Neural Networks 133 (2021) 112–122

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

AutoTune: Automatically Tuning Convolutional Neural Networks for Improved Transfer Learning

S.H. Shabbeer Basha *, Sravan Kumar Vinakota, Viswanath Pulabaigari, Snehasis Mukherjee, Shiv Ram Dubey

Indian Institute of Information Technology, Sri City, Chittoor, Andhra Pradesh, 517646, India.

ARTICLE INFO

Article history:
Received 18 February 2020
Received in revised form 21 August 2020
Accepted 16 October 2020
Available online 27 October 2020

Keywords:
Convolutional Neural Networks
Fine-tuning
Bayesian optimization
Neural Architecture Search
Transfer learning

ABSTRACT

Transfer learning enables solving a specific task having limited data by using the pre-trained deep networks trained on large-scale datasets. Typically, while transferring the learned knowledge from source task to the target task, the last few layers are fine-tuned (re-trained) over the target dataset. However, these layers are originally designed for the source task that might not be suitable for the target task. In this paper, we introduce a mechanism for automatically tuning the Convolutional Neural Networks (CNN) for improved transfer learning. The pre-trained CNN layers are tuned with the knowledge from target data using Bayesian Optimization. First, we train the final layer of the base CNN model by replacing the number of neurons in the softmax layer with the number of classes involved in the target task. Next, the CNN is tuned automatically by observing the classification performance on the validation data (greedy criteria). To evaluate the performance of the proposed method, experiments are conducted on three benchmark datasets, e.g., CalTech-101, CalTech-256, and Stanford Dogs. The classification results obtained through the proposed AutoTune method outperforms the standard baseline transfer learning methods over the three datasets by achieving 95.92%, 86.54%, and 84.67% accuracy over CalTech-101, CalTech-256, and Stanford Dogs, respectively. The experimental results obtained in this study depict that tuning of the pre-trained CNN layers with the knowledge from the target dataset confesses better transfer learning ability. The source codes are available at https://github.com/JekyllAndHyde8999/AutoTune_CNN_TransferLearning.

1. Introduction

The ability of Convolutional Neural Networks (CNN) to perform feature extraction and decision making in one-shot creates enormous demand in several application areas, such as object recognition (Krizhevsky, Sutskever, & Hinton, 2012), language translation (Zhang, Zong, et al., 2015), and many more. However, the performance of the deep learning models is sensitive w.r.t. the small changes made in both network hyperparameters settings, such as the number of layers, filter dimension of a convolution layer, etc. and it is also sensitive to the other training parameters, such as learning rate, activation function, and so on. Most of the CNNs available in the literature are carefully designed in terms of these hyperparameters by the domain experts (He, Zhang, Ren, & Sun, 2016; Krizhevsky et al., 2012; Simonyan & Zisserman, 2014; Szegedy et al., 2015).

In recent years, researchers have made substantial efforts to automatically learning the structure of a CNN for a specific task (Liu et al., 2018; Zoph, Vasudevan, Shlens, & Le, 2018), known as Neural Architecture Search (NAS). Although these methods out-perform most of the hand-engineered architectures, the search process requires huge computational resources and time to train the proxy CNNs that are explored during the architecture search process (Zoph & Le, 2016). Particularly while working on small datasets, the process of NAS becomes a challenge. Transfer learning is a popularly adopted technique to reduce the demand for both large computational resources and training data by providing promising performance over small datasets.

Many machine learning algorithms assume that the training (source) data and future data (target) have the similar distribution. In this direction, many Metric-learning algorithms (Dong, Du, Zhang, & Zhang, 2017; Hu, Lu, & Tan, 2015) are proposed in the literature. To mention a few, *Deep Transfer Metric Learning (DTML)* method introduced by Hu et al. (2015), in which a discriminative distance network is trained for cross-dataset visual recognition by maximizing the inter-class variations and minimizing the intra-class similarity. Similarly, Dong et al. (2017) proposed an *Ensemble Discriminative Local Metric Learning (EDLML)* which aims at learning a sub-space to keep all the intra-class samples as close as possible, while the samples belong to different

* Corresponding author.
E-mail address: shabbeer.sh@iiits.in (S.H.S. Basha).

https://doi.org/10.1016/j.neunet.2020.10.009
0893-6080/

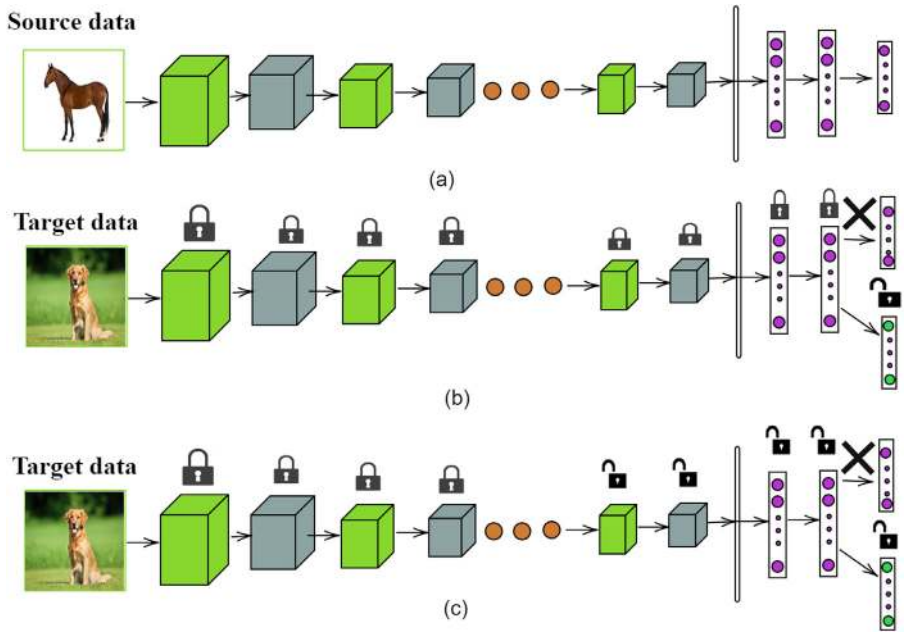

Fig. 1. Overview of the proposed method. (a) Typically, the CNN models used for transferring the knowledge are initially trained over a large-scale image dataset. (b) Conventionally, while transferring the learned knowledge from the source task to the target task, the last one or a few layers of the pre-trained CNN are fine-tuned over the target dataset. (c) The proposed AutoTune method tunes the optimal number of layers automatically using Bayesian Optimization (Frazier, 2018). Note that the lock and unlock symbols are used to represent the frozen and fine-tuned layers, respectively. Finally, the tuned CNN layers can be re-trained over the target dataset for improved transfer learning. Different colors represent different CNN layers.

classes are well separated. Shi, Du, and Zhang (2015) presented a semi-supervised domain adaptation approach which finds new representations of the images belong to the source domain using multiple linear transformations.

Typically, while transforming the learned knowledge from source task to the target task, the classification layer of the pre-trained CNN is dropped, after which a new softmax layer is stacked that is trained over the target dataset during transfer learning. The number of layers to be fine-tuned can be decided majorly based on the size of the target dataset and the similarity between the source and target datasets (Karpathy & Johnson, 2017). However, the pre-trained CNN model is designed for the source dataset, which may not perform well over the target dataset.

In this paper, we attempt to automatically tune the pre-trained CNN to make it suitable for the target task/dataset. To achieve this objective, initially, we drop the softmax layer of the pre-trained CNN by replacing it with a new softmax layer having the neurons equal to the number of classes in the target dataset. Next, we automatically tune the layers of CNN using Bayesian Optimization (Frazier, 2018). It is a well-known fact that the initial layers of CNN represent primitive features such as edges and blobs which are generic to many tasks. On the other hand, the final layers of CNN represent the features that are very specific to the learning task (Zeiler & Fergus, 2014). Based on the above idea, in our work, the layers of the CNN are tuned from right to left (i.e., from final layers to initial layers) by observing the network performance on the validation data. The results obtained through our experiments indicate that tuning the optimum number of layers with respect to the target task leads to better image classification performance in the context of transfer learning. Fig. 1 shows an overview of the proposed idea of improving the transfer learning process. Next, we provide a survey of literature in the specific research area focused in this study.

2. Related works

The hierarchical feature extraction ability of deep neural networks enables the adoption of a deep network. The pre-trained network over a large-scale dataset (source task) can be utilized to solve the specific task/ problem (target task) through transferring the knowledge learned from the source task. Transfer learning has been widely adopted in domains where collecting the annotated examples is expensive (labor-intensive) and time-consuming task, such as biomedical (Raghu, Sriraam, Temel, Rao, & Kubben, 2020; Shin et al., 2016), agriculture (Kamilaris & Prenafeta-Boldú, 2018), signature generation (Nahmias, Cohen, Nissim, & Elovici, 2020) and many more. Khan, Islam, Jan, Din, and Rodrigues (2019) proposed an average-pooling based classifier to detect and classify breast cancer images. Han, Liu, and Fan (2018) introduced a two-step method to improve the generalization ability of CNN over the target dataset using web data augmentation. Yosinski, Clune, Bengio, and Lipson (2014) conducted a study to measure the generality and specificity of different neurons involved in a deep neural network. In other words, they introduced a mechanism to quantify the transfer-ability of features learned by each layer of a CNN. Wang, Du, and Guo (2019a) introduced an approach to induce a common representation feature space for both source and target domains using a CNN model. A semi-supervised domain adaptation approach proposed by Shi et al. (2015), which finds new representations of the images belong to different classes from the source domain.

Automatically searching for better performing CNN architectures for a given task (also called NAS) has gained interest among the researchers in recent years (Elsken, Metzen, & Hutter, 2019). Before the emergence of NAS, hyperparameter optimization has achieved great success in tuning the machine learning algorithms (Bergstra, Bardenet, Bengio, & Kégl, 2011; Snoek, Larochelle, & Adams, 2012). NAS-based CNNs available in the literature consume an enormous amount of search time and GPU hours to find better performing CNNs. For instance, Reinforcement Learning (RL) based NAS method proposed by Zoph and Le (2016) trained 12,800 proxy CNN models for 28 days using 800 GPUs. Moreover, NASNet (Zoph et al., 2018) utilized 500 GPUs for 4 days to train 20,000 proxy CNN models that are sampled during the architecture search. Very recently, Jiang

Table 1
The search space considered for the hyperparameters involved in the convolutional neural network layers such as Convolution, max/average-pooling, and dense layers.

S.No.	Type of the layer	Parameters involved	Parameter values
1	Convolution	Receptive filter size	{1, 2, 3, 5}
		Stride	Always 1
		#receptive filters	{64, 128, 256, 512}
2	Max-pooling	Receptive filter size	{2, 3}
		Stride	Always 1
3	Average-pooling	Receptive filter size	{2, 3}
		Stride	Always 1
3	Fully Connected or Dense	FC layers	{1, 2, 3}
		# neurons	{64, 128, 256, 512, 1024}
4	Dropout	Dropout factor	[0, 1] with offset 0.1

et al. (2020) proposed a multi-objective NAS that is intended to minimize both classification error and network parameters.

Transfer learning allows the applications to utilize the advantage of deep neural networks by reusing the learned knowledge from the source task. Fine-tuning the pre-trained CNN over the target dataset may produce satisfactory results. However, the pre-trained CNN is designed for the source task which may not be suitable for the target task, especially the deeper layers. Therefore, tuning the optimal number of CNN layers and suitable hyperparameters with the knowledge from the target dataset may produce better results compared to the traditional fine-tuning approaches over the target dataset.

After transferring the learned knowledge from a source task, the capacity of the network increases for the target task (Molchanov, Tyree, Karras, Aila, & Kautz, 2016), if the target dataset has a very less number of training examples. Molchanov et al. (2016) proposed a mechanism to progressively prune the least important feature maps to reduce the model's capacity while fine-tuning over the target task. Various studies have reported that the CNN representations learned from a large-scale image dataset in a source domain, can be successfully transferred to a target domain Azizpour, Razavian, Sullivan, Maki, and Carlsson (2015), Yosinski et al. (2014). However, the target domain has limited data compared to the source domain in practice. In such scenarios, transfer learning suffers from overfitting. To address this problem, Liu, Wang, and Qiao (2017) introduced a framework called Hybrid-TransferNet to improve the network's generalization ability while transferring the knowledge from the source domain to the target domain by removing the redundant features. Similarly, Ayinde, Inanc, and Zurada (2019) proposed a method to reduce the inference time of deep neural networks by pruning the redundant feature maps (filters) based on the relative cosine similarities in the feature space.

Recently, Basha, Vinakota, Dubey, Pulabaigari, and Mukherjee (2020) proposed a framework called AutoFCL for automatically tuning the Fully Connected (FC) layers of a pre-trained CNN with the knowledge from the target dataset while transferring the knowledge from the source task. However, in this work, the search space is limited to fully connected layers. To extend this work, we design a mechanism for automatically tuning the CNN w.r.t. the target dataset for improved transfer learning.

The contributions of this research can be summarized as follows:

- This paper introduces a framework called AutoTune which finds the number of layers to be fine-tuned automatically for a target dataset for improved transfer learning.
- Bayesian Optimization technique is applied to learn the pre-trained CNN layers with the knowledge of the target dataset.
- Several experiments are conducted over CalTech-101, CalTech-256, and Stanford Dogs datasets to justify the efficacy of the proposed model using the popular pre-trained CNNs, namely, VGG-16 (Simonyan & Zisserman, 2014), ResNet-50 (He et al., 2016), and DenseNet-121 (Huang, Liu, Van Der Maaten, & Weinberger, 2017). The obtained results are compared with state-of-the-art that includes both transfer learning and non-transfer learning-based methods.

Next, we discuss the proposed method in detail.

3. Methods

The task of automatically tuning a pre-trained CNN w.r.t. the target dataset can be formulated as a black-box optimization problem where we do not have direct access to the objective function. In this paper, tuning the CNN layers with the knowledge of the target dataset for improved transfer learning is achieved using Bayesian Optimization (Frazier, 2018). Let F be the objective function, which is of the form,

$$F : \mathbb{R}^d \rightarrow \mathbb{R}. \tag{1}$$

The objective of the Bayesian optimization can be represented mathematically as follows,

$$x_* = \underset{x \in \mathcal{S}}{\operatorname{argmax}}\, F(x), \tag{2}$$

where $x \in \mathbb{R}^d$ is the input, the hyperparameter search space is denoted by $\mathcal{S}$ which is shown in Table 1. Evaluating the value of the objective function F at any point in the search space is expensive. It can be obtained by fine-tuning (re-training) the proxy CNN layers (explored during the architecture search) over the target dataset.

The optimal configuration of the hyperparameters involved in the tuned layers (learned using Bayesian optimization) is represented using x_*. Hence, the pre-trained CNN having the last few layers with the optimal structure (x_*) is responsible for obtaining the best performance on the validation (held-out) data Validation_{Data}. The proposed method for automatically tuning the CNN with the knowledge from the target dataset yields improved performance while transferring the learned information from the source dataset to the target dataset. The proposed idea is outlined in Algorithm 1.

3.1. Proposed AutoTune method

The objective of the proposed AutoTune method is to find the optimal structure of pre-trained CNN for improved transfer learning. To achieve this objective, Algorithm 1 takes Base_{CNN} (B) i.e., a pre-trained CNN, hyperparameter search space (HParam_{space}), Training_{Data}, Validation_{Data}, and maximum number of Epochs (E) to train each proxy CNN as input. It tunes the pre-trained CNN layers w.r.t. the target dataset using Bayesian optimization (Bayes Opt) (Frazier, 2018).

Algorithm 1 AutoTUNE: Automatically tuning the CNN for improved transfer learning using Bayesian optimization

Inputs: B (Base$_{CNN}$), *HParam_space* (hyperparameters search space), Training$_{Data}$, Validation$_{Data}$, Epochs (num of epochs to train each proxy CNN).
Output: A CNN with target-dependent network architecture.

```
procedure AutoTune
    Assume Gaussian Process (GP) prior on the objective function F
    Find and observe the objective F at initial m_0 points n = m_0
    while k ∈ n + 1, .., N do                ▷ explore the hyperparameter search space
        Update the posterior distribution on F using the prior
        Choose next sample x_k that maximizes the acquisition function value
        Observe y_k = F(x_k)
    return x_k                               ▷ return a point with best FC layer structure
```

Bayesian optimization has been used widely to tune the hyperparameters involved in machine learning algorithms such as deep neural networks (Snoek et al., 2012). Bayesian optimization is a type of machine learning based optimization problem in which the objective is a black-box function. Here, we provide brief details about Bayesian optimization, for complete details we recommend the reader to refer the original paper (Frazier, 2018). Bayesian optimization includes two key components, including a surrogate model and an acquisition function. The surrogate model is a Bayesian statistical model that builds an approximation for the objective function using Gaussian Process (GP) Regression (Rasmussen, 2003). The acquisition function utilizes this Bayesian statistical model to make the search process productive by proposing the next point to explore from the search space.

As outlined in Algorithm 1, initially the objective function F is observed at m_0 points which are chosen uniformly random (in our experimental settings m_0 is considered as 20). Among the m_0 function evaluations, we pick the optimal point x^+, the hyperparameter configuration corresponding to the maximum validation accuracy ($F(x^+)$) observed while fine-tuning the CNN over the target dataset. Next, we need to conduct an evaluation at new point x_{new}, which is $F(x_{new})$. At this moment, the optimal objective value is either $F(x^+)$ if $F(x^+) \geq F(x_{new})$ or $F(x_{new})$ if $F(x^+) \leq F(x_{new})$. The improvement or gain in the value of objective after observing at this point is $F(x_{new}) - F(x^+)$ which is either positive when $F(x_{new}) > F(x^+)$, or zero when $F(x_{new}) \leq F(x^+)$. We can represent the value of improvement as $[F(x_{new}) - F(x^+)]^+$, where $[]^+$ denotes the positive quantity of improvement. At iteration $k = m_0+1$, we choose x_{new} at which the improvement is high. However, the value of objective $F(x_{new})$ is unknown before evaluation (which is a typically expensive task). Alternatively, we can compute the expected value of the improvement and choose x_{new} for which the improvement is maximized. The Expected Improvement (EI) is defined as follows:

$$EI_{m_0}(x) := E_{m_0}[[F(x_{new}) - F(x^+)]^+] \tag{3}$$

where E_{m_0} is the expectation computed under posterior distribution given the evaluations of objective F at the points $x_1, x_2, \ldots, x_{m_0}$. The posterior probability of $F(x_{new})$ given $F(x_{1:m_0})$ is normally distributed with mean $\mu_{m_0}(x_{new})$ and variance $\sigma^2_{m_0}(x_{new})$, which can be formally represented as,

$$F(x_{new})|F(x_{1:m_0}) \sim Normal(\mu_{m_0}(x_{new}), \sigma^2_{m_0}(x_{new})) \tag{4}$$

where $\mu_{m_0}(x_{new})$ and $\sigma^2_{m_0}(x_{new})$ are analytically computed as,

$$\mu_{m_0}(x_{new}) = \sum_0(x_{new}, x_{1:m_0}) \sum_0(x_{1:m_0}, x_{1:m_0})^{-1} \times (F(x_{1:m_0}) - \mu_0(x_{1:m_0})) + \mu_0(x_{new}) \tag{5}$$

$$\sigma^2_{m_0}(x_{new}) = \sum_0(x_{new}, x_{new}) - \sum_0(x_{new}, x_{1:m_0}) \times \sum_0(x_{1:m_0}, x_{1:m_0})^{-1} \sum_0(x_{1:m_0}, x_{new}) \tag{6}$$

The Expected Improvement (EI) acquisition function lets the Algorithm 1 to evaluate at the point which results in a maximum improvement in the expectation.

$$x_{k+1} = \text{argmax}\, EI_k(x) \tag{7}$$

We update the posterior distribution upon observing the objective F at each point using Eq. (4). Throughout our experiments, the evaluations (N) of the objective F are performed for 50 (including 20 initial evaluations which are sampled in a uniform random manner) in the case of Bayesian Optimization. On the other hand, 100 evaluations are performed in case of random search.

4. Architecture tuning search space

Here, we provide a detailed discussion on the used search space for the hyperparameters involved in the different layers of a deep neural network, such as convolution, max-pooling, average pooling, and dense layers. In our experimental settings, we consider the plain as well as skip connection based CNNs. More concretely, we consider the popular CNNs such as VGG-16 (Simonyan & Zisserman, 2014), ResNet-50 (He et al., 2016) and DenseNet-121 (Huang et al., 2017). These networks are originally pre-trained over ImageNet (Deng et al., 2009) dataset for automatically tuning w.r.t. the target dataset for improved transfer learning.

Basha et al. (2020) observed that learning the structure of Fully Connected (FC) layers with the knowledge from the target dataset and fine-tuning these learned FC layers over the target dataset leads to better performance compared to just fine-tuning with the original CNN. To extend this work, in this article, we propose a framework for automatically tuning the CNN (beyond FC layers) for improved transfer learning. From the literature, we found that a shallow/deeper CNN, which has 3 FC layers (Krizhevsky et al., 2012; Simonyan & Zisserman, 2014) including the output layer achieves comparable performance. Hence, we consider, the search space for the "number of FC layers" hyperparameter in the range {1, 2, 3} including the output FC layer. Another key hyperparameter involved in FC layers is the number of neurons, for which our search space is {64, 128, 256, 512, 1024}. Dropout (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014) is a popular regularization method adopted to generalize the network performance over unseen data. We employed dropout after each dense layer of proxy CNNs explored during the architecture search. The dropout rate is tuned within the range [0, 1] with an offset 0.1, i.e., the optimal dropout factor is learned from the

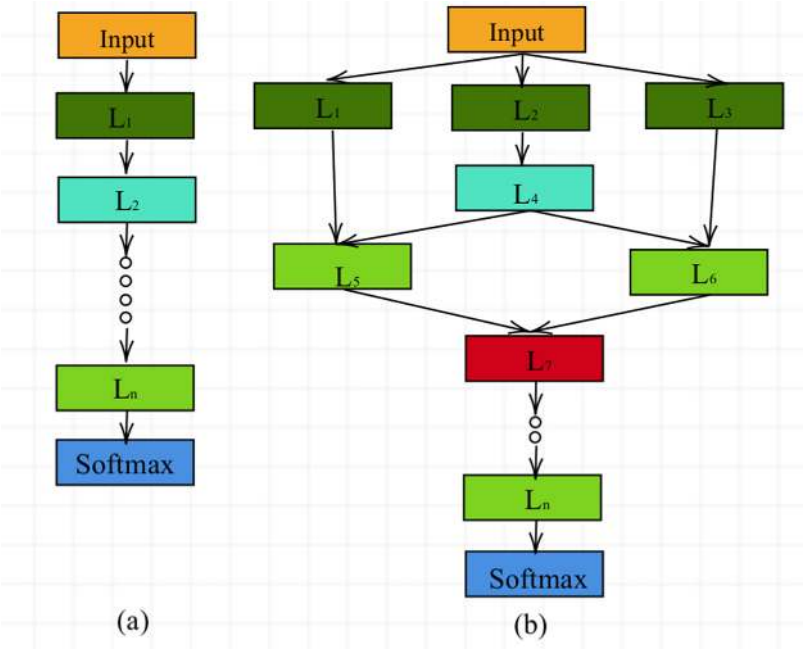

Fig. 2. The pictorial representation of different types of CNNs. Each rectangle box represents a layer in CNN such as convolution, max-pooling, and dense layers. Different colors are used to represent different layers. (a) Illustration of chain-structured CNNs. (b) The more complex type of architecture involving skip connections, where a layer receives input feature-map from one or more layers.

values $\{0.0, 0.1, \ldots, 0.9, 1.0\}$. The hyperparameters involved in other layers such as Convolution, Pooling are shown in Table 1. Our search space involves 6 operations in both convolution and pooling layers which is less as compared to 8 operations used in Liu et al. (2018) and 13 operations used in Zoph et al. (2018).

Throughout our experiments, we have not modified the connectivity of the CNN models that are utilized to carry out the experiments. We utilize two more operators, including (1) 1×1 convolution and (2) upsample operations (similar to un-pooling) to tackle the tensor dimension mismatch in depth and spatial dimensions, respectively.

5. Experimental settings

We first demonstrate the details about the hyperparameters such as learning rate, optimizer, etc., employed while training the CNNs in Section 5.1. The pre-trained deep neural networks used to tune the CNN w.r.t. the target dataset are discussed in Section 5.2, and the datasets utilized for conducting the experiments are discussed in Section 5.3.

5.1. Training details

We call the CNNs explored during the search process as proxy CNNs. To train the proxy CNNs, the parameters or weights involved in the layers (which are tuned during the search process) are initialized with "He-normal" initialization (He, Zhang, Ren, & Sun, 2015). The CNNs are trained using Adagrad optimizer (Duchi, Hazan, & Singer, 2011) with an initial value of the learning rate as 0.01. The learning rate is decreased by a factor of $\sqrt{0.1}$ if there is no reduction in the validation loss. Rectified Linear Unit (ReLU) is employed as the activation function throughout the experiments. Since the architecture search is a time-consuming task, to alleviate this situation each proxy CNN is trained for 50 epochs. To reduce the over-fitting, data augmentation such as shearing, zooming the images, horizontal, and vertical flip image transformations are employed. Batch Normalization (Ioffe & Szegedy, 2015) is used after every dense layer to accelerate the training process and to increase the generalization ability.

5.2. Deep CNNs used for AutoTune

From the literature of deep neural networks, we can broadly classify the CNNs into two types based on the network connectivity, i.e., (i) Chain-structured or Plain CNNs (Elsken et al., 2019) and (ii) CNNs with skip connections.

Plain CNNs: A plain CNN consists of collection of n layers stacked sequentially. The kth layer L_k receives the input from layer L_{k-1} and its output feature map is supplied as input to the layer L_{k+1}. For instance, the hand-designed CNNs proposed in the initial years (2012–2014) such as LeNet (LeCun, Bottou, Bengio, Haffner, et al., 1998), AlexNet (Krizhevsky et al., 2012), ZFNet (Zeiler & Fergus, 2014), VGGNet (Simonyan & Zisserman, 2014), and NAS based models (Baker, Gupta, Naik, & Raskar, 2016; Baker, Gupta, Raskar, & Naik, 2017) have chain-structured connections. A typical structure of plain CNNs is shown in Fig. 2(a). In this article, we utilize the VGG-16 (Simonyan & Zisserman, 2014), a plain structured deep CNN for conducting the experiments.

CNNs with skip connection: This class of CNNs involves skip-connections which allow a layer to receive input feature maps from more than one layer. Let us consider, a CNN is having n layers which are represented as $\{L_1, L_2, \ldots, L_k, \ldots, L_{n-1}, L_n\}$, where the layers L_1, L_n are input, output layers, respectively. In the CNNs involving skip connections, the layer L_k receives input feature-maps from both layers $L_k - 1$ and $L_k - 2$ as in ResNet (He et al., 2016). Similarly, in DenseNet proposed by Huang et al. (2017), layer L_k receives the input feature-maps from all of its previous layers $L_1, L_2, \ldots . L_{k-1}$. In 2015, Szegedy et al. (2015) developed a CNN called GoogLeNet by investing more time and human efforts. Recently proposed NAS based CNNs such as Liu, Yang, and Li (2015) and Zoph et al. (2018) discover more complex CNNs during their search process. Fig. 2(b) demonstrates the abstract view of CNNs involving skip connections. In this research, we use ResNet-50 (He et al., 2016) and DenseNet-121 (Huang et al., 2017) for conducting the experiments of automatically tuning CNNs for improved transfer learning.

Table 2
Comparing the classification performance obtained using the proposed method which is developed using both Bayesian Optimization (BO) and Random Search (RS) with state-of-the-art methods over CalTech-101, CalTech-256, and Stanford Dogs datasets. We compare our results with both transfer learning and non-transfer learning based methods. The best and second-best classification accuracies on each dataset are highlighted in ***bold–italic*** and **bold**, respectively. Note that the results of existing methods are taken from the respective papers.

Dataset	Method	Accuracy	Transfer learning
CalTech-101	Lee, Grosse, Ranganath, and Ng (2009)	65.4	✘
	Zeiler and Fergus (2014)	86.5	✓
	Cubuk, Zoph, Mane, Vasudevan, and Le (2019)	86.9	✓
	Sawada et al. (2019)	91.8	✓
	Basha et al. (2020)	94.38 ± 0.015	✓
	AutoTune (BO) + VGG-16	**95.83 ± 0.004**	✓
	AutoTune (RS) + VGG-16	93.54 ± 0.02	✓
	AutoTune (BO) + ResNet-50	93.57 ± 0.034	✓
	AutoTune (RS) + ResNet-50	91.52 ± 0.004	✓
	AutoTune (BO) + DenseNet-121	***95.92 ± 0.025***	✓
	AutoTune (RS) + DenseNet-121	93.71 ± 0.031	✓
CalTech-256	Zeiler and Fergus (2014)	74.2	✓
	Wang, Zhang, Li, Zhang, and Lin (2016)	74.2	✘
	Schwartz et al. (2018)	83.6	✘
	Chu, Madhavan, Beijbom, Hoffman, and Darrell (2016)	71.4	✓
	Cai, Zhang, Zuo, and Feng (2016)	83.3	✓
	Zheng, Zhao, Wang, Wang, and Tian (2016)	83.27	✓
	Mahmood, Bennamoun, An, and Sohel (2017)	82.1	✓
	Wang et al. (2019b)	81.32	✓
	AutoTune (BO) + VGG-16	82.47 ± 0.003	✓
	AutoTune (RS) + VGG-16	81.83 ± 0.021	✓
	AutoTune (BO) + ResNet-50	84.31 ± 0.005	✓
	AutoTune (RS) + ResNet-50	83.74 ± 0.001	✓
	AutoTune (BO) + DenseNet-121	***86.54 ± 0.023***	✓
	AutoTune (RS) + DenseNet-121	**85.96 ± 0.045**	✓
Stanford Dogs	Murabito et al. (2018)	70.5	✘
	Lee, Sattigeri, and Wornell (2019)	55.2	✓
	Li, Grandvalet, and Davoine (2020)	77.1	✓
	Dubey et al. (2018)	**83.75**	✓
	Shen, Wang, Song, Sun, and Song (2019)	65.5	✓
	AutoTune (BO) + VGG-16	81.23 ± 0.002	✓
	AutoTune (RS) + VGG-16	78.21 ± 0.005	✓
	AutoTune (BO) + ResNet-50	83.17 ± 0.022	✓
	AutoTune (RS) + ResNet-50	82.4 ± 0.06	✓
	AutoTune (BO) + DenseNet-121	***84.67 ± 0.014***	✓
	AutoTune (RS) + DenseNet-121	83.19 ± 0.004	✓

5.3. Datasets

We use three publicly available standard datasets, including CalTech-101 (Fei-Fei, Fergus, & Perona, 2004), CalTech-256 (Griffin, Holub, & Perona, 2007), and Stanford Dogs (Khosla, Jayadevaprakash, Yao, & Li, 2011), to perform the experiments.

5.3.1. CalTech-101

The CalTech-101 dataset is proposed by Fei-Fei et al. (2004). It contains images from 101 classes. This dataset has 9144 images, among which 7315 images are utilized for training and the remaining 1839 images are used for validating the classification performance of the CNN models. The dimension of the images is adjusted to $224 \times 224 \times 3$ to fit the images as per the input needed to the deep neural networks. A few samples of images are depicted in Fig. 3(a).

5.3.2. CalTech-256

The CalTech-256 (Griffin et al., 2007) dataset is an improvement made over CalTech-101 (Fei-Fei et al., 2004) dataset in many aspects such as the total number of images increased from 9144 to 30607, the minimum number of images in each class increased from 31 to 80, the total number of classes are more than twice the number of images in CalTech-101 dataset, and many more. This dataset has 30,607 images, out of which 80% of the images, i.e., 24,485 are utilized for training the CNNs and remaining images are used to validate the performance of the proposed method. The spatial dimension of the images is re-sized to $224 \times 224 \times 3$ to make the image dimension to fit as input to the CNNs. Some sample images of the CalTech-256 dataset are shown in Fig. 3(b).

5.3.3. Stanford Dogs

Stanford Dogs dataset (Khosla et al., 2011) consists of 20,580 images belonging to 120 different dog breeds. This dataset has developed using the images and annotations from ImageNet (Deng et al., 2009) dataset for fine-grained image recognition. To tune and train the CNNs, 12000 of the total images are utilized and the remaining 8580 images are used to evaluate the performance of the models. The objects in the images are extracted using the bounding boxes information provided with the dataset. A small set of images from the Stanford Dogs dataset are depicted in Fig. 3(c).

6. Results and discussion

To demonstrate the improved results obtained using the proposed method, this section provides a detailed discussion about the classification performances obtained using the proposed AutoTune method. To find better-performing CNNs, experiments are conducted on three benchmark datasets, including CalTech-101, CalTech-256, and Stanford Dogs. To find the target-specific CNN layers for improved transfer learning, three popular CNNs, including VGG-16 (Simonyan & Zisserman, 2014), ResNet-50 (He et al., 2016), and DenseNet-121 (Huang et al., 2017) are utilized. It is a standard practice in NAS based works (Liu et al., 2018; White,

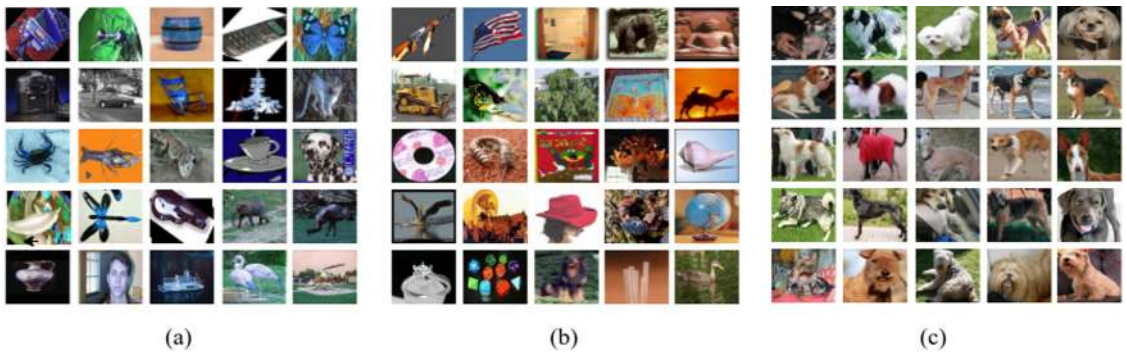

Fig. 3. A set of random examples from (a) CalTech-101 (Fei-Fei et al., 2004), (b) CalTech-256 (Griffin et al., 2007), and (c) Stanford Dogs datasets (Khosla et al., 2011).

Table 3
The best configuration of CNN layers learned for better transfer learning using Bayesian optimization. The base CNN model's layers are tuned w.r.t. the target datasets such as CalTech-101, CalTech-256, and Stanford Dogs. Note that the listed Fully connected (FC) layer's configuration does not include the output FC layer as it always has the number of neurons same as the number of classes.

CNN	Dataset	FC layer			Convolution layer			Max-pooling layer		Validation accuracy
		#layers	#neurons	Dropout factor	#layers	filter size	# filters	#layers	filter size	
VGG-16	CalTech-101	1	1024	0.7	–	–	–	1	2×2	94.82
	CalTech-256	1	1024	0.6	–	–	–	1	3×3	80.16
	Stanford Dogs	0	–	–	–	–	–	–	–	80.02
ResNet-50	CalTech-101	1	256	0.2	2	$[3 \times 3, 3 \times 3]$	[512, 512]	–	–	92.01
	CalTech-256	1	1024	0.6	1	3×3	512	–	–	82.96
	Stanford Dogs	1	256	0.4	–	–	–	–	–	81.63
DenseNet-121	CalTech-101	1	1024	0.3	2	$[5 \times 5, 2 \times 2]$	[512, 128]	–	–	95.21
	CalTech-256	1	1024	0.1	2	$[2 \times 2, 2 \times 2]$	[512, 256]	–	–	85.44
	Stanford Dogs	1	512	0.5	–	–	–	–	–	83.26

Table 4
The optimal structure of CNN layers found for improved transfer learning using random search. The CNN's layers are tuned w.r.t. the target datasets, such as CalTech-101, CalTech-256, and Stanford Dogs. The optimal configuration of the FC layers does not include the output FC layer.

CNN	Dataset	FC layer			Convolution layer			Max-pooling layer		Validation accuracy
		#layers	#neurons	Dropout factor	#layers	filter size	# filters	#layers	filter size	
VGG-16	CalTech-101	1	512	0.6	–	–	–	–	–	92.94
	CalTech-256	1	1024	0.3	–	–	–	1	3×3	79.99
	Stanford Dogs	2	[1024, 512]	[0.3, 0.5]	–	–	–	–	–	77.16
ResNet-50	CalTech-101	1	512	0.0	2	$[2 \times 2, 3 \times 3]$	[512, 256]	–	–	90.29
	CalTech-256	1	1024	0.1	–	–	–	–	–	82.84
	Stanford Dogs	1	1024	0.5	–	–	–	–	–	81.29
DenseNet-121	CalTech-101	1	1024	0.6	1	2×2	512	–	–	92.55
	CalTech-256	1	1024	0.2	1	3×3	256	–	–	85.16
	Stanford Dogs	1	512	0.2	–	–	–	–	–	82.7

Neiswanger, & Savani, 2019; Zoph & Le, 2016) to compare with Random Search (RS). Thus, we also compare the results obtained using the proposed method with the random search approach.

6.1. Classification results over CalTech-101

The proposed AutoTune method learns the suitable structure of CNN layers for improved transfer learning using Algorithm 1. Table 3 lists the optimal configuration of the hyperparameters involved in CNN layers that are learned using Bayesian Optimization. For instance, fine-tuning the ResNet-50 (He et al., 2016) (which is pre-trained on ImageNet) over the CalTech-101 dataset with the following structure of CNN layers allows improved transfer learning ability by achieving 92.01% validation accuracy under our experimental settings:

- An additional Fully Connected (FC) layer with 256 neurons and having a dropout rate of 0.2 along with the output FC layer.
- The last two convolution layers having 512 filters of dimension 3×3 in each layer.

The proposed AutoTune finds a better configuration of hyperparameters compared to random search. It is evident from the results portrayed in Table 4. After learning the suitable structure of CNN layers, we fine-tune the learned CNN layers using Adagrad optimizer (Duchi et al., 2011) for 200 epochs. Similar to the setting of training proxy CNNs, we consider the learning rate as 0.01 and decreased its value with a rate of $\sqrt{0.1}$ for every 50 epochs. Fine-tuning the base CNN models with the optimal structure of CNN layers which are tuned using Bayesian optimization produce state-of-the-art results while transferring the learned knowledge

(a)

#neurons in FC2 \ #neurons in FC1	64	128	256	512	1024
-	89.91	92.22	92.61	93.27	92.83
64	85.99	89.58	91.34	91.67	92.00
128	88.2	91.45	91.51	92.5	93.05
256	89.41	91.84	91.95	92.44	93.44
512	90.79	91.62	92.44	92.84	93.32
1024	91.06	92.44	92.22	93.05	93.38

(b)

#neurons in FC2 \ #neurons in FC1	64	128	256	512	1024
-	70.81	74.84	78.35	79.54	80.3
64	60.29	70.01	73.42	74.58	75.89
128	66.93	72.57	75.82	77.26	77.85
256	69.16	74.82	76.65	78.12	79.22
512	71.23	75.35	77.78	79.2	79.81
1024	71.82	76.23	78.42	79.58	77.55

(c)

#neurons in FC2 \ #neurons in FC1	64	128	256	512	1024
-	68.7	73.15	74.95	75.7	75.00
64	56.41	72.06	73.76	73.88	73.46
128	71.2	74.41	76.06	75.41	75.21
256	73.53	74.8	76.53	76.6	76.91
512	73.63	75.31	77.04	77.28	77.84
1024	74.52	76.45	76.87	77.06	76.96

Fig. 4. Illustrating the sensitivity of the FC layer's hyperparameters like as the number of FC layers and the number of neurons in FC layers on the performance of the deep neural network. The value of a cell (i,j) represents the validation accuracy obtained by fine-tuning (re-training) the FC layers having the mentioned number of neurons corresponding to row i (FC2) and column j (FC1). (a), (b), and (c) present the FC layer's configuration of VGG-16 and the validation accuracy obtained over CalTech-101, CalTech-256, and Stanford Dogs datasets, respectively.

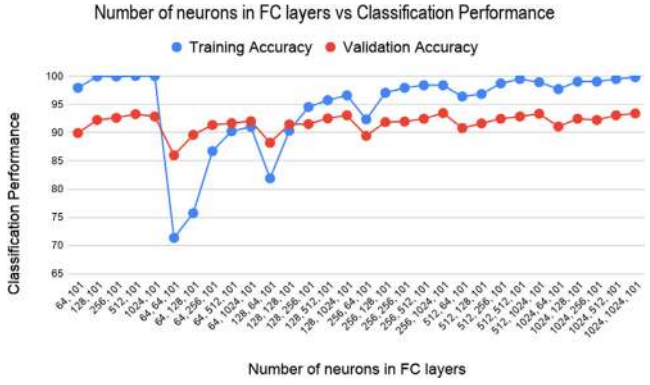

Fig. 5. Demonstration of classification performance as a function of the number of neurons in FC layers. The comparison between the training and validation accuracy obtained by re-training the FC layers of VGG-16 with the specified number of neurons over the CalTech-101 dataset.

from source task to the target task. From Table 2, we can observe that the tuning of last few layers of DenseNet (Huang et al., 2017) and VGG-16 (Simonyan & Zisserman, 2014) using proposed AutoTune method secures the best (95.92%) and the second-best (95.83%) results over the CalTech-101 dataset.

6.2. Classification results over CalTech-256

We consider CalTech-256 as another dataset to learn the suitable structure of CNN layers for improved transfer learning. Tables 3 and 4 present the optimal structure of CNN layers learned using Bayesian Optimization and random search, respectively. From Table 3, we can note that fine-tuning (training) the CNN with suitable CNN layers over the target dataset enables improved transfer learning ability. For example, fine-tuning the DenseNet-121 (Huang et al., 2017) network with the below structure of CNN layers over the CalTech-256 dataset produces the state-of-the-art results.

- An additional FC layer with 1024 neurons and the dropout factor as 0.1 along with the output FC layer.
- The last two convolution layers having filter dimension as 2×2 with 512, 256 number of filters, respectively.

Fine-tuning the proxy CNN with the above specification for DenseNet-121 results in 85.44% classification accuracy, which is state-of-the-art for the CalTech-256 dataset. Furthermore, re-training the DenseNet-121 with the optimal structure of CNN layers for 200 epochs yields 86.54% classification accuracy. Table 2 presents a comparison of the results obtained by employing the proposed AutoTune method with standard baseline results that include both the transfer learning and non-transfer learning based methods. We notice from Table 2 that automatically tuning the CNN over the CalTech-256 dataset using DenseNet-121 (Bayesian Optimization) and DenseNet-121 (Random Search) results in the best (86.54%) and the second-best (85.96%) performance as compared to the state-of-the-art results.

6.3. Classification results over Stanford Dogs

To generalize the significance of the proposed method over different varieties of image classification datasets, we consider the Stanford Dogs dataset having a high degree of inter-class similarity. It is evident from Table 2 that the proposed AutoTune method achieves the state-of-the-art result (i.e., 84.67% classification accuracy) over the Stanford Dogs dataset. From this result, we can observe that though the source and target datasets are either from the same domain or from different domains, tuning the CNN w.r.t. the target dataset yields improved transfer learning results using the proposed AutoTune method.

The optimal structure of CNN layers found for Stanford Dogs dataset using Bayesian Optimization and random search methods are shown in Tables 3 and 4, respectively. For instance, fine-tuning the ResNet-50 w.r.t. the Stanford Dogs dataset with an additional FC layer having 256 neurons and dropout factor as 0.4 along with the output FC layer offers improved performance over validation data.

6.4. Comparing the results with traditional transfer learning

To demonstrate the improved transfer learning using the proposed method, we compare the results obtained for CalTech-101 using the proposed method with conventional fine-tuning and shown the same in Table 5. From this Table, we can observe that the proposed method achieves significant gain in the performance (with a very less number of trainable parameters) compared to the traditional fine-tuning of CNN layers over the target dataset. For instance, traditional fine-tuning of the base VGG-16 model (Simonyan & Zisserman, 2014) achieves 84.21% validation accuracy for CalTech-101, which requires to train 120 Million parameters involved in top 3 layers. On the other hand, fine-tuning the VGG-16 model tuned using the proposed AutoTune method attains 95.83% validation accuracy by training

Table 5
The comparison of transfer learning results obtained for CalTech-101 using the conventional fine-tuning and the proposed AutoTune method which is implemented using Bayesian Optimization (BO) and Random Search (RS).

CNN architecture	Fine-tuning type	#Layers finetuned	#Trainable parameters (in Millions)	Validation accuracy	Total FLOPs (in 10^8) / FLOPs corresponding to the learned layers (in 10^8)
VGG-16	AutoTune (BO)	3	88.7	95.83	153.7/0.51
	AutoTune (RS)	2	26.2	93.54	153.5/0.12
	Conventional	3	120	84.21	154.6/1.2
ResNet-50	AutoTune (BO)	4	4.8	93.57	37.6/0.8
	AutoTune (RS)	4	2.4	91.52	37.2/0.38
	Conventional	4	4.7	93.55	38.5/2.2
DenseNet-121	AutoTune (BO)	4	1.3	95.92	28.4/0.25
	AutoTune (RS)	3	0.6	93.71	28.3/0.1
	Conventional	4	0.31	92.94	28.2/0.1

Table 6
Comparing the results obtained for CalTech-256 with the conventional fine-tuning and the proposed AutoTune method which is implemented using Bayesian Optimization (BO) and Random Search (RS).

CNN architecture	Fine-tuning type	#Layers finetuned	#Trainable parameters (in Millions)	Validation accuracy	Total FLOPs (in 10^8) / FLOPs corresponding to the learned layers (in 10^8)
VGG-16	AutoTune (BO)	3	75.7	82.47	153.6/0.2
	AutoTune (RS)	3	42.7	81.83	153.6/0.2
	Conventional	3	12.5	79.34	154.6/1.2
ResNet-50	AutoTune (BO)	3	3.1	84.31	38.6/0.6
	AutoTune (RS)	4	2.3	83.74	38.5/0.02
	Conventional	3	3.9	83.99	38.5/0.03
DenseNet-121	AutoTune (BO)	4	1.1	86.54	28.6/0.5
	AutoTune (RS)	3	0.6	85.96	28.3/0.07
	Conventional	4	0.47	84.51	28.2/0.1

Table 7
The transfer learning results obtained for Stanford Dogs using the traditional fine-tuning and the proposed AutoTune implemented using Bayesian Optimization (BO) and Random Search (RS).

CNN architecture	Fine-tuning type	#Layers finetuned	#Trainable parameters (in Millions)	Validation accuracy	Total FLOPs (in 10^8) / FLOPs corresponding to the learned layers (in 10^8)
VGG-16	AutoTune (BO)	1	3	81.23	153.4/0.03
	AutoTune (RS)	3	42.7	78.21	153.7/0.2
	Conventional	1	0.5	78.25	154.6/0.5
ResNet-50	AutoTune (BO)	2	0.55	83.17	38.5/0.005
	AutoTune (RS)	2	2.2	82.4	38.5/0.02
	Conventional	2	1.3	82.21	38.5/0.5
DenseNet-121	AutoTune (BO)	2	0.58	84.67	28.2/0.005
	AutoTune (RS)	2	0.5	83.19	28.2/0.005
	Conventional	2	0.16	83.35	28.2/0.02

88.7 parameters, which is 0.26 times fewer than the parameters involved in traditional fine-tuning.

We have also reported the fine-tuning results obtained for CalTech-256 and Stanford Dogs datasets in Tables 6 and 7, respectively. We can observe from these tables that the trainable parameters and FLOPs resulted with the proposed method are less than the conventional method in most of the cases. It happens because the last few layers are learned w.r.t. the target dataset that has less number of training images. On the other hand, in a few cases (i.e., ResNet-50 and DenseNet-121), the resulted trainable parameters with our method are slightly high since we consider additional FC layers in the search space. Furthermore, to demonstrate the sensitivity of the hyperparameters such as the number of neurons in FC layers, we conduct experiments with VGG-16 by varying the number of FC layers, the number of neurons in each FC layer. The results are illustrated in Fig. 4. From this figure, we can observe the validation accuracies obtained for CalTech-101, CalTech-256, and Stanford Dogs datasets using different configurations of FC layers. For example, fine-tuning (re-training) the weights involved in FC1, FC2, and output FC layers with 512, 1024, and 101 neurons over the CalTech-101 dataset results in 93.05% classification performance. Note that the dropout factor is considered as 0.5 after every FC or Dense layer except the output FC layer. The significance of learning the CNN architecture with the knowledge of target dataset is illustrated in Fig. 5. The sensitivity of the FC layer's hyperparameters can be observed over the classification performance in Fig. 5. The above analysis shows that choosing the suitable FC layer's configuration leads to improve the generalization ability of the deep neural network over the target dataset.

7. Conclusion and future work

We propose a novel framework for automatically tuning the pre-trained CNN w.r.t. the target dataset while transferring the learned knowledge from the source task to the target task. We compare the Bayesian and Random search strategies to perform the tuning of network hyperparameters. Experiments are

conducted using VGG-16, ResNet-50, and DenseNet-121 models over the CalTech-101, CalTech-256, and Stanford Dogs datasets. The models are originally trained over the large-scale ImageNet dataset. The experimental results suggest that the tuning of the CNN layers with the knowledge of the target dataset improves the transferring ability. Automatically tuning the CNN by utilizing the knowledge from the target dataset demonstrates a significant reduction in the error by 27.4%, 17.9%, and 5.6% over the CalTech-101, CalTech-256, and Stanford Dogs datasets, respectively. We further observe that the proposed AutoTune improves performance using both Bayesian and Random search strategies. Moreover, the proposed AutoTune performs much better than the conventional fine-tuning of transfer learning as depicted by the results reported in this study. The FLOPs corresponding to the CNN that is learned using the proposed approach are also optimum. In the future, we would like to extend this idea by employing filter pruning methods after finding the suitable CNN architecture to discover a light-weight CNN for efficient transfer learning.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

We are grateful to NVIDIA corporation for supporting us by donating an NVIDIA Titan Xp GPU (GPU-900-1G611-2500-000T), which is used for this research.

References

Ayinde, B. O., Inanc, T., & Zurada, J. M. (2019). Redundant feature pruning for accelerated inference in deep neural networks. *Neural Networks*, *118*, 148–158.

Azizpour, H., Razavian, A. S., Sullivan, J., Maki, A., & Carlsson, S. (2015). Factors of transferability for a generic convnet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *38*(9), 1790–1802.

Baker, B., Gupta, O., Naik, N., & Raskar, R. (2016). Designing neural network architectures using reinforcement learning. ArXiv preprint arXiv:1611.02167.

Baker, B., Gupta, O., Raskar, R., & Naik, N. (2017). Accelerating neural architecture search using performance prediction. ArXiv preprint arXiv:1705.10823.

Basha, S., Vinakota, S. K., Dubey, S. R., Pulabaigari, V., & Mukherjee, S. (2020). AutoFCL: Automatically tuning fully connected layers for transfer learning. ArXiv preprint arXiv:2001.11951.

Bergstra, J. S., Bardenet, R., Bengio, Y., & Kégl, B. (2011). Algorithms for hyper-parameter optimization. In *Advances in neural information processing systems* (pp. 2546–2554).

Cai, S., Zhang, L., Zuo, W., & Feng, X. (2016). A probabilistic collaborative representation based approach for pattern classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2950–2959).

Chu, B., Madhavan, V., Beijbom, O., Hoffman, J., & Darrell, T. (2016). Best practices for fine-tuning visual classifiers to new domains. In *European conference on computer vision* (pp. 435–442). Springer.

Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., & Le, Q. V. (2019). AutoAugment: Learning augmentation strategies from data. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 113–123).

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *Computer vision and pattern recognition, 2009. CVPR 2009. IEEE Conference on* (pp. 248–255). IEEE.

Dong, Y., Du, B., Zhang, L., & Zhang, L. (2017). Dimensionality reduction and classification of hyperspectral images using ensemble discriminative local metric learning. *IEEE Transactions on Geoscience and Remote Sensing*, *55*(5), 2509–2524.

Dubey, A., Gupta, O., Guo, P., Raskar, R., Farrell, R., & Naik, N. (2018). Pairwise confusion for fine-grained visual classification. In *Proceedings of the European conference on computer vision* (pp. 70–86).

Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, *12*(Jul), 2121–2159.

Elsken, T., Metzen, J. H., & Hutter, F. (2019). Neural architecture search: A survey. *Journal of Machine Learning Research*, *20*(55), 1–21.

Fei-Fei, L., Fergus, R., & Perona, P. (2004). Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *2004 conference on computer vision and pattern recognition workshop* (p. 178). IEEE.

Frazier, P. I. (2018). A tutorial on bayesian optimization. ArXiv preprint arXiv:1807.02811.

Griffin, G., Holub, A., & Perona, P. (2007). *Caltech-256 object category dataset*. California Institute of Technology.

Han, D., Liu, Q., & Fan, W. (2018). A new image classification method using CNN transfer learning and web data augmentation. *Expert Systems with Applications*, *95*, 43–56.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision* (pp. 1026–1034).

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

Hu, J., Lu, J., & Tan, Y.-P. (2015). Deep transfer metric learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 325–333).

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4700–4708).

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. ArXiv preprint arXiv:1502.03167.

Jiang, J., Han, F., Ling, Q., Wang, J., Li, T., & Han, H. (2020). Efficient network architecture search via multiobjective particle swarm optimization based on decomposition. *Neural Networks*, *123*, 305–316.

Kamilaris, A., & Prenafeta-Boldú, F. X. (2018). Deep learning in agriculture: A survey. *Computers and Electronics in Agriculture*, *147*, 70–90.

Karpathy, A., & Johnson, J. (2017). CS231n convolutional neural networks for visual recognition-transfer learning.

Khan, S., Islam, N., Jan, Z., Din, I. U., & Rodrigues, J. J. C. (2019). A novel deep learning based framework for the detection and classification of breast cancer using transfer learning. *Pattern Recognition Letters*, *125*, 1–6.

Khosla, A., Jayadevaprakash, N., Yao, B., & Li, F.-F. (2011). Novel dataset for fine-grained image categorization: Stanford dogs. In *Proc. CVPR workshop on fine-grained visual categorization (Vol. 2) (No. 1)*.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097–1105).

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

Lee, H., Grosse, R., Ranganath, R., & Ng, A. Y. (2009). Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th annual international conference on machine learning* (pp. 609–616). ACM.

Lee, J., Sattigeri, P., & Wornell, G. (2019). Learning new tricks from old dogs: Multi-source transfer learning from pre-trained networks. In *Advances in neural information processing systems* (pp. 4372–4382).

Li, X., Grandvalet, Y., & Davoine, F. (2020). A baseline regularization scheme for transfer learning with convolutional neural networks. *Pattern Recognition*, *98*, Article 107049.

Liu, J., Wang, Y., & Qiao, Y. (2017). Sparse deep transfer learning for convolutional neural network. In *Thirty-first AAAI conference on artificial intelligence*.

Liu, G.-H., Yang, J.-Y., & Li, Z. (2015). Content-based image retrieval using computational visual attention model. *Pattern Recognition*, *48*(8), 2554–2566.

Liu, C., Zoph, B., Neumann, M., Shlens, J., Hua, W., Li, L.-J., et al. (2018). Progressive neural architecture search. In *Proceedings of the European conference on computer vision* (pp. 19–34).

Mahmood, A., Bennamoun, M., An, S., & Sohel, F. (2017). Resfeats: Residual network based features for image classification. In *2017 IEEE international conference on image processing* (pp. 1597–1601). IEEE.

Molchanov, P., Tyree, S., Karras, T., Aila, T., & Kautz, J. (2016). Pruning convolutional neural networks for resource efficient transfer learning. (Vol. 3). ArXiv preprint arXiv:1611.06440.

Murabito, F., Spampinato, C., Palazzo, S., Giordano, D., Pogorelov, K., & Riegler, M. (2018). Top-down saliency detection driven by visual classification. *Computer Vision and Image Understanding*, *172*, 67–76.

Nahmias, D., Cohen, A., Nissim, N., & Elovici, Y. (2020). Deep feature transfer learning for trusted and automated malware signature generation in private cloud environments. *Neural Networks*.

Raghu, S., Sriraam, N., Temel, Y., Rao, S. V., & Kubben, P. L. (2020). EEG based multi-class seizure type classification using convolutional neural network and transfer learning. *Neural Networks*.

Rasmussen, C. E. (2003). Gausslan processes in machine learning. In *Summer school on machine learning* (pp. 63–71). Springer.

Sawada, Y., Sato, Y., Nakada, T., Yamaguchi, S., Ujimoto, K., & Hayashi, N. (2019). Improvement in classification performance based on target vector modification for all-transfer deep learning. *Applied Sciences*, *9*(1), 128.

Schwartz, E., Karlinsky, L., Shtok, J., Harary, S., Marder, M., Kumar, A., et al. (2018). Delta-encoder: an effective sample synthesis method for few-shot object recognition. In *Advances in neural information processing systems* (pp. 2845–2855).

Shen, C., Wang, X., Song, J., Sun, L., & Song, M. (2019). Amalgamating knowledge towards comprehensive classification. In *Proceedings of the AAAI conference on artificial intelligence (Vol. 33)* (pp. 3068–3075).

Shi, Q., Du, B., & Zhang, L. (2015). Domain adaptation for remote sensing image classification: A low-rank reconstruction and instance weighting label propagation inspired algorithm. *IEEE Transactions on Geoscience and Remote Sensing*, *53*(10), 5677–5689.

Shin, H.-C., Roth, H. R., Gao, M., Lu, L., Xu, Z., Nogues, I., et al. (2016). Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning. *IEEE Transactions on Medical Imaging*, *35*(5), 1285–1298.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. ArXiv preprint arXiv:1409.1556.

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems* (pp. 2951–2959).

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, *15*(1), 1929–1958.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., et al. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1–9).

Wang, Z., Du, B., & Guo, Y. (2019a). Domain adaptation with neural embedding matching. *IEEE Transactions on Neural Networks and Learning Systems*.

Wang, L., Wu, Z., Karanam, S., Peng, K.-C., Singh, R. V., Liu, B., et al. (2019b). Sharpen focus: Learning with attention separability and consistency. In *Proceedings of the IEEE international conference on computer vision* (pp. 512–521).

Wang, K., Zhang, D., Li, Y., Zhang, R., & Lin, L. (2016). Cost-effective active learning for deep image classification. *IEEE Transactions on Circuits and Systems for Video Technology*, *27*(12), 2591–2600.

White, C., Neiswanger, W., & Savani, Y. (2019). BANANAS: Bayesian optimization with neural architectures for neural architecture search. ArXiv preprint arXiv:1910.11858.

Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? In *Advances in Neural information processing systems* (pp. 3320–3328).

Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European conference on computer vision* (pp. 818–833). Springer.

Zhang, J., Zong, C., et al. (2015). Deep neural networks in machine translation: An overview.

Zheng, L., Zhao, Y., Wang, S., Wang, J., & Tian, Q. (2016). Good practice in CNN feature transfer. ArXiv preprint arXiv:1604.00133.

Zoph, B., & Le, Q. V. (2016). Neural architecture search with reinforcement learning. ArXiv preprint arXiv:1611.01578.

Zoph, B., Vasudevan, V., Shlens, J., & Le, Q. V. (2018). Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 8697–8710).