AutoFCL: Automatically Tuning Fully Connected Layers for Handling Small Dataset

S.H.Shabbeer Basha, Sravan Kumar Vinakota, Shiv Ram Dubey, Viswanath Pulabaigari, Snehasis Mukherjee
Indian Institute of Information Technology Sri City, India.

***Abstract*— Deep Convolutional Neural Networks (CNN) have evolved as popular machine learning models for image classification during the past few years, due to their ability to learn the problem-specific features directly from the input images. The success of deep learning models solicits architecture engineering rather than hand-engineering the features. However, designing state-of-the-art CNN for a given task remains a non-trivial and challenging task, especially when training data size is less. To address this phenomena, transfer learning has been used as a popularly adopted technique. While transferring the learned knowledge from one task to another, fine-tuning with the target-dependent Fully Connected (FC) layers generally produces better results over the target task. In this paper, the proposed AutoFCL model attempts to learn the structure of FC layers of a CNN automatically using Bayesian optimization. To evaluate the performance of the proposed AutoFCL, we utilize five pre-trained CNN models such as VGG-16, ResNet, DenseNet, MobileNet, and NASNetMobile. The experiments are conducted on three benchmark datasets, namely CalTech-101, Oxford-102 Flowers, and UC Merced Land Use datasets. Fine-tuning the newly learned (target-dependent) FC layers leads to state-of-the-art performance, according to the experiments carried out in this research. The proposed AutoFCL method outperforms the existing methods over CalTech-101 and Oxford-102 Flowers datasets by achieving the accuracy of 94.38% and 98.89%, respectively. However, our method achieves comparable performance on the UC Merced Land Use dataset with 96.83% accuracy. The source codes of this research are available at https://github.com/shabbeersh/AutoFCL.**

I. Introduction

Deep Convolutional Neural Networks (CNN) based features have outperformed the hand-designed features in most of the computer vision problems such as object recognition [1], [2], speech recognition [3], medical applications [4], and many more. Although several complicated research problems have been solved by deep learning models, generally, the performance of these models relies on hard-to-tune hyperparameters. Finding the best configuration for the hyperparameters such as the number of layers, convolution filter dimensions, number of filters in a convolution layer, and many more to build a CNN architecture suitable for a given task, is the most demanding research theme in the area of Automated Machine Learning (AutoML) [5], [6]. Based on the previous studies reported in the literature, learning a suitable architecture for a given task is termed as Neural Architecture Search (NAS) [7]. Reinforcement Learning (RL) methods have been widely employed to find the suitable CNN architecture for given task [8]. However, these methods are focused to find the structure of CNN from scratch which requires hundreds of GPU hours. We propose a method called AutoFCL to automatically tune the structure of the Fully Connected (FC) layers with respect to the target dataset while transferring the knowledge from source task to the target task.

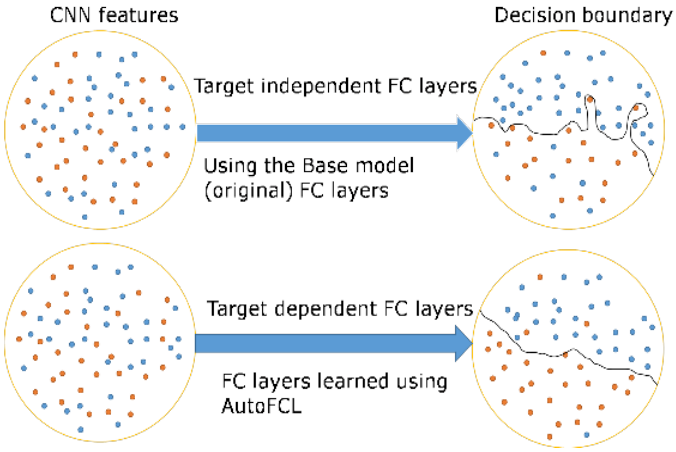

Fig. 1. While transferring the knowledge learned from source task to the target task, learning the optimal structure of FC layers with the knowledge of target dataset and fine-tuning the learned FC layers leads to better performance.

Typically, every CNN contains one or more FC layers based on the depth of the architecture [9]. For instance, the popular CNN models proposed to train over large-scale ImageNet dataset [10] have the following number of FC layers.

- AlexNet [1], ZFNet [11], and VGG-16 [12] have 3 dense (FC) layers. Note that these models contain 5, 5, and 13 convolution layers, respectively.
- GoogLeNet [2], ResNet [13], DenseNet [14], NASNet [5], and other modern deep neural networks have a single FC layer which is responsible for generating the class scores.

The CNN models [1], [11], [12] introduced in the initial years (during the years from 2012 to 2014) have a huge number of trainable parameters in FC layers. Whereas the recent models [13], [14], [5] are generally deeper, and hence, have a single FC layer which is responsible for generating the class scores. The state-of-the-art CNN architectures proposed for ImageNet dataset are shown in Table I. This table summarizes the total number of trainable parameters and also the trainable parameters correspond to FC layers. It is evident from Table I that as the depth of CNN increases, both the number of dense layers and the parameters in dense layers gradually decrease.

A large number of parameters involved in the FC layers

Please cite this article as: Basha, S.H.S., Vinakota, S.K., Dubey, S.R. et al. AutoFCL: automatically tuning fully connected layers for handling small dataset. Neural Comput & Applic (2021). https://doi.org/10.1007/s00521-020-05549-4.

of a CNN increases the possibility of overfitting. Xu *et al*. [15] shown that removing the connections among FC layers having less weight magnitude (SparseConnect) leads to better performance. Basha *et al*. [9] performed a study to observe the necessity of FC layers given the depth and width of both datasets and CNN architectures. To find the best set of hyperparameters of an Artificial Neural Network (ANN), Mendoza *et al*. [16] proposed an automated mechanism to tune the ANN using Sequential Model-based Algorithm Configuration (SMAC).

CNNs are used in a wide range of applications in recent years. However, their performance is poor if the amount of training data is very limited. Transfer Learning is a way to reduce the need for more training data and huge computational resources by reusing the knowledge learned from the source task. A common approach for classifying such limited images is re-using the pre-trained models to fine-tune over other datasets [19]. However, while transferring the learned knowledge from one task to another, fine-tuning the original FC layers' structure may not perform well over the target dataset because the FC layers are designed for the source task.

Fig. 1 illustrates the motivation behind learning the target-dependent fully connected layer's structure to obtain better performance over the target task. While transferring the learned knowledge from source task to target task, the efficacy (capacity) of the CNN increases for the target task, which may result in overfitting. The extracted features from convolutional layers (shown in the left side) are mapped into more linearly separable feature space (shown on the right side) by FC layers. Moreover, we believe that learning the FC layers' structure with the knowledge from the target dataset may lead to better linearly separable feature space which results in better performance over the target dataset. In this work, we propose a novel framework for automatically learning the target-dependent fully connected layers structure in the context of transfer learning. We use Bayesian optimization [20] for optimizing the hyperparameters involved in forming the FC layers while transferring the knowledge from one task to another.

II. Related Works

Due to the dense connectivity among the FC layers, the deep CNNs contain an enormous amount of trainable parameters. For example, the first ImageNet Large Scale Visual Recognition Competition (ILSVRC)-2012 [21] winning CNN model called AlexNet [1] contains a total of 60 million trainable parameters, among which 58 million parameters belong to the FC layers. Likewise, VGG-16 [12], a 16 layer deep CNN comprises 138 million trainable parameters, among which 123 million parameters correspond to FC layers. In practice, the over-parameterization leads to overfitting the CNN. Xu *et al*. [15] proposed SparseConnect model to reduce the overfitting effect by removing the connections with smaller weight values.

Transfer learning is a widely adopted technique to obtain a reasonable performance with limited data and less computational resources. Li *et al*. [22] analyzed various approaches for transferring the knowledge learned in different scenarios. Fine-tuning the deep CNNs with limited training data often leads to overfitting the CNN model [23]. Han *et al*. [24] introduced a two-phase strategy by combining transfer learning with web data augmentation to reduce the amount of over-fitting. They also tuned the hyperparameters such as learning rate, type of optimizer (Adagrad [25], Adam [26], etc.) and many more using Bayesian Optimization.

Mendoza *et al*. [16] proposed Auto-Net, which automatically tunes an artificial neural network without any human intervention. To learn a distinct set of hyperparameters automatically, they used the Sequential Model-based Algorithm Configuration (SMAC). The hyperparameters such as the number of FC layers, number of neurons in each FC layer, batch size, learning rate, and so on are tuned automatically. Motivated by this work, we propose a framework to automatically learn the structure of FC layers concerning the target dataset for better transfer learning.

Many researchers have employed Bayesian Optimization [20] to learn the entire CNN architecture automatically. Wistuba *et al*. [27] combined Bayesian Optimization with Incremental Evaluation to find the optimal neural network architecture. However, they limited the depth of the CNN to 5 layers due to the limited computational resources. Jin *et al*. [28] proposed a network morphism mechanism for neural architecture search using Bayesian Optimization. Liu *et al*. [6] proposed a method to build the CNN architecture progressively using the Sequential Model-Based Optimization (SMBO) based algorithm. However, these methods require a considerable amount of computational resources and search time. Recently, Gupta *et al*. [29] employed Bayesian Optimization to conduct a study for efficient transfer optimization.

Transfer Learning allows the pre-trained networks to adopt for the new tasks [30]. Many researchers utilized the advantage of transfer learning for various applications [19], [31]. Ji *et al*. [28] proposed a framework called Double Reweighting Multi-source Transfer Learning (DRMTL) to utilize the decision knowledge from multiple sources to perform well over the target domain. Generally, after adaptation, the efficacy (capacity) of the CNN increases for the target task. Molchanov *et al*. [32] proposed a framework for iteratively pruning the parameters of a CNN to reduce its capacity for the target task. With regard to our knowledge, no effort has been made in the literature to learn the structure of FC layers automatically for better transfer learning. Neural Architecture Search algorithms consume thousands of GPU hours [5] to find better performing architectures. So, we made this attempt in the context of transfer learning to reduce the architecture search time.

Basha *et al*. [9] analyzed the necessity of FC layers based on the depth of a CNN. However, to conduct this study they performed experiments by adding new FC layers manually before the output FC layer. Moreover, the hyperparameters involved in FC layers like the number of neurons in every FC layer, the dropout factor, type of activation, and so on

TABLE I

THE STATE-OF-THE-ART DEEP NEURAL NETWORKS PROPOSED FOR THE IMAGENET DATASET, THE TOTAL NUMBER OF TRAINABLE PARAMETERS AND THE NUMBER OF PARAMETERS BELONG TO FC LAYERS ARE SHOWN.

S.No.	CNN Model	Total #trainable parameters (in Millions)	#parameters in FC layers (in Millions)
1	AlexNet [1]	60 M	58 M
2	ZFNet [11]	62.3 M	58.6 M
3	VGG16 [12]	138.3 M	123.6 M
4	VGG19 [12]	143.6 M	123.6 M
5	InceptionV3 [17]	23.8 M	2 M
6	ResNet50 [13]	25.5 M	2 M
7	MobileNet [18]	4 M	1 M
8	DenseNet201 [14]	20 M	1.9 M
9	NASNetLarge [5]	88 M	4 M
10	NASNetMobile [5]	5 M	1 M

are chosen manually. In this paper, we attempt to learn the target-dependent FC layers' structure automatically for better transfer learning.

In brief, our contributions in this work are as follows,

- We propose a novel method to automatically learn the target-dependent FC layers structure using Bayesian Optimization.
- By conducting experiments on three benchmark datasets, we discover the suitable (target-dependent) FC layers structure specific to the datasets.
- The performance of the proposed method is also compared with state-of-the-art transfer learning and non-transfer learning-based methods.
- To compare the results obtained using Bayesian Optimization, we employed the random search to find the best set of hyperparameters involved in FC layers.

III. PROPOSED AUTOFCL MODEL

We formulate the task of learning the structure of fully connected layers as a black-box optimization problem. Let f is an objective function whose objective is to find x_*, which is represented as

$$x_* = \underset{x \in \mathcal{H}}{\operatorname{argmax}}\, f(x) \tag{1}$$

where $x \in \mathbb{R}^d$ is the input, usually $d \leq 20$ [20], $\mathcal{H}$ is the hyperparameter space as depicted in Table II, and f is a continuous function. Finding the value of function f at x requires training (fine-tuning) the learned FC layers (explored during the architecture search) of a pre-trained CNN (B) on training data (Train_{Data}) and evaluating its performance on the held-out (validation) data Val_{Data}.

The x_* is a CNN with an optimal FC layer's structure learned using the Bayesian Optimization for efficient transfer learning. Therefore, the CNN architecture x_* is responsible for maximizing the performance on the Val_{Data}. The proposed AutoFCL method is outlined in Algorithm 1. Given the base CNN model (B), hyperparameters search space (Param_space), Train_{Data}, Val_{Data}, and the number of epochs (E) to train each proxy CNN as an input, the proposed method learns the most suitable structure of FC (dense) layers using Bayesian Optimization [20].

The Bayesian Optimization (Bayes Opt) is the most popular method used for finding the best set of hyperparameters involved in deep neural networks [33]. Bayes Opt builds a surrogate model to approximate the objective function using Gaussian Process (GP) regression [34]. Algorithm 1 observes the value of f without noise for initial n_0 points which are chosen uniformly random (n_0 is 20 in our experimental settings). After observing the objective at initial n_0 points, we can infer the objective value at a new point x_{new} using Bayes rule [35] as follows,

$$f(x_{new})|f(x_{1:n_0}) \sim Normal(\mu_{n_0}(x_{new}), \sigma^2_{n_0}(x_{new})) \tag{2}$$

The $\mu_{n_0}(x_{new})$ and $\sigma^2_{n_0}(x_{new})$ are computed as follows,

$$\begin{aligned}\mu_{n_0(x_{new})} = \sum_0 (x_{new} \\ : x_{1:n_0}) \sum_0 (x_{1:n_0}, x_{1:n_0})^{-1} (f(x_{1:n_0}) - \mu_0(x_{1:n_0})) \\ + \mu_0(x_{new})\end{aligned} \tag{3}$$

$$\begin{aligned}\sigma^2_{n_0}(x_{new}) = \sum_0 (x_{new}, x_{new}) \\ - \sum_0 (x_{new}, x_{1:n_0}) \sum_0 (x_{1:n_0}, x_{1:n_0})^{-1} \sum_0 (x_{1:n_0}, x_{new})\end{aligned} \tag{4}$$

The probability distribution given in Eq. 2 is called posterior probability distribution. In the above equations, μ_0 , $\sum_0$ are mean function and covariance functions, respectively.

The optimal configuration of FC layers is one among the previously evaluated points (initial n_0 points) with the maximum f value ($f(x^+)$). Now, if we want to evaluate the value of objective 'f' at a new point x_{new}, which is observed as $f(x_{new})$. After evaluating the value of f at iteration $n_0 + 1$, the optimal f value will be either $f(x_{new})$ (if $f(x_{new}) \geq f(x^+)$) or $f(x^+)$ (if $f(x^+) \geq f(x_{new})$). The improvement or gain in the objective f is $f(x_{new}) - f(x^+)$ if its value is positive, or 0 otherwise.

However, the f(x_{new}) value is unknown until observing its value at x_{new} which is typically expensive. Instead of

Algorithm 1 AutoFCL: A Bayesian Search method for automatically learning the structure of FC layers

Inputs: B (Base Model), Param_space, Train$_{Data}$, Val$_{Data}$, E (num epochs).
Output: A CNN with target-dependent FC layers structure.

procedure AUTOFCL
  Place a Gaussian Process (GP) prior on the objective f
  while $t \in 1, 2, ...n_0$ **do** ▷ Observe the value of f at initial n_0 points
    $M_t \leftarrow build\_CNN(B, Param\_space)$ ▷ sample the initial CNN randomly
    $T_t \leftarrow Train\_CNN(M_t, Train_{Data}, E)$
    $V_t \leftarrow Validate\_CNN(T_t, Valid_{Data})$
  $n = n_0$
  while $t \in n + 1, .., N$ **do**
    Update the posterior distribution on f using the prior ▷ Using Eq. 2
    Choose the next sample x_t that maximizes the acquisition function value
    Observe $y_t = f(x_t)$
  return x_t ▷ return a point with best FC layer structure

evaluating f at x_{new}, we can compute the Expected Improvement (EI) and choose the x_{new} that maximizes the value of EI. Expected Improvement [36] is the most commonly used acquisition function for guiding the search process by proposing the next point to sample.

For a specified input x_{new}, EI can be represented as,

$$EI(x) = \mathbb{E}[max(f(x_{new}) - f(x^+), 0)] \tag{5}$$

where $f(x^+)$ is the maximum validation accuracy obtained so far and x^+ is the FC layer's structure for which best validation accuracy is obtained. Formally, x^+ can be represented as,

$$x^+ = \underset{x_i \in x_{1:n_0}}{\operatorname{argmax}} f(x_i) \tag{6}$$

which utilizes the information about the models that were already explored and finds the next point that maximizes the expected improvement. After observing the objective at each point, we update the posterior distribution using the Eq. 2.

IV. Hyperparameter Search Space

This section provides a detailed discussion about the search space used for finding the target-dependent FC layer's structure for efficient transfer learning. A single fully connected layer of a CNN involves various hyperparameters. To mention a few, the number of neurons, dropout rate, and many more. The proposed AutoFCL aims to learn the suitable structure for the FC layers, which includes the number of FC layers, dropout rate, type of activation, and the number of neurons in each FC layer to obtain the better performance over the target dataset. Table II shows the hyperparameter search space considered in our experimental settings.

As most of the CNN architectures available in the literature have a maximum of 3 FC layers [1], [12] including the output layer. Therefore, we consider the search space for the number of FC layers in the range [1,3] (i.e., 1, 2, and 3). The other important hyperparameter is the number of neurons required in each FC layer, for which the proposed method finds the best set of configuration within the range $[64, 1024]$ in powers of 2 ($\{64, 128, 256, 512, 1024\}$). Besides these hyperparameters, we consider activation function as another hyperparameter. Three popular non-linear activations Sigmoid, Tanh, and ReLu are utilized for the same. To reduce the over-fitting caused due to a large number of trainable parameters in FC layers, dropout [1] is widely adopted in deep learning. We consider dropout as another hyperparameter to learn, the value of which is learned in the range [0, 0.5] with an offset 0.1 i.e., the proposed AutoFCL finds the suitable dropout factor within the values $\{0.0, 0.1, 0.2, 0.3, 0.4, 0.5\}$.

V. Experimental Settings

In this section, we brief the training details, CNN architectures utilized to learn the structure of FC layers and the datasets used to evaluate the performance of the developed image classification models in the context of transfer learning.

A. Training Details

Training Proxy CNNs: The CNN architectures generated in the search process of Bayesian Optimization (also called proxy CNNs) are trained using AdaGrad optimizer [25]. The initial value of the learning rate is set to 0.01 and its value is reduced by a factor of $\sqrt{0.1}$ for every 5 epochs if there is no reduction in the validation loss. Since training the CNNs is a time-consuming task, we train each proxy CNN for 20 epochs as in [6]. Batch Normalization [37] is used after employing dropout. The suitable dropout rate is learned using Bayesian Optimization. The parameters (weights) corresponding to the FC layers are initialized using He normal initialization [38].

B. CNN Architectures used for Fine-Tuning

To learn the target-dependent FC layers structure automatically, we use two kinds of CNN architectures which include i) chain structured (plain) CNNs like VGG-16 [12] and ii) CNNs involving skip connections like ResNet [13], DenseNet [14], and many more.

TABLE II

HYPERPARAMETER SEARCH SPACE CONSIDERED IN THIS PAPER, WHICH INCLUDES BOTH NETWORK HYPERPARAMETERS SUCH AS THE NUMBER OF FULLY CONNECTED LAYERS AND PER-LAYER HYPERPARAMETERS LIKE ACTIVATION FUNCTION, DROPOUT FACTOR, AND THE NUMBER OF NEURONS ARE PRESENTED IN THIS TABLE.

	Name	Values	Type
Network hyperparameters	number of FC layers	{1,2,3}	integer
Hyperparameters per single FC layer	activation function	{ReLu, Tanh, Sigmoid}	categorical
	dropout rate	{0.0, 0.1, 0.2, 0.3, 0.4, 0.5}	float
	number of neurons	{64, 128, 256, 512, 1024}	integer

We conduct the experiments using the popular CNN models that are trained over ImageNet dataset such as VGG-16 [12], ResNet [13], DenseNet [14], MobileNet [18], and NASNet-Mobile [5]. In this article, we are interested in finding the optimal structure of fully connected layers for efficient transfer learning. To achieve this objective, the parameters (weights) involved in convolution layers of the above CNNs trained over ImageNet dataset [10] are frozen. In other words, the convolution layers of the above CNNs use the pre-trained weights of ImageNet dataset. The parameters involved in newly added FC layers are learned using the back-propagation algorithm [39]. The structure of the FC layers is tuned automatically using Algorithm 1.

1) Chain Structured CNNs (Plain CNNs): In the initial years of deep learning, the CNN architectures proposed such as LeNet [55], AlexNet [1], ZFNet [11], and VGG-16 [12] have the varying number of trainable layers (convolution, Batch Normalization, and fully connected layers) and involves a different set of hyper-parameters. However, the connectivity among the different layers in these architectures remains the same such that layer L_{i+1} receives the input feature map from layer L_i. Similarly layer L_{i+2} receives the input from layer L_{i+1} and so on. We consider VGG-16, a 16 layer chain structured deep CNN to learn the structure of FC layers for efficient transfer learning.

2) CNNs involving Skip Connections: Szegedy *et al*. [2] introduced a deep CNN named GoogLeNet with a careful handcrafted design which allows increasing the depth of the model. GoogLeNet has a basic building block called Inception block that uses multi-scale filters. Later on, the concept of skip connections became very popular after the emergence of ResNet in 2016 [13]. The skip connections are also used by recent models such as DenseNet [14], etc. Moreover, it also became popular among the CNNs learned using NAS methods such as NASNet [5], PNAS [6], etc. A layer in the CNNs involving skip connections receives multiple input feature maps from its previous layers. For example, layer L_{i+1} receives the input from both layers L_i and L_{i-1} as in ResNet [13]; layer L_n receives the input feature map from all of its previous layers $\{L_1, L_2, ... L_{n-1}\}$ as in DenseNet [14]. We utilized ResNet-50, MobileNet, DenseNet-121, and NASNet-Mobile CNNs involving skip connections to learn the structure of FC layers.

C. Datasets

To validate the performance of the proposed method, experiments are conducted on three different kinds of benchmark datasets such as CalTech-101, Oxford-102 Flowers, and UC Merced Land Use.

1) CalTech-101 Dataset: CalTech-101 [40] dataset consists of images belong to 101 object categories. Each class has the number of images between 40 and 800. The most common image categories such as human faces tend to have more images compared to others. The total number of images are 9144 and each image has a varying spatial dimension. To conduct the experiments, we utilize 80% of the data for training (i.e., 7315 images) and the remaining 20% images to validate the performance of the deep neural networks. To fit these images as input to the CNN models, we re-size the image dimension to $224 \times 224 \times 3$. A few samples from CalTech-101 dataset are presented in Fig. 2(a).

2) Oxford-102 Flowers Dataset: Oxford-102 [41] dataset comprises images belong to 102 flower categories that are commonly visible in the United Kingdom. This dataset contains 8189 images such that each class has a varying number of flower images ranging from 40 to 258. We utilize 80% of the dataset (6551 images) for training the CNNs and remaining 1638 images for validating the performance of the CNNs. To input the images to the CNN models, the image dimension is re-sized to $224 \times 224 \times 3$. Some example images from Oxford-102 Flowers dataset are shown in Fig. 2(b).

3) UC Merced Land Use Dataset: UC Merced Land Use dataset [42] contains images belonging to 21 categories of lands. This dataset has a total of 2100 images with 100 images in each class. The developed CNN models have trained over 80 images in each class, and the remaining 20 images are used to validate the performance of the models. The image dimensions are resized from $256 \times 256 \times 3$ to $224 \times 224 \times 3$. A few images from the UC Merced Land Use dataset are shown in Fig. 2(c). Next we present the result obtained by the proposed method when applied on the benchmark datasets.

VI. RESULTS AND DISCUSSIONS

The NAS based CNNs available in the literature generally learn better performing CNN architectures for popular image classification datasets such as ImageNet [10], CIFAR-10 [56] which have large number of training examples. However, the image datasets having less amount of training data are not experienced with the advantage of NAS methods. In this paper, we utilize three benchmark datasets for automatically tuning the FC layers of CNNs for better transfer learning. Due to the above reason, we compare the results obtained

(a) (b) (c)

Fig. 2. (a) A few set of images belong to CalTech-101 [40]. (b) A few random images from Oxford-102 Flowers [41]. (c) Some example images from UC Merced Land Use dataset [42].

TABLE III

THE BEST SET OF FC LAYERS' HYPERPARAMETERS LEARNED FOR CALTECH-101 DATASET USING THE BAYESIAN SEARCH AND RANDOM SEARCH TECHNIQUES. THE OPTIMAL STRUCTURE OF FULLY CONNECTED LAYERS (EXCLUDING THE OUTPUT FC LAYER) FOR POPULAR CNNS SUCH AS VGG-16, RESNET-50, MOBILENET, DENSENET-121, AND NASNET-MOBILE IS PRESENTED.

S.No	Model	Search Method	#FC layers	Activation	#neurons	dropout rate	validation accuracy
1	VGG-16	Bayesian	1	ReLu	256	0	92.72
		random	1	ReLu	512	0.3	92.34
2	ResNet	Bayesian	0	-	-	-	90.15
		random	1	Sigmoid	256	0.2	89.83
3	MobileNet	Bayesian	1	ReLu	1024	0.3	92.50
		random	1	ReLu	256	0	88.73
4	DenseNet	Bayesian	1	Sigmoid	1024	0.3	90.21
		random	1	ReLu	1024	0	88.79
5	NASNet-Mobile	Bayesian	1	ReLu	1024	0.1	88.51
		random	1	Sigmoid	256	0	86.65

TABLE IV

RESULTS COMPARISON BETWEEN THE PROPOSED AUTOFCL AND THE STATE-OF-THE-ART METHODS OVER CALTECH-101, OXFORD-102 FLOWERS, AND UC MERCED LAND USE DATASETS. THE STATE-OF-THE-ART INCLUDING BOTH TRANSFER LEARNING-BASED AND NON-TRANSFER LEARNING-BASED METHODS ARE LISTED IN THIS TABLE. THE ROWS CORRESPONDING TO THE BEST AND SECOND-BEST PERFORMANCE OVER EACH DATASET ARE HIGHLIGHTED IN **BOLD** AND ***bold-italic***, RESPECTIVELY.

Dataset	Method	Accuracy	Transfer Learning/Non Transfer Learning
CalTech-101	Lee *et al.* [43]	65.4	Non Transfer Learning
	Cubuk *et al.* [44]	86.9	Transfer Learning
	Sawada *et al.* [45]	91.8	Transfer Learning
	Ours (VGG-16 + AutoFCL)	**94.38 ± 0.005**	**Transfer Learning**
	Ours (ResNet-50 + AutoFCL)	91.13 ± 0.004	Transfer Learning
	Ours (MobileNet + AutoFCL)	***92.07 ± 0.004***	***Transfer Learning***
	Ours (DenseNet-121+ AutoFCL)	89.5 ± 0.005	Transfer Learning
	Ours (NASNetMobile+ AutoFCL)	87.77 ± 0.005	Transfer Learning
Oxford-102 Flowers	Huang *et al.* [46]	85.66	Non Transfer Learning
	Lv *et al.* [47]	92.00	Non Transfer Learning
	Murabito *et al.* [48]	79.4	Non Transfer Learning
	Simon *et al.* [49]	97.1	Transfer Learning
	Karlinsky *et al.* [50]	89	Transfer Learning
	Ours (VGG-16 + AutoFCL)	**98.83 ± 0.001**	**Transfer Learning**
	Ours (ResNet-50 + AutoFCL)	***97.21 ± 0.05***	***Transfer Learning***
	Ours (MobileNet + AutoFCL)	58.6 ± 0.04	Transfer Learning
	Ours (DenseNet-121 + AutoFCL)	60.91 ± 0.03	Transfer Learning
	Ours (NASNetMobile + AutoFCL)	41.3 ± 0.006	Transfer Learning
UC Merced Land Use	Shao *et al.* [51]	92.38	Non Transfer Learning
	Yang *et al.* [52]	93.67	Non Transfer Learning
	Akram *et al.* [53]	**97.6**	**Transfer Learning**
	Wang *et al.* [54]	94.81	Transfer Learning
	Ours (VGG-16 + AutoFCL)	***96.83 ± 0.006***	***Transfer Learning***
	Ours (ResNet-50 + AutoFCL)	78 ± 0.03	Transfer Learning
	Ours (MobileNet + AutoFCL)	88 ± 0.004	Transfer Learning
	Ours (DenseNet-121 + AutoFCL)	80.8 ± 0.015	Transfer Learning
	Ours (NASNetMobile + AutoFCL)	72.28 ± 0.016	Transfer Learning

TABLE V

THE OPTIMAL STRUCTURE OF FC LAYERS LEARNED FOR OXFORD-102 FLOWERS DATASET USING THE BAYESIAN SEARCH AND RANDOM SEARCH. THE VALUES OF VARIOUS HYPERPARAMETERS FOR VGG-16, RESNET-50, MOBILENET, DENSENET-121, NASNET-MOBILE MODELS ARE SHOWN IN THIS TABLE.

S.No	Model	Search Method	#FC layers	Activation	#neurons	dropout rate	validation accuracy
1	VGG-16	Bayesian	1	ReLu	256	0	96.64
		random	1	ReLu	64	0.1	94.33
2	ResNet	Bayesian	1	Sigmoid	512	0.3	96.31
		random	0	-	-	-	91.73
3	MobileNet	Bayesian	1	Sigmoid	512	0.5	61.29
		random	1	Sigmoid	512	0.1	55.67
4	DenseNet	Bayesian	1	Sigmoid	1024	0.3	68.06
		random	1	ReLu	256	0	55.18
5	NASNet-Mobile	Bayesian	1	ReLu	256	0.2	40.37
		random	1	ReLu	512	0.1	38.37

by the proposed AutoFCL with such state-of-the-art methods which include both transfer learning and non-transfer learning based methods.

A. CalTech-101 Image Classification Results

To learn the best set of hyperparameters involved in the FC layers of a CNN, we employ two popularly adopted search methods in the literature of Neural Architecture Search (NAS). Those two search methods include i) Bayesian Optimization and ii) Random Search. Random search chooses the hyperparameters to explore randomly. In our experimental settings, the number of iterations for random sampling is set to 100. Table III presents the comparison among the performance of proxy CNN models (fine-tuning the best FC layer structure learned during the search process) found using Bayesian search and Random search over CalTech-101 dataset. Table III also lists the best possible set of hyperparameter values like the number of FC layers, type of activation, number of neurons in each FC layer, and the dropout factor for each FC layer that are learned during the search process. For example, the best structure of FC layers learned using Bayesian optimization for VGG-16 results in 92.72% validation accuracy. After finding the best set of FC layers' hyperparameters using Algorithm 1, we fine-tune the FC layers of the developed CNN models over the CalTech-101 dataset. The CNN models are trained for 200 epochs using AdaGrad optimizer [25]. We consider the values of other hyperparameters such as the learning rate similar to the setting of training the proxy CNNs explored during the search process. Fine-tuning the FC layers (learned using the proposed AutoFCL) results in state-of-the-art accuracy 94.38% on CalTech-101 dataset.

B. Oxford-102 Flowers Image Classification Results

The optimal FC layers hyperparameters learned for the Oxford-102 Flowers dataset using Bayesian Optimization and random search are shown in Table V. Similar to CalTech-101 dataset, once the search process is completed, the FC layers of the CNN (the best FC layer structure found during the search process) are fine-tuned over the Oxford-102 Flowers dataset for 200 epochs using AdaGrad optimizer [25]. The proposed AutoFCL achieves the state-of-the-art accuracy of 98.83% on Oxford-102 Flowers dataset. Table IV summarizes the performance obtained using the various CNN models with the target-dependent FC layer structure. The VGG-16 and ResNet-50 achieve the best and second-best state-of-the-art accuracy, respectively over Oxford-102 Flowers dataset.

C. UC Merced Land Use Image Classification Results

We consider UC Merced Land Use as another image dataset to learn the best structure of FC layers for efficient transfer learning. The proposed method produces comparable results over UC Merced Land use dataset as presented in Table IV. From Table IV we can observe that fine-tuning the FC layers learned using the proposed AutoFCL for VGG-16 produces 96.83% validation accuracy, which is second best state-of-the-art accuracy. Table VI lists the best configuration of hyperparameters involved in FC layers found using both Bayesian search and random search. We also compared the performance of the proposed method with fine-tuning original CNN architectures over the target dataset. Fine-tuning the target-dependent FC layer's structure of a CNN over the target dataset results in better performance compared to fine-tuning with the target-independent FC layer's structure. Fig. 3 demonstrates that the proposed AutoFCL outperforms traditional fine-tuning of original FC layers of CNN architectures.

VII. CONCLUSION AND FUTURE SCOPE

We propose AutoFCL, a method to learn the best possible set of hyperparameters belonging to Fully Connected (dense) layers of a CNN for improved transfer learning. Finding the structure of FC layers with the knowledge of target data results in better performance while transferring the knowledge from one task to other. The Bayesian Optimization algorithm is used to explore the search space for the number of FC layers, the number of neurons in each FC layer, activation function and dropout factor. To learn the structure of FC layers, experiments are conducted on benchmark datasets. The proposed AutoFCL method outperforms the state-of-the-art on most of the datasets. In future, the proposed idea of tuning the pre-trained CNN layers may be extended to tuning

TABLE VI

THE FC LAYERS' HYPERPARAMETERS ARE TUNED FOR UC MERCED LAND USE DATASET AUTOMATICALLY USING THE BAYESIAN SEARCH AND RANDOM SEARCH ARE PRESENTED.

S.No	Model	Search Method	#FC layers	Activation	#neurons	dropout rate	validation accuracy
1	VGG-16	Bayesian	1	ReLu	512	00.3	96.42
		random	1	ReLu	64	0.1	95.23
2	ResNet	Bayesian	1	Tanh	1024	0.2	83.8
		random	1	Tanh	1024	0.4	82.14
3	MobileNet	Bayesian	1	Sigmoid	1024	0.5	89.52
		random	1	ReLu	1024	0.1	87.38
4	DenseNet	Bayesian	1	Sigmoid	1024	0.0	82.38
		random	1	ReLu	128	0.2	81.42
5	NASNet-Mobile	Bayesian	1	ReLu	128	0	74.76
		random	1	Sigmoid	512	0.4	73.33

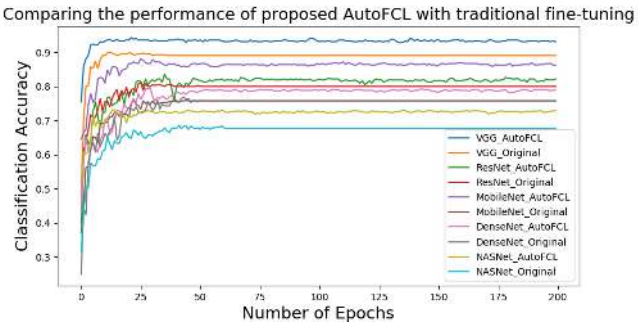

Fig. 3. The performance comparison between the proposed AutoFCL and traditional Fine-tuning methods over UC Merced Land Use dataset. Learning the optimal structure of FC layers with the knowledge of the target dataset and fine-tuning the learned FC layers leads to better performance.

the number of Convolution layers of a CNN based on the similarity between the source and target datasets.

ACKNOWLEDGMENT

We appreciate NVIDIA Corporation's support with the donation of GeForce Titan XP GPU (Grant number: GPU-900-1G611-2500-000T), which is used for this research.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[2] 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*, 2015, pp. 1–9.

[3] G. Hinton, L. Deng, D. Yu, G. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, B. Kingsbury *et al.*, "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal processing magazine*, vol. 29, 2012.

[4] M. Wang, S. Abdelfattah, N. Moustafa, and J. Hu, "Deep gaussian mixture-hidden markov model for classification of eeg signals," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 4, pp. 278–287, 2018.

[5] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8697–8710.

[6] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy, "Progressive neural architecture search," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 19–34.

[7] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *arXiv preprint arXiv:1808.05377*, 2018.

[8] Y. Jaafra, J. L. Laurent, A. Deruyver, and M. S. Naceur, "Reinforcement learning for neural architecture search: A review," *Image and Vision Computing*, vol. 89, pp. 57–66, 2019.

[9] S. S. Basha, S. R. Dubey, V. Pulabaigari, and S. Mukherjee, "Impact of fully connected layers on performance of convolutional neural networks for image classification," *Neurocomputing*, 2019.

[10] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. Ieee, 2009, pp. 248–255.

[11] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*. Springer, 2014, pp. 818–833.

[12] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

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

[14] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[15] Q. Xu, M. Zhang, Z. Gu, and G. Pan, "Overfitting remedy by sparsifying regularization on fully-connected layers of cnns," *Neurocomputing*, vol. 328, pp. 69–74, 2019.

[16] H. Mendoza, A. Klein, M. Feurer, J. T. Springenberg, and F. Hutter, "Towards automatically-tuned neural networks," in *Workshop on Automatic Machine Learning*, 2016, pp. 58–65.

[17] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[18] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[19] H.-W. Ng, V. D. Nguyen, V. Vonikakis, and S. Winkler, "Deep learning for emotion recognition on small datasets using transfer learning," in *Proceedings of the 2015 ACM on international conference on multimodal interaction*. ACM, 2015, pp. 443–449.

[20] P. I. Frazier, "A tutorial on bayesian optimization," *arXiv preprint arXiv:1807.02811*, 2018.

[21] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[22] X. Li, Y. Grandvalet, F. Davoine, J. Cheng, Y. Cui, H. Zhang, S. Belongie, Y.-H. Tsai, and M.-H. Yang, "Transfer learning in computer vision tasks: Remember where you come from," *Image and Vision Computing*, vol. 93, p. 103853, 2020.

[23] J. Hu, "Discriminative transfer learning with sparsity regularization for

single-sample face recognition," *Image and vision computing*, vol. 60, pp. 48–57, 2017.

[24] D. Han, Q. Liu, and W. Fan, "A new image classification method using cnn transfer learning and web data augmentation," *Expert Systems with Applications*, vol. 95, pp. 43–56, 2018.

[25] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[26] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[27] M. Wistuba, "Bayesian optimization combined with successive halving for neural network architecture optimization." in *AutoML@ PKDD/ECML*, 2017, pp. 2–11.

[28] D. Ji, Y. Jiang, P. Qian, and S. Wang, "A novel doubly reweighting multisource transfer learning framework," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 3, no. 5, pp. 380–391, 2019.

[29] A. Gupta, Y.-S. Ong, and L. Feng, "Insights on transfer optimization: Because experience is the best teacher," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 51–64, 2017.

[30] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Advances in neural information processing systems*, 2014, pp. 3320–3328.

[31] M. Xie, N. Jean, M. Burke, D. Lobell, and S. Ermon, "Transfer learning from deep features for remote sensing and poverty mapping," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[32] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient transfer learning," *arXiv preprint arXiv:1611.06440*, vol. 3, 2016.

[33] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in neural information processing systems*, 2012, pp. 2951–2959.

[34] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006, vol. 2, no. 3.

[35] C. E. Rasmussen, "Gaussian processes in machine learning," in *Summer School on Machine Learning*. Springer, 2003, pp. 63–71.

[36] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global optimization*, vol. 13, no. 4, pp. 455–492, 1998.

[37] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[38] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[39] H. J. Kelley, "Gradient theory of optimal flight paths," *Ars Journal*, vol. 30, no. 10, pp. 947–954, 1960.

[40] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 4, pp. 594–611, 2006.

[41] M.-E. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008.

[42] Y. Yang and S. Newsam, "Bag-of-visual-words and spatial extensions for land-use classification," in *Proceedings of the 18th SIGSPATIAL international conference on advances in geographic information systems*. ACM, 2010, pp. 270–279.

[43] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proceedings of the 26th annual international conference on machine learning*. ACM, 2009, pp. 609–616.

[44] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "Autoaugment: Learning augmentation strategies from data," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 113–123.

[45] Y. Sawada, Y. Sato, T. Nakada, S. Yamaguchi, K. Ujimoto, and N. Hayashi, "Improvement in classification performance based on target vector modification for all-transfer deep learning," *Applied Sciences*, vol. 9, no. 1, p. 128, 2019.

[46] B. Huang, Y. Hu, Y. Sun, X. Hao, and C. Yan, "A flower classification framework based on ensemble of cnns," in *Pacific Rim Conference on Multimedia*. Springer, 2018, pp. 235–244.

[47] X. Lv and F. Duan, "Metric learning via feature weighting for scalable image retrieval," *Pattern Recognition Letters*, vol. 109, pp. 97–102, 2018.

[48] F. Murabito, C. Spampinato, S. Palazzo, D. Giordano, K. Pogorelov, and M. Riegler, "Top-down saliency detection driven by visual classification," *Computer Vision and Image Understanding*, vol. 172, pp. 67–76, 2018.

[49] M. Simon, E. Rodner, T. Darrell, and J. Denzler, "The whole is more than its parts? from explicit to implicit pose normalization," *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[50] L. Karlinsky, J. Shtok, S. Harary, E. Schwartz, A. Aides, R. Feris, R. Giryes, and A. M. Bronstein, "Repmet: Representative-based metric learning for classification and few-shot object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5197–5206.

[51] W. Shao, W. Yang, G.-S. Xia, and G. Liu, "A hierarchical scheme of multiple feature fusion for high-resolution satellite scene categorization," in *International Conference on Computer Vision Systems*. Springer, 2013, pp. 324–333.

[52] M. Y. Yang, S. Al-Shaikhli, T. Jiang, Y. Cao, and B. Rosenhahn, "Bilayer dictionary learning for remote sensing image classification," in *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. IEEE, 2016, pp. 3059–3062.

[53] T. Akram, B. Laurent, S. R. Naqvi, M. M. Alex, N. Muhammad *et al.*, "A deep heterogeneous feature fusion approach for automatic land-use classification," *Information Sciences*, vol. 467, pp. 199–218, 2018.

[54] E. K. Wang, Y. Li, Z. Nie, J. Yu, Z. Liang, X. Zhang, and S. M. Yiu, "Deep fusion feature based object detection method for high resolution optical remote sensing images," *Applied Sciences*, vol. 9, no. 6, p. 1130, 2019.

[55] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[56] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.