JOURNAL OF
EMERGING INVESTIGATORS

Article

Transfer Learning for Small and Different Datasets: Fine-Tuning A Pre-Trained Model Affects Performance

Ananya Gupta[1] and Meghna Gupta[2]
[1] The International School Bangalore, Bangalore, Karnataka, India
[2] Adarsh Palm Retreat, Bangalore, Karnataka, India

SUMMARY

Machine learning and deep learning algorithms are rapidly becoming integrated into everyday life. Whether it is in your face-ID to unlock your phone or the detection of deadly diseases like melanoma, neural networks have been traditionally designed to work in isolation to achieve amazing tasks once thought impossible by computers. However, these algorithms are trained to be able to solve extremely specific tasks. Models have to be rebuilt from scratch once the source and target domains change and the required task changes. Transfer learning is defined as a field that leverages learnings and weights from one task for related tasks. This process is quite smooth if one has enough data and the task is similar to the previous, already learnt task. However, research on when these two conditions are not met is scarce. The purpose of this research is to investigate how fine-tuning a pre-trained image classification model will affect accuracy for a binary image classification task. Image classification is widely used, and when only a small dataset is available, transfer learning becomes an important asset. Convolutional neural networks and the VGG-16 model trained on Imagenet will be used. Through this study, I am investigating whether there are specific trends in how fine-tuning affects accuracy when used for a small dataset which is dissimilar from Imagenet. This will allow for the beginning of investigating quantifiable methods to train a model when using Transfer Learning techniques.

INTRODUCTION

The increasing use of artificial intelligence (computer systems that mimic human intelligence) (1), or AI, in myriad tasks and fields (2) has created a need for large amounts of data collection. Weights and biases are learnable parameters of a machine learning model which improve while the model trains on training data to provide an optimum model and correct predictions (3). Supervised learning is when the neural network is fed labelled data known to be correct (4). However, in this natural, chaotic world, the ability to collect large amounts of data and labelling data points is tedious, often restricted, and uses many resources which are not readily available. This could reduce one's ability to use supervised learning, one of the most common and powerful methods to train classification-based machine learning models. Supervised learning works by, for example, provide an image of a cat as input and the label "cat" as the output, and an image of a dog with the "dog" label. With this input and output, the network calculates loss based on its predictions from its weights. The network then uses an algorithm, to modify the weights and biases to better map the input to the correct output and to minimize loss. Hence, it predicts whether an image is a "cat" or "dog". As AI and machine learning tasks become more common in the general public, the field must reduce the need for large data to create accurate models. This would promote innovation that is truly democratic and open, free from the burden of having to amass large datasets. Doing so, AI is made less niche and more accessible for commercial and social use.

AI has been growing exponentially, getting closer and closer towards general AI which is AI that can learn to do myriad tasks, unlike specialized AI made for one specific task (1). One large step towards this has been the development of transfer learning. Transfer learning techniques are being used extensively due to their abilities to provide high accuracy by leveraging pre-learnt weights from a pretrained model to another (5). This field is significant because it reduces dependence on large datasets without compromising accuracy (6). Traditional machine learning techniques are based on the model of isolated, single task learning wherein knowledge from a past task is not leveraged for other tasks. Transfer learning, on the other hand, relies on previously learnt tasks which could allow the learning process to be faster, have higher accuracy and require less training (6). Transfer learning allows a source task to affect the inductive bias of the target task.

Fine-tuning is the process of unfreezing different layers in the base model (pre-trained model) (5). We refer to nomenclature in transfer learning of "unfreezing" which means "making a base layer trainable.''. Usually, transfer learning is conducted with deep neural networks by fine-tuning a pretrained model on the source task using data from the target task. This study investigates how fine-tuning a model (freezing and unfreezing different layers) will affect the performance (measured using accuracy) for small and dissimilar datasets. The VGG16 convolutional neural network, which was trained on the ImageNet dataset (6), is used as a base model for this investigation. A convolutional neural network is merely a deep learning algorithm which

Unfrozen Layer	5	4	3	2	1	0
Training Accuracy	**99.69%**	**99.69%**	**96.52%**	**91.99%**	**88.36%**	**87.76%**
Validation accuracy	**86.67%**	**90.00%**	**79.87%**	**76.67 %**	**73.47%**	**69.73%**

Table 1. Unfreezing lower layers causes validation accuracy to first increase and then decrease. It also causes training accuracy to stagnate and then decrease. These values were based on analogous training conditions with the only difference being which layer was unfrozen. These values are after 100 epochs of training.

takes in an input image, assigns importance (learning weights and biases) to various aspects or objects in the image and is able to differentiate one from the other. The target task is to classify cartoon illustrations of textbooks as male or female, which is quite different from the source task of the base model which is classifying objects (10). The dataset of 640 training images is also dissimilar to the Imagenet dataset, which is an important factor in this study. While other research has focused on how performance varies with fine-tuning, this investigation focuses on specifically small and different datasets (11-12). The size of training data used here is neither at the extreme of one-shot learning or the large datasets in deep neural networks. Hence, through these experiments we try to investigate whether quantifiable trends are noticed in how performance varies with fine-tuning patterns. We predict that quantifiable patterns will be noticed, as performance will first increase and then gradually decrease.

RESULTS

The pre-trained model used in this study is the VGG16 convolutional neural network, which was trained on the ImageNet dataset (6). On top of the base VGG16 model, further convolutional neural network layers are built. This specific experimental setup was chosen due to the widespread use of VGG16 in the Transfer Learning corpus, thus becoming a widely accepted model. By fine-tuning different layers and training with this small, dissimilar dataset we have found that the validation accuracy follows a clear trend with layers becoming unfrozen (**Table 1**). Validation accuracy first increases from 86.67% when layer 5 is unfrozen to 90.00% as the lower layer, layer 4 is unfrozen. This trend then shifts as validation accuracy continues decreasing when lower layers are unfrozen. There is a linear trend observed once layer 3 is reached, with validation accuracy falling linearly up to layer 0 being unfrozen. Hence, there is an initial increase and then a gradual fall in validation accuracy (**Figure 1**).

The training accuracy follows a distinct trend as layers become unfrozen (**Table 1**). Training accuracy first stagnates for layers 5 and 4 unfrozen, with an accuracy of 99.69%. After this plateau, there is a linear decrease in the training accuracy until layer 1. This is a relatively steep decrease. After reaching layer 1, there is a fall in the rate of decrease of the training accuracy. This means that the decrease per layer becomes lesser at the last data point, between layer 1 and 0. Hence, the graph follows the phases- stagnation, steep and linear decrease, and finally a gradual linear decrease. These phases are attained as lower and lower layers are unfrozen (**Figure 1**).

DISCUSSION

The trend followed by the validation accuracy and training accuracy is quite similar as seen in (**Figure 2**). The similar bell-shaped trends for both portray that measures of performance change in similar patterns as lower layers are unfrozen. This corresponds to the weights and biases learnt, allowing for a justified analysis since it conveys that the model is not overfitting as the validation and training accuracy have a similar trend. Overfitting is when the model only works well on a particular dataset (such as its train dataset) but is unable to perform well on other datasets (such as the validation or test dataset) (7). Since our validation and training accuracies

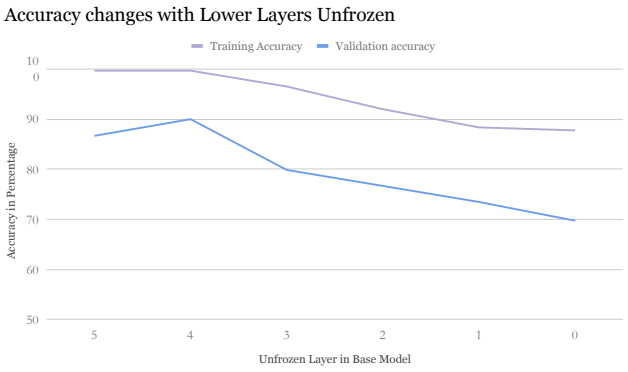

Figure 1. Unfreezing lower layers causes validation accuracy to first increase and then decrease. It also causes training accuracy to stagnate and then decrease. A line of the data of **Table 1** was plotted using linear interpolation. This interpolated line between each consecutive data point is used only to analyze the rate of change of accuracy with lower layers unfrozen for the discrete data values.

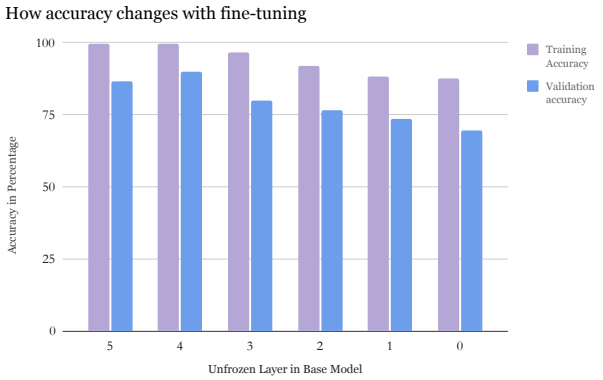

Figure 2. A bar graph showing how unfreezing lower layers causes validation accuracy to first increase and then decrease. It also causes training accuracy to stagnate and then decrease. Results of **Table 1** are used. Both types of accuracies were plotted to enable comparisons, portraying that the model is not overfitting.

are similar, it is fair to say that the model is not overfitting. Alongside, we note that the dataset used is relatively small, at only 640 training images, and is very different from the original Imagenet dataset. This serves the purpose of this study to investigate small datasets that are different from the original dataset. Data pre-processing is conducted in the form of Keras's data augmentation techniques which enables greater diversity of data available for training models, improving generalizations.

When only layer 5 is being fine-tuned, it is known that the weights learnt by the pre-trained model are more for the specific target task and domain of binary classification of male and female cartoon caricatures. This specificity is why it has the highest training accuracy and a high validation accuracy. This conclusion that specific training is better for a small dataset with data dissimilar from the ImageNet data is explained by the fact that there are very specific differences in the source and target domains. What this means is that for an unfamiliar task for the pre-trained model, general features such as edges are common but specific differences such as hair outline are extremely different, which makes specific training a necessity. However, its validation accuracy is lesser than when only layer 4 is being fine-tuned. This is because there is a fine line between being specific and being extremely specific wherein the general features are learnt incorrectly or become counterproductive. When only layer 4 is being fine-tuned, a great balance between the specificity and generality is struck. When taking a deeper look into the function of Layer 4 and 5 in-terms of specificity of weight and feature learning, this distinction becomes lucid.

Layer 5 has the potential to have higher validation accuracy if there was a greater amount of data, as this would lead to better generalizations.

When only layer 3 is unfrozen, there is a marked fall in training accuracy and validation accuracy compared to the fall in performance between unfreezing layers 5 and layer 4. This steep and sudden fall is at the hands of a shift in learning to more general weights. The weights learnt are not specific to the target task, causing a fall in training accuracy and validation accuracy as the model is not well trained for this dissimilar dataset. Layer 2 and 1 follow a similar trend of falling accuracies as less specific features and weights are learnt. Accuracies could be increased if there was a larger, more similar dataset, but for a small and dissimilar dataset this trend is followed while fine-tuning lower layers.

Finally, if no layers are fine-tuned, training accuracy and validation accuracy decreases once again. This is because the dataset is dissimilar to the ImageNet dataset. Hence, due to dissimilarities, the pre-trained model trained on ImageNet would not perform well on the current task. There are no updates to the weights, so the model is not learning new weights. Thus, it performs poorly for this dissimilar dataset.

Therefore, the results support the hypothesized trend that performance will first increase and then gradually decrease, as lower layers are left unfrozen. These results and corresponding theoretical justifications support the original hypothesis. With these results and hypothesis, we have found a quantified trend in how performance varies with fine-tuning as lower layers are unfrozen. These results on optimizing the fine-tuning combination to obtain higher model performance will allow better transfer learning models to be created for small, dissimilar datasets. One could further expand the scope of this investigation by exploring how performance changes with different combinations of unfrozen layers (specifically for VGG16, one could explore 32 combinations of 5C5 which gives 32). One could also perform similar investigations using different pre-trained models such as Inception, ResNet, VGG19 and so on to obtain the optimum fine-tuning combinations for these base models since this investigation focuses on the VGG16 base model.

MATERIALS AND METHODS

The VGG16 convolutional neural network pre-trained model, which was trained on ImageNet, was used for this experiment. VGG16 is a convolutional neural network model that consists of 16 weight layers, 13 convolutional layers, and 3 fully connected layers with 138 million total number of parameters (8). The model achieved 92.7% test accuracy in ImageNet (8). It has slowly become one of the most popular pre-trained models used, especially in Transfer Learning tasks, alongside models like ResNet, VGG19, and many more. It was trained on the ImageNet dataset which is a dataset of over 15 million labeled high-resolution images belonging to roughly 22,000 categories. The images were collected from the web and labeled by human labelers using Amazon's Mechanical Turk crowd-sourcing tool, ImageNet, which consists of variable-resolution images. Therefore, the ImageNet images were down sampled to a fixed resolution of 256×256. Given a rectangular image, the image is rescaled and cropped out the central 256×256 patch from the resulting

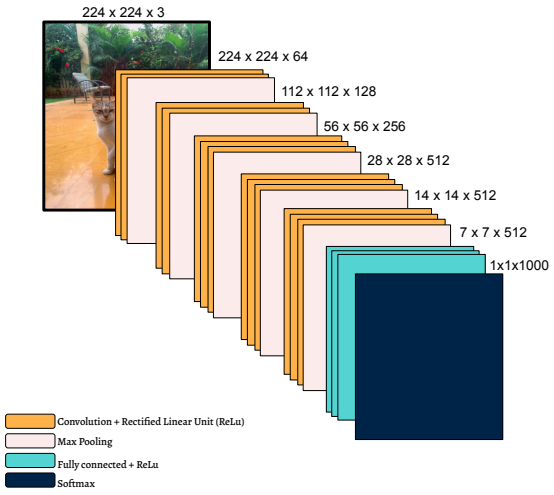

Figure 3. The VGG16 convolutional neural network pre-trained model architecture is detailed. The different types of layers, along with the input shapes are portrayed. A legend was also used to visualize how the model was built. (8)

image (9). The model architecture of VGG16 is depicted in (**Figure 3**) (6, 17).

Our dataset contains 2 classes - male and female textbook illustrations. It consists of 640 training images and 65 validation images. It was created manually since such a dataset does not exist to date, and alongside is dissimilar to the ImageNet dataset (which is a key part of this investigation) (10). We further confirmed that the images are dissimilar by attempting to classify the illustrations using purely the VGG16 model, which produced incorrect predictions. Alongside, after going through the categories classified in ImageNet, it is known that the dataset used in this investigation is different. We built the dataset by taking images from the Karnataka State Board textbooks in India, and our dataset can be seen here (10). Manual data collection was conducted to ensure uniqueness of the dataset. The data was split into 335 images of female illustrations and 337 male illustrations, to prevent any sampling bias. First, all the images are resized to 150x150 pixels, with 3 channels of red, green, and blue. The data is split into 3 datasets: training data, validation data, and test data. The images are converted to arrays, and each image is given a class of either male or female, i.e., 1 or 0. The images with their corresponding classes are shuffled. Shuffling helps reduce variance and reduce overfitting. It ensures that the gradient descent algorithm does not become "stuck" to a certain local minimum which prevents it from reaching the global minimum, by making sure that the input X changes with each iteration and is not static (13).

Data augmentation is effective when using small datasets in convolutional neural networks as it helps in reducing overfitting through rotation, rescaling, flip, etc. (14). For this project, the ImageDataGenerator factors of Keras are used for data augmentation and normalization of training dataset, which are detailed in **Table 2** below.

The ImageDataGenerator factors used for data augmentation and normalization of validation dataset are given by a Rescale factor of 1/255. These several parameters allow for powerful data augmentation techniques. The data augmentation techniques are applied on the data (**Figure 4**).

According to which layers are to be fine-tuned (or unfrozen), the model is built with setting, for example, block 5

Type	*Factor*
Rescale	1/255
Zoom Range	0.3
Rotation Range	50
Width Shift Range	0.2
Height Shift Range	0.2
Shear Range	0.2
Horizontal Flip	True
Fill Mode	Nearest

Table 2. The different data augmentation techniques used for the training dataset with their corresponding numerical factor. This is a part of our data pre-processing and is especially useful for small datasets like ours.

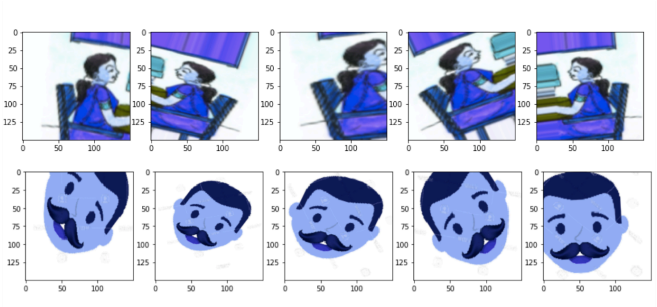

Figure 4. Sample of data augmentation conducted on the images in the dataset. The augmentation techniques are described in Table 2. Random combinations of data augmentation techniques were used. The image above displays examples of images from the two classes - 'male' and 'female'.

trainable as true. A Keras with tensorflow Sequential model is built, and the pre-trained model with the parameter of "trainable" set to "true" or "false" is added to the Sequential model. On top of that, convolutional neural network layers are added, as seen in **Table 3**. First, a Dense layer with a Rectified Linear Unit (ReLu) activation function is added. The Dense layer is a fully connected layer, which means that each neuron in a layer is connected to those in the following layer, and the parameters used here were 512 units (units are the output dimensions) and input dimension is the same as input shape. ReLu is a simple calculation that relays back the value given as the input directly or returns 0.0 if the number is ≤ 0.0. To train deep neural networks like the one used here and to use stochastic gradient descent with backpropagation of errors, an activation function is needed which can behave as if it were a linear function but in reality is a nonlinear function pertaining to complex relations in the data from which the model learns weights. The function must also allow greater sensitivity to the activation sum input and avoids easy saturation. Hence, the answer to all these criteria is the Relu activation function (15). It acts linear for all values greater than 0 but is actually a non-linear graph since it is flat for numbers ≤ 0.0. This is why an activation function is used for the Dense layer. Then, a Dropout layer is added with a 0.3 dropout rate. The function of the Dropout layer is to, at each step during training, randomly set input units to 0 with a frequency of 0.3. This is key to prevent overfitting and helps ensure better regularization. After this, another Dense layer followed by a Dropout layer is added, with the same parameters as specified before. The last layer added is a Dense layer with one unit to provide a one-dimensional output, along with the sigmoid activation function. The sigmoid activation function (also known as the logistic function) takes in the input and the function converts it into an output of a value in the range 0.0 to 1.0 (16). Binary cross-entropy is used to calculate loss since this is a binary classification problem, where y is the label and p(y) is the predicted probability of it being that class:

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

It computes a score of the average difference between

the true and predicted probability distributions for predicting class 1 (17). Optimizers are algorithms which are implemented to vary the attributes of your neural network so as to obtain the minimum loss. They guide how a machine learning model learns and allows fast computation. They also prevent the model from believing that a local minimum has optimum weights because even though the loss function is a minimum at that local point, the optimum is only reached at the global minimum. This algorithm allows the model to find the globally optimum weights, serving its function as an optimizer. The RMSprop optimizer specifically does this by maintaining a moving average of square gradients and dividing the gradient by the root of this average, working on plain momentum, not Nesterov momentum. The effectiveness of this optimizer is why it is used for this task, and a learning rate hyperparameter of 1e-5 is also chosen. The learning rate was chosen such that the model does not get "stuck" at any local minima, and it also ensures that the model does not make any sudden, large updates to the weights such that it is unfavourable and harmful for the model (6).

Experiments were conducted on Google Colaboratory, with a 12GB NVIDIA Tesla K80 GPU. The model training is set with a batch size of 30 examples, 100 steps per epoch for 100 epochs and the Keras RMSProp optimizer. The validation steps were 50 steps. After 100 epochs, the training accuracy and validation accuracy are measured (**Table 1**).

Convolution2D	Conv Block 1
Convolution2D	
MaxPooling2D	
Convolution2D	Conv Block 2 - Frozen
Convolution2D	
MaxPooling2D	
Convolution2D	Conv Block 3 - Frozen
Convolution2D	
Convolution2D	
MaxPooling2D	
Convolution2D	Conv Block 4 - Trainable
Convolution2D	
Convolution2D	
MaxPooling2D	
Convolution2D	Conv Block 5 - Trainable
Convolution2D	
Convolution2D	
MaxPooling2D	
Flatten	VGG16 Base Model final layer
Dense	Added CNN layers
Dropout	
Dense	
Dropout	
Dense	

Table 3. The model architecture of the VGG16 convolution neural network. The layers within each of the blocks in the base VGG16 model, along with the added convolutional neural network layers are displayed. The base layers were optimized through the experiments.

Received: July 21, 2020
Accepted: October 13 2020
Published: October 18, 2020

REFERENCES

1. Wang, Pei. "On Defining Artificial Intelligence." *Journal of Artificial General Intelligence*, vol. 10, no. 2, 2019, pp. 1–37., doi:10.2478/jagi-2019-0002.
2. Poola, Indrasen. "How Artificial Intelligence in Impacting Real Life Every Day." *International Journal for Advance Research and Development*., vol. 2, Oct. 2017, pp. 96–100.
3. "Weight (Artificial Neural Network)." DeepAI, 17 May 2019, deepai.org/machine-learning-glossary-and-terms/weight-artificial-neural-network.
4. Amidi, Afshine, and Shervine Amidi. "Supervised Learning Cheatsheet Star." *CS 229 - Supervised Learning Cheatsheet, Stanford University*, stanford.edu/~shervine/teaching/cs-229/cheatsheet-supervised-learning.
5. Weiss, Karl, et al. "A Survey of Transfer Learning." *Journal of Big Data*, vol. 3, no. 1, 2016. Crossref, doi:10.1186/s40537-016-0043-6.
6. Shu, Mengying. "Deep Learning for Image Classification on Very Small Datasets Using Transfer Learning." *Iowa State University Digital Repository*, Iowa State University, 2019, lib.dr.iastate.edu/creativecomponents/345/.
7. Abu-Mostafa, Yaser. "DETERMINISTIC NOISE." Deterministic Noise - *Overfitting (Abu-Mostafa),* California Institute of Technology, 2012, work.caltech.edu/library/112.html.
8. Simonyan, Karen, and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition." *ArXiv.org*, University of Oxford, 10 Apr. 2015, arxiv.org/abs/1409.1556.
9. Fei-Fei, L., et al. "ImageNet: Constructing a Large-Scale Image Database." *Journal of Vision*, vol. 9, no. 8, 2010, pp. 1037–1037., doi:10.1167/9.8.1037.
10. Gupta, Ananya. "Ananyagup/Classification-of-Males-and-Females-in-Textbook-Illustrations-." *GitHub*, 18 July 2020, github.com/ananyagup/Classification-of-Males-and-Females-in-Textbook-illustrations-.
11. Kandel, Ibrahem, and Mauro Castelli. "How Deeply to Fine-Tune a Convolutional Neural Network: A Case Study Using a Histopathology Dataset." *Applied Sciences*, vol. 10, no. 10, 2020, p. 3359., doi:10.3390/app10103359.
12. 12. Guo, Yunhui, et al. "SpotTune: Transfer Learning Through Adaptive Fine-Tuning." *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, doi:10.1109/cvpr.2019.00494.
13. Meng, Qi, et al. "Convergence Analysis of Distributed

Stochastic Gradient Descent with Shuffling." *Neurocomputing*, vol. 337, 2019, pp. 46–57., doi:10.1016/j.neucom.2019.01.037.
14. Wang, Jason, and Luis Perez. "The Effectiveness of Data Augmentation in Image Classification Using Deep Learning." *cs231n*, Stanford University, 2017, cs231n.stanford.edu/reports/2017/pdfs/300.pdf.
15. Agarap, Abien Fred. "Deep Learning Using Rectified Linear Units (ReLU)." *ArXiv.org*, Cornell University Arxiv, 7 Feb. 2019, arxiv.org/abs/1803.08375.
16. Jure Leskovec et al. "Neural Nets and Deep Learning." *Mining of Massive Datasets*, 3rd ed., Cambridge University Press, Cambridge, 2020, pp. 498–543.
17. Bishop, Christopher *M. Pattern Recognition and Machine Learning*. Springer New York, 2016.