Multi-path Neural Networks for On-device Multi-domain Visual Classification

Qifei Wang *[1], Junjie Ke *[1], Joshua Greaves[2], Grace Chu[1], Gabriel Bender[2], Luciano Sbaiz[1], Alec Go[1], Andrew Howard[1], Ming-Hsuan Yang[1], Jeff Gilbert[1], Peyman Milanfar[1], and Feng Yang[1]

[1]Google Research
[2]Google Brain
{*qfwang, junjiek, joshgreaves, cxy, gbender, sbaiz, ago, howarda, minghsuan, jegilbert, milanfar, fengyang*}*@google.com*

Abstract

Learning multiple domains/tasks with a single model is important for improving data efficiency and lowering inference cost for numerous vision tasks, especially on resource-constrained mobile devices. However, hand-crafting a multi-domain/task model can be both tedious and challenging. This paper proposes a novel approach to automatically learn a multi-path network for multi-domain visual classification on mobile devices. The proposed multi-path network is learned from neural architecture search by applying one reinforcement learning controller for each domain to select the best path in the super-network created from a MobileNetV3-like search space. An adaptive balanced domain prioritization algorithm is proposed to balance optimizing the joint model on multiple domains simultaneously. The determined multi-path model selectively shares parameters across domains in shared nodes while keeping domain-specific parameters within non-shared nodes in individual domain paths. This approach effectively reduces the total number of parameters and FLOPS, encouraging positive knowledge transfer while mitigating negative interference across domains. Extensive evaluations on the Visual Decathlon dataset demonstrate that the proposed multi-path model achieves state-of-the-art performance in terms of accuracy, model size, and FLOPS against other approaches using MobileNetV3-like architectures. Furthermore, the proposed method improves average accuracy over learning single-domain models individually, and reduces the total number of parameters and FLOPS by 78% *and* 32% *respectively, compared to the approach that simply bundles single-domain models for multi-domain learning.*

*equal contribution

1. Introduction

Numerous methods based on deep learning have made significant advances in a wide range of vision tasks, including image classification [13], object detection [6], and segmentation [10], to name a few. However, most deep neural networks (DNNs) are developed for a single task with data coming from the same domain. There is growing interest in training a joint model to tackle multiple domains and tasks, i.e., multi-domain learning (MDL) as well as multi-task learning (MTL).

MDL and MTL are two overlapping but distinct problems. MTL [32] focuses on performing multiple related tasks, e.g., segmentation and depth estimation on a given data sample. On the other hand, MDL aims to build a joint model to perform the same task, e.g., classification, across multiple visual domains, such as Internet images, characters, glyph, animal, sketches, etc. Building a joint MDL model is especially important on resource-constrained mobile devices, where deploying a separate model for each domain can introduce high latency, large memory footprint and power consumption. Although numerous mobile friendly models (e.g., MobileNetV3 [11], ShuffleNet [35], MNAS [30], and ENAS [21]) have been proposed for single domain learning, much less attention has been paid to develop efficient MDL systems for mobile devices.

One straightforward MDL approach is to train a universal feature extractor for all domains. However, its performance may degrade significantly due to the negative transfer between different domains. Existing MDL approaches [23, 24] start from a pre-trained model and fine-tune with added domain specific modules. Each domain can therefore selectively use the features from the fixed main backbone network. Although this approach naturally reduces the *negative transfer* effect between domains, it does not encourage *positive transfer* between domains due to

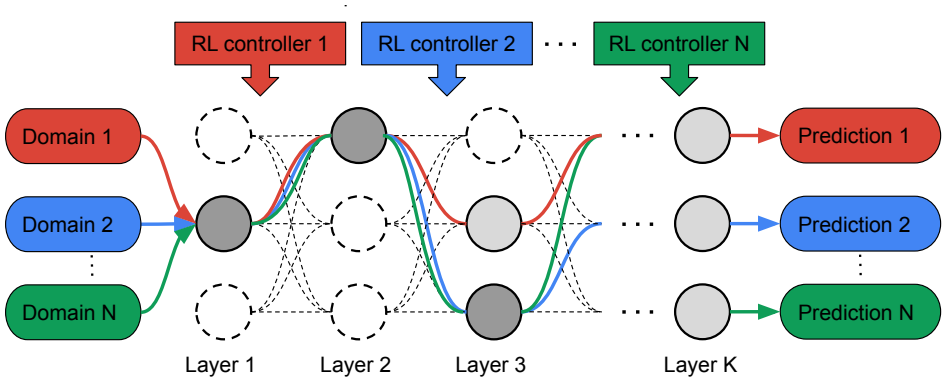

Figure 1: Multi-path network for multi-domain visual classification. Different color paths show model architectures for different domains. For example, the red path represents the model architecture formed for domain 1. Paths that run through the same node end up sharing the weights of the shared node (dark grey).

the frozen backbone network. Manually designing domain-adaptive modules can also be challenging and tedious since it is difficult to transfer the manual modules onto different architectures to achieve both high classification accuracy across domains and low costs in terms of computation and memory consumption.

Another challenging factor for learning a joint MDL model is domain prioritization during training. Different domains usually have diverse degrees of difficulty which may not be constant during the training process. Although various optimization-based approaches [12, 5, 16, 8, 27] have been proposed to deal with this issue, jointly balancing domain difficulties to achieve optimal performance across multiple domains remains a difficult problem.

In this paper, we show that a good MDL model for mobile devices should: 1) achieve high accuracy while keeping number of parameters and FLOPS low; 2) learn to enhance positive knowledge sharing while mitigating negative transfer among domains; and 3) effectively optimize the joint model across domains and be adaptive to domain difficulties. As such, we propose a neural architecture search (NAS) method for designing effective MDL models, and train it with adaptive balanced domain prioritization. A multi-path NAS approach is developed to automatically design an MDL model with a learned parameter sharing strategy among domains. Based on the single-domain efficient NAS framework [1], the proposed multi-path NAS for MDL uses multiple reinforcement learning (RL) controllers, where each selects an optimal path from the super-network for each domain. To ensure mobile efficiency, the multi-path NAS network is learned based on the MobileNetV3-like search space and each RL controller optimizes its domain path based on the trade-off between accuracy and inference cost. Fig. 1 shows the multi-path architecture search framework. The model for one domain is represented by the nodes wired by a single color path.

An adaptive balanced domain prioritization algorithm is further proposed to balance the learning progress dynamically for domains of diverse difficulties in both the architecture search and model training phases. The selected multi-path model demonstrates consistent advantages when evaluated on the Visual Decathlon dataset [23], reducing the model size and FLOPS by 78% and 32% without sacrificing the average accuracy compared to the approach that simply constructs single-domain models.

The main contributions of this work are:

- We propose a generic multi-path neural architecture search (MPNAS) approach as the first work to apply ENAS to solve the challenges of on-device MDL, including data imbalance, domain diversity, negative transfer, domain scalability, and large search space of possible parameter sharing strategies. The MPNAS approach learns a cross-domain parameter sharing strategy to encourage positive knowledge transfer and also mitigate negative knowledge transfer via domain-wise path selection. The MPNAS approach achieves state-of-the-art accuracy, model size, and FLOPS on MobileNetV3-like architectures.
- We improve MDL model optimization on domains of diverse difficulties by using the generic adaptive balanced domain prioritization in the objective function.
- We present comprehensive studies for detailed insights into the network architecture of the learned MobileNetV3-like MDL model and the effect of different domain weighting approaches for MDL.

2. Related Work

Multi-Domain Learning. MDL focuses on learning a joint deep neural network that performs well across multiple in-

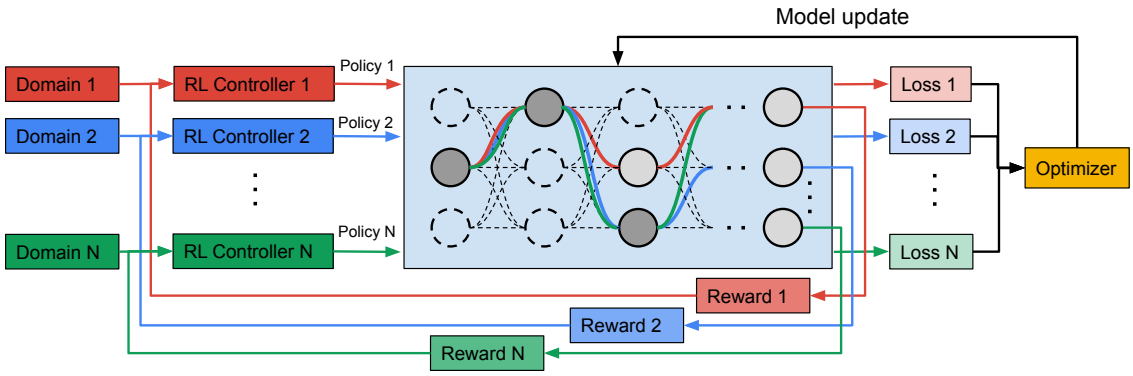

Figure 2: Multi-path neural architecture search framework. Each circle represents a candidate node in the super-network. Dark nodes are shared among domains. Light nodes are used by a single domain (non-shared). Dashed nodes are not selected by any domain and will be removed from the final model.

put domains. Most recent methods adapt pre-trained models to new tasks by adding few task-specific parameters. Bilen and Vedaldi [3] proposed a single network to learn the universal features for all domains by sharing all the parameters in the feature extraction layers except batch and instance normalization layers. Rebuffi et al. [23] extended the universal feature extraction network by introducing an adapter residual module to the standard residual network which makes the network adaptive to different visual domains. Further extensive studies [24] have been made on series and parallel residual adapters, joint adapter compression, and parameter allocations, which results in an empirically optimized network for MDL. Berriel et al. presented a budget-aware adapter module [2] to select the most relevant feature channels for each domain. On the other hand, incremental learning [26] is adopted for learning new domains sequentially without forgetting the knowledge learned on the previous domains. Existing MDL approaches showed that positive knowledge transfer can improve the performance over single domain learning with limited increase of model size. However, the hand-crafted sharing schemes and empirical optimization approaches which are designed for certain types of networks such as residual networks may not be generally applicable to other network architectures, e.g., mobile friendly architectures such as MobileNet [11].

Multi-Task Learning. MTL aims to simultaneously learn a diverse set of tasks, e.g., object detection, segmentation, depth estimation, by sharing knowledge among them. Different MTL models differ in strategies of sharing filters [19, 29, 28, 18] or sharing operations [17, 31, 9] among tasks, aiming to improve generalization by mining task relations and to suppress negative transfer. In contrast to our work, this line of research focuses on learning a diverse set of tasks in the same visual domain, usually all operating on the same image.

Other than hand-crafted MTL networks, routing networks [25] use an RL agent to learn the best route layer-by-layer. However, this method scales poorly with the size of the search space. A multi-task architecture search (MAS) [20] is proposed to search for a universal feature extraction architecture, but suffers from negative transfer among tasks.

Domain/Task Balanced Optimization. Since different domains/tasks have different levels of difficulties, one of the main challenges in training an MDL/MTL model is to strike a balance between domains/tasks when training simultaneously. Task balancing has been studied extensively in the MTL literature. Kendall et al. [12] used homoscedastic uncertainty as the weight which favors the task with less noisy annotations. Dynamic weight averaging (DWA) [16] is proposed to balance the learning pace across tasks to optimize all the tasks at a similar pace. Neither approach takes into consideration the task difficulties which may vary greatly in applications. To address this issue, the Dynamic task prioritization (DTP) [8] method is developed to prioritize difficult tasks by adjusting the weight of each single-domain loss dynamically. However, the DTP method needs a surrogate measurement for task difficulty, which may be impractical for certain problems. To be agnostic to the task difficulties, the balanced multi-task learning loss (BMTL) function [14] is proposed and shown to achieve promising results on MTL. Similar to the BMTL scheme, we develop a domain/task balancing method for MDL. In MDL, loss functions for different domains often share the same type and are of the same magnitude, which means loss function can be a good surrogate for task difficulties. However, this can be unrealistic in MTL (e.g., classification and segmentation tasks) where loss types differ. More comprehensive overview on MTL optimization can be found in [32].

Neural Architecture Search. NAS is a powerful paradigm for automatically designing neural architectures. Recent NAS work shift the focus from the expensive evolutionary NAS to efficient NAS [21, 15, 34, 4] by constructing

a super-network graph based on the search space and learning the architecture via end-to-end path sampling. While the existing efficient NAS frameworks achieve significant success on searching architectures for a single domain, finding effective MDL models still remains an active area. We propose a multi-path NAS method for MDL based on the latest single-domain NAS work TuNAS [1] which provides the advantages of handling substantially large and difficult search spaces to build mobile-friendly models without domain specific prior knowledge.

3. Multi-path NAS for MDL

The main modules of the proposed multi-path NAS algorithm for MDL are shown in Fig. 2. At the search stage, a super-network with multiple candidate paths (both solid and dot paths in Fig. 2) is constructed. Assuming that there are N domains $D = \{D_1, D_2, \ldots, D_N\}$, the proposed multi-path network uses N RL controllers $C = \{C_1, C_2, \ldots, C_N\}$, to manage the path selection for each domain. The circles represent candidate nodes in the super-network which are defined by the search space and the RL controllers learn to select nodes to build paths for each domain. The final model is formed by merging all the selected paths into a single network.

At each search iteration, C_i samples a single path for domain D_i. Each sampled path forms a sampled model for domain D_i. We first update the model weights using a single batch of images from each domain on the training set, then update the weights of RL controllers using the validation set for the corresponding domain. Specifically, the RL controller C_i receives a reward $\mathcal{R}_i$ (see section 3.1 for a function of validation accuracy and latency) from the model prediction. This $\mathcal{R}_i$ is then used to perform policy gradient update (REINFORCE [33]) on C_i. The process is repeated until it reaches the maximum epochs. The main search steps for the multi-path network are summarized in Algorithm 1.

At the end of the search phase, we take the most likely paths from each RL controller and form a single joint model with all the paths. As shown in Fig. 2, the model for one domain is constructed with the nodes connected by a path in one color. If more than one domain is connected to the same node in the super-network, the weights in that shared node will be used by all those domains (dark nodes with solid circles). If only one domain is routed to a node, the node will be exclusively used by that domain (light nodes with solid circles). Inside a conv node, each domain can selectively use a subset of filters. The filter number is also selected by the RL controller. The nodes not used by any domain will be removed from the final model (nodes with dashed circles).

The joint model is then trained from scratch with training data from all domains. We obtain domain predictions by running the corresponding domain data through its own path in the joint model. The final training process is similar to Algorithm 1 but with a fixed model and no RL controller. In each batch we compute each domain's training loss TrainLoss[i] by running the fixed model on domain training set. The joint TrainLoss, which is the sum of each individual domain's TrainLoss[i], will be back-propagated to update the model weights . The gradients for parameters in those shared nodes will be updated jointly.

Algorithm 1: Multi-path neural architecture search

```
Result: Multi-path network
Initialize RLControllers;
Initialize super-network from the search space;
for Epochs = 1 : MaxEpochs do
    for i = 1 : DomainCount do
        Sample one path for Domain[i] to form
          model;
        Run model on validation set to get
          Reward[i];
        Run model on training set to get
          TrainLoss[i];
    end
    Backprop the joint TrainLoss to update model
      coefficients in super-network;
    for i = 1 : DomainCount do
        Update RLControllers[i] with Reward[i]
          using REINFORCE;
    end
end
```

Fig. 3 shows a partial multi-path model architecture for the two domains on the Visual Decathlon dataset. Given that each RL controller independently selects the path based on the domain accuracy reward, the path similarity can also manifest the correlations among domains (see Section 5.2 for domain correlation analysis).

3.1. Reward Function

In order to determine architectures with good accuracy and latency trade-offs, we use the parameterized RL reward function proposed in the MnasNet [30] (also adopted by the ProxylessNAS [4]):

$$r(\alpha) = Q(\alpha) \times (T(\alpha)/T_0)^{\beta}, \tag{1}$$

where $Q(\alpha)$ represents the quality (validation accuracy) of a candidate architecture α, $T(\alpha)$ is the estimated inference time and T_0 is the target inference time. The cost exponent $\beta < 0$ is a tunable hyper-parameter. The reward function is maximized when model quality $Q(\alpha)$ is high and inference time $T(\alpha)$ is small.

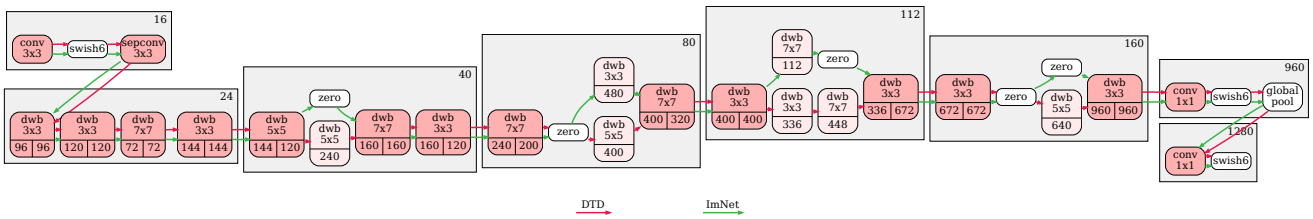

Figure 3: Multi-path model for ImageNet72 (green path) and Describable Textures (red path). Darker nodes are shared between domains while lighter nodes are independent. The numbers under the nodes denote the number of filters used by each domain in that conv node.

4. Domain Prioritization in Multi-path NAS

Handling domains with diverse difficulties is one of the main challenges in MDL. Given each domain has a different number of classes and difficulties, a method that directly optimizes the sum of losses is unlikely to perform well as shown in the ablation studies (Section 5.3). We propose an adaptive balanced domain prioritization (ABDP) method to achieve balanced training performance for all the domains by introducing a transformed loss objective function:

$$\min_{\theta} \sum_{i=1}^{N} h(\mathcal{L}(D_i;\theta)) + r(\theta), \tag{2}$$

where θ denotes the model parameters, $\mathcal{L}(D_i;\theta)$ represents the loss on domain D_i with the current model parameters, and $r(\theta)$ is the regularization term.

Since all domains share the same loss function in MDL, the loss $\mathcal{L}(D_i;\theta)$ can be a good surrogate to indicate the difficult of domain D_i. The boosting function $h(\cdot)$ in equation (2) is introduced to transform the loss subspace to a new subspace to boost the priorities of more difficult domains. Its derivative $h'(\mathcal{L}(D_i;\theta))$ can be viewed as the weight of the current domain D_i during gradient update. When $h(\cdot)$ is monotonically increasing, the domains with larger losses are favored. The loss function in Eq. (2) is adaptive in nature since it can dynamically adjust domain weights during the entire training phase.

If $h(\cdot)$ is a linear function, the objective function regresses to a linear weight of the single-domain losses which cannot achieve desired domain prioritization since $h'(\cdot)$ is constant. In the ablation studies (see Section 5.3), we evaluate multiple options for the boosting function $h(\cdot)$, including linear, polynomial, and exponential functions. Both the polynomial and exponential functions can amplify the loss which means the optimizer favours more difficult domains over easier ones. Our ablation studies show that a nonlinear boosting function can improve the model performance in MDL. Empirically, the exponential boosting function can most effectively boost performance on hard domains.

We further make the joint loss function adjustable during the search and training process by introducing a domain prioritization coefficient w in the boosting function. Assuming that we use the exponential function as the boosting function, the adaptive boosting function can be defined as:

$$h = \exp\Big(\frac{\mathcal{L}(D_i;\theta)}{w}\Big). \tag{3}$$

In Eq. (3), the adaptive parameter w can be put on a decay schedule throughout the training phase (e.g. linear decay from w_{max} to w_{min}). As w decreases, the domains with larger loss will become increasingly more important, which means the model favors difficult domains more at the later search/training stage. From our ablation studies, decreasing schedule of w outperforms constant schedule on the Visual Decathlon dataset. The results verify that adaptive prioritization can further optimize the MDL performance.

5. Experiments

We evaluate the proposed mobile-friendly MDL models in three aspects. First, we evaluate the number of parameters, FLOPS, and Top-1 accuracy of the MDL model constructed by the proposed multi-path NAS (MPNAS) model vs. other state-of-the-art approaches based on MobileNetV3-like architectures, including single domain NAS model bunlding, single path MobileNet pre-searched model, multi-domain single path NAS model, multiple random path model, and task routing layer (TRL) model proposed in [28]. Second, we analyze the selected MPNAS model and evaluate domain similarity by calculating the Jaccard similarity score [7] between domain path pairs. Third, we carry out ablation studies of the adaptive balanced domain prioritization.

Datasets. We carry out experiments on the Visual Decathlon dataset [23], which is composed of ten widely-used image classification problems representing sufficiently diverse visual domains. Each domain of this dataset has diverse image/class sizes and difficulties, which is crucial to verify the performance of MDL. The image resolution of the ten domains are normalized to 72×72 pixels which makes the classification task more challenging.

Evaluated methods. We evaluate the proposed method against the state-of-the-art approaches including:

Table 1: Number of parameters, FLOPS(G), and Top-1 accuracy (%) of MDL models on the Visual Decathlon dataset. All the methods are built based on the MobileNetV3-like search space. Blue number indicates the best result.

Models	# par.	FLOPS (G)	ImageNet72	Airc.	C100	DPed	DTD	GTSR	Flwr	OGlt	SVHN	UCF	Mean
SD-NAS	5.7x	1.08	**61.9**	22.6	69.6	**98.4**	34.2	99.5	66.7	**78.7**	**93.3**	**74.4**	69.9
SP-MN	1.0x	0.09	30.2	6.2	59.5	59.9	31.0	52.5	60.2	1.3	23.1	28.8	35.2
MDSP-NAS	**0.7x**	**0.04**	34.6	18.8	56.4	75.1	30.8	88.6	65.6	9.3	25.5	47.5	45.2
MRP	1.3x	0.43	50.7	21.5	67.3	96.1	31.5	**99.8**	61.8	75.8	91.9	69.5	66.6
TRL [28]	1.0x	0.71	27.1	16.1	60.0	66.2	33.9	95.8	63.3	54.3	85.7	46.2	54.9
MPNAS	1.3x	0.73	57.0	**29.7**	**81.2**	96.0	**36.6**	**99.8**	**79.2**	74.2	92.3	71.8	**71.8**

- Single domain NAS model (SD-NAS): A domain independent model selected on the MobileNetV3-like search space and trained from scratch on a single domain dataset.
- Single path MobileNet pre-searched model (SP-MN): A model whose backbone is searched using ImageNet72 based on the MobileNetV3-like search space. Domain specific output heads are attached to the final FC layer. All the domains share the main feature extraction network body.
- Multi-domain single path NAS model (MDSP-NAS): A single-path model selected by a single RL controller based on the MobileNetV3-like search space. The joint RL controller learns to select the path with the average accuracy as reward. Like SP-MN, all domains use a shared network body with each output head attached to the final FC layer.
- Multiple random path model (MRP): A joint model with a random path assigned to each domain. The paths are randomly selected from the super-network generated by the MobileNetV3-like search space. Results are averaged across ten trials to reduce noise.
- Task routing layer model (TRL): The model with task routing layer for MDL proposed in [28]. For fair comparisons, we add the task routing layer on top of the MobileNetV3-Large baseline model. Since the capacity of MobileNetV3-like models is very small, we set the ratio of active filters for each task to 0.9 which achieves the best performance in our experiments.
- Multi-path NAS model (MPNAS): The proposed multi-path model obtained by using multiple RL controllers to select paths for each domain based on the MobileNetV3-like search space. The shared and domain specific nodes are learned automatically by the architecture search.

Architecture search space. The proposed MPNAS is developed based on the MobileNetV3-like search space within the TuNAS [1] framework. The MobileNetV3-like [11] search space represents the state-of-the-art mobile-friendly classification model. It consists of a stack of blocks with convolution layers, inverted bottleneck layers, hard-swish activation layers, a compact head, and optional Squeeze-and-Excite layers. The searchable parameters includes the number of layers per block, the positions of Squeeze-and-Excite layers, kernel size, bottleneck expansion factors within $\{1, 2, \cdots, 6\}$ and different filter sizes. For fair comparisons, we apply the TRL method on the MobileNetV3-like baseline model.

Implementation details. All the model search and training jobs are conducted on Cloud TPU V3 with 32 TPU chips. Each search takes 90 epochs on ImageNet72. The batch size in search is 1024 for each task. The super-network model and the searched model weights are trained using RMSProp with a cosine decay learning rate schedule. For search, the learning rate schedule starts at 8e-3. After it converges, the selected multi-path networks are trained from scratch with the learning rate schedule starting at 0.0325. During search, our RL controllers start training after 1/8 total steps using Adam with a constant learning rate of 0.165. For the RL reward function (1), we set $T_0 = 84ms$ following [1]. We tuned $\beta \in [-0.09, -0.03]$ and selected $\beta = -0.07$ based on the average accuracy. The training stage runs until reaching 360 epochs on ImageNet72. The training stage batch size is set to 416 for each domain.

5.1. Model Accuracy

Table 1 summarizes the top-1 accuracy for all the evaluated methods on the Visual Decathlon dataset. Compared to the SD-NAS approach, the performance of SP-MN degrades significantly due to the negative interference among domains. The MDSP-NAS model slightly improves over the SP-MN model due to joint architecture search over all domains instead of searching on ImageNet72 only. The multi-path models, including MRP and MPNAS significantly improves the accuracy over the MDSP-NAS model by 25% to 30% due to the domain specific feature learning supported by the diverse path for each domain. MPNAS also significantly outperforms TRL on the MobileNetV3-

Table 2: Top 1 accuracy (%) of the highly correlated domains. Blue number indicates the best in each column.

Models	C100	ImageNet72	Flwr	Avg
SD-NAS	69.63	**61.87**	66.73	66.07
MPNAS	**79.74**	59.62	**71.76**	**70.37**

Large architecture by 16.9% since the MPNAS method supports selecting different operations between different paths if the two domains learn not to share. In addition, the proposed MPNAS model achieves a slight improvement of 1.9% over the SD-NAS scheme because of the positive knowledge transfer among domains. The MPNAS method therefore achieves the best performance over all the MobileNetV3-like architectures.

To demonstrate the positive transfer effect among correlated domains, we evaluate the proposed MPNAS method on three domains which are visually similar and empirically correlated: ImageNet72, Cifar-100, and VGG_Flowers. Table 2 shows that the MPNAS model can improve the accuracy by 10% on Cifar-100 and 5% on VGG_Flower with very small (2%) sacrifices on ImageNet72.

5.2. Architecture Analysis

Intuitively, more correlated domains should share more nodes in order to maximize positive knowledge transfer. To measure architecture similarity between domains, we compute the pairwise Jaccard similarity score [22] between the set of selected nodes in two domains, as shown in Fig. 4. A higher Jaccard similarity score means the paths are more similar and thus the domains correlation is higher. The results show the paths generated by MPNAS are aligned with the empirical experience. For example, the paths for similar domains, e.g., ImageNet, Cifar-100 and VGG_Flower, have high Jaccard similarity scores while the paths for dissimilar domains, e.g., GTSR and UCF, DPed and UCF, have low Jaccard similarity scores. These results again demonstrate that the proposed MPNAS can learn an effective parameter sharing strategy to increase positive transfer.

5.3. Ablation Studies

5.3.1 Adaptive Balanced Domain Prioritization

We carry out experiments on the MDSP-NAS, SP-MN, MRP, and MPNAS models with or without the proposed ABDP method for ablation studies. Table 3 summarizes the pairwise experimental results. Overall, applying the ABDP can always improve average training accuracy on Visual Decathlon dataset regardless of model architecture. For the single-path models (MDSP-NAS and SP-MN), the ABDP method greatly improves the performance since the

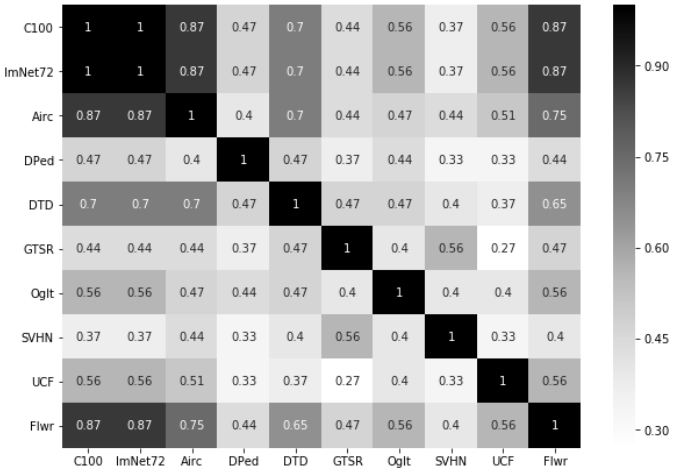

Figure 4: Confusion matrix for the Jaccard similarity score between the paths for the ten domains. The score value ranges from 0 to 1. A greater value indicates two paths share more nodes.

model optimization on the nodes affects all the domains. For the multi-path models (MRP and MPNAS), the performance gain is smaller since the joint loss gradient on a certain node does not affect domains that do not use this node. The results also show that the ADBP method is effective in prioritizing more difficult domains and achieving larger performance gains.

5.3.2 Evaluation of Joint Optimizer

In this ablation study, we evaluate model accuracies with various MTL loss functions, including uncertainty weighted loss (UWL) [12], empirical weighted loss (Empirical), identity (no boosting, x), ABDP with quadratic polynomial boosting (x^2) and ABDP with the exponential boosting ($\exp(x/w)$) functions.

The exponential boosting loss function can be further enhanced by decreasing or increasing the domain prioritization coefficient w during the training stage. For the linear decreasing schedule $\exp(x/w\ \downarrow)$, w is decreased from w_{max} to w_{min} throughout the training process which means harder domains are weighted more in the later search/training stage (we use $w_{max} = 2$ and $w_{min} = 1$). For a linear increasing schedule $\exp(x/w\uparrow)$, w is increased which means favoring harder domains in earlier stages. For the empirical weighted loss, we double the loss weight for ImageNet72 since it is one of the most challenging domains.

The results in Table 4 show that all the boosting functions outperform the uncertainty weighted loss (UWL) [12] method. The exponential boosting function with decay coefficient schedule improves the average accuracy over UWL by 3.5%. That improvement also validates the previous as-

Table 3: Top 1 accuracy (%) between the approaches using and not using adaptive balanced task prioritization. Blue numbers denote better results in pairwise comparisons.

Domains	MDSP -NAS	MDSP -NAS +ABDP	SP -MN	SP -MN +ABDP	MRP	MRP +ABDP	MPNAS	MPNAS +ABDP
ImageNet72	27.50	**34.57**	21.84	**30.15**	44.46	**50.70**	48.46	**57.02**
Airc	7.10	**18.78**	5.29	**6.19**	21.48	**21.53**	26.12	**29.65**
C100	52.90	**56.43**	50.26	**59.45**	**67.70**	67.35	74.86	**81.16**
DPed	73.60	**75.07**	51.00	**59.87**	96.10	96.10	**97.27**	96.02
DTD	32.40	**30.80**	25.92	**31.03**	30.71	**31.49**	36.43	**36.61**
GTSR	51.90	**88.64**	40.77	**52.50**	**99.89**	99.77	**99.92**	99.80
Flwr	59.90	**65.59**	54.00	**60.19**	60.96	**61.75**	72.70	**79.18**
Oglt	1.80	**9.33**	**1.59**	1.27	**76.06**	75.76	**76.34**	74.19
SVHN	**27.60**	25.52	20.66	**23.06**	**92.29**	91.89	**92.35**	92.34
UCF	27.00	**47.53**	19.52	**28.75**	**71.11**	69.48	**75.17**	71.79
Avg	36.17	**45.23**	29.08	**35.25**	66.08	**66.58**	69.96	**71.78**

Table 4: Top 1 accuracy (%) achieved by various loss functions. Blue numbers are the best in the column.

Funcs.	ImageNet72	Airc.	C100	DPed	DTD	GTSR	Flwr	OGlt	SVHN	UCF	Mean
UWL [12]	41.78	24.65	72.09	95.40	34.46	**99.93**	71.72	74.91	92.16	**75.46**	68.26
Empirical	53.78	26.57	76.22	96.48	36.86	99.90	76.24	75.80	92.06	69.88	70.38
x	48.46	26.12	74.86	97.27	36.43	99.92	72.70	76.34	92.35	75.17	69.96
x^2	54.86	27.88	80.13	97.18	38.55	99.88	74.32	74.55	92.35	72.10	71.18
$\exp(x/w)$	55.29	30.04	76.78	96.73	37.54	99.59	71.54	76.07	91.49	69.98	70.51
$\exp(x/w \uparrow)$	49.16	**31.17**	78.65	**97.38**	**38.89**	99.87	78.19	**77.87**	**92.49**	73.40	71.71
$\exp(x/w \downarrow)$	**57.02**	29.65	**81.16**	96.02	36.61	99.80	**79.18**	74.19	92.34	71.79	**71.78**

sumption that transforming the loss subspace can help to optimize the model within the given training strategy over the weighted sum approaches. Moreover, both the exponential and square functions outperform the identity function which has no loss boosting. These results show that the proposed ABDP method improves MDL optimization when training jointly with multiple diverse domains. The linear decay schedule slightly improves the accuracy over the linear increasing schedule on the Visual Decathlon dataset as it reduces the interference between domains when the training process is converging.

Compared with setting domain weights empirically, the ABDP approach does not require prior knowledge of relative domain difficulties, which makes it easier to generalize and adapt to different settings.

6. Conclusions

In this paper, we propose a multi-path NAS approach as the first work of applying efficient NAS to solve the challenges of on-device multi-domain learning (MDL), including data imbalance, domain diversity, negative transfer, domain scalability, and large search space of possible parameter sharing strategies. By learning to selectively share parameters among domains, the multi-path network is able to reduce the total number of parameters and FLOPS, encourage positive knowledge transfer, and mitigate negative interference. The adaptive balanced domain prioritization algorithm achieves balanced performance among domains by dynamically adjusting the weighting of each domain during the search and training phases. Extensive experiments demonstrate that the proposed multi-path network achieves state-of-the-art accuracy on the 10-domain Visual Decathlon dataset with the mobile-friendly MobileNetV3-like network architecture. The selected model searched in the MobileNetV3-like search space reduces the number of parameters and FLOPS by 78% and 32%, respectively compared to the approach by simply constructing the single-domain models without sacrificing model accuracy.

References

[1] Gabriel Bender, Hanxiao Liu, Bo Chen, Grace Chu, Shuyang Cheng, Pieter-Jan Kindermans, and Quoc V Le. Can weight sharing outperform random architecture search? an investigation with tunas. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 14323–14332, 2020.

[2] Rodrigo Berriel, Stephane Lathuillere, Moin Nabi, Tassilo Klein, Thiago Oliveira-Santos, Nicu Sebe, and Elisa Ricci. Budget-aware adapters for multi-domain learning. In *IEEE International Conference on Computer Vision*, pages 382–391, 2019.

[3] Hakan Bilen and Andrea Vedaldi. Universal representations: The missing link between faces, text, planktons, and cat breeds. *arXiv preprint arXiv:1701.07275*, 2017.

[4] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332*, 2018.

[5] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International Conference on Machine Learning*, pages 794–803, 2018.

[6] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014.

[7] John C Gower and Matthijs J Warrens. Similarity, dissimilarity, and distance, measures of. *Wiley StatsRef: Statistics Reference Online*, pages 1–11, 2014.

[8] Michelle Guo, Albert Haque, De-An Huang, Serena Yeung, and Li Fei-Fei. Dynamic task prioritization for multitask learning. In *European Conference on Computer Vision*, pages 270–287, 2018.

[9] Pengsheng Guo, Chen-Yu Lee, and Daniel Ulbricht. Learning to branch for multi-task learning. *arXiv preprint arXiv:2006.01895*, 2020.

[10] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *IEEE International Conference on Computer Vision*, pages 2961–2969, 2017.

[11] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *IEEE International Conference on Computer Vision*, pages 1314–1324, 2019.

[12] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 7482–7491, 2018.

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[14] Sicong Liang and Yu Zhang. A simple general approach to balance task difficulty in multi-task learning. *arXiv preprint arXiv:2002.04792*, 2020.

[15] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

[16] Shikun Liu, Edward Johns, and Andrew J Davison. End-to-end multi-task learning with attention. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1871–1880, 2019.

[17] Yongxi Lu, Abhishek Kumar, Shuangfei Zhai, Yu Cheng, Tara Javidi, and Rogerio Feris. Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5334–5343, 2017.

[18] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *European Conference on Computer Vision*, pages 67–82, 2018.

[19] Alejandro Newell, Lu Jiang, Chong Wang, Li-Jia Li, and Jia Deng. Feature partitioning for efficient multi-task architectures. *arXiv preprint arXiv:1908.04339*, 2019.

[20] Ramakanth Pasunuru and Mohit Bansal. Continual and multi-task architecture search. In *ACL*, 2019.

[21] Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. *arXiv preprint arXiv:1802.03268*, 2018.

[22] Anand Rajaraman and Jeffrey David Ullman. *Mining of massive datasets*. Cambridge University Press, 2011.

[23] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. In *Advances in Neural Information Processing Systems*, pages 506–516, 2017.

[24] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Efficient parametrization of multi-domain deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 8119–8127, 2018.

[25] Clemens Rosenbaum, Tim Klinger, and Matthew Riemer. Routing networks: Adaptive selection of non-linear functions for multi-task learning. *arXiv preprint arXiv:1711.01239*, 2017.

[26] Amir Rosenfeld and John K Tsotsos. Incremental learning through deep adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

[27] Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems*, pages 527–538, 2018.

[28] Gjorgji Strezoski, Nanne van Noord, and Marcel Worring. Many task learning with task routing. In *IEEE International Conference on Computer Vision*, pages 1375–1384, 2019.

[29] Ximeng Sun, Rameswar Panda, and Rogerio Feris. Adashare: Learning what to share for efficient deep multi-task learning. *arXiv preprint arXiv:1911.12423*, 2019.

[30] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2820–2828, 2019.

[31] Simon Vandenhende, Stamatios Georgoulis, Bert De Brabandere, and Luc Van Gool. Branched multi-task net-

works: deciding what layers to share. *arXiv preprint arXiv:1904.02920*, 2019.

[32] Simon Vandenhende, Stamatios Georgoulis, Marc Proesmans, Dengxin Dai, and Luc Van Gool. Revisiting multi-task learning in the deep learning era. *arXiv preprint arXiv:2004.13379*, 2020.

[33] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 1992.

[34] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. Snas: stochastic neural architecture search. *arXiv preprint arXiv:1812.09926*, 2018.

[35] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 6848–6856, 2018.