

Machine Learning Engineer Nanodegree

Capstone project

Project Report

Vijaybhasker Pagidoju

[vijay@linkedin](#)

Definition

Project Overview

This project is intended to resolve a problem of identifying invasive species. The Solution is drawn by using Machine learning Algorithms, Deep Neural Networks and Transfer Learning.

The Intention of the project is to help researchers and environment by identifying invasive species, harmful to the habitat and eco system.

The Solution will save researchers valuable time, find the invasive species within very less time.

There is a good amount research heading towards solving problems related to saving the environment and plants. A few of the problems can be found at the below links

<https://www.kaggle.com/c/plant-seedlings-classification>

<https://www.kaggle.com/c/species-presence-prediction>

<https://www.kaggle.com/c/species-presence-prediction-2015-2016>

Problem Statement

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.

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.

Because scientists cannot sample a large quantity of areas, some machine learning algorithms are used in order to predict the presence or absence of invasive species in areas that have not been sampled. The accuracy of this approach is far from optimal, but still contributes to approaches to solving ecological problems.

This Problem is part of Kaggle's competition.



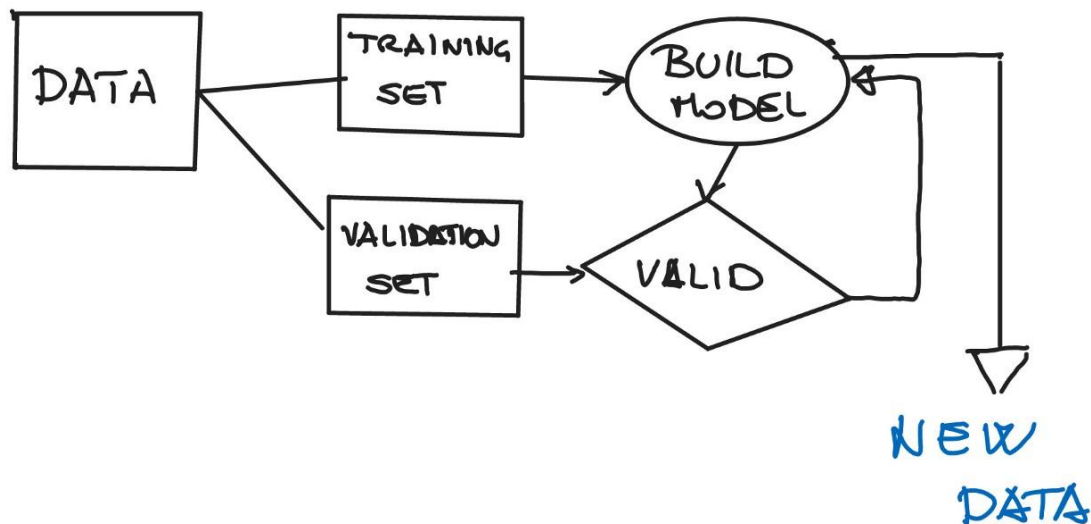
Our solution is to identify invasive species in the images like the above.

Analysis:

From the Problem statement, it is clear that the problem is a image classification based one and there is a training and testing sets of data provided with the label for the training set.

Being a labels training set and the solution's motive is to attain most accurate labels for the testing data set.

This is a Supervised Learning Problem



So, the intended steps are as below

Importing Datasets

Preprocessing the Data

Creating a Cross Validation Set

Building a Model

Testing the Model built

Optimizing the Model

Predicting the Labels of the Test Set

Datasets

The Data is provided on kaggle as part of the competition

The Datasets provided are

Train Dataset: A training dataset consisting of 2295 images of plants and the environment.

Test Dataset: A testing dataset consisting of 1531 images of the plants and the environment.

Training Labels: Labels for the training dataset of 2295 images are available in train_labels.csv file.

Sample submission file: A sample submission file in csv format is provided with 1531 rows with headers, as a sample format for the final submission labels csv file.

Sample Images from out Train Set





Sample Images from our Test Set





The Datasets in Train and Test set are not considered of the same resolution, so we need to preprocess the data to make sure all are of the same resolution. For this we need to have a specific resolution of images to decide to use the same resolution across the model training and testing.

We are preprocessing all the images to the same resolution of 224,224, to use in our model to train and predict so that all the images are of the same size and there is a right measure of accuracy. As we can compare apples to apples.

Metrics

The submission labels will be evaluated based on area under ROC curve between the predicted probability and observed target.

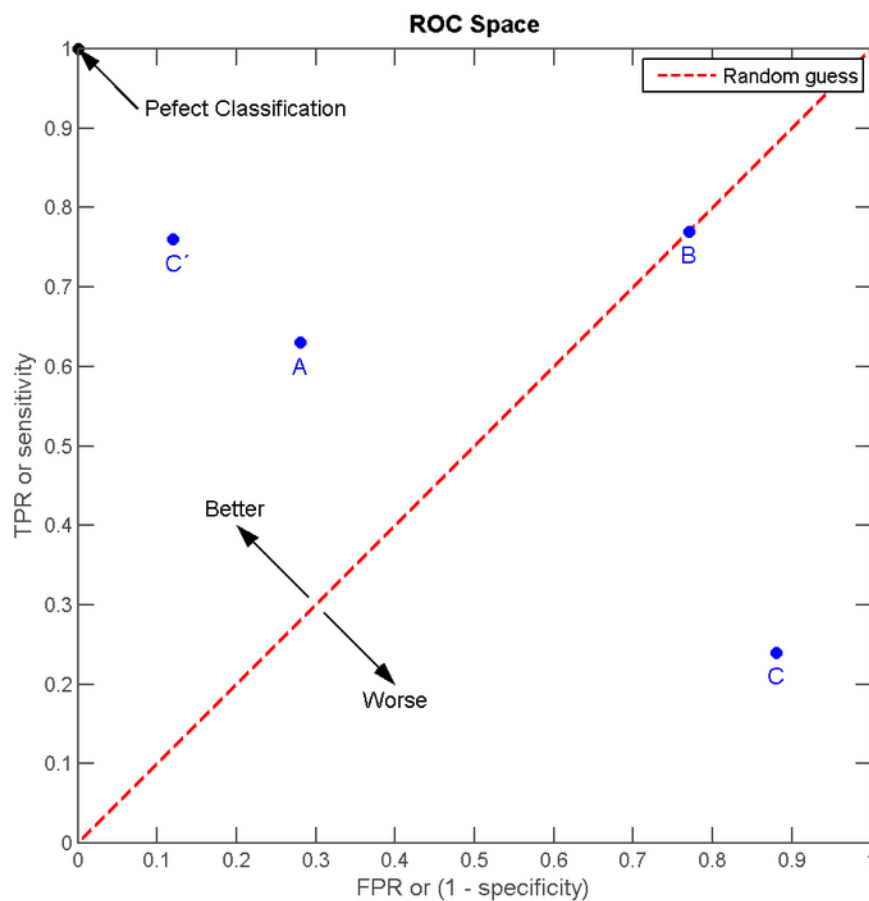
The Project will evaluate the train and cross validation sets in the metrics of accuracy.

The project will make use of various loss functions such as rmsprop to identify the loss and evaluate to mitigate.

Accuracy and Loss will be evaluated and observed further on graph visualizations to understand the progress.

The competition evaluates the model using Area under ROC curve. A ROC is curve is used to visualize the performance of a binary classifier. As we are identifying an invasive species among all the species, evaluating the performance under the ROC curve is a right measure.

An area under ROC curve is evaluated by observing the found performance among the area formed by true positive rate (TPR) and false positive rate (FPR). A ROC space is defined by FPR and TPR as x and y axes.



Accuracy is one right measure to find the performance of our model post training and tuning, which can be easily evaluated using python libraries.

Accuracy found will be evaluated by observing its performance on the test set for Area under ROC.

Accuracy can be found using the average performance of the model across all the iterations indetermining right prediction. Mathematical formula for the accuracy is as below.

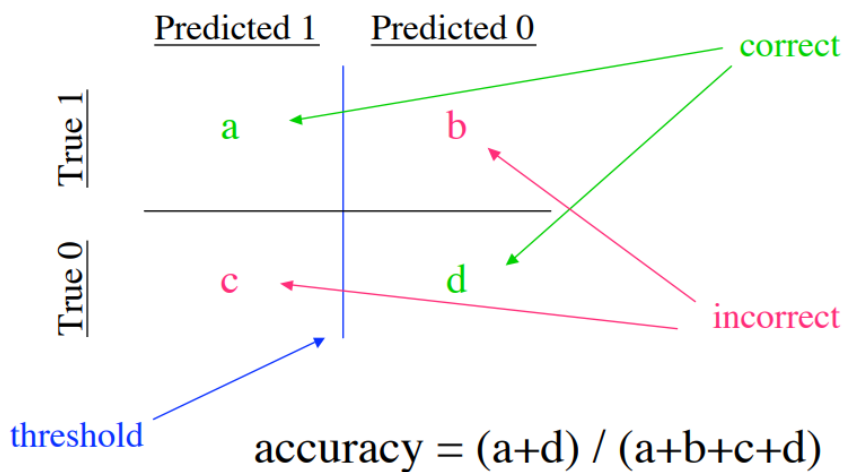
$$accuracy = \frac{\sum_{i=1 \dots N} (1 - (target_i - threshold(f(\vec{x}_i))))^2}{N}$$

Target: 0/1, -1/+1, True/False, ...

Prediction = $f(\text{inputs}) = f(x)$: 0/1 or Real

Threshold: $f(x) > \text{thresh} \Rightarrow 1$, else $\Rightarrow 0$

threshold($f(x)$): 0/1



4

In layman terms

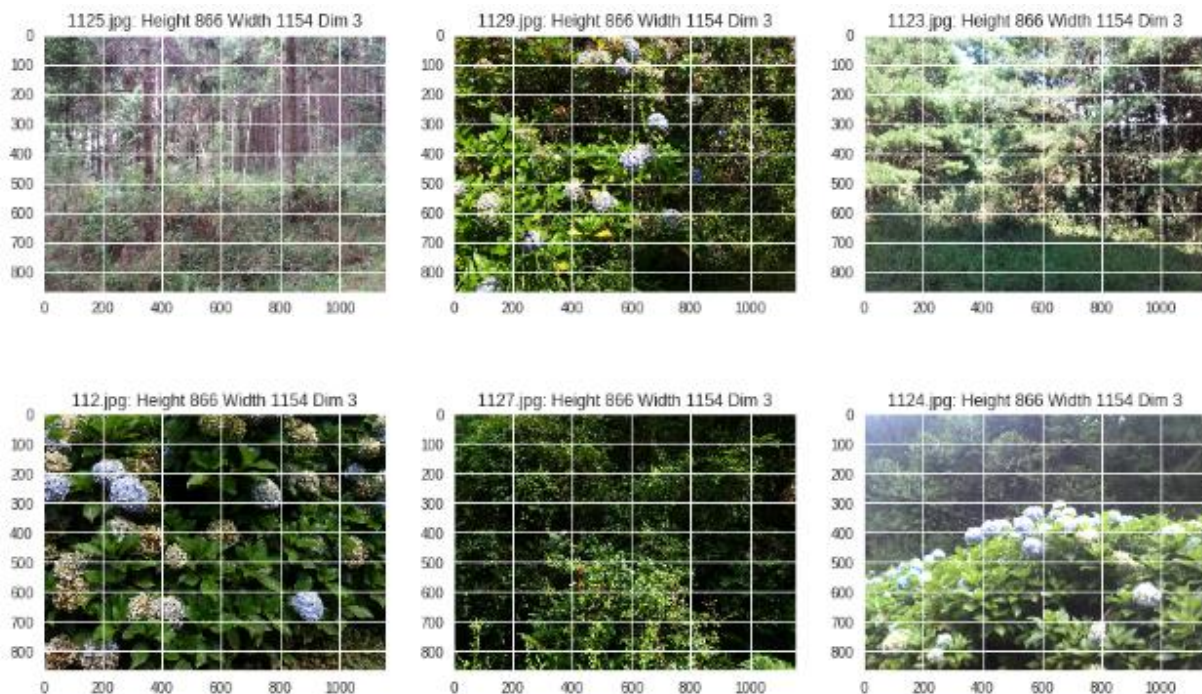
Accuracy is calculated by measuring right predictions to total predictions as shown above.

Exploratory Visualization

To get an insight of the data , lets visualize the images deeper as segments of boxes with dimensions.

The Visualization of the data represents various dimensions of the data. The Image data should all be consistent enough to have equal light as the brightness of a few images will skew the accuracy in the training as it is not even across all the dataset.

We need to make sure the data is consistent enough and pixels are identifiable to create a pattern to recognize the species at a good accuracy.



Algorithms and Techniques

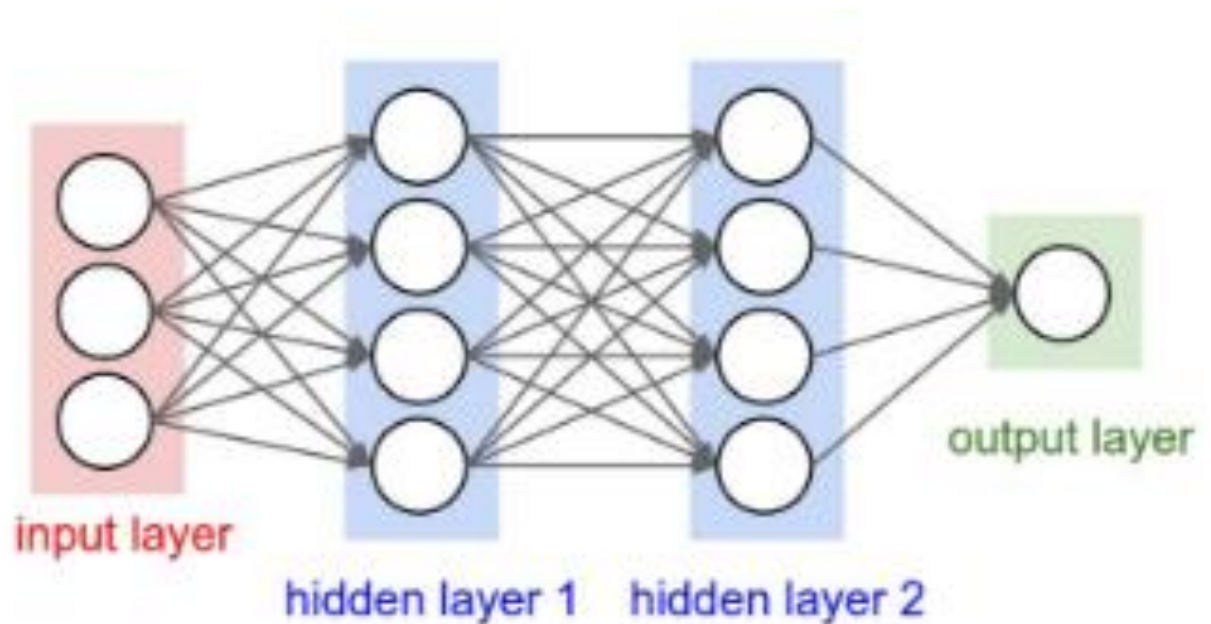
.For the present Model architecture, we will be making use of Convolution Neural Networks.

CNN's are most prominently used for the image classification. Convolutional Neural Networks take advantage of the inputs, that consist of images and they constrain the architecture in a more sensible way. In particular, unlike a regular Neural Network, the layers of a ConvNet have neurons arranged in 3 dimensions: width, height, depth. The dimension of depth usually refers to the feature layers.

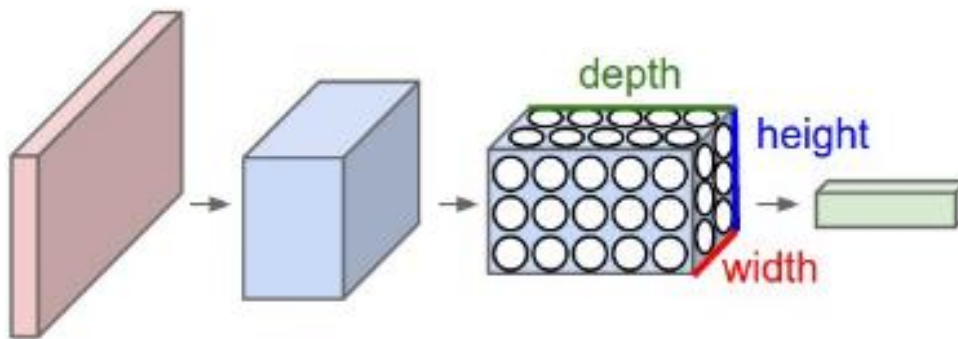
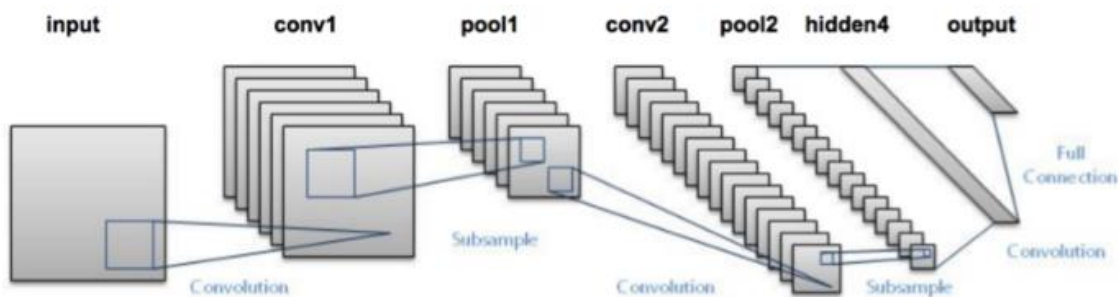
CNN's are preferred for its feature data extraction by using various hyperparameters such as filters and a more clear view of the features be used for training. CNN's also have a feature of pooling. Pooling is one of the best feature of CNN, which not only helps in dimensionality reduction, which also helps in reducing overfitting problem.

CNN's provides two types of pooling, namely MaxPooling and GlobalAveragePooling. MaxPooling feature reduces the dimensionality of the features based on pool size provided, default to 2. Usually a Maxpooling layers is used in the hidden layers to mitigate the problem of overfitting. GlobalAveragePooling is mostly used just before the output layer as it reduces the dimensionality significantly to 1. CNN also provides many hyper parameters like kernel_size, padding, activation are prominent among all.

Example of Normal Neural Network



Examples of Convolution Neural Network



A Convolution considers all the dimensions of an image, for the feature extractions. As pixels in various dimensions are important to understand an image better for the training, CNN are one of the best models in the industry for the image classification.

There are many award winning neural networks on ImageNet dataset using Convolutional Neural Networks such as ResNet50, VGG16, VGG19, InceptionV3 etc

For our solution to build a model, we will make use of Convolution neural network layers. As mentioned above, we will make use of Pretrained networks of CNN from Keras to optimize our model performace, when necessary.

CNN's and Pretrained CNN's are readily available in Keras libraries.

As example of CNN MaxPooling layer transformation with pool size of 1 can found as below

1	1 _{x1}	1 _{x0}	0 _{x1}	0
0	1 _{x0}	1 _{x1}	1 _{x0}	0
0	0 _{x1}	1 _{x0}	1 _{x1}	1
0	0	1	1	0
0	1	1	0	0

4	3	

Solution

The Project intends to use Neural networks for the building the model with various layers of nodes. The model will make use of various hyper parameters to tune the model.

A continuous evaluation of validation loss and accuracy during the training.

There will be further evaluation of the training and validation accuracies and then based on them, the project will import the pretrained networks based on ImageNet dataset(Resnet50,vgg16,vgg19,inceptionv3,etc).

The project intends to create a hybrid model using the pretrained network and fully connected dense layers after the pretrained network.

Finally, the the labels for the testing dataset will be identified.

Benchmark Model

The project being a part of the kaggle's competition, the benchmark model at this time of this project submission is at a score 0.9970, which is highly unlikely to be achieved. The Project intends to understand the problem and route to the solution with clear steps and leading to a optimal validation accuracy and test accuracy to be in top 50% submissions for the 1st ever kaggle competition submission.

Methodology

Importing Libraries

Initially there are a stack of imports that are required for the project to set go..

The project imports following

Pandas, NumPy, Matplotlib, OpenCV, Keras, Sklearn, tqdm, Pretrained networks from keras,PIL, ImageDatagenerator, Keras Datasets,Conv2D.

Data Preprocessing

The training labels should first be evaluated to find any discrepancy and should delete if any NA values are available and the same with sample submission csv file.The training and testing datasets should be evaluated to find the number of images provided in them.

After the initial evaluation, the images are set as a numpy array and then further to this the images are centered and brought to a common pixels so that there is a consistency across all the images to have a consistent training of the datasets.

Then, all the images in the datasets should be normalized by dividing by 255 to get rid of any inconsistency and effects.

```
def centering_image(img):
    size = [256,256]

    img_size = img.shape[:2]

    # centering
    row = (size[1] - img_size[0]) // 2
    col = (size[0] - img_size[1]) // 2
    resized = np.zeros(list(size) + [img.shape[2]], dtype=np.uint8)
    resized[row:(row + img.shape[0]), col:(col + img.shape[1])] = img

    return resized
```

```
x = []
for i, file_path in enumerate(file_paths):
    #read image
    img = cv2.imread(file_path)
    grey = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

    #resize
    if(img.shape[0] > img.shape[1]):
        tile_size = (int(img.shape[1]*256/img.shape[0]),256)
    else:
        tile_size = (256, int(img.shape[0]*256/img.shape[1]))

    #centering
    img = centering_image(cv2.resize(img, dsize=tile_size))

    #out put 224*224px
    img = img[16:240, 16:240]
    x.append(img)

x = np.array(x)
```

There should be cross validation set that should be formed, to test the trained model, which should not be a part of Training and testing set.

The Training set is shuffled to make the dataset random, to make the input more generic and less prone to Overfitting. Then, the Training set is divided to Training set and Cross Validation set. The Data is usually split as per the standard 80:20 dataset split.


```
val_split_num = int(round(0.2*len(y)))
x_train = x[val_split_num:]
y_train = y[val_split_num:]
x_test = x[:val_split_num]
y_test = y[:val_split_num]

print('x_train', x_train.shape)
print('y_train', y_train.shape)
print('x_test', x_test.shape)
print('y_test', y_test.shape)

x_train (1836, 224, 224, 3)
y_train (1836,)
x_test (459, 224, 224, 3)
y_test (459,)
```

Now, the images from the dataset should randomly be witnessed to see if all the images are available.



Free-Form Visualization

Lets visualize the above images in various forms Original, RGB, Lab and Gray

Image- 1

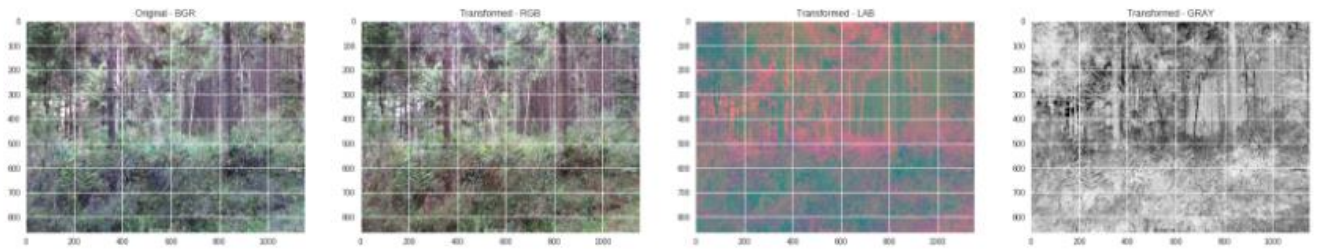


Image- 2

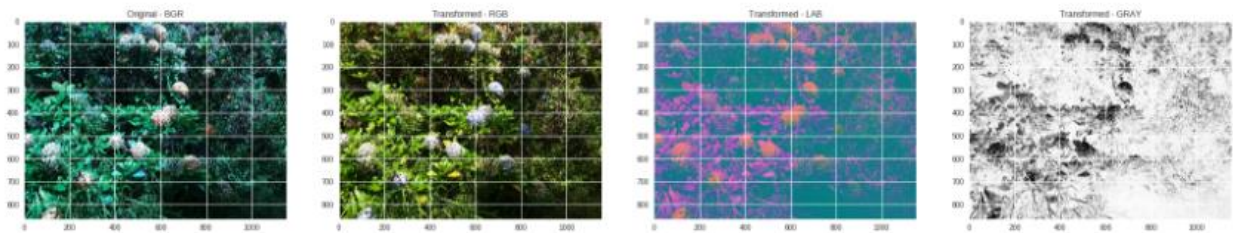


Image- 3

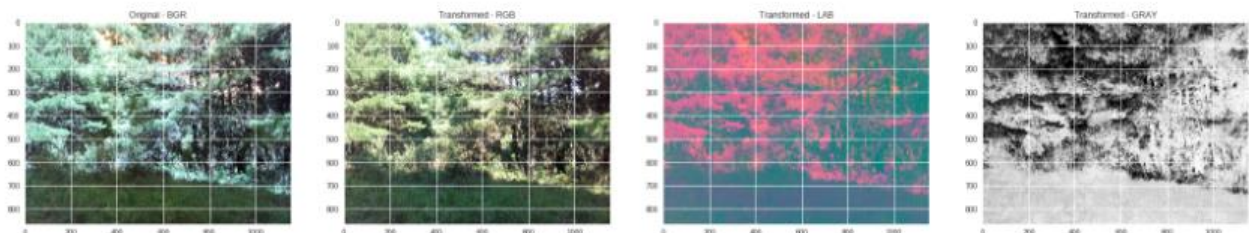


Image- 4

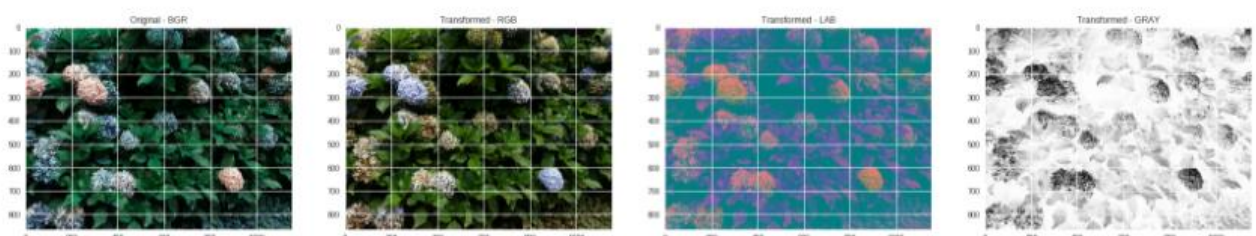


Image- 5

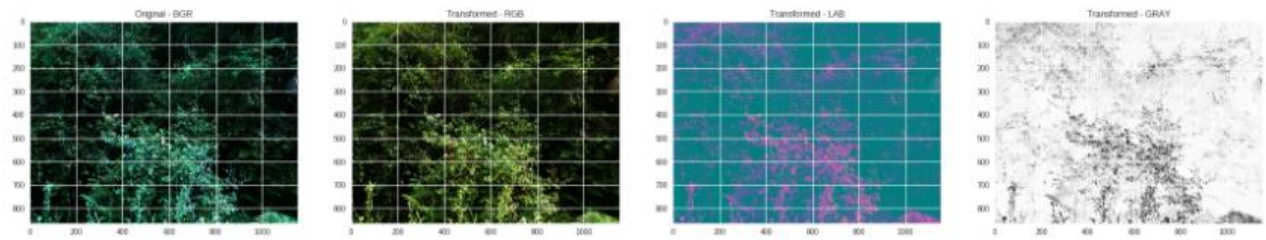
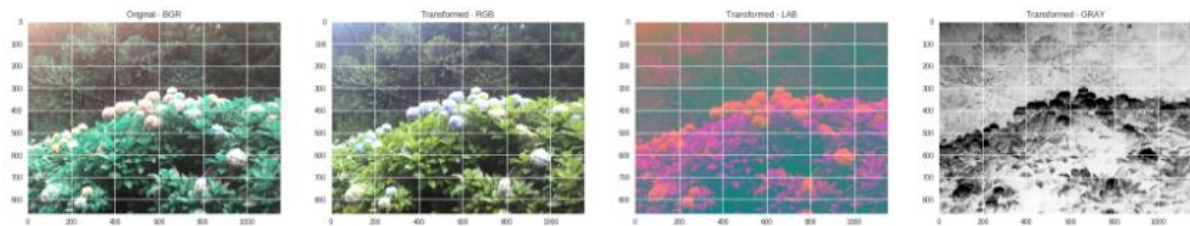


Image- 6



Building a Model

The Model architecture is built from scratch using Convolutional neural network layers.

The Model is built using

One Input layer

Three Hidden layers

One output layer

The Batch normalization input layer is produced with the input as required.

The three hidden layers use varied number of filters, even padding, same kernel size.

After every hidden layer, there is a Maxpooling with pool_size set to 2, to reduce the high feature dimensionality from the filter sizes, which will help the model to mitigate overfitting.

One more measure taken is to add dropout layer at a random permutation, so that there is a random drop of input to make the training more generic and this also helps the model to be away from problem of overfitting.

Before adding the final Dense layer, A globalAveragepooling layer is added to make the feature dimensionality reduced to 1, to input it to final layer.

The Model is using relu as an activation functions for all the hidden layers and sigmoid as an activation function for the final layer.

This entire model built results in a total parameters to be trained as 101,965.

The Built Model is further compiled using Adam as an optimizer, with loss as binary cross entropy and accuracy as metrics.

The Model is trained for 60 epochs with a batch size of 30. The weights that result in incremental accuracy and decrementing the loss are stored in a file to further load them to the model after the training to improve accuracy of the model.

```
model = Sequential()
print('Training set is ',x_train.shape[0])
print('Validation set is ',x_test.shape[1])

model.add(BatchNormalization(input_shape=(224, 224, 3)))

model.add(Conv2D(filters = 256,kernel_size=2,padding='same',activation = 'relu'))
model.add(MaxPooling2D(pool_size=2))

model.add(Dropout(0.3))
model.add(Conv2D(filters = 64,kernel_size=2,padding='same',activation = 'relu'))
model.add(MaxPooling2D(pool_size=2))

model.add(Conv2D(filters = 128,kernel_size=2,padding='same',activation = 'relu'))
model.add(MaxPooling2D(pool_size=2))

model.add(GlobalAveragePooling2D())
model.add(Dense(1,activation = 'sigmoid'))

model.summary()
```

```
model.compile(optimizer='Adam', loss='binary_crossentropy', metrics=['accuracy'])
```

```
from keras.callbacks import ModelCheckpoint
```

```
epochs = 60
```

```
checkpointer = ModelCheckpoint(filepath='weights.best.from_scratch.hdf5',  
                               verbose=1, save_best_only=True)
```

```
model_trained = model.fit(x_train, y_train,  
                          validation_data=(x_test, y_test),  
                          epochs=epochs, batch_size=30, callbacks=[checkpointer], verbose=1)
```

Optimizing the Model building using Transfer Learning

After observing the built model validation accuracy and validation loss , in order to optimize the models accuracy, A hybrid model using transfer learning is built.

A Pretrained Neural Network on ImageNet Dataset is loaded and then the pretrained network is loaded to the new model and then a fully connected layers are added to make the model specific to images in the dataset.

In the present project, VGG16 neural network is used from the keras applications.

Once the vgg16 pretrained network is loaded with the same input shape of 224,224. There are three hidden dense layers and one output layers added to the model.

The three hidden layers have varied filter sizes and use same activation function relu. The final layer is reduced to 1 filter size, using activation function sigmoid.

As the hybrid model is built, its compiled using same loss function and metrics as accuracy. The Models uses the same optimizer with different tailored parameter to decreased the learning rate.

This results in total trainable parameters to 21,178,689.

The Model uses imagedatagenerator to rotate the images at different angles to further get a good training accuracy for the images.

The Model is then trained for 20 epochs with a batch size of 32 and the best weights are stores as earlier to load the model for better working.

```
add_model = Sequential()
add_model.add(Flatten(input_shape=base_model.output_shape[1:]))

add_model.add(Dense(256, activation='relu'))

add_model.add(Dense(128, activation='relu'))

add_model.add(Dense(64, activation='relu'))

add_model.add(Dense(1, activation='sigmoid'))

vgg16_model = Model(inputs=base_model.input, outputs=add_model(base_model.output))
vgg16_model.compile(loss='binary_crossentropy', optimizer=Adam(lr=1E-4, beta_1=0.9, beta_2=0.999, epsilon=
1e-08, decay=1E-4),
                    metrics=['accuracy'])
vgg16_model.summary()
```

```
batch_size = 32
epochs = 20

vgg16_train_datagen = ImageDataGenerator(
    rotation_range=31,
    width_shift_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True)
vgg16_train_datagen.fit(x_train)

vgg16_checkpointer = ModelCheckpoint(filepath='weights.best.vgg16.hdf5',
                                     verbose=1, save_best_only=True)

vgg16_history = vgg16_model.fit_generator(
    vgg16_train_datagen.flow(x_train, y_train, batch_size=batch_size),
    steps_per_epoch=x_train.shape[0] // batch_size,
    epochs=epochs, callbacks=[vgg16_checkpointer],
    validation_data=(x_test, y_test),
)
```

Refinement

After the model is built, the very first step taken is to manipulate the optimizer from **rmsprop** to **Adam**, as Adam being a high performer peformed well and tuned the model and increased the Val accuracy from 0.8230 to 0.9455 with an increase in epochs from 55 to 60.

After observing that the hybrid model built with Transfer learning, has stopped growing after 8 iterations, the optimizer used was Adam, which is further tuned to learn at an exponential rate, epochs are changed from 32 to 27 to 20 from continuous evaluation of the graph results after every training, batch size is

changed from 50 to 32, These has improved the validation accuracy from 96.12 to 99.13 to 94.12.

When observed that the for the first three trials one with Built from scratch and two with hybrid models validation accuracy is optimal but the test accuracy is not growing up, further measures of hyper parameter tuning is used. In this case, as discussed above the third iteration has used optimizer Adam with exponential learning rate and reduced the epochs to 20.

This has increased our Test accuracy from our last three submissions to the highest

The yield of the evaluation has resulted in the following output

The best and optimal combination of hyper parameters is for epochs: 20, batchsize: 32, optimizer: Adam(lr=1E-4, beta_2=0.999, epsilon=1e-08, decay=1E-4).

Model Evaluation and Validation

CNN Model built from scracth:

The Model has performed well both on training and validation set. Ther were no skewed responses when we observe the history graph of the training validation performance.

The Accuracy and Loss graph of both training and validation sets go along most of the time, which represents a very good training and validation performance.

The Graphs performance clearly shows that the dropout layers have helped to mitigate overfitting to a good extent.

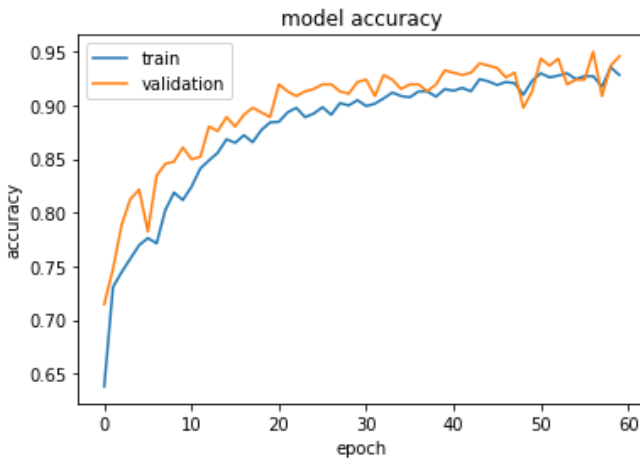
Training Accuracy : 0.9281

Validation Accuracy: 0.9455

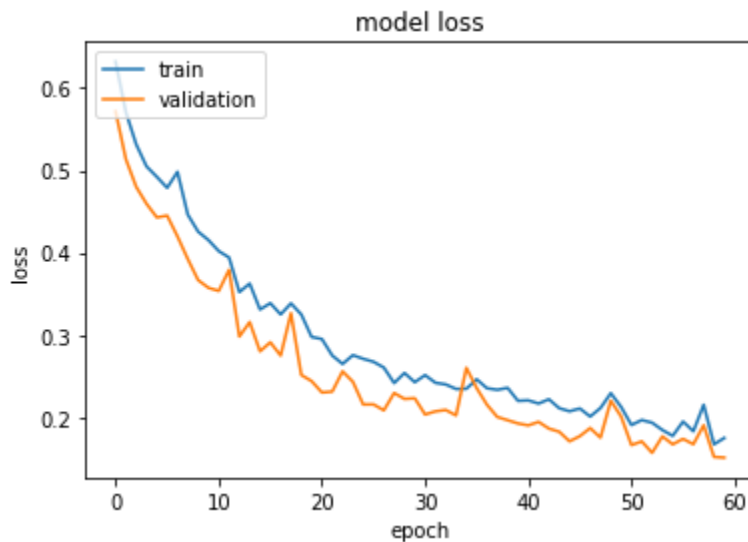
Training Loss : 0.18

Validation Loss: 0.15

Training and Validation Accuracy Graph



Training and Testing Loss Graph



Hybird Model Using Transfer Learning(pretrained VGG16 network)

The Hybrid Model has performed well with training accuracy but it has lot of skewed effect during the validation accuracy and finally set to converge with the training accuracy which is good.

There is no overservation of overfitting except with the mid diverged performance of validation set, then it again converged to training performance path.

Training and Validation Loss go along except the validation has volatile response and then converges along with the training loss to decrease.

Training Accuracy: 0.9989

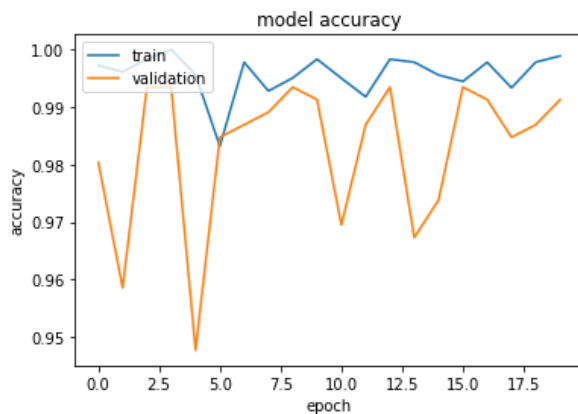
Training Loss: 0.00

Validation Accuracy: 99.13

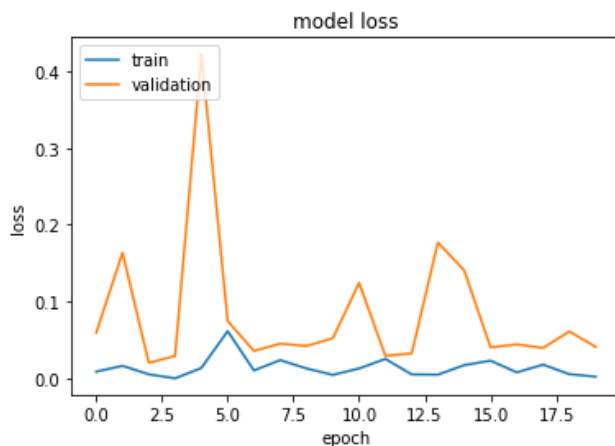
Validation Loss: 0.04

Training and Validation accuracy have increment by a good amount from the previous model built from scratch.

Training and Validation Accuracy Graph



Training and Validation Loss Graph



After observing the above skewed performance of training and validation set, the model is tuned to use epochs: 20, batchsize: 32, optimizer: Adam(lr=1E-4, beta_2=0.999, epsilon=1e-08, decay=1E-4).

This yielded us a converged performance across both Training and Validation set although validation score went down a little but there is no observation of overfitting.

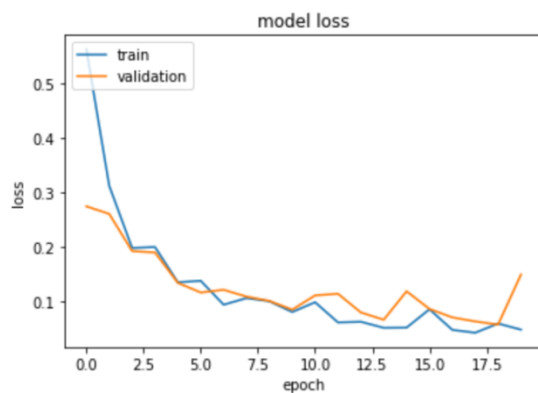
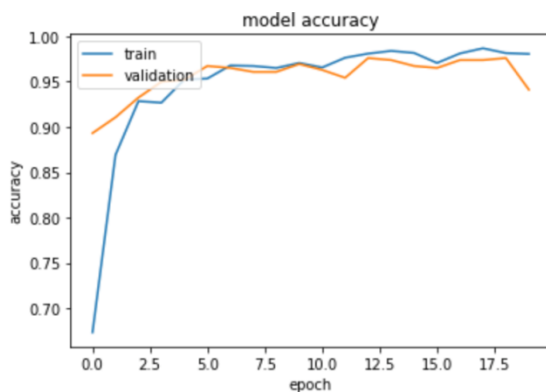
This yielded

Training Accuracy: 98.06

Training Loss: 0.05

Validation Accuracy: 94.12

Validation Loss: 0.15



Test Prediction

As, the Problem is part of the Kaggle competition,

Using our trained hybrid model, Prediction is made for all the images in the test dataset and the labels are stored in the submit.csv file and uploaded to the kaggle submissions.

Our Final Model will be set to predict the labels of the testing set of images and the performance observed will be evaluated on Area under ROC curve by the competition evaluation team.

Built from scratch CNN Model has scored 0.5000 on Area under the curve with cross validation accuracy of 0.9455.

Hybrid CNN model has score 0.5000 on Area under the curve with cross validation accuracy of 0.98.83, which quite improved from build from scratch but the test score remained at 0.5000.

There is one more attempt made on the hybrid model changing optimizer to Adam, then the validation accuracy increased to 99.13 but the testing accuracy remained at 0.5000.






We have then tuned the model to further optimizing it by setting the parameters as epochs: 20, batchsize: 32, optimizer: Adam(lr=1E-4, beta_2=0.999, epsilon=1e-08, decay=1E-4), which yielded a 0.98997 score on testing although the validation score went down to 94.12%.

The testing score meant to reserve a position in top 19% submissions and amongst top 100 submissions on leaderboard, which is really a good score.

Kaggle submission score

4 submissions for Vijay		Sort by	Most recent
All Successful Selected			
Submission and Description		Public Score	Use for Final Score
vgg16submit.csv an hour ago by Vijay Vgg16		0.98997	<input type="checkbox"/>
submit_Inceptionv3.csv 21 hours ago by Vijay Inception V3 Version		0.50000	<input type="checkbox"/>
submit.csv a day ago by Vijay Submission		0.50000	<input type="checkbox"/>
submit.csv a day ago by Vijay Ipython Notebook Used transfer Learning		0.50000	<input type="checkbox"/>

Kaggle leaderboard position based on the score, as the submission deadline has ended.

96	▼ 19	OnlyMyRailgun		0.98988	18	10mo
97	▼ 18	Naonao		0.98980	23	1y
98	▼ 18	NAOC_wqx		0.98980	10	1y
99	▼ 18	RakeshNikam		0.98979	13	1y
100	▼ 18	Shaana		0.98979	21	10mo

Justification

The Benchmark model set is the top on the leaderboard with high test performance of 0.9970, which is optimal.

The attempt was made to follow the finest approach by understanding the data preprocessing and visualizing the graph and analysing them. The model has pursued fine measures for the analysing the data preprocessing, normalizing the data to feed.

The Algorithm attempted a built from scratch Model and analyzed the training and validation performances and observed a continuous performance growth and decrease in loss in both training and validation. We have further attempted to visualize the graph to understand the history of training and validation done, to

understand the model more accurately and further a hybrid model is built using vgg16 and then observed the performance enhancement in both training and test accuracy.

The Overall attempt is to observe some good measures followed by the benchmark model and develop a model with right architecture and a clear understanding of all the phases of machine learning model development, which was clearly achieved. The competition with the score is not achieved and which was not the projects goal but the project has clearly documented all the phases of model designing, followed best practices to visualize the data and used Transfer learning, GirdsearchCV to optimize the tuning, which makes the attempt of the project implementation a good model.

Our achieved score of 0.98997 stands among top 19% and top 100 on public leaderboard. This is not the best performing algorithm but the reasearchers can definitely make use of this good score which will save their time and plants can be saved.

The attempt of the project to help the researchers and the environment has reasonably served the purpose.

This Project implementation will definitely help in the future project developments and gives a way to compete in the active competitions on Kaggle to attempt further best scores.

End to End Solution

As discussed earlier, the problem of finding invasive species to safeguard our environment is vital. Identifying these invasive species requires domain knowledge, so researchers spend whole lot of time moving around the locations to identify the invasive species and eradicating them amongst others. This is cost and time consuming and leads to less better usage of researchers time.

So we have comeup with a model which can identify the invasive species amongst other with a high accuracy of 0.98997, which is a reliable score. By using the model, there is no requirement of researchers to move around the locations

rather train employees to eradicate them based on the locations the model has identified that there is an invasive species with optimal probability.

Further to this, the model can be deployed on a software and then be deployed on a Quadcopter, which can move around the location and take arial images which after every full loop, can identify the invasive species location and then the trained employees can remove them.

This will save an immense amount of time to save our environment and help researchers to do more productive work, save the cost required and finally our Plants and ecosystem are saved.

Further Ideas

The Idea of improvement to the model relies on the images, which are taken a sometime back, rather if a arial image capturing is used as a input, that will further give us an live and reliable data that can be used to identify the species.

The dataset is good enough for competition, but further more datafeed can help to make the system more accurate on the test as the identification is vital.

Would like to use gridsearchcv on the data using sklearn and keras library, so that an optimal hyperparameters can be found very early.

Would like to use early stopping technique for long epochs which saves a lot of time and cost too.

Challenges

Once the Model is built, there was a point where there is not change of cross validation accuracy, which struck me for a couple of days. After some reasearch and trials, got to see volatility once after the optimizer has changed, Optimizers play a major role in the training, a right optimizer for a model has a lot of prominence, which can totally struck the the model training with no change to

validation accuracy even with continuous increment of training accuracy, in turn which will lead to overfitting rather increment in test accuracy.

When training a model, same number of epochs, batches, optimizers resulted in different results, initially I didn't get it, then after got to see shuffling the data is the trick, and random dropout layer also has its place.

Overfitting is one other major problem that struck the progress when building the all new model, the resolution is an implementation of dropout layers, maxpooling layers.

One big challenge for any machine learning model development is the processor, this problem may look common and easily solvable but during this project development, moving to powerful processor delayed the project progress by 10 days, as there was a phase when Udacity had aws configuration, during its configuration to my system, they all of a sudden have taken that off, without intimation, I was just emailing them waiting for a response as there was no easy way online to setup a gpu configured cloud platform from local. Finally, I have never heard back from except a response 2 days back but I was close to my deadline, but this has made me research and learn configure three main platforms AWS, Google Cloud Platform and Floyd. So , the problem has turned out to be a learning which will take me longway in configuring the cloud platforms.