Report: Improving CNN by Reusing Features Trained with Transductive Transfer Setting

Chetak Kandaswamy*, Luís Silva, and Luís Alexandre

Instituto de Engenharia Biomdica (INEB),Porto, Portugal,
NNIG Technical Report No. 2/2014,
Project Reusable Deep Neural Networks: Applications to Biomedical Data,
(PDTC/EIA-EIA/119004/2010)
http://paginas.fe.up.pt/~nnig

1 Transfer Learning Paradigm

The behavioral neuroscientist believe that the knowledge gained to distinguish an apple becomes useful when one tries to distinguish a orange or a pear. The study of transfer learning is inspired by this ability of human to a learn from a problem and able to reuse on a different but related problem. Reusing the knowledge of a network trained on source problem S to improve performance of target problem T to attain its goal function is called transfer learning and we use "$\Rightarrow$" symbol to denote transference. The various types of transfer learning approaches and settings are discussed in [1][2].

In Inductive transfer learning solves a more general problem before solving a more specific problem. In this case, distributions from source and target problems are related, $P(X_S) \approx P(X_T)$. In transductive transfer the machine trains on a specific problem to solve another specific problem. In this paper we are interested in transductive transfer problems with arbitrary distribution but not necessarily related to the distribution of the source problem $P(X_S) \neq P(X_T)$. In this paper we are interested in transductive transfer setting as it performs better than inductive transfer settings from Vladmir Vapnik et al. [12] and Chetak et al. [1].

Let design data set $X_{ds} = \{x_1, x_2, ..., x_n\}$ has n_{ds} distinct training instances. These n_{ds} instances are assumed to be random samples with probability distribution $P(X)$ from the classifier input X. The label set $\Omega = \{1, ..., c\}$ has c distinct class labels. The source and target problems may have complete or partial or no information about labels. Even when the information is available, the label set for the source Ω_S and target Ω_T may be same or different. Similarly the number of instances for the source n_S and target n_T may be equal or different.

The features (or filters) corresponds to a vector of weights and biases of neural network. The feature set $w = \left\{w^1, w^2, ..., w^K\right\}$ represent features of K

* This work was financed by FEDER funds through the *Programa Operacional Factores de Competitividade* COMPETE and by Portuguese funds through FCT Fundação para a Ciência e a Tecnologia in the framework of the project PTDC/EIA-EIA/119004/2010.

layers of the network. The unsupervised feature set is represented as $U(w)$ and supervised feature set is represented as $S(w)$.

2 Related Work

The Neocognitron, a seminal work of Fukushima [3] in early 1980's introduced Convolutional Neural Network (CNN). CNN is a self-organizing neural network which is unaffected by shift in position for pattern recognition. This work was later improved by LeCun [4] in 1998 by training multi-layer neural network with back propagation algorithm for gradient based learning. CNN algorithm outperformed all the algorithms till date on hand written digit recognition and many more image classification datasets and became a mile stone for object and speech recognition algorithms.

CNN belongs to hierarchal neural network whose convolutional layers [4] alternate with subsampling layers inspired by primary visual cortex of Wiesel and Hubel [5] in 1959 made up of simple and complex cells. CNN is better explained in three stages. The image processing stage, the alternating convolutional and subsampling stage and finally the classification stage. The image processing stage is a pre-processing stage of predefined filters that are kept fixed during training. The convolutional and subsampling are architectural ideas to ensure some degree of shift and distortion invariance. The convolution layer convolute the input with set of filters like Gabor filters or trained filters producing feature maps(Simple cells). These feature maps are further reduced by subsampling (Complex cells). Finally, feature or kernal size of convolution filters and subsampling are chosen such that the output maps of the last convolutional layer are downsampled to 1 pixel per map or fully connected layer and fed to classification stage. The depth of the CNN is a function of number of alternating convolutional and subsampling stage.

Max-pooling [6][7] is a form of non-linear down-sampling. Max-pooling is used instead of subsampling layer. Max-pooling partitions the input image into a set of non-overlapping rectangles and, for each such sub-region, outputs the maximum value. Scherer et al. [7] found that max-pooling can lead to faster convergence, select superior invariant features, and improve generalization. The Multi Column CNN [8] shows that averaging several CNN into multiple columns reduces the overall error rate. CNN using transfer learning approaches [14] perform better for latin and Chinese characters.

3 Architectures

Convolutional neural network: Given design set $X_{ds.source}$ and test set $X_{ts.source}$, we can design a classifier to train CNN [4] with baseline approach by applying Algorithm 1.

The convolutional network with filter size of the kernal as [20, 50] and max training epochs of 200. The learning rate of 0.1 is set with batch training of 100. All the experiments was done with 10 repetition.

Algorithm 1 Baseline approach for either SDA or CNN.

Given design set $X_{ds.source}$ and test set $X_{ts.source}$,

1. Randomly initialise a classifier network;
2. IF "Model == Stacked Denoising Autoencoder";
   Pre-train the network using $X_{ds.source}$;
   Fine-tune the network using $X_{ds.source}$;
3. IF "Model == Convolution Neural Network";
   Train (Fine-tune) the network using $X_{ds.source}$;
4. Test the network using $X_{ts.source}$, obtaining baseline classification error ε.

Stacked denoising Autoencoder: An Autoencoder is a simple neural network that reconstructs the original input from the hidden representations. It encodes the input to generate hidden representations and attempts to decode by reconstructing the original input X. A denoising Autoencoder is a variant of the autoencoder where corrupted version of the input is used to reconstruct the original input X. Stacking multiple denoising Autoencoder's one on top of another, gives the model the advantage of hierarchical features, hence it is called Stacked Denoising Autoencoders (SDA) [13].

Given design set $X_{ds.source}$ and test set $X_{ts.source}$, we can design a classifier to train SDA with baseline approach by applying Algorithm 1. Training a SDA in baseline approach is composed mainly of two stages [1]: an unsupervised pre-training stage followed by a supervised fine-tuning stage.

We used pre-training and fine-tuning learning rates of 0.001 and 0.1, respectively. The stopping criteria for pre-training was fixed to 40 epochs; stopping criteria for fine-tuning was set to a maximum of 1000 epochs. The number of neurons in three hidden layers and one output layer is pyramidal structure with $[16 \times 6^2, 16 \times 5^2, 16 \times 4^2, c]$ neurons.

4 Supervised Layer based Feature Transference

In Supervised Layer based Feature Transference approach we transfer the fine-tuned parameters of a model from the source to the target problem. Inspired by the human primary visual cortex has simple cells that correspond to low-level feature representations. The higher level visual cortex that has complex cells that correspond to higher-level feature representations [9]. This concept is extended to deep representation. We use low-level feature transference to transfer low-level representations and similarly we use high-level feature transference to transfer high-level representations, or both low and high-level feature transference depending on the target problem.

In this approach only in case of SDA the source features w_S are pre-trained $pretrain\,(w_S)$ until the K^{th} hidden layer. Then we fine-tune the already pre-trained features, $finetune\,(pretrain\,(w_S))$ with labeled source data using stochastic gradient descent and back-propagation.

In case of CNN we fine-tune the filters, $finetune\,(pretrain\,(w_S))$ with labeled source data using stochastic gradient descent and back-propagation.

Table 1: S-CNN feature transference approach and respective methods

Methods	Transference	Target problem[1]
reuse FT	$w_S \Rightarrow w_T$	$finetune\,(w_T)$
reuse L1+L2+L3	$w_S^1, w_S^2, w_S^3 \Rightarrow w_T^1, w_T^2, w_T^3$	$finetune\,(c_T)$
reuse L1+L3	$w_S^1, w_S^3 \Rightarrow w_T^1, w_T^3$	$finetune\,\left(w_T^2, c_T\right)$
reuse L2+L3	$w_S^2, w_S^3 \Rightarrow w_T^2, w_T^3$	$finetune\,\left(w_T^1, c_T\right)$
reuse L1+L2	$w_S^1, w_S^2 \Rightarrow w_T^1, w_T^2$	$finetune\,\left(w_T^3, c_T\right)$
reuse L3	$w_S^3 \Rightarrow w_T^3$	$finetune\,\left(w_T^1, {w_T}^3, c_T\right)$
reuse L2	$w_S^2 \Rightarrow w_T^2$	$finetune\,\left(w_T^1, w_T^3, c_T\right)$
reuse L1	$w_S^1 \Rightarrow w_T^1$	$finetune\,\left(w_T^2, w_T^3, c_T\right)$
Baseline[#]	-	-

[#] Baseline means traditional CNN, fully trained for the target problem without reusing anything. * Transfer only pre-trained parameters of the source problem. [1] Transfered parameters used for fine-tuning on the target problem

We use supervised layer based feature transference to choose which layer or layers to transfer. For example if we only need low-level features, we choose to transfer first layer parameters i.e., $w_S^1 \Rightarrow w_T^1$ which is listed as reuse L1 method in Table 2. This enables us to use deep transference of supervised features. Then we again fine-tune the entire model like a regular multi-layer perceptron with back-propagation except that at first layer we do not update the parameter, w_T^1 with the target problem dataset. In essence we are transferring the low-level features of source problem to the target problem. Similarly if we need both low-level and middle-level features to transfer both we choose to transfer both first and second layer parameters i.e., $w_S^1, w_S^2 \Rightarrow w_T^1, w_T^2$ which is listed as reuse L1+L2 method in Table 2. It is interesting to see that this enables various combinations to reuse supervised features for the target problem. In case of reuse FT method we reuse the fully trained supervised features of source problem and then refine-tune the entire network for the target problem. And for inductive transfer setting the reuse FT method cannot reuse the logistic regression layer as the label set for the source problem Ω_S with c_S classes is not equal to the target problem label set Ω_T with c_T classes.

5 Data and Experimental Results

We performed all our experiments on a computer with i7-377 (3.50GHz) 16GB RAM using Theano [11] a GPU compatible machine learning library on a GTX 770 GPU. The GPU paralle processing alows training both CNN's and SDA's deep neural networks with millions of neural connection, for very small learning rate, for large number of epochs, for very large datasets within several days.

Each of these experiments are repeated 10 times to increase confidence level of the results.

MNIST[1] and MADbase[2] are hand-written Latin and Arabic digits datasets respectively. Latin and Arabic datasets are representative names for the well-known MNIST and MADbase datasets of hand-written Latin and Arabic digits, respectively. The original Chars74k has 64 classes consisting of typed digits, lowercase and uppercase English language characters that was broken into three smaller datasets: Digits dataset with the 0-to-9 digits, the Lowercase dataset with the a-to-z lowercase letters and finally, the Uppercase dataset with the A-to-Z uppercase letters. All the three modified datasets are resized to 28×28 pixels from original 128×128 pixels image. The Latin-2 dataset is a modified version of MNIST to match the number of training and validation instances of the Lowercase datasets.

6 Transductive transfer: From digits to letters

Let us consider a problem of classifying images of English lowercase a-to-z by reusing fine-tuned features of English handwritten digits 0-to-9. In case of reuse L1, the average error rate of uppercase letters was significantly lower than the baseline. Similar results are obtained for the lowercase letters. Reusing single layer: L1, L2 or L3, we observe that the features of the lower layer leads to lower classification error. Reusing multiple layers: L1+L2, L2+L3, L1+L3, we observe that reusing L1+L3 perform better than the reusing L1+L2 for both uppercase and lowercase datasets. Reusing all three layers: L1+L2+L3 has degraded performs as the supervised features are well tuned for the source problem and fine-tuning only the logistic regression layer does not compensate for good features for the target problem. Thus reusing higher layer supervised features are not as good as reusing lower layer supervised features.

Table 2: Average Test Error for Transductive transfer setting

Source:	latin digits			
Target:	uppercase letters		lowercase letters	
$n_{ds.source}/c$:	1320	5000	1320	5000
SCNN Approach	Error	Error	Error	Error
reuse L1+L2+L3	5.96(0.13)	5.32(0.18)	6.13(0.13)	5.63(0.15)
reuse L1+L3	4.49(0.14)	4.24(0.10)	4.75(0.13)	4.57(0.09)
reuse L1+L2	3.61(0.12)	3.39(0.12)	3.83(0.06)	3.63(0.13)
reuse L3	4.30(0.13)	4.20(0.16)	4.62(0.18)	4.61(0.14)
reuse L2	3.54(0.14)	3.43(0.06)	3.72(0.11)	3.58(0.15)
reuse L1	3.43(0.11)	**3.35(0.09)**	**3.64(0.06)**	**3.56(0.11)**
Baseline	**3.42(0.10)**	3.42(0.10)	3.65(0.12)	3.65(0.12)

[1] http://yann.lecun.com/exdb/mnist/
[2] http://datacenter.aucegypt.edu/shazeem/

Algorithm 2 Experimental procedure.

Given design sets $X_{ds.source.full}$, $X_{ds.target.full}$ and test set $X_{ts.source}$, $X_{ts.target}$, For each data type (Latin handwritten digits, Arabic handwritten digits)

1. For each n_{ds} such that $\frac{n_{ds}}{c} \in [100, 250, 500, 1000, 1320, 2500, 5000]$,
   (a) Run the baseline approach;
   (b) Obtain $X_{ds.target}$ by randomly picking n_{ds} samples from $X_{ds.target.full}$;
   (c) For each *reuse LK approach* such that LK$\in [L1, L1 + L2]$,
      i. Fix LK^{th} layer of the network trained on $X_{ds.source}$;
      ii. Retrain the network using $X_{ds.target}$ except the LK^{th} layers;
      iii. Test the network using $X_{ts.target}$, obtaining classification error ε.

7 Importance of Sample size in Supervised Feature transference

From the previous result reusing L1, L2, L3, L1+L2, L2+L3, L1+L3, L1+L2+L3 we observe that reusing L1 and L1+L2 perform better other approaches for both uppercase and lowercase datasets. Thus we perform experiment to test the importance training sample size on the supervised feature transference.

In this section we describe the algorithm for the experimental procedure used.

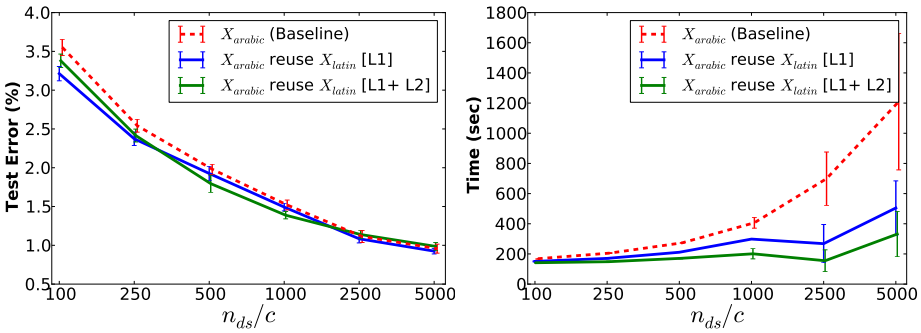

Fig. 1: Classification results on MAHDBase dataset (Arabic digits) for SCNN feature transference approach by reusing various layers, for different numbers n_{ds}/c of training samples per class. **Left:** Average classification test error rate. **Right:** Average time taken for classification.

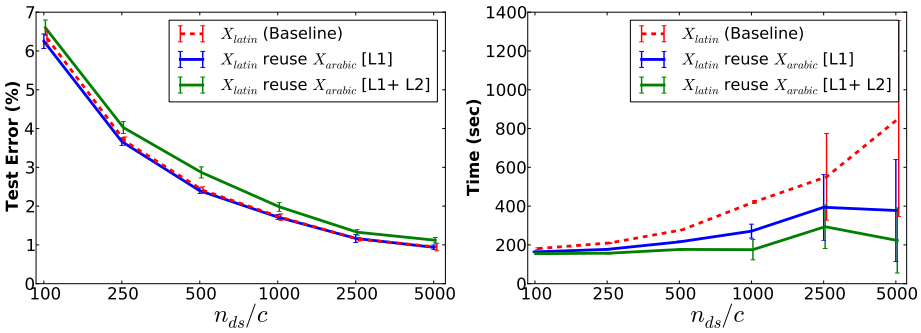

Fig. 2: Classification results on MNIST dataset (Latin digits) for SCNN feature transference approach by reusing various layers, for different numbers n_{ds}/c of training samples per class. **Left:** Average classification test error rate. **Right:** Average time taken for classification.

8 Classification Results and discussion

The transductive transfer setting does perform better than the inductive transfer setting. Thus we have conducted our experiments based on the transductive transfer settings. The conclusions and the results of transductive transfer learning [1] are used to continue experiments with CNN.

Table 3: Average Error by reusing latin-2 in transductive setting

Approaches	Lowercase **Test Error %**	Uppercase **Test Error %**
SDA Baseline [1]	4.95±0.16	5.01±0.27
SDA Baseline + reuse L1 (SSDA) [1]	4.72±0.17	4.72±0.18
SDA Baseline + pre-trained all layers (USDA) [1]	4.67±0.38	4.65±0.19
SDA Baseline + reuse FT (SSDA) [1]	4.57±0.08	4.58±0.19
CNN Baseline + reuse L1+L2 (SCNN)	3.83±0.06	3.61±0.12
CNN Baseline	3.65±0.12	**3.42±0.10**
CNN Baseline + reuse L1 (SCNN)	**3.64±0.06**	3.43±0.11

9 Conclusion

The supervised feature transference show better feature transference than the unsupervised feature transference of Stacked denoising autoencoder. Also looking at the layers of the convolution neural network the larger samples show lower feature transference. Thus the feature transference are more useful for the lower number of sampler per class.

References

1. C. Kandaswamy, J. Marques de S, L. M. Silva, L. A. Alexandre, J. M. Santos.: Improving Accuracy on Transductive Transfer Learning by Reusing SDA. (Under review), (2013)
2. S. J. Pan and Q. Yang.: A survey on transfer learning. Knowledge and Data Engineering, IEEE Transactions on, vol. 22, no. 10, pp. 1345–1359 (2010)
3. K. Fukushima.: Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics, vol. 36, no. 4, pp. 193–202 (1980)
4. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner.: Gradient-based learning applied to document recognition. In: proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324 (1998)
5. D. H. Hubel and T. N. Wiesel.: Receptive fields of single neurones in the cats striate cortex. The Journal of physiology, vol. 148, no. 3, pp. 574–591 (1959)
6. D. C. Cirecsan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber. Flexible, high performance convolutional neural networks for image classification. In: 22nd International joint conference on Artificial Intelligence. vol. 2, pp. 1237–1242 (2011)
7. D. Scherer, A. Muller, and S. Behnke.: Evaluation of pooling operations in convolutional architectures for object recognition. in Artificial Neural Network, 2010. Springer, pp. 92–101 (2010)
8. D. Ciresan, U. Meier, and J. Schmidhuber.: Multi-column deep neural networks for image classification. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 3642–3649 (2012)
9. Y. Bengio.: Learning deep architectures for AI. Foundations and Trends in Machine Learning, vol. 2, no. 1, pp. 1–127. now Publishers (2009)
10. T. de Campos, B. R. Babu, and M. Varma.: Character recognition in natural images. (2009)
11. J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio.: Theano: a CPU and GPU math expression compiler. In: Python for Scientific Computing Conference, vol. 4, (2010)
12. G. Alexander, V. Vovk, and V. Vapnik. "Learning by transduction. In: 14th conference on Uncertainty in artificial intelligence. Morgan Kaufmann Publishers Inc., (1998).
13. P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. J. Mach. Learn. Res.,vol. 11, pp. 33713408, (2010).
14. D. Ciresan, U. Meier, and J. Schmidhuber.: Transfer learning for Latin and Chinese characters with deep neural networks. In: 2012 IJCNN Conference, IEEE, (2012).