MultiTune: Adaptive Integration of Multiple Fine-Tuning Models for Image Classification

Yu Wang(✉), Jo Plested, and Tom Gedeon

Research School of Computer Science, Australian National University, Canberra, Australia
{yu.wang1,jo.plested,tom.gedeon}@anu.edu.au

Abstract. Transfer learning has been widely used as a deep learning technique to solve computer vision related problems, especially when the problem is image classification employing Convolutional Neural Networks (CNN). In this paper, a novel transfer learning approach that can adaptively integrate multiple models with different fine-tuning settings is proposed, which is denoted as ***MultiTune***. To evaluate the performance of MultiTune, we compare it to SpotTune, a state-of-the-art transfer learning technique. Two image datasets from the Visual Decathlon Challenge are used to evaluate the performance of MultiTune. The FGVC-Aircraft dataset is a fine-grained task and the CIFAR100 dataset is a more general task. Results obtained in this paper show that MultiTune outperforms SpotTune on both tasks. We also evaluate MultiTune on a range of target datasets with smaller numbers of images per class. MultiTune outperforms SpotTune on most of these smaller-sized datasets as well. MultiTune is also less computational than SpotTune and requires less time for training for each dataset used in this paper.

Keywords: Transfer learning · Image classification · Convolutional neural networks

1 Introduction

Modern convolutional neural networks, such as AlexNet [7], VGG [12] and ResNet [5] have been proven to be very successful, and able to achieve extraordinary performance on well-known large-scale images datasets, for instance ImageNet [3]. However, due to the large amount of data and limitation of computation, it is usually hard to train a convolutional neural network on a large dataset from scratch. Transfer learning has been introduced to mitigate this issue.

Common machine learning algorithms are often designed to solve single and isolated tasks. However, the study of transfer learning aims to develop methods to transfer knowledge learnt from one or more source tasks and apply this knowledge to improve the learning process in a different but related target task [13]. There are two important concepts frequently used in transfer learning, which

H. Yang et al. (Eds.): ICONIP 2020, CCIS 1332, pp. 488–496, 2020.
https://doi.org/10.1007/978-3-030-63820-7_56

are freezing and fine-tuning. Freezing, or feature extraction, means to freeze the weights that are learnt from the source task and only update the weights in the last classification layer [2]. The frozen weights will act as a feature extractor to solve the target task. Fine-tuning is the opposite of freezing. It is performed where all or most of the weights learnt from the source task are retrained and updated to fit the target task. These pre-trained weights act as a regularizer that prevents overfitting during the learning process of the target task [1].

In this paper, we propose a novel technique that can be used in transfer learning to enable the adaptive integration of multiple fine-tuning models with different fine-tuning settings. It is denoted as ***MultiTune*** and will be mentioned by using this name in this paper.

2 Related Work

2.1 Transferability

There have been numerous researches studying the transferability of features in deep neural networks. One of the most thorough researches was done by Yosinski et al. [14]. They discussed the transferability of features when using convolutional neural networks. It is sated in their paper that features on the first layer seems to occur regardless of the exact loss function and natural image dataset, which can be considered as 'general'. The features on the last layer depend largely on the dataset and task, which can be considered as 'specific'. Their study quantified to which a particular layer is general or specific and found that even features transferred from distant tasks are better than random weights.

It is found that fine-tuning the transferred weights has better performance than freezing the transferred weights, in both cases when the source dataset is highly related to the target dataset and when the source dataset is not so related to the target dataset [6,9,14]. It is a common practice used in image classification to pre-train the model on the ImageNet [3] dataset. Then the learnt weights are transferred to the target dataset and fine-tuned during training. It has been shown that image classification by using this approach can achieve extraordinary results on different target datasets [6].

2.2 Adaptive Fine-Tuning: SpotTune

SpotTune is an adaptive fine-tuning method, which is able to determine which layers to be frozen and which layers to be fine-tuned per training example. The adaptive fine-tuning is achieved by training a policy network together with two parallel CNN models. One of the CNN models has all the layers to be frozen. The other CNN model has all its layers to be fine-tuned. The policy network outputs a decision vector containing 0 or 1 for each layer, where 0 means the image will go through the frozen layer, and 1 means the image will go through the fine-tuned layer. As a result, the optimal route of an image in terms of frozen or fine-tuning can be determined [4]. SpotTune could be considered as a

state-of-the-art technique of transfer learning that achieved the highest score on the Visual Decathlon datasets [10] in 2019. It is used as a baseline to evaluate the performance of MultiTune proposed in this paper. Different to their work, MultiTune proposed by us involves two fine-tuning models with different settings instead of one freezing model and one fine-tuning model. Also, MultiTune does not contain a policy network to generate the routing decision, and therefore is less computational.

2.3 L2-SP Regularization

In transfer learning, it is assumed that the pre-trained model extracts generic features. These generic features are then fine-tuned to be more specific to fit the target task if fine-tuning is used. Thus, when using fine-tuning to solve a related target task, the neural network is initialized with pre-trained parameters (e.g. weights, bias) learnt from source task. However, it is found that some of these parameters may be tuned very far away from their initial values during the process of fine-tuning. This may cause significant losses of the initial knowledge transferred from the source task which is assumed to be relevant to the target task [8]. Li et al. proposed a novel type of regularization to reduce losses of the initially transferred knowledge. The pre-trained model is not only used as the starting point of the fine-tuning process but also used as the reference in the penalty to encode an explicit inductive bias. This novel type of regularization is called L2-SP regularization with SP referring to *Starting Point* of the fine-tuning process. Their results showed that L2-SP is much more effective than the standard L2 penalty that is commonly used in fine-tuning [8]. It can prevent overfitting and retain the knowledge learnt from the source task. So, the L2-SP is used as the regularizer when training the networks.

3 Proposed Approach

3.1 Network Architecture

The CNN architecture used in this paper is a type of ResNet with 26 layers, which is denoted as ResNet-26 [11]. There are 3 macro blocks of convolutional layers in this CNN. Each block has 64, 128, 256 output feature channels, respectively. Also, each macro block contains 4 residual blocks and every residual block consists of 2 convolutional layers with 3×3 filters and shortcut connection that usually used in ResNet. Average pooling with a stride of 2 is used to perform the downsampling and ReLU layers are used as the activation layers. It also contains a convolutional layer at the beginning and a fully connected layer at the end, which makes the total number of layers in this architecture to be 26.

3.2 MultiTune Implementation

The MultiTune is implemented by adding a single-layer neural network after the convolutional layers in the last block, which replaces the last fully connected

Fig. 1. Visualization of MultiTune.

Table 1. Settings of the two fine-tuning models.

Model	Reinitialization	Learning rate	Learning rate decay	LR decay rate
Fine-tuning A	Last block	0.1	[20, 50, 80]	0.1
Fine-tuning B	Last block	0.01 for last block, 0.1 for others	[20, 50, 80]	0.1

layer. This single fully connected layer is denoted as *MultiTune layer* here. In detail, the features extracted by the two ResNet-26 models after the last blocks are concatenated and then pass the MultiTune layer to be classified. Theoretically, the MultiTune layer should determine which features to take from these two different fine-tuning models. The MultiTune model proposed here can be expressed in Eq. 1, where Z represents the output of the MultiTune layer, W represents the weights of the MultiTune layer, X_1 and X_2 are the outputs after the last convolutional blocks of the first and second fine-tuning models, and η is a factor that controls what portion of each model to be used in the MultiTune layer. The presence of η enables an option to allocate more weights to one of the models. This factor η is set to be 0.5 here, which means these two ResNet-26 models are treated equally. Figure 1 is a visualization of MultiTune.

$$Z = W * concat[\eta X_1; (1-\eta)X_2] \tag{1}$$

These two CNN models used here have different fine-tuning settings. For convenience, they are denoted as Fine-Tuning A and Fine-Tuning B. The settings of these two models are listed in the Table 1.

4 Experiments

4.1 Datasets

The datasets used in this paper are taken from the Visual Decathlon challenge. It contains 10 datasets from multiple visual domains. To reduce the computation burden of the evaluation process, the images in the Visual Decathlon Datasets are resized isotropically with a shorter side of 72 pixels [10]. Due to the computational limitations, it is hard to use all these 10 datasets to evaluate the performance of MultiTune. Inspired by Li et al.'s paper where they test their method with different target domains, the same approach is used for the selection

of datasets to evaluate the performance of the model [8]. It is hypothesized that the method outlined in this paper should improve performance in both generic target image datasets and specific target image datasets. So, FGVC-Aircraft and CIFAR100 are used, which represents a more specific and a more generic dataset, respectively. Table 2 summarizes the details of these two datasets.

Table 2. Details of FGVC-Aircraft and CIFAR100 datasets

Dataset	Description	Mean	Standard deviation
FGVC-Aircraft	10,000 images of aircraft, 100 images per class. Training, validation and testing with around 3,333 images for each	[0.47983041, 0.51074066, 0.53437998]	[0.21070221, 0.20508901, 0.23729657]
CIFAR100	60,000 colour images for 100 object categories. 40,000 for training, 10,000 for validation, 10,000 for testing	[0.50705882, 0.48666667, 0.44078431]	[0.26745098, 0.25647059, 0.27607843]

4.2 Training of the Network

As SpotTune will be taken as the baseline in this paper, most of the settings used in MultiTune are set the same as SpotTune to keep consistency. Both of these methods are run with 110 epochs without early stopping. Cross Entropy loss is used because the target task is image classification. And, the optimizer used here is SGD with a momentum of 0.9. SpotTune uses a learning rate of 0.1 for the CNN models, and a learning rate of 0.01 for the policy network. The learning rate decay is set after the 40^{th}, 60^{th} and 80^{th} epoch [4]. MultiTune does not include a policy network, and has different learning rates for each CNN model. The learning rate decay is also set to be different from SpotTune. Also, L2-SP regularization is used in MultiTune. Defining the weights of layers except for the last one as w and the weights of the last layer as $w_{\bar{S}}$, the L2-SP regularizer can be shown in Eq. 2 [8]. α and β in this equation are the factors that control the strength of the penalty, which are both set to 0.01 [8].

$$\Omega(w) = \frac{\alpha}{2}||w - w^0||_2^2 + \frac{\beta}{2}||w_{\bar{S}}||_2^2 \tag{2}$$

The Cross Entropy loss is modified to add this L2-SP regularizer. Equation 3 shows the modified CE loss with L2-SP regularizer, where t_i is the ground truth and y_i is CNN's output for each class i in the dataset C.

$$L(y,t) = -\sum_{i}^{C} t_i log(y_i) + \frac{\alpha}{2}\sum_{i}^{W}||w_i - w_i^0||_2^2 + \frac{\beta}{2}||w_{\bar{S}}||_2^2 \tag{3}$$

5 Results and Analysis

The results of the SpotTune are taken as the baseline in this paper. To let the model see more images for better training and testing, the original code of SpotTune includes the validation set in the training set for each dataset. This means the model is trained on this larger combined training set but still evaluated by using the validation set which is a part of the training set. As a result, the validation accuracy reaches a large figure, almost 100%, after tens of epochs. It is very hard to evaluate the performance of the model in this case.

Table 3. Results of SpotTune and MultiTune on Aircraft and CIFAR100.

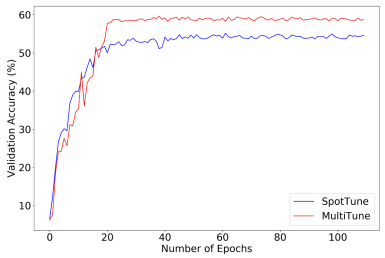
Dataset	SpotTune		MultiTune	
	Validation accuracy	Total time used (mins)	Validation accuracy	Total time used (mins)
Aircraft	55.15%	47.49	59.59%	38.19
Aircraft-20	45.60%	29.15	47.85%	22.50
Aircraft-15	39.20%	21.84	40.73%	16.88
Aircraft-10	30.70%	14.67	29.90%	11.51
Aircraft-5	17.40%	7.57	18.80%	5.91
CIFAR100	78.45%	454.80	79.31%	321.37
CIFAR100-20	59.15%	34.60	59.00%	22.80
CIFAR100-15	55.73%	23.74	56.40%	16.87
CIFAR100-10	49.10%	16.52	49.10%	11.37
CIFAR100-5	33.40%	8.96	29.20%	5.86

To address this issue, the validation set is removed from the training set when loading the datasets. Then the model can be trained only on the training set and evaluated by the unseen validation set.

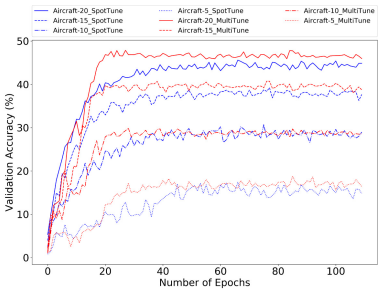
The figures of results shown in Guo et al.'s paper are testing results obtained by submitting the results to the Visual Decathlon Challenge website [4]. Because only two datasets of the Visual Decathlon Datasets are used here, the results of them are not submitted to the website. The validation results instead of testing results are used to compare these methods.

The validation accuracy and training time of SpotTune and MultiTune on Aircraft and CIFAR100 datasets are listed in Table 3. Aircraft-20 means the smaller-sized Aircraft dataset with 20 images per class. Figure 2 shows the validation accuracy versus the number of epochs of SpotTune and MultiTune on these two datasets. To distinguish the results, the validation accuracy of SpotTune is illustrated by *blue* lines, the validation accuracy of MultiTune is shown by *red* lines.

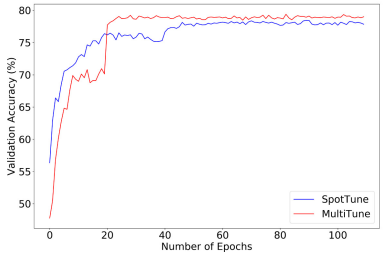
The validation accuracy of MultiTune is consistently higher than that of SpotTune after 20 epochs, and the difference is around 4.5% for the Aircraft



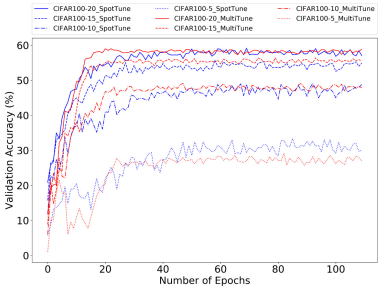
(a) Validation Accuracy versus No. of Epochs on Aircraft Dataset.



(b) Validation Accuracy versus No. of Epochs on Smaller Aircraft Datasets.



(c) Validation Accuracy versus No. of Epochs on CIFAR100 Dataset.



(d) Validation Accuracy versus No. of Epochs on Smaller CIFAR100 Datasets.

Fig. 2. Validation accuracy versus the number of epochs of SpotTune and MultiTune on aircraft, smaller aircraft, CIFAR100 and smaller CIFAR100 datasets. (Color figure online)

dataset and around 1% for the CIFAR100 dataset. The obtained results indicate that MultiTune has better performance than SpotTune on both a more specific dataset (Aircraft) and a more generic dataset (CIFAR100) when running these methods on the whole datasets. This higher performance achieved by MultiTune is due to the integration of two different fine-tuning models. This integration enables the model to extract more useful features from the image datasets. Reinitializing the last blocks lets the layers in the last blocks to learn from scratch, so that the features learnt can be more specific to the target dataset. Adjusting the learning rate in the last block of Fine-Tuning B reduces the update amount in the last block and facilitates the model to find the global minimum. The total training time is also taken as one of the considerations of the performance. As shown in Table 3, the percentage of reduction in the total training time by using MultiTune is 19.58% and 29.34% for Aircraft and CIFAR100.

Aircraft dataset has around 33 images per class, the number of images per class is roughly reduced by 43% and 60% for Aircraft-20 and Aircraft-15. In these two situations, the results of MultiTune are consistently better than SpotTune after 20 epochs. The difference is around 2.3% and 1.5%, respectively. But, when it comes to extremely small datasets, in the Aircraft-10 and Aircraft-5 datasets,

the differences between these two methods are not so obvious. In Aircraft-10 and Aircraft-5, the number of images per class is roughly reduced by 70% and 85%. The inconspicuous differences in these two datasets may be due to the extremely small size of datasets. The significantly reduced size makes the model hard to learn enough knowledge to predict unseen data. It is clear in Fig. 2 that the smaller sized CIFAR100 datasets follow a similar pattern.

6 Conclusion

A novel transfer learning technique denoted as MultiTune is proposed, which can adaptively integrate multiple fine-tuning CNN models with different settings. It has been applied to image classification with two image datasets taken from the Visual Decathlon challenge. MultiTune is able to achieve a validation accuracy of 59.59% on the Aircraft dataset, which is around 4.5% higher than the result obtained by SpotTune. It outperforms SpotTune on the CIFAR100 dataset by around 1%. In addition, MultiTune achieves higher performance than SpotTune on most of the smaller-sized datasets. It also needs much less training time than SpotTune on all the datasets used in this paper. The results outlined in this paper indicate that the proposed MultiTune technique can improve the performance of transfer learning on the image classification problem. This makes MultiTune an excellent approach to be further adopted and applied in the fields of transfer learning and tasks related to image classification.

References

1. Agrawal, P., Girshick, R., Malik, J.: Analyzing the performance of multilayer neural networks for object recognition. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8695, pp. 329–344. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10584-0_22
2. Azizpour, H., Razavian, A.S., Sullivan, J., Maki, A., Carlsson, S.: Factors of transferability for a generic convnet representation. IEEE Trans. Pattern Anal. Mach. Intell. **38**(9), 1790–1802 (2016)
3. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: a large-scale hierarchical image database. In: CVPR 2009 (2009)
4. Guo, Y., Shi, H., Kumar, A., Grauman, K., Rosing, T., Feris, R.S.: SpotTune: transfer learning through adaptive fine-tuning. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4800–4809 (2019)
5. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. CoRR. abs/1512.03385 (2015)
6. Kornblith, S., Shlens, J., Le, Q.V.: Do better imagenet models transfer better? In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2661–2671 (2019)
7. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
8. Li, X., Grandvalet, Y., Davoine, F.: Explicit inductive bias for transfer learning with convolutional networks. CoRR (2018)

9. Plested, J., Gedeon, T.: An analysis of the interaction between transfer learning protocols in deep neural networks. In: Gedeon, T., Wong, K.W., Lee, M. (eds.) ICONIP 2019. LNCS, vol. 11953, pp. 312–323. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-36708-4_26
10. Rebuffi, S., Bilen, H., Vedaldi, A.: Learning multiple visual domains with residual adapters. CoRR (2017)
11. Rebuffi, S., Bilen, H., Vedaldi, A.: Efficient parametrization of multi-domain deep neural networks. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 8119–8127 (2018)
12. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: International Conference on Learning Representations (2015)
13. Torrey, L., Shavlik, J.: Transfer learning. In: Handbook of Research on Machine Learning Applications. IGI Global (2009)
14. Yosinski, J., Clune, J., Bengio, Y., Lipson, H.: How transferable are features in deep neural networks? In: Advances in Neural Information Processing Systems, pp. 3320–3328 (2014)