A Pre-Trained Vs Fine-Tuning Methodology in Transfer Learning

Neeraj Gupta
Department of Computer Engineering and Applications,
GLA University, Mathura, Uttar Pradesh, India
Email: neeraj.gupta@gla.ac.in

Abstract. Transfer learning from a pre-trained and fine-tuning methodology has been utilized for the image classification. In this paper, we classify images of cats and dogs. It is much faster and easier than training from scratch using a pre-trained network with transfer learning. This saved network was trained previously on a huge dataset and known as a pre-trained model. The pre-trained model can be used in two ways, either it is used as it is, second using the transfer learning approach to adapt the pre-trained network to a specific goal. Since the idea for image classification using transfer learning is that it is trained on the general and large dataset and further this model will be efficiently worked on the visual perception task of the real world. In fine-tuning, instead of tuning with the weights of generic feature map of based model, it is tuned accordingly to the dataset. In this paper, we take benefit of these learned feature maps deprived of having to start from scratch by utilizing a deep learning technique on a huge dataset and analyze the various methodology. The experimental results show that fine-tuning methodology outperformed over the pre-trained with feature extraction on accuracy.

Keywords: ConvNet, Convolutional Neural Network (CNN), Transfer learning

1. Introduction
In computer vision, based on visual content images are classified in the task of Image classification and still, it is a challenging task to classify image robustly. An image-classification task, a pre-trained deep learning model is a saved network that was earlier trained on a huge dataset. This pre-trained model could be used in two ways, first, it is performed as it is, second using the transfer learning approach to adapt the model for the specific task. Transfer learning [1] is a machine learning’s research problem, which focuses on learning knowledge added during solving one task and applying it to another similar task. The main idea behind the transfer learning approach for classification of the image is that if a deep learning model is trained on a huge dataset. Further, this model will successfully and efficiently solve the generic task of the visual world by using the learned feature maps on the huge dataset.

In this paper, we have utilized a pre-trained model and customize it in two ways:
Feature Extraction: It is used in representing the previously learned network for extracting the features from new data. We have added a new classifier, at the top of the pre-trained network model which is trained from scratch. We can reutilize the feature maps from the pre-trained network which is trained on the huge dataset. We need not have to re-train the model. Generally, for classifying the image the base convolutional network already has the features. However, specific to the original classification task, the final classification of the pre-trained model has been utilized.

Fine-Tuning: It is a process in which a few of the top layers are unfreeze in a frozen model base and further combinedly train both the last layers of the frozen base model and newly added classifier layers. This concept allows us to “fine-tune” the pre-trained model by utilizing the higher-order representation of features for resolving the particular task.

The rest of this paper is organized as follows. Section 2, describes the technique of transfer learning. Section 3, illustrate the experimental results and discussion on the performance of the proposed deep learning classifiers. Finally, Section 4, illustrate the conclusion and future aspects.

2. Method

In this paper, Convolution Neural Networks have been utilized for transfer learning techniques [3-8]. The workflow is shown in Figure 1. Data preprocessing is the initial step of the model. In these steps first, it loads pre-defined data and cached it, and returns the dataset object. For deploying data and piping it into our CNN network model efficient and powerful methods. The whole dataset is split into three categories train, test and validation and amount of data is 80%, 10%, 10% for them respectively. For the training the dataset object has a pair of (image, label) where the label is scalar, image is having three channels and variable in size. To create uniformity in the dataset, we resize image and rescale the input channels of all the three categories (train, test, validation). The image is resized to 160 and a range of rescaling of the input channels is [-1,1].

2.1 MobileNet V2 as Base Model:

At Google, the MobileNet V2 model [2] has been developed and we have used it to create the base model. ImageNet dataset [5] has been utilized to train this model. This is huge dataset having 1000 classes and the total number of images is 1.4 Million. A wide range of groupings like syringe, jackfruits etc. has been available in ImageNet dataset [5]. However, for our specific dataset of dogs
and cats [7], this knowledge base has been utilized for classification techniques.

Initially, for feature extraction, we choose the layer of MobileNet V2 [2]. The topmost or very last layer of classification is not so much useful. In place of it, the very last layer before the flatten operation can be used as a common practice. This layer is known as the “bottleneck layer”. As compared to the top layer generally “bottleneck layer” features preserve more. Further, we have instantiated a MobileNet V2 deep learning model [2].

| Input | Operator | t | c | n | s |
|-------|----------|---|---|---|---|
| 224^2 × 3 | conv2d | - | 32 | 1 | 2 |
| 112^2 × 32 | bottleneck | 1 | 16 | 1 | 1 |
| 112^2 × 16 | bottleneck | 6 | 24 | 2 | 2 |
| 56^2 × 24 | bottleneck | 6 | 32 | 3 | 2 |
| 28^2 × 32 | bottleneck | 6 | 64 | 4 | 2 |
| 14^2 × 64 | bottleneck | 6 | 96 | 3 | 1 |
| 14^2 × 96 | bottleneck | 6 | 160 | 3 | 2 |
| 7^2 × 160 | bottleneck | 6 | 320 | 1 | 1 |
| 7^2 × 320 | conv2d 1x1 | - | 1280 | 1 | 1 |
| 7^2 × 1280 | avgpool 7x7 | - | - | 1 | - |
| 1 × 1 × 1280 | conv2d 1x1 | - | k | - | - |

Figure 2: Architecture of MobileNet V2

This model has been loaded with weights pre-trained on ImageNet [5]. The statement include_top = False as an argument is used to remove the topmost classification layer and extract the features. Each image of size 160 x 160 x 3 is converted into the block of feature having size 5 x 5 x 1280 using feature extractor.

2.2 Feature Extraction

It is the phase in which we freeze the base model as depicted in the previous phase and only utilize the feature extractor. Moreover, we have added a classifier on the top of the base model and further train it.

2.2.1 Freeze the Base Model

Before compiling and training the model, it is vital to freeze the convolutional base. In the process of freezing does not update the weights during the training of model for a given layer. This process is done by setting layer.trainable = False. However, there are various layers in MobileNet V2 [2], all the layers of it have been frozen by implementing the trainable flag to False for the entire model.
2.2.2 Addition of Classifier

To produce the prediction from the feature block for the spatial location of 5x5. We have been utilizing a `tf.keras.layers.GlobalAveragePooling2D` layer to change the features into a vector of size 1280 element for an image. To get a single prediction per image, we have applied a `tf.keras.layers.Dense` layer \[10\] to alter these features. Since we do not require any activation function. A logit or raw prediction values have been used here. The negative value predicts class 0 and positive value predict class 1.

2.2.3 Model Compilation and Training

Before training the model it must be compiled. As Dogs vs Cat is a binary classification problem, it requires binary cross-entropy loss. Linear output is generated by the model using `from_logits = True`. MobileNet V2 \[2\] models’ 2.5 million parameters are frozen, and in the Dense layer still, there are 1.2K trainable parameters. The summary of CNN Model is shown in Figure 4. These can be further distributed between weights and biases. The accuracy of 96% is achieved after training it for 10 epochs.

2.3 Fine-Tuning
As described in the feature extraction implementation, we have been training the small number of layers on the top of the base model MobileNet V2 [2]. In this training, the weights were not updated of the pre-trained network. Further to enhance the performance, there is a concept of “fine-tune” is utilized. In this concept, we have trained the weight of the top layers of the base model in conjunction with the training of the added classifier. In fine-tuning, instead of tuning with the weights of generic feature map of based model, it is tuned accordingly to the dataset. However, we have implemented to fine-tune a few top layers rather than the full base model. In the convolution neural network model, the initial few layers learn general and simple features while the higher layers learn the special features. The summary of the CNN model using fine-tuning is shown in Figure 5.

![Figure 5: A Summary of CNN model using “Fine-Tuning”](image)

Moreover, the fine-tuning take advantage of the specialized features which are available at the higher layer of the base model. Further for implementing the fine-tuning concept, the higher-order layers of the base model are un-freeze and fixed the initial layers of the base model as untrainable. Later, we have to recompile the model to adapt the changes and restart training the model. This “fine-tuning” approach has enhanced the accuracy by a few percentage points.

3 Results and Discussion

3.1 Datasets

In this paper, we have used for transfer learning a cats_vs_dogs images standard dataset collected from Microsoft [7]. It is a collection of a large set of images of cats and dogs. The dataset sample images are shown in Figure 2. The size of images is 160 x 160 pixels. The dataset is having 2 classes separated in different folders. These classes are “CAT” or “DOG”. The size of the whole dataset is 786.68 MB. The source code of dataset is tfds.image_classification.cats_vs_dogs.CatsVsDogs [10].
3.2 Evaluation Metric

Evaluation metrics for using transfer learning approach using pre-trained Convolution Neural Network models is validation accuracy. It is a standard method in the literature which show the percentage of correctly classified images of the dataset. It is simply the summation of true positives (TP) and true negatives (TN) divided by the total values of confusion matrix components. The mathematical formula of Accuracy is given in Equation 1.

\[
\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}
\]  

3.3 Implementation Details

The steps of the convolution neural network model were implemented using Python 3.7.6 on a workstation having Intel Xeon 5222 3.8 GHz processor and the MobileNet V2 [2] experiments were implemented in Python (version 3.7.6) using the various libraries like Kera, Tensorflow [10] etc., the workstation having GPU with dual NVIDIA™ Quadro RTX4000 8 GB GDDR5X.

3.4 Quantitative Analysis of CNN Model

In this paper, MobileNet V2 Models [2] has been utilized for binary classification of Dog vs Cat. Quantitative analysis has been performed for identifying the outperformer approach among pre-trained feature extraction vs fine-tuning.

3.4.1 Training and Validation Accuracy/Loss Curves

The training and validation accuracy/loss curves are shown in Figure 7(a) when implementing the pre-trained MobileNet V2 model [2] as a base and having static feature extractor. Figure 7(b) also shows the training and validation accuracy/loss graph when implementing the MobileNet V2 with Fine-tuning last few top layers and classifier on the top of it. The training loss is much less than validation loss as a result produce some overfitting. It also gets some overfitting as the new dataset of Dogs vs Cats [7] is relatively small as compared to ImageNet dataset [5] of MobileNet V2. The 98% accuracy has been attained after fine-tuning the base model.
In this paper, a MobileNet V2 Model [2] has been used for binary classification of cat vs dog images. A benchmark dataset of the cats_vs_dogs [7] images has been utilized for experimentation and study. We have implemented the models, training and testing have been done in two ways. First using a pre-trained network with feature extraction has total parameter 2,259,265 including trainable and Non-trainable parameters. Here trainable parameters are 1,281 and non-trainable parameters are 2,257,984. Second, using fine-tuning techniques has total parameter 2,259,265 including trainable and Non-trainable parameters. Here trainable parameters are 1,863,873 and non-trainable parameters are 395,392. The evaluation metric used here is Training and Validation Accuracy. The experimental result of MobileNet V2 with pre-trained feature extraction is as the training accuracy is 0.9483 and validation accuracy is 0.9514 and MobileNet V2 with fine-tuning the training accuracy is 0.9967 and validation accuracy is 0.9802 as shown in Table 1.

Table 1: Model Parameters & Accuracy

| CNN Model                                      | Total Parameters | Trainable Parameters | Non-Trainable Parameters | Accuracy Training | Accuracy Validation |
|------------------------------------------------|------------------|----------------------|--------------------------|-------------------|---------------------|
| MobileNet V2 with Pre-Trained Feature Extraction | 2,259,265        | 1,281                | 2,257,984                | 0.9483            | 0.9514              |
| MobileNet V2 with Fine-Tuning                   | 2,259,265        | 1,863,873            | 395,392                  | 0.9967            | 0.9802              |
4 Conclusion
In this work, we have analyzed the pre-trained model for binary classification task of cat vs dog. It can be used further used in two ways: the first method is feature extraction using a pre-trained model. In this method the pre-trained model is “frozen” and during the training, only weights are updated for the added classifier at the top of the base model. The second method is to fine-tuning the pre-trained model. In this method, few top layers having the high-level features are added with the classifier to train the weights. It has been observed from the experimental results that MobileNet V2 with pre-trained feature extraction achieve 0.9515 while MobileNet V2 with fine-tuning achieve 0.9802 validation accuracy. The fine-tuning techniques outperformed over the pre-trained with feature extraction on accuracy.

References
[1] Weiss, K., Khoshgoftaar, T. M., & Wang, D. (2016). A survey of transfer learning. Journal of Big data, 3(1), 9.
[2] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4510-4520).
[3] Wang, L., Guo, S., Huang, W., & Qiao, Y. (2015). Places205-vggnet models for scene recognition. arXiv preprint arXiv:1508.01667.
[4] Tang, Y., Wu, X., & Bu, W. (2016, October). Deeply-supervised recurrent convolutional neural network for saliency detection. In Proceedings of the 24th ACM international conference on Multimedia (pp. 397-401).
[5] Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009, June). Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition (pp. 248-255).
[6] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).
[7] Gupta, N., & Jalal, A. S. (2019). Integration of textual cues for fine-grained image captioning using deep CNN and LSTM. Neural Computing and Applications, 1-10.
[8] Gupta, N., & Jalal, A. S. (2020, February). Text or Non-text Image Classification using Fully Convolution Network (FCN). In 2020 International Conference on Contemporary Computing and Applications (IC3A) (pp. 150-153). IEEE.
[9] https://www.microsoft.com/en-us/download/details.aspx?id=54765
[10] https://www.tensorflow.org/