Transferable AutoML by Model Sharing over Grouped Datasets

Chao Xue[1], Junchi Yan[2], Rong Yan[1], Stephen M. Chu[1], Yonggang Hu[3], Yonghua Lin[1]
[1]IBM Research – China, [3]IBM Systems
[2]Department of CSE and MoE Key Lab of Artificial Intelligence, Shanghai Jiao Tong University
{xuechao,yanrong,schu,linyh}@cn.ibm.com, yanjunchi@sjtu.edu.cn, yhu@ca.ibm.com

Abstract

Automated Machine Learning (AutoML) is an active area on the design of deep neural networks for specific tasks and datasets. Given the complexity of discovering new network designs, methods for speeding up the search procedure are becoming important. This paper presents a so-called transferable AutoML approach that leverages previously trained models to speed up the search process for new tasks and datasets. Our approach involves a novel meta-feature extraction technique based on the performance of benchmark models, and a dynamic dataset clustering algorithm based on Markov process and statistical hypothesis test. As such multiple models can share a common structure while with different learned parameters. The transferable AutoML can either be applied to search from scratch, search from predefined models, or transfer from basic cells according to the difficulties of the given datasets. The experimental results on image classification show notable speedup in overall search time for multiple datasets with negligible loss in accuracy.

1. Introduction and Related Work

With the wide adoption of deep networks, identifying proper network architectures tailored to specific task and dataset is attractive while it still mainly relies on human expertise. This has motivated rapidly growing research on automatically discovering the tailored network models[1] for training in a fully automatic fashion without human intervention, which is referred to as AutoML.

There has been considerable literature on AutoML using methods based on genetic algorithms, random search, Bayesian optimization, reinforcement learning and continuous differentiable methods. Genetic algorithm based method [33] is introduced in the last century to find both architectures and weights. However, they fail to match the performance of handcrafted networks. [32] proposes an evolutionary searching methods for generating architectures. The accuracy approaches those of handcrafted models in some cases [14, 41, 15]. Similarly, random search has been explored to choose hyper-parameters. [6] shows empirically and theoretically that random search is more efficient than grid search for hyper-parameter optimization. In particular, the method Hyperband [22] speeds up random search using an early-stopping strategy to allocate resources adaptively. Bayesian optimization [5, 34, 16, 12, 7] also provides a sound foundation for AutoML. One choice for Bayesian optimization is to model the generalization performance as a sample from a Gaussian process (GP) [34], which can reach expert-level optimization performance for many machine learning algorithms. The work in [3] develops a Q-learning agent to pick the modules of the layers in a neural network architecture. A Neural architecture search (NAS) method proposed by [42] employs a policy gradient method to learn networks from a recurrent network and has achieved good results in tasks in both vision and language. Recently, the continuous domain differentiable method has gained interests for AutoML. Unlike evolution or reinforcement learning that is over a discrete search space, DARTS [25] uses a continuous relaxation of the architecture representation and search the architecture by gradient descent.

However, most existing AutoML approaches require considerable overhead for model searching. To improve efficiency, one idea is to share information across different trials. Multi-task Bayesian optimization and Gaussian processes (GP) are proposed in [35, 8, 4]. In these methods, different datasets are regarded as different tasks, and the covariances between hyper-parameter and task pairs are defined. The multi-task Bayesian optimization and Gaussian process show some success with established theoretical foundation. Meanwhile warm-starting of sequential model-based algorithm configuration is introduced by [24, 26, 30]. In [24], it exploits one trial's performance to warm-start its model configuration on the new job type. The work [26] uses meta-learning for initializing the Bayesian optimizer and automated ensemble construction from configurations

[1]For deep learning, the meaning of 'model' in this paper includes both the network architecture and learning related hyper-parameters.

978-1-7281-3293-8/19/$31.00 ©2019 IEEE
DOI 10.1109/CVPR.2019.00921

evaluated during optimization. The work [30] learns model similarity by building a shared (multi-task) representation for the hyper-parameter space. While these multi-task and warm-starting methods are based on GP or Bayesian optimization, Wong *et al*. [37] focus on sharing knowledge across tasks in deep RL architecture. Net2Net [10] accelerates the search process by transferring the knowledge from a previous network to a new deeper or wider one, and ENAS [31] shares weights for all child models to build a one-shot model and thus it speeds up the convergence process, which is similar to what DARTS [25] does. In [43] the authors show the basic cell searched on CIFAR-10 can be transferred to ImageNet classification without much modification. However, the interpretation of the effectiveness of sharing model among different datasets is still not clear.

This paper addresses this important setting that more than one datasets are combined to share one common structure of network due to their inherent correlation, while the parameters are different and individually learned to fit with each dataset. Such a mechanism can be used to achieve a tradeoff between search time and model accuracy. Under this setting, we propose a novel approach that involves a meta-feature extraction technique and a dynamic dataset clustering algorithm to reuse the appropriate model (architecture + hyper-parameters) for multiple datasets with a reduction in search time. Our method enjoys some flexibilities against existing methods in three folds:

i) Search algorithms. Unlike multi-task solutions [35, 8, 4] designed for Bayesian optimization, or transfer learning with AutoML [37] based on reinforcement learning, our approach can be easily combined with most existing AutoML techniques in an out-of-box fashion. Examples include genetic methods [32], reinforcement learning [3, 42], Hyperband [22] and DARTS [25]. This is because our method focuses on dataset clustering which is orthogonal with the specific model search algorithm.

ii) Search mechanisms. Our transferable AutoML can be applied to different search schemes: search from scratch; search from predefined models (e.g. reuse GoogleNet architecture and weights of bottom layers to search an architecture for higher layers) and transfer from basic cells (transfer the searched normal/reduction cell [25, 31] of source datasets to target datasets). This feature makes it more flexible to handle datasets under limited time budget.

iii) Online setting. Our method can be used to the online setting whereby the datasets come sequentially and one need to search model for the new arrival datasets efficiently.

Specifically we develop techniques for dataset clustering and model sharing among clustered datasets which has not been well studied in literature to our best knowledge. The main contributions and novelties of the paper are:

i) Meta-features extracting. To enable effective model search and sharing, we propose a new meta learning method for dataset feature representation using their evaluation results on a suite of benchmark models.

ii) Dataset clustering. The extracting meta-features is then coupled with Markov process and hypothesis test mechanism for dataset clustering. These two components can handle Type II error and Type I error for dataset grouping (incorrectly accepting grouping and incorrectly rejecting grouping), respectively.

Now we present the overview of our approach. We first represent a dataset d_n with a benchmark model-specific representation (see more details in Sec. 2.1) $\boldsymbol{x}_n$ in the dataset feature space. The basic idea is to leverage such a meta-learning representation to measure the similarity over datasets such that certain datasets can be grouped for model search and sharing. Since the grouping need be performed online, we adopt the Markov analysis for sequential clustering using the above representation which also involves the concept of cluster sets in the dataset feature space for Bayesian inference. Such a clustering step can handle Type II error. To control the Type I error, we further impose a hypothesis test to inhibit the unwanted grouping.

As such, as the datasets continue to come, either a new model will be searched for the new dataset by using some AutoML methods, or the new dataset will be assigned to a cluster set with existing datasets, to share with a common model (including both hyper-parameters and architecture or basic cell), though their weights are different.

2. Shared Model Search by Grouping Datasets

We first justify the motivation of our benchmark based meta-learning method for dataset feature representation. Then we adopt Markov analysis of sequential clustering and statistical hypothesis tests to group datasets such that the searched model can be reused within each group.

2.1. Dataset Feature Extraction

We introduce a meta learning method to express a dataset d in a dataset feature space Ω_d. To prove this representation can work well for AutoML, we consider the AutoML problem first. The basic idea for AutoML is to identify the model m from a given dataset d:

$$m^* = \arg\max_m p(m|d) \tag{1}$$

It is reasonable to share m^* among the datasets that above posterior distributions conditioned on are closely approximated. To compare the posterior distributions over models between two different datasets, $p(m|d_1)$ and $p(m|d_2)$, we use Kullback-Leibler (KL) divergence:

$$\begin{aligned} &KL\left(p(m|d_1)||p(m|d_2)\right) \\ &= \int p(m|d_1) \ln \left\{ \frac{p(m|d_1)}{p(m|d_2)} \right\} dm \\ &\approx \sum_{b_i} \frac{p(b_i)p(d_1|b_i)}{\sum_{b_j} p(b_j)p(d_1|b_j)} \ln \left(\frac{p(d_1|b_i)}{p(d_2|b_i)} \frac{\sum_{b_j} p(b_j)p(d_2|b_j)}{\sum_{b_j} p(b_j)p(d_1|b_j)} \right) \end{aligned} \tag{2}$$

The discretization approximation is close to the continuous KL divergence when the benchmark model set $\{b_i\}$ expands the whole model space. It is well known that the relative entropy satisfies $KL(p||q) \geq 0$ with equality if and only if $p(x) = q(x)$. Therefore, given some benchmark models sampled from the feasible model space, and if the model evidences $p(d|b)$ are similar over all benchmark models between two datasets, the KL divergence in Eq. 2 is then approximated by 0. This leads to the feasibility for sharing one model over multiple datasets which also lies foundation for this paper.

Now we discuss how to use the benchmark models to express a given dataset. Formally, assume there are B benchmark (deep network) models in the model space: $b_1, b_2, \ldots, b_B$. The configuration of benchmark models spreads over different neural networks, w.r.t. the number of hidden layers, the number of hidden unit, the kernel size, the stride, the skip pattern, the number of nodes in a cell, etc., as well as different hyper-parameters like learning rate, weight decay, momentum, batch size etc.. Without loss of generality, in this paper the benchmark model set is formed by Monte Carlo sampling over the feasible region from uniform or log-uniform distribution. We leave more effective construction for future work.

Then the dataset d can be represented by a feature vector $\boldsymbol{x}_n = f(d, b_1, ..., b_B) \in \Omega_d$ in the derived dataset feature space Ω_d. In particular the value at dimension i is set by: $\boldsymbol{x}_n(i) = g(d_n, b_i)$ where $g(d, b)$ returns the evaluation result (e.g. accuracy) for dataset d using benchmark model b[2]. Based on such a normalized representation, in the following we will show how dataset grouping can be performed online to reduce the overall model search overhead by model sharing within the dataset group (see the comparison with the peer method [26] in Table 2).

It is worth noting that there are alternative meta-learning approaches [24, 26, 17] for dataset feature representations. But our proposed method above is based on the benchmark models' performance probing to datasets. The hope is that the similarity between datasets can be measured by their closeness in terms of the model performance rather than other criteria.

2.2. Markov Analysis for Sequential Clustering

To group the streaming datasets $d_1, d_2, \ldots, d_n$ over time, as a common practice in dynamic clustering, we introduce K surplus clusters $V^1, V^2, \ldots, V^K$ for initialization (K is very large and it is pruned out finally). Each cluster takes a randomly sampled value in the dataset feature space Ω_d and in general each dataset is assigned to a certain cluster and the feature value of that cluster will be updated accordingly. In the beginning when no dataset arrives, each cluster is empty with no assigned dataset. There are enumerable states to encode the assignment of existing datasets into the clusters. In this paper, we denote each state by $s_n = i$ where i denotes a certain datasets vs. clusters assignment.

Consider the following Markov chain for assignment decision modeling from state s_{n-1} to s_n:

$$\begin{aligned} &P[s_n = i|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n-1}, \boldsymbol{x}_n] \qquad (3)\\ =&\sum_j P[s_n = i|\boldsymbol{x}_n, s_{n-1} = j] P[s_{n-1} = j|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n-1}]\\ =&P[s_n = i|\boldsymbol{x}_n, s_{n-1} = i'] P[s_{n-1} = i'|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n-1}] \end{aligned}$$

The equation shows that there is only one state that the current state transitioned from. The first part of the product in Eq. 3 can be computed by Bayes' rule:

$$\begin{aligned} &P[s_n = i|\boldsymbol{x}_n, s_{n-1} = i']\\ =&\frac{p[\boldsymbol{x}_n|s_n = i, s_{n-1} = i'] P[s_n = i|s_{n-1} = i']}{p[\boldsymbol{x}_n|s_{n-1} = i']} \qquad (4) \end{aligned}$$

For dataset grouping, the hope is that the grouped datasets assigned to the same cluster shall have similar feature representation $\boldsymbol{x} \in \Omega_d$, and we further assume $\boldsymbol{x}$ with the same cluster obey a Gaussian distribution (recall the dimension of space Ω_d is B). In fact, the assumption of Gaussian distribution for validation error (accuracy) is widely used in hyper-parameter tuning [5, 35]. By denoting the state $s_n = i$ as the updated state when d_n is assigned to cluster V^k while $s_{n-1} = i'$, we have:

$$\begin{aligned} &p[\boldsymbol{x}_n|s_n = i, s_{n-1} = i']\\ =&p[\boldsymbol{x}_n|d_n \to V^k, s_{n-1} = i']\\ =&N(\boldsymbol{x}|\boldsymbol{\mu}^k, \boldsymbol{\Sigma}^k)\\ =&\frac{\exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}^k)^\top (\boldsymbol{\Sigma}^k)^{-1} (\boldsymbol{x} - \boldsymbol{\mu}^k)\right)}{(2\pi)^{B/2} |\boldsymbol{\Sigma}^k|^{1/2}} \qquad (5) \end{aligned}$$

where $\boldsymbol{\mu}^k \in \mathcal{R}^B$ is the mean and $\boldsymbol{\Sigma}^k \in \mathcal{R}^{B \times B}$ is the covariance matrix[3].

Now we go back to the right part of the numerator in Eq. 4. This part can be viewed as class priors:

$$\begin{aligned} P[s_n = i|s_{n-1} = i'] =&P[d_n \to V^k|s_{n-1} = i'] \qquad (6)\\ =&\begin{cases} w \cdot \frac{|V^k|}{n-1}, & V^k \neq \emptyset \\ (1-w) \cdot \frac{1}{K - |I^k|}, & V^k = \emptyset \end{cases} \end{aligned}$$

where $|\cdot|$ denotes the cardinality of the set. Similar to the expectation maximization for mixture of Gaussian clustering, the prior probability of one cluster is assumed to be

[2]Note to obtain a more reliable performance estimation, multiple randomly initialized trials are performed for each model, and here $\boldsymbol{x}_n$ denotes the mean of these trials for a model on the validation set.

[3]In this paper we slightly abuse the notation for cluster: V^k is a set *with additional value attributes in the dataset feature space*. While we treat it as a standard set in formulas: when it is empty we denote $V^k = \emptyset$ and use $V^k \bigcup \{d_n\}$ for a set union operation.

proportional to the number of assigned datasets. For empty clusters, their prior probabilities are supposed to be equal. Hence we use the piecewise function in Eq. 6, where the set of nonempty clusters is defined as: $I^k = \{k|V^k \neq \emptyset, k = 1, \ldots, K\}$. In addition, w is the prior probability of a dataset belonging to the nonempty clusters ($V^k \neq \emptyset$), which increases as the number of nonempty clusters grows. Hence the following function is devised to model w:

$$w = 1 - \exp(-|I^k| \cdot \gamma) \tag{7}$$

where γ is a hyper-parameter, and its value can be chosen by using grid search across its sensitive range that is shown in supplementary material. Finally, considering the denominator of Eq. 4, it can be marginalized over all preset clusters in the dataset feature space:

$$\begin{aligned} & p[\boldsymbol{x}_n|s_{n-1} = i'] \\ = & \sum_{k \in I^k} w \frac{|V^k|}{n-1} N(\boldsymbol{\mu}^k, \boldsymbol{\Sigma}^k) + \sum_{k \notin I^k} (1-w) \frac{N(\boldsymbol{\mu}^k, \boldsymbol{\Sigma}^k)}{K - |I^k|} \end{aligned} \tag{8}$$

The initial conditions are as follows:

$$P[s_1 = 1|\boldsymbol{x}_1] = 1 \tag{9}$$

$$\boldsymbol{\mu}^k = \boldsymbol{\mu}_0^k \tag{10}$$

$$\boldsymbol{\Sigma}^k = \boldsymbol{\Sigma}_0^k = \frac{\sigma_0^2 I}{K^{\frac{1}{B}}} \tag{11}$$

where $\boldsymbol{\mu}_0^k$ can be sampled randomly from uniform distribution when the bound of range is known or is sampled from a Gaussian distribution when the mean and variance are known. Here $\boldsymbol{\Sigma}_0^k$ is set as a diagonal matrix given unknown correlation in the beginning, and σ_0^2 can be roughly set around the estimation of the variance of separate trials, whose scale is inversely proportional to $K^{\frac{1}{B}}$ as more preset clusters can lead to less variance. These parameters can be easily updated by likelihood maximization, the detailed update criteria are shown as follows:

$$V^k = \begin{cases} V^k \bigcup \{d_n\}, & d_n \to V^k \\ V^k, & d_n \to V^{k'}, k' \neq k \end{cases} \tag{12}$$

The update for $\boldsymbol{\mu}_n^k$ and $\boldsymbol{\Sigma}_n^k$ are as follows:

$$\boldsymbol{\mu}_n^k = \begin{cases} \frac{\sum_i^{d_i \in V^k} \boldsymbol{x}_i}{|V^k|}, & d_n \to V^k \bigwedge V^k \neq \emptyset \\ \boldsymbol{x}_n, & d_n \to V^k \bigwedge V^k = \emptyset \\ \boldsymbol{\mu}_{n-1}^k, & d_n \to V^{k'}, k' \neq k \bigwedge V^k \neq \emptyset \end{cases} \tag{13}$$

$$\boldsymbol{\Sigma}_n^k = \begin{cases} \frac{\sum_i^{d_i \in V^k} (\boldsymbol{x}_i - \boldsymbol{\mu}^k)(\boldsymbol{x}_i - \boldsymbol{\mu}^k)^\top}{|V^k|}, & d_n \to V^k \bigwedge V^k \neq \emptyset \\ \frac{\sum_{t=1}^T (\boldsymbol{x}_n^t - \boldsymbol{x}_n)(\boldsymbol{x}_n^t - \boldsymbol{x}_n)^\top}{T}, & d_n \to V^k \bigwedge V^k = \emptyset \\ \boldsymbol{\Sigma}_{n-1}^k, \quad d_n \to V^{k'}, & k' \neq k \bigwedge V^k \neq \emptyset \end{cases} \tag{14}$$

where T is the number of trials by random initialization for one benchmark model. Recall $\boldsymbol{x_n}$ is the mean vector over the T trials. The update criteria are easy to understand: the parameters in the nonempty cluster that the current dataset is assigned to are updated by max-likelihood, and the parameters in other nonempty clusters remain unchanged. While the parameters in the empty cluster that the current dataset is assigned to are set to the estimation of itself, as the current dataset is the only one in the cluster.

As a consequence, the probability for next cluster assignment in Eq. 3 can be calculated with Eq. 4-14. Accordingly the best cluster can be found with the highest probability such that the datasets associated with that cluster can share the same model to avoid model search for new datasets.

However, due to the high demands of computation and memory requirements, computing Eq. 3 directly is often unfeasible as the state number is a Bell number [2] and grows exponentially with the number of datasets, even considering the simplification of states by reducing the empty clusters. To improve the efficiency, one solution is to model the problem as a shortest path problem and preserve only the most effective path. However the dynamic programming models like the Viterbi algorithm cannot improve performance in this case because the current state is transitioned from only one state. To obtain an approximately optimal solution, we make an assumption that the determinant of $|\boldsymbol{\Sigma}^k|$ is small enough, which is also found empirically. Then it is easy to show that the state with the highest posterior probability Eq. 4 at dataset d_n will be transitioned from that with the highest one at dataset d_{n-1}. Therefore, we can only take the state with the highest posterior probability Eq. 4 along the iteration process to get an approximately optimal solution.

As described above, our method can be viewed as a hard decision scheme. There are also soft decision sequential clustering methods like online EM algorithms [9, 23]. While grouping the datasets into clusters by hard decision is more direct because it neither need inefficient iterations nor requires a predefined accurate number of clusters. Since k-means is the well known hard decision cluster method, as well as its sequential version [13, 20], we will compare with k-means and sequential k-means in the experiments.

2.3. Statistical Hypothesis Test

The above Markov grouping procedure can control the Type II error. Now we introduce a hypothesis test technique to handle the Type I error. Statistical test was involved in k-means by [13], whereby the test is performed to detect whether the data assigned to a cluster is sampled from one Gaussian distribution. Since hypothesis test for fitting distributions requires a large number of samples [1] which is not suited in our model search setting, we turn to consider a test to detect whether two Gaussian distributions have the same mean. Specifically the alternative hypotheses are:

- **H_0:** the mean of the assigned datasets in the cluster is the same as the mean of the going-to assigned dataset;
- **H_1:** the mean of the assigned datasets in the cluster is not the same as the mean of the going-to assigned dataset;

For the feature vector of a new dataset, as mentioned earlier it consists of T raw versions from T random trials to obtain the mean value: $\mathcal{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T\}$ where T is the number of trials. Similarly for datasets assigned in cluster V^k, we have $\mathcal{Y} = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_T\}$ where $\boldsymbol{y}_t$ is the average over the features of datasets in that cluster for trial t.

Unlike the Markov analysis for sequential clustering in the above section that the mean and covariance matrix are calculated by the max-likelihood estimation, in this hypothesis test, these parameters are assumed unknown. Specifically, the elements in $\mathcal{X}$ and $\mathcal{Y}$ are supposed to be Gaussian distributions: $N(\boldsymbol{\mu}^1, \boldsymbol{\Sigma}^1)$ and $N(\boldsymbol{\mu}^2, \boldsymbol{\Sigma}^2)$, respectively. The goal is to detect whether $\boldsymbol{\mu}^1 = \boldsymbol{\mu}^2$. Here, Behrens-Fisher solution [18] is introduced, consider another set for $t = 1, 2 \ldots, T$:

$$\mathcal{Z} = \left\{ \boldsymbol{z}_i | \boldsymbol{z}_i = \boldsymbol{x}_i + \frac{\sum_{t=1}^{T} \boldsymbol{y}_t / T - \boldsymbol{y}_i}{\sqrt{|V^k|}} - \frac{\sum_{t=1}^{T|V^k|} \boldsymbol{y}_t}{T|V^k|} \right\} \tag{15}$$

The test statistics [1] can be represented as:

$$F = \frac{1}{B} \cdot T(T-B) \overline{\boldsymbol{z}}^\top [\sum_{i=1}^{T} (\boldsymbol{z}_i - \overline{\boldsymbol{z}})(\boldsymbol{z}_i - \overline{\boldsymbol{z}})^\top]^{-1} \overline{\boldsymbol{z}} \tag{16}$$

Where B is the number of benchmark models described above, $\overline{\boldsymbol{z}} = \frac{\sum_{i=1}^{T} \boldsymbol{z}_i}{T}$. Given the standard statistical significance level α, the rejecting region is $\{F > F_\alpha(B, T-B)\}$.

The significance level α is the desired probability of making a Type I error (i.e. incorrectly rejecting H_0). Usually, decreasing its value will cause lower Type I error, but higher Type II error (i.e. incorrectly accepting H_0). In our method, the Markov analysis for sequential clustering can be viewed as a method to control Type II error, so the α can be chosen only to consider the Type I error. Different values of α will be evaluated in the experiments.

2.4. Approach Summary and Discussion

We term the proposed approach *Transferable AutoML (Tr-AutoML) using benchmarks' performance based meta-features as well as Markov analysis and hypothesis test (MH)* whose details are depicted in Algorithm 1. The term **Tr-AutoML** reflects the meta-features extracting method and the framework proposed in the paper; while **MH** emphasizes the techniques as described above.

Algorithm 1 Transferable AutoML with Markov analysis and hypothesis test – abbr. Tr-AutoML (MH)

```
Input:
1: Set significance level $\alpha$, $\gamma$ and the max cluster number $K$;
2: Initialize the state with Eq. 9 10 11 for Markov clustering;
3: Setup the $K$ empty clusters $\{V_0^k\}_{k=1}^K$ in the dataset feature
   space $\Omega_D$ with random initialization (some of them will be
   updated according to the assigned datasets).
Output:
4: The $J$ searched models $m^j$ for nonempty cluster $V^j \neq \emptyset$ in
   stream. Note that 1) datasets may share the same model with
   different weights; 2) benchmark models are only for dataset
   feature computing rather than used as searched models.
5: for dataset $d_n$, n=1,2,... do
6:     For $d_n$, compute its benchmark model specific feature vec-
       tor $\boldsymbol{x}_n = f(d_n, b_1, ..., b_B) \in \Omega_d$;
7:     Find $V_n^k$ and assign $d_n \rightarrow V_n^k$ via Eq. 5 6 7;
8:     if $V_n^k = \emptyset$ i.e. the found cluster is empty then
9:         //can be coupled by standalone model search methods
10:        Perform standalone model search method e.g. Hyper-
           band, MetaQNN to search tailored model $m_n$ for $d_n$;
11:        Assign $d_n$ to $V_n^k$; set $V_n^k$'s searched model $m_n^k = m_n$;
12:    else if $V_n^k \neq \emptyset$ then
13:        Perform hypothesis test via Eq. 16.
14:        if hypothesis is not accepted then
15:            Search model $m_n$ for $d_n$;
16:            Randomly choose an empty cluster $V_n^{k'}$ and assign $d_n$
               to $V_n^{k'}$; set $V_n^{k'}$'s searched model $m_n^{k'} = m_n$;
17:        else
18:            Set $m_n^k$ as $d_n$'s searched model for sharing;//model
               reused
19:        end if
20:    end if
21:    Update the parameters with maximum-likelihood estima-
       tion via Eq. 12 13 14.
22: end for
```

3. Experiments

The experiments consider three settings: **i) search model from scratch**; **ii) search model from a predefined model** e.g. GoogleNet to leverage existing architectures; **iii) transfer from basic cells**. We will first list the compared baselines and then introduce the general settings in terms of evaluation metrics, model search algorithms, testing dataset sequence generation and platform etc.

3.1. General Settings

3.1.1 Compared Methods

We consider these baselines to compare with Tr-AutoML:

i) Our meta-features extraction method for AutoML is based on benchmark models' performance. We will empirically show its advantage over traditional statistical and categorized meta-feature generating approaches [26].

ii) The proposed model sharing approach is agnostic to specific strategies for dataset grouping. Apart from the devised Markov analysis and hypothesis test (MH) methods, some simple baselines e.g. random grouping, k-means, sequential k-means [13] can also be combined. In the experiments, we will show the advantage of our MH technique over these baselines under the Tr-AutoML framework.

iii) The proposed Tr-AutoML framework can be viewed as collaborative AutoML that leverage the knowledge of previous datasets. In the experiments, we will compare other collaborative AutoML–warmstart algorithm [24]–to see the advantage of our Tr-AutoML.

iv) Our Tr-AutoML framework with the MH techniques can incorporate existing standalone AutoML methods e.g. Hyperband [22], Bayesian optimization [34], MetaQNN [3], NAS [42], Net2Net [10], ENAS [31] and DARTS [25] in an out-of-box fashion, as specified by the step at line 10 in Algorithm 1. In our experiments, we will also show the performance when different AutoML algorithms are used.

3.1.2 Evaluation Protocols

To mimic online setting, datasets are sequentially processed with random order for 30 times, and the reported results in the experiments are the average ones. For each dataset, tailored model (including architecture and hyper-parameters) is searched. The architecture involves numbers of layers, convolutional kernel size, output channel size, pool kernel size and stride etc. while hyper-parameters involve initial learning rate, initial weight standard deviation etc.

Experiment runs on two Tesla K80 each with 12G memory. We set the hyper-parameters of our Tr-AutoML method $\alpha = 0.005$, $\gamma = 0.2$, the maximum cluster number $K = 1000$, and $\sigma_0^2 = 3e-4$. We set the number of benchmarks $B = 6$ and number of random trials $T = 8$ to obtain performance mean $\boldsymbol{x}$. For evaluation we use search wall clock time and total classification relative errors (TRE) defined as:

$$TRE = \frac{1}{N}\sum_i \frac{e_i - e_i^*}{e_i^*} \tag{17}$$

where e_n^* and e_n stand for the test set error of dataset n using the model generated by standalone search method and combined one, respectively. N is the total number of datasets.

3.1.3 Benchmark Models

Our method is based on feature representation of the dataset by benchmark model evaluation. Here six benchmark models combining hyper-parameter and structure of neural networks are chosen from Monte Carlo sampling over the feasible region from uniform or log-uniform distribution. It is important to note that the benchmark models are only used for computing dataset d_n's performance vector $\boldsymbol{x}_n$. The final searched model is tailored for the specific dataset.

3.2. Search from Scratch

To verify the generality, here the search is performed without any prior knowledge about the model and tested dataset. Seven datasets are used for evaluation including MNIST, CIFAR-10, FASHION-MNIST, SVHN, STL-10, MNIST-BACKGROUND-IMAGES, MNIST-ROTATED [19, 27, 11, 21, 38]. Each dataset has further another derived version with sampling ratio 50% of raw datasets. Splitting dataset into sub-dataset is widely used in multi-task learning and transfer learning [26, 35]. Hence in total there are 14 datasets for online model search. The datasets are processed with random orders for 30 trials.

Table 1: Dataset grouping examples by Tr-AutoML. Groups are formed for each trial: in model search from scratch.

1) mnist_1.0, mnist_0.5, svhn_0.1, svhn_0.5, fashion-mnist_0.1, fashion-mnist_0.5
2) stl10_0.5, stl10_1.0
3) mnist-background-images_0.5, mnist-background-images_1.0, mnist-rotated_0.5, mnist-rotated_1.0
4) cifar10_1.0, cifar10_0.5
1) mnist_1.0, mnist_0.5, fashion-mnist_0.5
2) stl10_1.0, cifar10_0.5, cifar10_1.0
3) mnist-rotated_0.5, mnist-background-images_1.0, mnist-background-images_0.5, mnist-rotated_1.0
4) svhn_1.0, svhn_0.5, fashion-mnist_0.1
5) stl10_0.5

We empirically find that in most trials, the MNIST-BACKGROUND-IMAGES and MNIST-ROTATED are automatically grouped by the Tr-AutoML approach. In a few cases, STL-10 and CIFAR-10, FASHION-MNIST/MNIST and SVHN also tend to be grouped together. Table 1 shows the dataset grouping results of two of these trials whereby similar datasets are grouped.

To compare our proposed Tr-AutoML with standalone model search schemes, we combine Tr-AutoML with Hyperband [22], MetaQNN [3] and ENAS [31]. Also, to demonstrate the efficiency of our proposed meta-features extracting method, we compare Tr-AutoML with [26] that uses its statistical meta-features (such as statistics about the number of data points, features, and classes, as well as data skewness, and the entropy of the targets) as meta-learning for dataset feature representations. To show the performance of our proposed Markov analysis and hypothesis test (MH) method, grouping baselines by k-means, sequential k-means and random clustering are evaluated. Furthermore, to compare the collaborative way for AutoML, we compare Tr-AutoML with a warmstart method: [24] by using the three AutoML methods as its user-specified default initialization. Table 2 shows the results, where the total search time already includes the overhead of running benchmark models listed in the last column. Comparing to the standalone model search methods, the combined one Tr-AutoML (MH) can reduce the search time by 3 to 4 times on average while maintaining a low level extra error almost the same as k-means (k is selected as the number of raw datasets), which is much less than Warmstart, Meta-learning and random clustering based methods. Tr-AutoML (MH) is more efficient than k-means (save about 40%-70% search time) because it can find the reusable models among the datasets even from different raw datasets.

The total relative error for combination with Hyperband

Table 2: Total search time (in days, including the overhead of running benchmark models), total classification relative errors (TRE) and benchmark overhead on 7 dataset: in model search from scratch setting.

Techniques combination	Total search time	TRE	Overhead
Hyperband [22]	10.40	0	0
Warmstart [24] + Hyperband	6.23	0.412	0
Meta-learning [26] + Hyperband	3.85	0.118	0
Tr-AutoML(Random) + Hyperband	2.96	1.653	0
Tr-AutoML(Kmeans) + Hyperband	5.44	0.059	0.2
Tr-AutoML(Seq. Kmeans [13]) + Hyperband	4.48	0.061	0.2
Tr-AutoML(MH) + Hyperband	3.17	0.062	0.2
MetaQNN [3]	16.29	0	0
Warmstart [24] + MetaQNN	7.19	0.276	0
Meta-learning [26] + MetaQNN	5.46	0.075	0
Tr-AutoML(Random) + MetaQNN	4.68	1.149	0
Tr-AutoML(Kmeans) + MetaQNN	7.87	0.036	0.2
Tr-AutoML(Seq. Kmeans [13]) + MetaQNN	6.32	0.041	0.2
Tr-AutoML(MH) + MetaQNN	4.85	0.039	0.2
ENAS [31]	12.22	0	0
Warmstart [24] + ENAS	5.10	0.132	0
Meta-learning [26] + ENAS	4.82	0.044	0
Tr-AutoML(Random) + ENAS	4.02	0.471	0
Tr-AutoML(Kmeans) + ENAS	6.17	0.017	0.2
Tr-AutoML(Seq. Kmeans [13]) + ENAS	5.04	0.019	0.2
Tr-AutoML(MH) + ENAS	4.22	0.019	0.2

is higher than the other two methods. We conjecture this is because in Hyperband, besides the architecture, the hyperparameters like learning rate, initial weight standard deviation are also auto-tuned simultaneously, making the shared model more sensitive to different datasets. The MetaQNN implemented in our experiments underperforms its original report in [3] as we set a time budget on it. But the comparison of standalone MetaQNN and combined MetaQNN scheme is not affected since we focus on the relative performance in the paper. Moreover, the overheads of running benchmarks are similar among three AutoML schemes because the benchmark models are the same among them.

Figure 1 shows the test accuracy over wall clock time on six raw datasets (due to space constraints, we report results for only six datasets). Hyperband is used to demonstrated the standalone AutoML, other AutoML schemes are similar. From the average searching time point of view, the accuracy grows quickly in transferable AutoML (Tr-AutoML); regarding with limited performance, pure standalone AutoML performs slightly better than Tr-AutoML. This is because Tr-AutoML reuses the model directly and makes no feedback action.

Figure 2 shows the effect of different benchmark numbers and different significance levels in hypothesis test. Large significance levels will force the dataset groups to shrink to smaller one, so it usually provides low error but a long search time. However, when the benchmark number is too small to capture the divergence of different datasets, both errors and search time are poor. In our experiments, selecting 4 to 6 benchmark models is good enough for Tr-AutoML when considering model search from scratch.

Table 3: Two examples of dataset grouping generated by running two trials of Tr-AutoML: in predefined setting.

1) scene_1.0, scene_0.5, flower_1.0, flower_0.5
2) action_1.0, action_0.5
3) sun_1.0, sun_0.5
1) scene_1.0, scene_0.5, flower_1.0, flower_0.5, action_1.0
2) action_0.5
3) sun_1.0, sun_0.5

Table 4: Searching time (in days) and total classification relative errors on four datasets: in model search from predefined model setting.

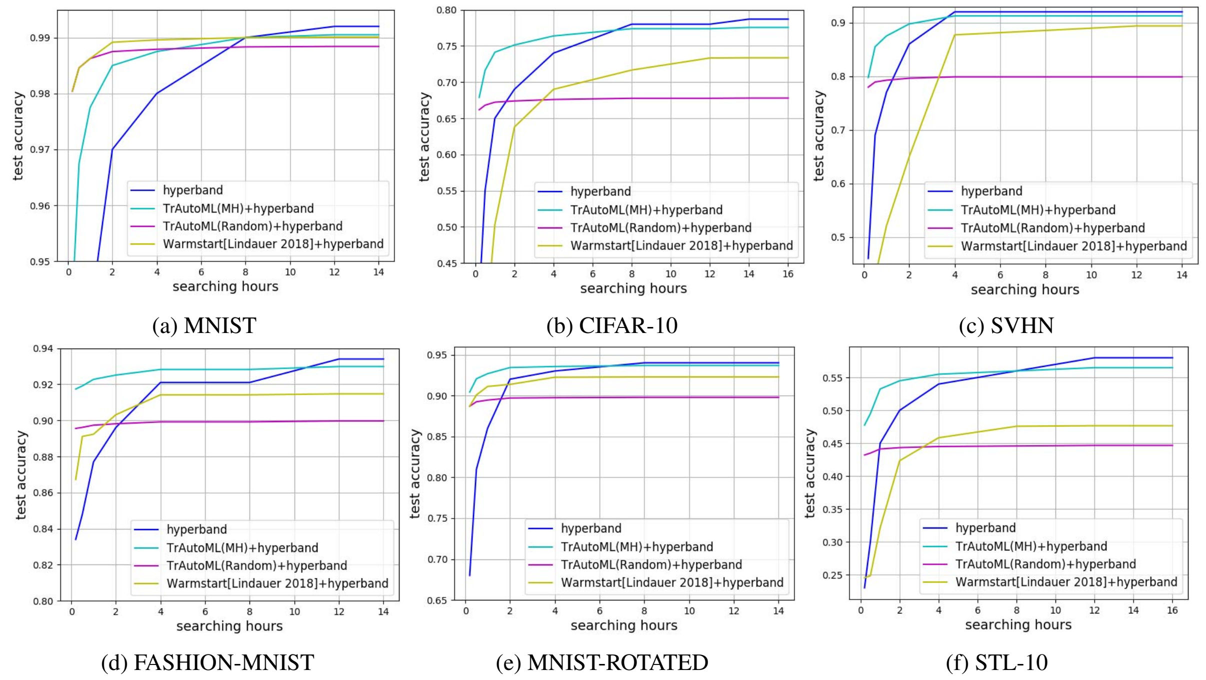
Algorithm	Time	TRE
MetaQNN	8	0
Tr-AutoML(Random) + MetaQNN	2.8	1.78
Tr-AutoML(Kmeans) + MetaQNN	4.5	0.069
Tr-AutoML(MH) + MetaQNN	3.0	0.074

3.3. Search from Predefined Model

In real-world applications, the image classification tasks are far more difficult than MNIST or CIFAR-10. Shallow networks are too weak to capture the high-level informations but training several deep networks on those tasks are time-consuming. In this case, our Tr-AutoML can also perform well on search from the predefined model or transferring from basic cells (in Sec.3.4). In this section, Tr-AutoML from a predefined model is evaluated. We involve four complicated datasets of large image size: FLOWER-102 [28], ACTION-40 [40], SCENE-15 [29], SUN-397 [39] along with their subsets of 0.5 sample rate version. The difference from the previous experiment is that we reuse GoogleNet [36] architecture and weights from bottom to inception (4e) layer, while search a chain-structure for higher layers. The dataset grouping results is shown in Table 3. One can find the SCENE-15 and FLOWER-102 datasets can be automatically combined due to their similarity. The performance comparison is shown in Table 4 whereby one can see our approach achieves a reasonable tradeoff for efficiency and efficacy.

3.4. Transfer from Basic Cells

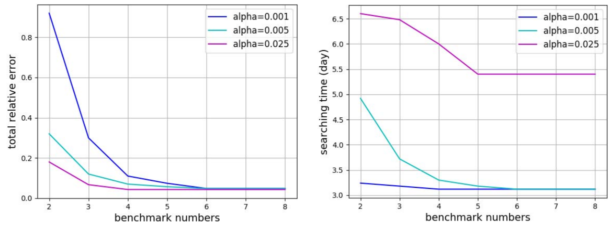
Searching model based on basic cells are widely used [31, 25], which can be viewed as a layered search strategy. AutoML algorithms focus on searching the nodes and their connections in the cells (normal cell and reduction cell), and then concatenate them to be a neural network. In [43] the authors show the basic cell searched on CIFAR-10 can be transferred to ImageNet classification without much modification. By adopting our proposed Tr-AutoML, it is intrinsic to share the basic cells within a dataset group. In this section, we evaluate three transferable cases: 1) source datasets



(a) MNIST (b) CIFAR-10 (c) SVHN

(d) FASHION-MNIST (e) MNIST-ROTATED (f) STL-10

Figure 1: Test Accuracy over wall clock time (searching hours) on six raw datasets: in model search from scratch setting.



(a) Total relative error (b) Searching time

Figure 2: Different significance levels α by different numbers of benchmarks: in model search from scratch setting.

are MNIST, CIFAR10, STL10 and SVHN, target dataset is FASHION-MNIST. 2) source datasets are MNIST, CIFAR10, FASHION-MNIST and SVHN, target dataset is STL10. 3) source datasets are MNIST, CIFAR10, STL10, SVHN and FASHION-MNIST, target dataset is ImageNet. We set $\alpha \rightarrow 0$ and $\gamma \rightarrow \infty$ in order to force Tr-AutoML to share models. From the Table 5, it is shown that for both easy datasets and complicated datasets, Tr-AutoML identify which datasets can share the best basic cell stably and efficiently. This is because the proposed dataset feature representations are based directly on model shared performance rather than other statistical or categorized features.

4. Conclusion

This paper explores the transferable AutoML for model search and sharing over sequentially arriving datasets. We propose a novel meta-learning approach by adaptively grouping the new dataset into previous ones, and reuse the previously discovered models to the model search for new data. Meanwhile, our framework is orthogonal to, and can be coupled with many existing AutoML techniques in an out-of-box fashion. On the image classification task, our method achieves notable overall speedup for model searching at negligible accuracy loss. It also works well on different search mechanisms and datasets.

Table 5: Search time (in days, including overhead). Test accuracy: in transfer from basic cells setting.

Target dataset	Techniques	Search time	Accuracy
FASHION-MNIST	Hyperband [22]	1.67	0.942
	Meta-learning [26]	0.001	0.939
	Tr-AutoML(Random)	0	0.936
	Tr-AutoML(MH)	0.013	0.939
STL10	ENAS [31]	1.08	0.734
	Meta-learning [26]	0.001	0.680
	Tr-AutoML(Random)	0	0.692
	Tr-AutoML(MH)	0.017	0.725
ImageNet	NASNet-A [43]	1800	0.740
	Meta-learning [26]	2.56	0.717
	Tr-AutoML(Random)	2.54	0.712
	Tr-AutoML(MH)	2.81	0.734

References

[1] T. Anderson. An introduction to multivariate statistical analysis, 3rd edition. 2003. 4, 5

[2] N. Asai, I. Kubo, and H. Kuo. Bell numbers, log-concavity, and log-convexity. *Acta Applicandae Mathematica*, 2000. 4

[3] B. Baker, O. Gupta, N. Naik, and R. Raskar. Designing neural network architectures using reinforcement learning. In *ICLR*, 2017. 1, 2, 6, 7

[4] R. Bardenet, M. Brendel, B. Kegl, and M. Sebag. Collaborative hyperparameter tuning. In *ICML*, 2013. 1, 2

[5] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kegl. Algorithms for hyper-parameter optimization. In *NIPS*, 2011. 1, 3

[6] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 2012. 1

[7] J. Bergstra, D. Yamins, and D. Cox. Making a science of model search: Hyper-parameter optimization in hundreds of dimensions for vision architectures. In *ICML*, 2013. 1

[8] E. Bonilla, K. Chai, and C. Williams. Multi-task gaussian process prediction. In *NIPS*, 2008. 1, 2

[9] O. Cappe and E. Moulines. Online em algorithm for latent data models. *arXiv preprint arXiv:0712.4273*, 2007. 4

[10] T. Chen, I. Goodfellow, and J. Shlens. Net2net: Accelerating learning via knowledge transfer. In *ICLR*, 2016. 2, 6

[11] A. Coates, H. Lee, and A. Ng. An analysis of single-layer networks in unsupervised feature learning. In *AISTATS*, 2011. 6

[12] T. Domhan, J. Springenberg, and F. Hutter. Speeding up automatic hyper-parameter optimization of deep neural networks by extrapolation of learning curves. In *IJCAI*, 2015. 1

[13] G. Hamerly and C. Elkan. Learning the k in k-means. In *NIPS*, 2003. 4, 6, 7

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1

[15] G. Huang, Z. Liu, L. Maaten, and K. Weinberger. Densely connected convolutional networks. *arXiv preprint arXiv:1608.06993*, 2016. 1

[16] F. Hutter, H. Hoos, and K. Brown. Sequential model-based optimization for general algorithm configuration. In *LION*, 2011. 1

[17] A. Kalousis. Algorithm selection via meta-learning. *PhD thesis*, 2002. 3

[18] S. Kim and A. Cohen. On the behrens-fisher problem: A review. *Journal of Educational and Behavioral Statistics*, 1998. 5

[19] A. Krizhevsky. Learning multiple layers of features from tiny images. 2009. 6

[20] B. Kulis and M. I. Jordan. Revisiting k-means: New algorithms via bayesian nonparametrics. *arXiv preprint arXiv:1111.0352*, 2011. 4

[21] H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the 24th international conference on Machine learning*, pages 473–480. ACM, 2007. 6

[22] L. Li, K. Jamieson, and G. DeSalvo. Hyperband: bandit-based configuration evaluation for hyper-parameter optimization. In *ICLR*, 2017. 1, 2, 6, 7, 8

[23] P. Liang and D. Klein. Online em for unsupervised models. *Proceedings of human language technologies*, 2009. 4

[24] M. Lindauer and F. Hutter. Warmstarting of model-based algorithm configuration. In *AAAI*, 2018. 1, 3, 6, 7

[25] H. Liu, K. Simonyan, and Y. Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018. 1, 2, 6, 7

[26] K. E. e. a. M. Feurer, A. Klein. Efficient and robust automated machine learning. In *NIPS*, 2015. 1, 3, 5, 6, 7, 8

[27] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 5, 2011. 6

[28] M.-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *Computer Vision, Graphics & Image Processing, 2008. ICVGIP'08. Sixth Indian Conference on*, pages 722–729. IEEE, 2008. 7

[29] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International journal of computer vision*, 42(3):145–175, 2001. 7

[30] V. Perrone, R. Jenatton, M. Seeger, and C. Archambeau. Multiple adaptive bayesian linear regression for scalable bayesian optimization with warm start. *arXiv preprint arXiv:1712.02902*, 2017. 1, 2

[31] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean. Efficient neural architecture search via parameter sharing. *arXiv preprint arXiv:1802.03268*, 2018. 2, 6, 7, 8

[32] E. Real, S. Moore, A. Selle, S. Saxena, Y. Suematsu, J. Tan, Q. Le, and A. Kurakin. Large-scale evolution of image classifiers. In *ICML*, 2017. 1, 2

[33] D. Schaffer, D. Whitley, and L. Eshelman. Combinations of genetic algorithms and neural networks: A survey of the state of the art. In *International Workshop on Combinations of Genetic Algorithms and Neural Networks*, 1992. 1

[34] J. Snoek, H. Larochelle, and R. Adams. Practical bayesian optimization of machine learning algorithms. In *NIPS*, 2012. 1, 6

[35] K. Swersky and J. Snoek. Multi-task bayesian optimization. In *NIPS*, 2013. 1, 2, 3, 6

[36] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015. 7

[37] C. Wong, N. Houlsby, Y. Lu, and A. Gesmundo. Transfer automatic machine learning. *arXiv preprint arXiv:1803.02780*, 2018. 2

[38] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017. 6

[39] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on*, pages 3485–3492. IEEE, 2010. 7

[40] B. Yao, X. Jiang, A. Khosla, A. L. Lin, L. Guibas, and L. Fei-Fei. Human action recognition by learning bases of action attributes and parts. In *Computer Vision (ICCV), 2011 IEEE*

International Conference on, pages 1331–1338. IEEE, 2011. 7

[41] S. Zagoruyko and N. Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016. 1

[42] B. Zoph and Q. Le. Neural architecture search with reinforcement learning. In *ICLR*, 2017. 1, 2, 6

[43] B. Zoph, V. Vasudevan, J. Shlens, and Q. Le. Learning transferable architectures for scalable image recognition. *arXiv preprint arXiv:1707.07012*, 2017. 2, 7, 8