Learning to Transfer: Generalizable Attribute Learning with Multitask Neural Model Search

Zhi-Qi Cheng[1,3], Xiao Wu[1*], Siyu Huang[2,3], Jun-Xiu Li[1], Alexander G. Hauptmann[3], Qiang Peng[1]
[1]Southwest Jiaotong University, [2]Zhejiang University, [3]Carnegie Mellon University
zhiqicheng@gmail.com,{wuxiaohk,lijunxiu,qpeng}@home.swjtu.edu.cn,siyuhuang@zju.edu.cn,alex@cs.cmu.edu

ABSTRACT

As attribute leaning brings mid-level semantic properties for objects, it can benefit many traditional learning problems in multimedia and computer vision communities. When facing the huge number of attributes, it is extremely challenging to automatically design a generalizable neural network for other attribute learning tasks. Even for a specific attribute domain, the exploration of the neural network architecture is always optimized by a combination of heuristics and grid search, from which there is a large space of possible choices to be searched. In this paper, *Generalizable Attribute Learning Model (GALM)* is proposed to automatically design the neural networks for generalizable attribute learning. The main novelty of *GALM* is that it fully exploits the *Multi-Task Learning* and *Reinforcement Learning* to speed up the search procedure. With the help of parameter sharing, *GALM* is able to transfer the pre-searched architecture to different attribute domains. In experiments, we comprehensively evaluate *GALM* on 251 attributes from three domains: animals, objects, and scenes. Extensive experimental results demonstrate that *GALM* significantly outperforms the state-of-the-art attribute learning approaches and previous neural architecture search methods on two generalizable attribute learning scenarios.

KEYWORDS

attribute learning, reinforcement learning, neural architecture search

ACM Reference Format:

Zhi-Qi Cheng, Xiao Wu, Siyu Huang, Jun-Xiu Li, Alexander G. Hauptmann, Qiang Peng. 2018. Learning to Transfer: Generalizable Attribute Learning with Multitask Neural Model Search. In *2018 ACM Multimedia Conference (MM '18), October 22-26, 2018, Seoul, Republic of Korea.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3240508. 3240518

1 INTRODUCTION

Attributes are nameable properties of objects, which are observable from visual images. As it possesses versatile properties, attribute learning serves as the basic building blocks for object classification and detection [17, 19, 38] as well as instance description and recognition [28, 29, 37]. Generally speaking, attribute learning can benefit many traditional learning problems (e.g., object recognition [17], face verification [12] and zero-shot classification [31]). Moreover, since attributes provide a natural human-computer interaction channel, attribute learning is also explored for fashion recognition [35, 42] and image retrieval [7–9, 16, 40].

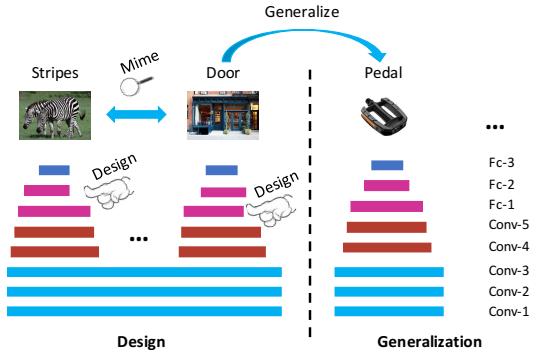

Figure 1: The goal of *GALM* is to automatically design a generalizable attribute learning method. It can automatically mime the attribute correlation and design the neural networks. Once it coverages on a specific attribute domain, the pre-search neural network architectures could be further transferred to other attribute domains.

Existing attribute learning methods mainly focus on utilizing the correlation among attributes to improve the performance of attribute prediction. The relationships between attributes and categories are explored to model the correlation in [1, 11, 20, 22]. With the success of deep learning, more and more multi-task attribute learning methods [12, 13, 25] are proposed to model the correlation with a tree-like neural architecture. More recently, a few research works begin to use the correlation among different attribute domains to improve the accuracy of attribute recognition in each single domain [11, 26, 36].

Although these methods achieve good performance, there are three obvious limitations: 1) The correlation among attributes is manually defined. Due to the limitation of human cognition, there are a lot of irrational correlations. 2) The generalization of attribute learning is ignored. Even for the same attribute learning tasks, existing well-designed neural networks cannot be used to generalize for other attribute domains. 3) The neural architectures of existing attribute learning models are designed by human experts, which require a wealth of experiences and skills. It has huge potential to be improved.

To address these issues, *Generalizable Attribute Learning Model (GALM)* is innovatively proposed in this paper. For easy understanding, a blueprint of *GALM* is illustrated in Fig. 1. Given a specific attribute domain, *GALM* first imitates the behaviors of human

*Xiao Wu is the corresponding author.

MM '18, October 22–26, 2018, Seoul, Republic of Korea

ACM ISBN 978-1-4503-5665-7/18/10…$15.00
https://doi.org/10.1145/3240508.3240518

experts to analyze the attribute correlation and keep exploring different network configurations to design the neural network architectures. When *GALM* is converged after a huge number of exploration, the optimized attribute correlation and the neural network architecture are obtained. Since the pre-trained *GALM* has learned a prior knowledge of the design strategy for the neural network, in addition to the automatically designing the neural network architecture for a single attribute domain, it can be used to address the attribute learning tasks for other attribute domains.

In general, the contributions of this paper are summarized as:

- **Generalization**: To the best of our knowledge, this is the first work to design the neural networks for the generalizable attribute learning task. Different from previous attribute learning method, our approach can simultaneously mine the correlation among attributes and design neural network architectures in an automated way. Moreover, once it converges on a specific attribute domain, the pre-searched neural architectures can be transferred to other domains to support other attribute learning tasks.
- **Efficiency**: Compared with previous neural architecture search methods, *Multi-Task Learning* is first integrated into *Reinforcement Learning* to significantly boost the efficiency by searching the neural network architectures of multiple attributes at the same time. As far as we know, this is the first method that can search neural network architectures on 251 attributes in a single Nvidia Titan XP GPU.
- **Experiments**: Extensive experiments demonstrate that *GALM* substantially outperforms the human-designed state-of-the-art attitude learning approaches on two general attribute learning scenarios. Since attribute learning can benefit other object detection and recognition tasks, our method has huge potential applications and commercial values.

The rest of this paper is organized as follows: The related work is first introduced in Section 2. The proposed GLAM is described in Section 3. The experiments are presented in Section 4. Finally, we conclude the paper in the last section.

2 RELATED WORK

Our work is related to *Attribute Learning* and *Neural Architecture Search*, which are briefly reviewed in this section.

2.1 Attribute Learning

Attribute learning could benefit many traditional learning tasks. In general, previous works mainly focus on modeling the correlation among attributes. Specifically, the relationships between attributes and categories are utilized to model the correlation for cross-category generalization [1, 11–13, 20, 22, 25]. A hyper-graph is deployed in [15] to explore the correlation among attributes for cross-category attribute prediction. A unified multiplicative framework is presented in [22], in which images and category information are jointly projected into a shared feature space for attribute prediction. Moreover, a huge number of zero-shot learning methods are proposed to use the attribute learning for image classification [6, 32, 41].

Recently, more and more deep neural networks have achieved great success in attribute learning [12, 13, 25]. A multi-task attribute learning method is proposed in [12], in which several group-shared loss-functions are applied on a manually designed tree structure to model the correlation. Meanwhile, a fully adaptive searching method is presented in [25] to automatically search a tree-structured neural network architecture instead of the efforts of human experts. In general, these works only investigate mining attribute information from the same domain, which are regarded as *Semantic Attribute Learning*.

More recently, a few works consider the knowledge transferring among different domains [11, 26, 36]. A novel CNN architecture is proposed in [36], in which a domain confusion loss is employed to learn a domain invariant representation for attribute prediction. An attention-guided transfer architecture is proposed in [26], where the coefficients of attention weights are utilized to deploy the cross-domain generalization. Following the convention of [26], they are treated as *Non-Semantic Attribute Learning*.

Although these methods have recognized the importance of exploiting the correlation and improving the generalization, they are unable to solve the generalizable attribute learning task. Different from existing attribute learning methods, our work aims to completely replace the human experts to analyze the attribute correlation and automatically design the neural networks for the generalizable attribute learning.

2.2 Neural Architecture Search

There is a growing amount of research works focusing on the neural architecture search to automatically design the neural network. A variety of approaches is proposed in previous works, including random search [3], Bayesian optimization [18, 34], evolutionary algorithm [31] and reinforcement learning [43, 44]. In general, one major challenge in neural architecture search is its huge computing cost. For instance, 800 GPUs are used to discover the convolutional architecture on the Cifar-10 dataset, which takes 28 days [43]. To address this problem, various approaches are proposed to accelerate search procedure from different perspectives, including performance prediction [2], iterative search method [23], hierarchical representation [24], hyper-networks [4], parameter sharing [30] and transfer leaning [5].

Although existing neural architecture search methods have been successfully deployed to design neural networks, they still need extensive sampling, constructing and training operations. On the other hand, there is a huge number of attributes for the generalizable attribute learning. When existing methods are directly used to search neural architecture for each attribute, there are countless attempts to obtain a generalizable neural network. Therefore, these methods are obviously incompetent to handle the generalizable attribute learning.

3 OUR APPROACH

In this section, we will elaborate the proposed *Generalized Attribute Learning Model (GALM)*. The problem formulation is first introduced in Section 3.1. The framework and each component of *GALM* will be described in Sections 3.2-3.4. Finally, the training algorithm and the generalization scheme are illustrated in Section 3.5.

Table 1: The search space of *GALM*.

Shared	Skip-Connection	Kernel	Channel
Conv-1	—	7*7	16
Conv-2	—	3*3	32
Conv-3	—	3*3	64
Search	**Skip-Connection**	**Kernel**	**Channel**
Conv-4	Conv-{1,2}-Conv-4	{3*3,5*5}	32
Conv-5	Conv-{1,2,3}-Conv-5	{3*3,5*5}	16
Fc-1	—	—	{64,128.256}
Fc-2	—	—	{64,128.256}
Fc-3	Fc-{1,2}-Fc-3	—	2

3.1 Problem Formulation

In contrast to conventional attribute learning methods, which manually design the neural networks for a specific domain of attributes, we propose *Generalizable Attribute Learning Model (GALM)*, which tries to automatically design the neural networks for attributes learning tasks. We have carefully reviewed the deep attribute learning methods in previous literatures. Without loss of generality, these methods are based on a tree-like structure to model the correlation among attributes, where the root node of the tree is shared for all attributes and the leaf nodes are designed for individual attributes. Inspired by these works, the goal of *GALM* is to automatically design a tree-like neural network architecture having the optimal learning performance.

By comprehensive consideration of parameter numbers, training efficiency and evaluation performance, the search space of *GALM* is described in Table 1, where the first three convolutional layers are shared for all attributes, while the last two convolutional and three fully connected layers are designed for each attribute. For each attribute, two sets of choices are considered: 1) Which previous layers are connected to, and 2) What configuration is to be used? Specifically, for the skip connection, two types of connections are considered. One is for Conv-4 and Conv-5, which is similar to [14]. Another one is for Fc-3, which determines if Fc-2 is used. For the configuration, we only search the kernel sizes of Conv-4, Conv-5 and the dimensions of Fc-1 and Fc-2. The Fc-3 layer is fixed for each attribute. After each convolutional layer, a ReLU layer, a max-pooling layer with a kernel size of 2*2 and a stride of 2, and a Batch Normalization (BN) layer are adopted. It is noted that BNs are fixed in the inference phase.

Since all choices are independent, there are $2 * 2 * 3 * 3 = 36$ configurations and $4*8*2 = 64$ skip connections. For N attributes, if we directly optimize on entire attribute domains as [43, 44], the number of possible networks will be extremely increased to $\{36 * 64\}^N = 2304^N$. Facing the huge scope of network search space, it is challenging to obtain an optimal neural architecture. Therefore, *Multi-Task Learning* is used to speed up the search. Specifically, we treat attributes as different tasks to reduce the search space. Finally, with the help of *Multi-Task Learning*, the search space is reduced to $36 * 64 * N = 2304 * N$. Noted that, it is still difficult to search for such search space. Inspired by previous works [30, 44], *Reinforcement Learning* is used to further speed up the searching.

Specifically, the searching process of *GALM* is treated as a *sequential decision process*, which is formulated as a tuple $< S, A, \phi, R >$:

- $S = \{s_1, s_2, .., s_T\}$ is the state set, which corresponds to neural network architectures at different timestamps, where T is the number of timestamps.

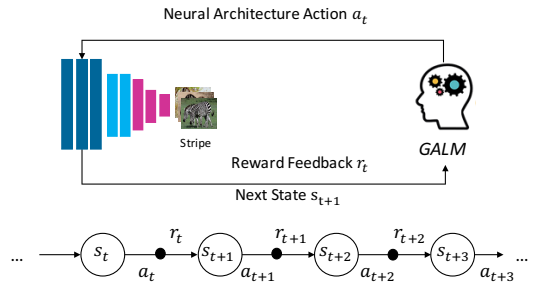

Figure 2: The sequential decision process of *GALM*.

- $A = \{a_1, a_2, ..., a_T\}$ is the action set, in which each action is a choice of neural network architecture settings.
- ϕ is the policy function with parameter θ, which is used to generate the corresponding neural architecture action a_t based on the current neural network architecture s_t as $\phi_\theta(s_t) = a_t$. Without confusion, we also treat θ as the parameter of *GALM* in this paper.
- $R = \{r_1, r_2, ..., r_T\}$ is the reward set. At each timestamp t, the reward r_t is calculated to evaluate the performance of the current neural network architecture. Specifically, in our implementation, the validation accuracy of current mini-batch data is used as the reward.

During the training phase, *GALM* keeps imitating the behaviors of human experts to construct the corresponding neural network architectures. As Fig. 2 shows, at timestamp t, a neural network is constructed by an action a_t, and some measurements (such as accuracy) are provided as the reward r_t to verify the action a_t. At the same time, a state s_{t+1} is estimated to model the state of the neural network. After countless attempts, an optimized neural network architecture is derived once *GALM* is converged. In addition to automatically designing the neural model for a given attribute learning task, the learned *GALM* can be further used to address attribute learning problems in other domains, since it has learned a prior knowledge of generalizable neural architectures and training strategies.

3.2 Framework

To demonstrate our contribution and innovation, the framework of previous neural architecture search method is first briefly described, followed by the framework of the proposed *GALM*.

3.2.1 Framework of previous neural architecture search methods. The framework of previous neural architecture search methods [43, 44] is illustrated in Fig. 2, in which RNN network is used to automatically design the neural network architecture and maximize the expected performance. At each timestamp, RNN network interactively samples a sequence of actions. Here, every action is an architecture design choice, such as heights, widths and strides of CNN filters. With these actions, different child neural networks are then constructed and trained until convergence. To train RNN network, the performance of each child neural network is served as a reward. The RNN network is then optimized by the policy gradient algorithm. Specifically, with the optimization of RNN network, the probability of the neural network actions having better performance is improved in the training phase. Once RNN network converges, an optimized probability distribution over the search space

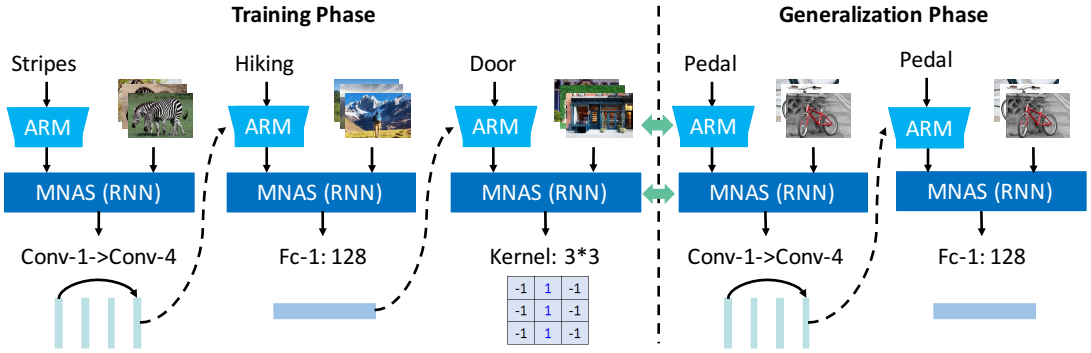

Figure 3: The framework of *GALM*. (1) The attribute correlation is maintained and updated by *ARM* to learn differentiated attribute embeddings over time. (2) At each iteration of the multitask training, an attribute is randomly sampled. The attribute embedding (i.e., the output of *ARM*) is passed to *MNAS* along with the selected neural architecture action embedding. The output of *GALM* is a series of neural architecture actions, such as heights, widths and strides of CNN filler, which are used to design the neural network architecture for different attribute learning tasks.

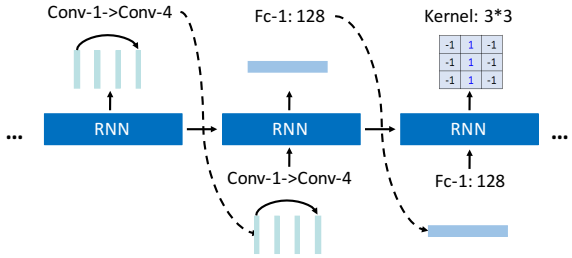

Figure 4: The previous neural architecture search framework, which is used to compare with *GALM*.

is learned. Finally, this probability distribution is used to obtain the optimized neural network architectures.

Although previous neural architecture search framework is successfully applied to design neural networks for image classification [43, 44], it still needs extensive sampling, constructing and training operations. Since we have to face a huge number of attributes among different domains, this framework is obviously unable to solve the generalizable attribute learning.

3.2.2 Framework of GALM. Compared with previous frameworks, *Generalizable Attribute Learning Model (GALM)* is proposed to efficiently discover the optimal neural networks for attributes learning tasks. The framework of GLAM is illustrated in Fig. 3. Two innovative components, *Attribute Representation Model (ARM)* and *Multitask Neural Architecture Search (MNAS)* are presented as follows:

- **Attribute Representation Model (ARM)** is proposed to learn the correlations of attributes, in which the correlations are maintained and updated with the neural network architecture search. Our method is modeled as a conditional framework with ARM, in which the actions are generated according to the embeddings of attributes, so that the model could simultaneously conduct neural architecture search on multiple attributes. Meanwhile, It helps to understand the correlation among attributes, so that the neural architecture of similar attributes could be efficiently transferred, which significantly speeds up the search procedure.
- **Multitask Neural Architecture Search (MNAS)**: Similar to the framework of previous neural architecture search, *MNAS* is also implemented with RNN network. Since attributes are commonly associated, multi-task learning is employed to obtain potential improvement by utilizing "correlation" among neural network architectures. In addition, with the help of sharing the parameters among *MNAS*, the pre-searched neural network architectures are efficiently transferred to other domains.

ARM is along with *MNAS* for complete search phase. At each timestamp, an attribute and its corresponding training examples are randomly sampled. *ARM* is employed to embed the attribute and the neural network design actions from previous timestamp. After that, embeddings are fed into *MNAS* for the neural architecture search. When these two models are converged, the optimal correlation and network architecture are obtained. Moreover, with sharing parameters, *GALM* can further transform the pre-optimized neural network architecture from source domains to any target domains.

In general, the main novelty of *GALM* lies in the simultaneous search neural network architecture over multiple attributes. Due to the correlations among attributes, different attributes may require similar neural network architecture. With the help of *ARM*, we can easily transfer the pre-searched neural network architectures to similar attributes. On the other hand, with the parameters sharing between *ARM* and *MNAS*, the pre-searched attribute learning networks could be efficiently transferred to different domains.

3.3 Attribute Representation Model

Attribute Representation Model (ARM) is proposed to embed the attributes. Specifically, given the attribute set $E = \{e_1, e_2, ..., e_N\}$, where e_i denotes an embedding of an attribute, which is randomly initialized. N is the number of attributes. A fully connected layer is used to embed e_i as:

$$\hat{e_i} = W_\theta e_i \tag{1}$$

where W_θ is the parameters of the fully connected layer. Once *ARM* converges, we extract embeddings from *ARM* as the attribute representation $\hat{e_i}$, and cosine distance is then calculated to obtain the correlation among attributes.

Different from previous works [12, 13], in which the correlations among attributes are manually defined, the correlation of *ARM* is maintained and updated with *MNAS*. During the training, *ARM* can automatically optimize the correlation to remedy the uncertainty, since ARM is a fully connected layer in essence, which is easier to be maintained and updated along with neural model search. When the optimal neural network architectures are obtained, the correlation among these attributes is refined at the same time.

Moreover, with the help of *ARM* model, our method is modeled as a conditional framework, in which the neural architecture actions are generated based on the embeddings of attributes. Using this conditional framework, *GALM* can use the *Multi-Tasking Learning* to search neural architectures for multiple attributes at the same time. Since similar attributes may require the approximately same neural network design [12, 13], *GALM* can easily transfer the pre-searched neural architectures to similar attributes. Compare with previous neural architecture search methods, *ARM* can significantly improve the efficiency of the neural architecture search.

3.4 Multitask Neural Architecture Search

Compared with previous works, *Multitask Neural Architecture Search (MNAS)* explores two novel aspects: 1) Instead of discovering the optimized neural network architecture for all attributes, it gradually optimizes the sub-neural networks for each attribute. It significantly reduces the search space and eliminates the negative impacts by assigning each attribute into each individual dynamic environment. 2) With the help of *ARM*, it is able to transfer the pre-searched neural network architectures to similar attributes in the search process. This means *MNAS* can use the "correlation" of neural networks to significantly improve the performance and effectively speed up the neural architecture search by obtaining a better initialization with the pre-search similar attributes.

As shown in Fig. 3, *MNAS* is synchronously trained on a set of N attributes. We uniformly sample an attribute at the beginning of each RNN training iteration. *ARM* is then used to represent the attribute to a unique embedding vector. Noted that, these attribute embeddings can extend *MNAS* into a condition model, in which the neural architecture action is generated based on these attribute embeddings. With this conditional framework, *MNAS* can simultaneously search differentiated architectures for multiple attributes.

Mathematically, *MNAS* predicts a list of actions $a_{1:T}$ to design the neural network architecture for all attributes. Once it converges, the designed neural network will achieve the reward set R on the validation dataset. We can use it as the reward signal and use *Reinforcement Learning* to train MNAS. To find the optimal architecture, we maximize its expected reward, represented as $J(\theta)$:

$$J(\theta) = E_{P(a_{1:T};\theta)}[R] \tag{2}$$

Since the reward signal R is non-differentiable, a policy gradient method is used to iteratively update θ. In this work, the REINFORCE rule from [39] is adopted:

$$\nabla_\theta J(\theta) = \sum_{t=1}^{T} E_{P(a_{1:T};\theta)}\Big[\nabla_\theta \log P(a_t|a_{(t-1):1};\theta)R\Big] \tag{3}$$

An empirical approximation of the above quantity is:

$$\frac{1}{M}\sum_{k=1}^{M}\sum_{t=1}^{T}\nabla_\theta \log P(a_t|a_{(t-1):1};\theta)r_k \tag{4}$$

where M is the number of different architectures that RNN network samples in one batch. T is the number of timestamps. r_k is the validation accuracy that the k-th neural network architecture achieves after training.

Noted that, the aforementioned update is an unbiased estimation for the gradient, but has a very high variance. To reduce the variance of this estimation, a baseline function is employed:

$$\frac{1}{M}\sum_{k=1}^{M}\sum_{t=1}^{T}\nabla_\theta \log P(a_t|a_{(t-1):1};\theta)(r_k - b) \tag{5}$$

where the baseline function b does not depend on current action, which is still an unbiased gradient estimation. In this work, b is an exponential moving average of previous architecture accuracies. In addition, we follow similar parameter sharing scheme as [30] to speed up the search for each attribute. The reward set R is used to select models that generalize well rather than models that overfit on the training set, which is computed on the validation set instead of on the training set.

3.5 Training & Generalization

3.5.1 Training algorithm. There are two sets of parameters in *GALM*. 1) The parameters of *ARM* and *MNAS* are denoted as θ and W_θ, respectively, 2) The parameters of attribute learning neural networks are denoted by Ω.

In general, the training procedure of *GALM* consists of two interleaved phases. The first phase trains the parameters of attribute learning networks (i.e., Ω) on the training dataset. The *binary cross entropy* loss with the *Sigmod* activation function is used as our loss function. The second phase trains the parameters of *ARM* and *MNAS* (i.e., θ and W_θ) for a fixed number of timestamps. Typically the number of timestamps is set to 2000 in our experiments. These two phases are alternatively performed during the training of *GALM*.

More details are listed in Algorithm 1, in which *ARM* and *MNAS* are jointly trained. For a new unseen attribute, we copy corresponding *ARM* from the most correlated pre-searched attribute to initialize *ARM*. The most correlated attribute is obtained by calculating the cosine distance among the attribute embeddings. Based on this way, our method could transfer the pre-search attribute correlations and neural networks among similar attributes.

3.5.2 Generalization scheme. As can be seen from the right part of Fig. 3, the parameters of *ARM* and *MNAS* are shared. Therefore, it is straightforward to perform a transfer from the pre-searched domain to other domains. For example, given a new attribute, *ARM* is used to generate the corresponding embeddings. By comparing the cosine distance among embeddings of pre-trained attributes, the most correlated attribute is selected. The neural architecture of the selected attribute is then utilized to initiate the new attribute. In the generalization phase, the neural architecture search is only conducted on the new attribute. Since *ARM* and *MNAS* have learned a generic architecture design from the selected attributes, the search procedure of the new attribute will significantly speed up.

Algorithm 1 The training algorithm of *GALM*.

Input: Training data D, learning rate β.
for each episode $l = 1, 2, \ldots, L$ **do**
  Shuffle to get the mini-batches sequence $D = \{D_1, D_2, \ldots, D_T\}$.
  for $t = 1, \ldots, T$ **do**
    ▷ **% Sampling Phase %**
    Uniformly sample an attribute as a task.
    if the sampled attribute is an un-seen attribute **then**
      Initialize *ARM* with the most correlated pre-searched attribute via cosine distance among the embeddings from ARM.
    end if
    Sample neural architecture action a_t.
    ▷ **% Constructing Phase %**
    Construct neural networks by the neural architecture action a_t.
    Train the constructed neural networks Ω on the training dataset until it converges.
    Receive reward r_t on the validation data and calculate the exponential moving average b.
    ▷ **% Updating Phase %**
    Update *MNAS* policy $\theta = \theta + \beta(r_t - b)\frac{\partial log P_{\theta_c}(a_t|a_{t-1:1};\theta)}{\partial \theta}$
    Retain the gradient of *MNAS* and update W_θ of *ARM*.
  end for
end for
Output: MNAS policy $p_\theta(a_{1:T})$, *ARM* parameter W_θ.

4 EXPERIMENTS

4.1 Implementation Details

In this subsection, the implementation details of *ARM* and *MNAS* are introduced. *ARM* and *MNAS* are implemented as a fully connected layer with 50 hidden, and a 2-layer LSTM with 100 hidden units, respectively. The weights of *MNAS* and the embedding model are uniformly initialized at random, yielding an approximately uniform distribution over actions. The learning rate is set to 10^{-4}. It is noted that *MNAS* is optimized at each timestamp, i.e. the batch size is set as 1. *ARM* and *MNAS* are jointly trained by REINFORCE [39]. When *ARM* and *MNAS* converge, we fine-tune the selected neural network on both of the training set and the validation set, and then report its performance on the testing set.

4.2 Experiment Settings

4.2.1 Evaluation Tasks. We evaluate our method on two generalized attribute learning tasks, i.e. *Semantic Attribute Learning* and *Non-Semantic Attribute Learning*.

- **Semantic Attribute Learning** means that the transfer is deployed on different class categories from the same attribute domain, which has been studied in [1, 20, 22].
- **Non-Semantic Attribute Learning** means the transfer is conducted on disjointed domains, in which the attribute sets are from different domains without intersection. The aim of the experiments is to verify whether the general compatibility of *GALM* can truly improve the performance.

4.2.2 Datasets. In order to be consistent with previous works [1, 20, 22], three datasets are used in our experiments. For fair comparison, the same split way is used.

- **AWA** [21] contains 30,475 images of 50 animal classes. Each class is annotated with 85 attributes. Following [1, 20], we divide the dataset into 40 classes (24,295 images) for training and 10 classes (6,180 images) for testing.
- **aPaY** [10] consists of 12,695 images from PASCAL VOC 2008 dataset as training set, and 2,644 images collected from Yahoo image search engine as testing set. Both sets have disjointed classes (i.e., 20 classes for PASCAL and 12 classes for Yahoo). Each class is annotated with 64 attributes.
- **SUN** [29] is a subset of SUN Database for fine-grained scene categorization. Totally, there are 14,340 images from 707 classes, in which 600 classes are used for training and 107 classes for testing. Each image is annotated with 102 binary attributes to describe the material and surface properties.

4.2.3 Baselines. We compare *GALM* with the following state-of-the-art attribute learning approaches. Since some baselines are not deep learning methods, VGG16 [33] is used to extract 4,096-dimensional CNN features for these methods.

- **DAP** [20]: It is a method for attribute prediction. Linear SVM is used to train attribute classifiers separately.
- **IAP** [20]: It first maps inputs to the seen classes and then predicts the attributes, which is an indirect method for attribute prediction.
- **ALE** [1]: A method for image classification, in which each class is embedded in the space of attribute vectors. It utilizes the alternative information (e.g. class hierarchies) for attribute learning.
- **HAP** [15]: It uses a hyper-graph to explore the correlation for attributes learning. Since it has a lot of variations, we use the kernel alignment version to obtain the best performance.
- **UMF** [15]: A unified multiplicative framework for attribute learning, in which images and category information are jointly projected into a shared feature space for attribute prediction.
- **MDG** [11]: It gears the techniques in multi-source domain generalization for the purpose of learning cross-category generalizable attribute detectors.
- **FMT** [25]: An automatic approach for designing compact multi-task deep learning architectures, in which a tree-like structure is learned by a greedy algorithm.
- **AMT**[13]: A multi-task deep convolutional neural network, in which an auxiliary network is used to explore the attribute relationships for improved performance.
- **FT** [27]: A fine-tuning based method, in which the model is first trained from the source domain, and then fine-tuned by the target domain.
- **AFS** [26]: An attention-guided transfer architecture, in which the coefficients of attention weights are utilized to deploy the transfer.

4.2.4 Evaluation metrics. Similar to previous works [1, 20, 22], the performance of attribute predictors are measured by mean area under ROC curve (mAUC).

4.3 Semantic Attribute Learning

The performance comparison of *semantic attribute learning* is listed in Table 2. The italic results are from our implementation, while others are reported by the authors. From Table 2, we can find that

Table 2: The performance comparison of *Semantic Attribute Learning*.

	AWA [21]	**aPaY** [10]	**SUN** [21]
DAP [20]	72.8	77.4	81.4
IAP [20]	*74.1*	*78.1*	*81.9*
ALE [1]	*65.7*	*69.2*	*74.5*
HAP [15]	74.2	*58.2*	*76.7*
UMF [22]	*76.7*	*79.7*	*80.5*
MDG [11]	82.2	82.3	*85.8*
AMT [13]	*85.5*	***84.5***	*82.5*
FMT [25]	*72.2*	*70.5*	*75.5*
GALM	**86.5**	84.2	**86.5**

GALM outperforms all baselines on two datasets, which clearly validates our assumption that *GALM* improves the generalization of attribute detectors for previously unseen categories. For aPaY dataset, although AMT achieves the best performance, our approach has the competitive performance. The main reason is that the last two fully connected layers of each attribute are all connected to other attributes, so that it uses more parameters than our method.

Except for AMT and FMT, we can directly compare our method with other baselines to evaluate the effectiveness of the generalization, because they share the same input. Since IAP learns a map from attributes to class categories, it has more generalization capability than DAP, having a better performance. Meanwhile, because ALE focuses on image classification, it does not perform quite well on the attribute prediction task. The generalization of HAP is unstable since it relies on a hyper-graph, which is sensitive for different datasets. UMF considers the attribute learning and classification at the same time, which has some improvement. MDG and AMT have a better performance than other baselines, from which the correlations of attributes are utilized in both methods. Similar to GALM, FMT also optimizes a tree-like structure by using a greedy method. However, the performance is not satisfactory. The main reason is that the greedy algorithm is easy to over-fitting.

4.4 Non-Semantic Attribute Transfer

The performance comparison of *non-semantic attribute learning* is listed in Table 3. For convenience, AWA, aPaY and SUN datasets are denoted as 1, 2 and 3, respectively. *S* and *D* refer to the source and target domains, respectively. For example, S1-D2 means that we generalize the model from AWA to aPaY. The impacts of the same domain are ignored. The last 3 columns use all attribute domains. Symbol $*$ indicates the improvement when the source domain is utilized. As FT and AFS are only used for non-semantic attribute learning, they do not have the symbol $*$.

From Table 3, we can find that GALM almost outperforms all baselines on every generalization scenario. The success of GALM is mainly due to the combination of transfer learning with the neural network architecture search. Although FT and AFS have better performance in two cases, overall performance is worse. Importantly, GALM significantly beats other baselines, when the attributes of the target domain are not used as the source domain, referring to the first 9 columns. It means GALM can effectively generalize the pre-search neural network for domains that are semantically unrelated to the source domain. On the other hand, other baselines are quite poor, in which symbol $*$ is very rare in the first 9 columns.

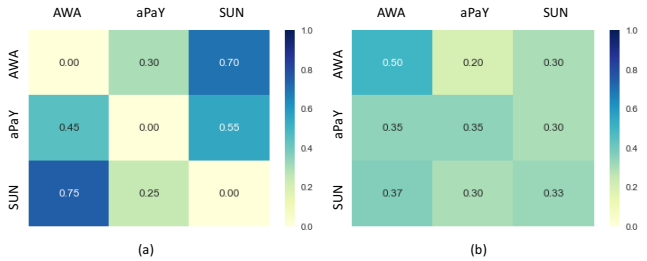

Figure 5: The correlation of different domains.

This indicates that they are not suitable for the generalizable attribute learning. Furthermore, since the last 3 generalization scenarios using more data, i.e, attributes from the target domain, they outperform other generalization scenarios. This result highlights the benefit of the correlations among different attribute domains.

4.5 Analysis of ARM

To analyze the effectiveness of *ARM*, the correlations among different attribute domains are illustrated in Fig. 5. We extract embeddings from *ARM* to represent attributes, and then cosine distance is calculated to obtain the correlations. Due to the large number of attributes, we group attributes by their domains. Rows represent target attribute domains and columns refer to source attribute domains. It evaluates if *ARM* has successfully learned the correlations and if the correlations are the same as our expectation. In addition, for a given target domain, we also verify whether the most impact attributes are from the same domain or disjointed domains.

Fig. 5-a illustrates the correlations among different domains, in which the impact of the same domain is ignored. Overall, scenes (SUN) are closely related to animals (AWA), which have relatively high scores. Objects (aPaY) are almost equally related to animals (AWA) and scenes (SUN). It is interesting to see that this result is consistent with our expectation. Scene and animal domains are very correlated mainly because animal images always contain scenes. When the impact of the same domain is considered, as Fig. 5-b shows, the difference becomes small. Objects, scenes and animals do benefit from semantically related attributes. The overall within-domain model similarity is lower than 50%, which proves the value of generalization for attribute learning.

Some examples of the correlated attributes are illustrated in Table 5 to demonstrate what attributes are transferred across domains. For animal attributes, "tough-skinned" gives us the feeling of an "Enclose-area" and "stressful" situation. A "fast" animals might make people "scary", and a "hunter" often brings "metal" and "shiny" objects. For object attributes, "vegetation", "leaf" and "flower" are usually associated with a farm scene (i.e., "tree', "flower"" and "shrubbery"), and animals live in the environments (i.e., "plains", "forest" and "jungle"). For scene attributes, "competing" scenes might contain "glass" and "headlight" objects, and the color of "railroad" is roughly like "tough-skinned" animals. In general, while correlated attributes are selected from disjointed domains, it is possible to explain some correlations, and most of them have intuitive explanation. This is indeed what we expected when performing the non-semantic transfer. Therefore, *ARM* can effectively mine the attribute correlations.

Table 3: The performance comparison of *Non-Semantic Attribute Learning*.

	S1-D2	S1-D3	S2-D1	S2-D3	S3-D1	S3-D2	S1,2-D3	S1,3-D2	S2,3-D1	S1,2,3-D1	S1,2,3-D2	S1,2,3-D3
DAP [20]	78.4*	80.1	71.4	81.2	73.5*	79.2*	80.1	78.1*	72.1	75.2*	83.4*	82.5*
IAP [20]	74.7	83.3*	75.5*	81.4	77.2*	75.3	79.2	74.2	72.4	74.7*	77.5	77.2
ALE [1]	64.2	73.9	64.3	75.2*	62.4	64.4	74.4	63.7	60.1	69.2*	61.2	72.4
HAP [15]	64.2*	72.7	78.5*	74.6	76.4*	64.5*	72.3	65.5	77.4*	82.3*	69.8*	77.3*
UMF [22]	82.3*	83.5*	77.5*	82.2*	78.5*	83.2*	83.5*	84.2*	79.5*	82.5*	85.7*	84.2*
MDG [11]	82.3*	86.9*	84.5*	87.4*	85.1*	84.7*	89.2*	85.3*	83.1*	87.2*	87.2*	91.2*
AMT [13]	**87.1***	84.5*	86.2*	83.7*	84.1	87.5*	83.2*	86.2*	87.1*	86.1*	89.2*	83.4*
FMT [25]	72.5*	77.3*	77.7*	76.5*	81.3*	72.4*	77.9*	74.7*	85.3*	87.2*	84.7*	79.5*
FT [27]	85.1	89.4	90.5	89.1	88.7	87.5	91.5	89.4	**92.2**	91.2	93.2	91.5
AFS [26]	84.7	87.7	87.4	87.9	**92.4**	84.5	89.5	82.5	92.1	88.3	93.8	91.4
GALM	87.0*	**89.9***	**90.9***	**89.3***	90.2*	**87.8***	**91.9***	**91.2***	90.7*	**92.5***	**94.5***	**93.4***

Table 4: The analysis of the optimal neural network architectures.

Layers	AWA 14_AWA	aPaY 09_aPaY	SUN 14_Sun
Conv-4	Conv-1-Conv-4 (15.3%) Conv-2-Conv-4 (52.2%) 3*3 (87.1%), 5*5 (12.9%)	Conv-1-Conv-4 (19.4%) Conv-2-Conv-4 (55.3%) 3*3 (84.2%), 5*5 (15.8%)	Conv-1-Conv-4 (29.4%) Conv-2-Conv-4 (34.4%) 3*3 (82.2%), 5*5 (17.8%)
Conv-5	Conv-1-Conv-5(12.7%) Conv-2-Conv-5 (21.6%) Conv-3-Conv-5 (20.4%) 3*3 (93.8%), 5*5 (6.2%)	Conv-1-Conv-5 (19.4%) Conv-2-Conv-5 (28.42%) Conv-3-Conv-5 (4.7%) 3*3 (88.4%), 5*5 (11.6%)	Conv-1-Conv-5 (12.1%) Conv-2-Conv-5 (45.3%) Conv-3-Conv-5 (6.4%) 3*3 (84.5%), 5*5 (15.5%)
Fc-1	64 (15.2.%), 128 (37.8%), 256 (57.0%)	64 (5.6%), 128 (32.4%), 256 (72.0%)	64 (14.3%), 128 (47.7%), 256 (37.0%)
Fc-2	64 (17.4%), 128 (48.6%), 256 (34.0%)	64 (12.7%), 128 (49.3%), 256 (38.0%)	64 (15.8%), 128 (54.4%), 256 (30.8%)

The values in the parentheses (%) denote the frequency of each design choice.

Table 5: Examples of correlated attributes. Red color is target domain. Values in parentheses are the degree of correlation.

AWA [21]	**aPaY** [10]	**SUN** [29]
Tough-skinned, Fast, Hunter	Metal (0.86), Shiny (0.74), Skin (0.54)	Enclosed-area (0.68), Scary (0.60), Stressful (0.58)
Plains (0.83), Forest (0.81), Jungle (0.78)	Vegetation, Leaf , Flower	Trees (0.87), Flowers (0.64), Shrubbery (0.59)
Solitary (0.76), Paws (0.74), Tough-skinned (0.67)	Headlight (0.54), Glass (0.48), Mast (0.42)	Competing, Medical-activity, Railroad

4.6 Analysis of MNAS

A statistical analysis is conducted on the optimized neural architectures. Due to the large number of attributes, we group attributes by their domains and analyze the ratios of different network design options, as listed in Table 4. The values in the parentheses (%) denote the frequency of each design choice.

In general, some insights can be obtained as follows: 1) The kernel size of 3×3 is better than 5×5. 2) Three fully connected layers are better than two layers. 3) The dimensions of the last fully connected layers should be smaller than previous ones. 4) The skip connections from the middle layers are more effective. Additionally, compared with aPaY, we find the kernel size of SUN is larger and more skip connections from Conv-1 are used, which demonstrates that larger kernel sizes and skip connections from the first few layers focus more on the background information.

4.7 Efficiency

Conventional Neural Architecture Search is computationally expensive and time-consuming. 800 GPUs are used in [43] for 28 days to optimize the neural architectures on CIFAR-10 dataset. On the contrary, our method can use one GPU to build a generalizable attribute learning model. For instance, with a single Nvidia Titan XP GPU, our method can search 251 attributes over three attribute domains in 27.6 hours. Compared with previous neural architecture search methods, the improvement comes mainly from two aspects: 1) The adoption of multitask learning: With the help of *ARM*, the architectures of similar attributes are shared. 2) Parameter sharing: With the parameter sharing of *MNAS*, our method can transfer the pre-searched architecture to other attribute domains.

5 CONCLUSION

In this paper, we explore the problem of generalizable attribute learning by multitask neural architecture search. *Generalizable Attribute Learning Model (GALM)* is novelly proposed to automatically design the neural networks for attribute learning tasks. By jointly exploring two components (i.e., *ARM* and *MNAS*) on previous neural architecture search methods, *GALM* can significantly accelerate the search efficiency by transferring the distilled knowledge from pre-searched architectures to unseen attributes. Moreover, with sharing parameters, *GALM* can further be generalized for other target domains. Experiments on three domains with a total of 251 attributes demonstrate the effectiveness and efficiency of *GALM*. It significantly outperforms the state-of-the-art attribute learning methods, and yields substantially faster convergence speed than previous neural architecture search methods.

ACKNOWLEDGEMENTS

The authors thank for the beneficial discussions with Hieu Pham. This work was supported by the National Natural Science Foundation of China (Grant No. 61772436), the China Scholarship Council (Grant No. 201707000083), the Sichuan Science and Technology Innovation Seedling Fund (Grant Nos. 2017RZ0015, 2017018 and 2017020), the Fundamental Research Funds for the Central Universities, the Cultivation Program for the Excellent Doctoral Dissertation of Southwest Jiaotong University (Grant No. 2015310084), and the Innovative & Practice Project of Graduate School of Southwest Jiaotong University (Grant No. 2018CYPY06).

REFERENCES

[1] Zeynep Akata, Florent Perronnin, Zaid Harchaoui, and Cordelia Schmid. 2013. Label-embedding for attribute-based classification. In *CVPR*. 819–826.

[2] Bowen Baker, Otkrist Gupta, Ramesh Raskar, and Nikhil Naik. 2018. Accelerating neural architecture search using performance prediction. *ICLR*.

[3] James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *JMLR* 13, Feb (2012), 281–305.

[4] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. 2018. SMASH: one-shot model architecture search through hypernetworks. *ICLR*.

[5] Han Cai, Tianyao Chen, Weinan Zhang, Yong Yu, and Jun Wang. 2018. Efficient Architecture Search by Network Transformation. *AAAI*.

[6] Soravit Changpinyo, Wei-Lun Chao, Boqing Gong, and Fei Sha. 2016. Synthesized classifiers for zero-shot learning. In *CVPR*. 5327–5336.

[7] Zhi-Qi Cheng, Yang Liu, Xiao Wu, and Xian-Sheng Hua. 2016. Video ecommerce: Towards online video advertising. In *ACM MM*. 1365–1374.

[8] Zhi-Qi Cheng, Xiao Wu, Yang Liu, and Xian-Sheng Hua. 2017. Video ecommerce++: Toward large scale online video advertising. *TMM* 19, 6 (2017), 1170–1183.

[9] Zhi-Qi Cheng, Xiao Wu, Yang Liu, and Xian-Sheng Hua. 2017. Video2Shop: Exactly Matching Clothes in Videos to Online Shopping Images. In *CVPR*. 4169–4177.

[10] Ali Farhadi, Ian Endres, Derek Hoiem, and David Forsyth. 2009. Describing objects by their attributes. In *CVPR*. 1778–1785.

[11] Chuang Gan, Tianbao Yang, and Boqing Gong. 2016. Learning attributes equals multi-source domain generalization. In *CVPR*. 87–97.

[12] Hu Han, Anil K Jain, Shiguang Shan, and Xilin Chen. 2018. Heterogeneous face attribute estimation: A deep multi-task learning approach. *TPAMI* (2018).

[13] Emily M Hand and Rama Chellappa. 2017. Attributes for Improved Attributes: A Multi-Task Network Utilizing Implicit and Explicit Relationships for Facial Attribute Classification.. In *AAAI*. 4068–4074.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*. 770–778.

[15] Sheng Huang, Mohamed Elhoseiny, Ahmed Elgammal, and Dan Yang. 2015. Learning hypergraph-regularized attribute predictors. In *CVPR*. 409–417.

[16] Sung Ju Hwang and Leonid Sigal. 2014. A unified semantic embedding: Relating taxonomies and attributes. In *NIPS*. 271–279.

[17] Dinesh Jayaraman and Kristen Grauman. 2014. Zero-shot recognition with unreliable attributes. In *NIPS*. 3464–3472.

[18] Kirthevasan Kandasamy, Willie Neiswanger, Jeff Schneider, Barnabas Poczos, and Eric Xing. 2018. Neural Architecture Search with Bayesian Optimisation and Optimal Transport. *arXiv:1802.07191* (2018).

[19] Pichai Kankuekul, Aram Kawewong, Sirinart Tangruamsub, and Osamu Hasegawa. 2012. Online incremental attribute-based zero-shot learning. In *CVPR*. 3657–3664.

[20] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. 2009. Learning to detect unseen object classes by between-class attribute transfer. In *CVPR*. 951–958.

[21] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. 2014. Attribute-based classification for zero-shot visual object categorization. *TPAMI* 36, 3 (2014), 453–465.

[22] Kongming Liang, Hong Chang, Shiguang Shan, and Xilin Chen. 2015. A unified multiplicative framework for attribute learning. In *ICCV*. 2506–2514.

[23] Chenxi Liu, Barret Zoph, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. 2017. Progressive neural architecture search. *arXiv:1712.00559* (2017).

[24] Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. 2017. Hierarchical representations for efficient architecture search. *arXiv:1711.00436* (2017).

[25] Yongxi Lu, Abhishek Kumar, Shuangfei Zhai, Yu Cheng, Tara Javidi, and Rogerio Feris. 2017. Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification. In *CVPR*. 5334–5343.

[26] Nils Murrugarra-Llerena and Adriana Kovashka. 2018. Asking Friendly Strangers: Non-Semantic Attribute Transfer. In *AAAI*. 951–958.

[27] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. 2014. Learning and transferring mid-level image representations using convolutional neural networks. In *CVPR*. 1717–1724.

[28] Devi Parikh and Kristen Grauman. 2011. Relative attributes. In *ICCV*. 503–510.

[29] Genevieve Patterson, Chen Xu, Hang Su, and James Hays. 2014. The sun attribute database: Beyond categories for deeper scene understanding. *IJCV* 108, 1-2 (2014), 59–81.

[30] Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. 2018. Efficient Neural Architecture Search via Parameter Sharing. *arXiv:1802.03268* (2018).

[31] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Quoc Le, and Alex Kurakin. 2017. Large-scale evolution of image classifiers. *ICML*.

[32] Bernardino Romera-Paredes and Philip Torr. 2015. An embarrassingly simple approach to zero-shot learning. In *ICML*. 2152–2161.

[33] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556* (2014).

[34] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. 2012. Practical bayesian optimization of machine learning algorithms. In *NIPS*. 2951–2959.

[35] Guang-Lu Sun, Zhi-Qi Cheng, Xiao Wu, and Qiang Peng. 2017. Personalized Clothing Recommendation with User Social Circle and Fashion Style. *MTA* (2017), 1–24.

[36] Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. 2015. Simultaneous deep transfer across domains and tasks. In *ICCV*. 4068–4076.

[37] LiMin Wang, Yu Qiao, and Xiaoou Tang. 2013. Motionlets: Mid-level 3d parts for human motion recognition. In *CVPR*. 2674–2681.

[38] Xiaoyang Wang and Qiang Ji. 2013. A unified probabilistic approach modeling relationships between attributes and objects. In *ICCV*. 2120–2127.

[39] Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*. 5–32.

[40] Hanwang Zhang, Zheng-Jun Zha, Yang Yang, Shuicheng Yan, Yue Gao, and Tat-Seng Chua. 2013. Attribute-augmented semantic hierarchy: towards bridging semantic gap and intention gap in image retrieval. In *ACM MM*. 33–42.

[41] Ziming Zhang and Venkatesh Saligrama. 2015. Zero-shot learning via semantic similarity embedding. In *ICCV*. 4166–4174.

[42] Bo Zhao, Xiao Wu, Zhi-Qi Cheng, Hao Liu, Zequn Jie, and Jiashi Feng. 2018. Multi-View Image Generation From a Single-View. In *ACM MM*.

[43] Barret Zoph and Quoc V Le. 2016. Neural architecture search with reinforcement learning. *ICLR*.

[44] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. 2017. Learning transferable architectures for scalable image recognition. *arXiv:1707.07012* (2017).