Agriculture Field Pattern Classification Using Deep Learning

Sarthak Jain, Shrey Dangayach, Jimit Panditputra, and Pasupunuri Bharath Kumar

Guide Name
Dr. Sanjeev Sharma
Assistant Professor, Department of Computer Science
Indian Institute of Information Technology Pune, India

Abstract

The success of deep learning in visual recognition tasks has driven advancements in multiple fields of research. Particularly, increasing attention has been drawn towards its application in agriculture. Nevertheless, while visual pattern recognition on farmlands carries enormous economic values, little progress has been made to merge computer vision and crop sciences due to the lack of suitable agricultural image datasets. Meanwhile, problems in agriculture also pose new challenges in computer vision. For example, classification of aerial farmland images requires inference over extremely large-size images with extreme annotation sparsity. This makes it more difficult from other problems.We try to solve this problem by developing an Image Classification model using the techniques of Deep Learning. We experiment by training 5 best-in-class pre-trained models for image classification (VGG16, ResNet50, DetNet, InceptionV3, EfficientNet) to find which gives the best accuracy, and later show that a Deep CNN architeture defined by us beats them all.
Keywords: Deep Learning, Agriculture, Image Classification,Deep CNN,VGG16, ResNet50, DetNet, InceptionV3, EfficientNet

1 Introduction

With the growth of global populations, agriculture is being placed under increasing pressure to increase yields and maximize efficiency. Satellite images have become regularly employed to monitor agricultural activities and provide important data to estimate crop conditions and yields.[1] The classifier developed in this project can be deployed by farmers and agricultural researchers to identify critical features in their field and better understand the impacts these can have on yield. Thereby they may take necessary steps to increase productivity of farmlands and get better yield. The project can identify certain patterns such as weed cluster which is undesirable in a particular situation.
This project seeks to develop a set of best practices that can be applied to current and historical satellite data to identify and classify images based on several important features that impact crop yield. We hope that by building this project, data science can be used to improve security for people worldwide and that farmers are secure in knowing that they have access to the best technology, research and data possible.

Deep Learning[2][3] is one of the fragments of machine learning method where its algorithms are inspired by the functions of the brain of humans which is so called as ANN (Artificial Neural Network) with Representation learning, where learning can be supervised, unsupervised and semi-supervised.[4] The success of deep learning in visual recognition tasks has driven advancements in multiple fields of research[5]. Particularly, increasing attention has been drawn towards its application in agriculture[6][7]. Nevertheless, while visual pattern recognition on farmlands carries enormous economic values, little progress has been made to merge computer vision and crop sciences due to the lack of suitable agricultural image datasets. In the recent years many existing techs in the IT sector have been applying technology in the agriculture sector with the concept of "smart farming"[8] for more productivity, food security and sustainability.[9] To address these challenges it is important to analyze and understand the agricultural ecosystems. This creates huge amount of data that needs to be stored and processed, data can be of images which can be processed with different image processing techniques based on Deep learning using CNN etc.[10]

The main objective is to classify aerial farmland or agriculture field images into four classes namely, **Cloud Shadow**, **Standing Water**, **Waterway** and **Weed Cluster**
based on these observable patterns.[11] These patterns are critical and can be observed from satellite images. To solve this multi class classification problem this paper elucidates the training and testing of various deep learning models and find the best model.

This Paper is organized as follows: Section 1 covers general introduction, motivation behind taking up this problem statement, Objectives and Scope of our Work. Section 2 Literature Survey gives information about the relevant research already done in this field, research gaps and possible Applications of our work. Section 3 explains the methodology undertaken to achieve the objectives. Section 4 provides the details of the various experiments performed and the results inferred. Section 5 concludes the project work and tells about Future work.

2 Literature Survey

Deep Learning is one of the fastest developing technologies and is used to solve a variety of problems.[12] Many domains have already been identified where this might prove to be useful. Deep Learning has found many applications in Agriculture[13]. Existing research in Land Classification [14], Classification of paddy crop [15] and Crop Identification using deep learning[16] among other applications[1] which involve remote sensing [17] have been studied in detail. However the research gaps are in accurately identifying and classifying other significant agricultural patterns.

A major direction of visual recognition in agriculture is aerial image semantic segmentation. Solving this problem is important because it has tremendous economic potential. Specifically, efficient algorithms for detecting field conditions enable timely actions to prevent major losses or to increase potential yield throughout the growing season. However, this is much more challenging compared to typical semantic segmentation tasks on other aerial image datasets.[18]

Team SCG Vision from the Arctic University of Norway tried to solve the problem of semantic segmentation by building on Self-Constructing Graph (SCG) module, they use multiple views in order to explicitly exploit the rotational invariance in airborne images and further develop an adaptive class weighting loss to address the class imbalance.[19]

For the same problem Team TJU from the Tonji University proposed applying a Switchable Normalization block to DeepLabV3+ segmentation model to alleviate the feature divergence. And use KullbackLeibler divergence.[20]

Daven from London University developed an Agriculture Satellite image classifier[21] and found that Both resnet and vgg16 were experiments that proved ineffective at categorizing the imbalanced dataset both with and without augmentation. The gradients exploded or went to zero and the multiple changes of batch size and loss function were unsuccessful.

3 Methodology

This section tells about the methodology implied to achieve the goals and objective of this project. To build the Image Classifier using deep learning models.

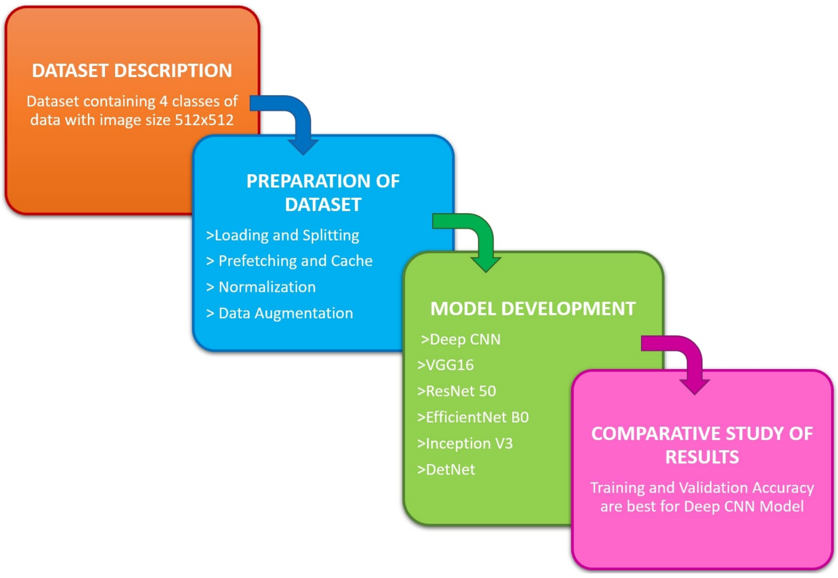


Figure 1: Complete Methodology

3.1 Dataset Description

We use a subset of the Agriculture Vision Dataset[18] for our research project. The original dataset contains 21,061 aerial farmland images captured throughout 2019 across the US. Each image consists of four 512x512 colour channels, which are RGB and Near Infra-red (NIR). Each image also has a boundary map and a mask. The boundary map indicates the region of the farmland, and the mask indicates valid pixels in the image. Regions outside of either the boundary map or the mask are not evaluated.
The subset we have used in the development of this project contains approximately 8000 aerial images in RGB colour format.
This dataset contains four types of annotations: Cloud shadow, Standing Water, Waterway and Weed cluster. These types of field anomalies have great impacts on the potential yield of farmlands; therefore, it is extremely important to accurately locate them.

3.2 Preparing the dataset

The subset of the Agriculture Vision dataset was manually prepared by the team so that it can be used to solve the problem statement at hand. The images initially grouped in a single folder, were segmented into the 4 labels.

3.2.1 Loading and Splitting

This dataset was then uploaded on Google Drive (a cloud storage device by Google) and then loaded into Colab Notebook (A product from Google Research, Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, deep learning, data analysis and education)
The dataset of aerial images was then split into training and testing datasets. We used 90 percent of images for training the model and 10 percent for testing or validation.

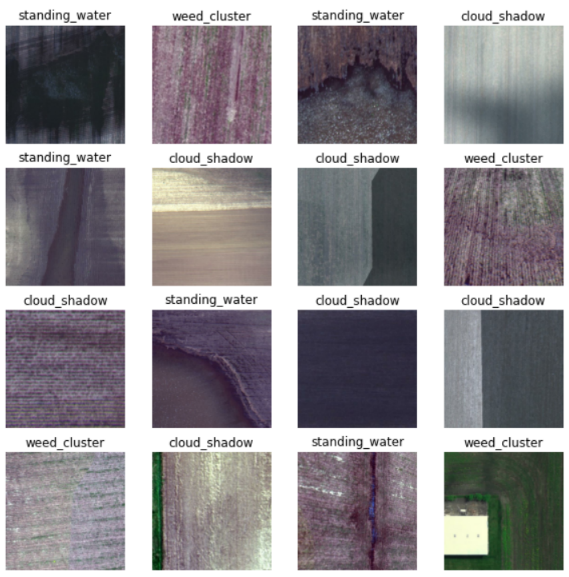


Figure 2: Sample images from Dataset

3.2.2 Use of Prefetching and Cache

We made use of buffered prefetching to enhance the performance. With this we can yield data from disk without having I/O become blocking to the process.Two important methods used when loading data, are:
Dataset.cache() keeps the images in memory after they're loaded off disk during the first epoch. This will ensure the data set does not become a bottleneck while training your model. When dataset is too large to fit into memory, this method creates a fast on-disk cache.
Dataset.prefetch() overlaps data preprocessing and model execution while training.While the model is executing training step s, the input pipeline is reading the data for step s+1. Doing so reduces the step time to the maximum (as opposed to the sum) of the training and the time it takes to extract the data.

3.2.3 Normalization

Using normalization learning becomes efficient also it can be used to avoid over-fitting of the model. The layer is added to the sequential model to standardize the input or the outputs.
We re-scaled the RGB colour channels from [0-255] to [0-1] in order to facilitate faster learning and uniform scaling.

3.2.4 Using Data Augmentation

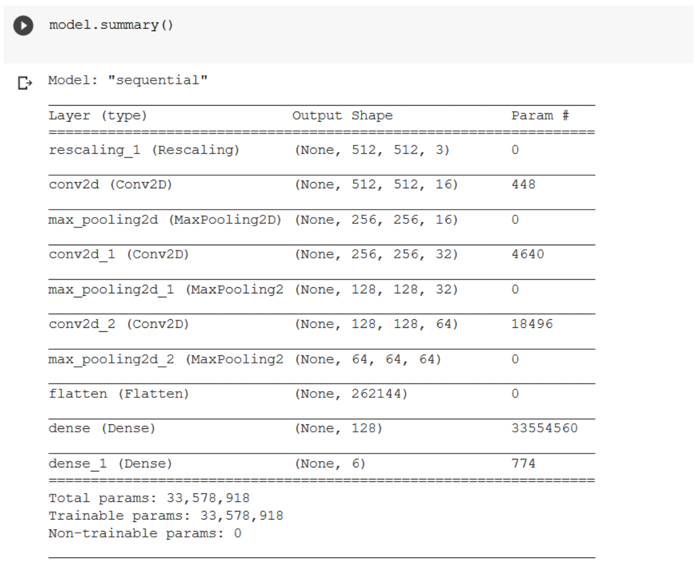
Data augmentation is a strategy that enables practitioners to significantly increase the diversity of data available for training models, without actually collecting new data. Data augmentation techniques such as cropping, padding, and horizontal flipping are commonly used to train large neural networks.
The various data augmentation techniques used in this project include Flipping the image left-right and top-bottom, Rotating, Skew tilting and random zoom. Using them the sample count of the dataset was increased to 8000 images.

3.3 Model Development

3.3.1 Deep CNN

We have trained a Deep Convolutional Neural Network (CNN) model[10] to work as the image classifier of aerial agricultural images based on the field patterns. A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The

Deep CNN model[22] is widely used by researchers across the globe to solve various existing problems
The various layers in our CNN are as follows:
1) Rescaling Layer
2) 3 Convolutional 2D Layers: with 16,32 and 64 filters, same padding and relu activation function
3) MaxPooling 2D Layer following each Convolutional 2D Layer
4) 1 Flattening layer to convert image dimensions into 1D Array
5) 2 Dense layers : Fully connected layers for classification



```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
rescaling_1 (Rescaling)      (None, 512, 512, 3)       0
_________________________________________________________________
conv2d (Conv2D)              (None, 512, 512, 16)      448
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 256, 256, 16)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 256, 256, 32)      4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 128, 128, 32)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 128, 128, 64)      18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 64, 64, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 262144)            0
_________________________________________________________________
dense (Dense)                (None, 128)               33554560
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 774
=================================================================
Total params: 33,578,918
Trainable params: 33,578,918
Non-trainable params: 0
_________________________________________________________________
```

Figure 3: Summary of the CNN Architecture defined

3.3.2 VGG16

The VGG-16 is one of the most popular pre-trained models for image classification. Developed at the Visual Graphics Group at the University of Oxford[23], VGG-16 beat the then standard of AlexNet and was quickly adopted by researchers and the industry for their image Classification Tasks.
The input to cov1 layer is of fixed size 224 x 224 RGB image. The image is passed through a stack of convolutional (conv.) layers, where the filters were used with a very small receptive field. Spatial padding of conv. layer is done so that the resolution is preserved. Pooling is carried out by five max-pooling layers over a 22 pixel window. Three Fully-Connected (FC) layers follow the convolutional layers. The final layer is the fully connected soft-max layer.

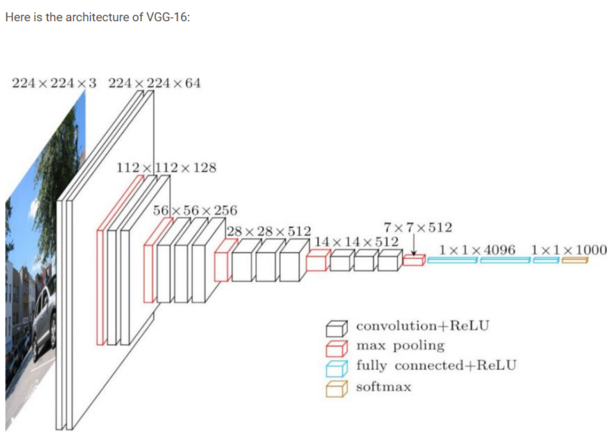


Figure 4: VGG 16 Architecture

Here is a more intuitive layout of the VGG-16 Model.

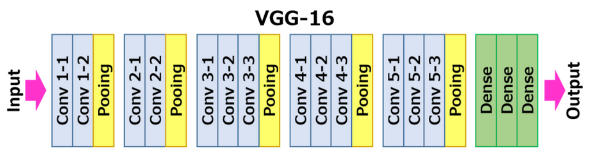


Figure 5: VGG 16 Architecture

3.3.3 ResNet-50

ResNet-50[24] short for Residual Networks is a convolutional neural network that is 50 layers deep. You can load a pretrained version of the network trained on more than a million images from the ImageNet database. The pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. This model was the winner of the ILSVRC 2015 classification task and since then is being widely used to solve many computer vision tasks.ResNet uses skip connection to add the output from an earlier layer to a later layer. This helps it mitigate the vanishing gradient problem. Due to its great efficiency in identifying high, mild and low level features it is a great tool.

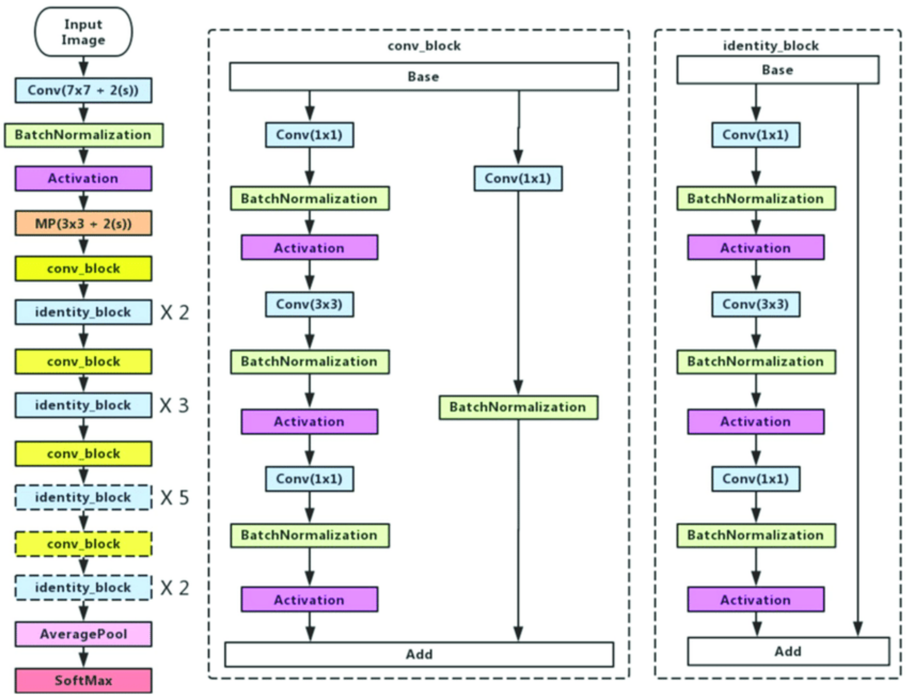
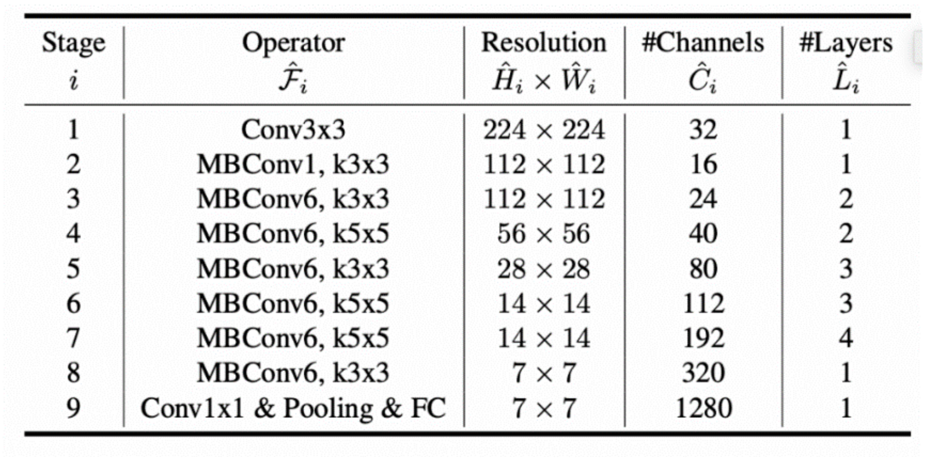


Figure 6: ResNet50 Architecture

3.3.4 Efficient Net

EfficientNet[25] is a widely used convolutional neural network architecture that uses a new Scaling Method called as Compound Scaling i.e if we scale the dimensions of the model by a fixed amount at the same time uniformly, we will achieve much better performance. The architecture of the model was developed by Google Research,Brain Team.
The researchers started with baseline model called as EfficientNet-B0. Then some scaling methods were applied on the baseline model to obtain a family of models called EfficientNets - EfficientNet-B0 to EfficientNet-B7. EfficientNet model is used for Image classification that has been shown to attain 76.3 percent with Baseline Model and 82.6 percent with EfficientNet-B7 on the imagenet database.

The base EfficientNet-B0 network is based on the inverted bottleneck residual blocks of MobileNetV2[26], in addition to squeeze-and-excitation blocks.
We have used EfficientNet-B0 as the base model for developing our image classifier.



Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

Figure 7: EfficientNet-B0 Architecture

3.3.5 InceptionV3

Inception v3 is a widely-used image recognition model that has been shown to attain greater than 78.1% accuracy on the ImageNet dataset. The model is the culmination of many ideas developed by multiple researchers over the years. It is based on the original paper: "Rethinking the Inception Architecture for Computer Vision"[27].

The model itself is made up of symmetric and asymmetric building blocks, including convolutions, average pooling, max pooling, concats, dropouts, and fully connected layers. Batchnorm is used extensively throughout the model and applied to activation inputs. Loss is computed via Softmax.

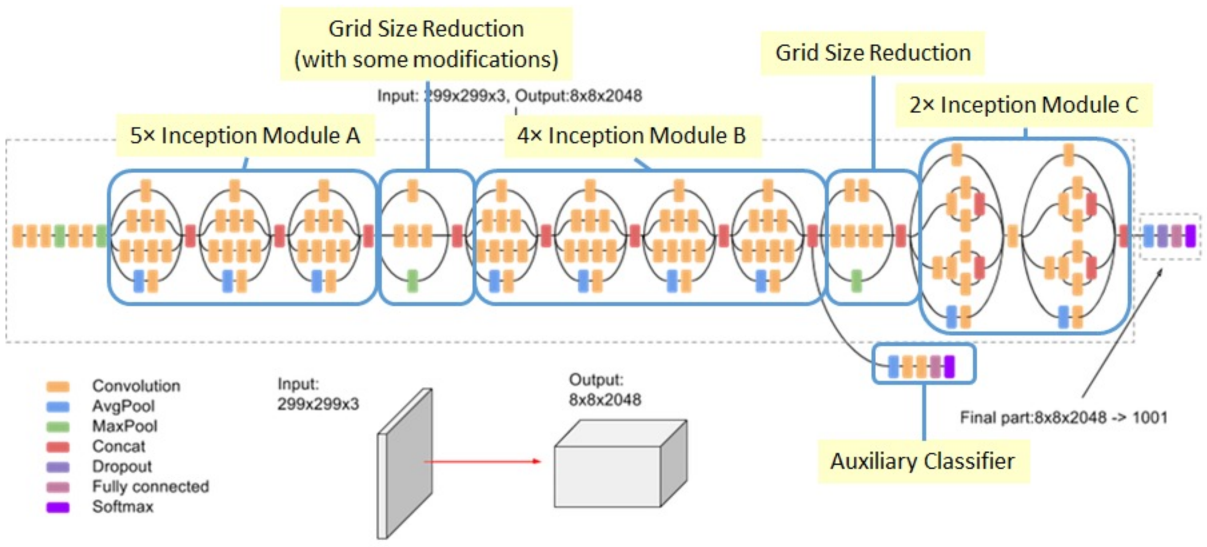


Figure 8: InceptionV3 Architecture

3.3.6 DetNet

DetNet is a backbone convolutional neural network for object detection. Different from traditional pre-trained models for ImageNet classification, DetNet maintains the spatial resolution of the features even though extra stages are included.
To keep the efficiency of DetNet, a low complexity dilated bottleneck structure is employed. It is based on the original paper: DetNet: A Backbone network for ObjectDetection by Zeming Li, Chao Peng and others[28]. Though its forte is object detection, it is well equipped to correctly perform multi-class classification with around 50% accuracy.

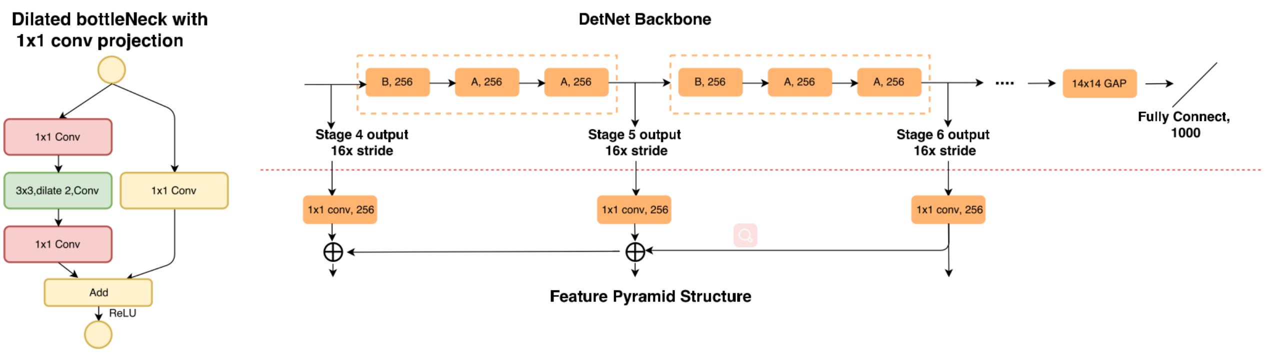


Figure 9: DetNet Architecture

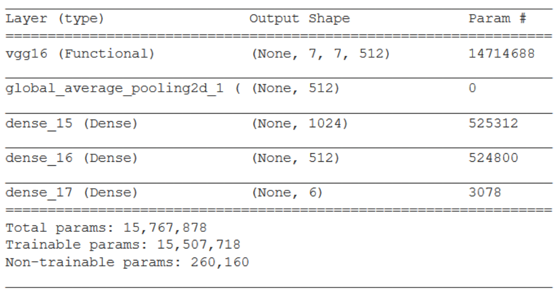
3.3.7 Transfer Learning

Transfer learning generally refers to a process where a model trained on one problem is used in some way on a second related problem.[29][30]
In deep learning, transfer learning is a technique whereby a neural network model is first trained on a problem similar to the problem that is being solved. One or more layers from the trained model are then used in a new model trained on the problem of interest.

Its advantages are as follows:
1) Useful Learned Features: The models have learned how to detect generic features from photographs, given that they were trained on more than 1,000,000 images for 1,000 categories.
2) State-of-the-Art Performance: The models achieved state of the art performance and remain effective on the specific image recognition task for which they were developed.
3) Easily Accessible: The model weights are provided as free downloadable files and many libraries provide convenient APIs to download and use the models directly.

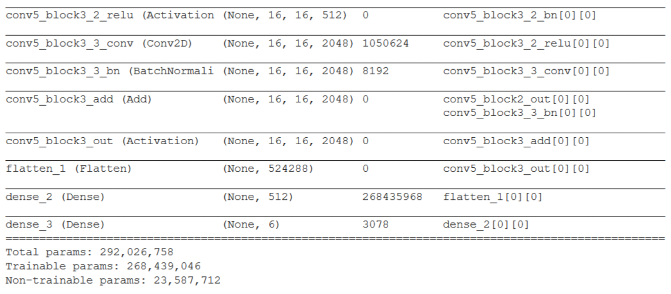
3.3.8 Modifying Architecture

The architecture of the above discussed pre-defined models was modified to suit our needs and classify our dataset. We removed the last layer from the original model and added 2 dense layers based on our specifications (relu and softmax activation functions resp). We then, compiled these modified versions and trained them on our dataset parallelly validating its accuracy too.



```
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Functional)           (None, 7, 7, 512)         14714688

global_average_pooling2d_1 ( (None, 512)               0

dense_15 (Dense)             (None, 1024)              525312

dense_16 (Dense)             (None, 512)               524800

dense_17 (Dense)             (None, 6)                 3078
=================================================================
Total params: 15,767,878
Trainable params: 15,507,718
Non-trainable params: 260,160
```

Figure 10: Modified VGG16 Model Summary



```
________________________________________________________________________________________________
conv5_block3_2_relu (Activation (None, 16, 16, 512)  0           conv5_block3_2_bn[0][0]
________________________________________________________________________________________________
conv5_block3_3_conv (Conv2D)    (None, 16, 16, 2048) 1050624     conv5_block3_2_relu[0][0]
________________________________________________________________________________________________
conv5_block3_3_bn (BatchNormali (None, 16, 16, 2048) 8192        conv5_block3_3_conv[0][0]
________________________________________________________________________________________________
conv5_block3_add (Add)          (None, 16, 16, 2048) 0           conv5_block2_out[0][0]
                                                                 conv5_block3_3_bn[0][0]
________________________________________________________________________________________________
conv5_block3_out (Activation)   (None, 16, 16, 2048) 0           conv5_block3_add[0][0]
________________________________________________________________________________________________
flatten_1 (Flatten)             (None, 524288)       0           conv5_block3_out[0][0]
________________________________________________________________________________________________
dense_2 (Dense)                 (None, 512)          268435968   flatten_1[0][0]
________________________________________________________________________________________________
dense_3 (Dense)                 (None, 6)            3078        dense_2[0][0]
================================================================================================
Total params: 292,026,758
Trainable params: 268,439,046
Non-trainable params: 23,587,712
```

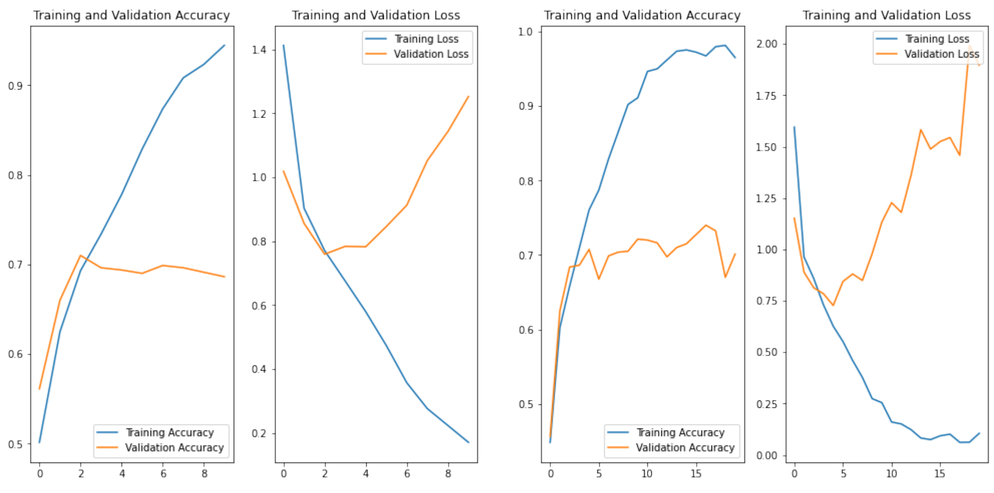
Figure 11: Last few layers of Modified ResNet50 Model Summary

4 Experiments and Results

4.1 Experiments on Deep CNN

4.1.1 Experiment 1

After defining the architecture of a Deep Convolutional Neural Network (Deep CNN or DCNN) The model was compiled using the Adam optimizer[31] and the loss was calculated using the SparseCategoricalCrossentropy function[32] and trained for 10 epochs with a batch size of 32. The training accuracy was 93.71% and validation accuracy of 71% in this experiment. Next, we increased the number of epochs to 20. Subsequently, the training accuracy rose to 98.65% and the validation accuracy rose to 74%

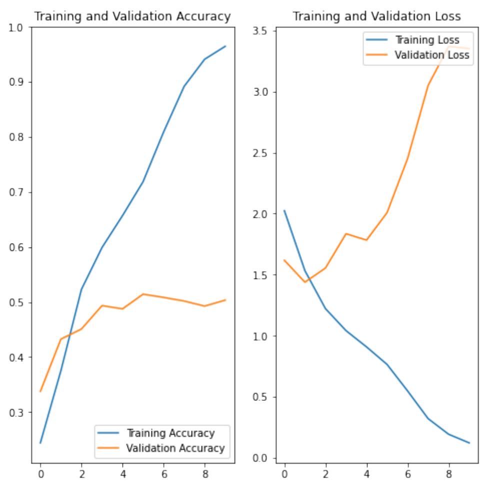


(a) 10 epochs (b) 20 epochs

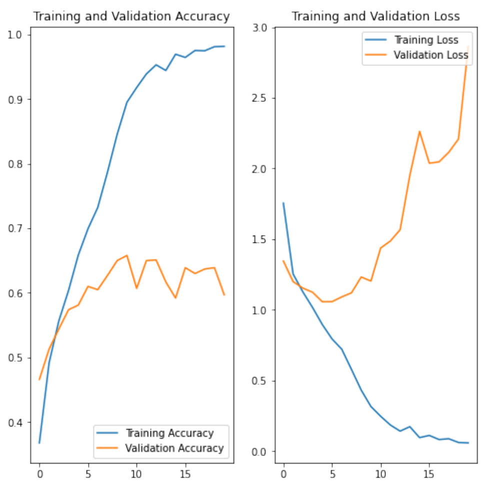
Figure 12: Deep CNN Exp 1 Accuracy and Loss Graph

4.1.2 Experiment 2

In this experiment, SGD optimizer[33] was used. After 10 epochs, the training accuracy was 95.93% and validation accuracy of 51.42% When the number of epochs was changed to 20, the new training accuracy calculated was 98.47% and the corresponding validation accuracy was 65.8%.



(a) 10 epochs



(b) 20 epochs

Figure 13: Deep CNN Exp 2 Accuracy and Loss Graph

4.2 Experiments on VGG 16

4.2.1 Experiment 1

We loaded the pre-trained VGG-16 Model with its weight calculated on the ImageNet dataset. Removing the top layer we added a Dense layer of 4 neurons with softmax activation function.
The model was compiled using the Adam optimizer and the loss was calculated using the SparseCategoricalCrossentropy function, and trained for 10 epochs (10 steps per epoch) with a batch size of 32. Training accuracy was 17.73% and validation accuracy of 16.83% in this experiment.

4.2.2 Experiment 2

After loading the pre-trained model VGG-16, we trained its last 8 scratch layers, (2 blocks of 4 layers 3 convolutional 2D layers followed by a MaxPooling 2D layer) and declared other layers untrainable in order to reduce the number of parameters to be trained. We added a Flattening layer and 3 dense layers with 256,128 and 4 neurons resp. using the activation functions and relu and softmax for the last layer. 2 Dropout layers were also present with probability 0.2
The model was compiled using the SGD optimizer with a learning rate[34] of 0.0001 and the loss was calculated using the SparseCategoricalCrossentropy function, and trained for 20 epochs (20 steps per epoch) with a batch size of 16. The input image shape was set to (224,224,3). Training accuracy was 33.77% and validation accuracy of 34.13% in this experiment.

4.2.3 Experiment 3

After loading the pre-trained model VGG-16, we trained its last 12 layers from scratch, (3 blocks of 4 layers 3 convolutional 2D layers followed by a MaxPooling 2D layer) and declared other layers untrainable in order to reduce the number of parameters to be trained. We added a GlobalAveragePooling 2D layer and 3 dense layers with 1024,512 and 4 neurons resp. using the relu activation function.
The model was compiled using the Adam optimizer and the loss was calculated using the SparseCategoricalCrossentropy function, and trained for 20 epochs (10 steps per epoch) with a batch size of 32. The image size was reduced from the original (512,512) to a target size of (224,224) in the input. We achieved a validation accuracy of 49% in this experiment.

4.3 Experiments on ResNet50

4.3.1 Experiment 1

We load The pretrained Resnet50 Model and freeze all the layers. Then we add 1 flattening layer and 2 dense layers with 512,4 neurons and relu and softmax activation function respectively.
The model was compiled using the SGD Optimizer and the loss was calculated using the SparseCategoricalCrossentropy function. The model was trained for 10 epochs(10 steps per epoch) with a batch size of 32. Training Accuracy was 30.23% and Validation Accuracy was 27.13% in this experiment.

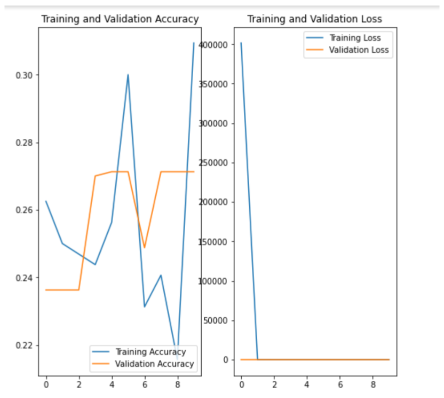


Figure 14: ResNet50 Exp 1 - Accuracy and Loss Graph

4.3.2 Experiment 2

In continuation with Experiment 1, the steps per epoch were increased to 20. Training Accuracy was 26.72% and Validation Accuracy was 27.13% in this experiment.

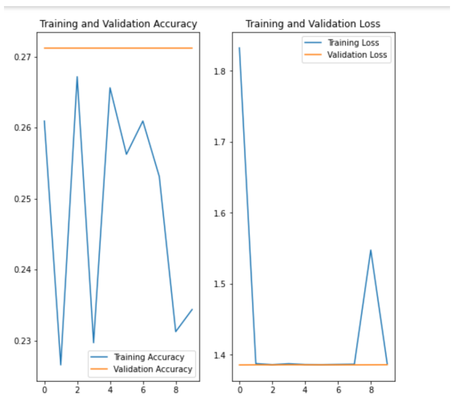


Figure 15: ResNet50 Exp 2 - Accuracy and Loss Graph

4.3.3 Experiment 3

In this experiment, we train only the last 12 layers. The model was compiled using the SGD Optimizer and the loss was calculated using the SparseCategoricalCrossentropy function. The model was trained for 10 epochs(10 steps per epoch) with a batch size of 32. Training Accuracy was 44.06% and Validation Accuracy was 44.37% in this experiment.

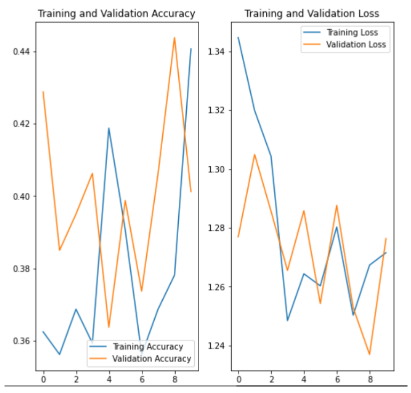


Figure 16: ResNet50 Exp 3 - Accuracy and Loss Graph

4.3.4 Experiment 4

Keeping other parameters same as in previous experiment, this time we increase steps per epoch to 20.Training Accuracy calculated was 43.28% and corresponding Validation Accuracy was 42.5% in this experiment.

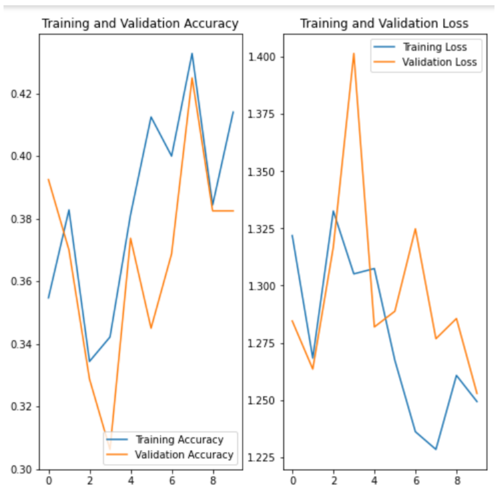


Figure 17: ResNet50 Exp 4 - Accuracy and Loss Graph

4.3.5 Experiment 5

Using the hyperparamters defined in previous experiments, the model was trained for 20 epochs(10 steps per epoch). Training Accuracy : 44.69% and Validation Accuracy : 44.37%.

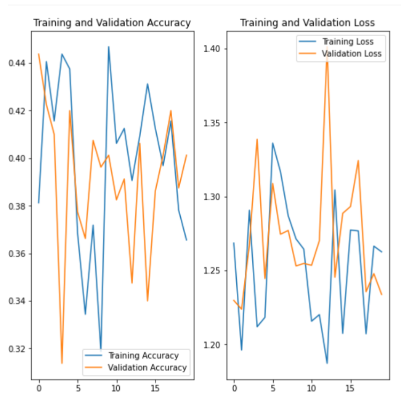


Figure 18: ResNet50 Exp 5 - Accuracy and Loss Graph

4.4 Experiments on EfficientNetB0

4.4.1 Experiment 1

We load the pretrained EfficientNetB0 Model with weight=None. We freeze all the layers of the model and add 1 flattening layer and 2 dense layers with of 512,4 neurons respectively with 'relu' activation function.

The model was compiled using the Adam Optimizer and the loss was calculated using the SparseCategoricalCrossentropy function. The model was trained for 10 epochs(10 steps per epoch) with a batch size of 32.Training Accuracy was 30.78% and Validation Accuracy was 24.5% in this experiment.

4.4.2 Experiment 2

We load The pretrained EfficientB0 Model and freeze all the layers except the last 12 layers. For the dense layers relu and sigmoid activation function were used. The model was compiled using the Adam Optimizer and the loss was calculated using the SparseCategoricalCrossentropy function. The model was trained for 10 epochs(10 steps per epoch) with a batch size of 32. Training Accuracy was 32.46% and Validation Accuracy was 27.13% in this experiment.

4.4.3 Experiment 3

In this experiment the steps per epoch were increased to 20. New Training Accuracy was 44.37% and Validation Accuracy was 43.87%.

4.5 Experiments on InceptionV3

4.5.1 Experiment 1

After defining the architecture of a InceptionV3 The model was compiled using the Adam optimizer and the loss was calculated using the SparseCategoricalCrossentropy function, and trained for 10 epochs with a batch size of 32. The training accuracy was 38.38% and validation accuracy of 28.50% in this experiment.

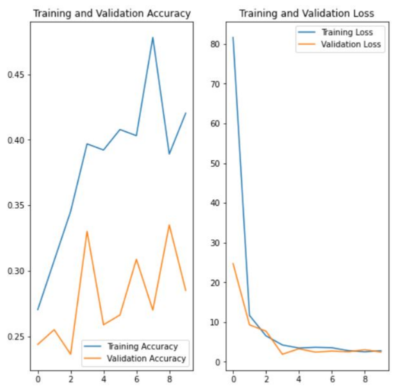


Figure 19: InceptionV3 Exp 1 - Accuracy and Loss Graph

4.5.2 Experiment 2

In this experiment, When the number of epochs was changed to 8, the new training accuracy calculated was 64.07% and the corresponding validation accuracy was 49.13%.

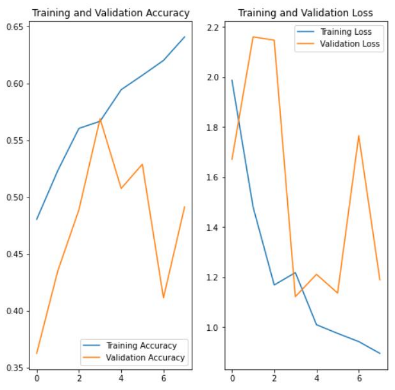


Figure 20: InceptionV3 Exp 2 - Accuracy and Loss Graph

4.5.3 Experiment 3

In this experiment, When the number of epochs was changed to 20, the new training accuracy calculated was 70.07% and the corresponding validation accuracy was 47.87%.

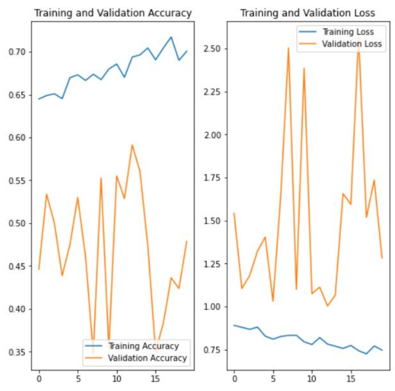


Figure 21: InceptionV3 Exp 3 - Accuracy and Loss Graph

4.6 Experiments on DetNet

4.6.1 Experiment 1

We use the model architecture defined in the original research paper. We use various methods such as changing the number of epochs, changing the optimizers and changing the steps per epoch to obtain different results.
No changes were made to the model itself in order to maintain its efficiency using the dilated bottleneck structure but specific changes were made such as changing the image size to (224, 224) so that the database could be incorporated into the model. This model employs Resnet-50 model as the baseline and then there are six convolution layers, averagepooling layer, flattening layer and denselayer in the end. The ReLU activation function is used throughout the model.

In this experiment we use the SGD (Stochastic gradient descent) optimizer; loss being calculated using the SparseCategoricalCrossentropy function and training for 20 epochs with 10 steps per epoch. For these arguments, Training accuracy was 78.71% and validation accuracy was 53.75%.

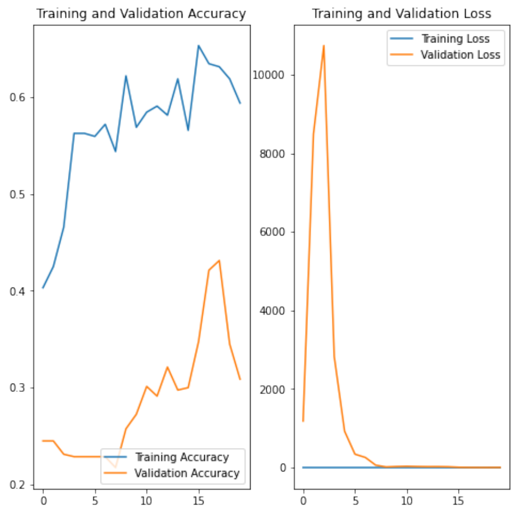


Figure 22: DetNet Exp 1 - Accuracy and Loss Graph

4.6.2 Experiment 2

Further in this experiment we try to increase the accuracy and the efficiency of the model by changing the number of epochs to 20(10 steps per epoch) using the Adam optimizer keeping the batch size at 32 ; we get the Training accuracy of 65.24% and Validation accuracy of 43.13%. If we further try to increase the steps per epoch to 20, we encounter an error stating the whole data set has been utilised. This implies that the accuracy obtained currently is the best for the current set of arguments.

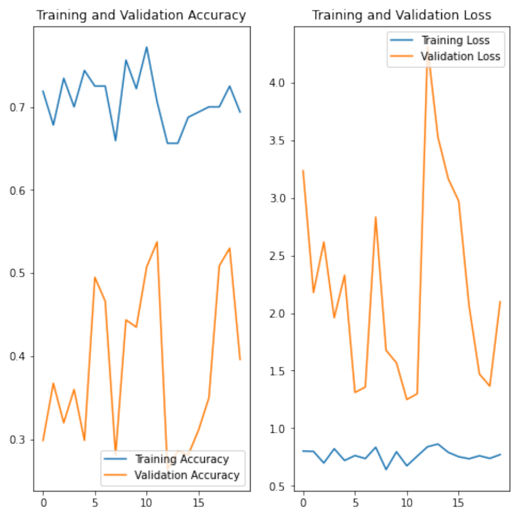


Figure 23: DetNet Exp 2 - Accuracy and Loss Graph

4.7 Evaluation Indexes

In order to evaluate the performance of the various models developed, and to make a comparative study to recognize the best performing model; this study uses various evaluation indexes as follows:

Training Accuracy:
It is the measure of how well is the model trained on the given dataset. In this case, the training dataset being the 90% of total dataset. The calculation of accuracy of single pair is done by checking if the predicted class is the same as the true class and returning the value in 0 or 1.Then the function calculates the overall accuracy of the data set, by using the conventional accuracy formula, which is defined as:

$$\text{Training Accuracy}=[\frac{amount\ of\ correct\ guesses\ in\ training\ dataset}{total\ amount\ of\ guesses\ in\ training\ dataset}](1)$$

Validation Accuracy:
It is the measure of the actual efficiency of the model. It shows how well the can predict the correct class when the input is new set of images which the model has not seen before. In this case the input amounts to 10% of the total dataset. The accuracy is defined as:

$$\text{Validation Accuracy}=[\frac{amount\ of\ correct\ guesses\ in\ validation\ dataset}{total\ amount\ of\ guesses\ in\ validation\ dataset}](2)$$

4.8 Summary Of Experiments

Experiment No.	Batch Size	Image Size	Optimizer	No. of Epochs	Training Accuracy	Validation Accuracy
1	32	512x512	Adam	10	93.71%	71%
1	32	512x512	Adam	20	98.65%	74%
2	32	512x512	SGD	10	95.93%	51.42%
2	32	512x512	SGD	20	98.47%	65.8%

Table 1: Deep CNN

Experiment No.	Batch Size	Image Size	Layers Trained	Layers Added	Activation Fns Used	No. of Epochs	Steps per Epoch	Optimizer	Training Accuracy	Validation Accuracy
1	32	512x512	12	1Dense(4)	Softmax	10	10	Adam	17.73%	16.83%
2	16	224x224	8	1Flat,3Dense(256,128,4),2Dropout	Relu,Softmax	20	20	SGD	33.77%	34.13%
3	32	512x512	12	1 GlobalAvgPooling,3 Dense(256,128,4)	Relu	20	10	Adam	47.91%	49%

Table 2: VGG 16

Experiment No.	Batch Size	Image Size	Layers Trained	Layers Added	Activation Fns Used	No. of Epochs	Steps per Epoch	Optimizer	Training Accuracy	Validation Accuracy
1	32	512x512	Nill	1Flat,2Dense(512,4)	Relu,Softmax	10	10	SGD	30.23%	27.13%
2	32	512x512	Nill	1Flat,2Dense(512,4)	Relu,Softmax	10	20	SGD	26.72%	27.13%
3	32	512x512	12	1Flat,2Dense(512,4)	Relu,Softmax	10	10	SGD	44.06%	44.37%
4	32	512x512	12	1Flat,2Dense(512,4)	Relu,Softmax	10	20	SGD	43.28%	42.5%
5	32	512x512	12	1Flat,2Dense(512,4)	Relu,Softmax	20	10	SGD	44.69%%	44.37%

Table 3: ResNet 50

Experiment No.	Batch Size	Image Size	Layers Trained	Layers Added	Activation Fns Used	No. of Epochs	Steps per Epoch	Optimizer	Training Accuracy	Validation Accuracy
1	32	512x512	Nill	1Flat,2Dense(512,4)	Relu	10	10	Adam	30.78%	24.5%
2	32	512x512	12	1Flat,2Dense(512,4)	Relu,Sigmoid	10	10	Adam	32.46%	27.13%
3	32	512x512	12	1Flat,2Dense(512,4)	Relu,Sigmoid	10	20	Adam	44.37%	43.87%

Table 4: EfficientNet B0

Experiment No.	Batch Size	Image Size	No. of Epochs	Optimizer	Training Accuracy	Validation Accuracy
1	32	512x512	10	Adam	38.38%	28.5%
2	32	512x512	8	Adam	64.07%	49.13%
3	32	512x512	20	Adam	70.07%	47.87%

Table 5: InceptionV3

Experiment No.	Batch Size	Image Size	No. of Epochs	Steps Per Epoch	Optimizer	Training Accuracy	Validation Accuracy
1	32	224x224	20	10	SGD	78.71%	53.75%
2	32	224x224	20	10	Adam	65.24%	43.13%

Table 6: DetNet

4.9 Comparative study of Results

As listed in the Experiments section several experiments were conducted in order to develop an image classifier for classifying the aerial images of Agricultural Field patterns. This project elucidates the accuracy achieved in training and then validating six different Deep Learning Models to serve the purpose. The table below enlists the best accuracy achieved so far in each model.

Sr. No.	Model Name	Optimizer	Best Training Accuracy	Best Validation Accuracy
1	Deep CNN	Adam	98.65%	74%
2	DetNet	SGD	78.71%	53%
3	InceptionV3	Adam	70.06%	49.13%
4	VGG16	Adam	47.91%	49%
5	ResNet50	SGD	44.69%	44.37%
6	EfficientNet B0	Adam	44.37%	43.85%

Table 7: Comparative Study of Results

The Overall best accuracy is achieved in case of the Deep CNN Model.

5 Conclusion and Future Scope

Through this project we have successfully trained six different deep learning models and achieved a decent level of accuracy in each case. The best accuracy was achieved in case of the deep convolutional neural network whose architecture was defined by us. The use of pre-trained models through the Transfer Learning technique proved quite inefficient in this regard. Possible explanations for the same include,
1) The classification of field patterns requires training on fewer parameters. The structure of the predefined models is very complex and is well - suited to extract even the smaller features. In this case, this confuses the model and results in poor accuracy.
2) These models need a larger data set in ten thousands to efficiently classify these images. Scarcity of farmland images becomes a bottleneck in this regard.

Also, both Adam and SGD optimizers proved to be efficient in this regard with Adam taking the upper hand.

The problem statement addressed by this project is unique and is hence a promising area of further research.

Future work includes building up on the Deep CNN model proposed in this project to get better accuracy. Extend the multi-class classification problem to classify more features/classes of images relevant to the domain.

References

[1] P. S. Luis Santos, Deep learning applications in agriculture, *AVC* (2019).

[2] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, and E. Muharemagic, Deep learning applications and challenges in big data analytics (2015).

[3] A. Shrestha and A. Mahmood, Review of deep learning algorithms and architectures (2017).

[4] B.-B. Benuwa, Y. Zhan, B. Ghansah, D. K. Wornyo, and F. B. Kataka, A review of deep machine learning (2016).

[5] W. D. VV Shakirov, Review of state-of-the-art in deep learning artificial intelligence, *Alerton Press* (2018).

[6] K. Dokic, L. Blaskovic, and D. Mandusic, From machine learning to deep learning in agriculture the quantitative review of trends (2020).

[7] N. Zhu, X. Liu, Z. Liu, K. Hu, Y. Wang, J. Tan, M. Huang, Q. Zhu, X. Ji, Y. Jiang, et al., Deep learning for smart agriculture: Concepts, tools, applications, and opportunities (2018).

[8] S.R.Rajeswari, P. Khunteta, A. R. S. Subham Kumar, and V. Pandey, Smart farming prediction using machine learning (2019).

[9] F. B. Andreas Kamaliaris, Deep learning in agriculture : A survey, *Springer* (2017).

[10] U. Z. Asifullah Khan, Anabia Sohail, A survey of the recent architectures of deep convolutional neural networks, *Springer* (2020).

[11] A. S. Nataliaa Kussul, Mykola Lavernuik, Deep learning classification of land cover and crop types using remote sensing data (2017).

[12] N. F. Hordri, S. S. Yuhaniz, and S. M. Shamsuddin, Deep learning and its applications: A review (2018).

[13] A. Bauer, A. G. Bostrom, J. Ball, C. Applegate, T. Cheng, S. Laycock, S. M. Rojas, and J. Kirwan, Combining computer vision and deep learning to enable ultra-scale aerial phenotyping and precision agriculture: A case study of lettuce production (2019).

[14] F. H. Manuel Taberner, Understanding deep learning in land use classification based on sentinel 2 time series, *nature research* (2020).

[15] S. P. Basavraj S. Anami, Naveen N. Malvade, Deep learning approach for recognition and classification of yield affecting paddy crop stresses using field images, *The Authors Publishing Services* (2020).

[16] S. K. Agila N, An efficient crop identification using deep learning, *International Journal of Scientific and Technology Research* (2020).

[17] J. Wolfe and H. Geospatial, Deep learning in agricultural remote sensing applications (2019).

[18] M. T. Chiu, X. Xu, Y. Wei, Z. Huang, A. Schwing, R. Brunner, H. Khachatrian, H. Karapetyan, I. Dozier, G. Rose, et al., Agriculture-vision: A large aerial image database for agricultural pattern analysis, *arXiv preprint arXiv:2001.01306* (2020).

[19] Q. Liu, M. Kampffmeyer, R. Jenssen, and A.-B. Salberg, Multi-view self-constructing graph convolutional networks with adaptive class weighting loss for semantic segmentation, *CVPRW* (2020).

[20] S. Yang, S. Yu, B. Zhao, and Y. Wang, Reducing the feature divergence of rgb and near-infrared images using switchable normalization, *CVPRW* (2020).

[21] D. dapsavoie and M. Piresten, Agricultural satellite image classifier (????).

[22] 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* (2014).

[23] K. Simonyan and A. Zisserman, Very deep convolutional networks for large-scale image recognition, *arXiv:1409.1556* (2015).

[24] K. He, X. Zhang, S. Ren, and J. Sun, Deep residual learning for image recognition (2020).

[25] M. Tan and Q. V. Le, Efficientnet: Rethinking model scaling for convolutional neural networks, *International Conference on Machine Learning* (2019).

[26] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, Mobilenetv2: Inverted residuals and linear bottlenecks, *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2018).

[27] C. Szegedy, V. Vanhoucke, S. Ioffe, and J. Shlens, Rethinking the inception architecture for computer vision, *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015).

[28] Z. Li, C. Peng, G. Yu, X. Zhang, Y. Deng, and J. Sun, Detnet: A backbone network for object detectionition, *arXiv pre-print :1804.06215v2* (2020).

[29] K. You, Z. Kou, M. Long, and JianminWang, Co-tuning for transfer learning, *NeurIPS* (2020).

[30] E. Rezende, G. Ruppert, T. Carvalho, F. Ramos, and P. de Geus, Malicious software classification using transfer learning of resnet-50 deep neural network (2018).

[31] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, *3rd International Conference for Learning Representations* (2015).

[32] H. Yessou, G. Sumbul, and B. Demir, A comparative study of deep learning loss functions for multi-label remote sensing image classification, *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)* (2020).

[33] S. Ruder, An overview of gradient descent optimization algorithms (2016).

[34] Y. Wu, L. Liu, and J. Bae, Demystifying learning rate policies forhigh accuracy training of deep neural networks (2019).