Train your classifier first: Cascade Neural Networks Training from upper layers to lower layers

TRAIN YOUR CLASSIFIER FIRST: CASCADE NEURAL NETWORKS TRAINING FROM UPPER LAYERS TO LOWER LAYERS

Shucong Zhang [1], *Cong-Thanh Do* [2], *Rama Doddipatla* [2], *Erfan Loweimi* [1], *Peter Bell*[1] *and Steve Renals* [1]

[1] Centre for Speech Technology Research, University of Edinburgh, Edinburgh, UK
[2] Toshiba Cambridge Research Laboratory, Cambridge, UK

ABSTRACT

Although the lower layers of a deep neural network learn features which are transferable across datasets, these layers are not transferable within the same dataset. That is, in general, freezing the trained feature extractor (the lower layers) and retraining the classifier (the upper layers) on the same dataset leads to worse performance. In this paper, for the first time, we show that the frozen classifier is transferable within the same dataset. We develop a novel top-down training method which can be viewed as an algorithm for searching for high-quality classifiers. We tested this method on automatic speech recognition (ASR) tasks and language modelling tasks. The proposed method consistently improves recurrent neural network ASR models on Wall Street Journal, self-attention ASR models on Switchboard, and AWD-LSTM language models on WikiText-2.

***Index Terms*—** top-down training, layer-wise training, general classifier, speech recognition, language model

1. INTRODUCTION

The lower layers (close to the input) of a deep neural network (DNN) can be interpreted as a feature extractor while the upper layers (close to the output) can be viewed as a classifier. The feature extractor learns general low-level features which could be transferable across datasets [1,2]. Transfer learning exploits this property – in general, the feature extractor, which is trained on a base dataset, is transferred to a target dataset with the classifier retrained on the target dataset [3–6]. Since the feature extractor learns general low-level features, the trained feature extractor should also be transferable within the same dataset. Therefore, retraining the classifier based on a frozen trained feature extractor using the same dataset should lead to little change of the model's performance. However, surprisingly, this retraining usually results in a performance drop [2].

In this work, experimentally, we observe that if the classifier is trained with more data, when transferred to unseen datasets, it leads to better generalisation (which is consistent with the transfer learning of feature extractors [3–6]). Furthermore, we observed retraining the feature extractor based on the frozen trained classifier within the same dataset usually does not lead to performance drops but results in better generalisation .

Based on these observations, We propose a *top-down*, layer-wise training method in which a network is trained by freezing the trained upper layers and retraining the lower layers in a cascade. We have applied this approach to train models for automatic speech recognition (ASR) using the clean Wall Street Journal (WSJ) dataset [7], the telephone conversation Switchboard (SWBD) [8] dataset and for language modelling (LM) using WikiText-2 dataset. The trained models include long short-term memory (LSTM) [9] based connectionist temporal classification (CTC) [10], LSTM based CTC-attention [11], Transformer [12] and AWD-LSTM [13]. We observed performance gains over the conventional training where all the layers are trained jointly; to the best of our knowledge, this is the first time that a layer-wise training method consistently outperforms conventional joint training across different domains, models and tasks (ASR and LM).

In summary, we present three novel contributions:

- Demonstration that the upper layers of neural networks are transferable within the same dataset;
- A top-down, layer-wise, training algorithm;
- Experiments indicating that top-down training outperforms conventional joint training across different domains and models.

2. RELATED WORK

Top-down training method can be viewed from different aspects. It is related to a variety of previous works, which we review in this section.

Layer-wise training without a joint fine-tuning stage has been investigated recently by a number of researchers [14–19]. However, unlike our top-down training algorithm, these training methods build DNNs in a bottom-up manner. In general, the reported experimental results do not indicate that these training methods surpass joint training. Belilovsky et al [18] and Nokland et al [19] have reported layer-wise training outperforms joint training for training convolutional neural

SZ was supported by a PhD Studentship funded by Toshiba Research Europe Limited.

networks for image classification. However, these methods are not effective for training networks with residual connections.

Zhang et al [20] propose to use the classifier trained on the clean data to train the feature extractor on noisy data, so the noisy feature extractor is forced to learn features which fit the clean classifier and thus the method has the effect of denoising. In this work we show the classifier is transferable not only across different datasets but also within the same dataset.

Both the proposed training method and dropout [21] reduce the size of the network during training. Dropout can be viewed as reducing the effective width of the network in a random manner, while our training method reduces the effective depth of the network by following a fixed schedule. When these two methods are combined, during training thinner and shallower sub-networks are trained while during testing, the model has the representation power of a wide and deep network. In our experiments, the combination of these two methods yields significant performance gains.

In gradient based optimization, the learning rate can either be dynamically tuned by the user or be tuned by optimization algorithms [22–26]. In top-down training, since the classifier is frozen, its learning rate can be viewed as 0 when retraining. Our experiments show that this procedure is compatible with popular methods of changing the learning rate dynamically.

3. TRANSFERABLE UPPER LAYERS

In this section, we empirically show: the quality of the frozen classifier decides the quality of the retrained model and in the joint training the quality of the classifier increases then decreases. We present the top-down training algorithm in Section 4 and present comprehensive comparisons between the top-down training method and joint training in Section 5.

We define *general classifier* by making an analogy to the concept of general feature extractor: if the classifier is more general, then training the feature extractor with respect to it on unseen data should lead to better generalisation compared to a classifier which is not general. We hypothesize that with more training data, the trained classifier becomes more general/has higher quality. To verify this, we conducted a series of end-to-end speech recognition experiments using the WSJ. We divide the WSJ training set into 5%, 10%, 20%, 40% and 80% size subsets, leaving 20% unseen training data. We train 4-layer BLSTM CTC models and the training stops if for 5 consecutive epochs there is no improvement on the development set (dev93). Other experimental setups follow [11]. We group the softmax layer and the topmost BLSTM layer as the classifier.

As shown in Table 1, with more training data, the model achieves better ASR performance. To test if the classifiers from each trained model become more general as the size of the training subsets increases, on the unseen 20% training set, we run experiments of training the feature extractor with these frozen trained classifiers. Figure 1 shows on the unseen training data, training with respect to the frozen classifier trained on larger subsets consistently yields lower character error rates

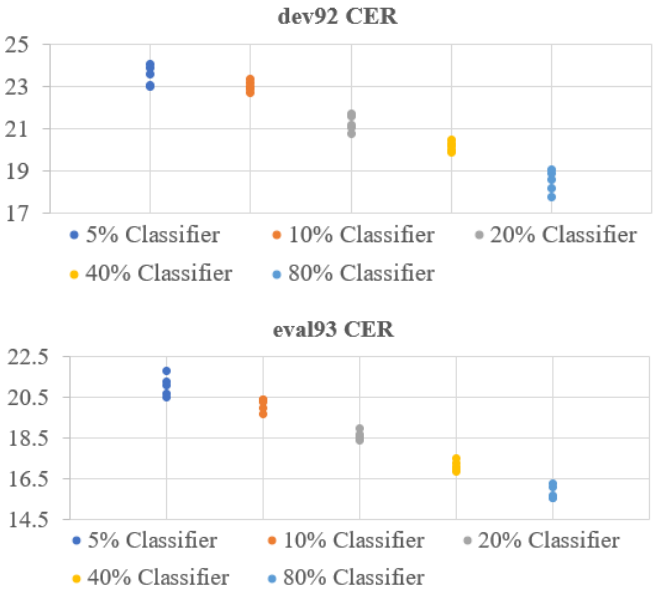


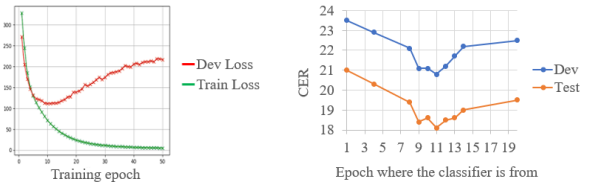
Fig. 1: The CERs of models trained on the 20% unseen data with frozen classifiers from trained models on different si284 subsets. The experiments are repeated 5 times with different random seeds.

Table 1: CERs of models trained on si284 subsets.

Training set	dev92/eval93 (CER)
5% train si284	34.4/32.9
10% train si284	28.3/26.1
20% train si284	22.5/19.9
40% train si284	17.5/14.7
80% train si284	13.2/10.6

(CERs). Thus, we conclude the classifiers trained on larger subsets are indeed more general/have higher quality, and training feature extractor with them leads to better generalisation.

We further investigate in the joint training, does the classifier become more general as the training proceeds? We train the CTC model on the 80% subset for a sufficiently large number of epochs. Then, we take the classifiers from the models among these training epochs and retrain the feature extractor based on them respectively on the 20% unseen data. Figure 2 shows on the 20% unseen data, the model trained with the frozen classifier which is from epoch 11 of the joint training on the 80% subset gives the lowest CERs. In the joint training on the 80% subset, the model around epoch 11 also gives the lowest development loss. On the 20% unseen data, the models trained with frozen classifiers which are from later epochs of the joint training on the 80% subset have increasing CERs. Also, in the joint training on the 80% subset, the development loss increases after epoch 11. Thus, the classifier firstly becomes general (low CERs when transferred to the unseen data), then it becomes overfitting and leads to worse generalisation (high CERs when transferred to the unseen data).



(a) Loss on 80% joint training (b) CERs on 20% unseen

Fig. 2: Train/dev loss for the joint training on the 80% subset and CERs of the models trained on the 20% unseen data with frozen classifiers taken from the epochs of the joint training.

If we freeze the classifier when it is general/of high-quality, we can prevent it from becoming overfitting and we can further train the feature extractor with the high-quality classifier. Based on this, we propose the top-down training algorithm, which is described in Section 4. The opposite strategy (freezing the feature extractor then retraining the classifier on the same dataset) usually leads to higher error rates [2].

4. TOP-DOWN TRAINING ALGORITHM

Based on the findings of Section 3, we develop a novel top-down, layer-wise training algorithm, which can be viewed as a process of searching for high-quality classifiers. The algorithm searches among models trained by conventional joint training along the epoch dimension and the layer dimension (layer index).

Algorithm 1 Greedy Layer-wise Top-down Training

Input: A trained n layer network M
$e \leftarrow$ validation error of M
for $i = 1$ **to** $n - 1$ **do**
  $M' \leftarrow M$
  Freeze top i layers (near output) of M'
  Reinitialize bottom $n - i$ layers of M'
  Retrain bottom $n - i$ layers of M'
  $e' \leftarrow$ validation error of M'
  if $e' > e$ **then**
    BREAK
  end if
  $M \leftarrow M'$
  $e \leftarrow e'$
end for
Output: A retrained n-layer network M

Searching the time/epoch dimension is useful since at the beginning of the training the classifier is under-trained and at the end of the training the classifier tends to be overfitting. We denote the model at epoch p as M^p. We search for the best classifier through all M^p.

Searching the layer dimension is essential since arbitrarily many upper layers can be grouped and viewed as the classifier (the feature extractor should at least have one layer). For a n-layer neural network M which is trained by the joint training, freezing the top i layers of it then reinitializing and retraining the bottom $n-i$ layers generates a new model $M_{i,n-i}$. For the model $M_{i,n-i}$, we can add newly retrained lower layers to the classifier and retrain the remaining lower layers. $M_{i,j,n-i-j}$ denotes a model trained by freezing the top $i+j$ layers of model $M_{i,n-i}$ followed by retraining the lower $n-i-j$ layers. This corresponds to a search of ordered sequences of natural numbers $(i, j, k, \cdots, r)$ such that $i + j + k + \cdots + r = n$; for a network M with three layers, the search would be across $M_{1,1,1}$, $M_{1,2}$ and $M_{2,1}$ (e.g., $M_{1,1,1}$ indicates train all the layers jointly;then freeze the topmost layer and retrain the bottom two layers; then freeze the top two layers and retrain the bottom most layer).

As the complete search along these two dimensions is expensive, we use a greedy search algorithm for layer-wise training (Algorithm 1), which uses a converged model for the epoch dimension. For the layer dimension, we freeze layers from top to bottom in a layer-by-layer manner (i.e. we only consider $M_{1,n-1}, M_{1,1,n-2}, \cdots M_{1,1,\cdots,1}$). We also use the validation set to halt the search. The complexity of this algorithm is $O(n)$, where n denotes the number of layers of the network. We omit the complexity of training the network since it is independent of the top-down training algorithm.

5. EXPERIMENTS

We apply the top-down training algorithm to train a range of neural networks on speech and text domains. We have used end-to-end speech recognition models. We test the end-to-end models rather than the conventional hybrid system since the end-to-end models have only one feature to output unit component, allowing an exact fair comparison between the joint and top-down training. For a hybrid system, better training of one component may not lead to a better performance of the entire system. For the same reason, we do not perform language model fusion or adaptation. All models were built using PyTorch [27].

5.1. Experiments on Speech Recognition

We employ our method to train CTC models on the full WSJ si284 training set. The experimental setup follows [11]. We used three different random seeds to build three baselines, and executed greedy top-down layer-wise training (Algorithm 1) for each baseline. For seed 1, we also tested freezing the lowest layer, followed by reinitializing and retraining the remaining upper layers.

Top-down training significantly reduces the CER for each of the baseline models (Table 2). For seed 1 and seed 2, the layer-wise training stops at the lowest layer. For seed 3 (where the proposed method brings the least improvement), the training procedure stops at the layer above the lowest layer (i.e. the final model is $M_{1,1,1,2}$). An additional experiment on seed 1, indicates that if we freeze the lowest layer and

Table 2: CERs for experiments on WSJ.

model	dev93/eval92 (CER)		
CTC	seed1	seed 2	seed 3
Baseline	12.4/9.7	12.6/10.4	12.6/10.2
+ top-down training	10.8/8.2	10.6/8.5	11.5/8.9
+ freeze lowest	13.1/10.5	_	_
Dropout 0.5			9.6/7.6
+ top-down training			**8.2/6.3**
CTC-attention			seed 1
Baseline A			8.0/6.2
+ freeze softmax layers from A			**7.3/5.4**
+ freeze softmax layers from B			7.5/5.7
VGG CTC-attention			seed 1
Baseline B			7.2/5.4
+ freeze softmax layers from B			7.0/5.2
+ freeze softmax layers from A			**6.7/5.2**
Previous works			
CTC DNN [28]			_ /10.0
CTC BLSTM [29]			_ /9.2
CTC-Attention [11]			11.3/7.4
+ pad silence [30]			7.8/5.8
+ deep encoder [31]			7.4/5.5

retrain the top layers, the accuracy of the model drops. This observation is also consistent with the findings of [2].

Using seed 3, we train a model with dropout probability 0.5 and then apply top-down training. We observe large gains in test accuracy over both the baseline and the model trained with dropout (38% and 17% relative gain, respectively). The combination of these two training methods performs impressively, with the CTC model giving CERs comparable to the joint CTC-attention baseline model, which is more flexible and has more capacity.

We apply the proposed training method to train hybrid CTC-attention models [11] on WSJ. The experimental setup of BLSTM CTC-attention follows [11] and the setup of VGG BLSTM CTC-attention follows [31]. We freeze only the softmax layer of the CTC part and the softmax layer of the attention based encoder-decoder part (the final model is $M_{1,n-1}$) due to the large number of experiments; and performance gains are already observed on $M_{1,n-1}$. Table 2 demonstrates that, to the best of our knowledge, top-down training results in state-of-the art character error rates for LSTM-based end-to-end models on WSJ, without language model fusion or adaptations.

We test the proposed training method on SWBD. We employ the Transformer [12] based CTC-attention model and the experimental setup follows [32]. We freeze only the softmax layer of the CTC part and the softmax layer of the Transformer due to large number of experiments. Table 3 shows the proposed method significantly reduces the word error rates (WERs) over the baseline, and gives noticeably better results compared to the previous works.

Table 3: WERs for experiments on SWBD

model	SWBD/Callhm (WER)
Baseline	9.0/18.1
+ freeze softmax layers	**8.6/17.2**
Previous works	
Transformer [32]	9.0/18.1
Very deep self-attention [33]	10.4/18.6
Multi-stride self-attention [34]	9.1/_

5.2. Experiments on Language Modelling

We apply the proposed method in training the AWD-LSTM language model [13] on WikiText-2 dataset. The architecture of the networks and the training procedure are same as in the previous work [13]. We run the retraining three times – with the same/different initial weights as the baseline. We freeze the topmost layer and retrain the bottom layers. The input/output embedding are tied in both of the joint training and the retraining (so the input embedding is also frozen during the retraining). This does not contradict the top-down approach since although tied with the input embedding, the output embedding is still the closest to the output and in general the gains of the tied embedding is from the output embedding [35]. The top-down method can be viewed as making the LSTM sufficiently trained by retraining the LSTM based on the trained and frozen word embedding.

As shown in Table 4, although the performance gains from the top-down training are not as large as the gains in previous experiments, the reduction of the perplexity is consistent. The proposed training method may implicitly overlap with some regularization techniques used in the training of AWD-LSTM. However, these regularization methods do not nullify the advantages of the top-down training.

Table 4: Perplexity of the AWD-LSTM language models on WikiText-2.

Model	Dev	Test
Baseline seed 1	68.7	65.6
+ freeze topmost layer seed 1	68.2	65.3
+ freeze topmost layer seed 2	68.2	**65.2**
+ freeze topmost layer seed 3	**68.1**	**65.2**
Previous work [13]	68.6	65.8

6. CONCLUSION

In this paper, we have shown that upper layers are in general transferable. Based on this, we develop a novel layer-wise top-down training method, which prevents the upper layers from overfitting. We demonstrate, for the first time, that layer-wise training can outperform conventional joint training across speech recognition and language modelling tasks. Future work will include reducing the complexity of the search.

7. REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *NeurIPS*, 2012.

[2] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *NeurIPS*, 2014.

[3] Y. Bengio, "Deep learning of representations for unsupervised and transfer learning," in *ICML workshop*, 2012.

[4] P. Swietojanski, A. Ghoshal, and S. Renals, "Unsupervised cross-lingual knowledge transfer in DNN-based LVCSR," in *IEEE SLT*, 2012.

[5] A. Ghoshal, P. Swietojanski, and S. Renals, "Multilingual training of deep neural networks," in *ICASSP*, 2013.

[6] J. Cho *et al.*, "Multilingual sequence-to-sequence speech recognition: architecture, transfer learning, and language modeling," in *IEEE SLT*, 2018.

[7] D. B. Paul and J. M. Baker, "The design for the Wall Street Journal-based CSR corpus," in *HLT'91*, 1992.

[8] J. Godfrey, E. Holliman, and J. McDaniel, "Switchboard: telephone speech corpus for research and development," in *ICASSP*, 1992.

[9] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, 1997.

[10] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *ICML*, 2006.

[11] S. Kim, T. Hori, and S. Watanabe, "Joint CTC-attention based end-to-end speech recognition using multi-task learning," in *ICASSP*, 2017.

[12] A. Vaswani *et al.*, "Attention is all you need," in *NeurIPS*, 2017.

[13] S. Merity, N. S. Keskar, and R. Socher, "Regularizing and optimizing LSTM language models," in *ICLR*, 2018.

[14] C. Hettinger *et al.*, "Forward thinking: Building and training neural networks one layer at a time," *arXiv:1706.02480*, 2017.

[15] K. Zhao, T. Matsukawa, and E. Suzuki, "Retraining: a simple way to improve the ensemble accuracy of deep neural networks for image classification," in *IEEE ICPR*, 2018.

[16] H. Mostafa, V. Ramesh, and G. Cauwenberghs, "Deep supervised learning using local errors," *Frontiers in neuroscience*, 2018.

[17] E. S. Marquez, J. S. Hare, and M. Niranjan, "Deep cascade learning," *IEEE Trans. Neural Netw. Learn. Syst*, 2018.

[18] E. Belilovsky, M. Eickenberg, and E. Oyallon, "Greedy layerwise learning can scale to ImageNet," in *ICML*, 2019.

[19] A. Nøkland and L. H. Eidnes, "Training neural networks with local error signals," in *ICML*, 2019.

[20] S. Zhang, C.-T. Do, R. Doddipatla, and S. Renals, "Learning noise invariant features through transfer learning for robust end-to-end speech recognition," in *ICASSP*, 2020.

[21] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *JMLR*, 2014.

[22] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural Networks*, 1999.

[23] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *JMLR*, 2011.

[24] M. D. Zeiler, "ADADELTA: an adaptive learning rate method," *arXiv:1212.5701*, 2012.

[25] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.

[26] T. Dozat, "Incorporating Nesterov momentum into Adam," in *ICLR 2016 workshop*, 2016.

[27] A. Paszke *et al.*, "Automatic differentiation in PyTorch," in *NeurIPS Workshop Autodiff*, 2017.

[28] A. Y. Hannun, A. L. Maas, D. Jurafsky, and A. Y. Ng, "First-pass large vocabulary continuous speech recognition using bi-directional recurrent DNNs," *arXiv:1408.2873*, 2014.

[29] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *ICML*, 2014.

[30] S. Zhang, E. Loweimi, P. Bell, and S. Renals, "Windowed attention mechanisms for speech recognition," in *ICASSP*, 2019.

[31] M. Delcroix, S. Watanabe, A. Ogawa, S. Karita, and T. Nakatani, "Auxiliary feature based adaptation of end-to-end ASR systems," in *INTERSPEECH*, 2018.

[32] S. Karita *et al.*, "A comparative study on transformer vs rnn in speech applications," in *ASRU*, 2019.

[33] N.-Q. Pham, T.-S. Nguyen, J. Niehues, M. Müller, and A. Waibel, "Very deep self-attention networks for end-to-end speech recognition," in *INTERSPEECH*, 2019.

[34] K. J. Han, J. Huang, Y. Tang, X. He, and B. Zhou, "Multi-stride self-attention for speech recognition." in *INTERSPEECH*, 2019.

[35] O. Press and L. Wolf, "Using the output embedding to improve language models," in *EACL*, 2017.