Plant Classification using Computer Vision

Chris Ernst

Abstract—With nearly 400,000 species of plants [1], accurate identification of different varieties remains a challenge for humans. Deep learning is applied to the problem of plant classification by training a network to classify four species of Sansevieria(commonly referred to as 'Snake Plant'). The network is trained on a GPU using GoogLeNet, and accuracy above 90 percent is achieved. The training set is comprised of 200 original images of each species, at varying angles of the plants, which are then augmented in various ways to generate 1,000 images for each class. The data is then split up to contain 75:25 (training:validation) images. The final model is then deployed to the NVIDIA Jetson TX2 for real-time inferencing.

Index Terms—Robotics, NVIDIA, IEEEtran, Udacity, LaTeX, deep learning.

1 Introduction

PLANT identification is a challenge to most humans, which creates an interesting problem to solve for machines. Many consumer applications have successfully been deployed for classifying certain plants, but with many varieties, additional research is necessary. While consumers desire a mobile application to use while exploring the natural world, the agricultural and houseplant industries are in need of an application to quickly classify and detect different varieties of plants. Once classification is achieved, detection is a next logical step for the tasks of harvesting, picking, and sorting. The initial task of classification is explored herein.

2 Background / Formulation

With the problem of plant classification clearly defined, the choice of a deep neural network is next. Instead of creating a new, potentially less effective network, the built and tested, GoogLeNet [2] is used. Training is accomplished using NVIDIA DIGITS. A relatively common initial learning rate of 0.01 is chosen, utilizing step down learning rate decay with gamma set to 0.1, and step size equal to 33%. (TABLE 1). Stochastic Gradient Descent(SGD) solver is selected. Initially, 10 epochs are used, but the network does not converge; 30 epochs are used in the final model. GoogLeNet expects input images of 256x256 RGB; the onboard camera of the NVIDIA Jetson is used to capture the training set.

TABLE 1
Learning Rate Schedule

Training Progress	Learning Rate
0-33%	0.01
34-66%	0.001
67-100%	0.0001

3 Data Acquisition

In order to collect an adequate training set, 200 RGB images of each of the four types of plants are captured locally. The

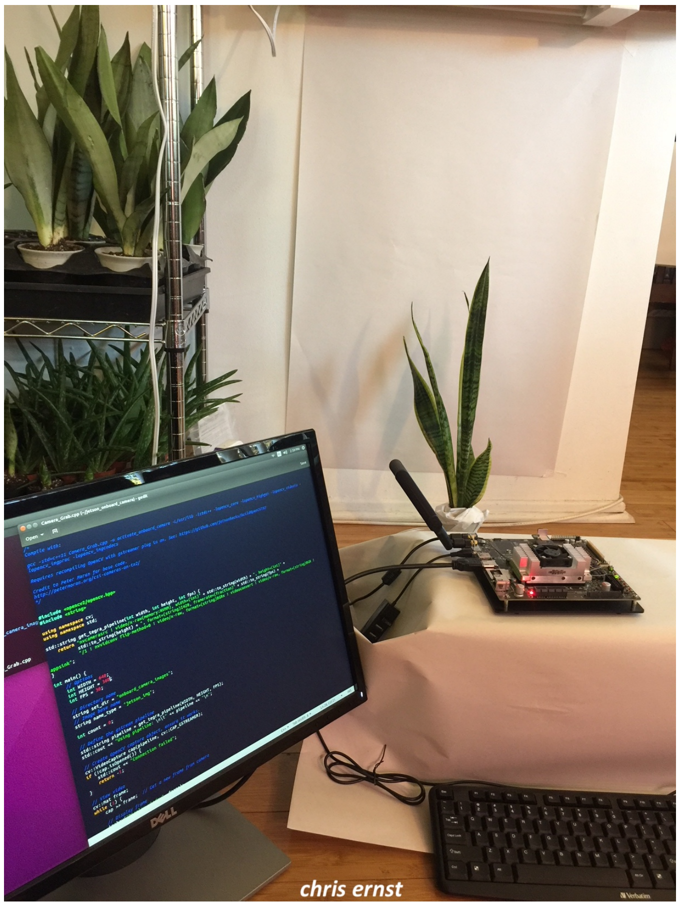


Fig. 1. Data Capture Setup

onboard camera of the NVIDIA Jetson TX2 is used for this purpose(Fig.1). The Jetson camera uses a 16:9 aspect ratio, so images are captured at 480 pixels wide, by 270 pixels in height, in order to keep storage requirements to a minimum, and remain above 256 pixels in both dimensions. Onboard capture is accomplished with a C++ helper script (thank you to Peter Moran for this). Each of the four varieties use multiple specimens to give the model depth and exposure to variations in growth patterns and foliage formation.

The locally captured data is then supplemented with images downloaded from Google to give the training set breadth, and exposure to 'imperfect' environments. A helper bash script from *Quora.com* was used to make this efficient, although a large amount of junk images are returned. The script can be found here.

3.1 Data Augmentation

With over 200 original images of each species, the next task is to augment those images to increase the training set by a factor of 5. After augmentation, each variety of Snake Plant will have 1,000 images each for the network to train on.

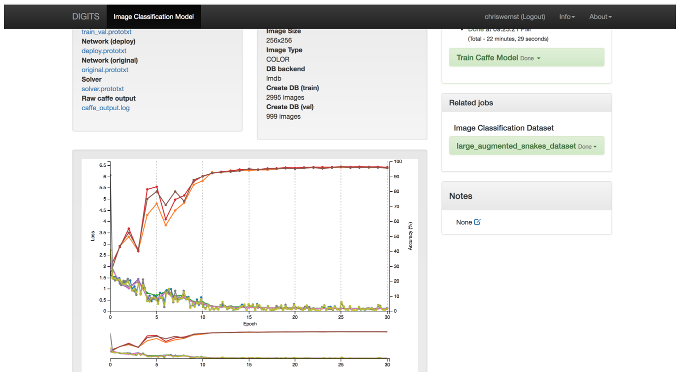
Although real-time augmentation is most desirable for speed and minimizing storage requirements, pre-processing is done using a Python library, Augmentor [3], which can be found here. Various forms of augmentation are used, including:

- Rotating
- Translating
- Shearing
- Mirroring/Flipping
- Cropping/Zooming/Resizing
- Skewing
- Tilting

Augmentation yields a more robust dataset that will give the network broader exposure to image variations during the task of inference.

4 Results

Of the four classes, the Jetson TX2 performs well in the task of classification with accuracy above 90 percent on all four classes. At certain angles, misclassification temporarily occurs, but if images are shown to the inference engine at similar perspectives as the training data, naturally classification is more accurate. As Figure 2 depicts, the network began to converge around 20 epochs, as the validation accuracy and training loss began to plateau.



train_val.prototxt
Network (deploy)
deploy.prototxt
Network (original)
original.prototxt
Solver
solver.prototxt
Raw caffe output
caffe_output.log

Image Size
256x256
Image Type
COLOR
DB backend
lmdb
Create DB (train)
2995 images
Create DB (val)
999 images

(Total - 22 minutes, 29 seconds)

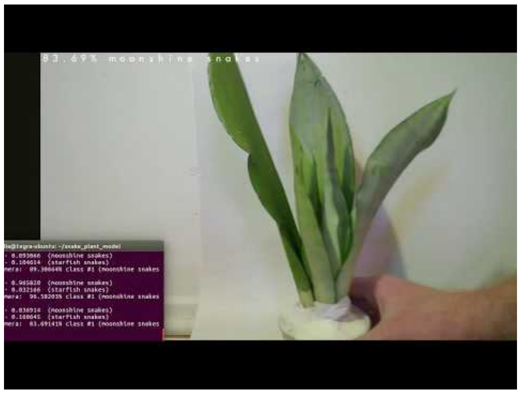
Image Classification Dataset
large_augmented_snakes_dataset Done

None

Fig. 2. Epoch 30 Training Progress

The trained model is then deployed to the NVIDIA Jetson TX2 to execute live, real-time classification of Snake Plants. A video of this testing can be seen at Figure 3.

Inferencing is accomplished in under 5ms on average, with greater than 90% certainty in each of the four classes.



```
83.69% moonshine snakes
```

```
ia@tegra-ubuntu: ~/snake_plant_model
- 0.893066  (moonshine snakes)
- 0.104614  (starfish snakes)
mera:  89.306648% class #1 (moonshine snakes

- 0.965820  (moonshine snakes)
- 0.032166  (starfish snakes)
mera:  96.582035% class #1 (moonshine snakes

- 0.836914  (moonshine snakes)
- 0.160645  (starfish snakes)
mera:  83.691418% class #1 (moonshine snakes
```

Fig. 3. Live Inferencing

5 Discussion

The results were surprisingly exceptional. Naturally, this was influenced by many variables. Predominately, it is assumed much of the accuracy comes from only having four categories of items to classify. Although, two of the Snake Plants are very similar in shape, and only differ in color (Moonshine vs. Yellow). Another variable that led to the success of this model was the originality of the training set, and the very different angles captures. Following original images, data augmentation likely played a large role. Giving the network skewed and noisy input data creates a much more robust model able to tolerate noise in the field. Due to plant identification not being a time-sensitive task, accuracy is more important in this specific use-case. Although, if latency was greater than 10ms, this could prove to be problematic in detection tasks.

6 Conclusion / Future work

The Snake Plant Classification Engine performed adequately in the environment it was trained for, accomplishing the purpose of this research. With just over an hour of training on a NVIDIA Tesla K40, the network learned very quickly. In its current state, this is not a commercially viable product, but at a later date, the network could be adapted to a detection network. A plant detector, paired with an end-effecting robotic arm could be a pick-and-place robot for the houseplant industry.

In the future, many improvements could be made to this project. Namely, by expanding the database of plants so the model is able to classify more than four varieties of houseplants. Data Augmentation should also be performed on-the-fly, in real-time to conserve memory. Andrew Janowczyk has a useful implementation of real-time augmentation performed using Python, and is compatible with NVIDIA DIGITS [4]. It would also be wise to capture a training set with diverse backgrounds so the model is more flexible and deployable in more noisy inferencing environments.

References

[1] Royal Botanic Gardens, Kew, United Kingdom, 2016.
[2] https://www.cs.unc.edu/ wliu/papers/GoogLeNet.pdf
[3] https://github.com/mdbloice/Augmentor
[4] http://www.andrewjanowczyk.com/real-time-data-augmentation-using-nvidia-digits-python-layer/