Advances in Science and Technology Research Journal

Advances in Science and Technology Research Journal 2021, 15(4), 101–109
https://doi.org/10.12913/22998624/142738
ISSN 2299-8624, License CC-BY 4.0

Received: 2021.09.16
Accepted: 2021.10.18
Published: 2021.11.01

Leveraging Transfer Learning to Identify Food Categories

Jyothi Vishnu Vardhan Kolla[1], Poorna Chandra Vemula[2*], Srinivasa L. Chakravarthy[1], Boya Shashidhar Naidu[1], Dharmesh Patibandla[1]

[1] Department of Computer Science, Gitam University, Vishakapatnam, India
[2] Department of Computer Science, VIT University, Vellore, India
* Corresponding author's e-mail: poorna883@gmail.com

ABSTRACT

In today's scenario, recognition of pictured food dishes automatically has significant importance. During the COVID-19 pandemic, there was a decline in people visiting restaurants for their dietary requirements. So many restaurants started offering their services online. This situation caused a demand for better categorization of food into various categories on a large scale by companies that facilitated these services. It is challenging to congregate a large dataset of food categories, so it is complex to build a generalized architecture. To solve this issue, In this paper, domain-specific transfer learning is used to build the model using some standard architectures like VGGNET, RESNET, and EFFICIENTNET family, which are trained on popular benchmark datasets such as IMAGENET, COCO, etc. The similarity between the source and target datasets is calculated to find the best source dataset, and the one with the highest similarity is chosen for transfer learning. The solution proposed in this paper outperforms some of the existing works on categorizing food items.

Keywords: domain similarity, transfer learning, fine-tuning, convolutional neural networks.

INTRODUCTION

Food Recognition plays a significant role in one's life, as it helps to know what to expect from different food items. It also reflects in the digital world; there are many services online which depend on food categorization. For Instance, Food recognition can help patients to calculate their calorie intake daily. Also, this enables food distribution companies to categorize food items and manage their delivery accordingly. However, most current solutions rely on nutrition experts [5] or Amazon Mechanical Turk [6] to label various dishes.

It is a challenging problem to categorize food items, as food does not have many discriminative features. The distinguishing characteristics in the food items are apparently hidden. For Example, Humans can be distinguished by the appearance of their face and body as these characteristics can be clearly compared. Consider a case of a mixed salad; such patterns associating ingredients cannot be found. Further, the nature of food often differs based on various textures and colors of its different local ingredients. Thus, Food recognition is a particular classification task requiring models that are able to exploit local components. Many previous works used different types of feature mining to build a good categorization model. Some of the works relied on weak supervised models such as Random Forests [7, 8] and others adopted discriminative mining of mid-level components [9-16]. This work relies on building a powerful model by using transfer learning.

In recent days, convolutional neural networks like VGGNET [2], RESNET [3], and EFFICIENTNETS [4] have shown a significant impact in the area of visual recognition. However, these state-of-art architectures require enormous amounts of data for training. Training the model with fewer data may lead to overfitting, and the model will not be generalized. However,

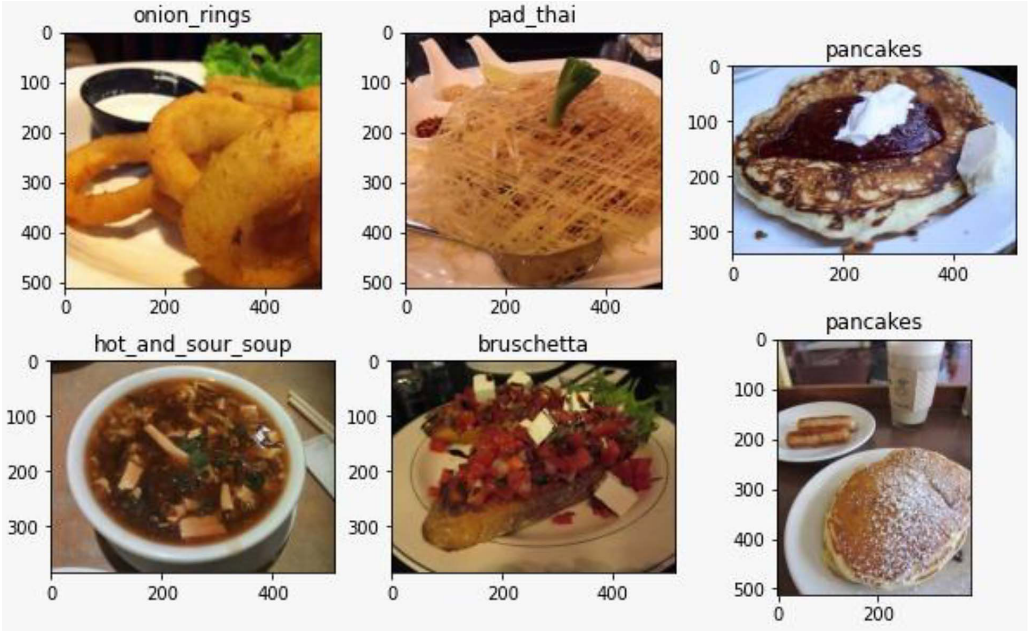

Fig. 1. Shows some random images from the dataset

obtaining vast amounts of data is a complex task. To tackle this issue, In this work, Transfer learning is adopted. Transfer learning is nothing but using pre-trained weights from some state-of-the-art architectures, which are trained on benchmark datasets like IMAGENET [17], COCO [18]. Transfer learning can be used either as a feature extractor [19-21] or by unfreezing some layers by fine tuning [22, 23] the model, which involves altering the learning rate. Due to these advantages, extensive works were done on understanding transfer learning [28, 29]. In this paper, as the problem of food recognition is being dealt with, architectures that were pre-trained on IMAGENET [17] are used via fine-tuning as the similarity between IMAGENET [17] and the dataset [26] being used is high when calculated using a metric called earth mover's distance [24, 25]. The later sections of this work are organized as follows. The details of the dataset [26] used in this paper are described in section 2. Methodology, Experiments & Environment setup, results are described in sections 3, 4, and 5, respectively. In the end, Section 6 concludes this work.

DATASETS

In this paper, we are using a dataset collected by Lukas Bossard et al. [26], where they have chosen images for a total of 101 popular dishes. This dataset consists of 750 training images and 250 test images for each class, amounting to a sum of 101'000 real-world images. Training images are not cleaned and contain some level of noise, mainly in the form of bright colors and incorrect labels. This is done on purpose to build a robust Deep learning algorithm that can be able to work on such weakly labeled images and scale up as the number of classes to be recognized increases. Images were scaled to have a side length of utmost 512 pixels. The dataset includes very diverse but also semantically and seemingly similar food classes. These images are collected by downloading images from foodspotting.com, which enables users to capture images of the food they are eating and also add information such as type of food and place online. So, this dataset provides us with an opportunity to work with real-world food images. Some of the Random images from the dataset are shown in Figure 1.

METHODOLOGY

Data Augmentation and Visualization

Initially, as we will be dealing with the architectures which favor the images in resolution (224, 224, 3), we have resized the images into those dimensions. As the Architectures VGGNET [2] and RESNET [3] do not contain any Normalization layers in them, we have normalized the images before feeding them to these networks.

In contrast, EFFICIENTNET Family has a Normalization layer included in it, so we have fed the raw resized images to EFFICIENTNET. Data Normalization is a technique of Rescaling the pixels in the image to be in the range of 0 to 1; this is because neural networks prefer all the Data Points being fed to them on the same scale. Normalization is done as shown in Eq. (1), where x_{scaled} denotes the normalized image and x denotes the image which we are normalizing, x_{min} denotes the smallest pixel in the image, and x_{max} denotes the largest pixel in the image.

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{1}$$

As images of Food items do not contain rich spatial information like other types of image Data like faces, animals to make our model more accurate in the time of testing, we have generated various types of new images in the training Data set with the help of Data augmentation. Some of the augmentation techniques we used include Flipping images horizontally, rotating the images by some random angle, increasing the width and height of the images randomly, and finally zooming some of the images Randomly.

Model Architectures

VGGNET

VGGNET was first introduced in the year 2014. It has secured first place in the task of localization and second place in classification in the IMAGENET challenge, which was conducted that year. This architecture was built by a team called the visual geometry group from the oxford university. One of the very interesting aspects of this network is that all the parameters are constant, except the depth dimension in the filters, which is gradually increased with the depth of the network. Many variants of VGGNET networks are proposed by the visual geometry group, whose basic idea remains the same and differs in terms of the number of layers. In our work, we have experimented with VGG19 to train our model and the results obtained are promising, which are shown detailly in section 5. The hierarchy of the model is as follows: To simplify the explanation, we explain the model in terms of blocks, where each block is separated by a Maxpool layer. Firstly, VGG16 starts with an input layer that feeds the model with images in batches. Block1 consists of two convolutional layers, each containing a filter size of 64 and a kernel_size of 3X3. As explained above, the features extracted by these two convolutional layers are subjected to the Max Pooling layer, where the most important features are captured. Block2 consists of two convolutional layers, each containing 128 filters, followed by Block3, which consists of three convolutional layers containing 256 filters each. Block4 and Block5 both consist of three convolutional layers with a filter size of 512. As mentioned earlier, all these Blocks are separated by a max-pooling layer, and all the other parameters remain constant throughout the network. The architectures of VGG16 and VGG19 are shown in Figure 3 and Figure 4.

RESNET

This architecture was built by a group of researchers from Microsoft. Its primary idea was to ease the complexity of training deep convolutional neural networks; this was proved by the fact even though it was eight times as large as VGGNET and yet had a lower complexity. Another important achievement is that this network solved the problem of degrading training and testing error rates with an increase in the number of stacked layers by mapping features to a Residual

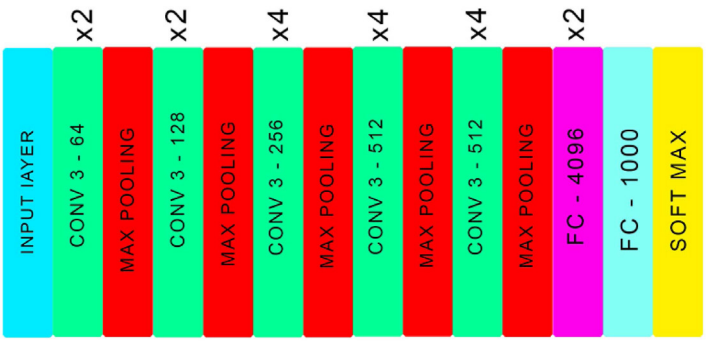

Fig. 2. Representation of VGG19 Architecture

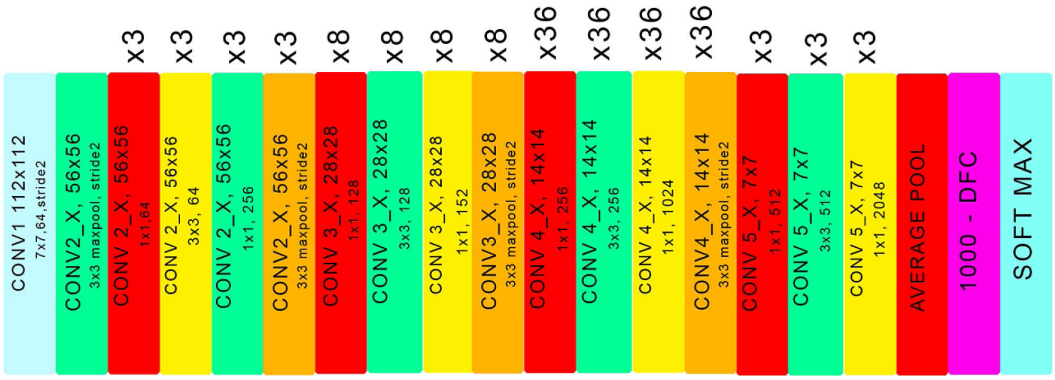

Fig. 3. Representation of RESNET-152 Architecture

function and using shortcut connections between the layers. These observations and proofs are explained in a detailed and clear way in their paper [3]. We have used RESNET with 152 layers which were trained on the IMAGENET with fine-tuning, and the results obtained were very good. To understand the architecture, let us use the terminology of blocks which we used while explaining VGGNET; the architecture starts with a convolutional layer with a kernel size of 7X7 with stride 2 consisting of 64 filters and then followed by a Maxpool layer with a pool size of 3X3 and a stride of 2. Block 1 starts with convolutional layers where the first consists of 64 filters with a kernel_size of 1X1 and the second consists of 64 filters with a kernel size of 3X3. Finally, the third one will have 256 filters with a kernel size of 1X1; this hierarchy repeats a total of 3 times, so Block 1 contains a total of 9 layers. Block2 begins with a convolutional layer with a filter size of 128 and a kernel size of 1X1, followed by another with a filter size of 128 and a kernel size of 3X3, then by another with a kernel size of 1X1 and 512 filters; this pattern repeats eight times which in total produces 24 layers in block2. Block 3 starts with a kernel size of 1X1 and 256 filters, which is then followed by a layer of kernel size 3X3 consisting of a total of 256 filters and the third, which is similar to that of the first one except that the number of filters is 1024, this order repeats for 36 times thus block3 itself contains 108 layers of the architectures'152 layers. At last, Block4 follows the pattern of a 1x1 kernel size with 512 filters, 3X3 kernel size With 512 filters, and a 1X1 kernel size with 2048 filters which is repeated a total of 3 times. Finally, the feature maps obtained are subject to a Global Average pooling layer and connected to a dense layer with softmax as the activation function.

Efficient-Net-Family

This was developed by the Google Ai research team. In this work [4], they have shown how effective a model can be built if the network's depth, width, and resolution are balanced using a simple compound coefficient. This compound is very simple but very effective; suppose if we want to utilize 2^N times more computational power, then we can simply increase the network's width by W^N and height by H^N, and the input image size by S^N. The idea is very logical because if the input size of the image is increased, we have to increase filter size to extract more feature maps and increase the depth to obtain more receptive field; in simple words, the important aspects like depth, width, resolution are uniformized in a constant ratio, and these set of models from Efficient-Family are far more efficient and faster than most of the architectures proposed till date. The two main observations found in this work [4] is that scaling up any one of the dimensions that are width, depth, the resolution will increase the model's accuracy. Still, this rate of increase diminishes after a while. Another observation is that in order to obtain good accuracy and efficiency, it is essential to balance all the dimensions width, depth, and resolution of the model architecture. The baseline architecture of EFFICIENTNET follows the pattern as follows, let us understand the architecture in terms of blocks as we did with previous architectures. Firstly Block1 contains a Convolutional layer with a kernel size of 3X3 consisting of 32 channels; Block2 contains a mobile inverted bottleneck convolutional layer with a kernel size of 3X3 and 16 filters. Block3 consists of two mobile inverted bottleneck convolutional layers with a kernel size of 3X3 and 24 filters. Similarly, Block 4 contains two mobile inverted bottleneck

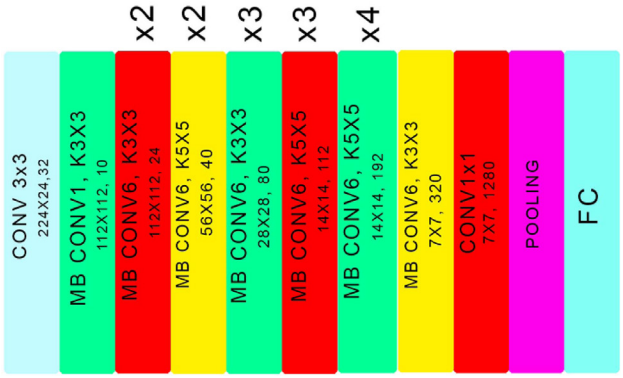

Fig. 4. Representation of EFFICIENTB0 Architecture

convolutional layers with a kernel size of 5X5 and 40 filters. Block 5 consists of three mobile inverted bottleneck convolutional layers, each with a kernel size of 3X3 and 80 filters. Block 6 contains three mobile inverted bottleneck convolutional layers, each with a kernel size of 5X5 and 112 filters. Block 7 consists of four mobile inverted bottleneck convolutional layers, each with a kernel size of 5X5 and 192 filters. Block 8 contains a mobile inverted bottleneck convolutional layer with a kernel size of 3X3 and 320 filters. Block 9 consists of a convolutional layer with a kernel size of 1X1 and a Pooling layer, FC layer with 1280 filters. The components W, H, S are then fixed as constants, and the baseline model is scaled up to build a family of models EFFICIENT B1 to EFFICIENT B7.

Domain similarity

Consider there is a source domain S and a target domain. The distance separating two images, s ∈ S and t ∈ T, is defined as the Euclidean distance between their feature representations:

$$d(s,t) = \| g(s) - g(t) \| \tag{2}$$

Where g(·) denotes the feature extractor for an image, to better calculate the similarity between the images, the feature extractor g(·) needs to extract high-level information from images in an unbiased and generic way. As a result, In this work, we are using g(.) as the extracted features from the last before the layer of a RESNET-101 trained on the JFT dataset. Here, g(s) and g(t) denote the feature extractor for an image on the source and target datasets, respectively.

In many cases, better transfer learning performance can be achieved by using more images. For the purpose of simplicity, in this study, we ignore the effect of domain scale (number of images). In particular, we normalize the number of images in both the source and target domain. As examined by some previous works, transfer learning performance increases logarithmically with training data. This suggests that the performance increase in transfer learning emanating from more training data would be unimportant when we already have a large enough dataset (e.g., ImageNet). Therefore, ignoring the domain scale is a reasonable assumption that simplifies the problem. Our definition of domain similarity can be generalized to take domain scale into account by adding a scale factor. However, we found that ignoring the domain scale already works well in practice.

Under this hypothesis, transfer learning can move images from the source domain S to the target domain T. The image distance Eq. 1 can be defined as the work done by transferring an image to another. Then, the distance between two domains can be defined as the least amount of total work needed. Earth Mover's Distance (EMD) is used to calculate domain similarity as per this definition.

To make the calculations more manageable, further simplifications can be made by representing all image features in a category. This is achieved by computing the mean of their features. Consider, the source domain $\mathcal{S} = \{(s_i, w_{s_i})\}_{i=1}^{m}$ and target $\mathcal{T} = \{(t_j, w_{t_j})\}_{j=1}^{n}$. Earth Mover's Distance(EMD) can be calculated using the equation as shown below where $f_{i,j}$ denotes the data point from source domain whereas $d_{i,j}$ denotes that of a target domain; m is the number of data points in the source domain, and n denotes the number of data points in the target domain.

$$d(\mathcal{S},\mathcal{T}) = \text{EMD}(\mathcal{S},\mathcal{T}) = \\ = \frac{\sum_{i=1,j=1}^{m,n} f_{i,j} d_{i,j}}{\sum_{i=1,j=1}^{m,n} f_{i,j}} \tag{3}$$

Prefetching, multithreading, and mixed precision-scaling

These concepts come from the core of computer science, and these are not at all related to any of the deep learning or machine learning algorithms. However, these play an important role in the training process of the algorithm by speeding up the process up 3 to 4 times the actual time required. If the model is being trained without prefetching, firstly, the CPU prepares a batch of input, and then the GPU computes it, and after that, the CPU again prepares another batch, and the GPU performs computations. In contrast, if prefetching is applied, the CPU will simultaneously prepare the batch of data while the GPU is computing the present batch; in this way, the GPU will be completely utilized except for the time of transferring the data from CPU to GPU. The process can be speeded up a lot more if we can ensure that prefetching and preparing the data is multithreaded. These three different types of processes are illustrated pictorially in Figure 5.

Mixed precision scaling is nothing but using tensors that are of both 16 bit and 32 bit. In order to maintain stability, the outer layers of the model are maintained as 32-bit float data type while the inner layers are maintained as 16-bit float data type. This drastically increases the performance because a 16-bit data type takes significantly less space than a 32 bit but lacks stability. Thus by using mixed precision scaling, we can leverage both the pros and train the model much faster than actual.

EXPERIMENTS AND ENVIRONMENT SETUP

Firstly, the dataset [26] was collected using the TensorFlow data API; as the amount of data that we are dealing with is huge, normally, training the model will not be that feasible. So as mentioned in the methodology section Prefetching, multithreading, and mixed precision-scaling, we have followed some advanced techniques to train the model. When it comes to hardware, an Intel i7 processor with a ram of 16GB and a Tesla T4 GPU with VRAM of 16GB was used to train all the architectures that we have used in this work. As EFFICIENTNET family already consists of a rescaling layer embedded in them, it is not required to manually rescale the image. As we have leveraged the power of transfer learning, firstly, we have experimented with EfficientnetB0 by freezing all the layers except the output layer, which is a fully connected layer with 101 neurons and softmax as an activation function as we are dealing with multiclass classification. The mathematical formulation of softmax is shown in the Eq. 4, where x_i denotes the input vector, and x_j denotes the output vector; finally, the model was compiled with adam [27] optimizer with a learning rate of 0.001 with sparse_categorical_cross_entropy as loss function. Then the model was fine-tuned as suggested by other previous works [28, 29], as the amount of training data was very high, all the

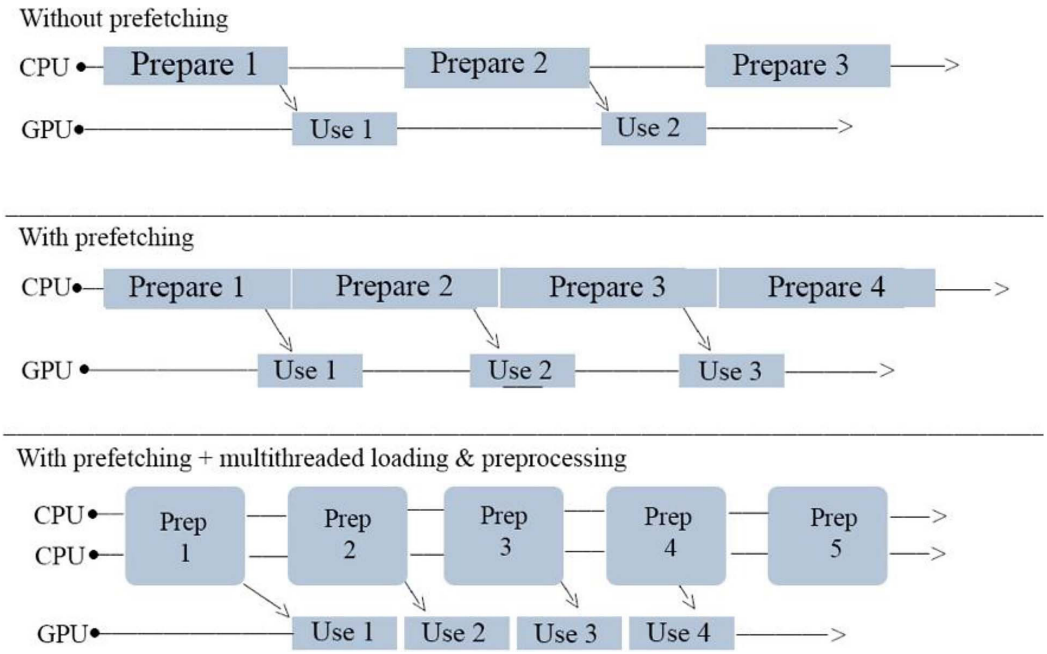

Fig. 5. Representation of prefetching and multithreading

layers in the pretrained EFFICIENTNET were fine-tuned with a reduced initial learning rate of 0.0001; also, the learning rate was reduced by a factor of 5 once the loss function stops reducing for at least 2 epochs and this reduction of learning will not go below 0.0000001.

$$\text{Softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \tag{4}$$

The same training pattern was followed to train and fine-tune the RESNET-125, VGG19 architectures with the dataset [26] except that these models do not have an rescaling layer embedded in them, we have manually rescaled the batch of images before feeding them to the model. The mathematical formulation of Rescaling is shown in the Eq. 1.

RESULTS

Of all the models used, EFFICIENTNET gave the best results with an accuracy of 80%, which is the best of all the architectures used, in terms of both accuracy and efficiency with only 11 million parameters. When it comes to RESNET-152, the accuracy achieved is almost close to 80% (79.9%). But in terms of efficiency and amount of time required to train the model, EFFICIENTNET is much better than RESNET-152 as the number of parameters in RESNET-152 is 60 million which is about 6 times that of in EFFICIENTNET. VGG-19 is the worst performing among all the models, with an accuracy of 74%. Moreover, the time required to train and resource consumption is higher relative to other models as the number of parameters is 144 million, which is double that of in RESNET-152 and around 13 times that of in EFFICIENTNET. The training and validation errors of all these architectures are shown in the below figures.

The accuracy we have obtained by fine-tuning EFFICIENTNET and RESNET is better than most previous works [26, 30]. To the extent of our knowledge, the paper by Liu et al. [30] has a top 1% accuracy of 77.4%, which we have successfully beaten in our work by applying the techniques of transfer learning and fine-tuning on the architectures EFFICIENTNETB0 and RESNET-152, which were trained on IMAGENET dataset.

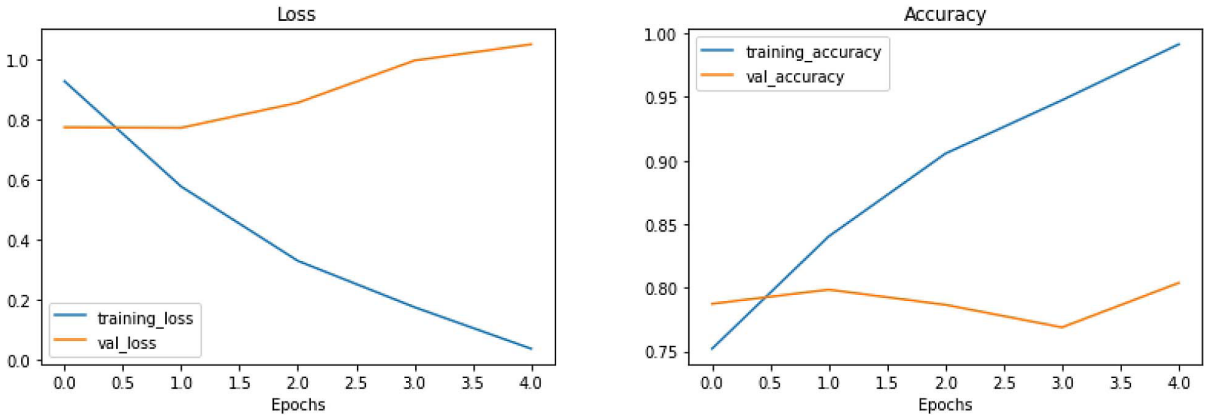

Fig. 6. Accuracy and Loss functions of VGGNET

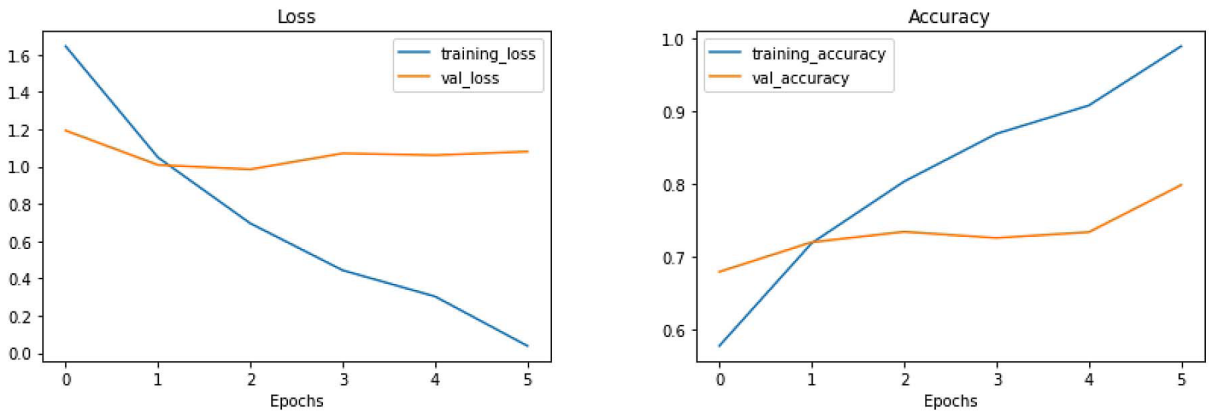

Fig. 7. Accuracy and Loss functions of RESNET

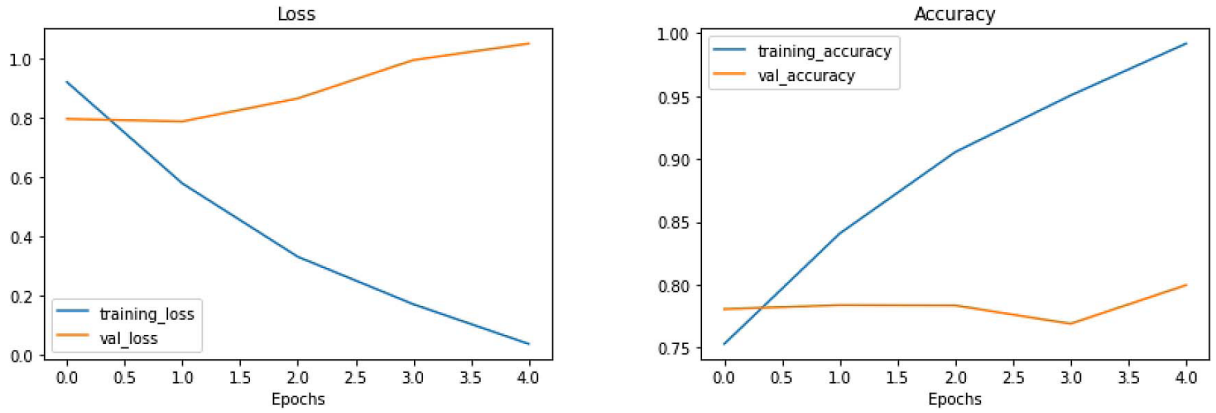

Fig. 8. Accuracy and Loss functions of EFFIECIENTNET

CONCLUSION

In conclusion, By using transfer learning, that is, by utilizing pre-trained weights via fine-tuning of some popular architectures such as RESNET, VGG-NET, and EFFICIENTNET, the results obtained are better than most of the previous studies to the extent of our knowledge. Leveraging the power of transfer learning, we can build robust models by finding the right target dataset using domain similarity based on the source dataset and using standard architectures trained on this source dataset. Further studies can include building a larger dataset by adding more images in each category, also involving numerous categories, and building a robust model which is specialized in food categorization.

Moreover, we have performed fine-tuning manually. However, as an extension, we can make studies on how to dynamically unfreeze the layers based on the image fed to the network rather than manually unfreezing some of the layers while fine-tuning, as layers are unfrozen dynamically for each specific image due to which if the pre-trained model has never seen the image, the top layers will be unfrozen and if the pre-trained model has already seen similar image then bottom layers are unfrozen; as a result, the performance of the model improves by a great extent.

REFERENCES

1. Cui Y., Song Y., Sun C., Howard A., Belongie S. Large Scale Fine-Grained Categorization and Domain-Specific Transfer Learning. 2018;4109-4118. DOI: 10.1109/CVPR.2018.00432.
2. Simonyan K., Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition. 2014;1.
3. He K., Zhang X., Ren S., Sun J. Deep Residual Learning for Image Recognition. 2016;770-778. DOI: 10.1109/CVPR.2016.90.
4. Tan M., Le Q. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks; 2019.
5. Martin C.K., Correa J., Han H., Allen H., Rood J., Champagne C., Gunturk B., Bray G. Validity of the Remote Food Photography Method (RFPM) for Estimating Energy and Nutrient Intake in Near Real-Time. Obesity. 2012;20.
6. Noronha J., Hysen, E., Zhang, H., Gajos K.Z. 2011. Platemate: crowdsourcing nutritional analysis from food photographs. Proceedings of the 24th annual ACM symposium on User interface software and technology.
7. Breiman L. Random Forests. Machine Learning. 2001;45:5–32. https://doi.org/10.1023/A:1010933404324
8. Ho T.K. Random decision forests. In: Proceedings of 3rd International Conference on Document Analysis and Recognition. 1995;1:278-282. DOI: 10.1109/ICDAR.1995.598994.
9. Doersch C., Gupta A., Efros A.A. Mid-Level Visual Element Discovery as Discriminative Mode Seeking. NIPS. 2013.
10. Sun J., Ponce J. Learning Discriminative Part Detectors for Image Classification and Cosegmentation, IEEE International Conference on Computer Vision 2013, 3400-3407. DOI: 10.1109/ICCV.2013.422.
11. Wang X., Wang B., Bai X., Liu W., Tu Z. Max-margin multiple-instance dictionary learning. NIPS. 2013.
12. Li Q., Wu J., Tu Z. Harvesting Mid-level Visual Concepts from Large-Scale Internet Images. IEEE Conference on Computer Vision and Pattern Recognition 2013, 851-858.
13. Endres I., Shih K., Jiaa J., Hoiem D. Learning Collections of Part Models for Object Recognition. CVPR. 2013.

14. Juneja M., Vedaldi A., Jawahar C.V., Zisserman A. Blocks That Shout: Distinctive Parts for Scene Classification. IEEE Conference on Computer Vision and Pattern Recognition 2013, 923-930.
15. Singh S., Gupta A., Efros A.A. Unsupervised Discovery of Mid-Level Discriminative Patches. ECCV, 2012.
16. Yao B., Khosla A., Fei-Fei L. 2011. Combining randomization and discrimination for fine-grained image categorization. CVPR. 2011;1577-1584.
17. Deng J., Dong W., Socher R., Li L., Li K., Fei-Fei L. ImageNet: A large-scale hierarchical image database. IEEE Conference on Computer Vision and Pattern Recognition 2009, 248-255. DOI: 10.1109/CVPR.2009.5206848.
18. Lin T., Maire M., Belongie S.J., Hays J., Perona P., Ramanan D., Dollár P., Zitnick C.L. Microsoft COCO: Common Objects in Context. ECCV. 2014.
19. Razavian A.S., Azizpour H., Sullivan J., Carlsson S. CNN Features Off-the-Shelf: An Astounding Baseline for Recognition. IEEE Conference on Computer Vision and Pattern Recognition Workshops 2014, 512-519. DOI: 10.1109/CVPRW.2014.131.
20. Donahue J., Jia Y., Vinyals O., Hoffman J., Zhang N., Tzeng E., Darrell T. DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition. ICML. 2014.
21. Zhou B., Lapedriza A., Khosla A., Oliva A., Torralba A. Places: A 10 Million Image Database for Scene Recognition. IEEE transactions on pattern analysis and machine intelligence. 2018;40(6):1452–1464. DOI: 10.1109/TPAMI.2017.2723009.
22. Girshick R., Donahue J., Darrell T., Malik J. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. IEEE Conference on Computer Vision and Pattern Recognition 2014, 580-587. DOI: 10.1109/CVPR.2014.81.
23. Oquab M., Bottou L., Laptev I., Sivic J. Learning and Transferring Mid-level Image Representations Using Convolutional Neural Networks. IEEE Conference on Computer Vision and Pattern Recognition 2014, 1717-1724. DOI: 10.1109/CVPR.2014.222.
24. Rachev S.T. The monge–kantorovich mass transference problem and its stochastic applications. Theory of Probability & Its Applications. 1985.
25. Rubner Y., Tomasi C., Guibas L. The Earth Mover's Distance as a Metric for Image Retrieval. International Journal of Computer Vision. 2004;40:99-121.
26. Bossard L., Guillaumin M., Van Gool L. Food-101 – Mining Discriminative Components with Random Forests. In: Fleet D., Pajdla T., Schiele B., Tuytelaars T. (eds) Computer Vision – ECCV 2014. ECCV 2014. Lecture Notes in Computer Science, Springer, Cham; 2014;8694. DOI: 10.1007/978-3-319-10599-4_29
27. Kingma D.P., Ba J. Adam: A Method for Stochastic Optimization. 2015.
28. Bengio Y. Deep learning of representations for unsupervised and transfer learning. In ICML Workshop on Unsupervised and Transfer Learning. 2012;1
29. Guo Y., Shi H., Kumar A., Grauman K., Simunic T., Feris R. SpotTune: Transfer Learning Through Adaptive Fine-Tuning. *IEEE/CVF Conference on Computer Vision and Pattern Recognition* 2019, 4800-4809.
30. Liu C., Cao Y., Luo Y., Chen G., Vokkarane V., Ma Y. DeepFood: Deep Learning-Based Food Image Recognition for Computer-Aided Dietary Assessment. 2016.