Capstone Project

Machine Learning Engineer Nanodegree

Divyansh Sharma

July 3rd, 2017

Definition

Project Overview

Tangles of kudzu overwhelm trees in Georgia while cane toads threaten habitats in over a dozen countries worldwide. These are just two invasive species of many which can have damaging effects on the environment, the economy, and even human health. Despite widespread impact, efforts to track the location and spread of invasive species are so costly that they're difficult to undertake at scale. Hydrangea common names hydrangea or hortensia is a genus of 70–75 species of flowering plants native to southern and eastern Asia (China, Japan, Korea, the Himalayas, and Indonesia) and the Americas. By far the greatest species diversity is in eastern Asia, notably China, Japan, and Korea. Most are shrubs 1 to 3 meters tall, but some are small trees, and others lianas reaching up to 30 m (98 ft) by climbing up trees. They can be either deciduous or evergreen, though the widely cultivated temperate species are all deciduous

Currently, ecosystem and plant distribution monitoring depends on expert knowledge. Trained scientists visit designated areas and take note of the species inhabiting them. Using such a highly qualified workforce is expensive, time inefficient, and insufficient since humans cannot cover large areas when sampling.

Fig 1. Invasive Hydrangea

There are many government organizations that monitor invasive species like hydrangea and other plants using various methods like mapping and tracking different areas, creating an inventory etc. Also many private organizations like astron are using drones, big data and computer learning to detect and predict invasive species.

This data set of foliage and forest images is publicly available on Kaggle to accurately identify hydrangea in the images. Techniques from computer vision alongside other current technologies like aerial imaging can make invasive species monitoring cheaper, faster, and more reliable.

Many researches in the area of classification in ecology has been done using random forest [1],and several other models including factor analysis, climate envelope, machine learning and expert model has been employed in the area of ecological classification with a comprehensive list available at arXiv. [2]Also a feature learning based approach [3] on high resolution images taken from UAV (unmanned aerial vehicles) is being done in the area of invasive species monitoring.

An extensive research in image classification using deep convolution network is possible through ImageNet, which is an image database organized according to the WordNet hierarchy (currently only the nouns), in which each node of the hierarchy is depicted by hundreds and thousands of images. A deep convolution network [4] for image classification using ReLu (Rectifier Linear Unit) and max pooling is discussed.

The aim of this project is to build a convolution neural network that classifies the presence of the invasive hydrangea in the image.

Problem Statement

The foliage and forest images data set was provided by certain individual contributors namely, Christian Requena Mesa, Thore Engel, Amrita Menon , Emma Bradley who are environmentalist and work for monitoring invasive species.

The goal of the project is to identify whether or not the image consist of invasive species hydrangea or not, thus this problem is a binary image classification problem.

The goal is to train a CNN that would be able to classify whether or not invasive species hydrangea is present or not.

As deep learning is very effective over the years in image classification a convolution neural network will be used to predict the required class label. We will be using transfer learning technique which is nothing but using weights from pre-trained neural network over large data sets. Many such CNN's like VCG16, VCG19,Inception-v3, Xception , ResNet50 etc. trained over imagenet challenge are available for public use .As CNN's require more data while training and as available data set is limited we will be using image augmentation techniques like rotating ,flipping ,Gaussian blur ,Inverting colors, cropping etc. on training data set images which are easily available through image processing libraries like opencv to generate few more images to train our CNN and fine tune its efficiency. As training pre-trained CNN like VCG19 etc are computationally costly we will be resizing and centering the image and working on grey scale images to make training computationally cheaper and feasible.

Metrics

The evaluation metric used by Kaggle to score in this competition is Area under ROC curve also called as **receiver operating characteristic curve**. ROC curve is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings.

$$TPR = \frac{TP}{TP+FN} \qquad FPR = \frac{FP}{FP+TN}$$

Submitted values will be evaluated on area under ROC curve between the predicted value and the observed target. To combine the FPR and the TPR into one single metric, we first compute the two former metrics with many different instances, then plot them on a single graph, with the FPR values on the abscissa and the TPR values on the ordinate. The resulting curve is called ROC curve, and the metric we consider is the AUC of this curve, which we call AUROC.

Analysis

Data Exploration

The data set contains pictures taken in a Brazilian national forest. In some of the pictures there is Hydrangea, a beautiful invasive species original of Asia. Based on the training pictures and the labels provided, the participant should predict the presence of the invasive species in the testing set of pictures.

The data set consists:

1. **Train Data**: A collection of 2295 images of size 1144(Height) x 866(Width) pixels, each having three channels for red, green, blue (RGB) in jpeg format.
2. **Train Labels**: A comma separated file (csv) containing name of the pictures (numbers) and class of image corresponding to that number (0 represents Hydrangea not present and 1 represents Hydrangea present)..
3. **Test Data**: A collection of 1531 images of size 1144(Height) x 866(Width) pixels, each having three channels for red, green, blue (RGB) in jpeg format ready to be labeled by the learning algorithm.

Training data consists 1448 images labeled with class 1,i.e. 1448 images contains invasive species and remaining 867 images represents class 0,i.e. invasive species not present. This represents that the classes are not balanced; balanced data is generally good for classification as machine learning algorithms work on majority rule. Notice that here the imbalance is not much that can create a bias for a particular class so balancing is not really necessary in this case. However, one case keep track the performance of unbalanced classification by using Precision/Recall which we will be using by creating a confusion matrix for the classifier built. Also, as Kaggle's testing set does not contain labels, we will be using validation sets to train and test our model. In project proposal I had chosen to use 10 fold cross validation over the training dataset, but due to large time in training CNN and computational resource cost I chose to split the data on 80-20 percent split, i.e. 80 percent of the training dataset will be used for training and 20 percent of training data will be used for validation.

Exploratory Visualization

Histograms represent the color distribution of an image by plotting the frequencies of each pixel values in the 3 color channels. In image classification histograms can be used as a feature vector with the assumption that similar images will have similar color distribution. Here we calculate the histograms for each image in the training set and find the result for the most similar image from the histograms with the Euclidean distance metric. Results for a randomly chosen sample image are given below:

Fig 2.Randomly selected sample having same label not invasive

Clearly the images are similar in the labels, but they don't look similar. However their histograms are quite similar.

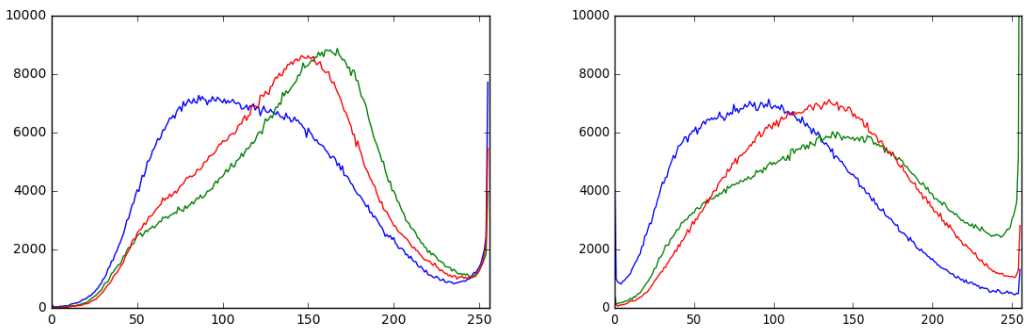


Fig 3.Color histogram comparison for randomly selected images of same label

Note that the benchmark model with k-nearest neighbors is also trained with the color histograms as features. However histograms completely ignore the shape, texture and the spatial information in the images and very sensitive to noise, so they can't be used to train an advanced model.

Algorithms and Techniques

Transfer Learning:

Transfer learning refers to the process of using the weights of a pre trained network trained on a large dataset applied to a different dataset (either as a feature extractor or by fine tuning the network). Fine tuning refers to the process of training the last few or more layers of the pre trained network on the new dataset to adjust the weight. Transfer learning is very popular in practice as collecting data is often costly and training a large network is computationally expensive. Here weights from a convolutional neural network pre trained on imagenet dataset is fine tuned to classify invasive species.

Benchmark method:

K-nearest classification: K-nearest classifier searches for the similar 'neighbors' of an instance (similar based on a given distance metric) and predicts the label based on the majority vote. K-nearest is simply used for establishing a benchmark based on the color histograms of the images as features.

VGG(16) Architecture :

Winner of the ImageNet ILSVRC-2014 (5) competition, VGGNet was invented by Oxford's Visual Geometry Group , The VGG architecture is composed entirely of 3x3 convolutional and maxpooling layers, with a fully connected block at the end. The pre trained model is available in Caffe, Torch, Keras, Tensorflow and many other popular DL libraries for public use.

Layers:

- **Convolution**: layers convolve around the image to detect edges, lines, blobs of colors and other visual elements. Convolutional layers hyperparameters are the number of filters, filter size, stride, padding and activation functions for introducing non-linearity.

- **MaxPooling**: Pooling layers reduces the dimensionality of the images by removing some of the pixels from the image. Maxpooling replaces a n x n area of an image with the maximum pixel value from that area to downsample the image.

- **Dropout** : Dropout (6) is a simple and effective technique to prevent the neural network from overfitting during the training. Dropout is implemented by only keeping a neuron active with some probability p and setting it to 0 otherwise. This forces the network to not learn redundant information.

- **Flatten** : Flatten the output of the convolution layers to feed into the dense layers.

- **Dense** : Dense layers are the traditional fully connected networks that maps the scores of the convolutional layers into the correct labels with some activation functions(softmax is used here).

Activation functions :

Activation layers apply a non-linear operation to the output of the other layers such as convolutional layers or dense layers.

- **ReLu Activation**: ReLu or Rectified Linear Unit computes the function

$$f(x) = \max(0, x).$$

- **Sigmoid Activation**: Sigmoid Function is applied to the output layer to convert the scores to monotonically increasing value from 0 to 1.

$$f(x) = \frac{1}{1 + e^{-x}}$$

Optimizers :

SGD: Stochastic gradient descent is also known as incremental gradient descent, is a stochastic approximation of the gradient descent optimization method. In other words, SGD tries to find minima or maxima by iteration.It is one of the most popular and widely used algorithm in machine learning applications and when used with back propogation it is the de facto training algorithm for neural networks.i have used SGD with learning rate as low as 0.0001 as epochs of the training was 50 large enough for a heavy trained neural net like VCG-16 to learn and momentum used was high i.e, 0.9 (range from 0 to 1) so that it increases for dimensions whose gradients point in the same directions and reduces updates for dimensions whose gradients change directions. As a result, we gain faster convergence and reduced oscillation

Adam: Adam (Adaptive moment estimation) is an update to RMSProp optimizer in which the running average of both the gradients and their magnitude is used. In practice Adam is currently recommended as the default algorithm to use, and often works slightly better than RMSProp. I have used adam for comparison purposes to other optimizers in my experiments.

Data Augmentation: Data augmentation is a regularization technique where we produce more images from the training data provided with random jitter, crop, rotate, reflect, scaling etc to change the pixels while keeping the labels intact. CNNs generally perform better with more data as it prevents overfitting.

Batch Normalization : Batch Normalization (7) is a a recently developed technique by Ioffe and Szegedy which tries to properly initializing neural networks by explicitly forcing the activations throughout a network to

take on a unit gaussian distribution at the beginning of the training. In practice we put the Batch norm layers right after Dense or convolutional layers. Networks that use Batch Normalization are significantly more robust to bad initialization. Because normalization greatly reduces the ability of a small number of outlying inputs to over-influence the training, it also tends to reduce overfitting. Additionally, batch normalization can be interpreted as doing preprocessing at every layer of the network, but integrated into the network itself.

Benchmark

K-Nearest Neighbors: A k-nearest neighbor model was trained on the color histogram model of the training dataset with Euclidian distance as the distance metric. Different values of k (Nearest Neighbors) were experimented and a graph between accuracy and k was plotted to check how accuracies were changing when changing k. After evaluating this optimum k from some set of selected k, 5 was chosen. The above model with selected k was used to predict against testing data and a score of 0.64540 was obtained on the kaggle platform.
A well designed neural net would be able to beat both naïve benchmarks easily and also k-nearest neighbors considering that even KNN surpasses both the naïve benchmarks easily.

Methodology

Data Preprocessing

As per using VGG16NET like architecture for transfer learning, images are preprocessed as performed in the original VGGNet paper (5). Creators of the original VCGNet subtracted the mean of each channel (R,G,B) first so the data for each channel had a mean of 0. Furthermore, their processing software expected input in (B,G,R) order whereas python by default expects (R,G,B), so the images had to be converted from RGB -> BGR. In the dataset provided the input images were of size 1154 x 866 , so they were resized to 224 x 224 x 3 to reduce size.
Dataset given by Kaggle does not have any validation set, so it was split into a training set and a validation set for evaluation. I have used 80-20 percentage split as using k fold cross validation would result in more time to train and test and would be computationally costly. Out of 2295 images, 1448 images labeled with class 1,i.e. 1448 images contains invasive species and remaining 867 images represents class 0,i.e. invasive species not present
The distributions of the classes are as visualized in Fig.4. Keras ImageDataGenerators were used to generate training data from the directories/numpy arrays in batches and processes them with their labels. Training data was also shuffled during training the model, while validation data was used to get the validation accuracy and validation loss during training.

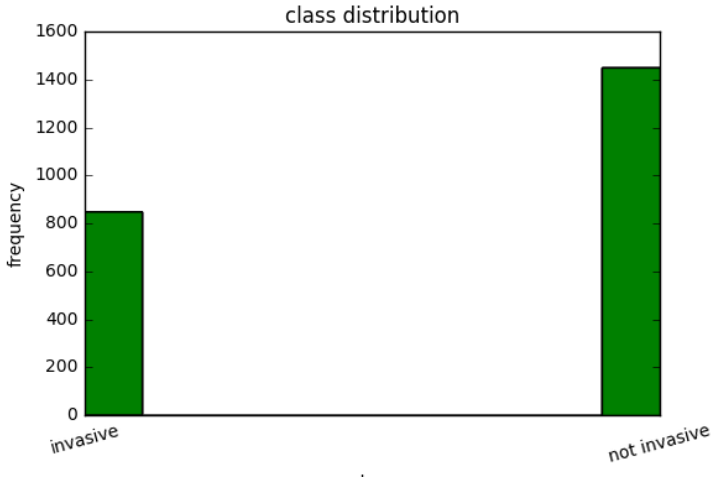


Fig 4.Class distribution diagram

Implementation

Initially the baselines with Knearest neighbors were implemented for comparison. After that the images were split into a training set and a validation set. Preprocessing operations such as subtracting the mean of each of the channels, resizing and centering of the images as mentioned previously was performed and VGG-16 architecture without the last fully connected layers was used to extract the convolutional features from the preprocessed images. The standard VGG16 architecture diagram without the fully connected layer is given in Fig.5.I have trained two different models one fully connected with dropout and batch normalization and other fully connected with data augmentation. The model summary as generated by keras for the models used is as in Table 1 and Table 2.

Refinement:

I applied batch normalization in the model to prevent arbitrary large weights in the intermediate layers as the batch normalization normalizes the intermediate layers thus helping to converge well. Also dropout was applied to the model to prevent overfitting. Even though in the model with batch-normalization and dropout enabled in most of the epochs training accuracy were much higher than validation accuracy. Since data is small and dropout and batch normalization already applied validation accuracies only reached highest peak till 89% while training accuracy close to 95%.After looking this behavior I used data augmentation like random rotations, cropping, flipping, shifting, shearing etc. to generate more data in training batches. In the second model dropout and batch normalization was removed because training accuracy as well as validation accuracies were not coming as expected. Also instead of using adam as optimizer SGD with low learning rate and high momentum was selected in the second model.

Layer (type)	Output Shape	Param #
input_2 (InputLayer)	(None, 224, 224, 3)	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
sequential_4 (Sequential)	(None, 1)	12980225

Total params: 27,694,913
Trainable params: 27,693,377
Non-trainable params: 1,536

Table 1: Keras Model summary for fully connected with dropout and batch normalization

In the specific dataset, initially image centering and resizing was already done in the pre processing state of the experiment so only operations used in data augmentation was rotation, width shift, height shift and horizontal flip.

In the first model I have used two fully connected dense layers of size 512 and 256 with 50 epochs which was very computationally costly and time taking and it took almost 30 hours to train them and to predict it took 2 hours.

In the second model however augmented data was infused already to the network I removed one fully connected dense layer of size 512 and dropout and batch normalization was not used in this model. As a result the same 50 epochs were less time taking and results were also satisfying.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         (None, 224, 224, 3)       0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
sequential_1 (Sequential)    (None, 1)                 6423041
=================================================================
Total params: 21,137,729
Trainable params: 21,137,729
Non-trainable params: 0
_________________________________________________________________
```

Table 2: Keras Model summary for fully connected with data augmentation

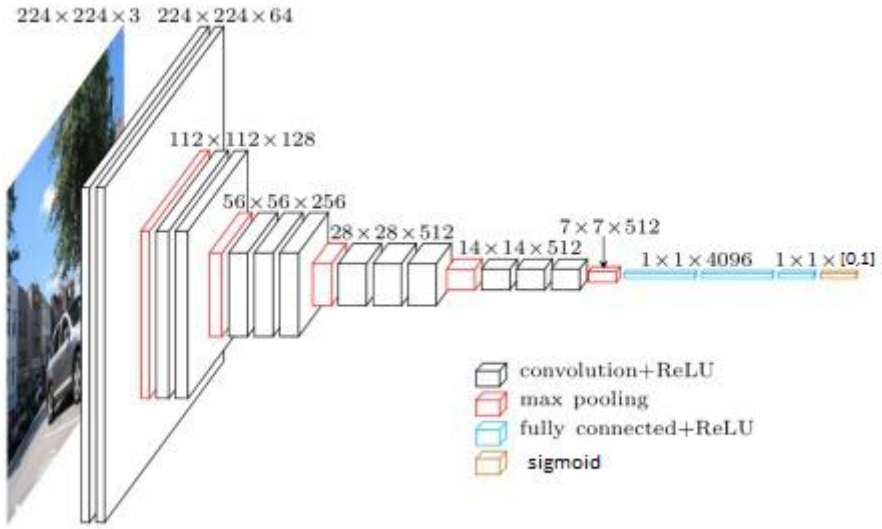


Fig.5 : VCG-16 Architecture

Results

Model Evaluation and Validation

For each experiment only the best model was saved along with their weights (a model only gets saved per epoch if it shows higher validation accuracy than the previous epoch)

To recap, the best model so far uses transfer learning technique along with data augmentation. To use transfer learning I've collected pre trained weights for the VGG-16 architecture, created by Oxford's visual geometry group (hence the name VGG) and used the similar architecture only with replacing the fully connected layers with different dropout and batch normalization. I built and tested the data on two models one with dropout and batch normalization and other only with data augmentation. Both models were trained and compared.

All the images were preprocessed first with image resizing and centering. For the final model I have used VCG16 excluding the final fully connected layer and adding a dense layer with pre trained weights and fine tuning using optimizer SGD with infusing augmented data. This model beats the K-nearest benchmark by 52.61% increase in accuracy over the previous kaggle score submitted using k-nearest benchmark. Table 3 compares the results of benchmark and the trained models.

The model with fully connected layer, dropout and batch normalization resulted in less AUROC value because of the aggressive dropout and normalization resulting in loss of information. However the model with fully connected layer and data augmentation yielded better result because of augmented infused data given the fact that input data to train CNN was not enough. Also as neural net is not overfitting the data because dropout and batch normalization in previous model were leading to information loss.

Model	Area Under ROC Curve
K-Nearest Neighbors(benchmark)	0.64540
VCG-16 with Fully connected layer, Dropout and Batch Normalization	0.96693
VCG-16 with Fully connected layer and Data Augmentation	0.98686

Table 3 : Experimental Results

To validate the model I generated predictions for the validation data which had an AUROC score of 0.9974 and loss of 0.05438. The leader board AUROC score is 0.98686, so the score is quite close. In the validation data out of 459 images, 451 images are classified accurately and only 8 images are incorrect. The confusion matrix (both normalized and non-normalized) plot of the predictions on the validation data is given below.

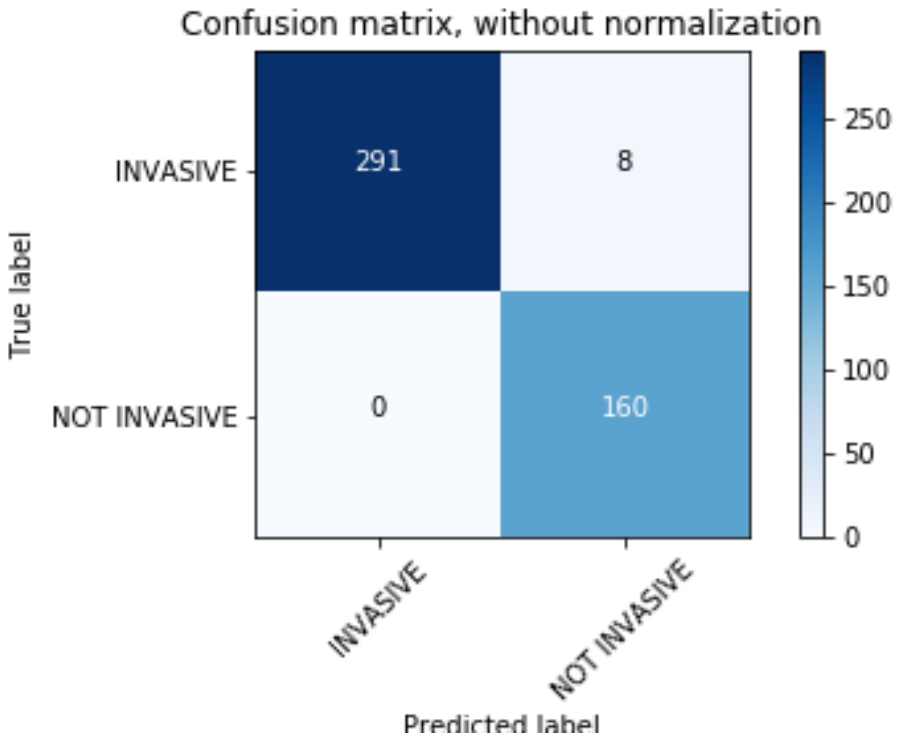


Fig 6 : Confusion matrix(non-normalized) for validation data

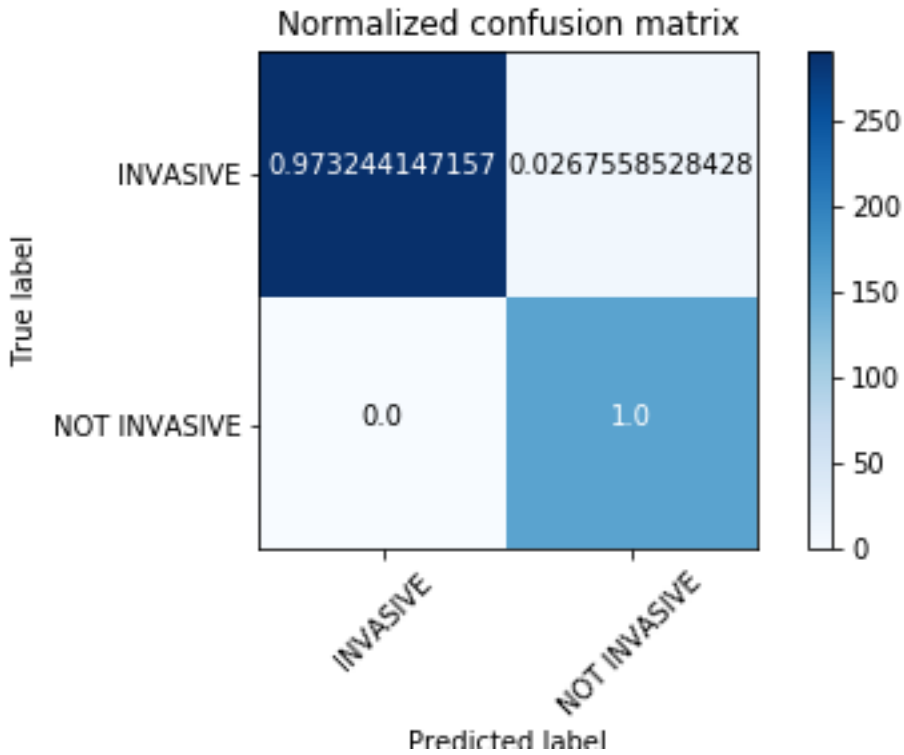
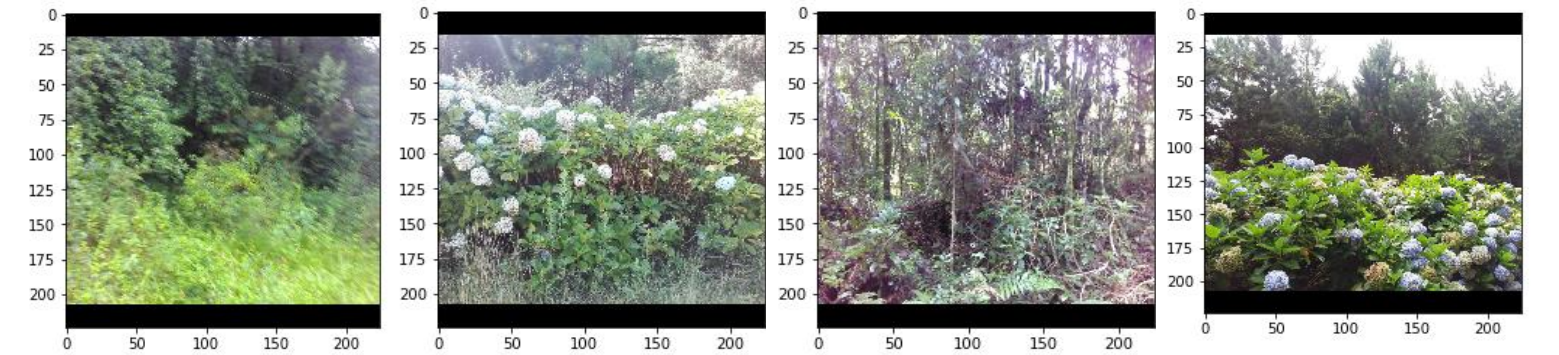


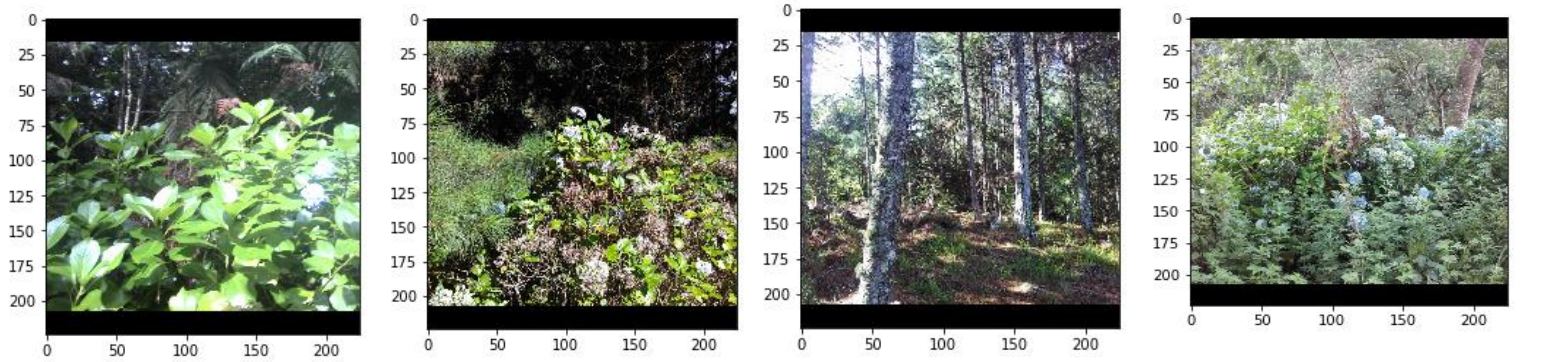
Fig 7 : Confusion matrix(normalized) for validation data

As visible from the confusion matrix the model is really good at predicting non invasive species as it predicts all non invasive species correctly and without predicting any non invasive species as invasive. Out of the 8 incorrect labels in the validation set all of them are invasive species which are incorrectly labeled as non invasive. To figure out any patterns associated to this wrong labeling of some invasive species as non invasive some true labeled images and wrong labeled images were taken at random. Fig 8 and Fig 9 show those correct and incorrect labels respectively.



True Label: Not invasive Predicted Label: Not invasive	True Label: invasive Predicted Label: invasive	True Label: Not invasive Predicted Label: Not invasive	True Label: invasive Predicted Label: invasive

Fig 8: Some correct labels at random



True Label: invasive Predicted Label: Not invasive	True Label: invasive Predicted Label: Not invasive	True Label: invasive Predicted Label: Not invasive	True Label: invasive Predicted Label: Not invasive

Fig 9: Some correct labels at random

By looking at the incorrect labels we can deduce that this may be due to the picture having only one or two flower and that too very small which almost seems hidden. Also as to reduce computational resources and decrease training time the images were centered and their size reduced due to which some flowers got out of the frame to be recognized. Also some images were having flowers that were too far and small to be recognized and only such images having blurred or only one or two small flowers were missed by the algorithm and classified as non invasive.

As data augmentation was used to train this model, it can also handle slight variations in the images such as horizontal flip, different illuminations, rotations and shifting up and down which the scenarios

real life with humans are taking images from their cameras.. In the plot of the accuracy and loss for this model per epoch, it's also seen that the training accuracy/loss is converging with the validation one per epoch (reproduction and further comparison on that in the free-form visualization section). The model was run for around 40in which multiple dense layers dropout and batch normalization was used and it took 2 hours to predict the test images, while our final model took less than 20 hours to train and less than 1 hour to predict the test images. I was running this model on Intel core I7 3.40 GHz CPU with 16 MB RAM, due to which multiple hours of training with 50 epochs were possible and such high accuracies with so much less data was obtained.

Conclusion

Free-Form Visualization

As the accuracy and loss of the models per epoch were recorded, the final model can be compared to the initial model with dense layer, dropout and batch normalization. Fig 10 shows the accuracy/loss graph of the model with batch normalization, dropout but without data augmentation and Fig 11 shows the accuracy/loss graph of the model with data augmentation which is chosen the final model for our experiment.

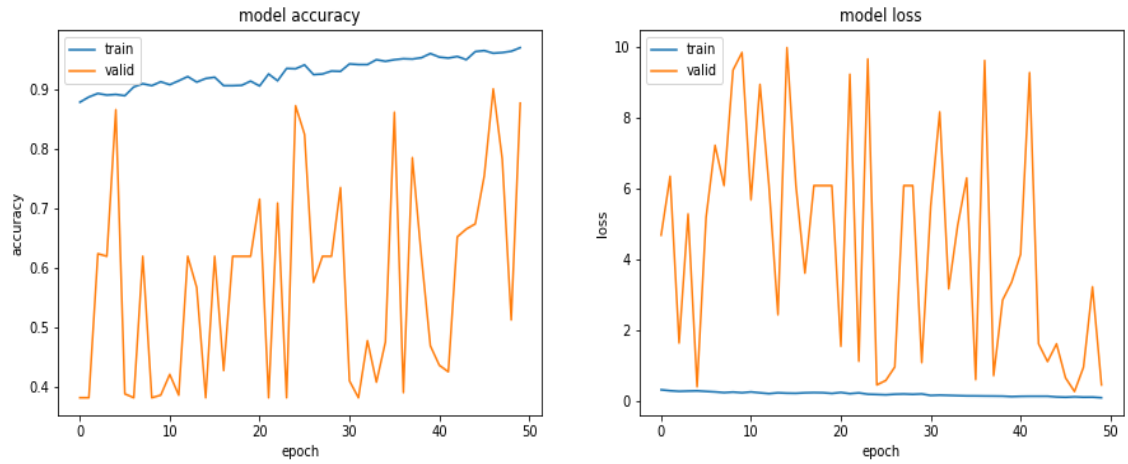


Fig 10 : Accuracy and loss graph for model with dense layers, dropout and batch normalization.

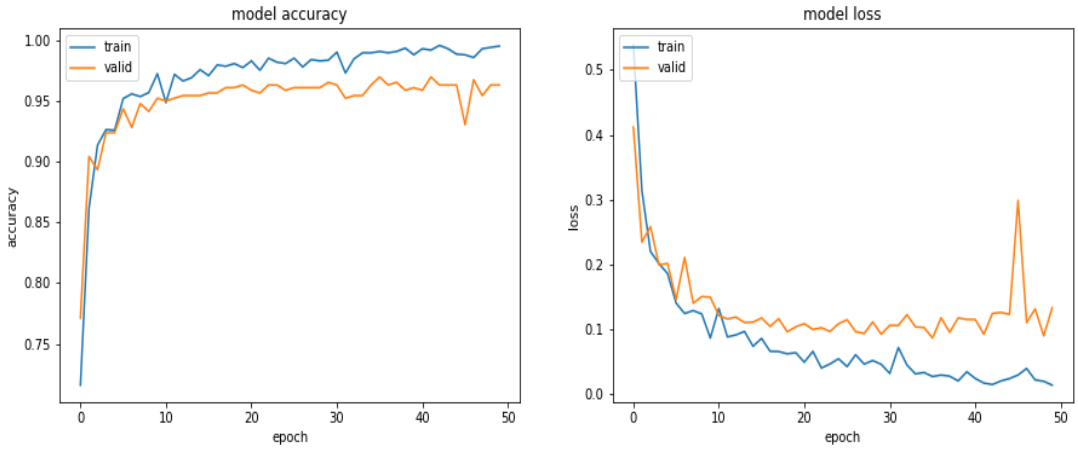


Fig 11 : Accuracy and loss graph for model with dense layers and data augmentation.

As we can see from the above figures that in our initial model data leak was a problem due to aggressive dropout and batch normalization the CNN was unable to learn and thus we can see the model is slowly converging but there is oscillations due to which shows there is information loss. Here we can see that training accuracy is increasing monotonically but there are large oscillations in validation accuracy which leads to our lowest accuracy 40 %

And most of the time the validation accuracy does oscillations in the 40-60 % range. The same is the case with loss in this model which oscillates and increases and decreases intermittently. Clearly this model is unable to learn and over fitting the data

If we see our final model whose graph is depicted in Fig 11 we can see how smoothly the training and validation accuracies are monotonically increasing. While training the train accuracy comes close to 100 % and validation accuracy comes near to 96%.Also loss of training comes close to zero and validation comes close to 0.1%.This model clearly outperforms the initial model and sends me in top 50 entry in leader board with AUROC score 0.98686.

Reflection

For reaching into this end to end solution, I've tried to progressively use more complex models to classify the images. The K-nearest neighbor on color histograms approach as a baseline was used to set a baseline. As training conv nets is resource un friendly task and it often required days of training and prediction, firstly images were centered, resized and reduced so that they can be less heavy over neural nets. Initially Its suggested and is very often that neural nets with less data generally over fit the data and perform bad in actual results, hence initially a model with aggressive dropout and batch normalization was used. Data augmentation really helped a lot to train the net effectively which infused neural net more data for it to learn and predict.

Even if the quality of this dataset is quite high, given it captured images taken by humans which contained many blurred images and unclear images of many invasive flowers. Also some

images capture different flowers of same color and type which the model is not able to detect. It is known the invasive hydrangea has different colors as they bud initially from green to light white to light blue to even dark brown when they are dying. Due to this color changes in the lifecycle of hydrangea predicting invasive hydrangea is difficult. As these pictures are taken by humans many flowers beside bushes not clearly visible because going there for a human is difficult is observed after exploring the dataset. I suggest that for these kind of experiments modern day drones and robots mounted with cameras are really useful to take high quality images in area where there is potential danger for humans to go. While cropping and resizing the image some corner flowers were missed in preprocessing this resulted in incorrect labels to overcome this problem automatic drones that capture these images and zoom in the flower images so that it will be easy for CNN to predict invasive species.

The most difficult task in training nets is essentially the training time, testing time and data adjustment which one has to do while taking neural nets in mind. It was difficult to monitor the training epochs which I trained on my local system, multiple early stopping methods were applied to narrow down the use of only two models and choose right number of epochs which should not be too high so as training doesn't take days to complete and not too low so that the neural net with such less data is about to learn. Many times my local system was crashed while trying with larger size images and large epochs due to which image preprocessing and data augmentation helped me to come up with a model with such a good accuracy.

Improvement

Due to time and computational cost it was not possible for me to run more experiments using different known architectures other than VGG-16 such as RESNET and Inception V-3 for this dataset. It's definitely possible that a different architecture would be more effective. A better approach as flowers are small and hidden in many pictures would be detecting flower and cropping the image according to it. Given enough time and computational power, I'd definitely like to explore the different approaches.

If more options of data augmentation were applied and more data generated the results would have been higher. Also, if we used same sized image or use some clever way to detect flower images and crop according to them the results would be extraordinary.

In this experiment 1154x866 sized image was convert to mere 224x224 sized imaged which can be a huge loss in information keeping data greedy neural nets in mind. If some more dense layers with low dropout and more augmented data with more number of epochs would be used the CNN would definitely be able to learn accurately near to 100 %.

But experimenting with different architectures, using more data and epochs needs more computational resources and time which in turn can increase the accuracy further more.

References

1. *RANDOM FORESTS FOR CLASSIFICATION IN ECOLOGY.* **D. Richard Cutler, Thomas C. Edwards Jr.,Karen H. Beard,Adele Cutler,Kyle T. Hess,Jacob Gibson,Joshua J. Lawler.** 1 November 2007, ECOLOGY | Ecological Society of America.

2. **Jane Elith(School of BioSciences, The University of Melbourne 3010.Australia).** Predicting distributions of invasive species. [Online] 2011. https://arxiv.org/ftp/arxiv/papers/1312/1312.0851.pdf.

3. *Feature Learning Based Approach for Weed Classification Using High Resolution Aerial Images from a Digital Camera Mounted on a UAV.* **Calvin Hung, Zhe Xu and Salah Sukkarieh.** Sydney : remote sensing, 2014, Vols. Remote Sens. 2014, 6, 12037-12054; doi:10.3390/rs61212037. ISSN 2072-4292.

4. *ImageNet Classification with Deep Convolutional.* **Alex Krizhevsky, Ilya Sutskever,Geoffrey E. Hinton.** s.l. : Conference on Neural Information Processing Systems.

5. **Karen Simonyan, Andrew Zisserman.** Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv.* [Online] April 10, 2015. https://arxiv.org/abs/1409.1556.

6. *Dropout: A Simple Way to Prevent Neural Networks from.* **Nitish Srivastava, Geoffrey Hinton,Alex Krizhevsky,Ilya Sutskever,Ruslan Salakhutdinov.** Toranto : Journal of Machine Learning Research, 2014.

7. **Sergey Ioffe, Christian Szegedy.** Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *ArXiv.* [Online] February 11, 2015. https://arxiv.org/abs/1502.03167.