Do deep neural networks learn shallow learnable examples first?

Anonymous Authors[1]

Abstract

In this paper, we empirically investigate the training process of deep networks relative to fully trained classical machine learning models (shallow models). We found empirical proof that deep neural networks (DNNs) train by learning to correctly classify those examples that are classifiable by classical machine learning techniques (shallow-learnable) in the early epochs before learning the *harder* examples. Furthermore, we utilize this to suggest a way for partitioning the dataset into *hard* and *easy* subsets that can be used for improving the overall training process. Incidentally, we also found evidence of a subset of examples across all the datasets we considered, that were shallow learnable but not deep-learnable, which we did not expect at the outset. We also release our code for this work at < `github repository link` >

1. Introduction

Analyzing the *temporal journey* taken by deep neural networks (DNNs) during training has elicited a lot of attention recently. The authors in (Arpit et al., 2017) suggested that *DNNs learn simple patterns first, before memorizing*. More specifically, they posit that real world datasets are littered with *easy* examples characterized by simple patterns that are learned in the initial epoch(s) before the conquest of *hard* examples in the training dataset. Tishby et al (Tishby & Zaslavsky, 2015) conjectured that DNN training was characterized by two distinct phases consisting of an initial fitting phase (memorization) and a subsequent compression phase. While this claim was questioned in (Saxe et al., 2018), the authors do remark that *when an input domain consists of a subset of task-relevant and task-irrelevant information, hidden representations do compress the task-irrelevant information*. These works do suggest that the easy-hard dichotomy in real-world datasets influence the learning in DNNs and goad a data-dependent approach towards understanding the capacity of DNNs. Taking cue from this, we strive to contribute to this growing body of literature by bringing in another viewpoint: The dichotomy between shallow learnable examples and deep learnable examples in the dataset. More specifically we try to address the questions:

1. Is the notion of *easiness* same for models with as different paramaterizations and architectures as classical machine learning models and deep networks the same? And hence is attached to the example independently of a model?
2. If we are to investigate the examples that a DNN learns to correctly classify over the training batches, do we observe a shallow learnable to deep learnable regime change?
3. Are there examples that are shallow learnable but somehow a DNN with a far better overall accuracy fails to classify? At the heart of this quest is to understand if shallow learnability is a good proxy for the *easiness* of an example.

The rest of the paper is organized as follows. In section 2, we present the quantitative methodology we used to answer these questions raised above. In section 3, we showcase our empirical experiments with the results covered in section 4. We conclude the paper in section 5.

2. Proposed Methodology

In this section we delineate our proposed method to study the learning process of a deep learning model $\mathcal{D}$ relative to a classical machine learning model $\mathcal{M}$.

2.1. Tracking the learning process

We propose to measure the generalization capability of a deep learning model $\mathcal{D}$ on unknown data using a custom contingency matrix $\mathcal{T}$. In other words, instead of just calculating the accuracy of the deep learning model after i steps of back propagation, $\mathcal{D}^{(i)}$ on the test set, we propose to calculate the contingency matrix $\mathcal{T}^{(i)}$. Furthermore, to study the learning process of model $\mathcal{D}$ relative to $\mathcal{M}$, we look at how the matrix $\mathcal{T}^{(i)}$ evolves as the model trains.

2.1.1. Preparing the Contingency Matrix

The contingency matrix $\mathcal{T}$ is defined using the pattern of classification of the test set examples by the models $\mathcal{D}$ and $\mathcal{M}$. Referring to Tabel 1, $\mathcal{T}_{00}^{(i)}$ denotes the number of examples on the test set that both the models $\mathcal{D}^{(i)}$ and $\mathcal{M}$ make an error on. Similarly, $\mathcal{T}_{10}^{(i)}$ denotes the number of test

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

Table 1. Construction of the contingency matrix $\mathcal{T}^i$ on the test for bench-marking $\mathcal{D}^{(i)}$ against $\mathcal{M}$.

	$\mathcal{M}$ incorrect	$\mathcal{M}$ correct
$\mathcal{D}^{(i)}$ incorrect	$\mathcal{T}^{(i)}_{00}$	$\mathcal{T}^{(i)}_{01}$
$\mathcal{D}^{(i)}$ correct	$\mathcal{T}^{(i)}_{10}$	$\mathcal{T}^{(i)}_{11}$

set examples that $\mathcal{D}^{(i)}$ classifies correctly but $\mathcal{M}$ classifies wrongly and so on and so forth for $\mathcal{T}^{(i)}_{01}$ and $\mathcal{T}^{(i)}_{11}$.

2.1.2. Analyzing Learning Dynamics

Several useful metrics can be derived from $\mathcal{T}$ that analyze different aspects of the learning process. Naively, accuracies of models $\mathcal{D}$ and $\mathcal{M}$ can be recovered simply by

$$Accuracy(\mathcal{D}^{(i)}) = \frac{\mathcal{T}^{(i)}_{10} + \mathcal{T}^{(i)}_{11}}{\mathcal{T}^{(i)}_{10} + \mathcal{T}^{(i)}_{11} + \mathcal{T}^{(i)}_{01} + \mathcal{T}^{(i)}_{00}}$$

$$Accuracy(\mathcal{M}) = \frac{\mathcal{T}^{(i)}_{01} + \mathcal{T}^{(i)}_{11}}{\mathcal{T}^{(i)}_{10} + \mathcal{T}^{(i)}_{11} + \mathcal{T}^{(i)}_{01} + \mathcal{T}^{(i)}_{00}}$$

Note that since we're tracking the learning dynamics for $\mathcal{D}^(i)$ and model $\mathcal{M}$ is kept the same, $Accuracy(\mathcal{M})$ actually remains constant throughout. More interesting metrics can be derived such as the accuracy of $\mathcal{D}^{(i)}$ on subsets of the training data that $\mathcal{M}$ classifies correctly ($\mathtt{R}^{\mathtt{i}}_{+}$) and those that $\mathcal{M}$ classifies wrongly ($\mathtt{R}^{\mathtt{i}}_{-}$).

$$\mathtt{R}^{\mathtt{i}}_{+} = \frac{\mathcal{T}^{(i)}_{11}}{\mathcal{T}^{(i)}_{11} + \mathcal{T}^{(i)}_{01}} \qquad \mathtt{R}^{\mathtt{i}}_{-} = \frac{\mathcal{T}^{(i)}_{10}}{\mathcal{T}^{(i)}_{10} + \mathcal{T}^{(i)}_{00}}$$

Also, to measure how many times more accurate $\mathcal{D}^{(i)}$ is on one subset versus the other, we study the ratio of the above two accuracies $\mathtt{R}^{\mathtt{i}} = \mathtt{R}^{\mathtt{i}}_{+}/\mathtt{R}^{\mathtt{i}}_{-}$.

3. Experiments

We perform our experiments under three different regimes divided on the basis of relative performance of classical Machine Learning models to Deep Networks to check the robustness of our observations. We experiment with (a) The MNIST dataset, where the ML models tend to perform competitively with the ConvNets (b) The CIFAR10 dataset where ML models perform worse than neural networks but not are still quite good and lastly (c) the CIFAR100 dataset, where the deep networks far outperform the classical Machine Learning models. We preprocess all the datasets to center and normalize the images before training models but do not augment data to avoid other potential confounding factors in our observations.

3.1. Datasets

MNIST: MNIST (Mixed National Institute of Standards and Technology) (LeCun, 1998) is a very popular toy dataset consisting of images of handwritten digits and their numeric value as their labels. It consists of a total of $70,000$ gray-scale 28×28 labelled images divided in training set of $60,000$ and another test set of $10,000$ images. It is widely used to examine image recognition techniques and considered a relatively simple starter dataset.

CIFAR: The CIFAR database (Krizhevsky, 2009) refers to two related but different datasets namely, the CIFAR10 and CIFAR100 datasets. CIFAR10 consists of $60,000$ color 32×32 images divided into 10 object categories like `airplane`, `bird`, `cat` etc. It consists of a training set of $50,000$ and a test set of $10,000$ images.

CIFAR100 also has the same characterstics and size except it consists of 100 image classes rather than just 10. Thus, it consists of 600 examples of each class which are grouped proportionally into 500 in the training set and 100 in the test set. The dataset also contains the 100 classes grouped into 20 super classes but in this work we use fine grained classes for comparing models on CIFAR100.

3.2. Training Classical ML Models

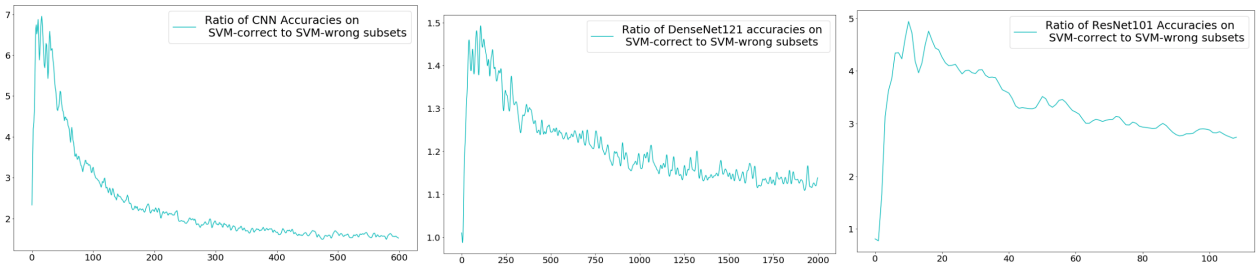
We train two types of machine learning models to study the learning process – Support Vector Machines (Cortes & Vapnik, 1995) and Random Forests (Breiman, 2001).

Random Forest: Random Forest are another family of successful Machine Learning models which have been applied to a large number of classification and regression problems (Liaw et al., 2002) like real time face detection (Cootes et al., 2012), Gene Selection (Díaz-Uriarte & De Andres, 2006), Remote sensing (Pal, 2005) and several other applications. In this paper, we train the Random Forests using 20 estimators and the `gini` index criterion (Pedregosa et al., 2011). We flatten the preprocessed images into a 784 dimensional vector in case of MNIST and into a 1024 dimensional vector for CIFAR to facilitate training the random forest.

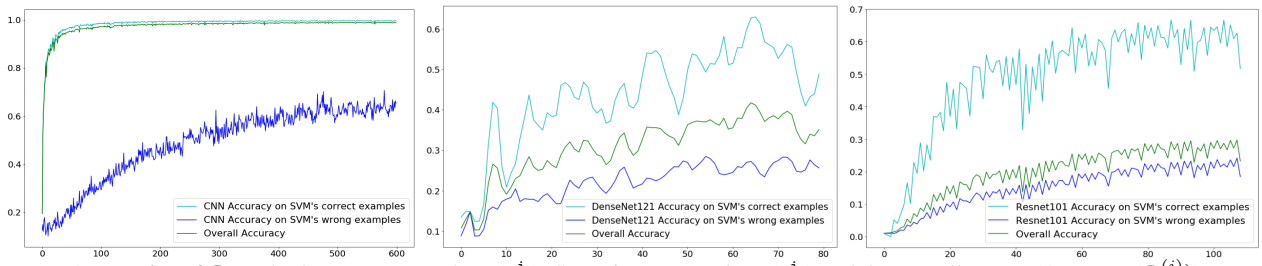
Support Vector Machines: Support Vector Machines (SVM) are extremely popular ML models shown to achieve good results in several limited data domains like hand written character recognition (Decoste & Schölkopf, 2002), face recognition (Heisele et al., 2001), hypertext classification (Pradhan et al., 2004) and several biological applications (Cuingnet et al., 2011; Yang, 2004). We train a SVM with slacks under the Radial Basis Kernel (Pedregosa et al., 2011) with hyperparameters $C = 1.0$ and $\gamma = 0.1$. Similar to Random Forests, we flatten out the images for training an SVM.

3.3. Training Deep Neural Networks

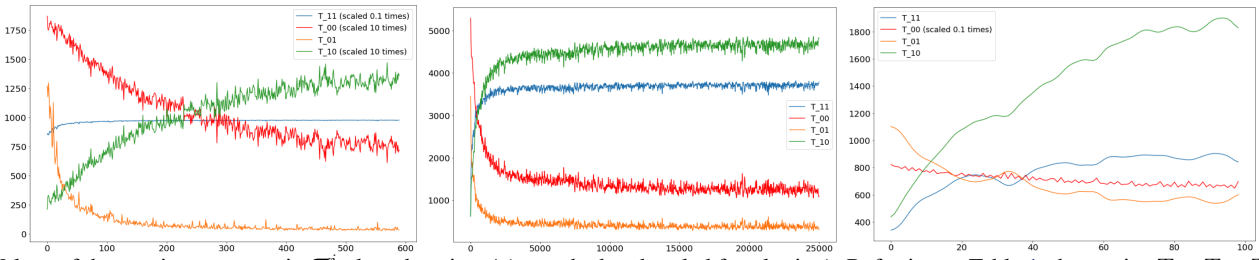
We train a different deep network on each dataset because of two reasons. First, some networks we train have a much larger capacity than others and hence are not suited for a small dataset like MNIST. Likewise, the architecture suited for MNIST is too small for a more complicated dataset like CIFAR and would not be a optimal choice for good



Ratio of Accuracies $\mathtt{R}^{\mathtt{i}}$ plotted against i with $\mathcal{M}$ being a Support Vector Machine.



Accuracies of $\mathcal{D}$ on the SVM-correct subset ($\mathtt{R}^{\mathtt{i}}_{+}$), SVM-incorrect subset ($\mathtt{R}^{\mathtt{i}}_{-}$) and the overall testset (Accuracy($\mathcal{D}^{(i)}$)).



Values of the contingency matrix $\mathcal{T}^i$ plotted against i (smoothed and scaled for plotting). Referring to Table 1, the entries T_{00}, T_{01}, T_{10} and T_{11} are denoted by red, orange, green and blue respectively.

Figure 1. Graphs showing evolution of various metrics on the test set (defined in Section 3) as training progresses for MNIST (left columns), CIFAR10 (middle column) and CIFAR100 (right column) datasets. Each X-tick represents: 640 images for MNIST, 128 images for CIFAR10 and 25,600 images for CIFAR100 dataset.

performance. Second, we want to check robustness of our observations across different model sizes and architectures and hence experiment with very popular but very different ontologies like DenseNet and ResNet.

Convolutional Neural Network: We train a small Convolution Network with the architecture : $[\mathtt{Conv} \triangleright \mathtt{ReLu} \triangleright \mathtt{MaxPool}] \mapsto [\mathtt{Conv} \triangleright \mathtt{ReLu} \triangleright \mathtt{MaxPool}] \mapsto [\mathtt{FC} \triangleright \mathtt{ReLu}] \mapsto \mathtt{FC} \mapsto \mathtt{SoftMax}$. The network is trained on the MNIST dataset with SGD with a learning rate 0.01, momentum set to 0.5 and a batch size of 64.

DenseNet121: DenseNet 121 (Huang et al., 2017) is a deep network that has been demonstrated to achieve a very good performance on that task of Image recognition on a variety of benchmarks. In a nutshell, DensetNet contains several DenseBlocks which inside themselves are a constant feature depth stack of ConvNets; connected in a fully connected fashion. We train DenseNet 121 on the CIFAR10 dataset with SGD with learning rate 0.1, momentum 0.5 and a batch size of 128. Further training details are in the appendix.

Resnet 101: ResNet 101 belongs to the very popular family of Residual Networks (He et al., 2016) and is widely used as a backbone in a variety of computer vision tasks like Image Recognition (He et al., 2016), Action Classification (Feichtenhofer et al., 2016), Image to Image Translation (Zhu et al., 2017) etc. We use ResNet 101 for image recognition on the CIFAR100 dataset where it is trained with SGD under same parameters as mentioned for DenseNet121 but with an additional weight decay of 5×10^{-4}.

4. Results

Here we discuss our observations while studying the training process under the lens of machine learning models. Table 2 reports the maximum accuracy achieved with the models discussed in Section 3.2 and 3.3. While the accuracies are reported for completeness, this work studies the learning process in the early stages and hence is not concerned with the maximum accuracy. Moreover, in cases of CIFAR10 and CIFAR100 datasets the operating regime for DenseNet121 and ResNet101 are far from their optimal performance.

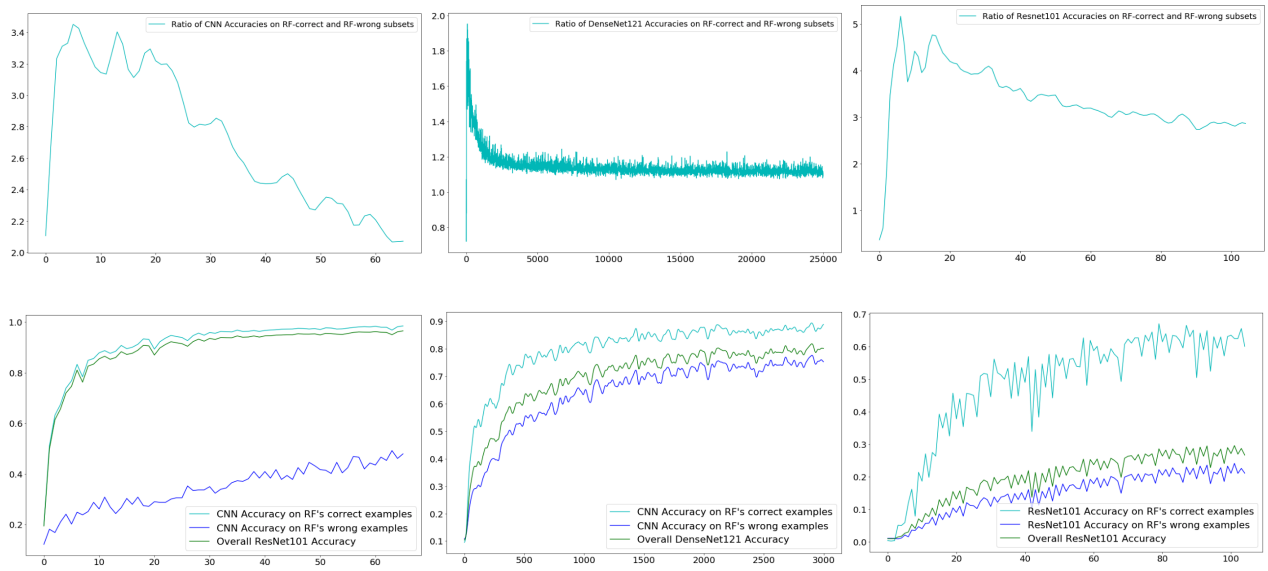


Figure 2. Graphs showing evolution of the same metrics compared to the Random Forest model. All graphs are arranged in the same configuration as Fig 1. We provide the plot for contigency matrix values for random forest in the appendix.

Table 2. Maximum Accuracy achieved by various models on the three Datasets. All accuracies are for top 1 prediction.

	MNIST	CIFAR10	CIFAR100
SVM	97.92%	40.08 %	14.42 %
Random Forests	96.14%	35.86 %	14.26 %
Deep Network	98.8% (2 layer CNN)	95.04% (DenseNet121)	77.78 % (ResNet 101)

4.1. Ratio of Accuracies

Referring to the top row in Fig 1 and 2, we observe how the ratio of accuracies ($\mathtt{R}^+$) changes during training. Note that if the two subsets, $\mathcal{M}$-correct and $\mathcal{M}$-incorrect were completely irrelevant and similar for training process of $\mathcal{D}$, $\mathtt{R}^+$ would remain identically 1. However, we observe that across datasets and ($\mathcal{M}$, $\mathcal{D}$) pairs, the curve has a right skewed unimodal shape with a sharp hump. We also note that this happens in very early stages of training, sometimes as early as just after $1/20^{th}$ epoch over the training set. Also, the accuracies can be very different on the two subsets with $\mathcal{D}$ being upto 8 times more accurate on $\mathcal{M}$-correct subsets than on $\mathcal{M}$-correct subsets in some cases. Furthermore, we observe that the curve shows a long tail as the ratio $\mathtt{R}^i$ returns back to 1. This observation is the cornerstone in confirming out hypothesis that deep networks training starts from quickly learning shallow classifiable *easy* examples and then slowly extends to the hard ones.

4.2. Accuracy Plots

The second row of Fig. 1 and Fig. 2 depicts test accuracies as training progresses. Note that the overall trend is increasing as expected from a network in early stages of training, however there are huge gaps in accuracy on the two different subsets. For example, in the case of CIFAR10 when the M-incorrect subset is $\sim 60\%$ of the total set, it weighs down the overall accuracy by over as much as 20% at times during training. Thus, identifying the hard examples with $\mathcal{M}$ incorrectness can help in training procedures like curriculum learning (Bengio et al., 2009), teacher forcing (Jordan, 1986; Pineda, 1988) and professor forcing (Lamb et al., 2016). Furthermore, weighing the training set examples on basis of whether they are correctly classified by $\mathcal{M}$ can provide a more balanced dataset for training.

4.3. Contigency Matrix Values

Observe the long tail decay of T_{00} and T_{11} and the very fast rise of T_{11} and T_{10} (bottom row Figure 1). This shows that the the slow learning of $\mathcal{M}$-incorrect examples is the major factor of slump in accuracy growth for deep networks after the often observed initial fast ascent. This observation can be used for iterating more on $\mathcal{M}$-incorrect *harder* data points after the initial phase and achieve faster convergence.

5. Conclusion

In this work, we track the training of deep networks relative to machine learning models and show results on analyzing the training trajectory relative to SVM and RF on three different datasets. Empirically, we observe that the during training the Deep Network quickly learns shallow classifiable *easy* examples first and then slowly learns the *hard* examples later on. Furthermore, we find that the notion of hardness of an example is largely independent of the model being used and can be evaluated reliably using a shallow learning model. This observation allows empirically to sort dataset in easy and hard categories that can improve network training. Lastly, we report the surprising observation that even a quite accurate trained Deep Network mis-classifies examples that a much smaller shallow network gets right which can be looked into in future work.

References

Arpit, D., Jastrzebski, S., Ballas, N., Krueger, D., Bengio, E., Kanwal, M. S., Maharaj, T., Fischer, A., Courville, A., Bengio, Y., et al. A closer look at memorization in deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 233–242. JMLR. org, 2017.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48. ACM, 2009.

Breiman, L. Random forests. *Machine learning*, 45(1): 5–32, 2001.

Cootes, T. F., Ionita, M. C., Lindner, C., and Sauer, P. Robust and accurate shape model fitting using random forest regression voting. In *European Conference on Computer Vision*, pp. 278–291. Springer, 2012.

Cortes, C. and Vapnik, V. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

Cuingnet, R., Rosso, C., Chupin, M., Lehéricy, S., Dormont, D., Benali, H., Samson, Y., and Colliot, O. Spatial regularization of svm for the detection of diffusion alterations associated with stroke outcome. *Medical image analysis*, 15(5):729–737, 2011.

Decoste, D. and Schölkopf, B. Training invariant support vector machines. *Machine learning*, 46(1-3):161–190, 2002.

Díaz-Uriarte, R. and De Andres, S. A. Gene selection and classification of microarray data using random forest. *BMC bioinformatics*, 7(1):3, 2006.

Feichtenhofer, C., Pinz, A., and Wildes, R. Spatiotemporal residual networks for video action recognition. In *Advances in neural information processing systems*, pp. 3468–3476, 2016.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Heisele, B., Ho, P., and Poggio, T. Face recognition with support vector machines: Global versus component-based approach. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 2, pp. 688–694. IEEE, 2001.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Jordan, M. Attractor dynamics and parallelism in a connectionist sequential machine. In *Proc. of the Eighth Annual Conference of the Cognitive Science Society (Erlbaum, Hillsdale, NJ), 1986*, 1986.

Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Lamb, A. M., Goyal, A. G. A. P., Zhang, Y., Zhang, S., Courville, A. C., and Bengio, Y. Professor forcing: A new algorithm for training recurrent networks. In *Advances In Neural Information Processing Systems*, pp. 4601–4609, 2016.

LeCun, Y. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

Liaw, A., Wiener, M., et al. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.

Pal, M. Random forest classifier for remote sensing classification. *International Journal of Remote Sensing*, 26(1): 217–222, 2005.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Pineda, F. J. Dynamics and architecture for neural computation. *Journal of Complexity*, 4(3):216–245, 1988.

Pradhan, S. S., Ward, W. H., Hacioglu, K., Martin, J. H., and Jurafsky, D. Shallow semantic parsing using support vector machines. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, 2004.

Saxe, A. M., Bansal, Y., Dapello, J., Advani, M., Kolchinsky, A., Tracey, B. D., and Cox, D. D. On the information bottleneck theory of deep learning. 2018.

Tishby, N. and Zaslavsky, N. Deep learning and the information bottleneck principle. In *2015 IEEE Information Theory Workshop (ITW)*, pp. 1–5. IEEE, 2015.

Yang, Z. R. Biological applications of support vector machines. *Briefings in bioinformatics*, 5(4):328–338, 2004.

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 2223–2232, 2017.

6. Appendix

6.1. Plots of $\mathcal{T}$ values for Random Forest

Figure 3 shows the plots of values of the contingency matrix $\mathcal{T}^i$ plotted against i (smoothed and scaled for plotting) for Random Forest. Referring to Table 1, the entries T_{00}, T_{01}, T_{10} and T_{11} are denoted by red, orange, green and blue respectively.

6.2. Training Details

The reported maximum accuracy (Table 2) results have been achieved with decaying the learning rate by

(a) a factor of 20 after 60, 120, 160 and 200 epochs for DenseNet121 on the CIFAR10 dataset
(b) a factor of 10 after 150 and 250 epochs for ResNet101 on the CIFAR100 dataset.

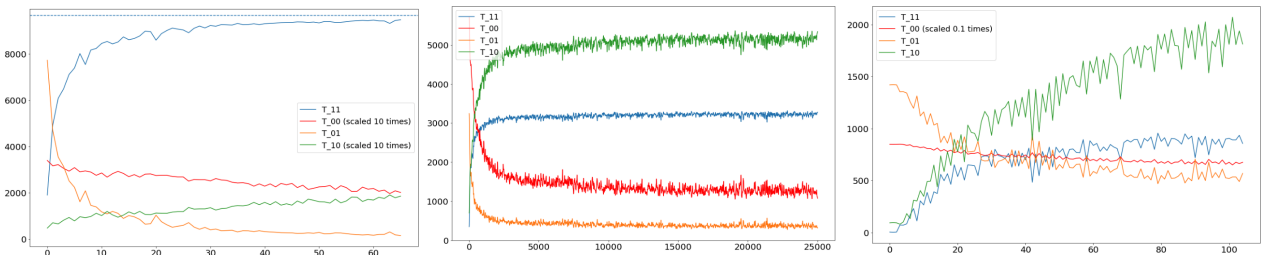


Figure 2. $\mathcal{T}^i$ plotted against i (smoothed and scaled for plotting).