NC STATE UNIVERSITY

Team :30 PLANT PHENOTYPING

Members: Adithya Raghu Ganesh(araghug), Prasanth Yadla (pyadla2), Sharath Narayana(snaraya9)

ECE 542 – Neural Networks

Spring, 2020

Motivation

- Water stress in plants are identified through its **visual info** which could enable an **optimum watering strategy** for plants.
- Number of **leaf tips**, width of the plant and **collars** of the plant can be phenotypic indicators of water stress.

Introduction (Problem/Task)

- Our objective is to apply Image Processing and **Neural networks** to given images to **detect** leaves and collars as key points.
- This would help in identifying the **phenotype features** of the plant in an automated fashion.

Related Work

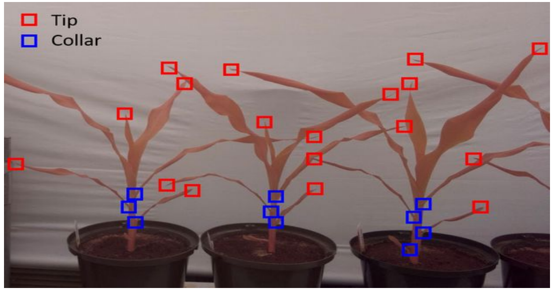
Work has been done on **leaf-spike detection** - Used SPIKE Dataset containing **images of wheat plots** to train R-CNN and **count spike regions**. **[1]**

Dataset and Data Pre-processing

- 476 Manually Annotated Images using MATLAB to **.mat format**.
- Converted the .mat format to python-friendly .dat format.
- Convert .dat file data to **consolidated CSV data format** (shown).
- Splitted into train and test.
- Generate **TF Records** for training.

	filename	width	height	class	xmin	ymin	xmax	ymax
2	pic_2019-12-21_15_00_17_273032_000001.jpg	1920	1080	collar	725	838	755	798
3	pic_2019-12-21_15_00_17_273032_000001.jpg	1920	1080	collar	717	887	694	849
4	pic_2019-12-21_15_00_17_273032_000001.jpg	1920	1080	collar	334	854	370	805
5	pic_2019-12-21_15_00_17_273032_000001.jpg	1920	1080	collar	334	889	291	864
6	pic_2019-12-21_15_00_17_273032_000001.jpg	1920	1080	collar	722	788	682	755

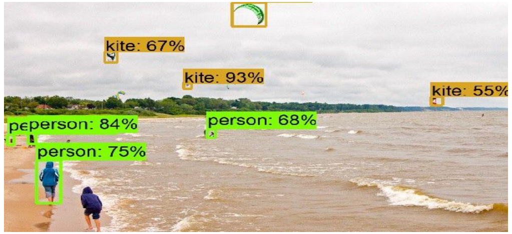
CSV formatted Data



Ground Truth

Methodology

[2]

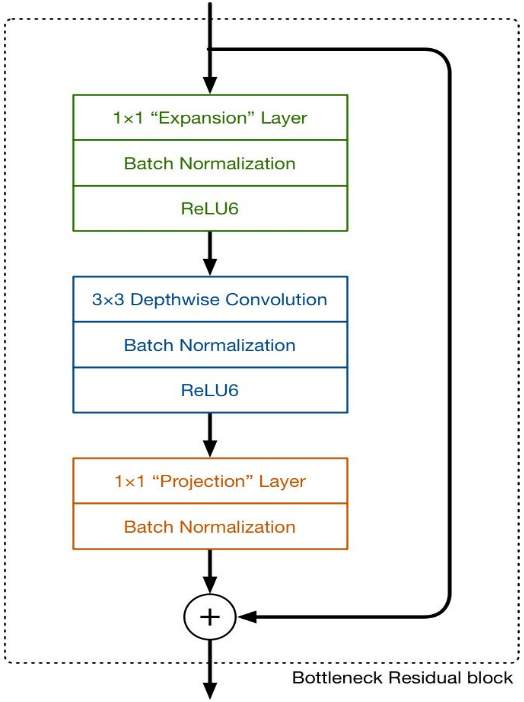


- Implemented using **python** with associated libraries such as **tensorflow**, protobuf and pillow. **[3]**
- Used **Tensorflow Object Detection Framework** which contains around 30 pre-trained models.
- Initially trained on CPU and then migrated our workload to ARC Cluster, **P4000, GTX super**.
- Choosing model configuration **Inception** and **MobileNet** with tensorboard visualization while training.
- Detection on test images using modified object detection script.

Model Architecture

SSD MobileNet

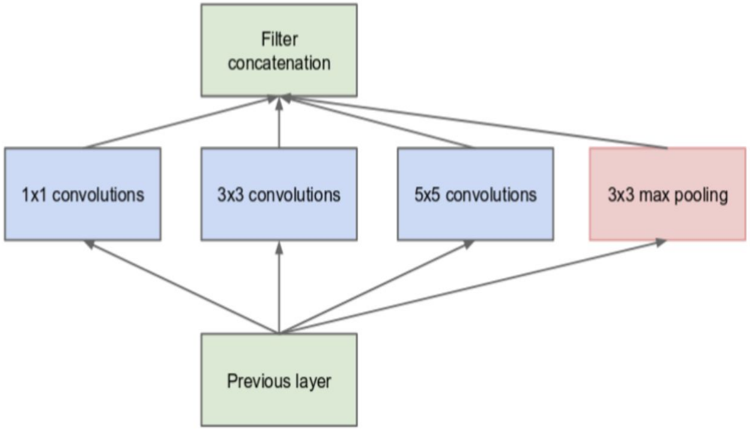
- Low computation.
- Mobile and embedded systems.
- Single filter based NN.
- Depthwise separable convolution.
- 53 Layers.
- 3.47 million params.



[5]

Inception Network

- Works when salient features are of varying size.
- Concatenates the convolutions.
- It uses multiple filters of different size on the same level.
- 22 Layers.
- 5 million parameters.



(a) Inception module, naïve version

[6]

Model configuration - Inception Network

```
model {
  ssd {
    num_classes: 2
    box_coder {
      faster_rcnn_box_coder {
        y_scale: 10.0
        x_scale: 10.0
        height_scale: 5.0
        width_scale: 5.0
      }
    }
    matcher {
      argmax_matcher {
        matched_threshold: 0.5
        unmatched_threshold: 0.5
        ignore_thresholds: false
        negatives_lower_than_unmatched: true
        force_match_for_each_row: true
      }
    }
```

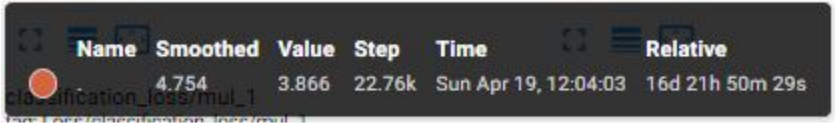
```
  box_predictor {
convolutional_box_predictor {
  min_depth: 0
  max_depth: 0
  num_layers_before_predictor: 0
  use_dropout: false
  dropout_keep_probability: 0.8
  kernel_size: 3
  box_code_size: 4
  apply_sigmoid_to_scores: false
  conv_hyperparams {
    activation: RELU_6,
    regularizer {
      l2_regularizer {
        weight: 0.00004
      }
    }
```

```
train_config: {
  batch_size: 24
  optimizer {
    rms_prop_optimizer: {
      learning_rate: {
        exponential_decay_learning_rate {
          initial_learning_rate: 0.004
          decay_steps: 800720
          decay_factor: 0.95
        }
      }
      momentum_optimizer_value: 0.9
      decay: 0.9
      epsilon: 1.0
    }
  }
```

Hyper parameter selection

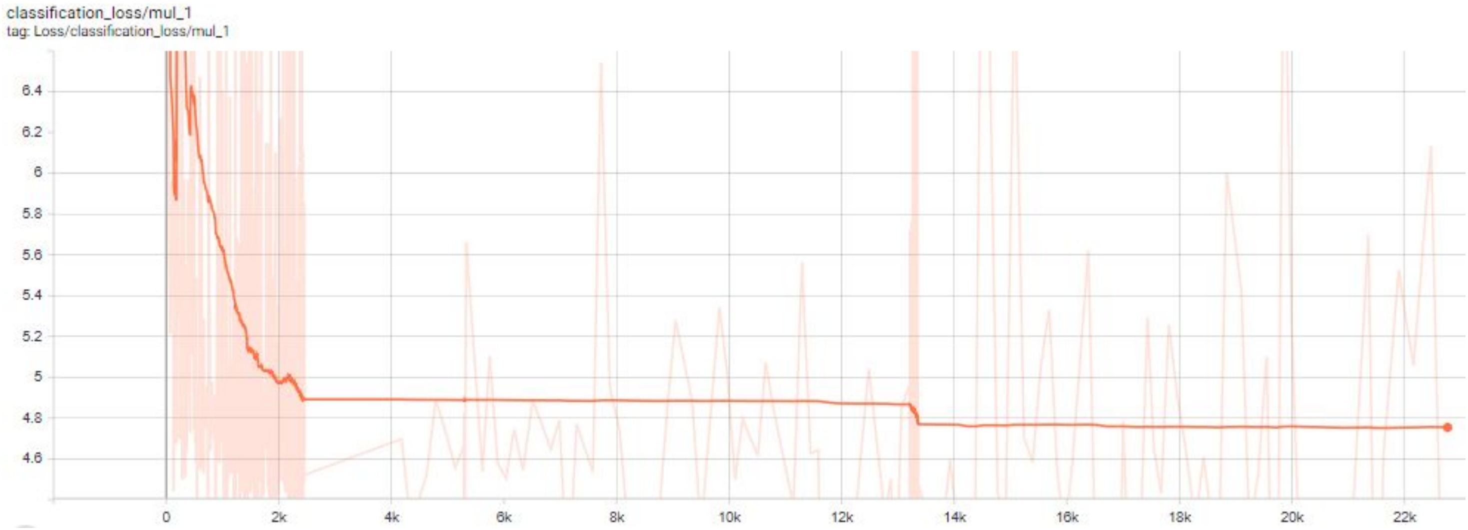
- Learning rate
  - Used 0.001, 0.004 and 0.00004
- Detection Threshold
  - From 0.5 to 0.4 and 0.3
- Image size
  - 1200*1080 to 600*600 and 400*400
- Activation function
  - Relu, sigmoid, tanh
- Batch Size
  - From 24 to 64, 128
- Number of steps
  - From 20000 to 15000

Result - Visualization



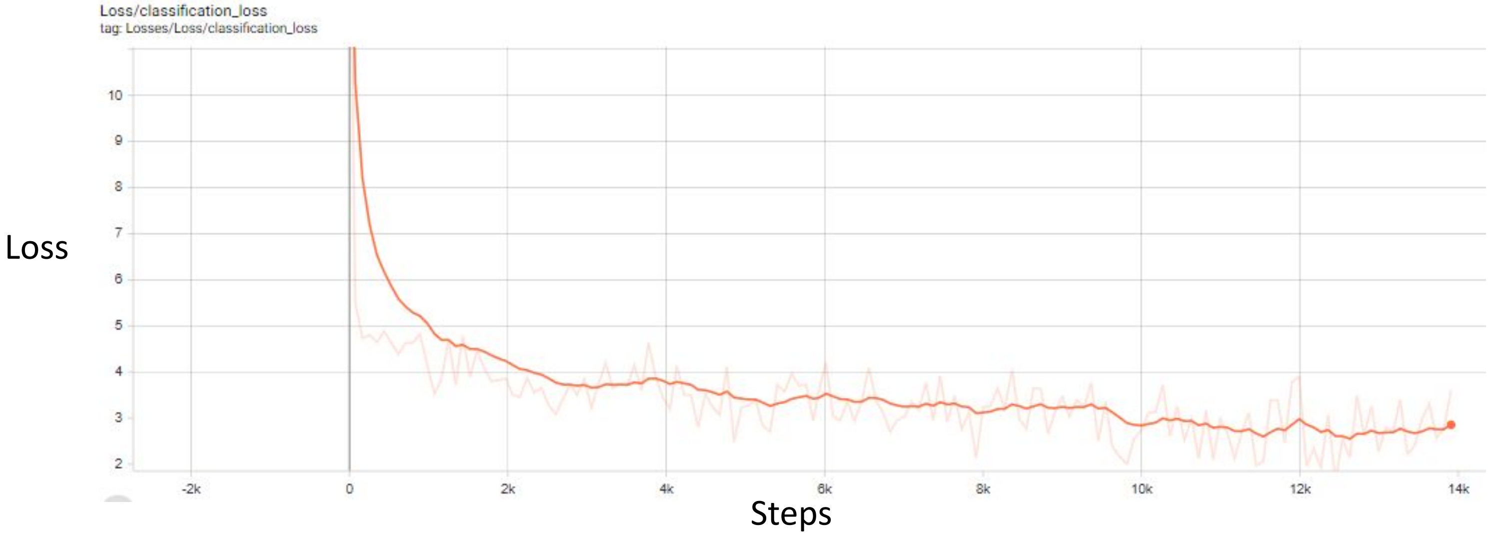
MobileNet V1 Loss Curve

Loss



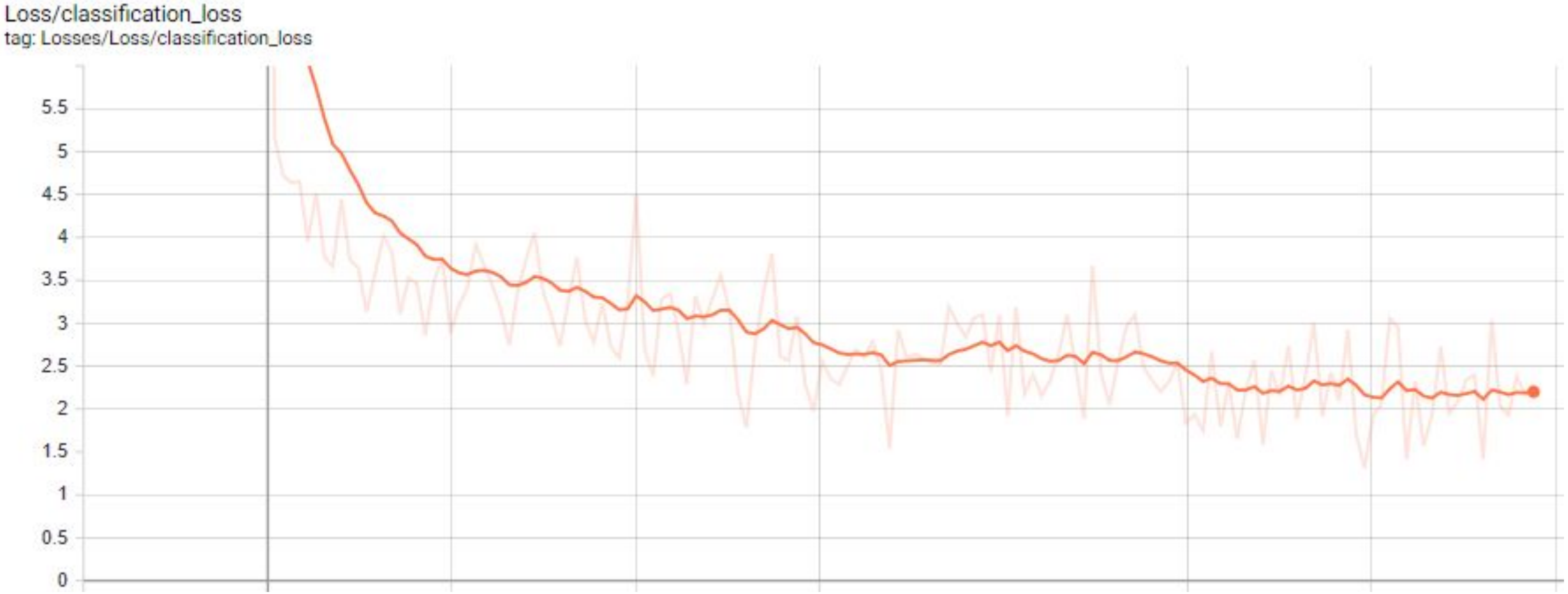
Steps

MobileNet v2 Loss Curve



SSD Inception V2 Loss Curve

Loss



Steps

14,000

Overfit (Inception) 20,000 steps vs 14,000 steps

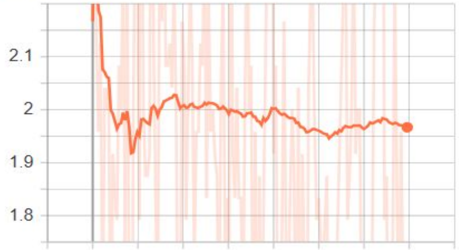
Test images (Inception Net)

Comparison of Results with baseline

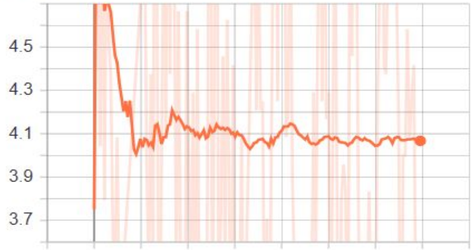
Model	Loss	Detection count ratio
MobileNet V1 (**baseline**)	4.754	0
MobileNet V2	3.125	45%
InceptionNet V1	2.25	55%

Model	Accuracy*
Baseline	10-20
InceptionNet V1	80-85

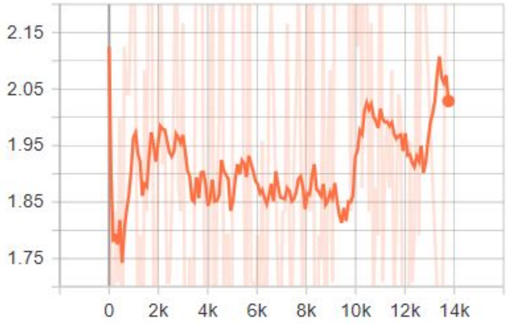
Loss/TargetAssignment
/AvgNumGroundtruthBoxesMatchedPerImage
tag: TargetAssignment/Loss/TargetAssignment
/AvgNumGroundtruthBoxesMatchedPerImage



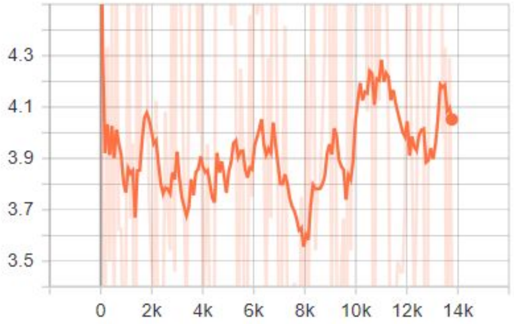
Loss/TargetAssignment
/AvgNumGroundtruthBoxesPerImage
tag: TargetAssignment/Loss/TargetAssignment
/AvgNumGroundtruthBoxesPerImage



Loss/TargetAssignment
/AvgNumGroundtruthBoxesMatchedPerImage
tag: TargetAssignment/Loss/TargetAssignment
/AvgNumGroundtruthBoxesMatchedPerImage



Loss/TargetAssignment
/AvgNumGroundtruthBoxesPerImage
tag: TargetAssignment/Loss/TargetAssignment
/AvgNumGroundtruthBoxesPerImage



*Of the key points detected, how accurate was it to ground truth

Conclusion

- Created a model to identify leaf tips and collars on images
- Other models used but didn't give results - SSD Resnet V2, Faster RCNN due to memory management issues in the server
- In the future, we plan to obtain better hardware to run more complex models for better detection.

REFERENCES

[1] Md Mehedi Hasan, Joshua P Chopin, H. Laga, and Stanley J. Miklav- cic. Detection and analysis of wheat spikes using convolutional neural networks. In Plant Methods, 2018..

[2] https://github.com/tensorflow/

[3] https://github.com/protocolbuffers/protobuf

[4] https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202

[5] https://medium.com/analytics-vidhya/image-classification-using-mobilenet-in-the-browser-b69f2f57abf

[6] https://pythonprogramming.net/introduction-use-tensorflow-object-detection-api-tutorial/

[7] https://towardsdatascience.com/custom-object-detection-using-tensorflow-from-scratch-e61da2e10087

Thank You