ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 5, Issue 11, November 2016

Transfer Learning for Image Classification and Plant Phenotyping

Aniruddha Tapas

***Abstract*— Convolutional Neural Networks make effective use of multiple stacked layers to perform the task of Image Classification, but this implementation demands availability of considerable image data, high computation capability and significant time for training. Plant Phenotyping refers to the comprehensive assessment of complex plant traits. Due to restrictions in accessibility of extensive plant image data, Image-based Plant Phenotyping requires modifications to the implementation of Convolutional Neural Networks. We explore the process of Transfer Learning on the 'Computer Vision Problems in Plant Phenotyping (CVPPP)' database. We retrained the GoogLeNet Convolutional Neural Network model which was trained by Google Inc. on the ImageNet dataset. The ImageNet dataset comprises of approximately 100,000 images of 1000 classes. We demonstrate the fine-tuning of the GoogLeNet model to classify the Arabidopsis and Tobacco plants images from the CVPPP database which were originally not present in the ImageNet dataset. We achieved an overall accuracy of 98% on the retrained model using Transfer Learning by remodeling the final layers of the GoogleNet model. Further, we elaborate on the working of Transfer Learning in this use-case and the methodology we followed to accomplish our results.**

***Index Terms*—Transfer Learning, Convolutional Neural Networks, Image Classification, Inception-v3, Plant Phenotyping.**

I. Introduction

Techniques used for Image Classification are advancing day by day with the improvements in Deep Learning. Convolutional Neural Networks or CNN, particularly, are widely used in image classification techniques. CNN are essentially a deep network that constructs features which would generally be required to be handcrafted if traditional machine learning approaches are to be used. These abstract features they create from training are so generalized that they account for variance. Hence, to train a CNN from scratch would require substantial computing power, time for training and abundant data. However, with the case of Phenotyping, data and training time are constrained and alternative approaches have to be considered. Transfer learning proves to be useful in such use-cases. In this paper, we demonstrate the use of Transfer Learning to classify images of Arabidopsis and Tobacco plants [2]. We used a pre-trained CNN model called Inception-v3 [1]. Inception model was originally trained by Google Inc. on the ImageNet database which comprises of 100,000 images of about 1000 classes. Transfer learning refers to the process of applying learning from a previous training session to a new training session. In the case of Inception model, we feed in an image as an input; and at each layer of the Inception CNN model, it will perform a series of operations on the data until it outputs a label and classification percentage. Each layer has a different set of abstractions. In the first layers the model basically teaches itself edge detection; then shape detection in the middle layers and they get increasingly more abstract up until the end. The last few layers are the highest level detectors for whole objects. For transfer learning, we retrained that last layer on features of Arabidopsis and Tobacco plants so that we can add their representations to the Inception model's repository of knowledge.

The paper firstly explains the concepts of Convolutional Neural Network and how it proceeds to classify images. Then the Inception Model is explained as how it has achieved state of the art results of image classification. The dataset used for the research is elaborated and the process of Plant Phenotyping is then explained. Finally, the methodology to implement Transfer learning to fine-tune the Inception-Model and retrain the last layers of the CNN model to classify the plants image data is described.

II. Previous works and background

Convolutional neural networks are where machine learning meets computer vision to take on the problems of image classification and object detection. The state-of-the-art in Convolutional neural networks is getting pushed further and further as the time is passing by. At the moment one of the leading models is the Inception-v4 model that achieved a 3.08% top error rate on the ImageNet dataset. [3] The Inception-v4 model has 75 trainable layers. To build such complex models, it takes about 2 weeks to train, coupled with substantial computational requirements.

Among numerous applications of computer vision, plant image analysis has as of late increasingly gained more consideration because of its potential effect on plant visual phenotyping, especially in comprehending plant development, evaluating the quality/execution of harvest plants, and enhancing crop yield. The data corresponding to plant phenotyping, however, is not available in larger scales. [4] This hinders the implementation of convolutional neural networks, due to its requirements of large amount of image data for training. Transfer learning thus becomes a viable option to tackle the issue of plant image classification, where the generalized features from a pre-trained model can be used to test classification on related datasets.

Manuscript received Nov, 2016.
***Aniruddha Tapas**, Computer Science, Shri Ramdeobaba College of Engineering and Management., Nagpur, India, +91787587551*

III. Problem Setup

A. *Convolutional Neural Networks*

Image classification analyzes the numerical properties of various image features and organizes data into categories. To extract key features, image data requires subject-matter expertise. Convolutional neural networks extract features automatically from domain-specific images, without any feature engineering techniques. This process makes CNNs suitable for image analysis [9]. DCNNs train networks with multiple layers. These layers work to construct an improved feature space. Initial layers learn 1st order features (e.g. color, edges etc.). Later layers learn higher order features (specific to input dataset). Lastly, final layer features are fed into classification layer(s).

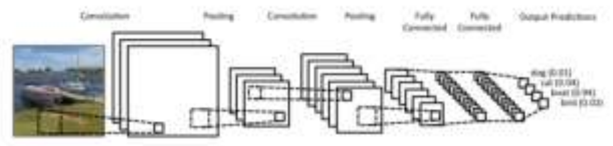

Fig 1: Convolutional Neural Network Architecture

Observable from Figure 1, a *convolutional neural network* consists of model layers that apply local filters and are stacked in a certain order.

Convolution: Convolution layers comprise of a rectangular matrix of neurons. The weights for this are the same for every neuron in the convolution layer. The convolution layer weights indicate the convolution channel.

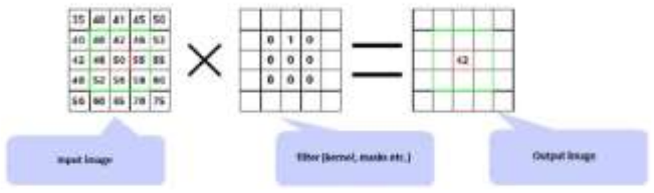

Fig 2: Convolution

Pooling: The pooling layer takes small rectangular blocks from the convolutional layer and subsamples it to produce a single output from that block. It is an operation that aims at reducing dimensionality.

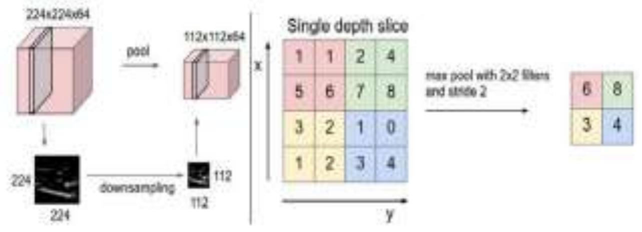

Fig 3: Pooling

Fully-Connected: Finally, after several convolutional and max pooling layers, the high-level reasoning in the neural network is done via fully connected layers. A fully connected layer takes all neurons in the previous layer (be it fully connected, pooling, or convolutional) and connects it to every single neuron it has.

B. *Inception-v3 Model*

In this paper, we are utilizing the GoogLeNet Deep Convolutional Neural Network, which was developed at Google. The GoogleNet (codenamed Inception) accomplished the new state of the art for classification and detection in the ImageNet Large-Scale Visual Recognition Challenge 2014 (ILSVRC14). Inspirations for this model were a simultaneously deeper as well as computationally inexpensive architecture. It is a 22 layers deep network. The exact structure of the extra network on the side, including the auxiliary classifier, is as follows [1]:

- An average pooling layer with 5_5 filter size and stride 3, resulting in an 4_4_512 output for the (4a), and 4_4_528 for the (4d) stage.
- A 1_1 convolution with 128 filters for dimension reduction and rectified linear activation.
- A fully connected layer with 1024 units and rectified linear activation.
- A dropout layer with 70% ratio of dropped outputs.
- A linear layer with softmax loss as the classifier (predicting the same 1000 classes as the main classifier, but removed at inference time).

The 22 layers of the Inception Model perform the task of image classification by learning abstract features, whose level increments as we move towards the final output layers. Thus the first few layers teach themselves low level abstract features like blob detection or edge detection. Pushing forward, the consequent layers prepare themselves to perform shape detection, and afterward color detection and so on. Accordingly, the features of particular images are captured and learned through the layers. The final two layers of the model teach themselves more specific features corresponding to a particular category. Thus the connections of layers that detect these high level abstract features are used to classify the images it was trained with.

C. *Combining Transfer Learning and Convolutional Neural Network*

The low-level and high-level features learned by a CNN on a source domain can often be transferred to augment learning in an alternate but related target domain. For target problems with abundant data, we can transfer low-level features, such as edges and corners, and learn new high-level features specific to the target problem. For target problems with limited amounts of data, learning new high-level features is difficult.

Nonetheless, if the source and target domain are adequately similar, the feature representation learned by the CNN on the source task can likewise be utilized for the target problem. Deep features extracted from CNNs trained on large annotated datasets of images have been used as generic features viably for an extensive variety of computer vision tasks.

IV. Plant Phenotyping and Datasets Used

We used three image datasets: two datasets showing Arabidopsis plants and one dataset showing Tobacco Plants, introduced in the public domain of Computer Vision Problems in Plant Phenotyping (CVPPP 2014) [2].

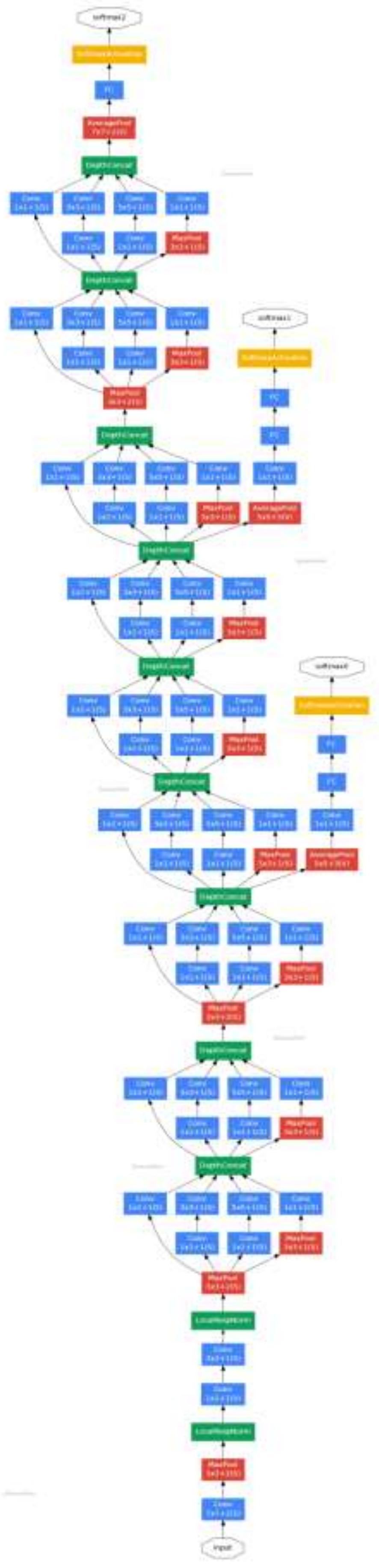

Fig 4: GoogLeNet network

A key factor for advancement in agriculture and plant research is the study of the phenotype expressed by cultivars (or mutants) of the same plant species under different environmental conditions. Identifying and assessing a plant's actual properties, i.e., its phenotype is relevant to, e.g., seed production and plant breeders. Image-based methodologies are picking up consideration among plant analysts to gauge and study visual phenotypes of plants. In the last decades, a variety of approaches based on images have been developed to measure visual traits of plants in an automated manner. Several laboratories developed customized image processing pipelines to analyze the image data acquired during experiments.

We performed multi-label image segmentation of leaves of rosette plants and tobacco plants, where only single images are given, as opposed to image sequences.

The three datasets, named respectively `A1', `A2', and `A3', consists of single-subject images of Arabidopsis and Tobacco plants, each accompanied by manual annotation of plant and leaf pixels, examples of which are shown in Figure 5,6 and 7. The `A1' dataset comprises of 161 images (width x height: 500 x 530 pixels). Additional 40 images (width x height: 530 x565 pixels), are comprised in `A2'. Whereas the `A3' dataset consists of 83 tobacco images (width x height: 2448 x 2048 pixels).

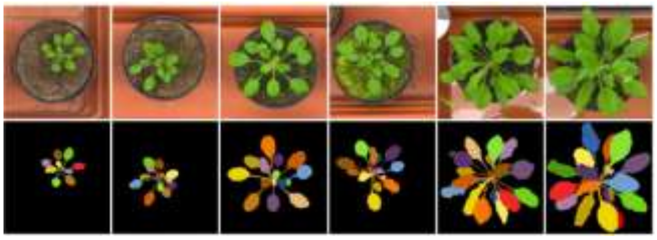

Fig 5: A1 plant dataset Arabidopsis (2012)

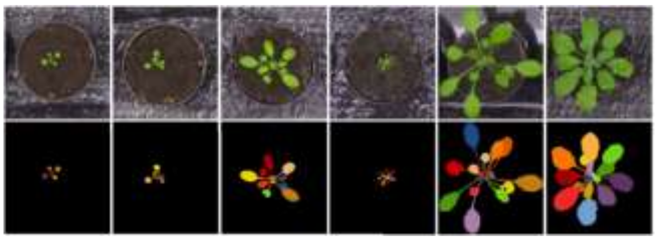

Fig 6: A2 plant dataset Arabidopsis (2013 canon)

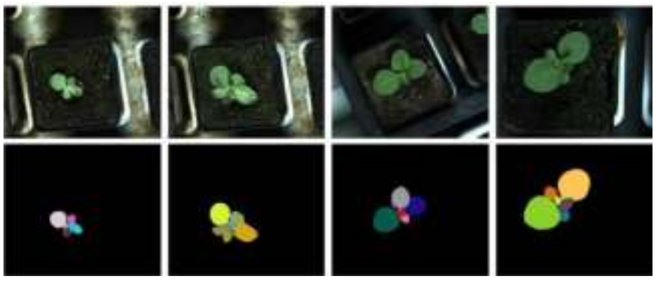

Fig 7: A3 plant dataset (tobacco)

The Inception model that accomplished the cutting edge in object detection and image classification was trained on 100,000 images; while the data what we had for plant phenotyping was only in the triple figures. Hence fine tuning the CNN model and retraining it in the plant dataset was required.

V. Transfer Learning and Fine-Tuning Convolutional Neural Networks

In practice, we don’t usually train an entire Deep Convolutional Neural Network from scratch with random initialization. This is on the grounds that it is relatively rare to have a dataset of abundant size that is required for the depth of network required. Rather, it is common to pre-train a DCNN on a very large dataset and then use the trained DCNN weights either as an initialization or a fixed feature extractor for the task of interest [10].

Fine-Tuning: Transfer learning methodologies rely upon different elements; however the two most essential ones are the size of the new dataset and its similarity to the original dataset. Knowing that DCNN features are more generic in early layers and more dataset-specific in later layers, there are four major scenarios:

1. New dataset is smaller in size and comparatively similar in content to the original dataset: If the data is small, overfitting concerns reduce the effectiveness of fine-tuning the DCNN. Since the data is similar to the original data, we expect higher-level features in the DCNN to be relevant to this dataset as well. Hence, the best idea would be to train a linear classifier on the CNN-features.
2. New dataset is relatively large in size and similar in content compared to the original dataset: Since we have more data, we can have more confidence that we would not over fit, if we were to try to fine-tune through the full network.
3. New dataset is smaller in size but very different in content compared to the original dataset: Since the data is small, it is likely best to only train a linear classifier. Since the dataset is altogether different, training the classifier from the top of the network, which contains more dataset-specific features, would not necessarily efficient. Rather, training a classifier from activations somewhere prior in the network would work better.
4. New dataset is relatively large in size and contrasted in content compared to the original dataset: Since the dataset is very large, we may expect that we can afford to train a DCNN from scratch. However, in practice it is very often still beneficial to initialize with weights from a pre-trained model. In this case, we would have enough data and confidence to fine-tune through the entire network.

Fine-tuning DCNNs: For this Plant Phenotyping problem, we fall under scenario 3. We fine-tuned higher-level layers of the network. This is propelled by the perception that the prior features of a DCNN contain more generic features (e.g. edge detectors or color blob detectors) that should be useful to many tasks, but later layers of the DCNN becomes progressively more specific to the subtle elements of the classes contained in the CVPPP dataset.

Transfer learning constraints: As we decided to use a pre-trained network, slight constraints in terms of the model architecture needed to be handled. For example, we couldn’t arbitrarily take out convolutional layers from the pre-trained network. However, due to parameter sharing, we could without much of a stretch, run a pre-trained network on images of different spatial size. This is obviously apparent in the case of Convolutional and Pool layers because their forward function is independent of the input volume spatial size. In case of Fully Connected (FC) layers, this still remains constant in light of the fact that FC layers can be changed over to a Convolutional Layer.

Learning rates: We used a smaller learning rate for DCNN weights that were being fine-tuned under the assumption that the pre-trained DCNN weights are effective. In order to keep the learning rate and learning rate decay small, we didn’t distort the weights too quickly or too much.

Data Augmentation: One of the drawbacks of non-regularized neural networks is that they are extremely flexible: they learn both features and noise equally well, increasing the potential for overfitting. In our model, we applied L2 regularization to avoid overfitting. But even after that, we observed a large gap in model performance on the training and validation plant images, indicating that the fine tuning process is overfitting to the training set. To combat this overfitting, we leveraged data augmentation for the plant image dataset.

There are many ways to do data augmentation, such as the popular horizontally flipping, random crops and color jittering. As the color information in these images is very important, we only rotated the images at different angles – at 0, 90, 180, and 270 degrees.

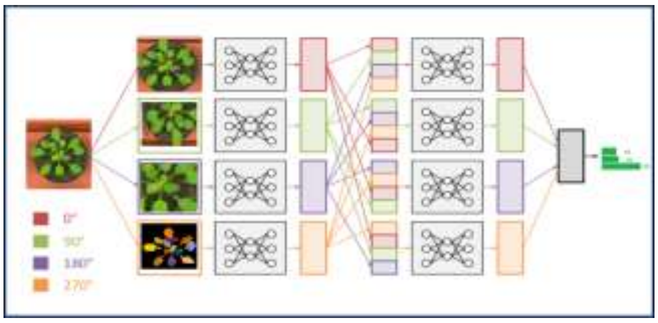

Fig 8: Transfer learning to classify plants

Fine-tuning Inception model: We fine-tuned all layers, except for the top 2 pre-trained layers which contain more generic data-independent weights. The original classification layer "loss3/classifier" outputs predictions for 1000 classes. We replaced it with a new binary layer to classify the three types of plants in the dataset.

VI. Approach

A. Tensorflow Setup and Scripting

To use transfer learning for classifying plant images, we used Tensorflow library [6] to load the Inception-v3 model on our local machine, retrain it on the plant phenotyping dataset and then classify new images to be one of the three categories - `A1', `A2', and `A3'. Tensorflow is an open source library for numerical computation, specializing in machine learning applications, released by Google Inc. We used Docker Toolbox to install and run the Tensorflow image to program our solution.

```
% docker run -it -v $HOME/Plant_Data:/Plant_Data gcr.io/tensorflow/tensorflow:latest-devel

#Code to install the Tensorflow image.
```

We stored our dataset in the directory' Plant_Data'. This directory contained the sub-directories: 'A1','A2 and,'A3' each comprising of the plant images corresponding to the categories `A1', `A2', and `A3'. We retrained the Inception model on this data to classify the images with the ground truth labels as the directory names. Inception is a huge image classification model with millions of parameters that can differentiate a large number of kinds of images. We only trained the final layer of that network, so training ended in a reasonable amount of time about 30 minutes.

```
$ python tensorflow/examples/image_retraining/retrain.py \
--bottleneck_dir=/Plant_Data /bottlenecks \
--how_many_training_steps 4000 \
--model_dir=/Plant_Data /inception \
--output_graph=/Plant_Data /retrained_graph.pb \
--output_labels=/Plant_Data /retrained_labels.txt \
--image_dir /Plant_Data

#Code to load the pre-trained Inception v3 model, remove the
#old final layer, and train a new one on the plant images
```

ImageNet was not trained on any of these plant species, originally. However, the kinds of information that make it possible for ImageNet to differentiate among 1,000 classes are also useful for distinguishing other objects. By using this pre-trained network, we used that information as input to the final classification layer that distinguished our plant classes.

B. Bottlenecks

The Inception v3 model comprises of many layers stacked on top of each other (see Fig 4). These layers are pre-trained and are already very valuable at finding and summarizing information that will help classify most images. We intended on training only the last layer; the previous layers retain their already-trained state. The first phase analyzes all the images on disk and calculates the bottleneck values for each of them. The layer just before the final output layer that actually does the classification is informally termed as 'Bottleneck'. This penultimate layer has been trained to output a set of values that's good enough for the classifier to use to distinguish between all the classes it's been asked to recognize. That means it has to be a meaningful and compact summary of the images, since it has to contain enough information for the classifier to make a good choice in a very small set of values. The reason our final layer retraining can work on new classes is that it turns out the kind of information needed to distinguish between all the 1,000 classes in ImageNet is often also useful to distinguish between new kinds of objects.

Because every image is reused multiple times during training and calculating each bottleneck takes a significant amount of time, it speeds things up to cache these bottleneck values on disk so they don't have to be repeatedly recalculated. By default they're stored in the /tmp/bottleneck directory. If we rerun the script, they'll be reused, so we didn't have to wait for this part again.

C. Classifying with the Retrained Model

The retraining script wrote out a version of the Inception v3 network with a final layer retrained to the plant categories `A1', `A2', and `A3' to /tmp/output_graph.pb and a text file containing the labels to /tmp/output_labels.txt. We used a Python script to load the retrained graph and the text labels using the Tensorflow library. We then inputted the new image to the graph to get first prediction.

$$x_i' = \frac{1}{\left\{1+e^{\left\{-\frac{\{x_i-\mu_i\}}{\sigma_i}\right\}}\right\}}$$

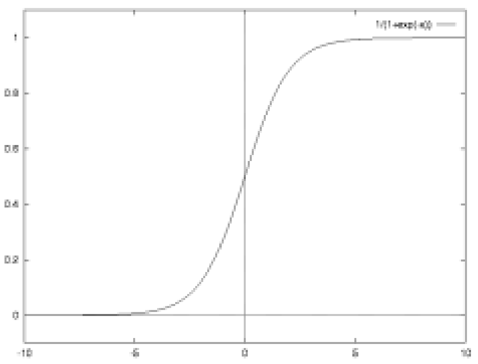

Fig 9: Softmax function

Finally, we used the softmax function in the final layer to map input data into probabilities of an expected output.

VII. Results

Once the bottlenecks were complete, the actual training of the top layer of the network began. We evaluated the training process using training accuracy, validation accuracy, and the cross entropy. The training accuracy shows what percent of the images used in the current training batch were labeled with the correct class. The validation accuracy is the precision on a randomly-selected group of images from a different set. The key difference is that the training accuracy is based on images that the network has been able to learn from so the network can overfit to the noise in the training data. A true measure of the performance of the network is to measure its performance on a data set not contained in the training data -- this is measured by the validation accuracy. If the train accuracy is high but the validation accuracy remains low, that means the network is overfitting and memorizing particular features in the training images that aren't helpful more generally. Cross entropy is a loss function which gives a glimpse into how well the learning process is progressing. The training's objective was to make the loss as small as possible, so we could tell if the learning was working by watching out for whether the loss continued inclining downwards, disregarding the short-term noise.

By default this script ran 4,000 training steps. Every progression picked ten images at random from the training set, found their bottlenecks from the cache, and feed them into the final layer to get predictions. Those predictions were then compared against the actual labels to update the final layer's weights through the back-propagation process. As the process continued, the reported accuracy improved, and after all the steps, a final test accuracy evaluation was run on a set of images kept separate from the training and validation pictures. This test evaluation is the best estimate of how the trained model will perform on the classification task. We achieved an accuracy value of 98%. This number is based on the percent of the images in the test set that are given the correct label after the model is fully trained.

VIII. Conclusion

We assessed a transfer learning approach that leverages recent deep learning advances and complex models built using convolutional neural networks to learn high-level feature representations of a myriad of objects. This knowledge is then transferred to specific data insufficient tasks in the spirit of transfer learning. We demonstrate an application of this idea in the context of plant phenotyping and introduce the concept of retraining the Inception model released by Google Inc. to classify plant image data. Using these features, we are able to approach the task of image classification by utilizing the data introduced in the field of plant phenotyping by CVPPP. Remarkably, our approach achieved 98 percent accuracy in correctly classifying the test data. To improve upon the efficiency of the transfer learning process, approaches that involve training more or fewer layers and observing how that affects the rate of convergence and overall accuracy would prove conducive. In transfer learning, we are mostly concerned with training the fully connected layers because the convolutional layers act as feature extractors; however there have been advantages in also training the final convolutional layer, so this would be an avenue worth exploring.

In future work, we hope to evaluate other types of models, including ones that do not involve complex architectures and deep learning and then compare the results; taking the predictions of several models and averaging them to come up with a more robust prediction which would prove to be a productive avenue. In addition, we hope to introduce approaches that can take heterogeneous features and temporal dimensions into account with the techniques mentioned.

Acknowledgment

Aniruddha Tapas would like to thank Google Inc. for releasing the Inception model and Tensorflow library to help researchers like us to progress in this fascinating field of Machine Learning and Artificial Intelligence. He would also like to thank all the other staff of Computer Science and Engineering Department of Shri Ramdeobaba College of Engineering and Management, Nagpur, for their enormous support and help. Finally, he would like to thank the Computer Vision Problems in Plant Phenotyping (CVPPP) workshop for releasing the Plant Phenotyping Database.

REFERENCES

[1] Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going Deeper with Convolutions." *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015): n. pag. Web.

[2] *Hanno Scharr, Massimo Minervini, Andreas Fischbach, Sotirios A. Tsaftaris. Annotated Image Datasets of Rosette Plants. Technical Report No. FZJ-2014-03837, Forschungszentrum Jülich, 2014.*

[3] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke: "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning", 2016; [http://arxiv.org/abs/1602.07261 arXiv:1602.07261].

[4] *Massimo Minervini, Mohammed M. Abdelsamea, Sotirios A. Tsaftaris, Image-based plant phenotyping with incremental learning and active contours, Ecological Informatics, Available online 6 August 2013,ISSN1574-9541, http://dx.doi.org/10.1016/j.ecoinf.2013.07.004.*

[5] Cruz, Jeffrey A., Xi Yin, Xiaoming Liu, Saif M. Imran, Daniel D. Morris, David M. Kramer, and Jin Chen. "Multi-modality Imagery Database for Plant Phenotyping." *Machine Vision and Applications* 27.5 (2015): 735-49. Web.

[6] Abadi, Martín. "TensorFlow: Learning Functions at Scale." Proceedings of the 21st ACM SIGPLAN International Conference on Functional Programming - ICFP 2016 (2016): n. pag. Web.

[7] Sun, Yi, Xiaogang Wang, and Xiaoou Tang. "Deep Learning Face Representation from Predicting 10,000 Classes." *2014 IEEE Conference on Computer Vision and Pattern Recognition* (2014): n. pag. Web.

[8] Yoshua Bengio, Aaron Courville: "Representation Learning: A Review and New Perspectives", 2012; [http://arxiv.org/abs/1206.5538 arXiv:1206.5538].

[9] Nguyen, K., Fookes, C., & Sridharan, S. (2015). Improving deep convolutional neural networks with unsupervised feature learning. *2015 IEEE International Conference on Image Processing (ICIP)*. doi:10.1109/icip.2015.7351206

[10] Oquab, M., Bottou, L., Laptev, I., & Sivic, J. (2014). Learning and Transferring Mid-level Image Representations Using Convolutional Neural Networks. *2014 IEEE Conference on Computer Vision and Pattern Recognition*. doi:10.1109/cvpr.2014.222

[11] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens: "Rethinking the Inception Architecture for Computer Vision", 2015; [http://arxiv.org/abs/1512.00567 arXiv:1512.00567].

[12] Pan, Sinno Jialin, and Qiang Yang. "A Survey on Transfer Learning." *IEEE Transactions on Knowledge and Data Engineering* 22.10 (2010): 1345-359. Web.

[13] Ding, Zhengming, and Yun Fu. "Robust Transfer Metric Learning for Image Classification." *IEEE Transactions on Image Processing* (2016): 1. Web.

[14] Li, Zengxi, Yan Song, Ian Mcloughlin, and Lirong Dai. "Compact Convolutional Neural Network Transfer Learning for Small-scale Image Classification." *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2016): n. pag. Web.

[15] Jason Yosinski, Jeff Clune, Yoshua Bengio: "How transferable are features in deep neural networks?", 2014, Advances in Neural Information Processing Systems 27, pages 3320-3328. Dec. 2014; [http://arxiv.org/abs/1411.1792 arXiv:1411.1792].

[16] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng: "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition", 2013; [http://arxiv.org/abs/1310.1531 arXiv:1310.1531].

[17] Mutka, Andrew M., and Rebecca S. Bart. "Image-based Phenotyping of Plant Disease Symptoms." *Frontiers in Plant Science* 5 (2015): n. pag. Web.

[18] Tsaftaris, Sotirios A., Massimo Minervini, and Hanno Scharr. "Machine Learning for Plant Phenotyping Needs Image Processing." *Trends in Plant Science* 21.12 (2016): 989-91. Web.

Aniruddha Tapas is pursuing Bachelor of Engineering in Computer Science & Engineering from Shri Ramdeobaba College of Engineering and Management, Nagpur, India. His research interests include Machine Learning, Computer Vision and Artificial Intelligence.