Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 152 (2019) 186–194

Procedia Computer Science

www.elsevier.com/locate/procedia

International Conference on Pervasive Computing Advances and Applications – PerCAA 2019

Deep Residual Networks for Plant Identification

Vinit Bodhwani[a], D. P. Acharjya[a], Umesh Bodhwani[a]

[a]*School of Computer Science and Engineering, Vellore Institute of Technology, Vellore and 632014,India*

Abstract

Advancing the knowledge and understanding of plants growing around us plays a very important and crucial role in medicinally, economically and in a sustainable agriculture. The identification of plant images in the field of computer vision has become an interdisciplinary focus. Taking benign conditions of quick advancement in computer vision and deep learning algorithms, convolutional neural networks (CNN) approach is taken into consideration to learn feature representation of 185 classes of leaves taken from Columbia University, the University of Maryland, and the Smithsonian Institution. For the large-scale classification of plants in the natural environment, a 50-layer deep residual learning framework consisting of 5 stages is designed. The proposed model achieves a recognition rate of 93.09 percent as the accuracy of testing on the LeafSnap dataset, illustrating that deep learning is a highly promising forestry technology.

Keywords:Plant Classification; Convolution Neural Networks; Data Augmentation; Computer Vision

1. Introduction

Plants are the essential resource for human well-being providing us with oxygen and food. So, the researchers along with breeding industry are making great efforts to continue agriculture for a long period without any interruptions. A good knowledge of plants is essential to help fully recognize new, different or uncommon plant species in order to support the ecosystem, promote the drug industry, and promote sustainability and productivity in agriculture. Different modern and advanced models have been proposed for automatic plant identification as the deep learning technology advances. In addition, the classification and learning of an item in an image in computer vision is a really challenging task. Today, most researchers use different variants in leaf properties as a comparable method for the study of new plants, and some leaf datasets, including Flavia, Swedish and ICL, have been transmitted, and fewer attempts have been made to extract local features of leaf, flower or fruit. However, much work has been conveyed towards identification and prediction in different applications. Additionally, computational intelligence techniques play a vital role in identification and prediction applications. For example, rough computing is hybridized with neural network [1, 2], genetic algorithm [3, 4], and soft set [5].

10.1016/j.procs.2019.05.042

Deep convolutional neural networks play a vital role in order to learn distinct features of an image using image classification techniques. To enhance the quality of images, we need to increase their resolution, which results to an establishment of deeper neural networks and increasing the number of stacked layers of the network become very crucial as mentioned in recent evidence [6,7]. These large number of hidden layers establish the problem of vanishing gradient descent [8] which could not be resolved by traditional machine learning techniques. To overcome aforementioned challenges and invigorated by the deep learning breakthrough in image recognition, a 50-layer deep learning model using residual networks is designed for uncontrolled plant identification on the LeafSnap dataset which consists of 185 different tree species. The model proposed achieves an accuracy rate of 93.09 percent with 0.24 percent error.

2. Literature Survey

The detection of plants plays an important role in helping different or unusual plant species to increase drug trafficking, maintain the ecosystem and increase agricultural productivity and property. Neeraj Kumar et al. [9] proposed a recognition system of images for plant species which can be identified automatically namely LeafSnap. They built a mobile app that assists users to recognize trees by capturing photos of their leaves and their current version covers all species of the given dataset. They used an indispensable measure to calculate the various functions of the curvature-based shape of the leaf at the border and based on these characteristics identified plants. Their recognition process mainly consists of four parts - classifying, segmenting, extracting and comparing. Furthermore, they followed nearest neighbors (NN) approach for identification of leaf type.

Sue Han et al. [10] employed the well trained convolutional neural network model to perform plant identification. They proposed a methodology to recognize the learned features using deconvolution networks (DN), instead of using the CNN. This approach was used to obtain visual perception of the features needed to recognize a leaf from various classes, thereby evading the need to manually design the features. They worked on a new dataset called MalayaKew Leaf Dataset with only 44 classes. They created a new dataset (called as D2) by extending the given dataset(D1) by cropping manually and rotating the images. They selected 34672 leaf patches for training and 8800 for testing randomly, which resulted in 99.6% accuracy on the D2 dataset and 97.7% on the D1 dataset.

For the purpose of leaf detection at various scales of plants, Jing Hu et al. [11] proposed a MSF-CNN (Multi-Scale Fusion Convolutional Neural Network). They down-sampled an input image with a list of bilinear interpolation operations into multiple low-resolution images. The images were then fed into the MSF- CNN architecture to learn different characteristics in various layers, step by step. The final feature for anticipating the input image plant species is obtained by aggregating all last layer information. They re-trained the DeepPlant on D1 dataset and predicted classes with an accuracy of 98.1% using Support Vector Machine(SVM) model and 97.7% accuracy using Multi-Layer Perceptron approach(MLP). They observed that some of the classes were misclassified and concluded that identifying the shape of plants is not a good choice to recognize plants. Then they trained their model on the D2 dataset and achieved 99.6%, which is higher than the D1 dataset. They concluded that the D2 has better performance than D1, is because venation of distinct orders is a more robust characteristic for plant recognition.

Mohamed Abba et al. [12] modified a trained model to visually identify the leaves in images. They showed how a model can be used on a small training dataset that is already trained on a large dataset. Its result was that the traditional machine learning methods were outperformed with the use of local binary patterns (LBPs). They didn't train their model from scratch and instead, took a CNN model trained on ImageNet. They worked on an ImageClef2013 dataset which includes images of clean as well as the cluttered background. Due to the shortage of the training data, it led to the problem of overfitting and high variability. They, therefore, applied transfer learning to avoid overfitting and made the AlexNet fine-tuning with the help of Caffe framework. They compared AlexNet from scratch with the help of random initialization with the fine-tuning versions which resulted in an accuracy of 71.17% on validation dataset and 70.0% on the testing dataset.

Yu Sun et al. [13] took BJFU100 dataset which consists of 100 species of ornamental plants where each category contains 100 different photos of 3120 × 4208 resolution. They compared a ResNet26(26 layers) with ResNet model of 18,34 and 50 layers and recommended ResNet26 is the best tradeoff between optimization difficulty and model capacity. They explained ResNet26 contains enough trainable parameter to learn the discriminative feathers, which

prevents both under-fitting and over-fitting and thus, resulted in fast and robust convergence during SGD optimization. Their model achieved 91.78% accuracy which enhances the overall performance up to 2.51%.

Kaiming He et al. [14] presented a framework for residual learning to speed up network training. It leads to a problem of vanishing gradient descent by increasing the number of hidden layers in the residual network. Therefore, they introduced a concept of shortcut connection which results in a high accuracy and lower complexity of their model. On the ImageNet dataset, they developed a residual net model with a depth of up to 152 layers (8 times deeper than VGG nets). They made a 3.57% error in the test set after using these residual networks.

In this paper, Residual networks of 50 layers are being chosen to train the model on the LeafSnap dataset. Since the dataset wasn't enough, we applied data augmentation techniques using open source library – Augmentor [15] which rotates flips, zoom and resized the given images into 64x64x3 dimensions. We tried to train our model on CPU, but it trained only 60% of our dataset in 24 hours. So, we trained our model on Google Colab K80 GPU [16] which took 4-5 hours in training. At final, we achieved an accuracy of 98.75% on training dataset with a loss rate of 0.0462% and 93.0926% accuracy on testing dataset with loss rate 0.247%.

3. Foundations of Convolutional Neural Network

The advantage of a very deep network is that it can learn distinct features of images at many different levels of abstraction and thus can represent very complex functions. However, it is not at all easy to work with deep networks as it leads to a problem of vanishing gradients thus making gradient descent unbearably too slow. In gradient descent, we backpropagate from the last layer back to the first layer (input layer) and multiply by the weight matrix [17] on each step (including hidden layers also) which makes the gradient, decrease exponentially quickly to zero as shown in Fig. 1.

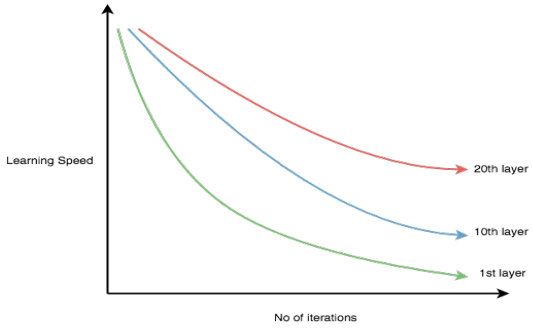

Figure 1. Problem of vanishing gradient descent as number of iterations increases.

Convolutional neural networks (CNN) have been successfully used in patter recognition and image classification techniques[18,19]. They are widely used to train especially structured networks on extensive data sets and the availability of parallel computing hardware [20]. Deep convolutional layers help to avoid the extraction process of features that is essential in classical algorithms for machine learning [21,22]. However, to avoid overfitting, deep neural networks require a lot of data. When the training data is insufficient and limited, alternative approaches are much needed [23,24].

3.1 Residual Networks

As we have got a huge number of images and classes, traditional CNNs approach wouldn't work as it requires high extensive memory resources, time and computational power [25, 26]. The chosen dataset leads to deep network and thus we need to use Residual Networks. The residual learning framework expedites the training of deep networks resulting in improved performance in each visual and non-visual task. These residual networks are a lot of

deeper than their 'plain' counterparts, nonetheless, they need an identical variety of parameters (weights). Fig. 2 shows basic building block without skip connection and residual building block with skip connection. Mathematically it is defined as:

$$H(x) = F(x) + x \quad (1)$$

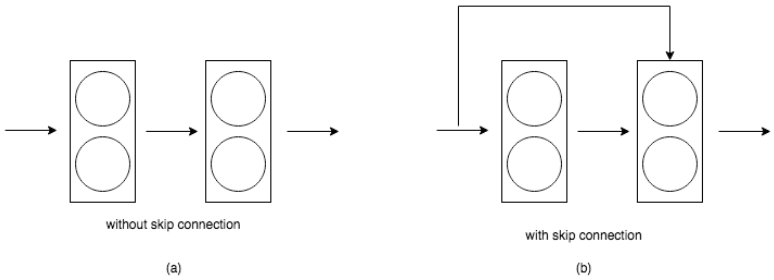

Figure 2. (a) A basic building block. (b) A residual building block

At Eqn. (1), x and H(x) are defined as the subnetwork's input and true output, and the F(x) function represents the residual mapping to be learned. We are explicitly forcing the network to learn an identity mapping by learning the input and output of subnetworks. The major difference between traditional neural nets and residual networks is traditional neural nets will try to learn H(x) directly, whereas ResNet model the layers to learn the input and output of subnetworks.

The overall network architecture of our 50-layer ResNet (i.e. ResNet50) model is depicted in Fig. 3. The model is mainly designed with the help of 5 stages. Initially, zero-padding pads the input image of dimension 64x64x3 with a pad of size (3,3). The resulted image is fed into a 7 × 7 convolution layer followed by batch normalization applied to the channels axis of the input and a 3 × 3 max pooling layers. In stage 2, the convolutional block uses three sets of filters of size (64×64×256) and stacking 2 identity blocks together using three sets of filters of size (64×64×256). The dimension of images increases suddenly and in the last stage, the convolutional block uses three sets of filters of size (512×512×2048) followed by stacking 2 identity blocks which use three sets of filters of size (256×256×2048). The network finally ends with an average pooling and the output is flattened by adding a fully connected layer that reduces the number of classes with softmax activation.

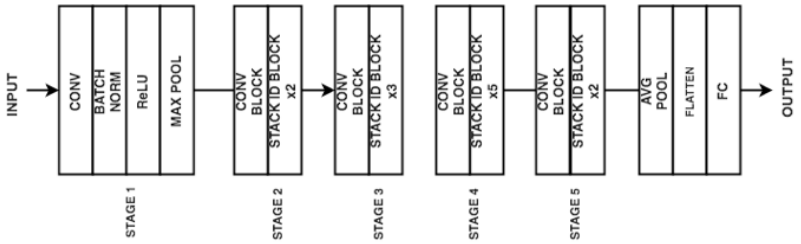

Figure 3. Architecture of our model ResNet50

4. Proposed Research Design

The proposed research design consists of 50-layer deep residual network. Each layer consists of an identity block and convolutional block. The identity block is the standard block used in ResNets, and corresponds to the case where the input activation and output activation have the same dimensions. We have executed a more powerful version (as shown in Fig. 4), in which 3 hidden layers are skipped rather than 2 which makes our model unique as it takes less time and results in a good accuracy.

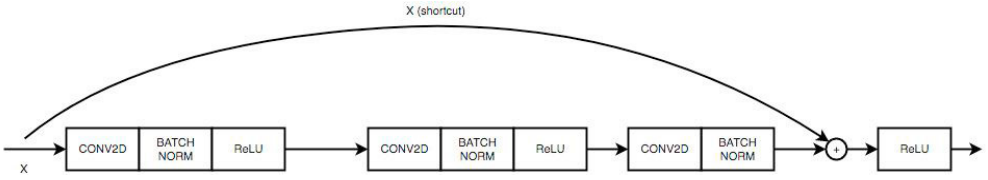

Figure 4. The identity block

The purpose of this block is to match input and output dimensions. The difference with the identity block is that in this block, there is an additional CONV2D layer in the shortcut path as shown in Fig.5. To resize the input to the different dimension, we used the CONV2D layer in the shortcut path so that the dimensions match up in the final addition required to add the shortcut value back to the main path.

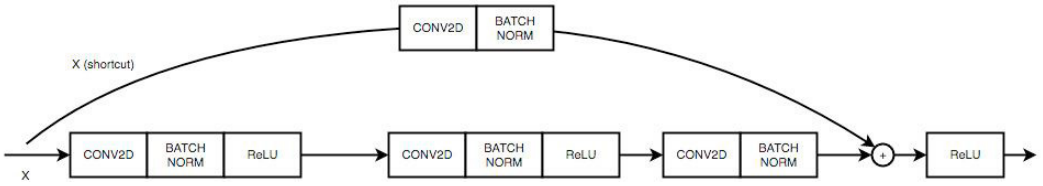

Figure 5. The convolutional block.

5. Implementation

The implementation of our model is based on an open source deep learning framework keras. All the experiments were conducted on Google's Cloud platform having Tesla K80 GPU processor. We used an Augmentor library, which is a standalone library in Python and is more convenient because it enables finer grain control over augmentation and implements the most relevant augmentation techniques in the real world. It uses a stochastic approach that uses building blocks to put operations together in a pipeline. The data set size [27] has been increased by rotating, zooming and rotating the images to improve accuracy. All the images were resized into the 64x64x3 dimension and then per- pixel value is divided by 255.

Algorithm 1: Augmentation of images

initialization:

N: no of images.

width: width of images we want to change.

height: height of images we want to change.

sample: after mentioning specific changes, we can sample it which will generate N augmented images based on our specifications.

path: path of the dataset where all images are present

begin

Initialize an empty pipeline

image←pipeline.Augmentor(path)

image←pipeline.rotate(max_left_rotation, probability, max_right_rotation)

image←pipeline.flip_left_right(probability)

image←pipeline.zoom_random(percentage_area=0.8, probability)

image← pipeline.flip_top_bottom(probability)

image←pipeline.resize(probability, width, height)

sample←pipeline.sample(N←27000)

5.1 Dataset distribution and converting into HDF5 format

We converted our dataset (which was consisted of images) into Hierarchical Data Format 5(hdf5) which is a set of file formats designed to store and organize huge volumes of data. We converted our dataset into hdf5 format because of its hierarchical structure (similar to files/folder) and its rate of compression and fast accessibility. After that, we prepared 150 images of each class (out of total 180 classes) which results in 27,000 images in total. We then split 80% of our dataset (21600 images) as training and 20% (5400 images) as testing.

Algorithm 2: Actual implementation of Residual networks
initialization:
filters: the number of filters in the CONV layers of the main path
image: input image
input_shape: images' shape
classes: number of classes, integer
epochs: no of epochs
batch_size: number of training examples utilised in one iteration
F: no of filters
begin:
training_dataset, test_dataset← load dataset
training_dataset← normalize training dataset by dividing pixels values with 255.
testing_dataset← convert test dataset using one hot encoding
model← call ResNet50 function with input_shape← 64x64x3 and classes ← 180
model← compile model using 'adam' optimizer and 'categorical_crossentropy' loss value.
fit model with training_dataset, testing_dataset, epochs and batch size as parameters
Function ResNet50(input_shape, classes)
Stage 1:
image←zero_padding(padding_shape)
image← 2D_Convolution(F,filters)
image←Relu_Activation(image)
image←max_pooling(window_shape)
Stage 2:
filter←64x64x256
stage←2
image← call convolutional_block function with image, filter, stage and block←'a'asparamenters.
image← call convolutional_block function with image, filter, stage and block←'b'asparamenters
image← call convolutional_block function with image, filter, stage and block←'c'asparamenters
Stage 3:
filter←128x128x512
image← call convolutional_block function with image, filter, stage and block←'a'asparamenters
image← call identity_block function with image, filter, stage and block←'b'asparamenters
image← call identity _block function with image, filter, stage and block←'c'asparamenters
image← call identity _block function with image, filter, stage and block←'d'asparamenters
Stage 4:
filter←256x256x1024
stage←4
image← call convolutional_block function with image, filters, stage and block←'a'asparamenters
image← call identity_block function with image, filters, stage and block←'b'asparamenters
image← call identity _block function with image, filters, stage and block←'c'asparamenters
image← call identity _block function with image, filter, stage and block←'d'asparamenters
image← call identity _block function with image, filter, stage and block←'e'asparamenters
image← call identity _block function with image, filters, stage and block←'f'asparamenters

Stage 5:
filter← 512x512x20148
image← call convolutional_block function with image, filters, stage←5 and block←'a'asparamenters
image← call identity_block function with image, filters, stage←5 and block←'b'asparamenters
image← call identity _block function with image, filters, stage←5 and block←'c'asparamenters
image←average_pooling(pool_size, padding_shape)
image← convert output into categorical values using softmax activation function
model← compile

Function identity_block(image, filters)
prev_image← image
image← 2D_Convolution(filters)
image←BatchNormalization(image)
image←Relu_Activation(image)
image← Add(prev_image, image)
image←Relu_Activation(image)
return image

Function convolutional_block(image, filters)
prev_image← image
image← 2D_Convolution(filters)
image←BatchNormalization(image)
image←Relu_Activation(image)
prev_image← 2D_Convolution(filters)
prev_image←BatchNormalization(prev_image)
image← Add(prev_image, image)
image←Relu_Activation(image)

6. Result Analysis

We present a performance assessment of our CNN model using residual networks in this section. We used "Adam" optimizer and "categorical_crossentropy" loss function as these are good for a model to predict multiple mutually-exclusive classes. The configuration parameters of our model are depicted in Table 1.

Table 1. Configuration of the parameters used during training.

Parameter	Value
Image size	64x64x3
Optimizer	Adam
Learning rate	0.001
Batch Size	120
Epochs	41

Fig.6(a) shows the training process of our ResNet50 model (of 50 layers). After the very first epoch, training accuracy improves quickly and stabilizes after 32 epochs. Our model yield 99.07% training accuracy and error rate started reducing from 1.3133% to 0.0344% as shown in Fig. 6 (b). The proposed ResNet50 results in 93.09% testing accuracy with a loss rate of 0.247% as shown in Table 2.

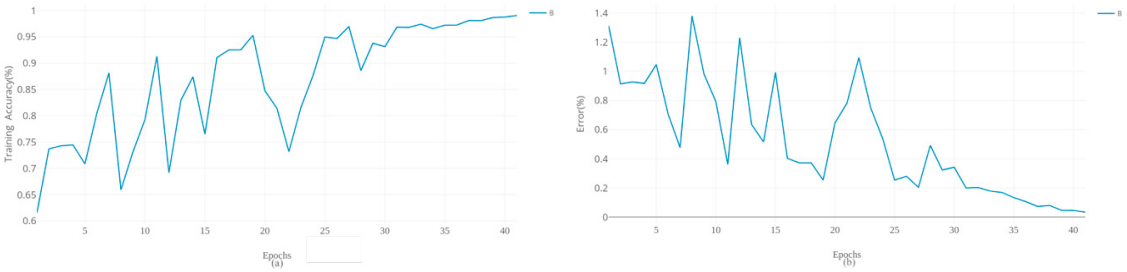

Figure 6. (a) Evolution of classification accuracy on training set. (b) Training Error Rate

Table 2. Performance evaluation of our model

Process	Number of images	Accuracy (%)	Error (%)
Training	21600	99.07	0.034
Testing	5400	93.09	0.247

7. Conclusion

This paper examined a methodology of deep learning using CNN to learn discriminatory characteristics for plant categorization from leaf images. We demonstrated how skip-connections in residual networks help to address the Vanishing Gradient problem which resulted in 93.09% accuracy in the test set, manifesting that deep learning is the promising technology for large-scale plant classification in the natural environment. We plan to try to evaluate different CNN architectures in our future work to increase performance. In particular, we plan to extend the deep learning model to include prediction, disease segmentation, insect detection and so on from the classification task.

Data Reference: Neeraj Kumar, Peter N. Belhumeur, Arijit Biswas, David W. Jacobs, W. John Kress, Ida C. Lopez, João V. B. Soares, LeafSnap Dataset, 1.0, 2014, http://leafsnap.com/dataset/

References

[1] A. Anitha and D. P. Acharjya. (2015) " Neural Network and Rough Set Hybrid Scheme for Prediction of Missing Associations." *International Journal of Bioinformatics Research and Applications*, Vol. 11 (6), pp. 503-524.
[2] A. Anitha and D. P. Acharjya, (2018) "Crop Suitability Prediction in Vellore District using Rough Set on Fuzzy Approximation Space and Neural Network." *Neural Computing & Applications*, Vol. 30 (12), pp. 3633–3650, DOI: 10.1007/s00521-017-2948-1.
[3] R. Rathi and D. P. Acharjya, (2018) "A Rule Based Classification for Vegetable Production for Tiruvannamalai District using Rough Set and Genetic Algorithm." *International Journal of Fuzzy System Applications*, Vol. 7 (1), pp. 74 – 100.
[4] R. Rathi and D. P. Acharjya, (2018) "A Framework for Prediction using Rough Set and Real Coded Genetic Algorithm." *Arabian Journal for Science and Engineering*, Vol. 43 (8), pp. 4215 – 4227.
[5] T. K. Das and D. P. Acharjya, (2014) "A Decision Making Model using Soft Set and Rough Set on Fuzzy Approximation Spaces." *International Journal of Intelligent Systems Technologies and Applications*, Vol. 13 (3), pp. 170-186.
[6] K. Simonyan and A. Zisserman, (2015) "Very deep convolutional networks for large-scale image recognition." *International Conference on Learning Representations*, arXiv:1409.1556.
[7] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Er-han, V. Vanhoucke, and A. Rabinovich, (2015) "Going deeper with convolutions." *Computer Vision and Pattern Recognition*, pp. 1–9, DOI: 10.1109/CVPR.2015.7298594.
[8] X. Glorot and Y. Bengio. (2010) "Understanding the difficulty of training deep feedforward neural networks." *Artificial Intelligence and Statistics*, Vol. 9, pp. 249-256.
[9] Neeraj Kumar, Peter N Belhumeur, Arijit Biswas, David W Jacobs, W John Kress, Ida C Lopez, and João VB Soares, (2012) "Leafsnap: A computer vision system for automatic plant species identification." *European Conference on Computer Vision*, Vol. 7573, pp. 502–516, Springer.
[10] S. H. Lee, C. S. Chan, P. Wilkin, and P. Remagnino, (2015) "Deep- plant: Plant identification with convolutional neural networks." *International Conference on Image Processing (IEEE)*, pp. 452–456, DOI: 10.1109/ICIP.2015.7350839.
[11] Jing Hu, Zhibo Chen*, Meng Yang, Rongguo Zhang and Yaji Cui, (2018) "A Multi-Scale Fusion Convolutional Neural Network for Plant Leaf Recognition." *IEEE Signal Processing Letters*; vol. 25, pp. 853 – 857.
[12] Mohamed Abbas Hedjazi, IkramKourbane, YakupGenc, (2017) "On Identifying Leaves: A Comparison of CNN with Classical ML Methods." *Signal Processing and Communications Applications Conference (SIU), IEEE*, pp. 1-4, DOI: 10.1109/SIU.2017.7960257.
[13] Yu Sun, Yuan Liu, Guan Wang, and Haiyan Zhang, (2017) "Deep Learning for Plant Identification in Natural Environment." *Computational*

Intelligence and Neuroscience, pp: 1-6, DOI: 10.1155/2017/7361042.
[14] Kaiming He, Xiangyu Zhang, ShaoqingRen, Jian Sun, (2016) "Deep Residual Learning for Image Recognition." *Computer Vision and Pattern Recognition (CVPR)*, IEEE, pp. 770-778, doi: 10.1109/CVPR.2016.90.
[15] https://github.com/mdbloice/Augmentor.
[16] https://medium.com/deep-learning-turkey/google-colab-free-gpu-tutorial-e113627b9f5d.
[17]Nan Cui, (2018) "Applying Gradient Descent in Convolutional Neural Networks." *Journal of Physics: Conference Series,* Vol. 1004(1), DOI: 10.1088/1742-6596/1004/1/012027.
[18] Liangji Zhou, Qingwu Li, GuanyingHuo, Yan Zhou, (2017) "Image Classification Using Biomimetic Pattern Recognition with Convolutional Neural Networks Features." *Computational Intelligence and Neuroscience,* DOI: 10.1155/2017/3792805.
[19] DeepikaJaswal, Sowmya.V, K.P.Soman, (2014) "Image Classification Using Convolutional Neural Networks." *International Journal of Advancements in Research &Technology*,vol.3.
[20] K.Abdelouahab, M.Pelcat, J.Serot, C.Bourrasset, J.C.Quinton, F.BerryHardware, (2017) "Automated Datafow Deployment of CNNs." *ArXiv*.
[21] Minh Tuan Pham, PhucHao Do, Kanta Tachibana. (2016) "Feature extraction for classification method using principal component based on conformal geometric algebra." *International Joint Conference on Neural Networks (IJCNN),* pp. 4112-4117.
[22] Samina Khalid, Tehmina Khalil, ShamilaNasree, (2014) "A survey of feature selection and feature extraction techniques in machine learning." *Science and Information Conference (IEEE),*pp. 372-378, DOI: 10.1109/SAI.2014.6918213.
[23] Wei Zhao, (2017) "Research on the Deep Learning of the Small Sample Data based on Transfer Learning. *AIP Conference Proceedings,* 1864,DOI: 10.1063/1.4992835.
[24] Michael Cogswell, Faruk Ahmed, Ross Girshick, (2016) "Reducing Overfitting in Deep Networks by Decorrelating Representations." *International Conference on Learning Representations (ICLR);* arXiv:1511.06068.
[25] Maurice Yang, Mahmoud Faraj, Assem Hussein, Vincent Gaudet, (2018) "Efficient Hardware Realization of Convolutional Neural Networks using Intra-Kernel Regular Pruning." *ArXiv*; pp. 180-185, DOI: 10.1109/ISMVL.
[26] Yu Cheng, Duo Wang, Pan Zhou, Tao Zhang, (2017) "A Survey of Model Compression and Acceleration for Deep Neural Networks. *ArXiv*, arXiv:1710.09282.
[27] http://leafsnap.com/dataset/