Learning a Deep Hybrid Model for Semi-Supervised Text Classification

Alexander G. Ororbia II, C. Lee Giles, David Reitter
College of Information Sciences and Technology
The Pennsylvania State University, University Park, PA
{ago109, giles, reitter}@psu.edu

Abstract

We present a novel fine-tuning algorithm in a deep hybrid architecture for semi-supervised text classification. During each increment of the online learning process, the fine-tuning algorithm serves as a top-down mechanism for pseudo-jointly modifying model parameters following a bottom-up generative learning pass. The resulting model, trained under what we call the *Bottom-Up-Top-Down* learning algorithm, is shown to outperform a variety of competitive models and baselines trained across a wide range of splits between supervised and unsupervised training data.

1 Introduction

Recent breakthroughs in learning expressive neural architectures have addressed challenging problems in domains such as computer vision, speech recognition, and natural language processing. This success is owed to the representational power afforded by deeper architectures supported by long-standing theoretical arguments (Hastad, 1987). These architectures efficiently model complex, highly varying functions via multiple layers of non-linearities, which would otherwise require very "wide" shallow models that need large quantities of samples (Bengio, 2012). However, many of these deeper models have relied on mini-batch training on large-scale, labeled data-sets, either using unsupervised pre-training (Bengio et al., 2007) or improved architectural components (such as activation functions) (Schmidhuber, 2015).

In an online learning problem, samples are presented to the learning architecture at a given rate (usually with one-time access to these data points), and, as in the case of a web crawling agent, most of these are unlabeled. Given this, batch training and supervised learning frameworks are no longer applicable. While incremental approaches such as co-training have been employed to help these models learn in a more update-able fashion (Blum and Mitchell, 1998; Gollapalli et al., 2013), neural architectures can naturally be trained in an online manner through the use of stochastic gradient descent (SGD).

Semi-supervised online learning does not only address practical applications, but it also reflects some challenges of human category acquisition (Tomasello, 2001). Consider the case of a child learning to discriminate between object categories and mapping them to words, given only a small amount of explicitly labeled data (the mother pointing to the object), and a large portion of unsupervised learning, where the child comprehends an adult's speech or experiences positive feedback for his or her own utterances regardless of their correctness. The original argument in this respect applied to grammar (e.g., Chomsky, 1980; Pullum & Scholz, 2002). While neural networks are not necessarily models of actual cognitive processes, semi-supervised models can show learnability and illustrate possible constraints inherent to the learning process.

The contribution of this paper is the development of the *Bottom-Up-Top-Down* learning algorithm for training a Stacked Boltzmann Experts Network (SBEN) (Ororbia II et al., 2015) hybrid architecture. This procedure combines our proposed top-down fine-tuning procedure for jointly modifying the parameters of a SBEN with a modified form of the model's original layer-wise bottom-up learning pass (Ororbia II et al., 2015). We investigate the performance of the constructed deep model when applied to semi-supervised text classification problems and find that our hybrid architecture outperforms all baselines.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 471–481,
Lisbon, Portugal, 17-21 September 2015.

2 Related Work

Recent successes in the domain of connectionist learning stem from the expressive power afforded by models, such as the Deep Belief Network (DBN) (Hinton et al., 2006; Bengio et al., 2007) or Stacked Denoising Autoencoder (Vincent et al., 2010), that greedily learn layers of stacked non-linear feature detectors, equivalent to levels of abstraction of the original representation. In a variety of language-based problems, deep architectures have outperformed popular shallow models and classifiers (Salakhutdinov and Hinton, 2009; Liu, 2010; Socher et al., 2011; Glorot et al., 2011b; Lu and Li, 2013; Lu et al., 2014). However, these architectures often operate in a multi-stage learning process, where a generative architecture is pre-trained and then used to initialize parameters of a second architecture that can be discriminatively fine-tuned (using back-propagation of errors or *drop-out*: Hinton et al., 2012). Several ideas have been proposed to help deep models deal with potentially uncooperative input distributions or encourage learning of discriminative information earlier in the process, many leveraging auxiliary models in various ways (Bengio et al., 2007; Zhang et al., 2014; Lee et al., 2014). A few methods for adapting deep architecture construction to an incremental learning setting have also been proposed (Calandra et al., 2012; Zhou et al., 2012). Recently, it was shown in (Ororbia II et al., 2015) that deep hybrid architectures, or multi-level models that integrate discriminative and generative learning objectives, offer a strong viable alternative to multi-stage learners and are readily usable for categorization tasks.

For text-based classification, a dominating model is the support vector machine (SVM) (Cortes and Vapnik, 1995) with many useful innovations to yet further improve its discriminative performance (Subramanya and Bilmes, 2008). When used in tandem with prior human knowledge to hand-craft good features, this simple architecture has proven effective in solving practical text-based tasks, such as academic document classification (Caragea et al., 2014). However, while model construction may be fast (especially when using a linear kernel), this process is costly in that it requires a great deal of human labor to annotate the training corpus. Our approach, which builds on that of (Ororbia II et al., 2015), provides a means for improving classification performance when labeled data is in scarce supply, learning structure and regularity within the text to reduce classification error incrementally.

3 A Deep Hybrid Model for Semi-Supervised Learning

To directly handle the problem of discriminative learning when labeled data is scarce, (Ororbia II et al., 2015) proposed deep hybrid architectures that could effectively leverage small amounts of labeled and large amounts of unlabeled data. In particular, the best-performing architecture was the Stacked Boltzmann Experts Network (SBEN), which is a variant of the DBN. In its construction and training, the SBEN design borrows many recent insights from efficiently learning good DBN models (Hinton et al., 2006) and is essentially a stack of building block models where each layer of model parameters is greedily modified while freezing the parameters of all others. In contrast to the DBN, which stacks restricted Boltzmann machines (RBM's) and is often used to initialize a deep multi-layer perceptron (MLP), the SBEN model is constructed by composing hybrid restricted Boltzmann machines and can be directly applied to the discriminative task in a single learning phase.

The hybrid restricted Boltzmann machine (HRBM) (Schmah et al., 2008; Larochelle and Bengio, 2008; Larochelle et al., 2012) building block of the SBEN is itself an extension of the RBM meant to ultimately perform classification. The HRBM graphical model is defined via parameters $\Theta = (\mathbf{W}, \mathbf{U}, \mathbf{b}, \mathbf{c}, \mathbf{d})$ (where $\mathbf{W}$ is the input-to-hidden weight matrix, $\mathbf{U}$ the hidden-to-class weight matrix, $\mathbf{b}$ is the visible bias vector, $\mathbf{c}$ is the hidden unit bias vector, and $\mathbf{d}$ is the class unit bias vector), and is a model of the joint distribution of a binary feature vector $\mathbf{x} = (x_1, \cdots, x_D)$ and its label $y \in \{1, \cdots, C\}$ that makes use of a latent variable set $\mathbf{h} = (h_1, \cdots, h_H)$. The model assigns a probability to the triplet (y,$\mathbf{x}$,$\mathbf{h}$) using:

$$p(y, \mathbf{x}, \mathbf{h}) = \frac{e^{-E(y,\mathbf{x},\mathbf{h})}}{Z}, \tag{1}$$

$$p(y, \mathbf{x}) = \frac{1}{Z} \sum_{\mathbf{h}} e^{-E(y,\mathbf{x},\mathbf{h})} \tag{2}$$

where Z is known as the partition function. The model's energy function is defined as

$$E(y, \mathbf{x}, \mathbf{h}) = -\mathbf{h}^T\mathbf{W}\mathbf{x} - \mathbf{b}^T\mathbf{x} - \mathbf{c}^T\mathbf{h} - \mathbf{d}^T\mathbf{e}_y - \mathbf{h}^T\mathbf{U}\mathbf{e}_y. \tag{3}$$

where $\mathbf{e}_y = (\mathbf{1}_{i=y})_{i=1}^{C}$ is the one-hot vector encoding of y. It is often not possible to compute $p(y, \mathbf{x}, \mathbf{h})$ or the marginal $p(y, \mathbf{x})$ due to the intractable normalization constant. However, exploiting the model's lack of intra-layer connections, block Gibbs sampling may be used to draw samples of the HRBM's latent variable layer given the current state of the visible layer and vice versa. This yields the following equations:

$$\begin{aligned} p(\mathbf{h}|y, \mathbf{x}) &= \prod_j p(h_j|y, \mathbf{x}), \\ p(h_j = 1|y, \mathbf{x}) &= \sigma(c_j + U_{jy} + \sum_i W_{ji}x_i) \end{aligned} \tag{4}$$

$$\begin{aligned} p(\mathbf{x}|\mathbf{h}) &= \prod_i p(x_i|\mathbf{h}), \\ p(x_i = 1|\mathbf{h}) &= \sigma(b_i + \sum_j W_{ji}h_j) \end{aligned} \tag{5}$$

$$p(y|\mathbf{h}) = \frac{e^{d_y + \sum_j U_{jy}h_j}}{\sum_{y^\star} e^{d_{y^\star} + \sum_j U_{jy^\star}h_j}} \tag{6}$$

where $\sigma(v) = 1/(1 + e^{-v})$. Classification may be performed directly with the HRBM by using its free energy function $F(y, \mathbf{x})$ to compute the conditional distribution

$$p(y|\mathbf{x}) = \frac{e^{-F(y,\mathbf{x})}}{\sum_{y^\star \in \{1,\cdots,C\}} e^{-F(y^\star,\mathbf{x})}} \tag{7}$$

where the free energy is formally defined as

$$-F(y, \mathbf{x}) = (d_y + \sum_j \psi(c_j + U_{jy} + \sum W_{ji}x_i)) \tag{8}$$

and ψ is the softplus activation function $\psi(v) = \log(1 + e^v)$.

To construct an *N*-layer SBEN (or *N*-SBEN), as was shown in (Ororbia II et al., 2015), one may learn a stack of HRBMs in one of two ways: (1) in a strict greedy, layer-wise manner, where layers are each trained in isolation on all of the data samples one at a time from the bottom-up; or (2) in a more relaxed disjoint fashion, where all layers are trained together on all of the data but still in a layer-wise bottom-up pass. To properly compute intermediate data representations during training and prediction in the SBEN, one must combine Equations 4 and 7. (The specific procedure for doing this can be found in the *computeLayerwiseStatistics* sub-routine in Algorithm 1.) This gives rise to the full SBEN architecture, which is depicted in Figure 1.

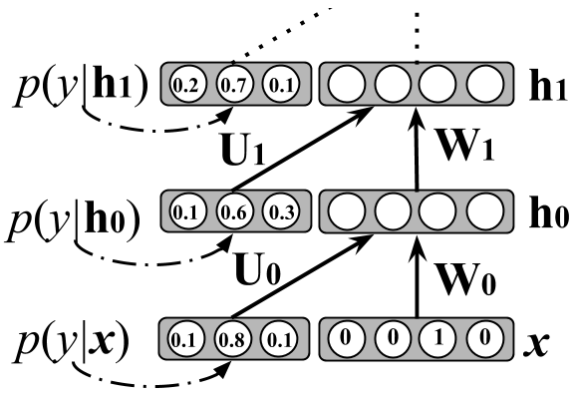

Figure 1: Architecture of the SBEN model. The model in feedforward mode can be viewed as a directed model, however, during training, connections are bi-directional.

3.1 Ensembling of Layer-Wise Experts

The SBEN may be viewed as a natural vertical ensemble of layer-wise "experts", where each layer maps latent representations to predictions, which differs from standard methods such as boosting (Schapire, 1990). Traditional feedforward neural models propagate data through the final network to obtain an output prediction y_t from a penultimate layer for a given $\mathbf{x}_t$. In contrast, this hybrid model is capable of a producing a label y_t^n at each level *n* for $\mathbf{x}_t$.

To vertically aggregate layer-wise expert outputs, we compute a simple mean predictor, $p(y|\mathbf{x})_{ensemble}$, as follows:

$$p(y|\mathbf{x})_{ensemble} = \frac{1}{N}\sum_{n=1}^{N} p(y|\mathbf{x})_n \tag{9}$$

This ensembling scheme provides a simple way to incorporate acquired discriminative knowledge of different levels of abstraction into the model's final prediction. We note that the SBEN's inherent layer-wise discriminative ability stands as an alternative to coupling helper classifiers (Bengio et al., 2007) or the "companion objectives" (Lee et al., 2014).

3.2 The Bottom-Up-Top-Down Learning Algorithm

With the SBEN architecture defined, we next present its simple two-step training algorithm, or the *Bottom-Up-Top-Down* procedure (*BUTD*), which combines a greedy, bottom-up pass with a subsequent top-down fine-tuning step. At every iteration of training, the model makes use of a single labeled sample (taken from an available, small labeled data subset) and an example from either a large unlabeled pool or a data-stream. We describe each of the two phases in Sections 3.2.1 and 3.2.2.

3.2.1 Bottom-Up Layer-wise Learning (BU)

The first phase of *N*-SBEN learning consists of a bottom-up pass where each layerwise HRBM can be trained using a compound objective function. Data samples are propagated up the model to the layer targeted for layer-wise training using the feedforward schema described above. Each HRBM layer of the SBEN is greedily trained using the frozen latent representations of the one below, which are generated by using the lower level expert's input and prediction. The loss function for each layer balances a discriminative objective $\mathcal{L}_{disc}$, a supervised generative objective $\mathcal{L}_{gen}$, and an unsupervised generative objective $\mathcal{L}_{unsup}$, fully defined as follows:

$$\begin{aligned}\mathcal{L}_{semi}(\mathcal{D}_{train}, \mathcal{D}_{unlab}) = \gamma\mathcal{L}_{disc}(\mathcal{D}_{train}) \\ +\alpha\mathcal{L}_{gen}(\mathcal{D}_{train}) \\ +\beta\mathcal{L}_{unsup}(\mathcal{D}_{unlab})\end{aligned} \quad (10)$$

Unlike generative pre-training of neural architectures (Bengio et al., 2007), the additional free parameters γ, α, and β offer explicit control over the extent to which the final parameters discovered are influenced by generative learning (Larochelle et al., 2012; Ororbia II et al., 2015). More importantly, the generative objectives may be viewed as providing data-dependent regularization on the discriminative learning gradient of each layer.

The objectives themselves are defined as:

$$\mathcal{L}_{disc}(\mathcal{D}_{train}) = -\sum_{t=1}^{|\mathcal{D}_{train}|} \log p(y|\mathbf{x}_t), \quad (11)$$

$$\mathcal{L}_{gen}(\mathcal{D}_{train}) = -\sum_{t=1}^{|\mathcal{D}_{train}|} \log p(y_t, \mathbf{x}_t), \text{ and} \quad (12)$$

$$\mathcal{L}_{unsup}(\mathcal{D}_{unlab}) = -\sum_{t=1}^{|\mathcal{D}_{unlab}|} \log p(\mathbf{x}_t) \quad (13)$$

where $\mathcal{D}_{train} = \{(\mathbf{x}_t, y)\}$ is the labeled training data-set and $\mathcal{D}_{unlab} = \{(\mathbf{u}_t)\}$ is the unlabeled training data-set. The gradient for $\mathcal{L}_{disc}$ may be computed directly, which follows the general form

$$\begin{aligned}\frac{\partial \log p(y_t|\mathbf{x})}{\partial\theta} = -\mathbb{E}_{\mathbf{h}|y_t,\mathbf{x}_t}\left[\frac{\partial}{\partial\theta}\left(\mathbb{E}(y_t, \mathbf{x}_t, \mathbf{h})\right)\right] \\ +\mathbb{E}_{y,\mathbf{h}|,\mathbf{x}}\left[\frac{\partial}{\partial\theta}\left(\mathbb{E}(y, \mathbf{x}, \mathbf{h})\right)\right]\end{aligned} \quad (14)$$

and can be calculated directly (see Larochelle et al., 2012 , for details) or through a form of *Dropping*, such as *Drop-Out* or *Drop-Connect* (Tomczak, 2013). The generative gradients themselves follow the form

$$\begin{aligned}\frac{\partial \log p(y_t, \mathbf{x})}{\partial\theta} = -\mathbb{E}_{\mathbf{h}|y_t,\mathbf{x}_t}\left[\frac{\partial}{\partial\theta}\left(\mathbb{E}(y_t, \mathbf{x}_t, \mathbf{h})\right)\right] \\ +\mathbb{E}_{y,\mathbf{x},\mathbf{h}}\left[\frac{\partial}{\partial\theta}\left(\mathbb{E}(y, \mathbf{x}, \mathbf{h})\right)\right]\end{aligned} \quad (15)$$

and, despite being intractable for any sample $(\mathbf{x}_t, y_t)$, may be approximated via the contrastive divergence procedure (Hinton, 2002). The intractable second expectation is replaced with a point estimate using a single Gibbs sampling step. To calculate the generative gradient for an unlabeled sample $\mathbf{u}$, a pseudo-label must be obtained by using a layer-wise HRBM's current estimate of $p(y|\mathbf{u})$, which can be viewed as a form of self-training or *Entropy Regularization* (Lee, 2013). The online procedure for computing the generative gradient (either labeled or unlabeled example) for a single HRBM can be found in Ororbia et al., (2015).

Setting the coefficients that control learning objective influences can lead to different model configurations (especially with respect to γ) as well as impact the gradient-based training of each model layer (i.e., α and β). In this paper, we shall explore two particular configurations, namely 1) by setting $\gamma = 0$ and $\alpha = 1$, which leads to constructing a purely generative model of $\mathcal{D}_{train}$ and

Algorithm 1 Top-down fine-tuning of an *N*-SBEN (*ensemble back-propagation*). Note that "$\cdot$" indicates a Hadamard product, ξ is an error signal vector, the prime superscript indicates a derivative (i.e., σ' means derivative function of the sigmoid), and $\widehat{\mathbf{z}}$ is the symbol for linear pre-activation values.

Input: $(\mathbf{x}_t, y_t) \in \mathcal{D}$, learning rate λ and model parameters $\Theta = \{\Theta_1, \Theta_2, ..., \Theta_N\}$
function FINETUNEMODEL($(\mathbf{x}_t, y_t)$, λ, Θ)
  $\Omega \leftarrow \emptyset$, $\mathbf{x}_n \leftarrow \mathbf{x}_t$, $y_n \leftarrow \emptyset$ ▷ Initialize list of layer-wise model statistics & variables
  // Conduct feed-forward pass to collect layer-wise statistics
  for $\Theta_n \in \Theta$ **do**
    $(\mathbf{h}_n, \widehat{\mathbf{z}}_n, y_n^h, \mathbf{x}_n) \leftarrow$ COMPUTELAYERWISESTATISTICS$(\mathbf{x}_n, \Theta_n)$
    $\Omega_n \leftarrow (\mathbf{h}_n, \widehat{\mathbf{z}}_n, y_n^h, \mathbf{x}_n)$, $\mathbf{x}_n \leftarrow \mathbf{h}_n$, $y_n \leftarrow y_n^h$
  // Conduct error back-propagation pass to adjust layer-wise parameters
  $\xi_l \leftarrow \emptyset$
  for $l \leftarrow N, l--,\ while\ l \geq 1$ **do**
    $(\mathbf{h}_l, \widehat{\mathbf{z}}_l, y_l^h, \mathbf{x}_l) \leftarrow \Omega[l]$ ▷ Grab relevant statistics for layer *l* of model
    if $i = N$ **then**
      $(\bigtriangledown_{disc}, \xi_l) \leftarrow$ COMPUTEDISCIMINATIVEGRADIENT$(y_t, \mathbf{x}_l, \emptyset, \mathbf{h}_n, \widehat{\mathbf{z}}, \Theta_l)$
    else
      $\xi_l \leftarrow \xi_l \cdot \sigma'(\widehat{\mathbf{z}}_l)$
      $(\bigtriangledown_{disc}, \xi_l) \leftarrow$ COMPUTEDISCIMINATIVEGRADIENT$(y_t, \mathbf{x}_l, \xi_l, \mathbf{h}_n, \widehat{\mathbf{z}}, \Theta_l)$
    $\Theta_n \leftarrow \Theta_n - \lambda(\bigtriangledown_{disc})$
function COMPUTELAYERWISESTATISTICS($\mathbf{x}_t, \Theta_n$)
  $y_t^h \leftarrow p(y_t|\mathbf{x}_t, \Theta_n)$ ▷ Equation 7 under the layerwise model
  $\widehat{\mathbf{z}} \leftarrow \mathbf{c} + W\mathbf{x}_t + U\mathbf{e}_{y_t}$ ▷ Can re-use $\widehat{\mathbf{z}}$ to perform next step
  $\mathbf{h}_t \sim p(\mathbf{h}|y_t^h, \mathbf{x}_t, \Theta_n)$ ▷ Equation 4 under the layerwise model
  return $(\mathbf{h}_t, \widehat{\mathbf{z}}, y_t^h, \mathbf{x}_t)$
function COMPUTEDISCIMINATIVEGRADIENT($y_t, \mathbf{x}_l, \xi_l, \mathbf{h}_n, \widehat{\mathbf{z}}, \Theta_l$)
  $\mathbf{o} \leftarrow p(y|\mathbf{h}_n, \Theta_l)$, $\xi \leftarrow softmax'(\mathbf{o}) \cdot -(y_t/\mathbf{o})$
  $\bigtriangledown_U \leftarrow \xi\mathbf{h}_n^T$, $\bigtriangledown_d \leftarrow \xi$, $\xi \leftarrow U\xi$, $\xi \leftarrow \xi \cdot \sigma'(\widehat{\mathbf{z}})$
  if $\xi_l \neq \emptyset$ **then**
    $\xi \leftarrow \xi \cdot \xi_l$
  $\bigtriangledown_W \leftarrow \xi\mathbf{x}_l^T$, $\bigtriangledown_c \leftarrow \xi$, $\bigtriangledown_b \leftarrow 0$, $\bigtriangledown_U \leftarrow \bigtriangledown_U + (\xi\mathbf{e}_{y_t}^T)$, $\xi \leftarrow W^T\xi$
  return $(\bigtriangledown \leftarrow (\bigtriangledown_W, \bigtriangledown_U, \bigtriangledown_b, \bigtriangledown_c, \bigtriangledown_d), \xi)$

$\mathcal{D}_{unsup}$, and 2) by setting $\gamma = 1$ with α freely varying (which recovers the model of Ororbia et al., 2015). In both scenarios, β is allowed to vary as a user-defined hyper-parameter. The second setting of γ allows for training the SBEN directly with only the bottom-up phase defined in this section. However, if the first setting is used, a second phase may be used to incorporate a top-down fine-tuning phase. A bottom-up pass simply entails computing this compound gradient for each layer of the model for 1 or 2 samples per training iteration. Notice that the first scenario reduces the number of hyper-parameters to explore in model selection, requiring only an appropriate value for β to be found.

3.2.2 Top-Down Fine-tuning (TD)

Although efficient, the bottom-up procedure described above is greedy, which means that the gradients are computed for each layer-wise HRBM independent of gradient information from other layers of the model. One way we propose to introduce a degree of joint training of parameters is to incorporate a second phase that adjusts the SBEN parameters via a modified form of back-propagation. Such a routine can further exploit the SBEN's multiple predictors (or entry points) where additional error signals may be computed and aggregated while signals are reverse-propagated down the network. We hypothesize that holistic fine-tuning ensures that discriminative information is incorporated into the generative

Algorithm 2 The *Bottom-Up-Top-Down* training procedure for learning an *N*-SBEN.

Input: $(\mathbf{x}_t, y_t) \in \mathcal{D}_{train}$, $(\mathbf{u}_t) \in \mathcal{D}_{unlab}$, rates λ & β, $\bar{p}$, & parameters $\Theta = \{\Theta_1, \Theta_2, ..., \Theta_N\}$
function BOTTOMUPTOPDOWN($(y_t, \mathbf{x}_t, \mathbf{u}_t, \lambda, \beta, \Theta)$
  APPLYBOTTOMUPPASS($y_t, \mathbf{x}_t, \mathbf{u}_t, \lambda, \gamma = 0, \alpha = 1, \beta, \Theta$) ▷ See (Ororbia II et al., 2015)
  // Up to two calls can be made to the top-down tuning routine
  FINETUNEMODEL($\mathbf{x}_t, y_t, \lambda, \Theta$) ▷ See Algorithm 1 for details
  $\mathbf{v}_t \leftarrow p_{ensemble}(y|\mathbf{x}, \Theta_n)$ ▷ Calculate pseudo-label probability using Equation 9
  if $max[\mathbf{v}_t] > \bar{p}$ **then**
    $\mathbf{v}_t \leftarrow$ TOONEHOT($\mathbf{v}_t$) ▷ Convert to 1-hot vector using argmax of model conditionals
    FINETUNEMODEL($\mathbf{u}_t, \mathbf{v}_t, \lambda, \Theta$)

features being constructed in the bottom-up learning step. Furthermore, errors from experts above are propagated down to lower layers, which were initially frozen during the greedy, bottom-up training phase.

Fine-tuning in the context of training an SBEN is different from using a pre-trained MLP that is subsequently fine-tuned with back-propagation. First, since the SBEN is a more complex architecture than an MLP, pre-initializing an MLP would be insufficient given that one would be tossing potentially useful information stored in the SBEN's class filters (and corresponding class bias vectors) of each layer-wise expert (i.e., U and $\mathbf{d}$). Second, merely using the SBEN as an intermediate model ignores the fact the SBEN can already perform classification directly. To avoid losing such information and to fully exploit the model's predictive ability, we adapt the back-propagation algorithm for training MLP's to operate on the SBEN, which we shall call *ensemble back-propagation* since the fine-tuning method propagates error derivatives down the network from many points of entry. Ensemble back-propagation is described in Algorithm 1.

With this second online training step, the *Bottom-Up-Top-Down* (*BUTD*) training algorithm for fully training an SBEN proceeds with a single bottom-up modification step followed by a single top-down joint fine-tuning step using the *ensemble back-propagation* procedure defined in Algorithm 1 for each training time step. A full top-down phase can consist of up to two calls to the *ensemble back-propagation* procedure. One is used to jointly modify the SBEN's parameters with respect to the sample taken from $\mathcal{D}_{train}$. A second one is potentially needed to tune parameters with respect to the sample drawn from $\mathcal{D}_{unlab}$. For the unlabeled sample, if the highest class probability assigned by the SBEN (using Equation 9) is greater than a pre-set threshold (i.e., $max[p_{ensemble}(y|\mathbf{u})] > \bar{p}$), a pseudo-label is created for that sample by converting the model's mean vector to a 1-hot encoding. The probability threshold $\bar{p}$ for the potential second call to the *ensemble back-propagation* routine allows us to incorporate a tunable form of pseudo-labeling (Lee, 2013) into the *Bottom-Up-Top-Down* learning algorithm.

The high-level view of the *BUTD* learning algorithm is depicted in Algorithm 2.

4 Experimental Results

We investigate the viability of our deep hybrid architecture for semi-supervised text categorization. Model performance was evaluated on the WebKB data-set [1] and a small-scale version of the 20NewsGroup data-set [2].

The original WebKB collection contains pages from a variety of universities (Cornell, Texas, Washington, and Wisconsin as well as miscellaneous pages from others). The 4-class classification problem we defined using this data-set was to determine if a web-page could be identified as one belonging to a Student, Faculty, Course, or a Project, yielding a subset of usable 4,199 samples. We applied simple pre-processing to the text, namely stop-word removal and stemming, chose to leverage only the k most frequently occurring terms (this varied across the two experiments), and binarized the document low-level representation (only 1 page vector was discarded due to presence of 0 terms). The 20NewsGroup data-set, on the other hand, contained 16242 total samples and was already pre-processed, containing 100 terms, binary-occurrence low-level representation, with

[1] The exact data-set we used can be found and downloaded at http://www.cs.cmu.edu/afs/cs/project/theo-20/www/data/

[2] The exact data-set we used can be found and downloaded at http://www.cs.nyu.edu/roweis/data.html.

tags for the four top-most highest level domains or meta-topics in the newsgroups array.

For both data-sets, we evaluated model generalization performance using a stratified 5-fold cross-validation (CV) scheme. For each possible train/test split, we automatically partitioned the training fold into separate labeled, unlabeled, and validation subsets using stratified random sampling without replacement. Generalization performance was evaluated by estimating classification error, average precision, average recall, and average F-Measure, where F-Measure was chosen to be the harmonic mean of precision and recall, $F1 = 2(\text{precision} \cdot \text{recall})/(\text{precision} + \text{recall})$.

4.1 Model Designs

We evaluated the BUTD version of our model, the *3-SBEN,BUTD*, as described in Algorithm 2. For simplicity, the number of latent variables at each level of the SBEN was held equal to the dimensionality of the data (i.e., a complete representation). We compared this model trained with *BUTD* against a version utilizing only the bottom-up phase (*3-SBEN,BU*) as in Ororbia et al. (2015). Both SBEN models contained 3 layers of latent variables.

We compared against an array of baseline classifiers. We used our implementation of an incremental version of Maximum Entropy, or *MaxEnt-ST* (which, as explained in Sarikaya et al., 2014, is equivalent to a softmax classifier). Furthermore, we used our implementation of the Pegasos algorithm (*SVM-ST*) (Shalev-Shwartz et al., 2011) which was extended to follow a proper multi-class scheme (Crammer and Singer, 2002). This is the online formulation of the SVM, trained via sub-gradient descent on the primal objective followed by a projection step (for simplicity, we opted to using a linear-kernel). Additionally, we implemented a semi-supervised Bernoulli Naive Bayes classifier (*NB-EM*) trained via Expectation-Maximization as in (Nigam et al., 1999). We also compared our model against the HRBM (Larochelle and Bengio, 2008) (effectively a single layer SBEN), which serves as a powerful, non-linear shallow classifier in of itself, as well as a 3-layer sparse deep Rectifier Network (Glorot et al., 2011a), or *Rect*, composed of leaky rectifier units.

All shallow classifiers (except *NB-EM* and the HRBM) were extended to the semi-supervised setting by leveraging a simple self-training scheme in order to learn from unlabeled data samples. The self-training scheme entailed using a classifier's estimate of $p(y|\mathbf{u})$ for an unlabeled sample and, if $max[p(y|\mathbf{u})] > \bar{p}$, we created a 1-hot proxy encoding using the *argmax* of model's predictor, where $\bar{p}$ is a threshold meta-parameter. Since we found this simple pseudo-labeling approach, similar in spirit to (Lee, 2013), to improve the results for all classifiers, and thus we report all results utilizing this scheme. [3] All classes of models (*SBEN*, *HRBM*, *Rect*, *SVM-ST*, *MaxEnt-ST*, *NB-ST*) were subject to the same model selection procedure described in the next section.

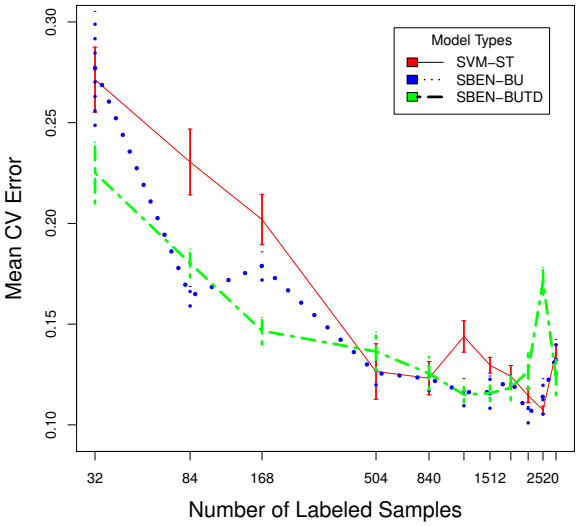

Figure 2: Mean CV generalization performance as a function of labeled sample subset size (using 200 features).

4.2 Model Selection

Model selection was conducted using a parallelized multi-setting scheme, where a configuration file for each model was specified, describing a set of hyper-parameter combinations to explore (this is akin to a course-grained grid search, where the points of model evaluation are set manually a priori). For the SBEN's, we varied the learning rate ($[0.01, 0.25]$) and β coefficient ($[0.1, 1.0]$) and

[3] All model implementations were computationally verified for correctness when applicable. Since most discriminative objectives followed a gradient descent optimization scheme and could be realized in an automatic differentiation framework, we checked gradient validity via finite difference approximation.

Table 1: WEBKB categorization results on 1% of the training data labeled (8 examples per class), rest unlabeled (i.e., 5-fold means with standard error of the mean, 250 features).

	Error	**Precision**	**Recall**	**F1-Score**
NB-EM	0.369 ± 0.039	0.684 ± 0.022	0.680 ± 0.028	0.625 ± 0.043
MaxEnt-ST	0.402 ± 0.026	0.623 ± 0.025	0.593 ± 0.015	0.583 ± 0.020
SVM-ST	0.342 ± 0.020	0.663 ± 0.010	0.665 ± 0.014	0.644 ± 0.015
HRBM	0.252 ± 0.023	0.740 ± 0.019	0.765 ± 0.016	0.741 ± 0.021
3-Rect	0.328 ± 0.020	0.673 ± 0.017	0.680 ± 0.021	0.654 ± 0.023
3-SBEN,BU	0.239 ± 0.015	0.754 ± 0.014	0.780 ± 0.016	0.754 ± 0.015
3-SBEN,BUTD	$\mathbf{0.210 \pm 0.011}$	$\mathbf{0.786 \pm 0.009}$	$\mathbf{0.784 \pm 0.014}$	$\mathbf{0.777 \pm 0.012}$

Table 2: 20NewsGroup data-set categorization results on 1% of the training data labeled (8 examples per class), rest unlabeled (i.e., 5-fold means with standard error of the mean).

	Error	**Precision**	**Recall**	**F1-Score**
NB-EM	0.275 ± 0.006	0.7176 ± 0.010	0.6685 ± 0.010	0.6697 ± 0.010
MaxEnt-ST	0.335 ± 0.005	0.643 ± 0.007	0.643 ± 0.007	0.639 ± 0.007
SVM-ST	0.346 ± 0.008	0.669 ± 0.016	0.644 ± 0.012	0.634 ± 0.011
HRBM	0.284 ± 0.006	0.706 ± 0.012	0.699 ± 0.009	0.696 ± 0.008
3-Rect	0.318 ± 0.009	0.661 ± 0.011	0.661 ± 0.012	0.657 ± 0.011
3-SBEN,BU	0.270 ± 0.006	0.715 ± 0.009	0.714 ± 0.009	0.710 ± 0.007
3-SBEN,BUTD	$\mathbf{0.256 \pm 0.007}$	$\mathbf{0.732 \pm 0.005}$	$\mathbf{0.727 \pm 0.006}$	$\mathbf{0.725 \pm 0.006}$

experimented with stochastic and mean-field versions of the models [4] (we found that mean-field did slightly better for this experiment and thus report the performance of this model in this paper). The HRBM's meta-parameters were tuned using a similar set-up to (Larochelle et al., 2012) with learning rate varied in ($[0.01, 0.25]$), α in ($[0.1, 0.5]$), and β in ($\{0.01, 0.1\}$). For the *SVM-ST* algorithm, we tuned its slack variable λ, searching in the interval $[0.0001, 0.5]$, for *MaxEnt-ST* its learning rate in $[0.0001, 0.1]$, and for $\bar{p}$ of all models (shallow and deep) that used pseudo-labeling we searched the interval $[0.1, 1.0]$. All models of all configurations were trained for a 10,000 iteration sweep incrementally on the data and the model state with lowest validation error for that particular run was used. The *SBEN*, *HRBM*, and *Rect* models were also set to use a momentum term of 0.9 (linearly increased from 0.1 in the first 1000 training iterations) and the *Rect* model made use of a small *L1* regularization penalty to encourage additional hidden sparsity. For a data-set like the 20NewsGroup, which contained a number of unlabeled samples greater than training iterations, we view our schema as simulating access to a data-stream, since all models had access to any given unlabeled example only once during a training run.

[4] Mean-field simply means no sampling steps were taken after computing probability vectors, or "means" in any stage of the computation.

4.3 Model Performance

We first conducted an experiment, using the WebKB data-set, exploring classification error as a function of labeled data subset cardinality (Figure 2). In this setup, we repeated the stratified cross-fold scheme for each possible labeled data subset size, comparing the performance of the SVM model against 3-SBEN,BU (blue dotted curve) and 3-SBEN,*BUTD* (green dash-dotted curve). We see that as the number of labeled examples increases (which entails greater human annotation effort) all models improve, nearly reaching 90% accuracy. However, while the performance difference between models becomes negligible as the training set becomes more supervised, as expected, it is in the less scarce regions of the plot we are interested in. We see that for small proportions, both variants of the SBEN outperform the SVM, and furthermore, the SBEN trained via full *BUTD* can reach lower error, especially for the most extreme scenario where only 8 labeled examples per class are available. We notice a bump in the performance of *BUTD* as nearly the whole training set becomes labeled and posit that since the *BUTD* involves additional pseudo-

Table 3: Top-most words that the SBEN (*BUTD*) model associates with the 4 NewsGroup meta-topics.

Meta-Topic	Associated Terms
*comp.**	$windows, graphics, card, driver, scsi, dos, files, display$
*rec.**	$players, hockey, season, nhl, team, league, baseball, games$
*sci.**	$orbit, shuttle, space, earth, mission, nasa, moon, doctor$
*talk.**	$jews, christian, religion, jesus, bible, war, israel, president$

labeling steps (as in the top-down phase), there is greater risk of reinforcing incorrect predictions in the pseudo-joint [5] tuning of layerwise expert parameters. For text collections where most of the data is labeled and unlabeled data is minimal, only a simple bottom-up pass is needed to learn a good hybrid model of the data.

The next set of experiments was conducted with only 1% of the training sets labeled. We observe (Tables 1 and 2) that our deep hybrid architecture trained via *BUTD* outperforms all other models with respect to all performance metrics. While the SBEN trained with simply an online bottom-up performs significantly better than the SVM model, we note a further reduction of error using our proposed *BUTD* training procedure. The additional top-down phase serves as a mechanism for unifying the layer-wise experts, where error signals for both labeled and pseudo-labeled examples increase agreement among all model layer experts.

For the 20NewsGroup data-set, we conducted a simple experiment to uncover some of the knowledge acquired by our model with respect to the target categorization task. We applied the mechanism from (Larochelle et al., 2012) to extract the variables that are most strongly associated with each of the clamped target variables in the lowest layer of a *BUTD*-trained SBEN. The top-scored terms associated with each class variable are shown in Table 3, using the 10 hidden nodes most highly triggered by the clamped class node, in a model trained on all of the 20NewsGroup data using a model configuration determined from CV results for the 20NewsGroup data-set reported in the paper. Since the SBEN is a composition of layer-wise experts each capable of classification, we note that this procedure could be applied to each level to uncover which unobserved variables are most strongly associated with each class target. We speculate that this could serve the basis for uncovering the model's underlying learnt hierarchy of the data and be potentially used for knowledge extraction, a subject for future work in analyzing black box neural models such as our own.

[5] We use the phrase "pseudo-joint" to differentiate a model that has all its parameters trained jointly from our own, where only the top-down phase of *BUTD* introduces any form of joint parameter modification.

5 Conclusions

We presented the *Bottom-Up-Top-Down* procedure for training the Stacked Boltzmann Experts Network, a hybrid architecture that balances both discriminative and generative learning goals, in the context of semi-supervised text categorization. It combines a greedy, layer-wise bottom-up approach with a top-down fine-tuning method for pseudo-joint modification of parameters.

Models were evaluated using two text corpora: WebKB and 20NewsGroup. We compared results against several baseline models and found that our hybrid architecture outperformed the others in all settings investigated. We found that the SBEN, especially when trained with the full *Bottom-Up-Top-Down* learning procedure could in some cases improve classification error by as much 39% over the Pegasos SVM, and nearly 17% over the HRBM, especially when data is in very limited supply. While we were able to demonstrate the viability of our hybrid model when using only simple surface statistics of text, future work shall include application of our models to more semantic-oriented representations, such as those leveraged in building log-linear language models (Mikolov et al., 2013).

Acknowledgments

A.G.O. acknowledges support from The Pennsylvania State University and the National Science Foundation (DGE-1144860). D.R. acknowledges support from the National Science Foundation (SES-1528409).

References

Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle, et al. 2007. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19:153.

Yoshua Bengio. 2012. Deep learning of representations for unsupervised and transfer learning. *Journal of Machine Learning Research–Workshop and Conference Proceedings*, 27:17–37.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, pages 92–100. ACM.

Roberto Calandra, Tapani Raiko, Marc Peter Deisenroth, and Federico Montesino Pouzols. 2012. Learning Deep Belief Networks from Non-stationary Streams. In *Artificial Neural Networks and Machine Learning - ICANN 2012*, number 7553 in Lecture Notes in Computer Science, pages 379–386. Springer Berlin Heidelberg.

Cornelia Caragea, Jian Wu, Kyle Williams, Sujatha Das, Madian Khabsa, Pradeep Teregowda, and C. Lee Giles. 2014. Automatic identification of research articles from crawled documents. In *Proceedings of the Workshop: Web-Scale Classification: Classifying Big Data from the Web*, New York, NY.

Noam Chomsky. 1980. Rules and representations. *Behavioral and brain sciences*, 3(01):1–15.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. 20(3):273–297.

Koby Crammer and Yoram Singer. 2002. On the algorithmic implementation of multiclass kernel-based vector machines. *The Journal of Machine Learning Research*, 2:265–292.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011a. Deep sparse rectifier networks. In *Proc. 14th International Conference on Artificial Intelligence and Statistics*, volume 15, pages 315–323.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011b. Domain Adaptation for Large-scale Sentiment Classification: A Deep Learning Approach. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 513–520.

Sujatha Das Gollapalli, Cornelia Caragea, Prasenjit Mitra, and C. Lee Giles. 2013. Researcher Homepage Classification Using Unlabeled Data. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 471–482. International World Wide Web Conferences Steering Committee.

Johan Hastad. 1987. *Computational limitations of small-depth circuits*. MIT press.

Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. 2006. A Fast Learning Algorithm for Deep Belief Nets. *Neural computation*, 18(7):1527–1554.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Geoffrey E. Hinton. 2002. Training Products of Experts by Minimizing Contrastive Divergence. *Neural Computation*, 14(8):1771–1800.

Hugo Larochelle and Yoshua Bengio. 2008. Classification using Discriminative Restricted Boltzmann Machines. In *Proceedings of the 25th International Conference on Machine Learning*, pages 536–543, Helsinki, Finland.

Hugo Larochelle, Michael Mandel, Razvan Pascanu, and Yoshua Bengio. 2012. Learning Algorithms for the Classification Restricted Boltzmann Machine. *The Journal of Machine Learning Research*, 13:643–669.

Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. 2014. Deeply-Supervised Nets. *arXiv:1409.5185 [cs, stat]*.

Dong-Hyun Lee. 2013. Pseudo-label: The Simple and Efficient Semi-supervised Learning Method for Deep Neural Networks. In *Workshop on Challenges in Representation Learning, ICML*, Atlanta, GA.

Tao Liu. 2010. A Novel Text Classification Approach Based on Deep Belief Network. In *Proceedings of the 17th International Conference on Neural Information Processing: Theory and Algorithms - Volume Part I*, ICONIP'10, pages 314–321. Springer-Verlag.

Zhengdong Lu and Hang Li. 2013. A Deep Architecture for Matching Short Texts. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 1367–1375. Curran Associates, Inc.

Shixiang Lu, Zhenbiao Chen, and Bo Xu. 2014. Learning new semi-supervised deep auto-encoder features for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 122–132, Baltimore, MD.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., Lake Tahoe, NV.

Kamal Nigam, John Lafferty, and Andrew McCallum. 1999. Using maximum entropy for text classification. In *IJCAI-99 workshop on Machine Learning for Information Filtering*, volume 1, pages 61–67.

Alexander G. Ororbia II, David Reitter, Jian Wu, and C. Lee Giles. 2015. Online learning of deep hybrid architectures for semi-supervised categorization. In *ECML PKDD*, Porto, Portugal. Springer.

Geoffrey K Pullum and Barbara C Scholz. 2002. Empirical assessment of stimulus poverty arguments. *The linguistic review*, 18(1-2):9–50.

Ruslan Salakhutdinov and Geoffrey Hinton. 2009. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978, July.

R. Sarikaya, G.E. Hinton, and A. Deoras. 2014. Application of Deep Belief Networks for Natural Language Understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(4):778–784.

Robert E. Schapire. 1990. The Strength of Weak Learnability. *Machine learning*, 5(2):197–227.

Tanya Schmah, Geoffrey E. Hinton, Steven L. Small, Stephen Strother, and Richard S. Zemel. 2008. Generative versus Discriminative Training of RBMs for classification of fMRI images. In *Advances in neural information processing systems*, pages 1409–1416.

Jürgen Schmidhuber. 2015. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117.

Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. 2011. Pegasos: Primal Estimated Sub-gradient Solver for SVM. *Mathematical programming*, 127(1):3–30.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 151–161. Association for Computational Linguistics.

Amarnag Subramanya and Jeff Bilmes. 2008. Soft-supervised learning for text classification. In *Empirical Methods in Natural Language Processing*, pages 1090–1099.

Michael Tomasello. 2001. Perceiving intentions and learning words in the second year of life. In Melissa Bowerman and Stephen Levinson, editors, *Language acquisition and conceptual development*, pages 132–158.

Jakub M. Tomczak. 2013. Prediction of Breast Cancer Recurrence using Classification Restricted Boltzmann Machine with Dropping. *arXiv preprint arXiv:1308.6324*.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. 2010. Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. *The Journal of Machine Learning Research*, 11:3371–3408.

Junbo Zhang, Guangjian Tian, Yadong Mu, and Wei Fan. 2014. Supervised Deep Learning with Auxiliary Networks. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 353–361, New York City, New York. ACM.

Guanyu Zhou, Kihyuk Sohn, and Honglak Lee. 2012. Online Incremental Feature Learning with Denoising Autoencoders. In *International Conference on Artificial Intelligence and Statistics*, pages 1453–1461.