Fair and Accurate Decision Making through Group-Aware Learning

Ramtin Hosseini [1] **Li Zhang** [1] **Bhanu Garg** [1] **Pengtao Xie** [1]

Abstract

The integration of machine learning models in various real-world applications is becoming more prevalent to assist humans in their daily decision-making tasks as a result of recent advancements in this field. However, it has been discovered that there is a tradeoff between the accuracy and fairness of these decision-making tasks. In some cases, these AI systems can be unfair by exhibiting bias or discrimination against certain social groups, which can have severe consequences in real life. Inspired by one of the most well-known human learning skills called *grouping*, we address this issue by proposing a novel machine learning (ML) framework where the ML model learns to group a diverse set of problems into distinct subgroups to solve each subgroup using its specific sub-model. Our proposed framework involves three stages of learning, which are formulated as a three-level optimization problem: 1) grouping problems into subgroups, 2) learning group-specific sub-models for problem-solving, and 3) updating group assignments of training examples by minimizing validation loss. These three learning stages are performed end-to-end in a joint manner using gradient descent. To improve fairness and accuracy, we develop an efficient optimization algorithm to solve this three-level optimization problem. To further decrease the risk of overfitting in small datasets using our *LBG* method, we incorporate domain adaptation techniques in the second stage of training. We further apply our method to differentiable neural architecture search (NAS) methods. The *LBG* implementation can be found in the *Skillearn* repository at here.

[1]Department of Electrical and Computer Engineering, UCSD, San Diego, USA. Correspondence to: Ramtin Hosseini <rhossein@eng.ucsd.edu>, Pengtao Xie <p1xie@eng.ucsd.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

1. Introduction

Learning by grouping is an outstanding human learning skill aiming to organize a set of given problems into different subgroups and domains where each subgroup contains similar problems that can be solved independently and efficiently. In this paper, we formulate *Learning by Grouping* (LBG) as an optimization problem and investigate its effectiveness in ML. Our proposed framework contains two types of model: 1) Group Assignment Model (GAM); and 2) Group-Specific Classification Models (GSCM). The GAM model takes a data example as input and predicts the subgroup it belongs to – a K-way classification problem, where K is the number of GSCM models (i.e., experts). For each subgroup k, a GSCM model performs the supervised learning on the target task. We then apply the GAM and the K GSCM models to improve the existing machine learning models' fairness and accuracy. Additionally, we extend our LBG formulation to the neural architecture search to obtain the most suitable task-specific GSCM models. We depict the high-level learning process in Fig 1.

We formulate LBG as a three-stage optimization problem. First, we learn the Group-Assignment Model (GAM); then, we train Group-Specific Classification Models (GSCMs); finally, we apply the GAM and the GSCMs to the validation set to learn the subgroups for subgroup assignment and the learnable architecture. We develop a gradient-based method to solve this three-level optimization problem. In previous related works, mixture-of-expert methods learn the experts – analogous to GSCMs; and the gating network – similar to GAM. The mixture-of-experts (MoE) methods learn the gating network and the experts jointly on the training data, which has a high risk of overfitting the gating network to the training data. We address this problem of overfitting by MoE methods by formulating a three-stage optimization framework that learns the subgroups for the subgroup assignment tasks on the validation set instead of the training examples.

Currently, the majority of state-of-the-art neural network performance is achieved through architectures that are manually designed by humans. However, this process of designing and evaluating neural network architectures by human experts is both time-consuming and may not end with the most suitable task-specific architecture. In recent years,

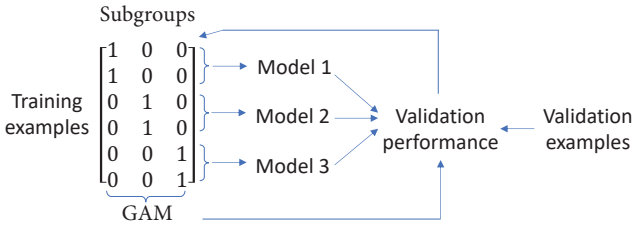

Figure 1. Illustration of Learning by Grouping (LBG) with three subgroups (i.e., $K = 3$). As shown, we update our *Group Assignment Model* (GAM) for the training examples by validating the performance of the validation set, which contrasts LBG method with the existing MoE approaches.

there has been a growing interest in automating this manual process, referred to as neural architecture search (NAS). On the other hand, humans possess powerful learning skills that have been developed through evolution. This study also examines the potential of using a human-based learning technique, known as learning by grouping, in differentiable NAS approaches.

The major contributions of this paper include:

- Drawing inspirations from the human learning technique of *Learning by Grouping* (LBG), we propose a new machine learning framework that utilizes grouping to divide a set of diverse problems into distinct subgroups. The proposed framework groups similar problems together within each subgroup and subsequently develops a group-specific solution for each subgroup.

- We propose a three-level optimization framework to formulate LBG. We provide a solution to solve the optimization problem jointly end-to-end via gradient descent: 1) learning to group problems into different subgroups; 2) learning group-specific sub-models; 3) learning group-assignments of training examples by minimizing the validation loss.

- We also propose *domain adaptive* LBG (DALBG) to mitigate the risk of overfitting within our LBG framework by utilizing domain adaption techniques.

- We extend the above formulation to the challenging neural architecture search (NAS) problem, and we show that LBG/DALBG can be applied to any differentiable NAS approach for further improvements.

- We perform experiments on CelebA, ISIC-18, CIFAR-10, CIFAR-100, and ImageNet datasets to showcase the effectiveness of our proposed method in both fairness and accuracy aspects. Additionally, we apply our proposed LBG to language understanding tasks by conducting experiments on GLUE datasets, which can be found in the Supplements.

2. Related Works

2.1. Mixture of Experts

Lately, a wide variety of works (Shazeer et al., 2017; Zhang et al., 2019; Wang et al., 2020) have proposed applying the mixture-of-experts (MoE) approach, which was initially proposed by (Jacobs et al., 1991), to varied deep learning tasks. Generally, deep learning MoE frameworks consist of expert networks and a gating function, where the gating function assigns each expert a subset of training data. The methods assume a set of latent experts where each expert performs a classification or regression task. A gating function assigns the given data example to an expert. Then this example is classified using the classification model specific to this expert. The MoE has been an active research area aiming to improve the vanilla ML approaches, such as (Shazeer et al., 2017; Zhang et al., 2019; Wang et al., 2020). (Shazeer et al., 2017) introduces a trainable gating function to assign the experts' sparse combinations for the given data. DeepMOE (Wang et al., 2020) proposes a deep convolutional network including a shallow embedding network and a multi-headed sparse gating network, where the multi-headed sparse gating network uses the mixture weights computed by the shallow embedding network to select and re-weight gates in each layer. In MGE-CNN(Zhang et al., 2019), experts are learned with the extra knowledge of their previous experts along with a Kullback-Leibler (KL) divergence constraint to improve the diversity of the experts. Recently, (Riquelme et al., 2021) proposed the Vision Transformer MoE (V-MoE) that can successfully reach state-of-the-art on ImageNet with approximately half of the required resources.

In the existing MoE methods, which are based on single-level optimization, the gating function and expert-specific Classification Models are learned jointly by minimizing the training loss. Hence, there is a high risk of the gating function overfitting the training data, which can lead to unfair and inaccurate decision-making. In our method, we address this issue via learning the group assignments of training examples by minimizing the validation loss instead and developing a multi-stage optimization problem rather than joint training. The results show the efficacy of our method.

2.2. Domain Adaptation

Domain adaptation (DA) is a technique in machine learning that aims to enhance the performance of models trained on one domain, known as the source domain, on a different yet related domain, referred to as the target domain. The objec-

tive is to transfer the knowledge acquired from the source domain to the target domain, where the input features and/or output labels may vary. This approach is particularly valuable in scenarios where the amount of labeled data in the target domain is scarce, but a large amount of labeled data is available in the source domain. Different methods for domain adaptation (Gretton et al., 2009; Gopalan et al., 2011; Pan et al., 2011; Jhuo et al., 2012) can be classified into three main categories: instance-based, feature-based, and adversarial-based approaches. These methods mostly focus on measuring and minimizing the distance between the source and target domains. Some well-established distance measuring approaches include Maximum Mean Discrepancy (MMD) (Long et al., 2015; Gretton et al., 2008), Correlation Alignment (CORAL) (Sun et al., 2017), Kullback-Leibler (KL) divergence (Kullback & Leibler, 1951), and Contrastive Domain Discrepancy (CDD) (Kang et al., 2019).

2.3. Multi-Level Optimization

In the past few years, Bi-Level Optimization (BLO) and Multi-Level Optimization (MLO) (Vicente & Calamai, 1994) techniques have been applied to Meta-Learning (Feurer et al., 2015; Finn et al., 2017), and Automated Machine Learning (AutoML) tasks such as neural architecture search (Cai et al., 2019; Liu et al., 2018b; Xie et al., 2019; Xu et al., 2020; Liang et al., 2019; Hosseini et al., 2021) and hyperparameter optimization (Feurer et al., 2015; Baydin et al., 2017) to learn the meta parameters automatically and reduce the required resources and reliance on humans for designing such methods. Lately, inspired by humans' learning skills (Xie et al., 2020), several existing works (Hosseini & Xie, 2022a; Chitnis et al., 2022; Hosseini & Xie, 2020; Garg et al., 2021; Du et al., 2020; Hosseini & Xie, 2022b; Sheth et al., 2021; Du & Xie, 2020; Zhu et al., 2022) have borrowed these skills from humans and extended them to ML problems in MLO frameworks to study whether these techniques can assist the ML models in learning better.

2.4. Neural Architecture Search

Recently, Neural Architecture Search (NAS) has attracted the researchers' attention to assist them in finding high-performance neural architectures for different deep learning applications. In the early stages, most of the proposed NAS methods were based on reinforcement learning (RL) (Zoph & Le, 2016; Pham et al., 2018; Zoph et al., 2018) and evolutionary learning (Liu et al., 2018a; Real et al., 2019). Reinforcement learning approaches use a policy network to generate architectures by maximizing the accuracy of the validation set, which is used as a reward. In evolutionary learning methods, architectures describe the individuals of a population, and the validation accuracy of the individuals is used as fitness scores. Replacing low fitness scores individuals with higher fitness scores individuals leads to enhanced performance. Reinforcement learning and evolutionary learning approaches are computationally expensive. To solve for the high computational cost by the RL and evolutionary learning-based methods, the research community introduced differentiable search methods (Cai et al., 2019; Liu et al., 2018b; Xie et al., 2019), which are extremely efficient compared to the previous methods since they use the weight-sharing techniques and perform the searching process using gradient descent. Differentiable NAS was first proposed by DARTS (Liu et al., 2018b). Lately, many following works (Chen et al., 2019; Xu et al., 2020; Liang et al., 2019) have worked on enhancing the search results and reducing the computational cost of differentiable NAS even further. For instance, P-DARTS (Chen et al., 2019) increases the depth of architectures progressively during the search. PC-DARTS (Xu et al., 2020) reduces the redundancy by evaluating only a subset of channels in the search process.

3. Methods

Our method consists of a Group-Assignment Model (GAM) and K Group-Specific Classification Models (GSCMs). The GAM model predicts and assigns the training samples to their corresponding GSCM expert model. Then the GSCM models predict the classes of the inputs. Lastly, we apply the GAM and the GSCMs to the validation set and minimize the validation loss to learn the assignments of training samples. The illustration of our proposed method is shown in Fig 2. In Section 3.1, we first begin with defining the three-level optimization framework to formulate LBG (Section 3.1.1), and then we integrate domain adaptation techniques to our proposed LBG to mitigate the risk of overfitting (Section 3.1.2). Afterward, we extend the LBG to the Neural Architecture Search problem in Section 3.1.3. Finally, in Section 3.2, we develop an efficient optimization algorithm to address the three-level optimization problem.

3.1. Three-Level Optimization Framework

Our framework is composed of two types of models: the Group-Assignment Model (GAM) and the Group-Specific Classification Models (GSCM). The GAM model takes a data example as input and assigns it to one of the subgroups, which is a K-way classification problem, where K is the number of GSCM models (i.e., experts). We propose an end-to-end three-stage optimization problem where: First, the Group-Assignment Model (GAM) is learned; then, the Group-Specific Classification Models (GSCMs) are trained; finally, the GAM and the GSCMs are applied to the validation set to determine the group assignments and the learnable architecture. As shown in Fig 1, the Group-Assignment Model (GAM) is updated for training examples by vali-

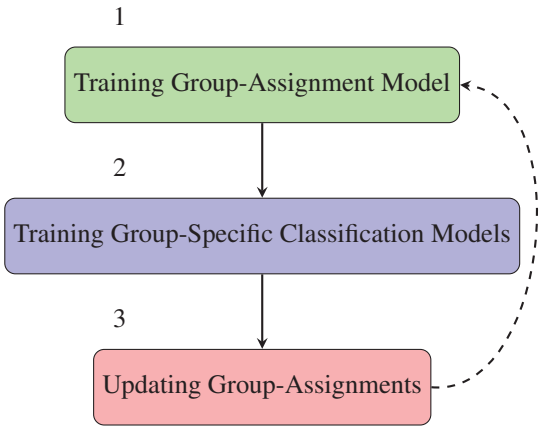

Figure 2. Overview of our proposed three-level optimization framework (Learning by Grouping).

dating the performance on the validation set, which distinguishes LBG from existing MoE approaches. As discussed in Section 3.1.1, the group assignments from GAM are continuous values $C_{nk} \in [0, 1]$. Therefore, to convert these probability distributions to one-hot encoded format (similar to Fig 1) we can compute the top-k and obtain the k-hot encoded matrix, where k is one in this case.

3.1.1. Learning by Grouping (LBG)

We assume there are K latent subgroups. Let C be a matrix denoting the learnable 'ground-truth' grouping of the training samples. The size of C is $N \times K$ where N is the number of training examples - row n represents the grouping of the n-th training example. We relax the values in each row from a one-hot encoding to continuous values in order to perform gradient descent, so that $C_{nk} \in [0, 1]$ denotes the probability that the n-th training example belongs to the k-th latent subgroup. Subgroups C are initialized randomly. The latent subgroup labels for subgroups are permutation-invariant. We then assign the n-th training example to subgroup j_n such that $j_n = \arg\max_e C_{ne}$, and let $G_n = C_{nj_n}$ be the probability of grouping the sample x_n to subgroup j_n. Let the GAM be represented by $f(x_n; T)$ with SoftMax output, which takes a data example x_n as input and predicts which subgroup x_n should be assigned to. T is the weights parameter of this network. The output of $f(x_n; T)$ is a K-dimensional vector, where the k-th element $f_k(x_n; T)$ denotes the probability that x_n should be assigned to the k-th subgroup. The sum of elements in $f(x_n; T)$ is one. Let $\hat{j}_n = \arg\max_e f_e(x_n; T)$, and let $E_n = f_{\hat{j}_n}(x_n; T)$ be the confidence of the GAM in assigning x_n the subgroup $\hat{j}_n$. We then have a GSCM classifier $f(x_n; S_{\hat{j}_n})$ for each latent subgroup $\hat{j}_n \in \{1 \dots K\}$, which predicts the class label for a data example x_n that has been assigned $\hat{j}_n$ as its GSCM by the GAM: $f(x_n; T)$. $S_{\hat{j}_n}$ are the network weights of this GSCM classifier.

Stage I. In the first stage, we optimize the GAM: $f(x; T)$ given C by solving the following 'relaxed' negative log-likelihood optimization problem:

$$T^*(C) = \underset{T}{\text{argmin}} \sum_{n=1}^{N} -G_n(C) \log f_{j_n}(x_n; T) \tag{1}$$

Note that we do not update the 'ground-truth' subgroups C in this stage.

Stage II. In the second stage, we learn the K GSCM models. For each latent subgroup k, there is a GSCM classifier $f(x; S_k)$ with parameters S_k, which predicts the class label for a data example x assigned k as its subgroup by the GAM. Let $D_n = \{x_n\}$ denote the subset of training data examples assigned subgroup k by GAM. We want to learn S_k by minimizing the following loss:

$$\sum_{x_n \in D_n} \ell(f(x_n; S_k), y_n) \tag{2}$$

where y_n is the class label of x_n . $l(\cdot, \cdot)$ is the cross-entropy loss. In addition, we take into account the confidence of the GAM in assigning the training example x_n to its corresponding GSCM $\hat{j}_n$. So we relax the above equation, and summarize the total loss of all GSCM models and objective of this stage as:

$$\{S_k^*(T^*(C))\}_{k=1}^K = \underset{\{S_k\}_{k=1}^K}{\text{argmin}} \sum_{n=1}^{N} E_n(x_n; T^*(C))\, \ell(f(x_n; S_{\hat{j}_n}), y_n) \tag{3}$$

where $S_k^*(T^*(C))$ for $k \in \{1 \dots K\}$ denotes the optimal solution set for the K GSCM classifiers.

Stage III. Given $T^*(C)$ and $S_k^*(T^*(C))$, we apply them to make predictions on the validation examples and update the 'ground-truth' matrix C. The validation loss is:

$$\min_C \sum_{i=1}^{M} \ell\left(E_{\hat{j}_i}(x_i; T^*(C))\, f\left(x_i; S_{\hat{j}_i}^*(T^*(C))\right), y_i\right) \tag{4}$$

where $\hat{j}_i = \arg\max_e f_e(x_i; T^*(C))$ and M is the number of validation examples. y_i is the class label of x_i. We update C by minimizing this validation loss.

Putting these pieces together, we have the following optimization problem:

$$\min_C \sum_{i=1}^{M} \ell\left(E_{\hat{j}_i}(x_i; T^*(C))\, f\left(x_i; S_{\hat{j}_i}^*(T^*(C))\right), y_i\right)$$

$$s.t. \{S_k^*(T^*(C))\}_{k=1}^K = \underset{\{S_k\}_{k=1}^K}{\text{argmin}} \sum_{n=1}^{N} E_n(x_n; T^*(C))\, \ell(f(x_n; S_{\hat{j}_n}), y_n) \tag{5}$$

$$T^*(C) = \underset{T}{\text{argmin}} \sum_{n=1}^{N} -G_n(C) \log f_{j_n}(x_n; T)$$

3.1.2. DOMAIN ADAPTIVE LBG

In our proposed *Learing by Grouping* (LBG) from Section 3.1.1, the N training examples are divided into K subgroups. As a result, each subgroup has approximately N/K training examples. The reduced number of training examples in small datasets can potentially lead to higher risk of overfitting for each subgroup. To address this problem, we propose domain-adaptive LBG (DALBG) where we treat each subgroup as a domain. During the second stage of our framework, when we are training a group-specific classifier for a subgroup k, we perform domain adaptation to adapt examples from other subgroups into subgroup k and use these adapted examples as additional training data for subgroup k. For the sake of simplicity, our proposed framework employs the MMD-based (Long et al., 2015) domain adaptation approach. However, it should be noted that other domain adaptation techniques can also be incorporated within our framework. For a specific subgroup, k, let $\{x_i^k\}_{i=k}^{N_k}$ represent the examples assigned to this subgroup and $\{x_j^{-k}\}_{j=1}^{N-N_k}$ represent the examples not assigned to this subgroup. In order to adapt $\{x_j^{-k}\}_{j=1}^{N-N_k}$ into subgroup k, we minimize the Maximum Mean Discrepancy (MMD) loss as follows:

$$M_k = \left\| \frac{1}{N_k} \sum_{i=1}^{N_k} \phi\left(x_i^k; S_k\right) - \frac{1}{N-N_k} \sum_{j=1}^{N-N_k} \phi\left(x_j^{-k}; S_k\right) \right\|_2^2 \tag{6}$$

where $\phi\left(x_i^k; S_k\right)$ denotes the embedding of x_i^k extracted by S_k. This loss can be relaxed to:

$$M_k = \left\| \frac{1}{N} \sum_{n=1}^{N} f_k\left(x_n; T^*(A)\right) \phi\left(x_n; S_k\right) - \frac{1}{N} \sum_{n=1}^{N} \left(1 - f_k\left(x_n; T^*(A)\right)\right) \phi\left(x_n; S_k\right) \right\|_2^2 \tag{7}$$

Thus, by adding M_k to our second stage Eq. (3) we define the following *domain adaptive* LBG (DALBG) problem:

$$\begin{aligned} &\min_{C} \sum_{i=1}^{M} \ell\left(E_{j_i}\left(x_i; T^*(C)\right) f\left(x_i; S_{j_i}^*\left(T^*(C)\right)\right), y_i\right) \\ &s.t. \{S_k^*(T^*(C))\}_{k=1}^K = \underset{\{S_k\}_{k=1}^K}{\operatorname{argmin}} \sum_{n=1}^{N} E_n\left(x_n; T^*(C)\right) \ell(f(x_n; S_{j_n}), y_n) + \lambda M_k \\ &T^*(C) = \underset{T}{\operatorname{argmin}} \sum_{n=1}^{N} -G_n(C) \log f_{j_n}(x_n; T) \end{aligned} \tag{8}$$

where λ is a tradeoff parameter. Note that our proposed LBG in Eq. (5) method is a special case of DALBG in Eq. (8) with $\lambda = 0$. For the sake of simplicity, we refer to both Learning by Grouping ***with/without*** domain adaptation as (DA)LBG.

3.1.3. NEURAL ARCHITECTURE SEARCH APPLICATION

In this section, we extend the formulation in Eq. (5) to be applicable to neural architecture search. Similar to (Liu et al., 2018b), the k-th GSCM has a differentiable architecture A_k. The search space of A_k is composed of large number of building blocks, where the output of each block is associated with a weight a indicating the importance of the block. After learning, the block whose weight a is among the largest are retained to form the final architecture. To this end, architecture search amounts to optimizing the set of architecture weights $A_k = \{a\}$.

Stage I and **Stage II** have the same procedure as Eq. (1) and Eq. (7). In the second stage, the network weights S_k of the expert model are a function of its architecture A_k. We keep the architecture fixed at this stage, and learn the weights $S_k(A_k)$. However, **Stage III** does not precisely follow Eq. (4). Given $T^*(C)$ and $S_k^*(A_k, T^*(C))$, we apply them to make predictions on the validation examples and update the 'ground-truth' matrix C, as well as the architectures A_k based on the validation loss, where $k \in \{1 \dots K\}$. Hence, we update Eq. (4) as follows:

$$\min_{C, \{A_k\}_{k=1}^K} \sum_{i=1}^{M} \ell\left(E_{j_i}\left(x_i; T^*(C)\right) f\left(x_i; S_{j_i}^*(A_{j_i}, T^*(C))\right), y_i\right) \tag{9}$$

Thus, the overall optimization problem with learnable architecture is as follows:

$$\begin{aligned} &\min_{C, \{A_k\}_{k=1}^K} \sum_{i=1}^{M} \ell\left(E_{j_i}\left(x_i; T^*(C)\right) f\left(x_i; S_{j_i}^*(A_{j_i}, T^*(C))\right), y_i\right) \\ &s.t. \{S_k^*(A_k, T^*(C))\}_{k=1}^K = \underset{\{S_k\}_{k=1}^K}{\operatorname{argmin}} \sum_{n=1}^{N} E_n\left(x_n; T^*(C)\right) \ell(f(x_n; S_{j_n}(A_{j_n}), y_n) + \lambda M_k \\ &T^*(C) = \underset{T}{\operatorname{argmin}} \sum_{n=1}^{N} -G_n(C) \log f_{j_n}(x_n; T) \end{aligned} \tag{10}$$

Our framework is orthogonal to existing differentiable NAS methods, and hence can be applied on top of any like DARTS (Liu et al., 2018b), P-DARTS (Chen et al., 2019), PC-DARTS (Xu et al., 2020), and DARTS− (Chu et al., 2020) among the others.

3.2. Optimization Algorithm

We promote an efficient algorithm to solve the LBG, DALBG, and LBG-NAS problems described in Eq. (5), Eq. (8), and Eq. (10), respectively. We utilize a fairly similar procedure as (Liu et al., 2018b) to calculate the gradient of Eq. (1) w.r.t T and approximately update $T^*(C)$ via one-step gradient descent. Then since DALBG in Eq. (8) is the generalized version of LBG in Eq. (5), we plug the approximation $T^{'}(C)$ into the Eq. (7) to get an O_{S_k}, which denotes the approximated objective of S_k. Similarly to the previous step, we approximate $S_k^*(T^{'}(C))$ using a one-step gradient descent update of S_k based on the gradient of the approximated objective. Note that in LBG-NAS, we approximate $S_k^*(A_k, T^*(C))$, which is also a function of architecture A_k. Finally, we plug the approximations $T^{'}(C)$ and $S_k^{'}(T^{'}(C))$ into the third stage equations to get the third approximate objective denoted by O_C. C can be updated using gradient descent on O_C. In LBG-NAS, we update the architectures $\{A_k\}_{k=1}^K$, as well. Thus, use the same approach to find the approximate objective of the architectures $\{A_k\} : O_{\{A_k\}}$ for each $k \in \{1 \dots K\}$, and we update it using gradient descent. We do these steps until convergence.

4. Experiments

In this section, we investigate the effectiveness of our proposed (DA)LBG framework with both fixed human-designed GSCMs and searchable GSCMs. The differentiable NAS approach consists of architecture search and evaluation stages, where the optimal cell obtained from the search stage is stacked several times into a larger composite network. We then train the resultant composite network from scratch in the evaluation stage. Please refer to the appendix (supplements) for information on adapting our method for language understanding tasks.

4.1. Datasets

Various experiments are conducted on four datasets: ISIC-18, CelebA, CIFAR-10, CIFAR-100, and ImageNet (Deng et al., 2009) for image classification. The CelebA dataset, consisting of 200k images of human faces with 40 features per image (Liu et al., 2015), is used in this study. From the dataset, we select a sample of 10,000 images, with 70% allocated for training, 15% for validation, and 15% for testing. The Skin ISIC 2018 dataset (Codella et al., 2019; Tschandl et al., 2018) consists of a total of 11,720 dermatological images, specifically curated for the purpose of 7-class skin cancer classification. In this research paper, we have identified gender (male and female) as the sensitive attribute that may introduce bias. To mitigate this potential bias, we have performed a partitioning of the dataset into training, validation, and testing sets. The training set comprises 10,015 images, the validation set contains 1,512 images, and the testing set consists of 193 images, collectively representing the entirety of the dataset. The CIFAR-10 dataset contains of 10 distinct classes, while the CIFAR-100 dataset encompasses 100 classes. Each dataset holds 60K images. For each of the datasets, during grouping and architecture search processes, we use 25K images as the training set, 25K images as the validation set, and the rest of the 10K images as the test set. During grouping and architecture evaluations, the combination of the above training and validation set is used as the training set of size 50k images. ImageNet carries 1.2M training images and 50K test images with 1000 classes. Due to extensive amount of images in ImageNet, the architecture search can be pretty costly. Thus, following (Xu et al., 2020), we randomly choose 10%, and 2.5% of the 1.2M images to create a new training set and validation set, respectively, for the architecture search phase. Then, we utilize all the 1.2M images through the evaluation.

4.2. Experimental Settings

We compare the (DA)LBG image classification tasks with fixed architectures to the following MoE baselines: ResNet (He et al., 2016), Swin-T (Liu et al., 2021), T2T-ViT (Yuan et al., 2021), DeepMOE (Wang et al., 2020), and MGE-CNN (Zhang et al., 2019). Next, we compare LBG-NAS on image classification with DARTS-based methods including DARTS (Liu et al., 2018b), P-DARTS (Chen et al., 2019), and PC-DARTS (Xu et al., 2020). To ensure the training costs of our methods with K GSCM models are similar to those of baselines, we reduce the parameter number of each expert to $1/K$ of the parameter number of the baseline models by reducing the number of layers in each GSCM model. In this way, the total size of our methods are comparable to the baselines. In addition, we train each group-specific sub-model only using examples assigned to its corresponding subgroup, rather than using all training examples. So the computation cost is $O(N)$ rather than $O(NK)$, where N is the number of training examples and K is the number of latent subgroups. In each iteration of the algorithm, we use minibatches of training examples to update sub-models, which further reduces computation cost. We utilize the Betty library (Choe et al., 2022) for the implementation of our multilevel optimization tasks.

Table 1. Results on CelebA with the target label of "attractive" and sensitive attribute of "gender".

Methods	Error (%)	DP	DEO	Architecture
ResNet18	17.57	0.5023	0.5683	Manual
LBG-ResNet18 (ours)	<u>17.02</u>	<u>0.2173</u>	**0.0596**	Manual
DALBG-ResNet18 (ours)	**16.84**	**0.2116**	<u>0.0835</u>	Manual
DARTS	16.39	0.4571	0.3606	NAS
LBG-DARTS (ours)	<u>15.91</u>	**0.2149**	**0.0535**	NAS
DALBG-DARTS (ours)	**15.22**	<u>0.2185</u>	<u>0.0891</u>	NAS

Table 2. Results of ISIC when the sensitive attribute is "gender".

Methods	Error(%)	SPD	EOD	AOD
ResNet18	14.3	0.114	0.143	0.170
LBG-ResNet18 (ours)	**12.8**	**0.051**	**0.088**	**0.074**
DARTS	10.2	0.121	0.139	0.154
LBG-DARTS (ours)	**8.4**	**0.048**	**0.065**	**0.069**

Human-Designed GSCMs. For experiments on CIFAR-10/100 and ImageNet datasets, we use ResNet (He et al., 2016), Swin-T (Liu et al., 2021), and T2T-ViT (Yuan et al., 2021) models as our base GSCM models in the conducted experiments. For consistency and a fair comparison, we apply $K = 2$ latent subgroups to our four image classification datasets. To train our models, first we apply our proposed LBG training, where we use half of training images as the training set and the other half as the validation set, for 100 epochs with early stopping technique to obtain the optimal subgroups. Then, we use the obtained subgroups to fine-tune our GSCM models using the standard training settings with SGD optimizer for 200 epochs on the entire training examples. The initial learning rate is set to 0.1 with momentum 0.9 and will be reduced using a cosine decay scheduler with the weight decay of 3e-4. The batch size for

Table 3. Test errors comparison of vanilla (base) models, baselines and LBG on CIFAR-10, CIFAR-100, and ImageNet.

Dataset	Model	Error(%)
CIFAR-10	ResNet56 (vanilla)	6.55
CIFAR-10	DeepMOE-ResNet56	6.03
CIFAR-10	MGE-CNN-ResNet56	5.91
CIFAR-10	LBG-ResNet56 (ours)	5.53
CIFAR-10	DALBG-ResNet56 (ours)	**5.47**
CIFAR-100	ResNet56 (vanilla)	31.46
CIFAR-100	DeepMOE-ResNet56	29.77
CIFAR-100	MGE-CNN-ResNet56	29.82
CIFAR-100	LBG-ResNet56 (ours)	27.96
CIFAR-100	DALBG-ResNet56 (ours)	**27.95**
ImageNet	ResNet18 (vanilla)	30.24
ImageNet	DeepMOE-ResNet18	29.05
ImageNet	MGE-CNN-ResNet18	29.30
ImageNet	LBG-ResNet18 (ours)	28.21
ImageNet	DALBG-ResNet18 (ours)	**28.08**
ImageNet	T2T-ViT-14 (vanilla)	17.16
ImageNet	LBG-T2T-ViT-14 (ours)	15.50
ImageNet	DALBG-T2T-ViT-14 (ours)	**15.47**
ImageNet	Swin-T (vanilla)	18.70
ImageNet	LBG-Swin-T (ours)	16.81
ImageNet	DALBG-Swin-T (ours)	**16.64**

CIFAR-10 and CIFAR-100 is set to 128, while for ImageNet we use the batch size of 1024. The rest of hyperparameter settings follows as (Gururangan et al., 2020). In all DALBG experiments we use $\lambda = 0.1$. In this study, the Adam optimizer has been employed to train all models on the CelebA dataset, utilizing a learning rate of 5e-4, and implementing a batch size of 64. On the other hand, for the ISIC-18 experiments, we have set the learning rate to 1e-3, and the batch size to 32. For the experiments involving CelebA and ISIC-18, we leverage models that have been pretrained on ImageNet. Our models are trained for a range of 30 to 50 epochs, incorporating early stopping techniques to enhance efficiency.

GSCMs with Searchable Architectures. We apply LBG to various DARTS-based approaches: DARTS (Liu et al., 2018b), P-DARTS (Chen et al., 2019), and PC-DARTS (Xu et al., 2020). The search spaces of these methods are the combination of (dilated) separable convolutions with two different sizes of 3×3 and 5×5, max pooling with the size of 3×3, average pooling with the size of 3×3, identity, and zero operations. Each LBG experiment was repeated five times with different random seeds. The mean and standard deviation of classification errors obtained from the experiments are reported.

In the architecture search stage, for CIFAR-10 and CIFAR-100, the architecture of each group-specific classification model contains 5 cells – reduced from 8 cells to 5 cells to match the parameter numbers of our baseline models – and each cell consists of 7 nodes. We use two group-specific sub-models (i.e., two subgroups $K = 2$) in the search process with the initial channels of 16. The search algorithm was based on SGD with a batch size of 64, the initial learning rate of 0.025 (reduced in later epochs using a cosine decay scheduler), epoch number of 50, weight decay of 3e-4, and momentum of 0.9. The rest of hyperparameters mostly follow the original settings in DARTS, P-DARTS, and PC-DARTS. For a fair comparison, in all the DALBG-NAS experiments $\lambda = 0.1$. For ISIC-18 and CelebA experiments, we utilize the same setting as described in the previous part.

During architecture evaluation, each GSCM sub-model is formed by stacking 11 copies (reduced from 20 layers to align with the baselines' sizes) of the corresponding optimally searched cell for CIFAR-10 and CIFAR-100 experiments. The initial channel number is set to 36. We train the networks with a batch size of 96 and 600 epochs on a single Tesla V100 GPU. For evaluation of ImageNet, we use the searched architectures on CIFAR-10 and we stack 8 copies (similarly reduced from 14 layers to match the baselines' sizes) of obtained cells are stacked into each GSCM larger network, which was trained using four Tesla V100 GPUs on the 1.2M training images, with the batch size of 1024 and initial channel number of 48 for 250 epochs. Finally, for the evaluation of architecture in ISIC-18 and CelebA, we follow the same settings as those described for fixed human-designed GSCMs. However, as we don't have models pretrained on ImageNet available, we supplement our training with additional data for both CelebA and ISIC-18.

4.3. Results

First, we evaluate and compare the fairness of our proposed methods with the our baselines on CelebA dataset. In line with the methodology of (Wang et al., 2022), we use "attractive" as the binary class label for prediction, and as bias-sensitive attribute, we consider "gender" (male and female) in relation to the predicted labels. For evaluation we use Demographic Parity (DP) and Difference in Equalized Odds (DEO) metrices similar to (Wang et al., 2022). The results of our experiments, as shown in Table 1, demonstrate that our proposed methods can improve accuracy while simultaneously mitigating unfair decision-making on minority groups. This is achieved through the use of group-specific models, which are trained on individual groups. We can also observe that DALBG improves accuracy more than LBG, but LBG achieves better fairness results on the DEO metric. This could be due to the fact that domain adaptation incorporated in DALBG may perpetuate or even amplify any existing biases present in the source domain, which may

Table 4. Test errors, number of model parameters (in millions), and search costs (GPU days on a Tesla v100) on CIFAR-100 and CIFAR-10. (DA)LBG-DARTS represents (DA)LBG applied to DARTS. Similar meanings hold for other notations in such a format.

Method	CIFAR-100			CIFAR-10		
	Error(%)	Param(M)	Cost	Error(%)	Param(M)	Cost
DARTS (Liu et al., 2018b)	20.58±0.44	3.4	1.5	2.76±0.09	3.3	1.5
LBG-DARTS (ours)	18.02±0.36	3.6	1.7	**2.62±0.08**	3.5	1.6
DALBG-DARTS (ours)	**17.97±0.43**	3.7	2.0	2.64±0.12	3.6	2.0
PC-DARTS (Xu et al., 2020)	17.96±0.15	3.9	0.1	2.57±0.07	3.6	0.1
LBG-PCDARTS (ours)	16.21±0.19	4.1	0.3	2.51±0.11	3.7	0.3
DALBG-PCDARTS (ours)	**16.18±0.21**	4.2	0.4	**2.48±0.15**	3.8	0.4
P-DARTS (Chen et al., 2019)	17.49	3.6	0.3	2.50	3.4	0.3
LBG-PDARTS (ours)	16.46±0.54	3.7	0.6	2.48±0.16	3.5	0.5
DALBG-PDARTS (ours)	**16.39±0.48**	3.9	0.6	**2.47±0.19**	3.7	0.6

not be fully removed if the target domain is significantly different.

Table 2 demonstrates the results of fairness experiments on the ISIC-18 dataset where gender is the sensitive attribute and we use Statistical Parity Difference (SPD), Equal Opportunity Difference (EOD), and Average Odds Difference (AOD) as metrics to evaluate a model's fairness. This table shows that our method not only boosts accuracy performance, but also improves fairness by effectively mitigating bias in both fixed human-designed neural networks and NAS. This performance improvement is attributable to our group-aware approach, which effectively groups similar samples with respect to unprotected sensitive attributes. This proves the advantage of our methods in addressing imbalanced attributes in the data.

Furthermore, in Table 3, we compare our proposed method with ResNet, Vision Transformers (Swin-T and T2T-ViT) ,and our MoE baselines (i.e., MGE-CNN and DeepMOE). The results in this table verify that our proposed method performs better than the baselines on all CIFAR-10, CIFAR-100, and ImageNet datasets considerably. This empirically verifies our claim that (DA)LBG reduces the overfitting risk found in MoE methods since the group assignments are learned by minimizing the validation loss during a multi-stage optimization.

Table 4 shows the comparison of our proposed methods and the existing works, which includes the classification errors with error bars, the number of model parameters, and search costs on CIFAR-10 and CIFAR-100 test sets. By comparing different methods, we make the following observation. Applying (DA)LBG to different NAS methods, including DARTS, P-DARTS, and PC-DARTS, the classification errors of these methods are greatly reduced. For instance, the original error of DARTS on CIFAR-100 is 20.58%; when DALBG is applied, this error is significantly reduced to 17.97%. As another example, after applying LBG to PC-DARTS and P-DARTS, the errors of CIFAR-100 experiments are decreased from 17.96% to 16.21% and 17.49% to 16.46%, respectively. Similarly for CIFAR-10, utilizing (DA)LBG in DARTS-based methods manages to reduce the errors and overfittings. These results strongly indicate the broad effectiveness of our framework in searching better neural architectures.

Table 5. Results of ImageNet with gradient-based NAS methods. Notations are the same as those in Table 4.

Method	Top-1 Error (%)	Top-5 Error (%)	Param (M)
DARTS-CIFAR10 (Liu et al., 2018b)	26.7	8.7	4.7
DALBG-DARTS-CIFAR10 (ours)	**24.9**	8.1	4.9
P-DARTS (CIFAR10) (Chen et al., 2019)	24.4	7.4	4.9
DALBG-PDARTS-CIFAR10 (ours)	**23.9**	6.9	5.0
PC-DARTS-CIFAR10 (Xu et al., 2020)	24.8	7.3	5.3
DALBG-PCDARTS-CIFAR10 (ours)	**23.1**	6.3	5.7

In Table 5, we compare different methods on ImageNet, in terms of top-1 and top-5 errors on the test set and number of model parameters, where the search costs are the same as the ones reported in Table 4. In these experiments, the architectures are searched on CIFAR-10 and evaluated on ImageNet similar to original DARTS (Liu et al., 2018b). DALBG-DARTS-CIFAR10 denotes that DALBG is applied to DARTS and performs search on CIFAR-10. Similar meanings hold for other notations in such a format. The observations made from these results are consistent with those made from Table 4. The architectures searched using our methods are consistently better than those searched by corresponding baselines. For example, DALBG-DARTS-CIFAR10 achieves 1.8% lower top-1 error than DARTS-CIFAR10. To the best of our knowledge, DALBG-PCDARTS-CIFAR10 is the new SOTA on mobile setting of Imagenet.

4.4. Ablation Studies

In this section, we conduct ablation studies to analyze the impact of individual components in our proposed frame-

works.

Ablation on tradeoff parameter λ: We study the effectiveness of tradeoff parameter λ in Eq. (8) on accuracy and fairness. We apply DALBG on CelebA dataset with two searchable GSCMs (i.e., $K = 2$) with the same setting as described in Section 4.2. In Table 6, we illustrate how the accuracy and fairness of DALBG on the test sets of CelebA are affected by increasing the tradeoff parameter λ. It can be observed that increasing λ from 0 to 0.1 leads to a decrease in fairness but an increase in accuracy, as a result of the MMD loss feedback. However, continuing to increase λ leads to a drop in accuracy as well. This is because placing too much emphasis on domain shift can result in less focus on in-domain performance ability.

Table 6. Ablation results on tradeoff parameter λ.

Methods	Error (%)	DEO
LBG-DARTS with $\lambda = 0$	15.91	**0.0535**
DALBG-DARTS with $\lambda = 0.01$	15.73	0.0754
DALBG-DARTS with $\lambda = 0.1$	**15.22**	0.0891
DALBG-DARTS with $\lambda = 1$	15.38	0.0917

Ablation on number of subgroups: Next, we examine how different numbers of GSCM models with different number of subgroups $K \in \{1, 2, 3, 4\}$ in Eq. (10) impact both accuracy and fairness performances. We apply (DA)LBG to DARTS. Table 7 indicates that for CelebA larger number of subgroups can decrease the classification error and improve the fairness. However, in our experiments number of subgroups $K = 3$ and $K = 4$ seem to achieve on par results, while $K = 3$ is more computationally efficient. The improved performance with a larger number of subgroups can be due to the fact that, in real life, many unprotected attributes may not be considered, but their combinations can still be used as proxies and affect decision-making processes. Thus, depending on the data and task, we can choose the most suitable number of subgroups, which can be different in various scenarios. Also, additional experiments and comparisons of (DA)LBG with bagging-based model ensemble can be found in the Supplements.

Table 7. Ablation results on number of subgroups.

Methods	Error (%)	DEO
LBG-ResNet18 with $K = 1$	17.59	0.5427
LBG-ResNet18 with $K = 2$	17.02	0.0596
LBG-ResNet18 with $K = 3$	16.88	0.0541
LBG-ResNet18 with $K = 4$	**16.85**	**0.0533**

Ablation on different domain adaptation techniques: In this study, we aim to investigate the efficacy of different distance measuring approaches, namely Maximum Mean Discrepancy (MMD), Correlation Alignment (CORAL), Kullback-Leibler (KL) divergence, and Contrastive Domain Discrepancy (CDD), by incorporating them into our framework. We conduct our experiments on DARTS with a similar experimental setup to Table 1. The results, presented in Table 8, indicate that MMD loss is the most effective approach in achieving both high accuracy and fairness compared to the other three methods. The superior performance of MMD in our framework can be attributed to its non-parametric nature and ability to capture non-linear relationships due to its kernel-based approach. In contrast, KL divergence relies on the assumption that both distributions are well-defined probability distributions and, along with CDD, may struggle to capture non-linear relationships in the data. Furthermore, while CORAL aligns the second-order statistics (i.e., covariance matrices) of the feature distributions, MMD maps the data into a Reproducing Kernel Hilbert Space (RKHS) using kernel functions. This capability enables MMD to capture more intricate relationships between data points, potentially resulting in improved performance within our framework. However, it is worth noting that the effectiveness of a domain adaptation technique may vary depending on the specific task and the degree of domain shift between the source and target domains. Thus, the choice of an appropriate technique should be based on the unique characteristics of the data and the task at hand.

Table 8. Ablation results on different domain adaptation techniques.

Methods	Error (%)	DEO
DALBG-DARTS-MMD	**15.22**	**0.0891**
DALBG-DARTS-CORAL	15.71	0.1137
DALBG-DARTS-KL	15.36	0.0945
DALBG-DARTS-CDD	15.80	0.1142

5. Conclusions and Discussion

In this paper, we propose a novel MLO approach, called *Learning by Grouping* (LBG), drawing from humans' grouping-driven methodology of solving problems. Our approach learns to group a diverse set of problems into distinct subgroups where problems in the same subgroups are similar; a group-specific solution is developed to solve problems in the same subgroups. We formulate our LBG as a multi-level optimization problem which is solved end-to-end. An efficient gradient-based optimization algorithm is developed to solve the LBG problem. We further incorporate domain adaptation in our framework to reduce the risk of overfitting. In our experiments on various datasets, we demonstrate that the proposed framework not only helps to mitigate overfitting and improve fairness, but also consistently outperforms baseline methods. The main limitation of LBG is that it cannot be applied to non-differentiable NAS approaches up to a point. In our future works, we will extend learning by grouping to reinforcement learning and evolutionary algorithms.

References

Baydin, A. G., Cornish, R., Rubio, D. M., Schmidt, M., and Wood, F. Online learning rate adaptation with hypergradient descent. *arXiv preprint arXiv:1703.04782*, 2017. 3

Cai, H., Zhu, L., and Han, S. Proxylessnas: Direct neural architecture search on target task and hardware. In *ICLR*, 2019. 3

Chen, X., Xie, L., Wu, J., and Tian, Q. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *ICCV*, 2019. 3, 5, 6, 7, 8

Chitnis, S., Hosseini, R., and Xie, P. Brain tumor classification based on neural architecture search. *Scientific Reports*, 12(1):19206, 2022. 3

Choe, S. K., Neiswanger, W., Xie, P., and Xing, E. Betty: An automatic differentiation library for multilevel optimization. *arXiv preprint arXiv:2207.02849*, 2022. 6

Chu, X., Wang, X., Zhang, B., Lu, S., Wei, X., and Yan, J. DARTS-: robustly stepping out of performance collapse without indicators. *CoRR*, abs/2009.01027, 2020. 5

Codella, N., Rotemberg, V., Tschandl, P., Celebi, M. E., Dusza, S., Gutman, D., Helba, B., Kalloo, A., Liopyris, K., Marchetti, M., et al. Skin lesion analysis toward melanoma detection 2018: A challenge hosted by the international skin imaging collaboration (isic). *arXiv preprint arXiv:1902.03368*, 2019. 6

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009. 6

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://www.aclweb.org/anthology/N19-1423. 15

Du, X. and Xie, P. Small-group learning, with application to neural architecture search. *arXiv preprint arXiv:2012.12502*, 2020. 3

Du, X., Zhang, H., and Xie, P. Learning by passing tests, with application to neural architecture search. *arXiv preprint arXiv:2011.15102*, 2020. 3

Feurer, M., Springenberg, J. T., and Hutter, F. Initializing bayesian hyperparameter optimization via meta-learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pp. 1128–1135. AAAI Press, 2015. ISBN 0262511290. 3

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pp. 1126–1135. PMLR, 2017. 3

Garg, B., Zhang, L. L., Sridhara, P., Hosseini, R., Xing, E., and Xie, P. Learning from mistakes - a framework for neural architecture search. In *AAAI Conference on Artificial Intelligence*, 2021. 3

Gopalan, R., Li, R., and Chellappa, R. Domain adaptation for object recognition: An unsupervised approach. In *2011 International Conference on Computer Vision*, pp. 999–1006, 2011. doi: 10.1109/ICCV.2011.6126344. 3

Gretton, A., Borgwardt, K., Rasch, M. J., Scholkopf, B., and Smola, A. J. A kernel method for the two-sample problem. *arXiv preprint arXiv:0805.2368*, 2008. 3

Gretton, A., Smola, A., Huang, J., Schmittfull, M., Borgwardt, K. M., Schölkopf, B., Candela, Q., Sugiyama, M., Schwaighofer, A., and Lawrence, N. D. Covariate shift by kernel mean matching. In *NIPS 2009*, 2009. 3

Gururangan, S., Marasović, A., Swayamdipta, S., Lo, K., Beltagy, I., Downey, D., and Smith, N. A. Don't stop pretraining: Adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964*, 2020. 7, 15

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016. 6

Hosseini, R. and Xie, P. Learning by self-explanation, with application to neural architecture search. *arXiv preprint arXiv:2012.12899*, 2020. 3

Hosseini, R. and Xie, P. Image understanding by captioning with differentiable architecture search. In *Proceedings of the 30th ACM International Conference on Multimedia*, MM '22, pp. 4665–4673, New York, NY, USA, 2022a. Association for Computing Machinery. ISBN 9781450392037. doi: 10.1145/3503161.3548150. URL https://doi.org/10.1145/3503161.3548150. 3

Hosseini, R. and Xie, P. Saliency-aware neural architecture search. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022b. URL https://openreview.net/forum?id=Ho6oWAslz5L. 3

Hosseini, R., Yang, X., and Xie, P. Dsrna: Differentiable search of robust neural architectures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6196–6205, 2021. 3

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. Adaptive mixtures of local experts. *Neural Computation*, 3:79–87, 1991. 2

Jhuo, I.-H., Liu, D., Lee, D. T., and Chang, S.-F. Robust visual domain adaptation with low-rank reconstruction. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2168–2175, 2012. doi: 10.1109/CVPR.2012.6247924. 3

Kang, G., Jiang, L., Yang, Y., and Hauptmann, A. G. Contrastive adaptation network for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4893–4902, 2019. 3

Kullback, S. and Leibler, R. A. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951. ISSN 00034851. URL http://www.jstor.org/stable/2236703. 3

Liang, H., Zhang, S., Sun, J., He, X., Huang, W., Zhuang, K., and Li, Z. Darts+: Improved differentiable architecture search with early stopping. *arXiv preprint arXiv:1909.06035*, 2019. 3

Liu, H., Simonyan, K., Vinyals, O., Fernando, C., and Kavukcuoglu, K. Hierarchical representations for efficient architecture search. In *ICLR*, 2018a. 3

Liu, H., Simonyan, K., and Yang, Y. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018b. 3, 5, 6, 7, 8

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019. 15

Liu, Z., Luo, P., Wang, X., and Tang, X. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pp. 3730–3738, 2015. 6

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10012–10022, 2021. 6

Long, M., Cao, Y., Wang, J., and Jordan, M. Learning transferable features with deep adaptation networks. In *International conference on machine learning*, pp. 97–105. PMLR, 2015. 3, 5

Pan, S. J., Tsang, I. W., Kwok, J. T., and Yang, Q. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210, 2011. doi: 10.1109/TNN.2010.2091281. 3

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. In *NIPS-W*, 2017. 15

Pham, H., Guan, M. Y., Zoph, B., Le, Q. V., and Dean, J. Efficient neural architecture search via parameter sharing. In *ICML*, 2018. 3

Real, E., Aggarwal, A., Huang, Y., and Le, Q. V. Regularized Evolution for Image Classifier Architecture Search. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):4780–4789, July 2019. ISSN 2374-3468. doi: 10.1609/aaai.v33i01.33014780. URL https://ojs.aaai.org/index.php/AAAI/article/view/4405. Number: 01. 3

Riquelme, C., Puigcerver, J., Mustafa, B., Neumann, M., Jenatton, R., Susano Pinto, A., Keysers, D., and Houlsby, N. Scaling vision with sparse mixture of experts. *Advances in Neural Information Processing Systems*, 34, 2021. 2

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., and Dean, J. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017. 2

Sheth, P., Jiang, Y., and Xie, P. Learning by teaching, with application to neural architecture search. *arXiv preprint arXiv:2103.07009*, 2021. 3

Sun, B., Feng, J., and Saenko, K. Correlation alignment for unsupervised domain adaptation. *Domain adaptation in computer vision applications*, pp. 153–171, 2017. 3

Tschandl, P., Rosendahl, C., and Kittler, H. The ham10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. *Scientific data*, 5(1):1–9, 2018. 6

Vicente, L. N. and Calamai, P. H. Bilevel and multilevel programming: A bibliography review, 1994. 3

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5446. URL https://www.aclweb.org/anthology/W18-5446. 15

Wang, X., Yu, F., Dunlap, L., Ma, Y.-A., Wang, R., Mirhoseini, A., Darrell, T., and Gonzalez, J. E. Deep mixture of experts via shallow embedding. In *Uncertainty in Artificial Intelligence*, pp. 552–562. PMLR, 2020. 2, 6

Wang, Z., Dong, X., Xue, H., Zhang, Z., Chiu, W., Wei, T., and Ren, K. Fairness-aware adversarial perturbation towards bias mitigation for deployed deep models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10379–10388, 2022. 7

Xie, P., Du, X., and Ban, H. Skillearn: Machine learning inspired by humans' learning skills. *arXiv preprint arXiv:2012.04863*, 2020. 3

Xie, S., Zheng, H., Liu, C., and Lin, L. SNAS: stochastic neural architecture search. In *ICLR*, 2019. 3

Xu, Y., Xie, L., Zhang, X., Chen, X., Qi, G., Tian, Q., and Xiong, H. PC-DARTS: partial channel connections for memory-efficient architecture search. In *ICLR*, 2020. 3, 5, 6, 7, 8

Yuan, L., Chen, Y., Wang, T., Yu, W., Shi, Y., Jiang, Z.-H., Tay, F. E., Feng, J., and Yan, S. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 558–567, 2021. 6

Zhang, L., Huang, S., Liu, W., and Tao, D. Learning a mixture of granularity-specific experts for fine-grained categorization. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pp. 8330–8339. IEEE, 2019. doi: 10.1109/ICCV.2019.00842. URL https://doi.org/10.1109/ICCV.2019.00842. 2, 6

Zhu, W., Wang, X., and Xie, P. Self-directed machine learning. *arXiv preprint arXiv:2201.01289*, 2022. 3

Zoph, B. and Le, Q. V. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016. 3

Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. V. Learning transferable architectures for scalable image recognition. In *CVPR*, 2018. 3

A. Optimization Algorithm

We develop an efficient optimization algorithm to solve our proposed LBG problem. Notations are given in Table 9. We define group j_n such that $j_n = \arg\max_k C_{nk}$, and let $G_n = C_{nj_n}$ be the ground truth assignments of the sample x_n to group j_n. Let $\hat{j}_n = \arg\max_k f_k(x_n; T)$, and let $E_n = f_{\hat{j}_n}(x_n; T)$ be the confidence of the GAM in assigning x_n to the group $\hat{j}_n$.

Table 9. Notations used in LBG

Notation	Meaning
A_k	Architecture of the Group-Specific Classification Model (GSCM) of group k (Only in NAS applications)
S_k	Network weights of the Group-Specific Classification Models $f(x; S_k)$ of group k
T	Network weights of the Group Assignment Model (GAM) $f(x; T)$
C	Learnable 'ground-truth' categorization matrix
D_n	Training data
D_i	Validation data

We approximate $T^*(C)$ using one step gradient descent w.r.t $\sum_{n=1}^{N} -G_n(C) \log f_{j_n}(x_n; T)$:

$$T^*(C) \approx T' = T - \eta_t \nabla_T \sum_{n=1}^{N} -G_n(C) \log f_{j_n}(x_n; T) \tag{11}$$

Then we plug T' into $\sum_{n=1}^{N} E_n\left(x_n; T^*(C)\right) \ell(f(x_n; S_{\hat{j}_n}(A_{\hat{j}_n}), y_n)$ and get an approximated objective. And we approximate $S^*_{\hat{j}_n}\left(T^*(C)\right)$ using one step gradient descent w.r.t the approximated objective:

$$S^*_{\hat{j}_n}\left(T^*(C)\right) \approx S'_{\hat{j}_n} = S_{\hat{j}_n} - \eta_s \nabla_{S_{\hat{j}_n}} \sum_{n=1}^{N} E_n\left(x_n; T'(C)\right) \ell(f(x_n; S_{\hat{j}_n}(A_{\hat{j}_n})), y_n) \tag{12}$$

Finally, we plug T' and $S'_{\hat{j}_n}$ into $\sum_{i=1}^{M} \ell\left(E_{\hat{j}_i}\left(x_i; T^*(C)\right) f\left(x_i; S^*_{\hat{j}_i}(A_{\hat{j}_i}, T^*(C))\right), y_i\right)$ and get an approximated objective. Then we update A by gradient descent:

$$C \leftarrow C - \eta_c \nabla_C \sum_{i=1}^{M} \ell\left(E_{\hat{j}_i}\left(x_i; T'(C)\right) f\left(x_i; S'_{\hat{j}_i}(A_{\hat{j}_i}, T'(C))\right), y_i\right), \tag{13}$$

where by applying chain rule it yields:

$$\begin{aligned} &\nabla_C \sum_{i=1}^{M} \ell\left(E_{\hat{j}_i}\left(x_i; T^*(C)\right) f\left(x_i; S^*_{\hat{j}_i}(A_{\hat{j}_i}, T^*(C))\right), y_i\right) = \\ &\frac{\partial T'}{\partial C} \sum_{i=1}^{M} \frac{\partial S'_{\hat{j}_i}}{\partial T'} \nabla_{S'_{\hat{j}_i}} \ell\left(E_{\hat{j}_i}\left(x_i; T'(C)\right) f\left(x_i; S'_{\hat{j}_i}(A_{\hat{j}_i}, T'(C))\right), y_i\right) + \\ &\frac{\partial T'}{\partial C} \nabla_{T'} \sum_{i=1}^{M} \ell\left(E_{\hat{j}_i}\left(x_i; T'(C)\right) f\left(x_i; S'_{\hat{j}_i}(A_{\hat{j}_i}, T'(C))\right), y_i\right) \end{aligned} \tag{14}$$

and $\frac{\partial T'}{\partial C}$ and $\frac{\partial S'_{\hat{j}_i}}{\partial T'}$ are computed as follows:

$$\frac{\partial T'}{\partial C} = -\eta_t \nabla^2_{C,T} \sum_{n=1}^{N} -G_n(C) \log f_{j_n}(x_n; T) \tag{15}$$

$$\frac{\partial S'_{\hat{j}_i}}{\partial T'} = -\eta_s \nabla^2_{T', S_{\hat{j}_i}} \sum_{n=1}^{N} E_n\left(x_n; T'(C)\right) \ell(f(x_n; S_{\hat{j}_n}(A_{\hat{j}_n})), y_n) \tag{16}$$

For the NAS applications we also update architectures $A_{\hat{j}_i}$, where $\hat{j}_i \in K$:

$$A \leftarrow A - \eta_a \nabla_A \sum_{i=1}^{M} \ell\left(E_{\hat{j}_i}\left(x_i; T'(C)\right) f\left(x_i; S'_{\hat{j}_i}(A_{\hat{j}_i}, T'(C))\right), y_i\right), \tag{17}$$

similar to group updating in Eq. 14, we apply chain rule as follows:

$$\nabla_A \sum_{i=1}^{M} \ell\left(E_{\hat{j}_i}\left(x_i; T^*(C)\right) f\left(x_i; S^*_{\hat{j}_i}(A_{\hat{j}_i}, T^*(C))\right), y_i\right) = \tag{18}$$
$$\sum_{i=1}^{M} \frac{\partial S'_{\hat{j}_i}}{\partial A_{\hat{j}_i}} \nabla_{S'_{\hat{j}_i}} \ell\left(E_{\hat{j}_i}\left(x_i; T'(C)\right) f\left(x_i; S'_{\hat{j}_i}(A_{\hat{j}_i}, T'(C))\right), y_i\right)$$

and $\frac{\partial S'_{\hat{j}_i}}{\partial A_{\hat{j}_i}}$ can be computed using the following equation:

$$\frac{\partial S'_{\hat{j}_i}}{\partial A_{\hat{j}_i}} = -\eta_s \nabla^2_{A_{\hat{j}_i}, S_{\hat{j}_i}} \sum_{n=1}^{N} E_n\left(x_n; T'(C)\right) \ell(f(x_n; S_{\hat{j}_n}(A_{\hat{j}_n})), y_n) \tag{19}$$

This algorithm is summarized in Algorithm 1.

Algorithm 1 Optimization algorithm for Learning by Grouping

while not converged **do**
  1. Update the group assignment model's weights T
  using Eq. 11.
  2. Update the group-specific classification models'
  weights $\{S_k\}_{k=1}^K$ using Eq. 12.
  if NAS application **then**
    3. Update the group-assignment matrix C and
    the group-specific classification models'
    architectures $\{A_k\}_{k=1}^K$ using Eq. 13 and Eq. 17.
  else
    3. Only update the group-assignment matrix C
    using Eq. 13.
  end if
end while=0

B. Additional Experiments

B.1. Comparison with Bagging-based Model Ensemble

In this section we compare our proposed method (LBG) with bagging-based model ensemble, which uses three models (the same as our method). Table 10 demonstrates the results. Our method works better than model ensemble because it uses a divide-and-conquer strategy. It divides data examples into groups where examples in the same group are similar; then for each group, an expert model is learned. Divide-and-conquer makes model training easier, because it is easier to train a highly-performant model for a group of similar examples than for a mixture of dissimilar examples from different groups. In ensemble learning, each model is trained on a mixture of dissimilar examples from different groups, which is a harder problem to solve. Additionally, in our method, the expert for each group can capture the unique data patterns in that group. Capturing group-specific data patterns can help to make more accurate predictions. In contrast, each model in ensemble learning is trained on all examples from different groups, which does not take group-specific data patterns into account.

B.2. Language Tasks

In this section, we apply LBG with fixed human-designed architectures to language understanding tasks.

Table 10. Comparison of our work with existing bagging-based model ensemble on CIFAR-100.

Methods	Test error (%)
Ensemble+DARTS-2nd	19.66±0.34
LBG-DARTS-2nd (ours)	**18.02**±0.36
Ensemble+P-DARTS	17.32±0.27
LBG-PDARTS (ours)	**16.46**±0.54

B.2.1. DATASETS

We conducted experiments on the various tasks of the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2018). GLUE contains nine tasks, which are two single-sentence tasks (CoLA and SST-2), three similarity and paraphrase tasks (MRPC, STS-B, and QQP), and four inference tasks (MNLI, QNLI, RTE, WNLI). We test the performance of LBG in language understanding by submitting our inference results to the GLUE evaluation server. GLUE offers training and development data splits, that are used as training and validation data. For the test dataset, and GLUE organisers provide a submission server that reports the performance on the private held out test dataset.

Table 11. Comparison of BERT-based and RoBERTa-based experiments on GLUE sets. LBG-BERT and LBG-RoBERTa results on the set are the medians of 5 runs.

Corpus	BERT	LBG-BERT	RoBERTa	LBG-RoBERTa
CoLA (Matthews Corr.)	60.5	**62.8**	68.0	**69.5**
SST-2 (Accuracy)	94.9	**96.5**	96.4	**96.8**
MRPC (Accuracy/F1)	85.4/89.3	**86.2/89.5**	**90.9**/92.3	90.2/**92.4**
STS-B (Pearson/Spearman Corr.)	87.6/86.5	**88.4/87.9**	92.4/92.0	**92.5/92.3**
QQP (Accuracy/F1)	89.3/72.1	**89.6/72.3**	92.2/-	**92.6/77.0**
MNLI (Matched/Mismatched Accuracy)	**86.7/85.9**	86.5/**85.9**	90.2/90.2	**91.1/91.1**
QNLI (Accuracy)	92.7	**93.5**	94.7	**94.9**
RTE (Accuracy)	70.1	**72.4**	86.6	**86.7**
WNLI (Accuracy)	65.1	**66.3**	**91.3**	86.3

B.2.2. EXPERIMENTAL SETTINGS

We examine our proposed method by conducting varied experiments on several different tasks and datasets. We compare LBG on language understanding tasks with fixed architectures using BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019). BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019) initialize the Transformer encoder with pre-trained BERT and RoBERTa, respectively, with the intentions of masked language modeling and next sentence prediction. Then, they utilize the pre-trained encoder and a classification head to build a text classification model. This text classification model latter will be fine-tuned on a target classification task.

To examine our method in language understanding, we employ BERT and RoBERTa as the group-specific sub-models with $K = 4$ latent subgroups on the GLUE tasks. LBG-BERT and LBG-RoBERTa are optimized using Adam optimizer (Paszke et al., 2017). The maximum length of text was set to 512 tokens. Our hyperparameter settings for BERT and RoBERTa experiments are the same as in (Gururangan et al., 2020). Each GLUE task has a different batch size, learning rate, and number of epochs, where they are within the batch sizes $\in \{16, 32\}$, learning rates $\in \{1e^{-5}, 2e^{-5}, 3e^{-5}, 4e^{-5}\}$, and number of epochs $\in \{3, 4, 5, 6, 10\}$.

B.2.3. RESULTS

Table 11 demonstrates the comparison of our methods with BERT and RoBERTa methods on nine different GLUE tasks. It is shown in this table that LBG can efficiently enhance the performance of existing base models in various language understanding tasks. In most of the tasks, LBG-BERT and LBG-RoBERTa outperform BERT and RoBERTa, respectively. In MNLI and MRPC, the results of our methods are on par with the baselines, while RoBERTa achieves a slightly better

result than our methods on the WNLI task.