Discriminative Alignment of Projected Belief Networks

Paul M. Baggenstoss Fraunhofer FKIE, Fraunhoferstrasse 20
53343 Wachtberg, Germany
Email: p.m.baggenstoss@ieee.org

***Abstract*—The projected belief network (PBN) is a deep layered generative network with tractable likelihood function (LF) and can be used as a Bayesian classifier by training a separate model on each data class, and classifying based on maximum likelihood (ML). Unlike other generative models with tractable LF, the PBN can share an embodiment with a feed-forward classifier network. By training a PBN with a cost function that combines LF with classifier cross-entropy, its network weights can be "aligned" to the decision boundaries separating the data class from other classes. This results in a Bayesian classifier that rivals state of the art discriminative classifiers. These claims are backed up by classification experiments involving spectrograms of spoken keywords and handwritten characters.**

I. Introduction

A. Motivation

A Bayesian classifier works by comparing the likelihood function (LF) of generative models (probability density estimators) trained individually on each data class. Generative classifiers are desirable because they posses a model of the data generation process, can generate synthetic data, and are robustness against outliers and adversarial attacks[1], but they generalize poorly compared to discriminative classifiers when sufficient labeled training data is available [2]. Reasons include model mismatch, differences in initialization and training of the various class models, and the difficulty of estimating high-dimensional probability distributions [3]. We therefore seek a method that exhibits the best qualities of both generative and discriminative classifiers.

B. Background and Proposed Approach

There have been many methods that seek to combine generative and discriminative classifiers [4], [5], [6], [7], [8], [9], [2], or to combine discriminative and generative training [2], [10], [11]. But, to truly combine both paradigms, one should seek a single model that is simultaneously generative and discriminative. Stacked restricted Boltzmann machines (RBMs) and auto-encoders, can be seen both as generative networks and as feed-forward networks, and therefore can co-exist with discriminative networks. In fact, these have been used as a pre-training approach for deep discriminative classifiers [12], [13], [14]. However, the LF of these generative networks intractable because the visible data is jointly distributed with the latent variables, which must be integrated out [15]. In a PBN, the latent variables are deterministically determined from the visible data, the integration to find the marginal is carried out on a manifold, resulting in a closed-form LF [16], [17]. Although the PBN LF is evaluated using the saddle point approximation (SPA), the error is for all practical purposes negligible, as we demonstrate in Section II-F. Having a tractable LF, the PBN can be trained and used as a true Bayesian classifier. And, being based on a feed forward network, the possibility exists to simultaneously train the network as a classifier.

This work was supported by Office of Naval Research Global under Research Grant - N62909-21-1-2024

Previous work on PBNs includes:

- Theoretical introductions [18], [19].
- Applications of PBN as a straight generative Bayesian classifier [18], [20], where a separate PBN is trained on each data class, either as a likelihood function, or as a type of auto-encoder based on reconstruction error.
- As as a regularizer for a discriminative classifier [21]. In this this approach, a single network is trained with a combined cost function: standard discriminative cross-entropy cost, and generative log-likelihood cost. The generative cost-function component acts as a regularization.

In the last item above, a discriminative classifier was trained by adding a generative component to the cost function. In contrast, this paper proposes to take a Bayesian classifier based on the PBN and add a discriminative component to the cost function. While training each class-dependent PBN model, a discriminative cost function is used to make the model more "sensitive" to other classes, effectively "aligning" the high-dimensional likelihood function to the decision boundaries. The result is a significantly improved generative classifier.

C. Paper Organization

In Section II, the mathematical background is presented including a review of PDF projection (Section II-A), the concept of discriminative alignment (Section II-B), a discussion of maximum entropy (MaxEnt) priors (Section II-C), the extension of PDF projection to multi-stage transformations (Section II-D), and its application to neural networks, called projected belief network (PBN) (Section II-E). In Section II-F, we discuss the saddle point approximation, which is used to calculate the likelihood function (LF) of the PBN. In Section III, the concepts are applied to a real data scenario consisting of spoken word commands. Finally, in the supplemental results (Section VII), results using a second data set consisting of handwritten characters are presented. Comments on computational loading are given in Section III-F and conclusions are presented in Section VI.

II. Mathematical Results

A. Review of PDF Projection and PBN

The PBN is an application of probability density function (PDF) projection [22], [16], [17], [23]. Consider an arbitrary differentiable dimension-reducing transformation $\mathbf{z} = T(\mathbf{x})$, mapping a N-dimensional input vector $\mathbf{x}$ to the M-dimensional output vector $\mathbf{z}$, where $M < N$. Let $\mathbb{X} \in \mathbb{R}^N$ be the input range, and $\mathbb{Z} \in \mathbb{R}^M$ be the output range. Let $g(\mathbf{z})$ be any distribution with support on $\mathbb{Z}$. Consider the function

$$G(\mathbf{x}) = \frac{p_0(\mathbf{x})}{p_0(\mathbf{z})} g(\mathbf{z}), \tag{1}$$

where $p_0(\mathbf{x})$ is some reference distribution (reference hypothesis) with support on $\mathbb{X}$, and $p_0(\mathbf{z})$ is the mapping of $p_0(\mathbf{x})$ with support on $\mathbb{Z}$. The PDF projection theorem [22] states that $G(\mathbf{x})$ is a distribution on $\mathbb{X}$ (it integrates to 1), and maps through $T(\mathbf{x})$ to $g(\mathbf{z})$. The distribution $G(\mathbf{x})$ is said to be the "projection" of $g(\mathbf{z})$ back to the input space. In fact, any distribution on $\mathbb{X}$ that maps to $g(\mathbf{z})$ is of the form (1) [16]. A unique projected PDF $G(\mathbf{x})$ arises if we restrict ourselves to reference distributions of maximum entropy [16]. In this paper, we do not consider discrete distributions because the calculation of $p_0(\mathbf{z})$ is intractable.

B. Discriminative Alignment

In Figure 1, an example projected distribution is shown. In this example, we used a linear transformation $\mathbf{z} = \mathbf{W}'\mathbf{x}$ and $p_0(\mathbf{x})$ was the uniform reference hypothesis for data values in the range $x_i \in [0,\ 1], \forall i$. The figure shows an intensity image and contour plot of the probability distribution $G(\mathbf{x})$. Since

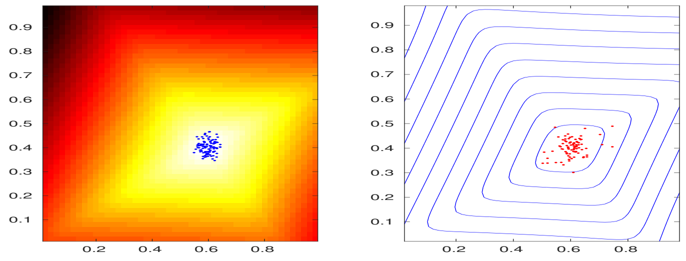

Fig. 1. Visualization of the implied (projected) PDF for a perceptron layer, reprinted from [18].

$G(\mathbf{x})$ is based on matrix $\mathbf{W}$, the contour lines are "aligned" with the linear column space of matrix $\mathbf{W}$. However, there is much freedom in the choice of these column spaces - there may be many alternative representations to get an equally good $G(\mathbf{x})$. In a discriminative classifier, however, these column spaces must be aligned with classification decision boundaries. Therefore, by training $\mathbf{W}$ for both purposes, to estimate $G(\mathbf{x})$ and to discriminate, it may be possible to achieve a $G(\mathbf{x})$ that is aligned to the decision boundaries, yet at the same time achieves a good representation of $G(\mathbf{x})$.

$\mathbb{X}$	a, b	MaxEnt Prior $p_e(x; a, b)$	$\lambda(a)$
$\mathbb{R}^N$	$0, -\frac{1}{2}$	$\phi(x, 0, \sigma^2)$ (Gaussian)	$\sigma^2 a$ (Linear)
$\mathbb{P}^N$	$0, -\frac{1}{2}$	$2\phi(x, 0, \sigma^2)$ (TG)	$\sigma^2 a + \sigma \frac{\phi(\sigma a;0,1)}{\Phi(\sigma a)}$ (TG)
$\mathbb{U}^N$	0,0	$\left(\frac{\alpha}{e^\alpha - 1}\right) e^{\alpha h}$ (TED)	$\frac{e^a}{e^a - 1} - \frac{1}{a}$ (TED)

TABLE I
MaxEnt priors and activation functions by input data range. TG="Trunc. Gauss.". TED="Trunc. Expon. Distr". Notation: $\phi(x; a, \sigma^2) = (2\pi\sigma^2)^{-1/2}\, e^{-(x-a)^2/(2\sigma^2)}$, $\Phi\,(x) \triangleq \int_{-\infty}^{x} \phi\,(x, 0, 1)$. The TED distribution simplifies to the uniform distribution for $a = 0, b = 0$.

C. MaxEnt Priors

Implementation of (1) requires a reference distribution $p_0(\mathbf{x})$, which can be seen as a Bayesian prior [16]. Having no information about the distribution of $\mathbf{x}$ aside from the data range, and perhaps variance (for normalized data), one appeals to the principle of maximum entropy (MaxEnt) to choose prior [24], [23], [25]. These MaxEnt distributions tend to be canonical distributions, which also simplifies the calculation of $p_0(\mathbf{z})$. In this paper, we consider three continuous-valued data ranges that cover most useful applications:

1) The *unconstrained* case,
$$\mathbb{X} = \mathbb{R}^N = \{\mathbf{x} : x_i \in [-\infty, \infty]\ \forall i\},$$
2) The *positive quadrant*,
$$\mathbb{X} = \mathbb{P}^N = \{\mathbf{x} : x_i \in [0, \infty]\ \forall i\},$$
3) The *unit hyper-cube*,
$$\mathbb{X} = \mathbb{U}^N = \{\mathbf{x} : x_i \in [0, 1]\ \forall i\}.$$

MaxEnt distributions are generally of the exponential class [26]. For the above data ranges, we need the class of distributions given by

$$p(\mathbf{x}|H_0) = p_e(\mathbf{x}; \mathbf{a}, b) = \prod_{i=1}^{N} p_e(x_i; a_i, b), \tag{2}$$

where
$$p_e(x; a, b) = \frac{1}{K(a, b)} e^{ax + bx^2}. \tag{3}$$

Since the entropy of a distribution increases with variance, entropy can grow without bounds in the ranges $\mathbb{R}^N$ and $\mathbb{P}^N$ unless some other constraints are imposed. We can impose necessary constraints through the parameters a, b in (3). A table of MaxEnt distributions and the selected values of a, b are given in Table I. In addition to the MaxEnt prior, the table shows the associated "activation function" $\lambda(a)$, which computes the expected value as a function of the parameter a.

D. Multi-Layer Networks

Implementing equation (1) requires a tractable expression for the output distribution $p_0(\mathbf{z})$, but this becomes intractable for complex transformations such as multi-layer networks. In these cases, equation (1) can be applied recursively one layer at a time and by assuming a separate MaxEnt prior for the input of each stage. To start, we assume that $\mathbf{z} = T(\mathbf{x})$ is just the first layer transformation. Then, $g(\mathbf{z})$ is itself written as a projected PDF based on the transformation of $\mathbf{z}$ occurring in the second layer. This recursion, called the chain-rule, repeats for an arbitrary numbers of layers ([22], page 676, Section

II.D). Consider a 3-layer network. Let $\mathbf{x}_{k-1}$ be the input of layer k, $\mathbf{x}_k = T_k(\mathbf{x}_{k-1})$ the output of layer k, and $\mathbf{x} = \mathbf{x}_0$ is the network input. For a 3-layer network, and the network output distribution is assumed to be $g(\mathbf{x}_3)$, the chain rule would be written

$$G(\mathbf{x}, g, T_1, T_2, T_3) = \frac{p(\mathbf{x}; H_0^0)}{p(\mathbf{x}_1; H_0^0)} \frac{p(\mathbf{x}_1; H_0^1)}{p(\mathbf{x}_2; H_0^1)} \frac{p(\mathbf{x}_2; H_0^2)}{p(\mathbf{x}_3; H_0^2)} g(\mathbf{x}_3), \tag{4}$$

where H_0^{k-1} is the MaxEnt prior distribution for layer k input.

E. Projected Belief Network

Applying (4) to a multi-layer perceptron network results in a projected belief network (PBN) [18], [19], [20], [21]. The LF of the PBN, given by (4), can be trained by gradient ascent. Each layer of a PBN consists of a dimension-reducing linear transformation (dense or convolutional), followed by a bias and non-linear activation function. The contribution of layer k to the overall LF (4) is given by

$$\frac{p(\mathbf{x}_k; H_0^k)}{p(\mathbf{x}_{k+1}; H_0^k)} = \frac{p(\mathbf{x}_k; H_0^k)}{p(\mathbf{z}_k; H_0^k)} |\mathbf{J}_k|, \tag{5}$$

where $\mathbf{z}_k$ is the output of the linear transformation of layer k, so $\mathbf{z}_k = \mathbf{W}_k' \mathbf{x}_k$, and $|\mathbf{J}_k|$ is the absolute value of the determinant of the Jacobian matrix for the invertable activation function, $|\mathbf{J}_k| = \prod_{i=1}^{M} \left| \frac{\partial x_{k+1,i}}{\partial z_{k,i}} \right|$, where M is the dimension of $\mathbf{z}_k$. The term $p(\mathbf{x}_k; H_0^k)$ is the MaxEnt prior distribution, which is specified.

F. Saddle Point Method: Approximate or Exact?

We have claimed that the PBN has a tractable LF, but unless $p(\mathbf{x}_k; H_0^k)$ is the Gaussian distribution, $p(\mathbf{z}_k; H_0^k)$ is not available in closed-form. However, note that the moment generating function (MGF) of $p(\mathbf{z}_k; H_0^k)$ is known exactly. Then, it is a matter of inverting the MGF, which is accomplished using the saddle point approximation (SPA) [27]. An arbitrary number of terms can be used to invert the MGF to achieve desired accuracy [28], [29], however this is unnecessary. The SPA approximates the shape of the MGF integrand at the saddle point with a Gaussian shape. Due to the central limit theorem, and helped by the benign canonical prior distribution, the Gaussian approximation becomes rapidly accurate as the dimension of $\mathbf{x}_k$ increases. The accuracy of the SPA holds even in the tails of the distribution.

To back up this claim, one can find a matrix $\mathbf{W}$ for which $p(\mathbf{z}_k; H_0^k)$ is tractable and can be compared with the SPA. For example, an equal-weighted sum of exponential random variables has a chi-squared distribution. In this case, the SPA was shown to have negligible error (See Figure 2 of [27]). Also, an equal-weighted sum of uniform random variables has an exact Irwin-Hall distribution which is compared in Figure 2 with the SPA (details found in the appendix of [30]). When $|z - 14| > 6.5$, numerical errors in the calculation of the theoretical distribution dominate, whereas the SPA remains accurate. Worst-case errors (inside the range where Irwin-Hall could be computed) were 6e-3, also negligible, and for a small N! We also tested the SPA for the truncated Gaussian prior.

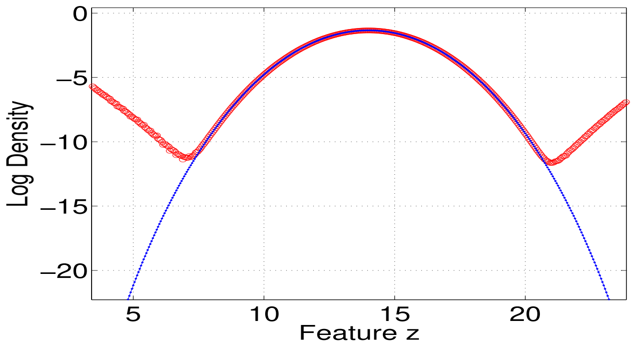

Fig. 2. Comparison of $\log p(\mathbf{z})$ for SPA (solid line) and Irwin-Hall (circles) for $N = 28$.

Because no case could be found with tractable distribution, we compared the SPA with a 2-dimensional histogram. Figure 3 shows the results for $N = 10$, $M = 2$ and 100,000 samples.

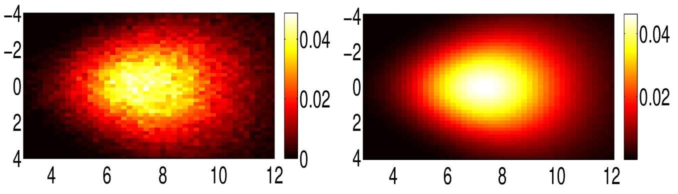

Fig. 3. Comparison of histogram (left) with SPA (right) for TG prior.

Not only is the error of the SPA implementation of the PBN negligible, but the SPA lends itself to exact gradient analysis for implementation of back-propagation and gradient ascent. The PBN-Toolkit [31] implements exact gradient analysis, and stochastic gradient training and evaluation of PBNs.

III. Classification of Words Commands

A. Data set

To obtain a data set that is at the same time relevant, realistic, and challenging, we chose a subset of the Google speech commands data [32], selecting three pairs of difficult to distinguish words: “three, tree”, “no, go”, and “bird, bed”. Classification experiments were conducted for each word pair. The data was sampled at 16 kHz and segmented into into 48 ms Hanning-weighted windows shifted by 16 ms. We used log-MEL band energy features with 20 MEL-spaced frequency bands and 45 time steps, for a frequency span of 8 kHz and a time span of 0.72 seconds. The input dimension was $N = 45 \times 20 = 900$. From each of the six classes, we selected 500 training samples, 150 validation samples, at random. The remaining samples were used to test, averaging about 1500 per class or about a total of 10000 testing samples.

B. Network

The networks had $L = 8$ layers. The first layer was convolutional (CONV) with 6 13×15 CONV kernels using CONV padding to create output feature maps equal in size to the input. Then, 3×4 down-sampling was used, resulting in 15×5 output maps, for a total dimension of 450. The second layer was CONV with 40 7×5 CONV kernels using no CONV padding. Then, 2×1 down-sampling was used, resulting in 5×1 output maps, for a total dimension of 200. The remaining layers were dense, with 64, 48, 32, 16, 8, and 2 hidden units. The activation function at the output of the first layer was linear, but all other layers except the last layer used the truncated Gaussian activation [19] which is similar to softplus (See Table I). The final output activation function was TED (See Table I). The network was implemented using PBN Toolkit [31].

C. Experimental Approach

The goal was to measure the improvement afforded by discriminative alignment (DA) of a strict Bayesian classifier constructed from PBNs, and then compare the results with a state of the art deep neural network (DNN). Let k, l represent the two words of a word pair and $p_k(\mathbf{x})$ the likelihood function, as computed by the PBN trained on word k. Then, the cost function to minimize when training $p_k(\mathbf{x})$ is given by $C_k = -\sum_{i=1}^{N_k} \log p_k(\mathbf{x}_i) + \gamma \sum_{j=1}^{N_k+N_l} E_{k,l}(\mathbf{x}_j)$, where $E_{k,l}(\mathbf{x}_j)$ is the cross-entropy cost function for classifying between classes k and l, γ is the factor that determines the amount of discriminative influence (typically about 100), i indexes over only data for word k, but j indexes over data for both words. Note that the network parameters affect both $p_k(\mathbf{x})$ and $E_{k,l}(\mathbf{x})$. With $\gamma = 0$, a straight PBN results, and for $\gamma > 0$, a PBN-DA results. Note that the generative part of the cost function depends only on a given data class, but the discriminative component depends on data from both words in the pair. Classification is accomplished by just comparing likelihood functions, i.e. $\arg\max_k p_k(\mathbf{x})$.

D. Classification Results

Table II lists the classification results for PBN and PBN-DA, as well as for a state of the art discriminative DNN, which was trained using the same network size and structure, but with max-pooling in place of down-sampling, and dropout regularization. As can be seen the PBN-DA results are significantly better than PBN, and much closer to the performance of the DNN.

	"three-tree"	"no-go"	"bird-bed"
PBN	15.75%	14.5%	5.75%
PBN-DA	11.7%	12.2%	4.8%
DNN	8.4%	11.5%	4.3%
DNN + PBN-DA	7.3%	9.3%	3.8%

TABLE II
COMPARISON OF DNN, PBN, AND PBN-DA ERROR PROBABILITY FOR THE THREE WORD PAIRS.

E. Classifier Combination

Ideally, when combining classifiers, the individual classifiers should be based on independent views of the data, and must have comparable performance. This is why a generative classifier with good performance is highly desirable - because it should combine well with a discriminative classifier. Figure 4 shows the results of combining DNN and PBN-DA individually for each word pair using the combined score $c_k(\mathbf{x}) + \alpha \log p_k(\mathbf{x})$, where $c_k(\mathbf{x})$ is the classifier score reported by the discriminative network and α is the combining factor. The combined classification error is plotted as a function of the combining factor. On the far right of each plot can be seen the PBN-DA performance, while on the far left is the DNN performance. In each case, a significant drop in error is seen, and at about the same value of combining factor.

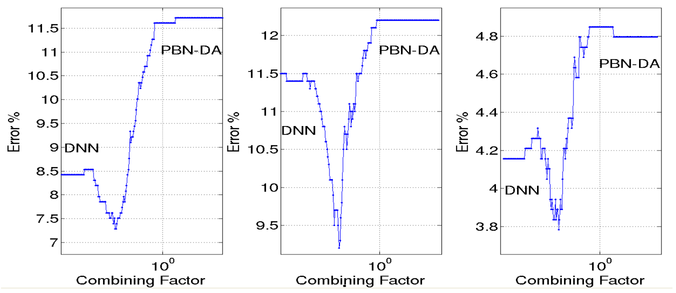

Fig. 4. Classifier error as a function of combining factor by word pair.

F. Implementation and Computational Loading

The PBN was implemented using PBN Toolkit [31] with graphical processing unit (GPU) acceleration. However, like any true Bayesian classifier, a model must be separately trained on each data class, increasing the computational load and adding to the already high requirements for PBN, which is an order of magnitude higher then a feed-forward network. For this reason, PBN-DA is best suited to problems with a low or moderate number of classes.

IV. Summary of Supplementary Results

Section VII provides additional results using a second data set composed of handwritten characters. As for the keyword classification experiments, the PBN-DA showed significant improvement over straight PBN. In fact, the PBN-DA performance was very close to the DNN performance.

V. Data and Software Availability

Software in Python as well as data and network parameters for the data set described in the supplemental material are made available online at http://class-specific.com/pbntk.

VI. Conclusions and Future Work

The PBN is a layered generative network with tractable LF and shares an embodiment with a multi-layer perceptron. This allows a Bayesian classifier constructed from PBNs to benefit from using a combined discriminative/generative cost function that works to "align" the weights with the decision boundaries. In this paper, we demonstrated the effectiveness of discriminative alignment of PBNs in classification experiments for spoken keywords as well as for handwritten characters. We also justified the claim that a PBN has a tractable likelihood function, despite using the saddle point approximation. Future work includes the application of PBN to larger data sets and the exploration of the relationship of PBN to the myriad generative methods currently being proposed, and how to best take advantage of the unique properties of PBN.

References

[1] C. Mayer and R. Timofte, “Adversarial sampling for active learning,” *arXiv:1808.06671, to appear WACV 2020*, 2019.

[2] J. Lasserre, C. Bishop, and T. Minka, “Principled hybrids of generative and discriminative models,” vol. 1, pp. 87– 94, 07 2006.

[3] V. Vapnik, *The Nature of Statistical Learning*. Springer, 1999.

[4] T. Jaakkola and D. Haussler, “Exploiting generative models in discriminative classifiers,” tech. rep., Dept. of Computer Science, Univ. of California, 1998.

[5] R. Raina, Y. Shen, A. Ng, and A. McCallum, “Classification with hybrid generative /discriminative models,” in *Proceedings of NIPS (Neural Information Processing Systems) 2004*, 2004.

[6] S. Fine, J. Navratil, and R. Gopinath, “Enhancing gmm scores using SVM hints,” in *Proceedings of the 7th European Conference on Speech Communication and Technology (EuroSpeech)*, 2001.

[7] A. Fujino, N. Ueda, and K. Saito, “A hybrid generative/discriminative approach to semi-supervised classifier design,” in *Proceedings of the National Conference on Artificial Intelligence*, vol. 20, 2005.

[8] A. Holub, M. Welling, and P. Perona, “Hybrid generative-discriminative visual categorization,” in *Proceedings of the International Journal of Computer Vision*, vol. 77, pp. 239–258, 2008.

[9] A. Bosch, A. Zisserman, and X. Muoz, “Scene classification using a hybrid generative/discriminative approach,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 4, pp. 712–727, 2008.

[10] T. Minka, “Discriminative models, not discriminative training,” *Microsoft Research Ltd, technical report*, 2005.

[11] C. Bishop and J. Lasserre, “Generative or discriminative? getting the best of both worlds,” *Bayesian Statistics*, vol. 8, pp. 3–24, 2007.

[12] M. Ranzato, Y. Boreau, and Y. LeCun, “Sparse feature learning for deep belief networks,” in *Proceedings of NIPS 2007*, 2007.

[13] Y. Bengio, P. Lamblin, D. Popovici, , and H. Larochelle, “Greedy layer-wise training of deep networks,” in *Proceedings of NIPS 2006*, 2006.

[14] G. E. Hinton, S. Osindero, and Y.-W. Teh, “A fast learning algorithm for deep belief nets,” in *Neural Computation 2006*, 2006.

[15] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep Learning*. Cambridge, MA: MIT press, 2016.

[16] P. M. Baggenstoss, “Maximum entropy PDF design using feature density constraints: Applications in signal processing,” *IEEE Trans. Signal Processing*, vol. 63, June 2015.

[17] P. M. Baggenstoss, “Uniform manifold sampling (UMS): Sampling the maximum entropy pdf,” *IEEE Transactions on Signal Processing*, vol. 65, pp. 2455–2470, May 2017.

[18] P. M. Baggenstoss, “On the duality between belief networks and feed-forward neural networks,” *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–11, 2018.

[19] P. M. Baggenstoss, “A neural network based on first principles,” in *ICASSP 2020, Barcelona (virtual)*, (Barcelona, Spain), Sep 2020.

[20] P. M. Baggenstoss, “Applications of projected belief networks (PBN),” *Proceedings of EUSIPCO, A Corunã, Spain*, 2019.

[21] P. M. Baggenstoss, “The projected belief network classifier: both generative and discriminative,” *Proceedings of EUSIPCO, Amsterdam*, 2020.

[22] P. M. Baggenstoss, “The PDF projection theorem and the class-specific method,” *IEEE Trans Signal Processing*, pp. 672–685, March 2003.

[23] P. M. Baggenstoss, “Beyond moments: Extending the maximum entropy principle to feature distribution constraints,” *Entropy*, vol. 20, no. 9, 2018.

[24] E. T. Jaynes, “On the rationale of maximum-entropy methods,” *Proceedings of IEEE*, vol. 70, no. 9, pp. 939–952, 1982.

[25] F. Palmieri and D. Ciuonzo, “Objective priors from maximum entropy in data classification,” *Information Fusion*, vol. 14, pp. 186–198, 04 2013.

[26] J. N. Kapur, *Maximum Entropy Models in Science and Engineering*. Wiley (Eastern), 1993.

[27] S. M. Kay, A. H. Nuttall, and P. M. Baggenstoss, “Multidimensional probability density function approximations for detection, classification, and model order selection,” *IEEE Transactions on Signal Processing*, vol. 49, pp. 2240–2252, Oct 2001.

[28] O. Barndorff-Nielsen and D. R. Cox, “Edgeworth and saddle-point approximations with statistical applications,” *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 41, no. 3, pp. 279–299, 1979.

[29] A. H. Nuttall, “Saddlepoint approximation and first-order correction term to the joint probability density function of M quadratic and linear forms in K Gaussian random variables with arbitrary means and covariances,” *NUWC Technical Report 11262*, December 2000.

[30] P. M. Baggenstoss, “Evaluating the RBM without integration using pdf projection,” in *Proceedings of EUSIPCO 2017, Island of Kos, Greece*, Aug 2017.

[31] P. Baggenstoss, “PBN Toolkit,” *http://class-specific.com/pbntk*.

[32] P. Warden, “Speech commands: A dataset for limited-vocabulary speech recognition,” *arXiv:1804.03209*, 2018.

[33] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, Nov 1998.

VII. Supplemental Results

To strengthen the claims made in this paper, we made similar experiments using a separate data corpus. The experiment follows the approach of Section III.

A. Data Description

As a supplemental data set, we chose a subset of the MNIST data corpus [33]. The three characters "3", "8", and "9" were chosen and down-sampled 2:1 to a data size of 14×14, with dimension $N = 196$. We used 500 samples of each class for training and the rest for testing. Because MNIST pixel data is coarsely quantized in the range [0,1], it is poorly suited for generative modeling with a continuous-valued distribution. To make the data suitable for generative modeling, the data was first dithered [1], then transformed to $\mathbb{R}^N$ by applying the inverse sigmoid function, resulting in a smooth Gaussian-like distribution in the range -10 and 10.

B. Network

The PBN networks had 6 layers. The first layer was convolutional (CONV) with 3 9×9 CONV kernels using CONV padding to create output feature maps equal in size to the input. Then, 2×2 down-sampling was used, resulting in 7×7 output maps, for a total dimension of 147. The second layer was CONV with 20 5×5 CONV kernels using no CONV padding. Then, 2×2 down-sampling was used, resulting in 2×2 output maps, for a total dimension of 80. The remaining layers were dense, with 32, 16, 8, and 3 hidden units. The activation function at the output of the first layer was linear, but all other layers except the last layer used the truncated Gaussian activation [19] which is similar to softplus (See Table I). The final output activation function was TED (See Table I). The network was implemented using PBN Toolkit [31].

C. Classification Results

Table III lists the classification results for PBN and PBN-DA, as well as for a state of the art discriminative DNN, which was trained using the same network size and structure, but with max-pooling in place of down-sampling, and dropout regularization. Similarly to the keyword data results presented in Section III, the PBN-DA results are significantly better than PBN and approach the DNN performance.

PBN	PBN-DA	DNN	DNN + PBN-DA
93.5 %	96.47%	97.08%	**97.43%**

TABLE III
Comparison of DNN, PBN, and PBN-DA error probability for the three word pairs.

Figure 5 shows the results for combining PBN-DA with the DNN, as described in Section III. Once again, the combined results are superior to either PBN-DA or DNN alone, indicating effective classifier combination, a result of combining classifiers with comparable performance and independent views of the data.

[1]For pixel values above 0.5, a small exponential-distributed random value was subtracted, but for pixel values below 0.5, a similar random value was added.

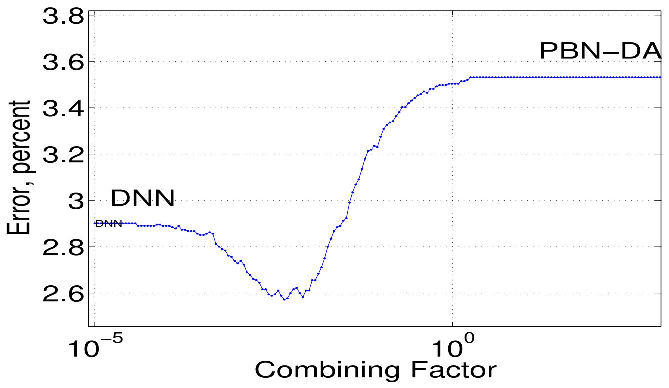

Fig. 5. Classifier error as a function of combining factor for the MNIST data set.

D. Data and Software Availability

To facilitate the duplication of these results by researchers, we have made all data and software (in Python scripts) available online at http://class-specific.com/pbntk.