data. (c) Comparison of validation accuracy on the Stanford Cars data. Training on full data

we freeze θ and only tune θ' and θ_f. This allows the network as a whole to retain the original representational power of the pre-trained network.

This allows the modular approach to more robustly handle small data than naive fine-tuning. To demonstrate this, we trained a network on CIFAR-10 data. The CIFAR-10 network was trained until it was 88.9% accurate, using the network in (5) with 3 residual units, for a total of 28 layers. We then compared two approaches. For the first approach, we simply fine tuned the CIFAR-10 network using training data from the CIFAR-100 dataset and replacing the final softmax. Next, we froze the original CIFAR-10 network and added an identical copy as a module, which would be trained on the same batches of data as the first approach. That is, we have: Network 1 – fine-tuning the base network and Network 2 – freezing the base network and fine-tuning a module. Network 1 and Network 2 have an identical number of weights and those weights have the same starting value. These two approaches are presented in eqs. 2 and 3 below.

$$\hat{y} = \sigma(G(\mathbf{x}; \theta_g + \Delta\theta_{g'})) \tag{2}$$

$$\hat{y'} = \sigma(H(\mathbf{x})) \tag{3}$$

where $\hat{y}$ denotes predictions made from a fine-tuned network, $\hat{y}'$ denotes predictions made from our modular architecture, and $\sigma(\cdot)$ is the softmax function.[3] G denotes the fine-tuned network with pre-trained weights θ_g learned from CIFAR-10 and H denotes a modular network as in Eq. 1 above.

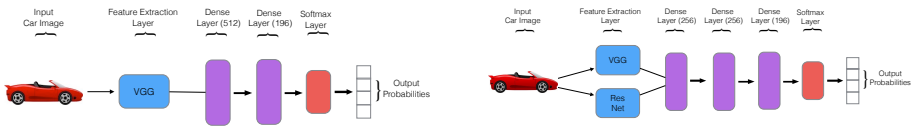
To train, we used the ADAM optimization algorithm (6)). We added an activation L2 regularization of $1e^{-6}$ to the module to help break degeneracy. We used batches of 200, where each batch contained two images per class. Each batch was iterated over five times, before the next batch was used. This iteration allowed simulating multiple epochs over small data. We recorded the results of the performance on the test set after each batch, in Fig. 1b.

We observe that for all amounts of training data, but particularly for small amounts of training data, the modular approach outperforms traditional fine-tuning. Of course, we chose to make the module a complete copy of the original CIFAR-10 network. This ensured we could compare with the same number of weights, same initialization, same data, etc. Further research will certainly reveal more compact module networks that outperform our example. We also tried retraining both the original network and the module, to see if the improvement was due simply to the increased capacity, and noted no improvement over training the module and freezing the original network. The module learns additional filters that boost performance on the transfer task.

To explore modules on a more real-world relevant task, we applied this approach to an imagenet-trained network (16). The Stanford Cars data set (7), which features 16,185 images of 196 classes of cars, is an example of a data set for fine-grained categorization. Rather than train a classifier to distinguish between fundamentally different objects like horses and planes, as required in the Large Scale Visual Recognition Challenge (16)), fine-grained categorization requires the classifier to learn subtle differences in variations of the same entity. For example, a classifier trained on the Stanford Cars data set would have to learn distinguishing features between a BMW X6 SUV from 2012 and an Isuzu Ascender SUV from 2008.

In these Stanford Cars experiments, both models utilize a transfer learning approach by leveraging the feature extraction output layers from the VGG16 model (18). The “fine-tuned" model passes the

[3]In this training setup, there is one softmax per category.



(a) (b)

Figure 2: (a) Network architecture used for Stanford Cars fine-tuned model. (b) ImageNet-based network architecture used for Stanford Cars module model.

VGG16 features to a fully connected layer of length 512 followed by a softmax layer of length 196, as seen in Fig. 2a. Gradient descent via RMSProp is used to train the dense layers (RMSProp performed better than ADAM on this task). The “module" model merges the fixed VGG16 features with a ResNet (4) model, whose output is then fed to two consecutive dense layers of length 256 and finally a softmax layer of length 196. The module model architecture is shown in Fig. 2b. Again, RMSProp is used to train ResNet and the dense layer weights after the two component models are merged, however the VGG feature-extraction weights remain unchanged ensuring that this architecture fully leverages the representational power of this sub-network.

As seen in Fig. 1c, after 50 epochs the module model appears to significantly outperform the fine-tuned model in validation accuracy. However, it should be noted that while the module model carries 19,537,990 trainable parameters the fine-tuned model only has 12,946,116 parameters. Thus, the generic module approach does not require any specific number of additional parameters. Furthermore, no hyperparameter optimization is performed on either model.

We further investigate the effects of our modular network approach by applying this method to a different modeling problem - text classification. Similar to image data, text represents an unstructured data-type that often exhibits long-term and interconnected dependencies that are difficult to model with simpler classifiers. Whereas in the case of images neighboring pixels may represent semantically related concepts or objects, in text words may exhibit long-term semantic or syntactic dependencies that can be modeled sequentially. These characteristics make text classification particularly well-suited to recurrent neural networks such as long short-term memory (LSTM) networks, but these learning methods typically require a great deal of data to be learned efficiently and to avoid overfitting.

To test our methodology, we evaluate a modular recurrent network against two individual recurrent neural networks, and an ensemble of these individual networks on the IMDB sentiment dataset.[4] Previous work has shown deep learning methods to be effective at sentiment classification performance on this dataset (10), however we add to this past work by presenting an analysis that demonstrates the effectiveness of modular networks in the case of extremely small training sets. To this end, we sample only 500 training examples from the original 25,000 available in the full training set, and evaluate on the full 25,000 validation examples. We use the same 500 training examples for each model evaluated in our experiments for consistency, and report accuracy for each model on the full validation set.

We evaluate four models in our text-classification experiments, two individual LSTM networks, an ensemble of these two LSTM networks, and a final model which is our modular recurrent network. The first model consists of three layers - an initial layer that projects sequences of words into an embedding space, a second LSTM layer with 32 units, and a final sigmoid layer for computing the probability of the text belonging to the positive class. Our second model is identical to the first except that we initialize the weights of the embedding layer using pre-trained GloVe word vectors.[5] In particular, we use 100-dimensional vectors learned from a 2014 version of Wikipedia. We also experimented with freezing the weights in the embedding layer of this second model, but found better performance when allowing this layer to be tuned in each epoch of training. For our final baseline, we ensemble the two individual models detailed above by training both networks in tandem on the same training set, and fitting a logistic classifier layer on the prediction probabilities of the two component networks.

To construct our modular network, we take the embedding and LSTM layers from our individual networks, and concatenate the output of both LSTM layers into a single tensor layer in the middle of

[4] http://ai.stanford.edu/~amaas/data/sentiment/

[5] http://nlp.stanford.edu/projects/glove/

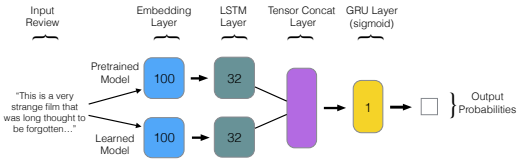


Figure 3: Diagram of architecture for our modular recurrent text classification network. Dimensionality for embedding layers and number of units for all other layers are given in boxes denoting those layers.

MODEL	**Base.**	**Pretrain**	**Ens.**	**Module**
ACCURACY (%)	61.9	64.9	65.3	**69.6**

Table 1: Accuracy results for text classification experiments, *u*sing only 500 training examples. Results are shown for the baseline model (**Base.**), pre-trained (GloVe) model (**Pretrain**), ensembled model (**Ens.**) and modular model (**Module**).

our modular network. Additionally, we modify the output of each of these component LSTM layers by forcing each to output a weight matrix that tracks the state of the LSTM layer across all timesteps throughout the dataset. In this way, we seek to fully leverage the sequential dependencies learned by these layers, and this method outperforms the simpler alternative method of simply outputting the final state of each of the LSTM layers. We then feed this concatenated layer to a gated recurrent unit (GRU) layer with a sigmoid activation function for calculation of class probabilities. We experimented with an LSTM and densely connected layers after the tensor concatenation layer, but found best performance with the GRU. All models were optimized with the ADAM algorithm, and trained for 15 epochs. An outline of this architecture can be seen in Figure 3.

Here, we report results for our classification experiments with the three networks described above. We see an accuracy of 61.9% for our first model which is trained entirely from the data without any pre-training. This is significantly lower than previously reported results, however we are training on only 2% of the available data to test our method's application to small training sets. We see better performance in terms of accuracy (64.9%) from our second model initialized with GloVe vectors. This seems to indicate that despite being trained on more formally written language in Wikipedia, these vectors can still boost performance on a task modeling text that is inherently subjective and opinion-based. The ensemble of these two individual networks further boosts accuracy to 65.3%, demonstrating that even simple combinations of these networks appears to aid model learning, despite receiving very little training data. Finally, we see an accuracy of 69.6% from our modular network, an increase of over 4% accuracy over the next best performing model. Because weight initializations of recurrent networks can greatly affect model performance, we ran the classification experiments with our modular network 10 times, and report the average accuracy across these 10 runs. As can be seen here, our modular approach improves on the best performing individual network, suggesting that this approach is useful for text classification even with very small amounts of training data. Further, the fact that our modular approach improves over an ensemble of the individual networks suggests that our modular architecture learns specific representations that are more advantageous than simply leveraging multiple models via standard ensembling techniques.

4 Conclusions

We have presented a neuro-modular approach to transfer learning. By mixing pre-trained neural networks (that have fixed weights) with networks to be trained on the specific domain data, we are able to learn the shift in distributions between data sets. As we have shown, often the new modules learn features that complement the features previously learned in the pre-trained network. We have shown that our approach out-performs traditional fine-tuning, particularly when the amount of training data is small – only tens of examples per class. Although it has a certain amount of computational complexity, its implementation simplicity and performance benefit over traditional fine-tuning more than justify the cost in many applications.

References

[1] Bengio, Yoshua et al. Deep learning of representations for unsupervised and transfer learning. *ICML Unsupervised and Transfer Learning*, 27:17–36, 2012.

[2] Ganin, Yaroslav and Lempitsky, Victor. Unsupervised domain adaptation by backpropagation. *arXiv preprint arXiv:1409.7495*, 2014.

[3] Gong, Mingming, Zhang, Kun, Liu, Tongliang, Tao, Dacheng, Glymour, Clark, and Schölkopf, Bernhard. Domain adaptation with conditional transferable components. In *Proceedings of The 33rd International Conference on Machine Learning*, pp. 2839–2848, 2016.

[4] He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[5] He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Identity mappings in deep residual networks. *arXiv preprint arXiv:1603.05027*, 2016.

[6] Kingma, Diederik and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[7] Krause, Jonathan, Stark, Michael, Deng, Jia, and Fei-Fei, Li. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 554–561, 2013.

[8] Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[9] Long, Mingsheng, Cao, Yue, Wang, Jianmin, and Jordan, Michael. Learning transferable features with deep adaptation networks. In *Proceedings of The 32nd International Conference on Machine Learning*, pp. 97–105, 2015.

[10] Maas, Andrew L., Daly, Raymond E., Pham, Peter T., Huang, Dan, Ng, Andrew Y., and Potts, Christopher. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P11-1015`.

[11] Mesnil, Grégoire, Dauphin, Yann, Glorot, Xavier, Rifai, Salah, Bengio, Yoshua, Goodfellow, Ian J, Lavoie, Erick, Muller, Xavier, Desjardins, Guillaume, Warde-Farley, David, et al. Unsupervised and transfer learning challenge: a deep learning approach. *ICML Unsupervised and Transfer Learning*, 27:97–110, 2012.

[12] Montone, Guglielmo, O'Regan, J Kevin, and Terekhov, Alexander V. The usefulness of past knowledge when learning a new task in deep neural networks. 2015.

[13] Oquab, Maxime, Bottou, Leon, Laptev, Ivan, and Sivic, Josef. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1717–1724, 2014.

[14] Pan, Sinno Jialin and Yang, Qiang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[15] Pan, Sinno Jialin, Tsang, Ivor W, Kwok, James T, and Yang, Qiang. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210, 2011.

[16] Russakovsky, Olga, Deng, Jia, Su, Hao, Krause, Jonathan, Satheesh, Sanjeev, Ma, Sean, Huang, Zhiheng, Karpathy, Andrej, Khosla, Aditya, Bernstein, Michael, Berg, Alexander C., and Fei-Fei, Li. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[17] Rusu, Andrei A, Rabinowitz, Neil C, Desjardins, Guillaume, Soyer, Hubert, Kirkpatrick, James, Kavukcuoglu, Koray, Pascanu, Razvan, and Hadsell, Raia. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[18] Simonyan, Karen and Zisserman, Andrew. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[19] Sugiyama, Masashi, Nakajima, Shinichi, Kashima, Hisashi, Buenau, Paul V, and Kawanabe, Motoaki. Direct importance estimation with model selection and its application to covariate shift adaptation. In *Advances in neural information processing systems*, pp. 1433–1440, 2008.

[20] Tzeng, Eric, Hoffman, Judy, Zhang, Ning, Saenko, Kate, and Darrell, Trevor. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014.

[21] Tzeng, Eric, Hoffman, Judy, Darrell, Trevor, and Saenko, Kate. Simultaneous deep transfer across domains and tasks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4068–4076, 2015.

[22] Wang, Xuezhi and Schneider, Jeff. Flexible transfer learning under support and model shift. In *Advances in Neural Information Processing Systems*, pp. 1898–1906, 2014.

[23] Yosinski, Jason, Clune, Jeff, Bengio, Yoshua, and Lipson, Hod. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pp. 3320–3328, 2014.

[24] Zhang, Kun, Schölkopf, Bernhard, Muandet, Krikamol, and Wang, Zhikun. Domain adaptation under target and conditional shift. In *ICML (3)*, pp. 819–827, 2013.