Aerial Cactus Identification using Deep Learning

Umang J. Patel[a], Vishal Yadav[b], Aswin Karthik TH.[c], and Kotla Nikhil[d]

***Abstract*—In this work, we investigate various convolutional neural network architectures for the aerial cactus identification task. Our main effort is a thorough experimentation and evaluation of performance of different networks to recognize a specific type of cactus in aerial imagery. Here, we present a deep-learning based approach for columnar cactus recognition. The dataset consisting of 21,500 images, was retrieved via a competition hosted by Kaggle and is a part of the VIGIA project [1]. The proposed approach uses various convolutional neural network models like DenseNet [2], InceptionV3 [3], ResNet [4] and VGG16 [5] for the respective problem. Experiment results have shown a high recognition accuracy of 1.0 and 0.99 ROC/AUC score, validating the use of the approach to achieve state-of-the-art results for the columnar cactus recognition**

***Index Terms*—Deep learning, cactus, Convolutional neural network, ROC/AUC score**

I. Introduction

THIS work is a part of the VIGIA project conducted by researchers in Mexico. The project aims to build a system for autonomous surveillance of protected areas. A first step in such an effort is the ability to recognize the vegetation inside the protected areas. Given the temporary restrictions of the whole project, it is intended to study and implement deep learning algorithms to detect automobiles and humans in an image captured by an UAV. It is assumed that he detection of humans or automobiles is sufficient to indicate human activity in the specified regions.

Deep learning (DL) based on Convolutional Neural Networks (CNN) has become a powerful technique to solve various pattern recognition problems such as object detection, face recognition, or image captioning. Unlike other classical machine learning algorithms like Support Vector Machines (SVM), Logistic Regression, K-Nearest Neighbours (KNN) where several features are extracted manually like shapes and edges and then passed to a classifier, DL automatically extracts the relevant and important features and performs a logistic regression.

In this work, we present a deep-learning based approach for recognizing the cactus species in the given images. The VIGIA project segregated this task to two parts. First part is to recognize these species and the other part, is to extend the system to segment and recognize other species in the future. In addition, the researchers of the VIGIA project have focused on the method of recognizing cactus from aerial images taken by small drones due to their low cost.

Several challenges are presented in our work. One of them is the small area that the cactus occupies in the given images. Another one is that they are of low resolution (32 x 32) and RGB channels. Moreover, the images seemed to be quite blurry and required some augmentation techniques to deal with various scenarios the cactus can be identified in the image.

Our contributions are: i) comparisons of CNN models for the cactus recognition problem, ii) validation of the models used in the problem, and iii) the analysis of the relevant features for cactus recognition.

II. Related Work

Autonomous plant detection has been an active research topic for many years, provided its importance in various fields such as preservation of natural areas, precision agriculture, disease detection. A large variety of approaches have been taken to address this problem, among which stand out various machine learning algorithms like Support Vector Machines, Random Forest, Multilayer Perceptron. These algorithms require manual feature engineering processes and complex extraction techniques. Thus, using these techniques, pattern recognition was difficult and complex to perform as it was time-consuming.

Enter the deep learning era, the model learns and extracts the relevant features directly from the images during the training step.

[a:] Umang J. Patel is with the Charotar University of Science and Technology, Gujarat, India (e-mail: umangpatel1947@gmail.com).
[b:] Vishal Yadav is with the Galgotias University, Greater Noida, India (e-mail: vishal.93101@gmail.com).
[c:] Aswin Karthik TH. is with the Sri Ramakrishna Engineering College, Tamil Nadu, India (e-mail: aswinharitharan@gmail.com).
[d:] Kotla Nikhil is with the S.R.K.R Engineering College, Andhra Pradesh, India (e-mail: kotlanikhil11@gmail.com).

CNNs have outperformed the traditional machine learning approaches for the task of image classification, what has led to high performance in various tasks like object detection, face recognition or image captioning.

An approach of deep learning is transfer learning. It is a well-known fact that deep learning requires large amount of data to be trained on, this has led to the use of pre-trained networks that are later fine-tuned with specific data of the problem addressed. The aim is to take advantage of the knowledge learnt from one problem and apply it to a closely-related problem. We have used recognized architectures such as DenseNet, VGG16, InceptionV3 and ResNet, to identify the presence of cactus in the given images.

Transfer learning enables faster experimentation on a closely-related image recognition tasks and train models in $1/100^{th}$ of the time in case of regular neural network model training. It turns out that ImageNet is already so good at recognizing objects in the real-world, therefore, it is a perfect candidate to make a very good image classifier.

Utilizing transfer learning, we have achieved the state-of-the-art result with 1.0 accuracy on the training set and 0.99 accuracy on the validation set, compared to 0.99 accuracy on the training set and 0.95 accuracy on the validation set as cited by the original paper present in the VIGIA project.

III. Methodology

In this section, we present multiple deep learning approaches for automatic cactus recognition. Deep learning is a machine learning technique that involves automatic feature extraction for the task. We have applied the supervised learning approach wherein we use large number of examples for training the neural network with associated truth labels for minimizing the error on predictions via a loss function and correcting model using an optimization algorithm. We have use various CNN architecture with pretrained

A. DenseNet

Umang J. Patel has followed the approach of using transfer learning on the DenseNet CNN architecture for the task of aerial cactus recognition. It takes input which is an image of size 128 x 128 pixels and outputs the probability of cactus present in the image. For this approach, a DenseNet architecture comprised of 161 layers was utilized. See figure 1 for the representation of a deep DenseNet with three dense blocks.

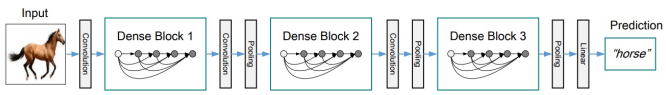

Fig. 1. A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling.

The DenseNet 161 [6] network is configured as follows. It receives as input a 3-channel image with resolution 128 x 128 x 128. Then, 3 convolution filters with size 7 x 7 are applied with stride two and padding three. Next, a max pooling operation is performed, using a kernel of size 3 x 3 with stride two and padding one. Then, the resultant latent space with passed through four dense blocks of layer configuration 6, 12, 36 and 24 having a growth rate k = 48. Eventually, an average pooling operation is applied of kernel size 1 x 1 and then the feature space is flattened to one-dimension vector. Finally, the feature space is passed to logistic regression module giving only a single output value representing the confidence of the prediction made by the model overall.

For the architecture, the binary cross entropy loss function was used with Adam optimization algorithm [7] for the backpropagation step during the training process.

$$\mathcal{L}(a, y) = -\{y * log(a) + (1 - y) * log\,(1 - a)\}$$

Here, a represents the predicted value y represents the truth value and $\mathcal{L}(a, y)$ represents the error for the prediction made by the model. This loss function is utilized during the training process in order to minimize errors made by the model during prediction via gradient descent optimization strategy

Moreover, it is to be noted that after each convolution operation, Batch Normalization [8] was performed in order to regularize the parameters of the network and reduce overfitting to a large extent. Training a neural network requires to set up a variety of hyper-parameters such as the learning rate, number of epochs and batch size. The learning rate is the most crucial one since it defines how much the weights are updated to decrease the loss. To solve this issue, this network was trained using a cyclical learning rate strategy to solve the local minima problem for the recognition task. The rest of the parameters are set empirically, more details are presented in the experiments section.

B. InceptionV3

Aswinkarthik has followed the approach of using transfer learning on the InceptionV3 CNN architecture for the task of aerial cactus identification. It takes an image as an input of dimensions 224 x 224 pixels and outputs the probability of the cactus being present in the image. This model is comprised of 48 layers. See the Figure 2 for the representation of the Inception V3 architecture.

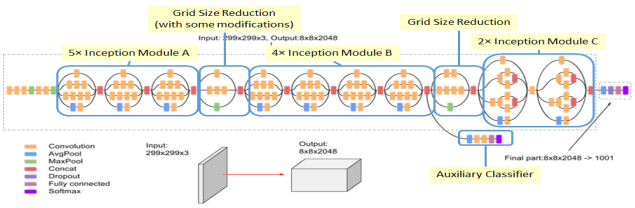

Fig. 2. A InceptionV3 architecture with blocks. Batch Norm and ReLU are used after Conv

The Inception model was introduced by GoogLeNet [9] in the year 2015. The input features pass through 3 convolutional filters of the kernel size 3 x 3 with stride two and padding three. Then, a max pooling operation is performed, using a kernel of size 3 x 3 with stride two and padding one. Moreover, the Adam optimizer is used, with binary cross-entropy function as the loss function for the model. For the batch normalization, the auxiliary classifier is utilized as the regularizer.

C. ResNet 50

Kotla Nikhil has followed the approach of using transfer learning on the ResNet CNN architecture for the task of aerial cactus recognition. It takes input which is an image of size 224 x 244 pixels and outputs the probability of cactus present in the image. For this approach, a ResNet architecture comprised of 50 layers was utilized. See Figure 3 for the representation of a deep ResNet with three dense layers.

Fig. 3. A block of ResNet with three dense layers.

ResNet converges faster compared to its plain counter-part of it. It achieves a top-5 validation error of 5.25 % which is better than Inception and VGG models on the ImageNet dataset. It won the ILSVRC-2015 [10] competition.

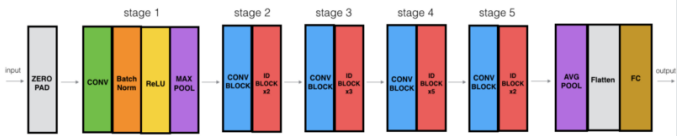

Fig. 4. Architecture of ResNet 50.

The ResNet50 model consists of 5 stages each with a convolution and identity block. Each convolution block has 3 convolution layers and each identity block also has 3 convolution layers. This model has over 23 million parameters to train. Further details are presented in the experiments section.

D. VGG 16

Vishal Yadav has followed the approach of using VGG16-CNN architecture for the task of aerial cactus recognition. It takes in input as an image of dimensions 224 x 224 pixels and outputs the probability of cactus being present in the aerial image. See Figure 5 for the representation of the VGG-16 architecture as follows.

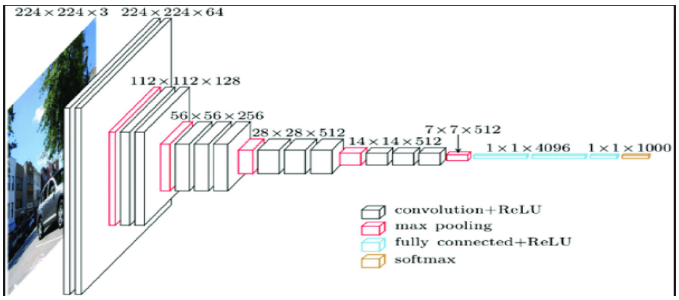

Fig. 5. VGG-16 architecture.

The VGG-16 architecture is configured as follows: the input to Conv1 layer is of fixed size 224 x 244 RGB image. The image is then passed through a stack of convolutional (Conv) layers, where the filters are of the size 3 x 3. It also utilizes 1 x 1 convolutional filters, which can be seen as a linear transformation of the input channels (followed by non-linearity). Spatial pooling is carried out by five max-pooling layers having 2 x 2 pixels window size with stride 2.

IV. Experimental Results

In this section, we present the experimental characterization of the proposed approaches. The implemented networks were trained on Kaggle Kernels having the NVIDIA Tesla P100 GPU hardware. The results of the experiments are further explained.

The dataset consists of 17500 images reserved for training phase and remaining 4000 images for the testing phase. The dataset can be visually plotted as follows.

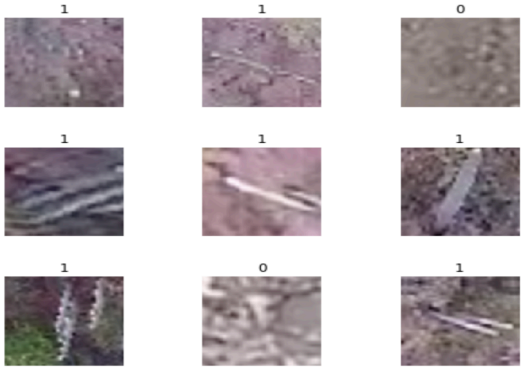

Fig. 6. Visual representation of the dataset.

A. DenseNet161

Umang J. Patel has implemented the DenseNet161 architecture using fast-ai and PyTorch libraries [11]. The hyperparameters used are: learning rate 0.03, epochs 5, batch size 64.

A common practice to improve the accuracy of a deep learning model is to do data augmentation. This helps in generalizing the model and reduce overfitting. The augmentation techniques used in this model are: horizontal and vertical flipping, warping, random rotations, contrast change, random zoom, random cropping. [12]

To train the model more rapidly, the fast-ai library offers a simple tool to identify the learning rate that can best train the model and suggest the minimum gradient on the learning rate curve as shown in Figure 7.

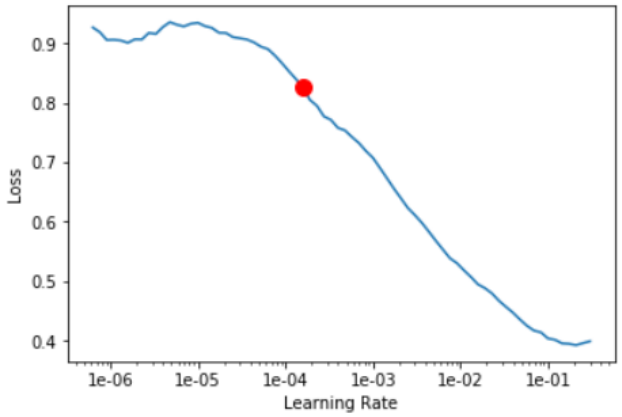

Fig. 7. Finding the optimal learning rate using cyclical learning rate strategy.

The model [13] used in this approach was trained for 5 epochs and used cyclical learning rate [14] strategy bounded by the maximum learning rate of 0.03 taken during each cycle. The training process is tabulated as follows in Figure 8.

epoch	train_loss	valid_loss	accuracy	error_rate	time
0	0.057346	0.079030	0.988571	0.011429	01:33
1	0.046300	0.000685	1.000000	0.000000	01:27
2	0.031888	0.090319	0.960000	0.040000	01:27
3	0.009858	0.000098	1.000000	0.000000	01:26
4	0.002915	0.000036	1.000000	0.000000	01:26

Fig. 8. Table consisting of losses, accuracy and error rate of the model during the training procedure.

To check whether the mode generalized well, 1% of the training was split randomly and results of confusion matrix for the model are shown in Figure 9.

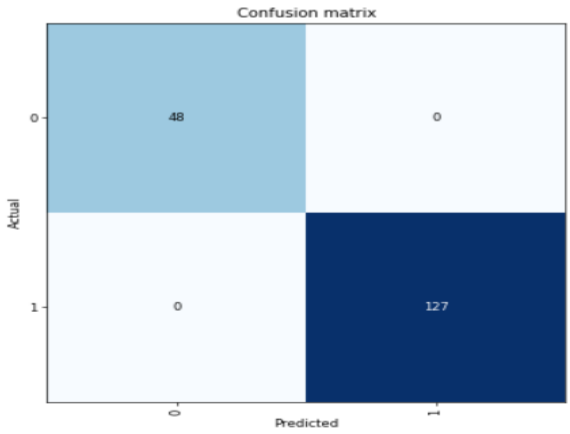

Fig. 9. Confusion matrix for the validation set of DenseNet161 model.

Moreover, the relevant features are extracted by the CNN model used in this approach and is visualized as shown in the Figure 10.

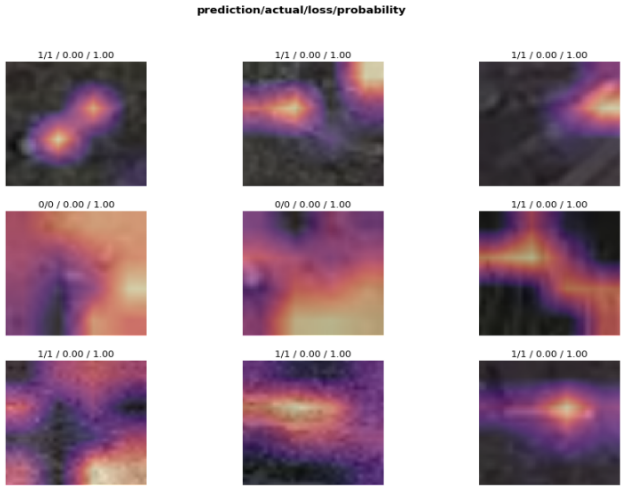

Fig. 10. Heatmap displaying the location of the relevant features of cactus present in the aerial imagery.

B. InceptionV3

Aswinkarthik has implemented the InceptionV3 architecture using Keras and TensorFlow [15] packages. The hyperparameters used are: learning rate 0.0001, epochs 100, batch size 250.

Again, similar to DenseNet 161 approach, data augmentation was necessary to reduce overfitting and make the model more generalized to new data. Techniques such as horizontal flipping, random height and width change, random zoom.

To check if the model performed better generalization, 10 % of the training data was used as the validation set. By this, the model obtained 1.0 training accuracy and 0.999 testing accuracy. Figure 11 depicts the accuracy and loss over the training phase as follows.

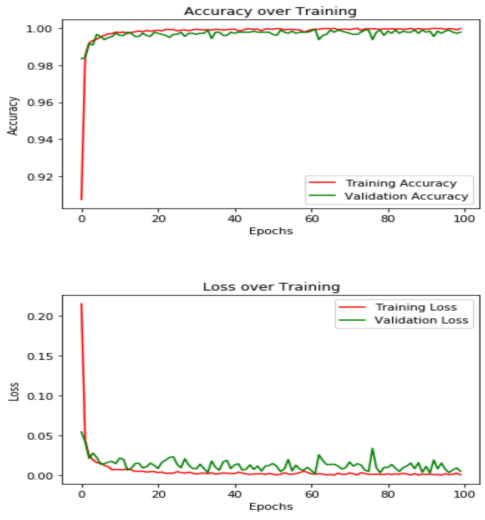

Fig. 11. Accuracy and loss curve over the period of training the InceptionV3 model.

C. *ResNet 50*

Kotla Nikhil has implemented the ResNet 50 architecture using both Keras (under TensorFlow) and Fast-ai (under PyTorch) libraries. The hyperparameters used in Keras implementation are: learning rate 0.01, epochs 10, batch size 32. The hyperparameters used in Fast-ai implementation are: learning rate 0.0003, epochs 6, batch size 128.

Again, similar to DenseNet 161 and InceptionV3 approaches, data augmentation was necessary to reduce overfitting and make the model more generalized to new data. Techniques such as horizontal flipping, random height and width change, random zoom.

Figure 12 illustrates the accuracy and loss curve over the training period for the Keras ResNet 50 implementation [16].

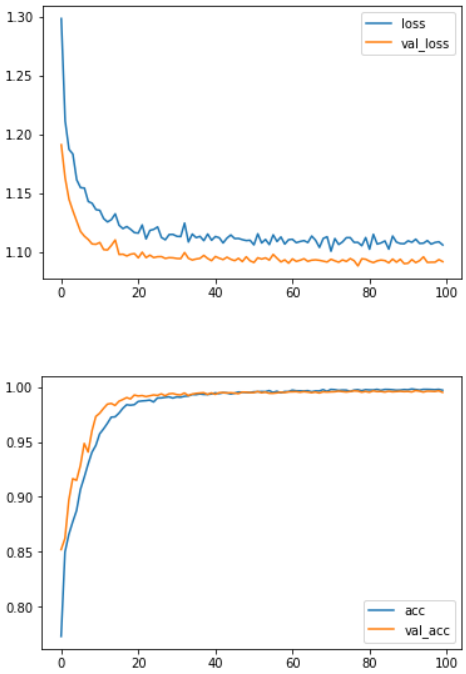

Fig. 12. Accuracy and loss curve over the period of training the Keras ResNet50 model.

Figure 13 illustrates the loss curve over the training phase of the Fast-ai ResNet50 implementation [17].

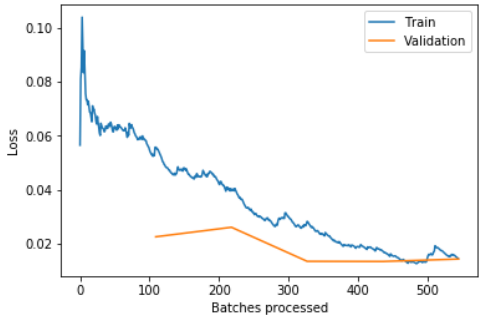

Fig. 13. Loss curve over the period of training the Fast-ai ResNet50 model.

To check whether the mode generalized well, all the testing set samples were used and results of confusion matrix for the model are shown in Figure 14.

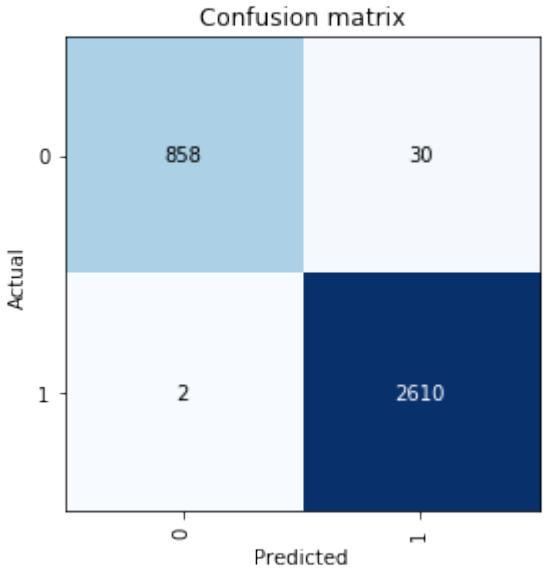

Fig. 14. Confusion matrix for the test set in Fast-ai ResNet 50 model.

D. *VGG16*

Vishal Yadav has implemented the VGG16 architecture using Keras and TensorFlow packages. The hyperparameters used are: learning rate 1.0, epochs 50, batch size 64.

Compared to previously discussed approaches, here the model has been trained from scratch, that is, without using transfer learning. Training the model took approximately 34 minutes using a Tesla P100 GPU available on the Kaggle Kernel.

Moreover, the model [18] uses the early stopping mechanism in order to prevent overfitting. In this process, whenever the validation loss starts to increase gradually compared to the training loss, the model stops the training procedure and saves the weights where the training and validation loss are minimum.

To check whether the model generalized enough, 10 % of the training data was randomly split to create the validation set. Figure 15 depicts the accuracy and loss curves for the implementation of the model.

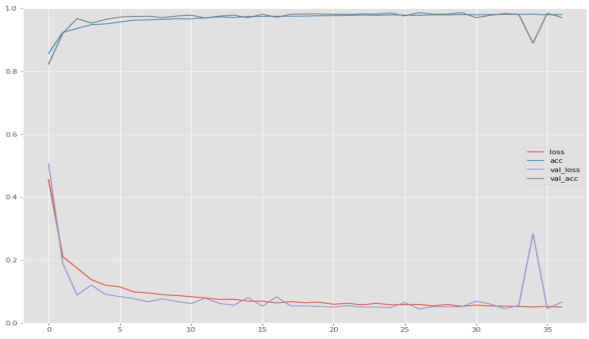

Fig. 15. Accuracy and loss curves over the training period for the VGG16 model.

V. Conclusion

We have compared the various deep learning approaches for columnar cactus recognition. In these approaches, we used transfer learning in order to obtain nearly correct predictions almost all the time. Our experiments have shown that the approaches reach a high recognition accuracy (0.99 for the testing set). The DenseNet161 and InceptionV3 approaches are ideal to perform automatic surveillance of the protected forest areas with drones.

Acknowledgment

The authors thank to Shambhavi Mishra for her help in providing the computational resources for the project.

References

[1] Irving Vasquez, "VIGIA: Autonomous Surveillance of Biosphere Reserves" Proposal : 2016-01-2341.
[2] G. Hang, Z. Liu, L. Maaten, and K.Q.Weinberger, "Densely Connected Convolutional Networks", arXiv:1608.06993v5
[3] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna *et al.,* "Rethinking the Inception Architecture for Computer Vision", arXiv:1512.00567v3.
[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition", arXiv:1512.03385v1
[5] K. Simoyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition", arXiv:1409.1556v6.
[6] Various Densenet implementations like DenseNet 121, DenseNet 161, DenseNet 169 and DenseNet 201 available on PyTorch https://github.com/pytorch/vision/blob/master/torchvision/models/densenet.py
[7] Diederik.P. Kingma and Jimmy Ba, "Adam : A Method for Stochastic Optimization", arXiv:1412.6980v9.
[8] Sergey Ioffe and Christian Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift", arXiv:1502.03167v3.
[9] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan., V. Vanhoucke, and A. Rabinovich, "Going Deeper with Convolutions", arXiv:1409.4842v1
[10] ILSCRV2015 – ImageNet Challenge, http://image-net.org/challenges/LSVRC/2015/
[11] Fast-ai library documentation, https://docs.fast.ai
[12] Augmentation in fast-ai, https://docs.fast.ai/vision.transform.html
[13] Implementation of DenseNet 161 model using fast-ai, https://www.kaggle.com/umangjpatel/aerial-cactus-cnn
[14] Leslie N. Smith, "Cyclical Learning Rates for Training Neural Networks", arXiv:1506.01196v6.
[15] Implementation of InceptionV3 model using Keras, https://www.kaggle.com/aswinharitharan/aerial-cactus-inception
[16] Implementation of ResNet50 model using Keras, https://www.kaggle.com/kotlanikhil222/kernel95b45cc5ce
[17] Implementation of ResNet 50 model using fast-ai, https://www.kaggle.com/srisravya484/aerial-cactus-identification-using-resnet50
[18] Implementation of VGG16 model using Keras, https://www.kaggle.com/vishal03092000/aerial-cactus-identi-vgg16