Automatic Gating of Attributes in Deep Structure

Xiaoming Jin1, Tao He1, Cheng Wan2, Lan Yi3, Guiguang Ding1, Dou Shen2
1 School of Software, Tsinghua University, Beijing, China
2 Baidu Corporation, Beijing, China
3 Department of DevNet, Cisco Systems

xmjin@tsinghua.edu.cn, het15@mails.tsinghua.edu.cn, befirefly@gmail.com,
layi@cisco.com, dinggg@tsinghua.edu.cn, doushen@gmail.com

Abstract

Deep structure has been widely applied in a large variety of fields for its excellence of representing data. Attributes are a unique type of data descriptions that have been successfully utilized in numerous tasks to enhance performance. However, to introduce attributes into deep structure is complicated and challenging, because different layers in deep structure accommodate features of different abstraction levels, while different attributes may naturally represent the data in different abstraction levels. This demands adaptively and jointly modeling of attributes and deep structure by carefully examining their relationship. Different from existing works that treat attributes straightforwardly as the same level without considering their abstraction levels, we can make better use of attributes in deep structure by properly connecting them. In this paper, we move forward along this new direction by proposing a deep structure named Attribute Gated Deep Belief Network (AG-DBN) that includes a tunable attribute-layer gating mechanism and automatically learns the best way of connecting attributes to appropriate hidden layers. Experimental results on a manually-labeled subset of ImageNet, a-Yahoo and a-Pascal data set justify the superiority of AG-DBN against several baselines including CNN model and other AG-DBN variants. Specifically, it outperforms the CNN model, VGG19, by significantly reducing the classification error from 26.70% to 13.56% on a-Pascal.

1 Introduction

Deep learning [Bengio, 2009; Deng, 2014] is well known for its capability of better representing data when compared with “shallow structures” (e.g., support vector machine (SVM) etc.). It can extract features from raw data in different abstraction levels. Deep learning has been applied in a variety of learning tasks such as speech recognition [Mohamed et al., 2012; Deng et al., 2013], image recognition [He et al., 2016], and natural language processing [Collobert et al., 2011; Bahdanau et al., 2014] etc.

Attributes are manually defined qualities, properties or characteristics to describe an object [Farhadi et al., 2009]. They have been successfully used in fields such as computer vision as additional information to help many tasks [Wang and Ji, 2016; Wei and Pal, 2011; Lampert et al., 2009; Wang and Mori, 2010; Hwang et al., 2011]. Unlike raw data, attributes usually represent higher level abstractions of an object. Moreover, different attributes may describe an object in different abstraction levels. Specifically, the attributes (e.g., “engine” in Figure 1) that describe general or holistic properties of an object [Abdulnabi et al., 2015], represent higher level abstraction; while others (e.g., “round” in Figure 1) that describe specific object parts or localized properties, represent lower level abstraction.

However, things become complicated and nontrivial when integrating attributes into deep structures to help target learning tasks. The great advantage of deep structure is to learn transformation of the data with less redundancies and to make it easier to extract useful information [Bengio, 2009]. Intuitively, different layers of deep structures represent different abstraction levels of the raw data or features [LeCun et al., 2015]. The motivation of this paper is that, unlike raw features which are usually treated as flatten visible inputs of deep structures, attributes are artificial descriptions in nature and should be treated in different abstraction levels. In other words, when integrating into deep structures, attributes should be connected to different hidden layers regarding their corresponding abstraction levels.

The performance of deep structure may be greatly hurt if we connect attributes to incorrect hidden layers. Specifically, a low abstraction level attribute will cause incomplete abstraction if it is connected to higher hidden layers; while a high abstraction level attribute will cause over abstraction if it is connected to lower hidden layers. Figure 1 is an example from a-Yahoo dataset. Through simple experiments on a two-layer neural network, it is observed that the test error rises up from 73.75% to 82.09% if the higher abstraction level attribute “engine” is connected to the lower hidden layer. Meanwhile, the test error rises up from 66.77% to 78.45%
if the lower abstraction level attribute “round” is connected to the higher hidden layer. The same situation also occurs for high abstraction level attribute “wool” and low abstraction level attribute “2D Boxy”. These phenomena clearly demonstrate the fact that attributes are in different abstraction levels, and should be connected to correct hidden layers in deep structure.

In this paper, an innovative network named Attribute Gated Deep Belief Network (AG-DBN) is proposed to automatically and adaptively integrate attributes into deep structures at different hidden layers. In AG-DBN, attributes are no longer treated as the same as raw features. Instead, each attribute unit is directly connected to their appropriate hidden layers in deep structure. To help determine appropriate connections for each attribute, a learnable gating mechanism is proposed in a probabilistic framework that allows training from potentially connections to all hidden layers, and then can be normalized to their appropriate ones.

The major contributions of this paper are summarized as follows: (1) It proposes a principled method to exploit different abstraction levels in attributes, and establish direct connections between attributes and corresponding appropriate hidden layers in deep structure through learnable gating mechanism. (2) It elaborates corresponding inference and learning algorithm. (3) It evaluates the proposed method on three real-world data sets and demonstrates its superiority over six baseline methods through experimental results. Specifically, the model proposed in this paper reduces error on a-ImageNet by 9.91% and on a-Yahoo by 12.88% compared with baselines, and reduces error on a-Pascal by 13.14% compared with CNN model, VGG19.

2 Related Work

Attribute related tasks Existing works utilize attributes as additional input data [Wang and Ji, 2016; Wei and Pal, 2011] to enhance object recognition, middle level representation [Lampert et al., 2009; Wang and Mori, 2010] for transfer learning, or additional output labels [Hwang et al., 2011] to build shared feature space. However, none of existing methods is able to learn and utilize the attributes abstraction level information. And in this paper, we mainly discuss the case where attributes are regarded as input to enhance object recognition or image classification task, rather than middle level representation or output labels.

Deep structure Restricted Boltzmann Machine (RBM) [Hinton, 2010] is an energy-based model, which can be represented by an undirected bipartite graph. Several variants of RBM are proposed to enhance its modelling performance, such as sparsity constraint [Wan et al., 2015] and conditional parametrization [Mnih et al., 2012]. RBM can be stacked to build Deep Belief Net (DBN) [Hinton, 2009] for providing hierarchical representation. DBN first initializes the network with unsupervised layer-wise pre-training, and then fine tunes the network parameters in supervised style. [Sohn et al., 2013] proposed a point-wise gated Boltzmann machine (PGBM) to learn useful high-level features when data contains irrelevant patterns. Basically, PGBM performs unsupervised RBM training and feature selection simultaneously. The function of the introduced gating mechanism in this paper is similar to that in PGBM, but the problem addressed, the network structure and the learning algorithm are all different.

Skip layer connection Skip-layer connection is a common pattern in deep structure [Bishop, 2006], where input data contribute as the lowest features and, meanwhile, as the highest abstraction features for output label layer. Although skip-layer connection and our method both suppose that it is not necessary to feed input data to deep structure from the lowest layer, skip-layer connection is a kind of human-specified connection, whereas our method adaptively connects attributes to different layers by learning from input data, rather than human knowledge. Two typical types of skip-layer connection will be compared with our method in the experiment section.

3 Attribute Gated Deep Belief Network

The level of hidden layers in deep structures naturally indicates the level of feature abstraction of the data. The higher level a layer is in, the more abstract representation it offers. Meanwhile, each attribute can be viewed as a description to the target object(s) in a certain abstraction level. The Attribute Gated Deep Belief Network (AG-DBN) is proposed by the motivation to automatically map the abstraction levels of attributes to the appropriate deep structure inner abstraction levels.

3.1 Model

Figure 2 is a demonstration of the proposed AG-DBN. It contains (1) a DBN constituted by raw feature layer $v$ and hidden layers $h_l$, and (2) a gating mechanism that connects attributes $a_j$ to the DBN. The connections between the attributes and the hidden layers are controlled by a gate matrix $G$, where $G_{ij}$ is a binary value indicating whether attribute unit $a_j$ connects to hidden layer $h_i$. For example, $G_{11} = 1, G_{21} = 0, G_{12} = 0$, and $G_{22} = 1$ denote that attribute unit $a_1$ connects to hidden layer $h_1$, and $a_2$ connects to $h_2$. $g_i$ or $G_i$ (the $l$-th row vector of $G$) denotes the connections between the $l$-th hidden layer and different attributes. Similarly, $g_j$ or $G_j$ (the $j$-th column vector of $G$) denotes the connections between the $j$-th attribute $a_j$ and different layers. By adjusting the values of gate matrix, the hidden layers that the attribute units should connect to are varied. For notations used in the rest of this paper, please refer to Table 1 for their descriptions.
Table 1: Notation table

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G$</td>
<td>gate matrix ($N_h \times N_a$)</td>
</tr>
<tr>
<td>$W_l$</td>
<td>weight matrix between $v_l$ and $h_l$</td>
</tr>
<tr>
<td>$g_{ij}$</td>
<td>$j$-th row in $G$ ($G_{ij}$)</td>
</tr>
<tr>
<td>$U_l$</td>
<td>weight matrix between $a$ and $h_l$</td>
</tr>
<tr>
<td>$d_i$</td>
<td>bias of hidden units in $i$-th layer</td>
</tr>
<tr>
<td>$b_i$</td>
<td>bias of visible units in $i$-th layer</td>
</tr>
<tr>
<td>$d_{ai}$</td>
<td>bias of attribute units</td>
</tr>
<tr>
<td>$v_l$</td>
<td>visible units in $l$-th layer</td>
</tr>
<tr>
<td>$n_l$</td>
<td>number of attribute</td>
</tr>
<tr>
<td>$G_l$</td>
<td>hidden units in $l$-th layer</td>
</tr>
<tr>
<td>$V$</td>
<td>visible units, input data $[v_1, v_2, \ldots, v_n]$</td>
</tr>
<tr>
<td>$A$</td>
<td>input attribute data $[a_1, a_2, \ldots, a_n]$</td>
</tr>
<tr>
<td>$N_a$</td>
<td>number of attribute</td>
</tr>
</tbody>
</table>

The overall learning algorithm is shown in Algorithm 1. It iteratively runs two learning phases: to learn DBN parameters (step 3 to step 5) by clipping gate matrix, and to learn gate matrix (step 6 to step 9) by clipping DBN. $Q(G)$ is the probabilistic distribution of $G$. Specifically, each column in $Q(G)$ is a probabilistic distribution, which represents the probability of an attribute connecting to a hidden layer. To compute $Q(G)$ in each iteration, the algorithm first accumulates $\sum D P(G^{(i)}|v_j, a, h_j)$ against training data $D$, and then normalizes $Q(G^{(i)})$ in column direction.

Algorithm 1 Overall Learning Algorithm of AG-DBN

**Input:**
- Training data $D = (V, A)$.
- Training iteration $N_0$.
- Number of layers $L$.
- Learning rate $\eta$. Updating epochs $N_e$. Batch size $s$.

**Output:**
- The values of gate matrix $G^{(N_0)}$.

1: Initialize $Q(G^{(0)})$
2: for $i = 1$ to $N_0$ do
3: for $j = 1$ to $L$ do
4: \hspace{0.5cm} updateRBM(D, $Q(G^{(i-1)})$, $\eta$, $N_e$, $s$); // learning RBM parameters with given gate matrix, see Algorithm 2
5: \hspace{1cm} end for
6: for $j = 1$ to $L$ do
7: \hspace{1.5cm} Compute $\sum D P(G^{(i)}|v_j, a, h_j)$
8: \hspace{1cm} end for
9: \hspace{0.5cm} Normalize $Q(G^{(i)})$
10: \hspace{0.5cm} end for

Algorithm 2 learns the $l$-th layer RBM parameters of AG-DBN given the gate matrix $G$ using CD [Hinton, 2002]. The network is randomly initialized before starting a new iteration, since we are using the new gate $Q(G^{(i-1)})$. From line 4, we can also find that we sample $G_l$ for each batch in each training iteration. So we learn the network with the expectation of $Q(G)$ as long as the number of batches is big enough.

At last, we obtain $G$ by sampling from distribution $Q(G^{(N_0)})$ obtained after the last iteration. With $G$, we run Algorithm 1 once again. But different from before, $G$ is no longer sampled in Algorithm 2, since it has been well learned. The alternative process in Algorithm 1 ensures that gates are updated to a better value, since RBM parameters are well-learned on fixed gates, and gates are updated on the basis of well-learned RBM parameters.

3.3 Analyzing Remarks

**Parameter Settings** In the learning algorithms of AG-DBN, the training iteration number $N_0$ and the batch size $s$ are critical parameters and have to be appropriately determined. In experiments, we found 15 iterations are enough to find the optimal values of gate matrix. The batch size $s$ actually controls...
Algorithm 2 Learning the parameters of $l$-th RBM with given gate matrix

**Input:**
- Training data $D$,
- Distribution of gate matrix $Q(G_l^{(i-1)})$,
- Training epochs $N_e$,
- Learning rate $\eta$,
- Batch size $s$.

**Output:**
- The parameters of AG-DBN, $W_1, U_1, b_i, c_i, d_i$.

\begin{align*}
P = (v, a, G_l^{(i-1)}) \text{ as the input } v_{i+1} \text{ of next layer.} \\
1: & \text{Randomly initialize } W_1, U_1, b_i, c_i, d_i \\
2: & \text{for } k = 1 \text{ to } N_e \text{ do} \\
3: & \text{for each training batch } B \text{ in training data } D \text{ do} \\
4: & \text{Sample } G_l^{(i-1)} \sim Q(G_l^{(i-1)}) \\
5: & \text{for each training instance } (v_i^{(0)}, a_i^{(0)}) \text{ in } B \text{ do} \\
6: & \text{Sample } h_i^{(0)}, v_i^{(1)}, a_i^{(1)}, b_i^{(2)} \\
7: & W_i \leftarrow W_i - \eta (h_i^{(0)}(v_i^{(0)})^T - P(h_i) = 1|v_i^{(1)}, a_i^{(1)}, G_l^{(i-1)}) (v_i^{(1)})^T) \\
8: & U_i \leftarrow U_i - \eta (h_i^{(0)}(a_i^{(0)} \odot G_l^{(i-1)})^T - P(h_i) = 1|v_i^{(1)}, a_i^{(1)}, G_l^{(i-1)}) (a_i^{(1)} \odot G_l^{(i-1)})^T) \\
9: & b_i \leftarrow b_i - \eta (v_i^{(0)} - v_i^{(1)}) \\
10: & c_i \leftarrow c_i - \eta (h_i^{(0)} - P(h_i) = 1|v_i^{(1)}, a_i^{(1)}, G_l^{(i-1)}) \\
11: & d_i \leftarrow d_i - \eta (a_i^{(0)} \odot G_l^{(i-1)} - a_i^{(1)} \odot G_l^{(i-1)}) \\
12: & \text{end for} \\
13: & \text{end for} \\
14: & \text{end for}
\end{align*}

the number of sampling $G$. With a large batch size, it easily incurs occasionality for lacking enough sampling. While with a small batch size, it results in long training time for the parallelism issue. With enough observation in experiments, the batch size $s$ is finally set to 50.

**Complexity** Compared to the learning algorithm of DBN, Algorithm 2 involves additional steps to learn $U_i$ and $d_i$. For convenience, assume that all hidden layers in AG-DBN contain the same number of hidden units $N_h$, the extra computing complexity to learn $U_i$ and $d_i$ is $O(N_h L N_a)$, where $N_a$ is the number of attribute units. The extra time complexity of learning $G$ is $O(N_h L N_a)$.

The learning algorithm of AG-DBN can also be viewed as searching for the optimal positions between attribute units and corresponding hidden layers. An AG-DBN with $L$ hidden layers and $N_a$ attribute units needs to check among $L^{N_a}$ candidates for the optimal ones. Therefore, in theory, the upper bound of time complexity of training AG-DBN is $O(L^{N_a})$.

**Extension** Although the inference and training algorithms introduced above are based on DBN, they can be easily extended and applied to CNN by the following two approaches. First, [Lee et al., 2009] proposed the convolutional DBN (CDBN), a hierarchical generative model which can be transformed to CNN. The basic building blocks of CDBN are convolutional RBM and probabilistic max-pooling, where each hidden layer can actually be represented by probabilistic inference. This helps the so called Attribute Gated CDBN(AG-CDBN) to infer the conditional probabilities of gate and attribute units in the same way as that in AG-DBN. Alternatively, we can extract features from pre-trained CNN models, e.g., AlexNet [Krizhevsky et al., 2012], VGG [Simonyan and Zisserman, 2014], and ResNet [He et al., 2016], etc., and directly feed the features and the corresponding attributes to our AG-DBN. The latter approach is not only restricted to features from CNN, but also can utilize features from other sophisticated models, which expands the application scope of AG-DBN. So this approach is evaluated in the experiment section, and the results show obvious improvements when compared with the state-of-the-art CNN model.

4 Experiments

The major goal of experiments is to validate the idea of connecting attributes to appropriate hidden layers in deep structures, therefore, only those attribute enhanced deep structures (see Figure 4) are involved to compare with proposed AG-DBN and two simple variants of AG-DBN in experiments. Meanwhile, two attribute-available data sets, i.e., a-ImageNet and a-Yahoo, are used in experiments.

**a-ImageNet** ImageNet [Russakovsky et al., 2015] is a widely used image data set. Among 1000 synsets, there are 25 available attributes for 400 popular synsets. In each synset, some images are annotated with the attributes of four groups. That is, (color) for black, blue, brown, gray, green, orange, pink, red, violet, white and yellow; (pattern) for spotted and striped; (shape) for long, round, rectangular and square; (texture) for furry, smooth, rough, shiny, metallic, vegetation, wooden and wet. The value of each attribute can be 0 for no and 1 for yes.

In the original 400 synsets, only a small part of the images are annotated with attributes. To make the experiments meaningful, 10 synsets with totally 8309 images are randomly selected in this experiments. Some images in the selected 10 synsets have been manually annotated. Due to the heavy human effort, the manually annotated data set is small compared with the original one. The ratios of images with attributes in each synset finally reach 14.30%, 14.94%, 1.98%, 4.33%, 10.34%, 2.67%, 2.38%, 2.82%, 99.64% and 78.77% respectively. We name the selected 10 synsets together with the manually annotated attributes a-ImageNet.

**a-Yahoo** a-Yahoo data set is collected by [Farhadi et al., 2009] from the Yahoo image search. There are twelve different attributes annotated with attributes.

\[1]\text{https://github.com/Tsinghua-IDE/a-ImageNet}
different objects targeted in this set, i.e., wolf, zebra, goat, donkey, monkey, statue of people, centaur, bag, building, jet ski, carriage, and mug. Each image contains an object and its corresponding values of 64 attributes, e.g., “has head”, “has leg”, “is furry”, “is shiny”, etc. For the full attribute list please refer to [Farhadi et al., 2009].

4.1 Experimental Settings

To evaluate the effectiveness of different models, object recognitions are done on both a-ImageNet and a-Yahoo. Models all consist of 2 hidden layers of size 500 and 300, and an input layer accepting 50*50 gray scale raw image pixels. All experimental results are averaged over 5 independent runs. We compare the AG-DBN with six baseline models: four baseline models in Figure 4: standard DBN, Transfer DBN (T-DBN), skip-layer with attribute (skip-a), and skip-layer with attribute and data (skip-ad), and two simple variants of our AG-DBN (Manual AG-DBN and Random AG-DBN). In order to do classification task, a softmax layer is added on the top of all DBN based models, including T-DBN, AG-DBN and its variants.

Standard DBN is trained only on raw features without considering any attribute information. T-DBN [Wei and Pal, 2011] is actually a special DBN whose visible units consist of both raw features and attributes. Skip-a and skip-ad are two similar models. Skip-a directly connects attribute units to both the first hidden layer and the output layer. Skip-ad enhanced skip-a by adding connections from visible pixel units to the output layer.

Manual AG-DBN is a special AG-DBN to verify the effectiveness of connecting attribute units to hidden layers using human prior knowledge. It presents one-hot values to G by observation, and keeps G constant in model training. Manual AG-DBN is only tested on a-ImageNet as its 25 attributes can be easily grouped into lower-semantic and higher-semantic as shown in Table 3. However, the 64 a-Yahoo attributes are not clearly stratified in semantic thus Manual AG-DBN is not tested. Random AG-DBN is designed to verify the idea in Section 3.3. It is actually a naïve method to randomly traverse the whole space to reach the optimal G. The expectation of traverse is \( L^{N_0} \). Random AG-DBN differs from AG-DBN in that it randomly generates G while not following Equation 2. Validation data are used to help find the best G in \( N_0 \) rounds.

4.2 Experimental Results

Table 2 shows the classification errors of all models. Without doubt, AG-DBN achieves the best performance both on a-ImageNet and on a-Yahoo. Specifically, when using raw pixels as input, AG-DBN reduces classification error on a-ImageNet by 4.01% and on a-Yahoo by 3.29% compared with the best baselines. Other observations include (1) The result shows that attributes can actually improve performance as T-DBN outperforms DBN. (2) Meanwhile, Manual AG-DBN performs better than T-DBN, skip-a, and skip-ad on a-ImageNet, which indicates the prior knowledge of connection does help. (3) Interestingly, skip-a and skip-ad perform close to DBN and T-DBN on a-ImageNet, while much better on a-Yahoo. Obviously, used as only visual inputs, the small ratio of a-ImageNet annotations helps little in improving performance. While on a-Yahoo, fully available attributes helps a lot. (4) Finally, AG-DBN performs better than Random AG-DBN since the heuristic learning of G should be more effective than random searching.

Table 2: Classification error of AG-DBN and baselines on a-ImageNet and a-Yahoo.

<table>
<thead>
<tr>
<th></th>
<th>a-ImageNet</th>
<th>a-Yahoo</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBN</td>
<td>0.8188</td>
<td>0.8469</td>
</tr>
<tr>
<td>T-DBN</td>
<td>0.7912</td>
<td>0.5012</td>
</tr>
<tr>
<td>Skip-a</td>
<td>0.7696</td>
<td>0.8524</td>
</tr>
<tr>
<td>Skip-ad</td>
<td>0.7551</td>
<td>0.5001</td>
</tr>
<tr>
<td>Manual AG-DBN</td>
<td>0.7646</td>
<td>0.2544</td>
</tr>
<tr>
<td>Random AG-DBN</td>
<td>0.6845</td>
<td>0.2015</td>
</tr>
</tbody>
</table>

Table 3 shows the optimal connections learned by AG-DBN on a-ImageNet. In general, people would more like to categorize the shape attributes rectangular, square, and round into the same semantic level as that in Manual AG-DBN. However, the learned G connects rectangular to the second hidden layer and connects square and round to the first hidden layer. This may due to the fact the model learns only from limited a-ImageNet data, while people conclude in more general style. Table 4 shows the optimal AG-DBN connections learned on a-Yahoo. It is interesting that the attributes of shape (e.g., 2D Boxy, 3D Boxy, Round) and material (e.g. Plastic, Wood, Glass, Shiny) are categorized as low-level concepts, while the attributes of body parts (e.g., Nose, Mouth, Face, and Head, Ear) are categorized into different levels.

Table 3: Setting of attributes on a-ImageNet

<table>
<thead>
<tr>
<th>Simple Concepts (Lower Hidden Layers)</th>
<th>Complicated Concepts (Higher Hidden Layers)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manual Setting</td>
<td>Learned Setting</td>
</tr>
<tr>
<td>black,blue,brown,green</td>
<td>blue,brown,gray,long,metallic</td>
</tr>
<tr>
<td>orange, pink,yellow,spotted,</td>
<td>orange, rectangular,red,smooth</td>
</tr>
<tr>
<td>violet,gray,red,white</td>
<td>spotted, wet, yellow</td>
</tr>
<tr>
<td>square, striped,vegetation,</td>
<td></td>
</tr>
<tr>
<td>striped,vegetation,</td>
<td></td>
</tr>
</tbody>
</table>

Influence of the Number of Data Annotated with Attributes In order to further evaluate our model, the data set are rebuilt to contain different ratios, 0, 0.1, 0.3, 0.5, 0.7, 0.9, 1.0, of annotated images. Ratio 0 means that none annotated image is included, while 1.0 means that all images are annotated. Here, \( N_0 \) is set to be 30 so that Random AG-DBN is more likely to find a better optimal G. Figure 5 shows that AG-DBN performs the best in all ratios except 0. As the ratio goes up, the error of AG-
DBN decreases sharply. The error of Random AG-DBN also falls down, however, in slower speed than AG-DBN. The error rates of Manual AG-DBN, T-DBN and skip-ad go down when the ratio goes up from 0 to 0.5 and finally remain nearly stable after ratio 0.5. The error of skip-a continues to decrease even after ratio 0.5. It performs better than skip-ad. Such difference can be viewed as an evidence to prove that attributes and raw images are of different abstraction levels hence should be connected to different hidden layers. It can also be found that Random AG-DBN performs better than Manual AG-DBN and slightly worse than AG-DBN. This is related to the fact that gates in Random AG-DBN are stochastic, and the number of iteration to learn Random AG-DBN is large (30 in this experiment). The reported results of Random AG-DBN are the best results over 30 iterations using evaluation data, while the reported results of AG-DBN are the results after the 30th iteration. As we explained in Section 3.3, the best results found by Random AG-DBN will be better as the number of iterations goes larger.

(a) Classification error of AG-DBN and baselines on a-Yahoo data set (input = AlexNet feature).

(b) Classification error of AG-DBN and baselines on a-Pascal data set (input = VGG16 or VGG19 feature).

Table 5: Classification error when using CNN feature.

Learning from CNN Feature In order to validate the extension method proposed in Section 3.3, we design an additional experiment. In this experiment, all models use features from the last pooling layer of CNN (AlexNet, VGG16, or VGG19) pre-trained on the whole ImageNet data set. and 2 hidden layers are both of size 4096 to make the comparison fair. And as an alternative to DBN in above experimental setting, we fine-tune the CNN model as baseline.

Table 5 shows the test error of AG-DBN and baselines. First, AG-DBN still achieves the best performance, and specifically it reduces test error by 3.68% compared with AlexNet on a-Yahoo, by 13.37% compared with VGG16, and by 13.14% compared with VGG19 on a-Pascal (this data set is also collected by [Farhadi et al., 2009], and attributes in a-Pascal is the same as a-Yahoo). Second, the performances of T-DBN, Skip-ad and Skip-a are very close. This dues to that all the three models use robust and discriminative features compared to raw image pixels, and that human defined attribute-layer connections help little in this situation, while with learned appropriate connections, AG-DBN can better utilize attributes.

5 Conclusion and Future Work

This paper proposes a novel deep structure named AG-DBN to better integrate attributes into deep learning. AG-DBN introduces appropriate connections between attributes and hidden layers automatically through a learnable gating mechanism. Experimental results on a-ImageNet, a-Yahoo and a-Pascal show that AG-DBN significantly outperforms the baselines in term of classification error. Particularly, the proposed AG-DBN also shows superiority against CNN model like AlexNet, VGG. Besides, we also provide empirical analysis on the influence of attributes presented, and the method for generating missing attributes in AG-DBN. In future, we intent to integrate our technique into other sophisticated deep structures, such as LSTM, etc., to improve their performance.

Acknowledgments

The work is supported by National Basic Research Program of China (2015CB352300), National Natural Science Foundation of China (61571269), and Key technology R&D Program of Liaoning and Shenyang (2017231005, 17-192-9-00).
The authors would also like to thank anonymous reviewers for their suggestions on the paper.

References


