# A CapsNet Based Solution for Precision Agriculture

Sihan Chen
University of California, Berkeley
330 Yan'an Complex
sihan_chen@berkeley.edu

Dihan Yang
University of California, Berkeley
410 Yan'an Complex
yangdihan@berkeley.edu

## Abstract

*We created our own dataset on two species of plants and developed a vision-based classifier, which will enable future development in weed/crop detection algorithms and precision agricultural robotics. The classifier is based on recent researches on Capsule Networks (CapsNet) [6] and Residual Networks (ResNet) [3]. The effects of various improving methods on the neural net architectures and the image processing techniques are explored.*

## 1. Introduction

Due to an unstable political climate regarding immigration, it is becoming increasingly difficult for American farming communities in the Central Valley to find traditional seasonal laborers to tend crops and harvest them [2]. Without skilled labor, vegetable producers can lose the majority of their crop due to pests or improperly tended plants. Recent progresses in Deep Learning and Computer Vision have the potential to alleviate some of the tasks that are hard to staff, *e.g*. applying herbicide and irrigating. By using automated technology for weed control, farmers will be able to reduce herbicide usage, and optimize scarce labor resources towards picking and cutting operations by leaving more mundane tasks to robots.

Our project is sponsored by a farm in Bakersfield California. One of the major crops there is the Chinese Broccoli (also known as Gai-Lan), and the most prevalent weed in the plots is Nut Grass (as shown in Figure 1). Our scope for this project is limited to developing a vision-based binary classifier that identifies Chinese Broccoli from Nut Grass. The farm provided us aerial-view images from its Chinese Broccoli plot taken in summer 2018, based upon which we chose our approaches.

Because of the uniqueness of our dataset and the limitation in the time span of this project, we decided to build our classifier via deep learning models, since they have shown powerful adaptivity in recent researches. We examined the popular deep learning frameworks, and deemed the CapsNet by Hinton *et al*. [6] and the related modifications discussed by Xi *et al*. [7] are feasible approaches given the nature of our objectives and our data. We also examined the possible improvements by integrating the ResNet introduced by He *et al*. [3].

Figure 1. Appearance of the Chinese Broccoli (left) and the Nut Grass (right). Image was taken at the Bakersfield Farm

## 2. Related Work

In paper *Dynamic Routing Between Capsules*, Hinton *et al*. [6] discussed the limitation of traditional Convolutional Neural Networks (CNN), that is the insensitivity towards orientational relationships and relative spatial hierarchies among feature components, as well as the incapability of recognizing same objects from different viewpoint [5].

Consequently, they developed the concept of Capsule Networks as a promising solution. Capsule Networks use Capsule Layers that numerically incorporates the relative relationships between objects into a 4D pose matrix. Between the Capsule Layers, the state of the features are encapsulated as probability vectors, instead of scalar values that are usually used in CNN. The major differences between Capsules and traditional neurons are summarized in Figure 2. They claimed that their CapsNet model achieved a training accuracy of 89.4% on CIFAR10 dataset using 7 ensembles.

In his another paper *Matrix Capsules With EM Routing*, Hinton *et al*. [4] argued that CapsNet is able to reduce the number of errors by 45% on the smallNORB dataset and no worse performance on MNIST or CIFAR10 dataset compared to a state-of-the-art CNN, while using only a fraction

| Capsule vs. Traditional Neuron | | | |
|---|---|---|---|
| Input from low-level capsule/neuron | | vector($\mathbf{u}_i$) | scalar($x_i$) |
| Operation | Affine Transform | $\widehat{\mathbf{u}}_{j\|i} = \mathbf{W}_{ij}\mathbf{u}_i$ | – |
| | Weighting | $\mathbf{s}_j = \sum_i c_{ij}\widehat{\mathbf{u}}_{j\|i}$ | $a_j = \sum_i w_i x_i + b$ |
| | Sum | | |
| | Nonlinear Activation | $\mathbf{v}_j = \frac{\|\mathbf{s}_j\|^2}{1+\|\mathbf{s}_j\|^2}\frac{\mathbf{s}_j}{\|\mathbf{s}_j\|}$ | $h_j = f(a_j)$ |
| Output | | vector($\mathbf{v}_j$) | scalar($h_j$) |

Figure 2. Important differences between capsules and neurons. [5]

of the data.

In *Capsule Network Performance on Complex Data*, Edgar *et al*. [7] suggested several improving methods toward the original CapsNet. They claimed the best observation was a validation accuracy of 71.550% over 50 epochs using a 4 ensemble models with 2 convolution layers.

In *Deep Residual Learning for Image Recognition*, He *et al*. [3] showed the effectiveness of bottleneck and shortcut architecture for preventing degradation as well as speeding up the training. We deemed these properties to be helpful when the CapsNet models become deeper after applying the improving methods.

# 3. Problem Statement

Our objective is to develop a deep learning based classifier for our weed-crop dataset. Hence this problem has two major counterparts: dataset preparation and deep net architecture design.

## 3.1. Data Preparation

Since there's no existing dataset for our problem, we have to prepare our own. The raw data is 170 images taken from a Chinese Broccoli plot in the Bakersfield farm. Human labellers are required for the ground truth. The resulting image-label pairs are then preprocessed and packed in batches as input for our deep net.

## 3.2. Deep Net Binary Classifier

A traditional CNN architecture is not sufficient for our problem because of its incapability of viewpoint invariance and insensitivity towards spatial hierarchies [5]. In fact, viewpoint invariance and spatial hierarchy are both critical for our problem.

### 3.2.1 Viewpoint Invariant

Unlike other popular datasets *e.g*. MNIST and CIFAR10 whose images are all upright, the input images for our classifier are taken from aerial views, *i.e*. the gravity directions

Figure 3. This image contains considerable features of Nut Grass's (the sharp leaves), however it should be classified as a Chinese Broccoli which is rooted in the center of the frame.

in our input images is no longer pointing toward negative vertical axis of the image plane, but random horizontal directions instead. Yet our classifier has to produce inferences that are invariant of viewpoints.

### 3.2.2 Spatial Hierarchy

The weeds often grow closely by the crops and sometimes obscure one another in the same frame. In fact, the appearance of certain features of weed does not necessarily indicate that the input image should be classified as weed. Figure 3 shows an example. Therefore the spatial hierarchy relationships between objects have to be taken into account.

### 3.2.3 Necessity of CapsNet

For these reasons, we favored the CapsNet over the plain CNN frameworks as Hinton *et al*. proved the CapsNet's superiority in viewpoint invariance and spatial hierarchies related tasks. Another motivation for us to choose CapsNet is the data size. Hinton *et al*. claimed a competitive performance of CapsNet with only a fraction of the data that a CNN would use. [5]. On the other hand, we eventually obtained 3000 image-label pairs from the raw data, which is indeed a small size comparing to those of CIFAR10 or MNIST.

### 3.2.4 Other Improving Approaches

We also decided to improve our CapsNet architectures with the methods suggested by Edgar *et al*. in [7]. We selected 4 of the methods that have demonstrated stable improvements for our CapsNet, *i.e*. Increasing the number of primary capsules; Ensemble averaging; Modify scaling of reconstruction loss; Increase number of convolution layers before capsule layers. The last one increases the depth of our neural net considerably, and ResNet in this case becomes a natural choice in order to avoid degradation as well as speed up training process.

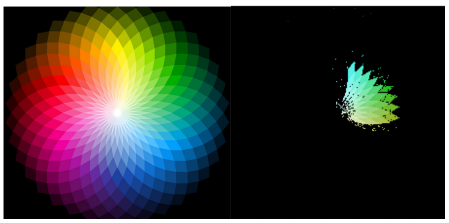

Figure 4. Green Mask Range

# 4. Technical Approaches

## 4.1. Data Preparation

### 4.1.1 Raw Data and Ground Truth

The raw data are taken by a SONY SLT-A37 camera with a 20-millimeter focal length and a 1/250-second exposure time. We used the free service (community account) of Labelbox, a cloud-based labelling platform, to create ground truth labels on the raw images. The labels and corresponding pixels from the images are then exported into our dataloader in pairs. 4 volunteers and 2 reviewers helped generate nearly 3000 image-label pairs from the raw images in the end.

### 4.1.2 Data Prefiltering

The shapes of the plants' leaves are considered to be the most essential features for classification in the by expert opinion (the farm manager). However, the warm tone in the original images makes the leaves' edges obscure in the brownish background soil, whose textures also serve as a huge noise source. Therefore we determined prefilter the images.

We first used the Non-local Means Denoising algorithm to remove the white noise [1] that appeared mostly in the background soil. Then we used a green color mask to remove all the wavelengths outside the range of green. The color range is shown in Figure 4. In the end, the image section is normalized to a size of 32 pixels by 32 pixels. Figure 5 visualizes the 6 steps of the pipeline.

## 4.2. Capsule Networks

The CapsNet has two parts: encoder and decoder. For the original CapsNet [6], the first 3 layers serve as the encoder; the second 3 layers serve as the decoder. We further expanded this architecture to another 3 configurations: a 3 convolutional encoder and deconvolutional decoder model; a 4 convolutional encoder and deconvolutional decoder model; a 7 convolutional encoder with residual connections and deconvolutional decoder model. Each of them are tested with none, 4 and 7 ensembles. The results from several of the most representative cases can be found in **Results** section.

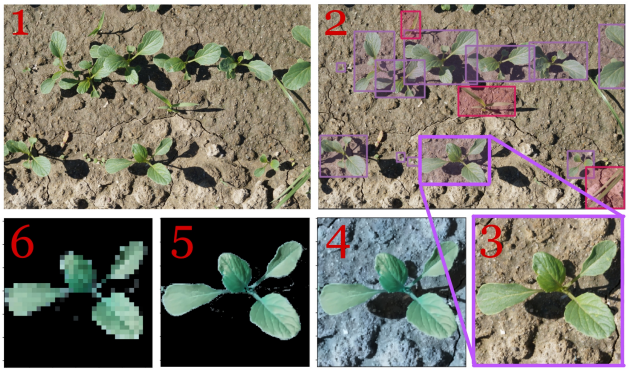


Figure 5. Data preparation pipeline: (1) raw image, (2) data labeling, (3) crop out labelled section, (4) denoising, (5) green mask, (6) down sampling

### 4.2.1 Encoder

We start with Hinton's MNIST model with one Convolutional layer, one PrimaryCaps layer and one DigitCaps layer as out baseline model. It takes a $3 \times 32 \times 32$ input size. Building upon this, we tried different improving methods.

First we tried to add several convolutional layers before the PrimaryCaps layer, gradually decrease the output size in each channels and increase the number of channels between layers. We have tried 3 and 4 convolutional layers separately, and found the latter works better. In order to increase the number of layers before the PrimaryCaps layer without introducing parameters excessively for the sake of higher accuracy, we tried ResNet in the encoder..

We followed Hinton *et al.*'s original model [6] to have 32 capsules in the PrimaryCaps layer. Each capsule takes the output of the convolutional layers as the input and produces a 3-dimensional tensor. Since there are 32 capsules totally, we end up with 32 three-dimensional blocks after the PrimaryCaps layer. We then tried to increase the number of capsules to 64, which does improves the accuracy.

The DigitCaps layer takes the several 3-dimensional blocks as input and uses 3 dynamic routing to produce $m$ capsules, each representing a 16-dimensional vector because each vector embeds one class. Finally, we take L2 norm of each vector and use a Softmax to produce the final probability of each class. In our cases, we only have two classes, so $m = 2$.

### 4.2.2 Capsule Loss

The loss function for CapsNet is actually similar to that of SVM. For each class, we have $L_c = T_c \max(0, m^+ - ||v_c||)^2 + \lambda(1 - T_c) \max(0, ||v_c|| - m^-)^2$. For the correct label, $T_c$ will be 1 and for the other labels, $T_c$ will be zero.

$m^+$ and $m^-$ are two thresholds. For the correct label, the larger $||v_c||$ is, the smaller the loss function is; For other labels, the smaller $||v_c||$ is, the smaller the loss function is. We took the sum of the loss function in each classes for our entire encoder loss.

Here, we fixed the coefficient $\lambda$ to be 0.5, which showed to have the most numerical stability during training. For the correct labels, we hope our prediction to be larger than 0.9; for other labels, we hope the prediction should be smaller than 0.1. Hence we fixed $m^+$ to be 0.9 and $m^-$ to be 0.1.

#### 4.2.3 Decoder

The decoder in Hinton *et al.*'s original model is composed of three fully connected layers. It takes the correct DigitCap vector during training and ignores the others as input and learns to recreate a $3 \times 32 \times 32$ pixels image. In a sense the decoder is actually used as a regularization with the loss function being the Euclidean distance between the reconstructed image and the input image. We set the regularization coefficient to be 0.0001. Decoder forces the network to learn the features which are useful to reconstruct the original image. The closer the reconstructed image to the input image, the better the classification is.

Here, we used deconvolutional layers instead of the original fully connected layers in the decoder since it is the inverse process to the convolutional layers. We deem this to be a better reconstruction approach.

### 4.3. Number of Parameters

#### 4.3.1 Parameters in Encoders

Our encoder uses stacked convolutional layers. In the first convolutional layer for the baseline model each filter has a size of $9 \times 9$. It transfers 3 channels into 256 channels, *i.e.* it has $(9 \times 9 + 1) \times 256 \times 3 = 62976$ trainable parameters.

For the model stacked with 3 convolutional layers, all filters are $3 \times 3$. The first layer has 64 channels; the second one has 128 channels; the final one has 256 channels. Hence the total number of parameters is $(3 \times 64 + 64 \times 128 + 128 \times 256) \times (3 \times 3 + 1) = 411520$.

Similarly the model stacked with 4 convolutional layers has 32 channels with filter size of $3 \times 3$, and the total number is $(3 \times 32 + 32 \times 64 + 64 \times 128 + 128 \times 256) \times (3 \times 3 + 1) = 431040$.

For the model combined with 7-layers ResNet, we have 3 ResNet blocks contains of two convolutional layers with filters with size of $3 \times 3$ followed by a convolutional layer with filters with size of $7 \times 7$, and total number is 2762112.

#### 4.3.2 Parameters in PrimaryCaps Layer

Each capsule of the PrimaryCaps layer has 8 channels of filters with size of $9 \times 9$. There are $(9 \times 9 + 1) \times 256 \times 8 = 167936$ trainable parameters in each capsule. Hence for the entire 32 capsules there are 5373952 parameters.

The ResNet's convolutional layers has 64 capsules, each with output size of 512 channels. Hence there are overall $(9 \times 9 + 1) \times 512 \times 8 \times 64 = 21495808$.

#### 4.3.3 Parameters in DigitCaps Layer

In the DigitCaps layer of baseline model, we have $32 \times 8 \times 8$ 8-dimensional vectors, *i.e.* 2048 input vectors in total. Each input vector has its own $16 \times 8$ weight matrix mapping a 8-dimensional input space to a 16-dimensional capsule output space. So there are 2048 matrices for each capsules, and another 2048 c-coefficients and 1152 b-coefficients used in the dynamic routing. So totally we have $2048 \times 16 \times 8 + 2048 + 2048 = 266240$ trainable parameters in each capsules. Multiplying this number by 2 classes we obtain 532480 parameters.

For the model stacked with 3 convolutional layers, the input size is $32 \times 3 \times 3$ and the parameter number is $32 \times 3 \times (16 \times 8 + 2) \times 2 = 74880$. Similarly for the ResNet the number is 149760.

#### 4.3.4 Parameters in Decoder

In the decoder, our baseline model have one fully connected layers which transfer $16 \times 2$ tensors into $32 \times 8 \times 8$ tensors. Hence it has $16 \times 2 \times 32 \times 8 \times 8 = 65536$ parameters.

Our 4 convolutional-deconvolutional model has all with filters with size of $3 \times 3$; the first layer has 32 channels, the second has 16, the third has 8, the fourth has 4 and the last one has 3, so the total number of parameters is $(32 \times 32 + 32 \times 16 + 16 \times 8 + 8 \times 4 + 4 \times 3) \times 3 \times 3 = 15372$.

### 4.4. Optimization and Error Metric

Considering the stability throughout training, we used the momentum Stochastic Gradient Descent (SGD) method to optimize the loss function with a learning rate of 0.001, a momentum of 0.9 and a weight decay of $5 \times 10^{-4}$. We took the accuracy of our prediction as the error metric.

### 4.5. Ensemble Methods

Since the loss function of the networks have considerable local optimum, the results of the final variables vary constantly. We ensemble several results of the same network to compute the average as our final results. We tried 4 and 7 ensembles of same networks together and resulting with large improvements in our final results.

Figure 6 visualizes the architecture of our "7 residual convolutional encoder-Capsule Layer -deconvolutional decoder" model pipeline.

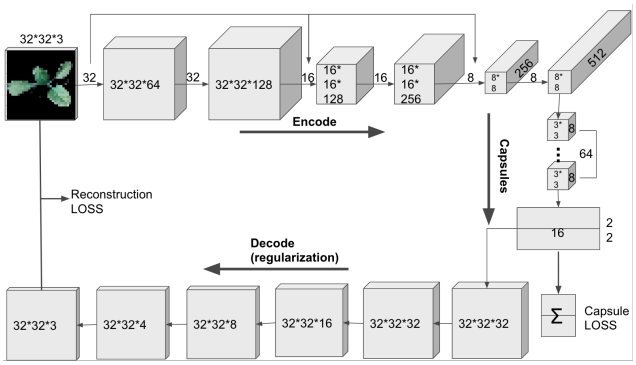


Figure 6. 7 residual convolutional layers as the encoder, Hinton's Capsule Layers, deconvolutional layers as decoder

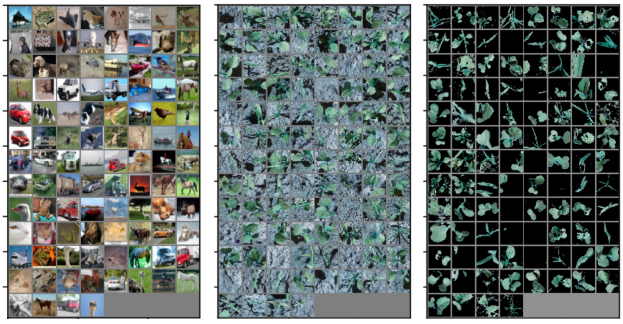

Figure 7. Sample Batches of CIFAR10 (left), non-prefiltered weed-crop (middle), prefiltered weed-crop (right)

## 5. Experiments and Results

### 5.1. Training

We trained each model described in section **4.2** with 3 different input datasets: the CIFAR10 dataset (as reference), the non-prefiltered weed-crop dataset and the prefiltered weed-crop dataset. We chose batch sizes to be 100. On the 10-class CIFAR10 dataset we trained 20 epoches; On the binary weed-crop dataset we trained 10 epoches. The training sessions were conducted on Google Colab with runtime configured as Python3 and GPU. Figure 7 shows a sample batch of these 3 datasets. The results from 7 of the most representative models are demonstrated is the **Results** section.

### 5.2. Results

Figure 8 shows the training and testing accuracy for 20 different cases. Each subplot shows the evolution of training and testing accuracy using the model corresponding to that row and the dataset corresponding to that column over certain epoches.

The 7 rows represents 7 of the different models we have experimented: The first row is the baseline CapsNet without any ensemble; The second row is the baseline CapsNet with 4 ensembles; The third row is the CapsNet with 3 convolutional encoders and deconvolutional decoders as the reconstruction method with 4 ensembles; The fourth row is the CapsNet with 4 convolutional encoders and deconvolutional decoders with 4 ensembles; The fifth row is the CapsNet with 4 convolutional encoders and deconvolutional decoders with 7 ensembles; The sixth row is the CapsNet with 7 residual convolutional encoders with 7 ensembles; The seventh row is the 7-layer ResNet without any ensemble. The larger the row number, the more complexity (more hyper-parameters) of the corresponding model.

The 3 columns represents 3 datasets we input to train our models. The first column is the CIFAR10 dataset; The second column is the weed-crop dataset without prefiltering; The third column is the weed-crop dataset with prefiltering. Note that we didn't train the ResNet model on CIFAR10.

Detailed descriptions for these model architectures and filter pipelines can be found in section **Technical Approaches**.

We observed that both training and testing accuracy increases as the model becomes more complex for the CIFAR10 and prefiltered weed-crop dataset, whereas the non-prefiltered one does not show such clear positive correlation. The best testing accuracy achieved by the prefiltered weed-crop dataset is higher than that of the non-prefiltered one, and the training accuracy of the prefiltered weed-crop dataset increases more fiercely as the model becomes more complex than the other two do. Overall the weed-crop binary classification achieves higher accuracy within 10 epoches than the CIFAR10 classification does within 20 epoches. It is noteworthy that the training accuracy for prefiltered weed-crop dataset actually stay lower than it testing accuracy for the first 3 models.

On the other hand, the introduction of the ResNet architectures indeed boosts the performance. In fact, a simple ResNet structure also achieves comparable performance in the end, although the CapsNet+ResNet model converges within fewer epoches and performs more stably. ResNet architectures in our experiments are not able to resolve the overfitting problems yet.

The training loss for the model-dataset combinations is visualized in Figure 9. Note that the losses for ResNets are not included because they are not computed by the same Capsule Loss function as other models do. The evolution of losses agrees with the accuracy performances, in that the more complex models lead to smaller losses. Note that the prefiltering of the data leads to significant decrements in the overall losses.

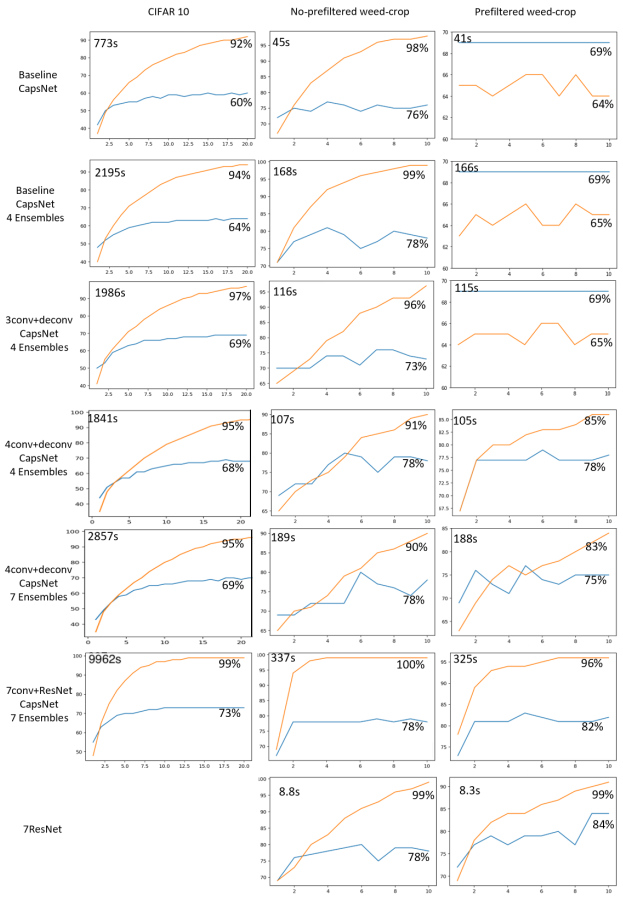


Figure 8. Performance of 20 cases: 7 models and 3 input datasets. For each subplot: The vertical axis represents the percentage of accurate outputs; The horizontal axis represents the epoch numbers; The orange curve represents the training accuracy; The blue curve represents the testing accuracy. The final accuracy values are shown at the end of the curves. The time consumption (in seconds) of each training session is shown at the top-left corner

### 5.3. Time Consumption

On each model, training 10 epoches on the prefiltered dataset is slightly faster than that of the non-prefiltered dataset, whereas the CIFAR10 dataset takes nearly 10 times longer. For each input dataset, more ensembles take longer training time; the increment in convolutional layer depth does not necessarily increase the training time. A simple ResNet without Capsule Layers or ensembles takes way shorter time (approximately 40 times shorter). Another noteworthy point is that the prefiltering process in data preparation actually makes the dataset 3 times slower to load, mainly due to the Non-Local Means Denoising applied before green mask and down sampling.

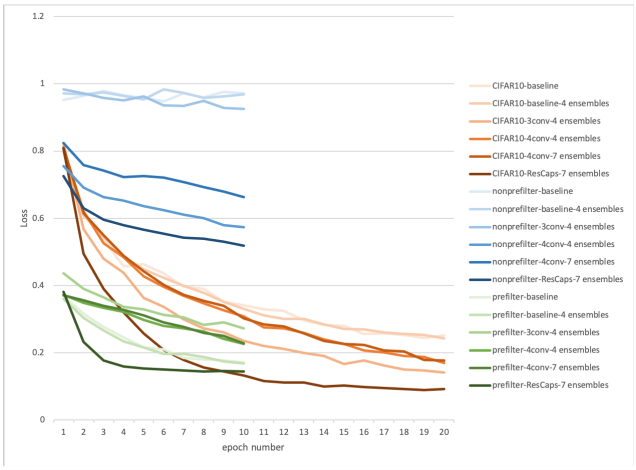


Figure 9. Loss of 18 cases: 6 models and 3 input datasets. Darker lines represent more complex models

## 6. Discussion

The results demonstrated in the **Results** section show that, by using 4 of the 7 improving methods suggested by Edgar *et al*. [7], our modified CapsNet architectures are able to achieve better performance than the 71.55% accuracy claimed by Edgar *et al*. [7]; however, we are not able to achieve the 89.4% accuracy benchmark claimed by Hinton *et al*. in [6].

ResNet architecture is indeed a powerful algorithm to enhance the accuracy. In fact, we have to admit that ResNet is the most cost-effective method for our binary classification task on weed-crop dataset, amongst all the models we have experimented with.

Using denoising and color-mask filters for data preprocessing can enhance the deep net performance significantly. However, a faster denoising algorithm is desired.

## 7. Conclusion and Future Work

### 7.1. Conclusion

This project delivers a comprehensive pipeline to develop CapsNet and ResNet based deep neural network models for a binary plants classification task. Despite overall more stable performance of the CapsNet+ResNet model, a simple ResNet model will be the more cost-effective choice. The aid of traditional image processing techniques can also enhance the performance of deep learning model significantly.

### 7.2. Potential Improvements

For the prefiltering process: more reviewers to scrutinize the labels; a faster denoising algorithm is desired; a normalization based on image white balance for the green mask will be more adaptive.

For the deep net model: Different loss function can be experimented. *e.g*. cross-entropy is another possible option for capsule loss; MAE can be a substitution to the MSE used in this project for the reconstruction loss (regularizer). In addition, more Capsule Layers could be constructed given more computational resources.

### 7.3. Possible Extensions

The goal of this project is the eventual application in precision agricultural robotics. Therefore, the classification model should eventually be embedded as a counterpart of an object detection model such as YOLO. The inference process should be optimized so that it can perform realtime onboard inference. In addition, the classifier could be trained with more species to make this application scalable.

## References


[1] A. Buades, B. Coll, and J.-M. Morel. Non-Local Means Denoising. *Image Processing On Line*, 1:208–212, 2011. 3

[2] C. Dickerson and J. Medina. California farms backed trump but now fear losing field workers. 2017. 1

[3] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 1, 2

[4] G. Hinton, S. Sabour, and N. Frosst. Matrix capsules with em routing. 2018. 1

[5] M. Pechyonkin. Understanding hintons capsule networks. 2017. 1, 2

[6] S. Sabour, N. Frosst, and G. E. Hinton. Dynamic routing between capsules. *CoRR*, abs/1710.09829, 2017. 1, 3, 6

[7] E. Xi, S. Bing, and Y. Jin. Capsule Network Performance on Complex Data. *arXiv e-prints*, page arXiv:1712.03480, Dec 2017. 1, 2, 6