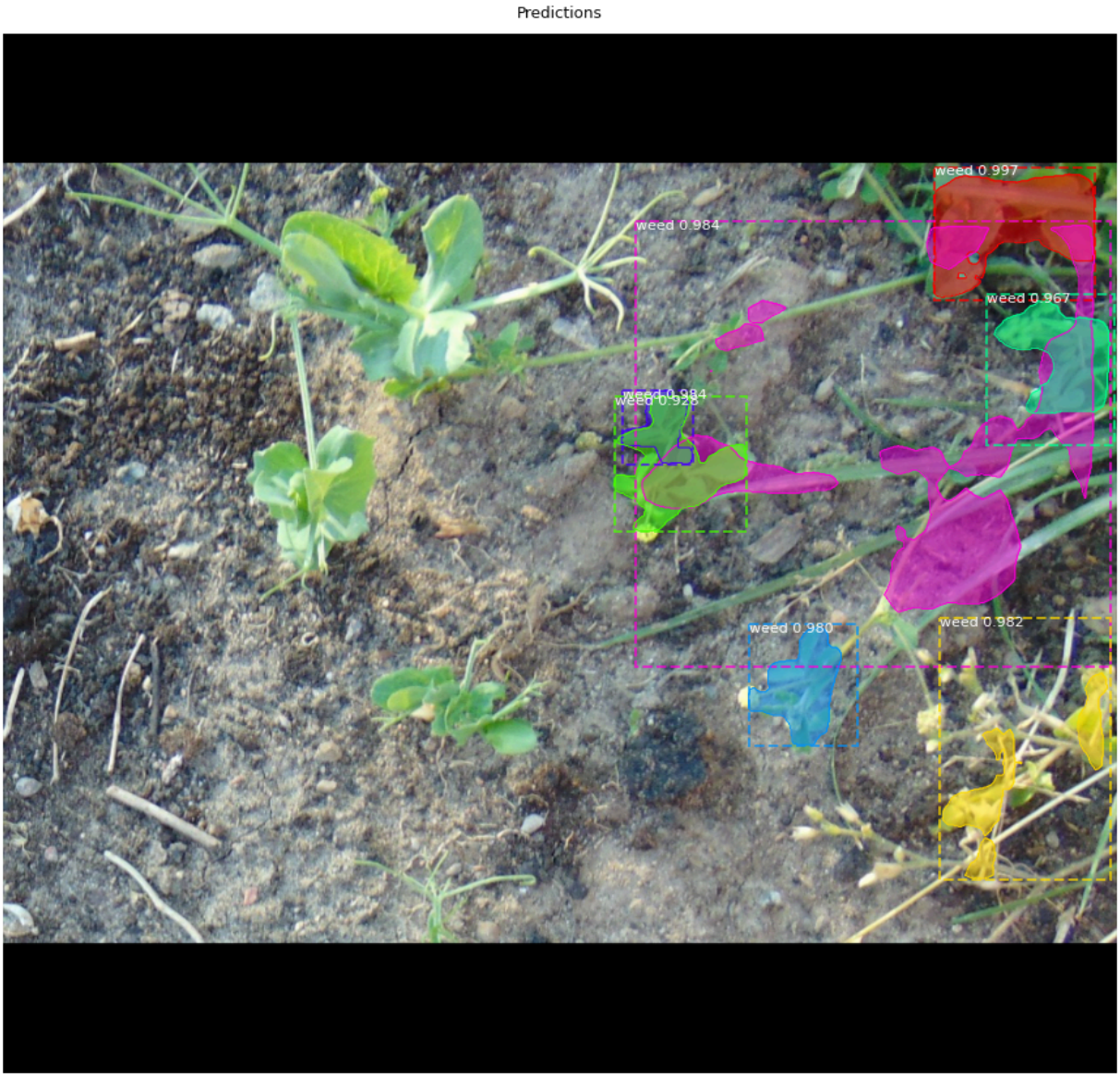
Robotics Interest Group(RIG) Project Report

Weed detection using image processing

By Ajin J and Abel C Dixon. Mentors:Pradeeshwar and Chandralekha

Link to project: [Weed Detection]

Technology Stack : tensorflow,keras,numpy,matplotlib,mrcnn

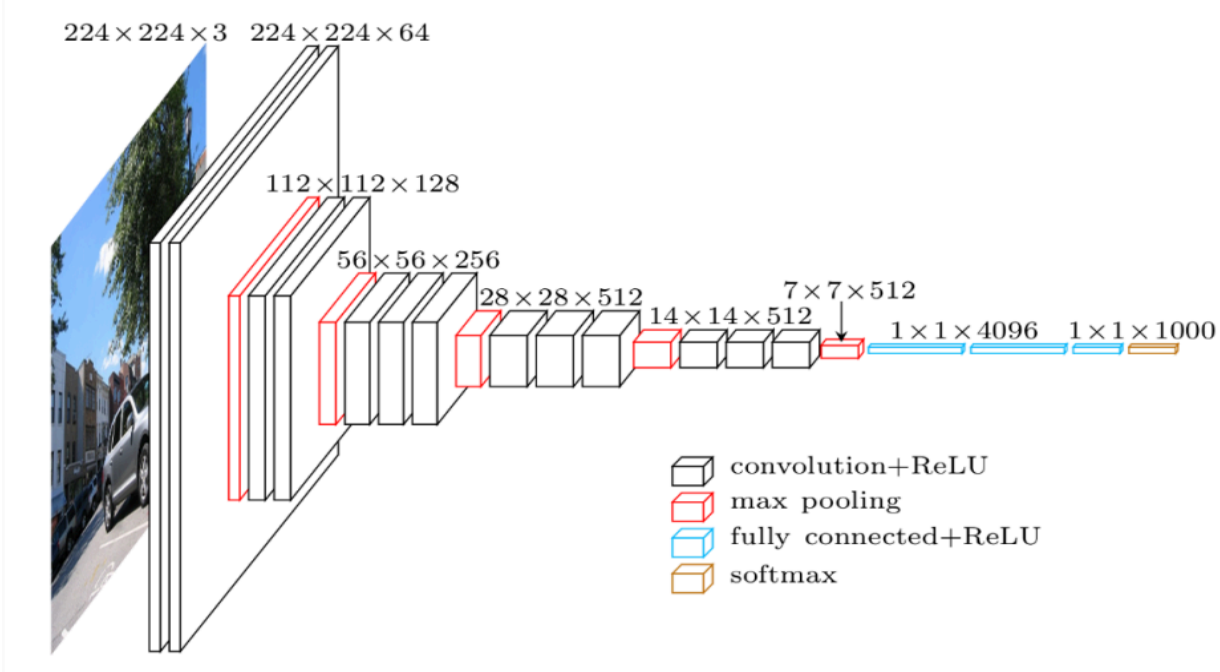


Introduction

In order to meet the demand for food products due to the growing population we need greater productive capability in agricultural sectors.Green revolution caused a rapid acceleration in this field terms of profit from the cash crops.The increased use of pesticides and herbicides had a huge impact and as a side effect it cause a huge damage to the environment.So in this project,we have implemented methods to reduce the usage of these herbicides by spraying them only in the areas where weeds are present.We have implemented image processing using MaskRCNN to detect weed area in an image.

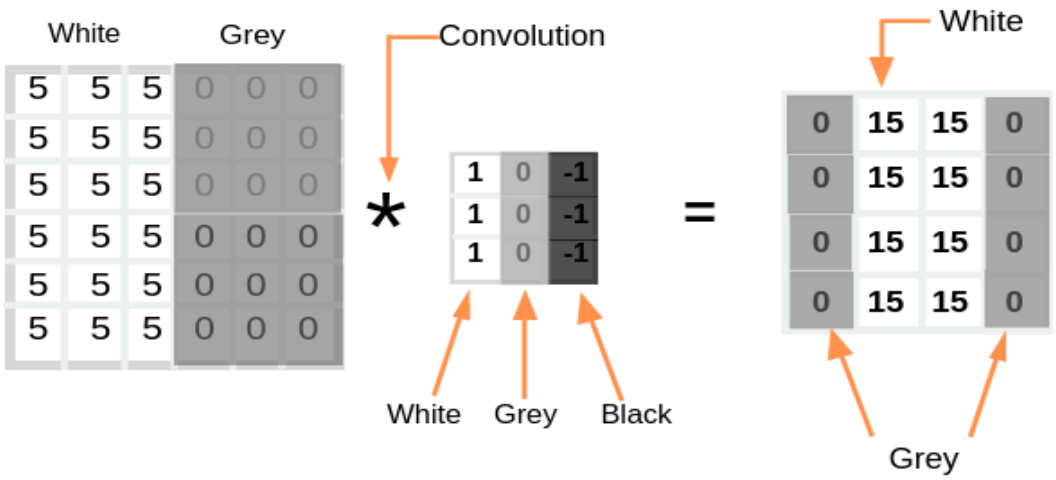
Understanding Convolutional Neural Network

ConvNets are the superheroes that took working with images in deep learning to the next level. With ConvNets, the input is an image ,or more specifically, a 3D Matrix. In simple word what CNN does is, it extract the feature of image and convert it into lower dimension without losing its characteristics



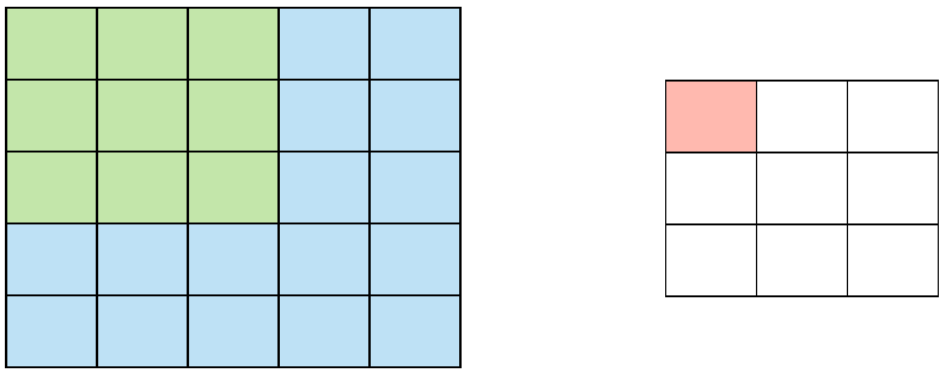
Edge Detection

Every image has vertical and horizontal edges which actually combine to form an image. Convolution operation is used with some filters for detecting edges. Suppose you have a grayscale image with dimension 6 x 6 and filter of dimension 3 x 3(say). When a 6 x 6 grey scale image convolves with a 3 x 3 filter, we get 4 x 4 image. First of all, the 3 x 3 filter matrix gets multiplied with the first 3 x 3 size of our grayscale image, then we shift one column right up to end , after that we shift one row and so on.



Stride and Padding

Stride denotes how many steps we are moving in each step in convolution.By default it is one.



We can observe that the size of output is smaller than input. To maintain the dimension of output as in input , we use padding. Padding is a process of adding zeros to the input matrix symmetrically.

Layers in CNN

There are five different layers in CNN

- Input layer
- Convo layer (Convo + ReLU)
- Pooling layer
- Fully connected(FC) layer
- Softmax/logistic layer
- Output layer

Input Layer

Input layer in CNN should contain image data. Image data is represented by a three dimensional matrix as we saw earlier. You need to reshape it into a single column. Suppose you have an image of dimension 28 x 28 =784, you need to convert it into 784 x 1 before feeding into input. If you have “m” training examples then the dimension of input will be (784, m).

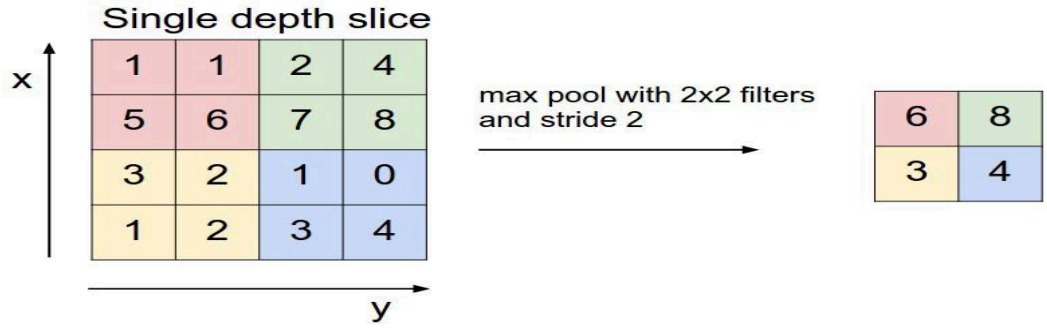
Convo Layer

Convo layer is sometimes called feature extractor layer because features of the image are extracted within this layer. First of all, a part of the image is connected to the Convo layer to perform convolution operation as we saw earlier and calculates the dot product between the receptive field(it is a local region of the input image that has the same size as that of filter) and the filter.Result of the operation is a single integer of the output volume. Then we slide the filter over the next receptive field of the same input image by a Stride and do the same operation again. We will repeat the same process again and again until we go through the whole image. The output will be the input for the next layer.

The Convo layer also contains ReLU activation (y=max(x,0)) to make all negative values to zero.

Pooling Layer

Pooling layer is used to reduce the spatial volume of the input image after convolution. It is used between two convolution layers. If we apply FC after the Convo layer without applying pooling or max pooling, then it will be computationally expensive.So, the max pooling is the only way to reduce the spatial volume of the input image. In the above example, we have applied max pooling in a single depth slice with Stride of 2. You can observe the 4 x 4 dimension input is reduced to 2 x 2 dimension.



Fully Connected Layer(FC)

The input to the fully connected layer is the output from the final Pooling or Convolutional Layer, which is flattened and then fed into the fully connected layer.

This Flattened vector is then connected to a few fully connected layers which are the same as Artificial Neural Networks and perform the same mathematical operations.

For each layer of the Artificial Neural Network, the g(Wx+b) is calculated where g(x) is the activation function,W is the weight matrix and b is the bias term.

Softmax / Logistic Layer

Softmax or Logistic layer is the last layer of CNN. It resides at the end of the FC layer. Logistic is used for binary classification and softmax is for multi-classification.

Output Layer

Output layer contains the label which is in the form of one-hot encoded.

Optimizers

Optimizers define how neural networks learn,they find the values of parameters such that loss function is at its lowest.Gradient descent involves taking small steps iteratively until it reaches the correct weights Θ and it updates its weights for every epochs sometimes so gradients can go typically large and may hover over the original value without actually reaching the optimal point .

$\Theta=\Theta-\propto \nabla_{\theta}(\Theta)$ ($\propto$ is the learning rate)

The solution to this is to update the parameters more frequently like in the case of stochastic gradient (SGD) descent where it updates the weight after seeing each datapoint in this case each image.But it too has it's side effects that it can create noisy jumps in the weights away from the optimal values that are getting updates on every sample.

Mini-batch gradient descent updates the weight only after a few samples over every epoch.An alternate way to reduce noise of SGD is to add the concept of momentum.

parameters of the model have a tendency to change in one direction as the example follows similar patterns. With this momentum the model can learn faster by paying little attention with updates made time to time.But blindly ignoring samples may be a costly mistake.Adding an acceleration term helps,But when it's find an odd sample it decelerate the weight update.Adagrad is an adaptive learning rate for every parameter.Adaptive learning rate optimizers able to learn more along one direction.But it's seen that learning rate decay to point it not longer update and hence no learning.Adadelta introduces a γ weight to the gradients and reduces the effects cause by Adagrad.Adam is a combination of Adadelta along with momentum it add the expected values of the past gradients,means it picks up speed over time and fast learning eventually faster convergence to optimal points.Hence we choose adam as a optimiser for the network head.

TRANSFER LEARNING

Transfer learning makes use of the knowledge gained while solving one problem and applying it to a different but related problem.

For example, knowledge gained while learning to recognize cars can be used to some extent to recognize trucks.

Pre-Training

When we train the network on a large dataset(for example: ImageNet) , we train all the parameters of the neural network and therefore the model is learned. It may take hours on GPU.

Fine Tuning

We can give the new dataset to fine tune the pre-trained CNN. Consider that the new dataset is almost similar to the original dataset used for pre-training. Since the new dataset is similar, the same weights can be used for extracting the features from the new dataset.

- If the new dataset is very small, it's better to train only the final layers of the network to avoid overfitting(where the model byheart the features), keeping all other layers fixed. So remove the final layers of the pre-trained network. Add new layers . Retrain only the new layers.
- If the new dataset is very much large, retrain the whole network with initial weights from the pretrained model.

RESENT-101 is one of the pre-trained models.

RESNET -101 Architecture

ResNet-101 is a convolutional neural network that is 101 layers deep. You can load a pre-trained 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.

Layer Name	Output Size	No. of Units
Convolution Layer 1	112x112	[7x7, stride 2, channel 64] x 1
Convolution Layer 2	56x56	[3x3 max pool, stride 2] x 1 [1x1, channel 64] x 3 [3x3, channel 64] x 3 [1x1, channel 256] x 3
Convolution Layer 3	28x28	[1x1, channel 128] x 4 [3x3, channel 128] x 4 [1x1, channel 512] x 4
Convolution Layer 4	14x14	[1x1, channel 256] x 23 [3x3, channel 256] x 23 [1x1, channel 1024] x 23
Convolution Layer 5	7x7	[1x1, channel 512] x 3 [3x3, channel 512] x 3 [1x1, channel 2048] x 3
Output	1x1	Average pool, 1000d-fc, soft-back

DATASET BUILDING

VGG Image Annotator

We are going to train an instance segmentation model, therefore we create pixel level mask annotations to define the boundaries of the objects in the dataset. Among various available tools, we choose an intuitive and well done tool: VGG Image Annotator (VIA).

This tool doesn't need any installation, you just download the package and open the via.html file with a modern browser.

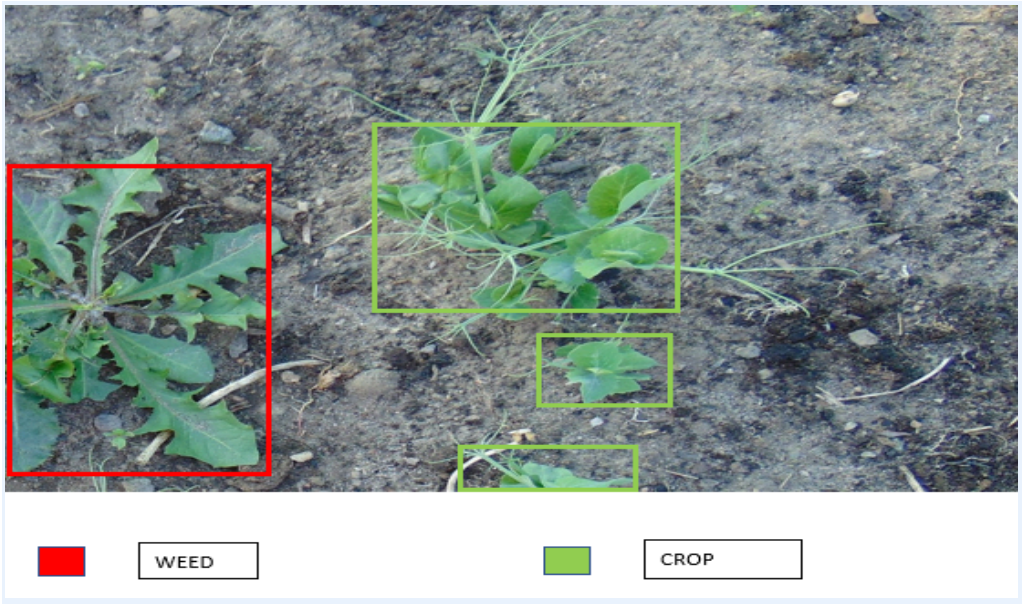


Image Annotated using VGG Image Annotator. Weed are annotated by polygon region shape.

When annotation is completed for all images, save the annotations as **JSON** files. Separately work for both Train and Test Datasets.

Image Classification

Detection and Segmentation

The several tasks in the computer vision includes semantic segmentation,classification along with localization,object detection and instance segmentation.

Understanding semantic segmentation

In semantic segmentation we want to input an image and then output a decision of category for every pixel in that image.For example consider this image where weed are present in the field

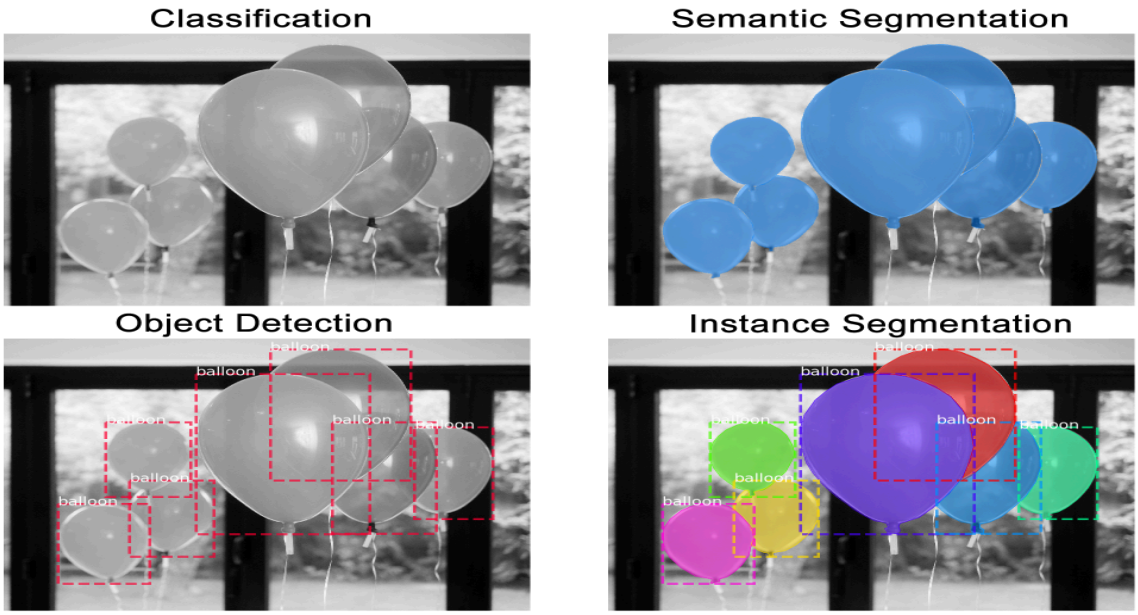
In the output we want to say for every pixel in the image we want to say does the the pixel belong to the plant or the background or some other set of categories.Rather than assigning single category image to entire image,we want to produce a category label for image for each pixel for the input or the image.Here we only need to annotate out the weed and the background,an this is called semantic segmentation.It does not differentiate instances.In the image shown above we have image with multiple similar crops close to each other the output does not distinguish each of the crops.Instead we get a whole mass of pixels that are all labeled as crops.This is a shortcoming of the semantic segmentation.

Understanding Instance Segmentation

Instance segmentation is the task of identifying object outlines at the pixel level.

- *Classification*: There is a balloon in this image.
- *Semantic Segmentation*: These are all the balloon pixels.

- *Object Detection*: There are 7 balloons in this image at these locations. We're starting to account for objects that overlap.
- *Instance Segmentation*: There are 7 balloons at these locations, and these are the pixels that belong to each one.



Background of MaskRCNN

- ❏ RCNN
- ❏ FastRCNN
- ❏ FasterRCNN

Network Backbone

- Region Proposal Network
- RoI(Region of Interest) Pooling
- RoI(Region of Interest) Align

Network Head

- Classification and Detection

- Segmentation

R-CNN (Region-based CNNs)

RCNN attempts to do object detection which use convolutional neural network as the feature extractor.An Input image extract 2000 regions from the image using the selective search which is called the region of proposal.Input image is resized according to input of the network.Instead of trying to classify a huge number of regions,it can just work with 2000 regions.

Selective search algorithm:

1. Generate initial sub-segmentation, we generate many candidate regions

2. Use greedy algorithm to recursively combine similar regions into larger ones

3. Use the generated regions to produce the final candidate region proposals

The extracted features are fed into an linear SVM(convert non linear feature to linear) to classify the presence of the object within that candidate region proposal and bounding box regression.

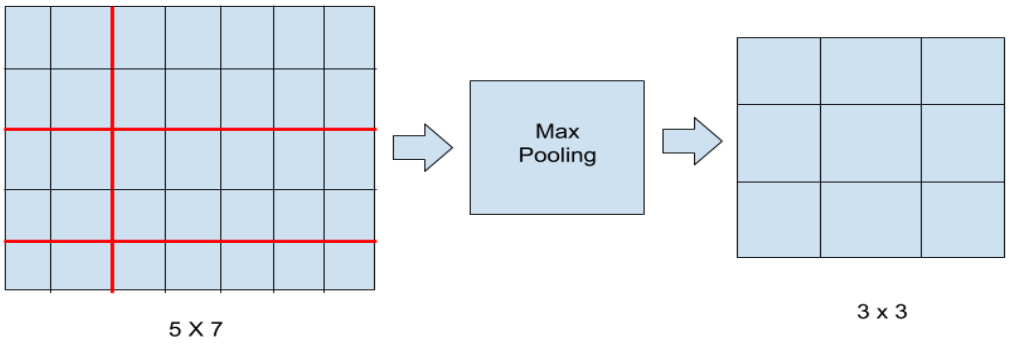
FastRCNN/FasterRCNN

FastRCNN did a number of changes to the pipeline of RCNN.It replaced the resizing operation with RoI pooling layer to get fixed size feature map.Secondly, it replace SVM's with fully connected layers which is responsible for classification and bounding box regression.But the region proposal is still based on selective search as mentioned above.

FasterRCNN introduced Region Proposal Network(RPN) instead of selective search.FasterRCNN doesn't perform object segmentation.

For a given image the Region Proposal Network propose good object candidates.RPN does that by dividing images to cells and for each cell it evaluates several anchor boxes of different size and aspect ratios and it also predicts their object scores.Anchor boxes with high object scores are fast forwarded in the pipeline.

RoI pooling in the FastRCNN works as follows given a matrix of 5x7 and you want to map it to matrix 3x3.To do that we divide the 5x7 into segments such that there are 3x3 of them and then we apply max pooling on each of these segments.This cause the mismatch between the input features and output features because of the different number of cells in each segments.RoI align in Mask RCNN fixes this problem.



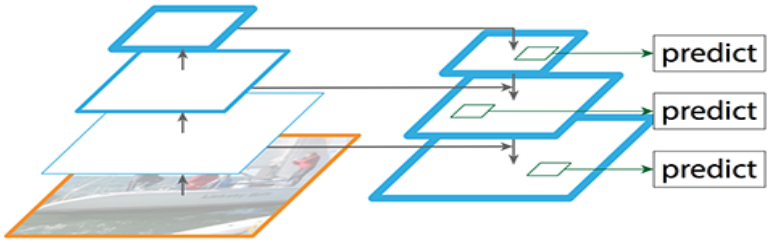
MaskRCNN

MaskRCNN adds a mask prediction branch to the last stage of FasterRCNN also RoI pooling layer is also replaced with RoI Align layer,which performs better in mask prediction than RoI pooling layer.MaskRCNN is also divided into two parts the backbone which is responsible for feature extraction (early layer detects low level features and later layer detect higher level features) and region proposal and head of the network which does the classification,regression and mask prediction.The MaskRCNN is built on Feature Pyramid Network and RestNet101 as backbone.

As the earlier model uses single feature map ,MaskRCNN is uses feature pyramid network

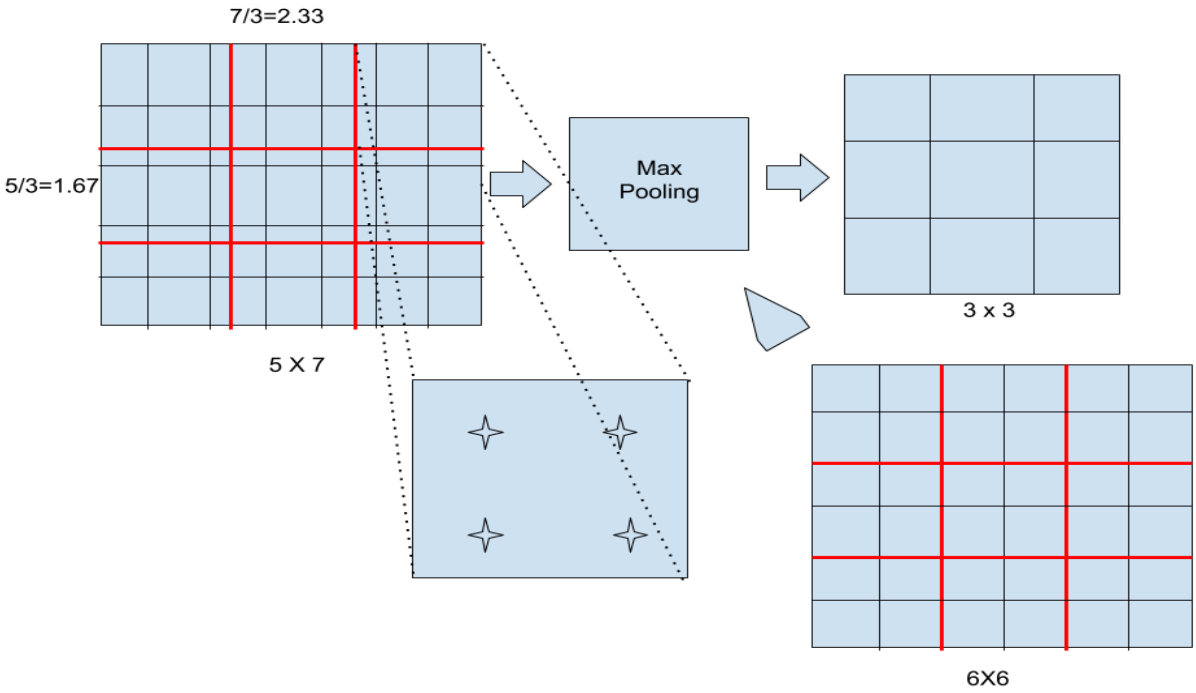
In single feature map architecture, an image goes through a series of convolutions and outputs a single feature map for prediction.However FPN architecture uses features from multiple convolution layers and we can choose any one or combination of them.This hierarchy of FPN gives a better prediction.

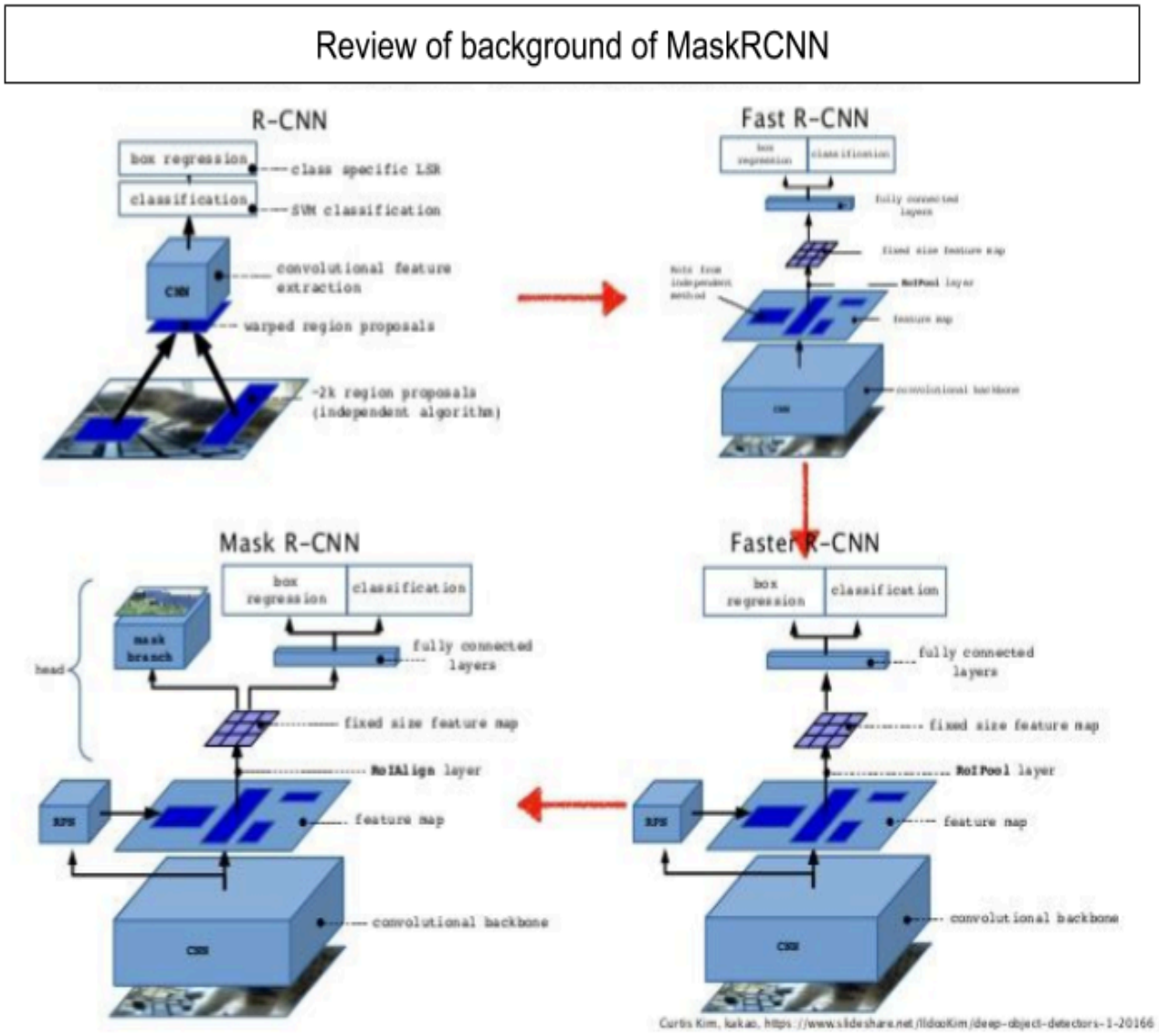
FPN Representation



RoI Align

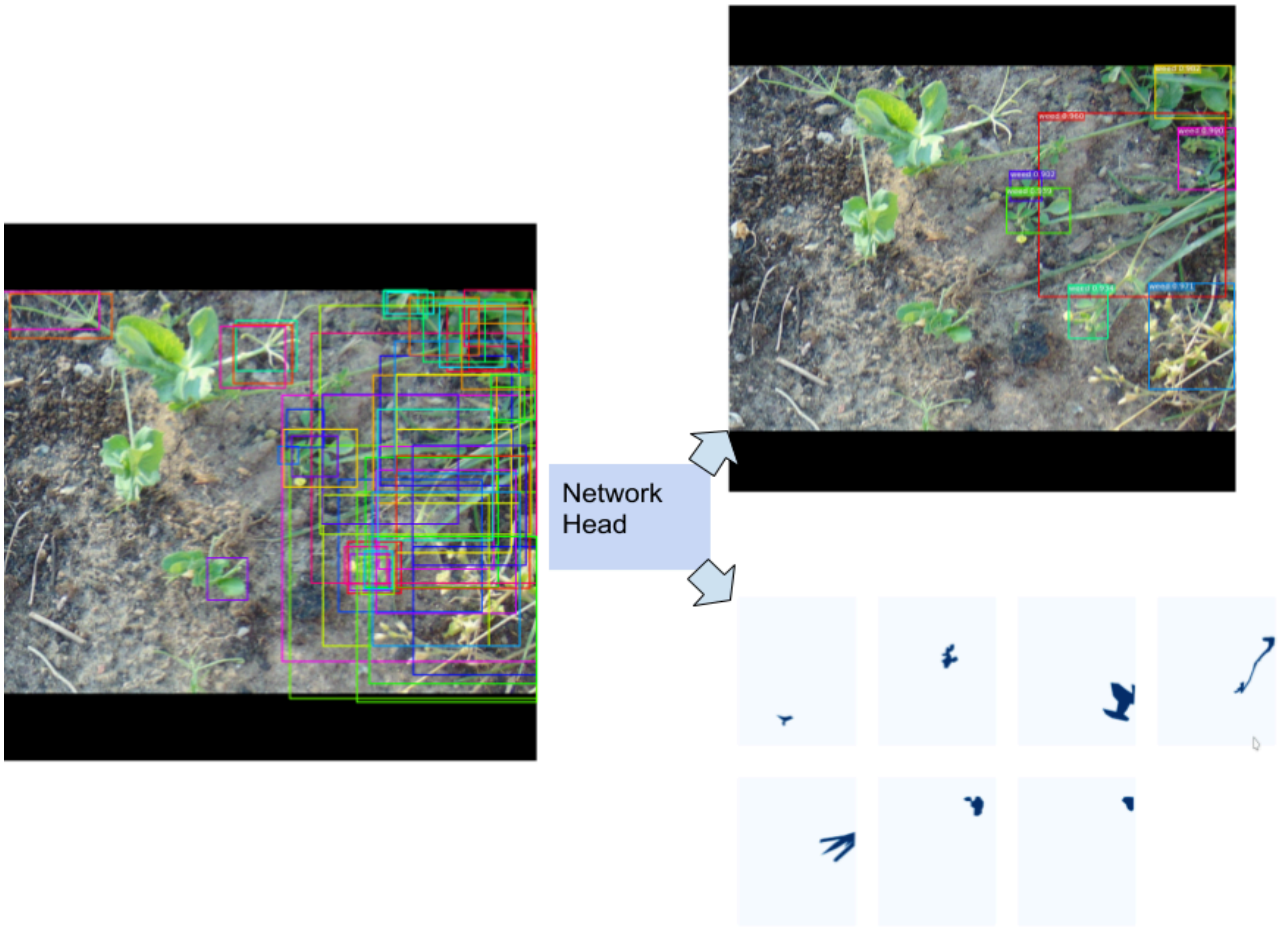
RoI align fixes the problem caused by the RoI pooling.It divides the feature map into 3x3 floating segments then in each segments it samples 4 points values are computed by linear interpolation.After this we end up with a 6x6 feature map which is equivalent to 5x7 feature map at this point we can use standard max pooling(each 2x2 block) to obtain output feature map.





Network Head

For the give image the bounding box without label is the proposed RPN, now it's the job of the network head to classify it with type with bounding box on it and generating the segmentation mask



Head:-(Classification and Detection)

The head has fully connected branch which classifies 2 (K+1 in general,K being the number of classes is the dataset and 1 is for the background) softmax for classification and predicts 4 (4K) bounding box regression targets (x,y,w,h).x and y are centers and w and h are the height and width of the bounding boxes.

This layer is trained using multiclass loss which is sum of classification loss and bounding box regression loss ,$\mathbf{L_{box}=L_{cls}+L_{reg}}$

Head:-Mask Prediction

This branch is fully convolutional and it trained using binary cross-entropy loss and the final loss is $\mathbf{L_{final}=L_{box}+L_{mask}}$

Mask Loss

Actual Image

Predicted Mask(used to compute the loss)

Mask loss is binary cross entropy however for a given image we get K mask predictions one for each class in the dataset.To compute the loss the mask for the class which is predicted by the classification layer is chosen.

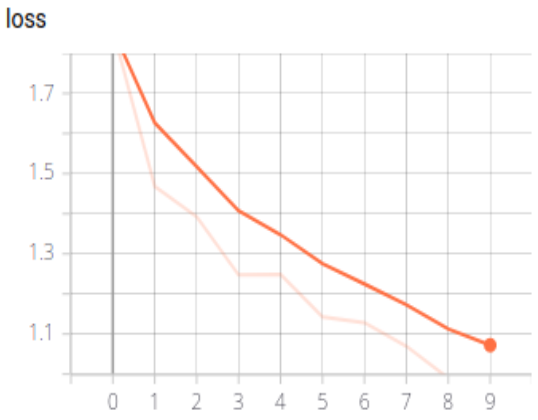
Datasets used by MaskRCNN

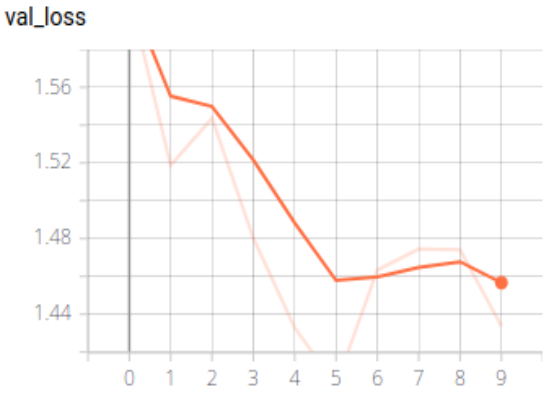
The main dataset used by MaskRCNN is MS COCO dataset which has 80 classes and 115 thousand training images.Evaluation metrics for bounding boxes and segmentation mask is based on Intersection over Union.We use the pretrained weight that model has learned on this datasets and used to train with our own datasets.

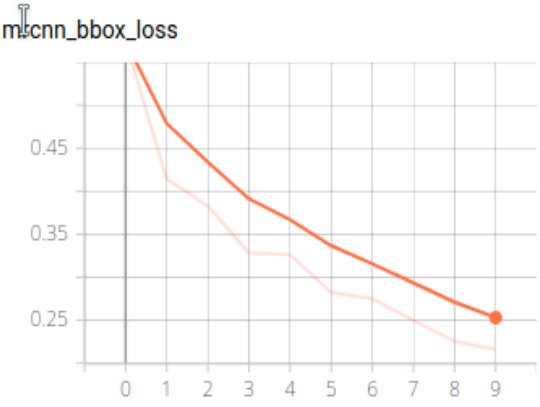
Results

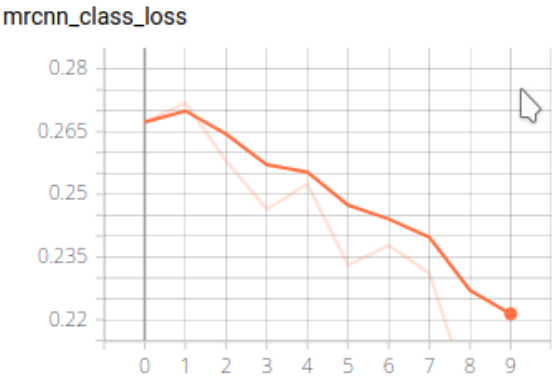
We used a dataset consisting of 202 images in which 183 images were used for training and 19 images for validation.It took around 44 minutes for 10 epochs with each epoch having 100 training batches and threshold of 0.9 for prediction. Training loss of 0.98 and validation loss of 1.43.Our model still achieved a decent amount of accuracy despite a small training dataset that we acquired.

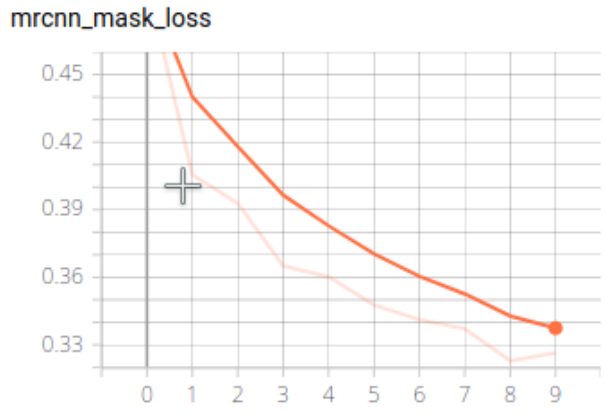
Tensorboard is used for data visualization





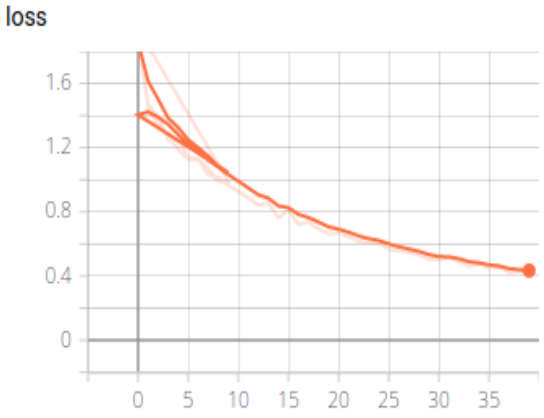






For 40 training epochs(77 minutes):

Here our model seems overfitting due to lack of a good amount of training data.

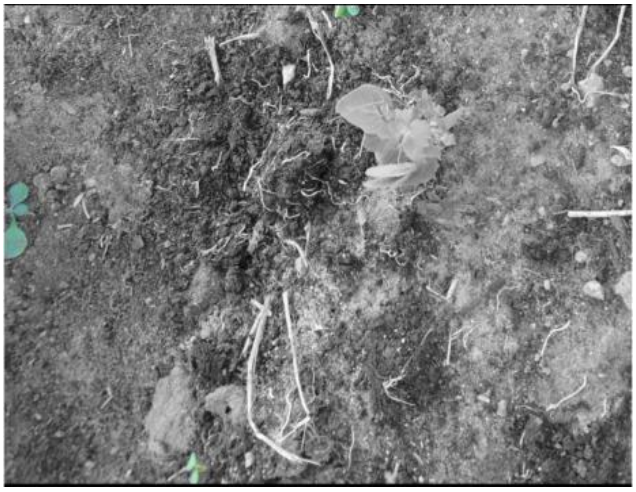


In our dataset we found different species of weeds (that vary in size and shape) are present,so extending the dataset by including more images with different weed species can increase the accuracy of our model.

Here are few results:

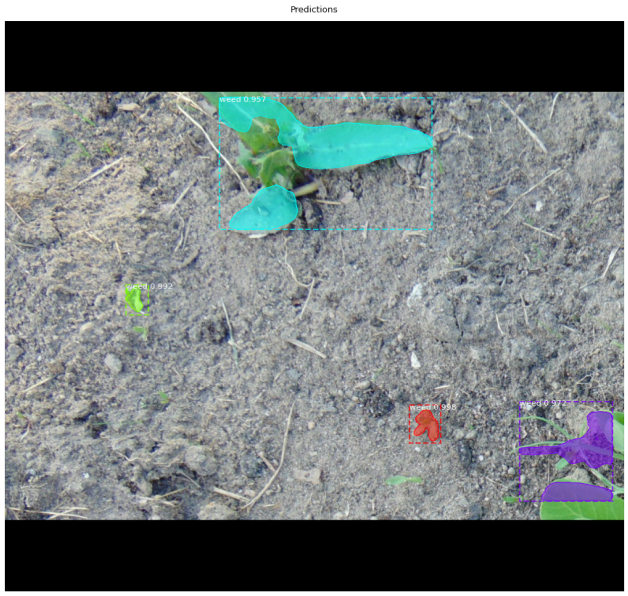
Color splash filter applied and corresponding mask on the weeds detected by the model



Predictions

Predictions



Future/Further improvements

The model can be further improved by gathering more datasets of various weed species and increasing the number of training iterations and image augmentation, which helps in implementing the same in the real agricultural field using drone technologies and robots for spraying. It can also be used to detect any specific type of weed with the click of an image on using the same model, lightweight version for smartphone applications.

References

1.) https://arxiv.org/pdf/1703.06870.pdf MaskRCNN paper
2.) https://arxiv.org/pdf/1506.01497.pdf FasterRCNN paper
3.) https://engineering.matterport.com/splash-of-color-instance-segmentation-with-mask-r-cnn-and-tensorflow-7c761e238b46
4.) https://github.com/matterport/Mask_RCNN
5.) https://arxiv.org/pdf/1504.08083.pdf FastRCNN paper