A Comprehensive Study on Deep Image Classification with Small Datasets

Gayani Chandrarathne[1], Kokul Thanikasalam[*2], Amalka Pinidiyaarachchi[#3]

[#]*Department of statistics and Computer Science, University of Peradeniya, Sri Lanka*

[1] gayani.indunil@gmail.com

[3] ajp@pdn.ac.lk

[*]*Department of Physical Science, Vavuniya Campus, University of Jaffna, Sri Lanka*

[2] kokul@mail.vau.jfn.ac.lk

Abstract

Convolutional neural networks (CNNs) showed state-of-the-art accuracy in image classification on large-scale image datasets. However, CNNs shows considerable poor performance in classifying tiny data since their large number of parameters over-fits the training data. We investigate the classification characteristics of CNNs on tiny data, which are important for many practical applications. This study analyzes the performance of CNNs for direct and transfer learning based training approaches. Evaluation is performed on two publicly available benchmark datasets. Our study shows that fine-tuning source and target network with lower learning rate gives higher accuracy for tiny image classification.

***Keywords*— Deep image classification, CNN, transfer learning.**

I. INTRODUCTION

Image classification is one of the major tasks in computer vision investigated for many years [1], [2], [3]. Classifying images are of great interest in many application areas such as image captioning [4], object tracking [5], [6], scene understanding [7], and event detection [8] and for a multitude of other purposes. Many approaches have been taken to solve it and machine learning has become the most promising method to classify images in human accuracy [9]. With the advancement of deep learning the recently introduced deep convolutional neural networks (DCNN) showed state-of-the art performance in image classification [1], [2], [9]. These DCNNs are able to outweigh challenging problems, background clutter, deformation, occlusion and variations in viewpoint and scale in image classification. The advantage of these DNNs are they come together with feature extractor and classifier which used to be two separate processes, such as Support Vector Machine (SVM) [3], k-nearest neighbour [10], logistic regression and decision trees for classification and , HOG [11], SIFG [12], as hand-crafted feature extraction methods used with those classifiers. And, the Deep CNNs are capable of binary or multi-class classifications in which a set of images is classified into one label or set of labels.

Even though, these DCNN showed most human level accurate results in image classification, obtaining significant results with deep learning is very difficult since deep learning approaches are often required enormous amount of images. It is shown that if a huge amount of data is available, CNNs are capable of achieving better than human performance in visual recognition [9]. The excellent results obtained DNNs are trained on large scale image classification dataset [13], and also have a large number of classes. The DNN requires large data sets in order to generalize the model properly and to avoid over fitting. Also, these remarkable DNN models, which showed state-of-the-art results, contain a large number of layers to distinguish the large number of classes very effectively. The reason for those deeper networks to perform exceptionally with large datasets is, since the larger number of layers is able to handle the large amount of parameters in the dataset with good learning capacity. Even though it is convincing that DNNs requires large datasets, identifying a most suitable size of the dataset is not possible and there is not any indication of the relation between the dataset size and the deep learning model. Hence, this situation becomes more tragedy since most of the real time applications suffer due to the data limitations [14]. For the time being researchers arbitrary choose the deep learning architectures to start training model for classifications.

On the other hand, there is no any ‘fit for all’ minimum dataset size to train a DNN. But, training deep CNN model with less number of layers or training with small datasets prevents getting more accurate results of the model due to over fitting and under fitting problems. CNN models with few layers are unable to utilize

the hierarchical features of a larger dataset through those limited layers and cause the lower accuracies. So, training a DCNN with small datasets is hectic because a small dataset in a DNN leads to under fitting the model. Due to these reasons parameter limitations, CNNs are still struggling [5] to reach the state-of-the-art accuracy for tiny data applications. Image classifier with limited data is beneficial for many real world application areas such as object tracking, scene understanding and real-time since many areas struggle with data dependence of deep learning. Collecting labelled data sets are very expensive in many areas such as medical images, environment science, etc. collecting larger datasets are challenging due to the expensive data labelling and in some applications images are limited.

To overcome the data limitations, researchers are trying to increase the data using different techniques. Data augmentation [1] is a common technique to increase the training data by generating data virtually. Even though these techniques increase the data by generating additional images, CNN models still cannot overturn the issue. In this work we analyse the performance of CNN based image classification with tiny datasets. This paper aims to investigate deep learning approaches for small datasets. We selected two publicly available benchmark datasets for this study. The rest of the paper is organized as follows: Section two describes the background including the related work. Problem overview, methodology, results and discussions are stated in sections three to six, respectively. Finally, the paper is concluded in section seven.

II. BACKGROUND

A. Image Classification with CNNs

CNNs are a special type of Neural Network that works in the same way of a regular neural network except that it has neurons in three dimensions. Though CNNs were introduced two decades back [15] a major breakthrough in image classification with CNN was the model called as AlexNet proposed by Krizhevsky et al [1] a (shown in Fig.1). After the great success of AlexNet, various CNN architectures are proposed for image classification such as VGG-Net [2], GoogleNet [16] and ResNet [9]. Most of the later introduced CNNs for image classification are mostly based on these popular architectures and improves the performance by increasing the number of convolutional layers [2], [16]. Thus, typically CNN for classification tasks is composed with a set of convolution layers and fully connected layers stacked on top of each other as the convolution layers at the beginning and fully connected layers on top of those convolutional layers. The convolutional layers of a these CNN extract the features and then the fully- connected layers perform the classification task [2], [17], [18]. As shown in Fig.1, the popular AlexNet architecture has five convolutional layers and three fully connected layers. Convolution layers represent image features in a layered hierarchical manner [17].

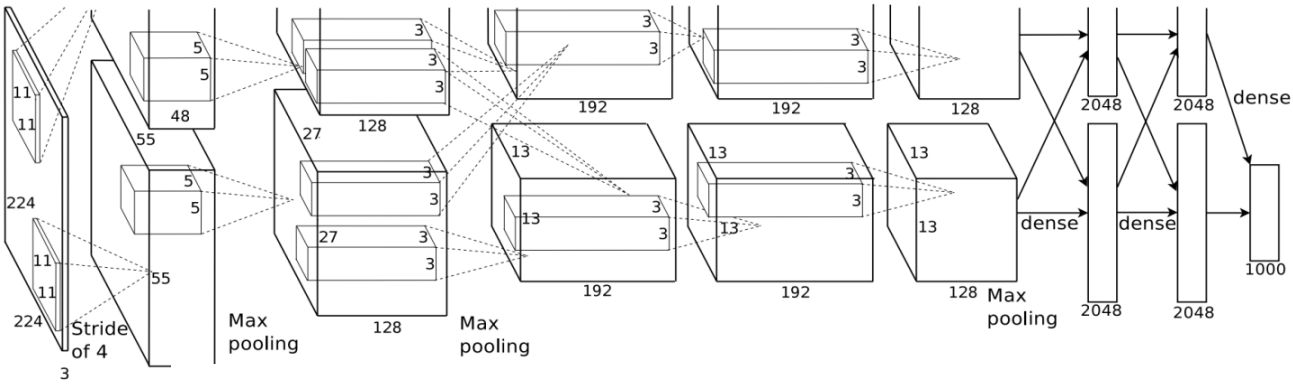

Fig. 1 AlexNet architecture. It has eight layers: five convolutional and three fully connected.

The depth of CNNs or the number of layers in a CNN does a crucial work in a better classification model. By varying the depth of the CNN the learning capacity of CNNs can be controlled [19]. By investigating the introduced models in ILSVRC it can be seen that deepen the model [1], [2], [16], [9]; the accuracy of the model is increased. But, the model depth only can be increased until the model accuracy is saturated [9]. After that, typical stacked layered CNN architectures are not capable to minimize the error [9]. As alternatives to the typical stacked layered architectures GoogleNet proposed a hierarchical convolutional layer structure for classification and ResNet proposed deeper network (up to 152 layers) architecture with a less complex structure.

The internal learning structure of CNN architectures has been investigated for various applications [17], [18]. It clearly shows that CNNs are representing the image features in a layered hierarchical structure. CNN layers tend to learn generic features such as edges, intensity and colour information in first few layers. So it is necessary to understand that more application specific features are learned by last few convolutional layers. Fig.2 shows the feature representation of popular VGG-16 architecture. As shown in the figure, while first convolutional layers represent more generic (low-level) features, last layers represent more specific (high-level) features.

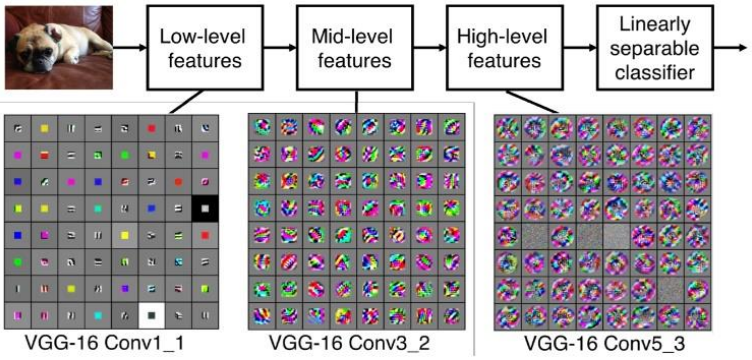

Fig. 2 Visualizing layers of VGG-16 [2] architecture. Image taken from [20]

Ultimately, the important fact is the identifying a good architecture with the concerns of model depth and the learning patterns. The selected CNN architecture should have the appropriate learning capacity for the available dataset and the capability of generalizing the dataset without over fitting. But there is no method of identifying the number of layers based on the dataset. Consequently, many attempts have been taken to classify small datasets using CNN [21], [22]. Even though these approaches improve the performance considerably, they face the over-fitting problem because of direct training (from scratch). So it leaves the question whether shallow architectures capable enough to capture all features for small data If not, how to train a large CNN architecture with tiny data. Therefore recent approaches avoid direct training and improve the performance by transfer learning approaches and some other techniques.

B. Deep Transfer Learning

Very recently, CNN models started using transfer learning approaches to train tiny data [5]. These approaches have addressed the data limitation issue by transferring learned knowledge transfer from one application domain to another relevant domain [23]. They transfer the knowledge from pre-trained CNNs, which are trained on large scale datasets such as ImageNet [13]. Transfer learning technique provides a better way to avoid training CNNs from the scratch.

Fundamental motivation for transfer learning for machine learning discussion started in 1995 with NISP workshop on learning to learn [24] which was focused on finding methods to that retrain and reuse previously learned knowledge. Nowadays this technique used widely in image classification tasks [25], [26]. Some recent approaches [27] try the "Off-the-shelf" technique for classifying tiny data. They use the pre-trained CNN on a different task as a generic feature encoder and train a shallow classifier (such as SVM) by using these deep convolutional features. Razavian et al showed that their "Off-the-shelf" approach outperforms traditional classification approaches with a large margin [27]. Deep domain adaptation (transfer learning) approaches [28], [29] are widely used for classifying small data. These approaches are classifying small data by transferring the learned knowledge from a source task to a similar task. Ganin and Lempitsky [28] propose an unsupervised domain adaptation technique. Maxime et al [29] transfer the features from a pre-trained CNN and then fine-tune the network with the small amount of target data. Kokul et al [6] proposed an online domain adaptation technique for visual tracking by fine-tuning VGG-M network.

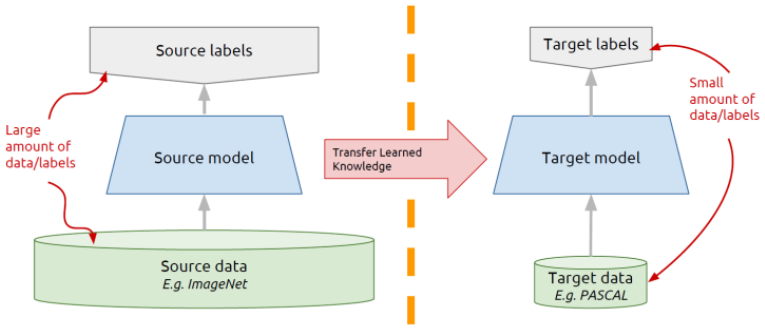

Fig. 3 Deep transfer learning diagram.

Nowadays, deep transfer learning has become the most popular solution for several applications [25], [26], which have limited data. As shown in Fig.3, these applications transfer the knowledge from a source model to a target model. In practice, the common way of deep transfer learning is using a pre-trained CNN as the source model, which is trained with large amount of data such as ImageNet [13]. As shown in Fig.3, source model is selected as a well-performing architecture, which showed high accuracy in source data. Target data could be similar or not to the source data. The objective of this technique is to train the target model with a little amount of target data by transferring learned knowledge of the source model. The hypothesis of transfer learning is that initial layers of source network are able to represent generic features and hence transfer to other tasks. The similarity between source and target data decides [23] the level of knowledge can be transferred. If source and target data are much similar, knowledge can be transferred from high number of source layers. If dissimilar, fewer numbers of layers can be transferred. Even though these transfer learning approaches are popular nowadays, there are very few studies that are conducted to analyse their performance and measure the level of knowledge transferring.

III. PROBLAM OVERVIEW

A. *Problem Specification*

As noted above, it is challenging to train a complex model such using only a small amount of training data. On the other hand it is unable to say the suitable architecture or the number of layers needs to be on a CNN model. The objective of this work is to conduct a comprehensive study on deep image classification techniques for tiny data and to analyse their performance on publically available benchmark datasets. As an initial step, we train CNN architecture directly for tiny data and measure the performance. Then we conduct the deep transfer learning and measure the performance at different levels of translation. In addition, we fine-tune the CNN architecture and measure the performance. We have selected two publicly available benchmark datasets for this study.

B. *Datasets*

We selected Caltech101 [30] and CIFAR10 [31] as tiny image data. ImageNet [13] dataset is selected as source data for deep transfer learning. Details of these datasets are given in Table 1.

TABLE I
DATASET COMPARISON (* INDICATES AVERAGE VALUES)

	Caltech101	**CIFAR10**	**ImageNet**
# Classes	101	10	1000
Image size	200 x 300 *	32 x 32	469 x 387 *
Images per class	40 to 800	10,000	120,000 *

As shown in Table 1, Caltech101 and CIFAR10 (target datasets) have a smaller number of images per classes compared to ImageNet (source dataset). Even though most class categories are same in target data and source data (such as bird, airplane and cat), image size, resolution and alignment are different. Test set of CIFAR10 contains exactly 1000 randomly-selected images from each class. We use 20% of randomly-selected images of each class to test the Caltech101.

IV. METHODOLOGY

A. Training from scratch

As the first step, we train CNN architecture from the scratch. We design the network based on VGG-M architecture. Our network takes input with the size of 32 x 32 and produces output as the number of classes in the dataset. In our architecture, convolutional layers are followed by ReLU activation [1]. Max pooling layers are placed in between convolutional layers. Convolutional layers are followed by three fully connected layers with the neurons of 512,256, number of classes (E.g. 10 for CIFAR10). Fully connected layers are interleaved with ReLU activation layer and dropout [32].

We conduct the scratch training evaluation process as follows:

1. The number of fully connected layers kept fixed.
2. The number of convolutional layers increased from small amount to high.
3. The whole network is trained with lower learning rate.
4. Classification accuracies are measured for the corresponding number of convolutional layers.

The high performing CNN architecture for CIFAR10 is shown fig.4.

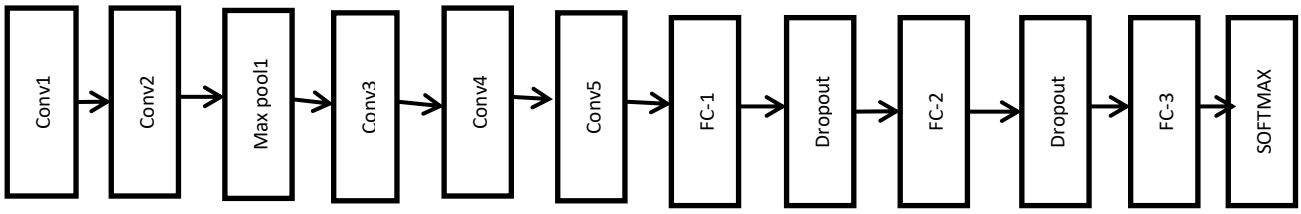

Fig. 4 Proposed CNN architecture for image classification from scratch

We conduct the training for each dataset through 100 epochs with the learning rate of 0.001. We have used Stochastic Gradient Descent (SGD) to optimize the training. The performance of the network is measured as average classification accuracy of the test set. The classification accuracy is measured as follows:

$$\text{Classification Accuracy} = \frac{\text{Number of true object Detactions}}{\text{Total number of test images}} \quad \text{Eq.1}$$

B. Training from deep transfer learning

In the second step of this study, we conduct the analysis of deep transfer learning based training. We have selected ImageNet and VGG-16 as the source data and the source network, respectively. Fig.5 shows the convolutional layer structure of VGG-16 architecture. We add three full connected layers with the convolutional layer structure of VGG-16. The number of neurons in fully connected layers is set as 512, 256 and the number of classes of each dataset. Fully connected layers are interleaved with ReLU activation and dropout. The Stochastic Gradient Descent (SGD) is used to optimize the network.

Fig.5 CNN layer structure of VGG-16 architecture. Kernel size and number of neurons are denoted in each layer.

In this step, we do not change the number of convolutional and fully connected layers. To measure the performance of transfer learning, we conduct the training process as follows:

1. The convolutional layers of VGG-16 are Re-initialized from top to bottom.

2. The re-initialized layers are trained (fine-tunned) with lower learning rate.
3. Remaining convolutional layers are frozen (kept fixed) throughout the training.
4. Classification accuracies are measured for corresponding number of fine-tunned layers.

We fine-tune the network through 100 epochs with the learning rate of 0.001. The network is trained with the batch size of 32 and SGD is used to optimize the training. We follow the E.q.1 to measure the classification accuracies.

V. RESULTS

A. *Implementation details*

This study is implemented in python with Keras [33]. This evaluation is conducted on four cores of 2.66 Intel Xeon with NVIDIA Tesla K40 GPU. The pre-trained VGG-16 model is obtained from Keras model zoo.

B. *Scratch training results*

We started the scratch training with a base network, which has two convolutional layers and three fully connected layers. Then the numbers of convolutional layers are increased one by one and for each architecture corresponding classification accuracies are measured.

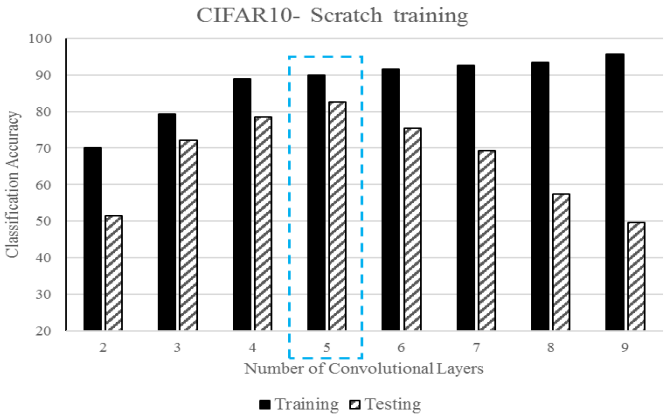

Fig. 6 Scratch training results of CIFAR-10 dataset. Training and testing accuracies are shown in adjacent columns. Best performing architecture is denoted with rectangle.

Fig.6 demonstrates the training and testing accuracies of scratch training for the CIFAR10 dataset. As can be seen from the graph, test accuracy is increased smoothly with the number of convolutional layers until the architecture reaches five layers. The maximum test and train accuracies are obtained at five-layer architecture with the values of 90 and 82.5 respectively. The accuracies start falling from six layer architecture. In addition, the difference between training and testing accuracies is increased while we include more convolutional layers.

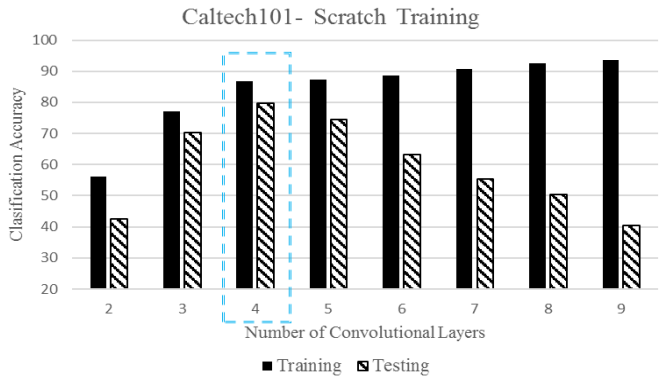

Fig. 7 Scratch training results of Caltech101 dataset. Training and testing accuracies are shown in adjacent columns. Best performing architecture is denoted with rectangle.

The scratch training results of Caltech101 is shown in Fig.7. The accuracy variation of this dataset is most similar to the results of CIFAR10. While comparing with CIFAR10, Caltech101 achieves the higher classification accuracy with fewer numbers of layers. The four convolutional layer architecture gives high training and test accuracies with the value of 86.8 and 79.6 respectively.

C. Transfer learning results

Fig.8 shows the classification accuracies of transfer learning for both datasets. We have used the VGG-16 architecture as the base network to evaluate the performance of transfer learning. The evaluation process starts by re-initialize and train the fully connected layers (this step is denoted as FC layers in the fig.8) of VGG-16. In the consequent steps, convolutional layers of VGG-16 are fine-tunned (re-initialised and trained) one by one and corresponding classification accuracies are measured. For example, the fine-tuned convolutional layer three means that the fully-connected layers and last three convolutional layers of VGG-16 are re-initialized and trained.

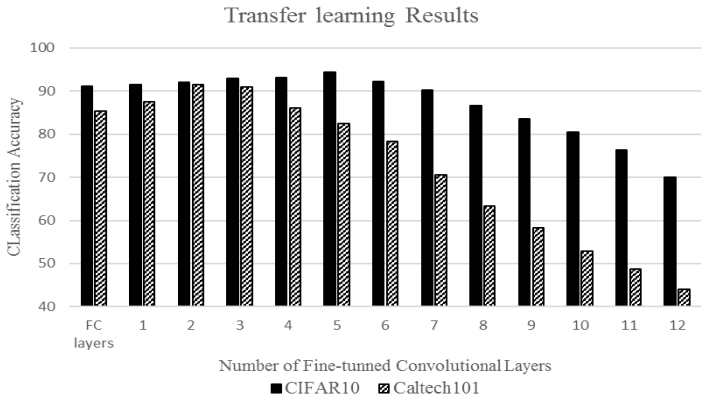

Fig. 8 Transfer learning results of CIFAR10 and Caltech101.

VI. **DISCUSSIONS**

Investigating the performance of convolutional neural networks for tiny image classification is important for many practical applications. We conduct this comprehensive study by analysis the performance of scratch (direct) and transfer learning (fine-tuning) based training on two publicly available tiny datasets.

Scratch training results (Fig.6 and Fig.7) clearly show that while training accuracy is increased with the number of convolutional layers, test accuracy starts falling after some layers. The main reason is that large CNN architectures fail to generalize the tiny data through a huge amount of parameters. Therefore the model is over-fitted to train data and train accuracy is increased with the number of layers. Because of the over-fitting, the difference between training and testing accuracies is increased in larger architectures. We can also see that CIFAR10 achieves higher classification accuracy with five layer architecture while Caltech101 gets high scratch training accuracy at four layer architecture. Caltech101 is almost seven times smaller than CIFAR10 and therefore face the over-fitting issue even at smaller architectures.

Fig.8 clearly shows that transfer leaning based training increase the classification accuracy for both datasets while comparing with scratch training. Since both datasets are much similar with ImageNet (source dataset), they achieve high accuracies with fewer number of fine-tunned layers. Similar to scratch training, accuracies of both datasets are decreased with the number of fine-tunned layers because of the over-fitting issue. Since the Caltech101 dataset fails to train a large number of re-initialized layers, its classification accuracy drops suddenly than the CIFAR10 for large number of fine-tunned layers.

In transfer learning, layers of source network are frozen and re-initialized layers are trained with target data. At fine-tuning, Caltect101 achieves high accuracy with an architecture, where last three convolutional layers are re-initialized. CIFAR10 achieves high accuracy while last five convolutional layers are re-initialized. We investigate these high-performing architectures further. Instead of freezing source layers, we train the whole network (source layer + re-initialized layers) with target data. We have used very low learning rate for source layers (0.0001) and low learning rate (0.001) for re-initialized layers. The results are compared in Table II.

TABLE II:
High performing results for different training approaches (HIGH accuracies are shown in bold)

Method	**Caltech101**	**CIFAR10**
Scratch Training	79.6	82.5
Fine-tunning (re-initialized layers only)	**91.4**	94.4
Fine-tunning (whole network)	87.8	**95.52**

As shown in Table II, Fine-tuning whole network increases the accuracy of CIFAR10 to 95.52 while decreasing the accuracy of Catech101 as 87.8. Since Caltech101 is a much smaller dataset for training the whole network (16 layers), fine-tuning whole network decreases the accuracy considerably. On the other hand, training source network with very smaller learning rate increases the classification accuracy of CIFAR10.

VII. CONCLUSION

In this paper, we conduct a comprehensive study to analyse the classification performance of convolutional neural networks for tiny data. This study is conducted on two publicly available datasets. This study evaluates the performance of two major approaches: scratch training and transfer learning based training. Classification accuracies of scratch training are measured for different architectures by changing the number of convolutional layers. Performance of transfer learning is measured by verifying the number of fine-tuned layers. Based on this study, we can conclude that fine-tuning is the best way to classify the tiny data. It gives high accuracies while the source and target data are much similar. In addition, this study found that accuracies of tiny datasets can be increased by training the source layers with a very low learning rate. These results will be very useful for many practical applications.

VIII. REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. Hinton, "Image{N}et classification with deep convolutional neural networks," in *Proceedings of the Advances in neural information processing systems*, pp. 1097-1105, 2012.

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

[3] Y. Lin, F. Lv, S. Zhu, M. Yang, T. Cour, K. Yu, *et al.*, "Large-scale image classification: fast feature extraction and svm training," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pp. 1689-1696, 2011.

[4] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3128-3137, 2015.

[5] T. Kokul, C. Fookes, S. Sridharan, A. Ramanan, and U. Pinidiyaarachchi, "Gate connected convolutional neural network for object tracking," in *IEEE International Conference on Image Processing (ICIP)*, pp. 2602-2606, 2017.

[6] T. Kokul, A. Ramanan, and U. A. J. Pinidiyaarachchi, "Online multi-person tracking-by-detection method using {ACF} and particle filter," in *Proceedings of the IEEE Seventh International Conference on Intelligent Computing and Information Systems (ICICIS)*, pp. 529-536, 2015.

[7] A. Geiger, M. Lauer, C. Wojek, C. Stiller, and R. Urtasun, "3{D} {T}raffic {S}cene {U}nderstanding {F}rom {M}ovable {P}latforms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, pp. 1012-1025, 2014.

[8] N. Rasheed, S. A. Khan, and A. Khalid, "Tracking and {A}bnormal {B}ehavior {D}etection in {V}ideo {S}urveillance {U}sing {O}ptical {F}low and {N}eural {N}etworks," in *Proceedings of the Advanced Information Networking and Applications Workshops* pp. 61-66, 2014.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv preprint arXiv:1512.03385*, 2015.

[10] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong, “Locality-constrained Linear Coding for Image Classification,” in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010, pp. 3360–3367.

[11] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proceedings of the Computer Vision and Pattern Recognition (CVPR)*, pp. 886-893 2005.

[12] T. Lindeberg, "Scale invariant feature transform," *Scholarpedia*, vol. 7, p. 10491, 2012.

[13] J. Deng, W. Dong, R. Socher, L. J. Li, L. Kai, and F.-F. Li, "Image{N}et: {A} large-scale hierarchical image database," in *Proceedings of the Computer Vision and Pattern Recognition (CVPR)*, pp. 248-255, 2009.

[14] H. C. Shin *et al.*, “Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning,” *IEEE Trans. Med. Imaging*, vol. 35, no. 5, pp. 1285–1298, 2016.

[15] K. Fukushima, "Neocognitron: A hierarchical neural network capable of visual pattern recognition," *Neural networks*, vol. 1, pp. 119-130, 1988.

[16] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1-9, 2015.

[17] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *arXiv preprint arXiv:1312.6034*, 2013.

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

[19] W. Train, D. Architectures, I. Representations, and S. Features, *Learning Deep Architectures for AI By Yoshua Bengio*, vol. 2, no. 1. 2009.

[20] Available: http://cs231n.stanford.edu/

[21] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Computer vision and pattern recognition (CVPR)*, pp. 3642-3649, 2012.

[22] D. C. Ciresan, U. Meier, J. Masci, L. Maria Gambardella, and J. Schmidhuber, "Flexible, high performance convolutional neural networks for image classification," in *International Joint Conference on Artificial Intelligence*, p. 1237, 2011.

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

[24] R. Caruana, T. Mitchell, D. Pomerleau, T. Dietterich, and O. State, “Multitask Learning,” in *Learning to Learn*, no. September, Springer, Boston, MA, 1997, pp. 95–133.

[25] M. Geng, Y. Wang, T. Xiang, and Y. Tian, "Deep transfer learning for person re-identification," *arXiv preprint arXiv:1611.05244*, 2016.

[26] G. M. Y. Dauphin, X. Glorot, S. Rifai, Y. Bengio, I. Goodfellow, E. Lavoie, *et al.*, "Unsupervised and transfer learning challenge: a deep learning approach," in *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, pp. 97-110, 2012.

[27] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: an astounding baseline for recognition," in *Computer Vision and Pattern Recognition Workshops (CVPRW), ,* pp. 512-519, 2014.

[28] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," *arXiv preprint arXiv:1409.7495*, 2014.

[29] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pp. 1717-1724, 2014.

[30] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories," *Computer vision and Image understanding*, vol. 106, pp. 59-70, 2007.

[31] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009.

[32] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, “Improving neural networks by preventing co-adaptation of feature detectors arXiv : 1207 . 0580v1 [cs . NE] 3 Jul 2012,” pp. 1–18.

[33] Chollet, François. "Keras.", 2015.