DESIGN DOCUMENT

DECEMBER 2, 2017

CS461 FALL 2017

REAL-TIME SEED IDENTIFICATION

PREPARED FOR

OREGON STATE UNIVERSITY CROP SCIENCE DEPARTMENT

DANIEL CURRY

PREPARED BY OMAR ELGEBALY

GROUP 11

GREEN TEAM

OMAR ELGEBALY

XIAOYI YANG

VINAYAKA THOMPSON

Abstract

The purpose of this document is to get an in depth idea for the architecture and design decisions regarding this project. We will discuss our intended audience, the design of each stage, and justify our design decisions. We will also discuss how we plan to use different technologies to implement our project.

CONTENTS

1 Overview

1.1 Scope

We will discuss the development of the core design elements of the Seed Classifier. It will encompass the relationship between components, specific technologies, and design viewpoints for each.

1.2 Purpose

To specify the content and organization of the core design elements of this project. It will include implementation methods, descriptions, and design viewpoints for each.

1.3 Intended Audience

Intended for developers, clients, and users of the seed classifier. It will guide future designers in providing new functionality to expand upon this project.

1.4 Conformance

The document conforms with client specifications and requirements provided to the design team.

1.5 Required Elements

- Nvidia Jettson TX2 Processor: Used for training and testing.
- Vibrating Table: Used to automate the process of transporting seeds
- Camera: Used to capture images of seeds
- Light Source: Used to provide better lighting for the seeds

2 Definitions

Off-type: Non desired seed

Tall Fescue: This is our target seed that we want in the batch. Any seed that is not Tall Fescue should be filtered out.

Learning: the process of training our classifier on images

Preprocessing: series of transformations data must go through in order to be optimally recognized by the classifier

Classifier: The software that is used to determine if a seed is an off-type or Tall Fescue

Top-1 Accuracy: The Top-1 accuracy indicates the probability of the classifiers top guess for the image being correct [3].

Untrained Data: Any images that have not undergone the training phase.

3 Overview of System

Our product seeks to streamline the arduous process of sorting off-type seeds from a batch of Tall Fescue seeds. As of now, seed analysts in the OSU Crop Science Lab will differentiate between them using a magnified lens setup and remove them by hand. Our goal is to create a program utilizing computer vision that will identify if a seed is not Tall-Fescue. It will then send a false signal to the vibrating table setup that the mechanical engineering group is working on.

Our system will be based off of a vibrating table setup. Attached to the vibrating table is a funnel, suspended camera, light source, and mechanical dividers. The seeds are inserted through the funnel onto the vibrating table. The camera is connected with Jettson TX2 The camera will be set to take a picture on second intervals. As the seeds pass through, the camera will be constantly feeding images to the computer. We will implement a script that will test the trained model against each batch of images. If the Top-1 accuracy is less than 80%, then there is an off-type in the batch. In that case, we will send a signal from the Jettson to the table indicating that it should trash those seeds. As mentioned before, we are only responsible for taking the pictures, testing them against our trained model, and sending a signal if the classifier is not at least 80% confident that all of the seeds in the batch are Tall Fescue.

4 System Architecture

4.1 Design

This section discusses the conceptual model for the project. The conceptual model will act as a guide for future developers and a high-level overview for the client.

Although we are only responsible for the software of this project, we are responsible for making it work with the hardware we are given. The seeds will be dropped onto the vibrating table via a funnel and will move through the system. The camera will capture images and run the trained model's testing script against it in real time. If the Top-1 accuracy for a captured image is less than 80%, we will send a signal to the system indicating that the seeds in that image should be discarded.

4.2 Decomposition

There is a natural, sequential decomposition between parts in this system. The system has four stages: data collection data preprocessing, data training, and data testing. In the data collection phase, the camera's sole job is to capture images. These images undergo transformations in scale in the preprocessing stage before being trained on our model. The model will train on the dataset of images. Once the model is finished training, we can test it's functionality by building the model and running the test script on untrained data.

4.3 Design Rationale

The goal of this project is to introduce a more efficient way of sorting through seed to aid the OSU Crop Science Department. The nature of this project makes it difficult to create a modular design as each stage relies on the previous one. We believe that this architecture is optimal as it follows the natural progression for any machine learning project. Splitting the project into four phases helps because it can help us more easily identify what stage we need to work on to identify bugs. For example, let us assume we have gotten to the testing stage but our classifier is giving us extremely low Top-1 percentages. This indicates the issue is likely in how we preprocessed our data, which means we would have to look into transform our data further and retraining it.

5 Design Stages

5.1 Introduction

This section will talk about the different design phases involved in this project and explain the design rationale behind each stage. This project has been split into four distinct stages: data collection, data preprocessing, data training, and

data testing. Vinayaka Thompson will be responsible for data collection. Omar Elgebaly will be responsible for data preprocessing and data training, since these two are heavily intertwined. Xiaoyi will be responsible for data testing.

5.2 Data Collection

In this stage, we are collecting our dataset of Tall Fescue seed images. We need approximately 10,000 images to start with.

We are using a setup provided to us by the mechanical engineering group we are collaborating with. They have setup a vibrating table with multiple attachments to help automate the process of data collection and testing. There is a funnel attachment, which allows you to direct the seeds onto the vibrating table. Hanging above the vibrating table, we have a suspended camera that will be set to capture images on one second intervals. There will also be a light source to illuminate the seeds.

Our strategy for tackling this problem is to take the seeds and place them into the funnel. The funnel will transport the seeds onto the vibrating table. The camera is set to capture an image every second. With the increased visibility provided by the light source, we can reduce motion blur by reducing the shutter speed of the camera. Once we have a dataset of 10,000 images, we will move on to the data preprocessing step.

We estimate that this step will take 4-5 days. We ideally want to start using this setup by January 15th but we cannot realistically expect the mechanical engineering group to be done so quickly. Since we cannot afford to wait on them too much, we plan on manually placing seeds on a conveyor belt setup that was developed by a group a few years ago. This process may be slower but we cannot proceed without gathering data.

5.3 Data Preprocessing

In this phase, we will need to preprocess the data we have collected. Learning algorithms generally benefit from standardization of the dataset as many of them assume the data is normally distributed, which is not necessarily the case.

We will use TensorFlow library APIs to preprocess the data in a way the classifier can recognize. We may use scikit-learn for additional image transformation if our classifier behaves incorrectly.

Since we have opted to use TensorFlow as a learning framework for now, we can use TensorFlow's built-in functions for image feature scaling and normalization. TensorFlow has an API (tf.image.per_image_standardization) that will linearly scale images to have zero mean and unit variance [1]. It is important we know the model we plan to train on and its limitations. Our chosen model, inception_resnet_v2, requires our input images to have color channels from [-1,1]. The default color channel is normally from [0, 255]. The problem with using the default is that the input is much larger than the network expects and will therefore be biased towards certain parts of the image with more weight, which can cause mispredictions.

It is important to know the format of the data that the learning algorithm prefers. In this case, reading data with a TF-Slim based model has two components: dataset descriptor and dataset reader. Since we are using a TF-Slim based model, the data should be converted to TFRecord format. We can convert the data into TFRecord using TensorFlow's dataset API.

We plan to commence image preprocessing as soon as all the data has been collected. Our estimated date of completion is January 25th.

5.4 Data Training

Once the data has been collected and gone through the necessary preparation to be trained in the classifier, we can commence the data training stage.

For this phase, our chosen processor will be the Nvidia Jettson TX2. It is a chip with both a GPU and a CPU that is designed specifically for AI computing, which makes it an ideal technology for this phase. If this processor proves to be not powerful enough, we may have to look into other platforms such as Amazon Cloud Services. This type of processor works best on a Linux operating system, so we plan to use Ubuntu. As for software, we will use TensorFlow as a training framework. The learning model we will use is inception_resnet_v2, which is based on the TensorFlow-Slim image classification library. The TensorFlow-Slim (TF-Slim) image classification library is a high-level lightweight API of TensorFlow for defining, training, and evaluating complex models [2]. If TensorFlow proves to be too slow, we will implement this model using OpenCV's library instead.

In this phase, the preprocessed image data will be fed into a classifier that will learn the data. As of now, our preliminary goal is to accumulate a dataset of 10,000 images. The model of classifier we plan on using is Google's Inception-ResNet-v2 architecture. It is a state of the art learning model that is based on a deep convolutional neural network. We plan to use this model with the TensorFlow framework as it integrates most smoothely . One of the advantages of using this model is that it supports both Python and C++. One of the main obstacles we will face is the time it takes to train 10,000 images. Even with the Jettson processor provided to us, it could take more than a few days to train all of the data from scratch. The initial goal is to attempt to use it with the Python implementation of the image classifier. If we find that it is too slow, we will use the C++ version as C++ is an objectively better performing language with regards to speed. TensorFlow also has multi-GPU support, which we may employ if we can muster the resources to speed up the training speed.

One of the difficult decisions we must make is whether to fine-tune a pre-trained model or train from scratch. Fortunately, the TF-Slim classifier supports both operations. The main disadvantage of training from scratch would be the long training time. Depending on the hardware setup, the process can take multiple days [2]. We may need another GPU to prevent bottlenecking. The Jettson Processor TX2 has both a CPU and a GPU, which can be run synchronously. We will resort to fine-tuning a pre-trained model if we find that we do not have the resources to efficiently train our data with our current setup. A pre-trained model will take significantly less computing power but more experimentation with parameters. We will pre-train our model on a dataset of Tall Fescue images that we will obtain either through the internet or our client. We will then fine-tune parameters on the final layer to train our dataset. According to the inception model's documentation, pre-training a model will yield significantly better results if we are limited by our hardware specs.

The dataset of images will be composed of images of Tall Fescue seed from the data collection stage. The plan is to train the classifier to accurately identify with an 80% degree of confidence if a seed is a Tall Fescue. Hence, we will train the classifier on as many images of Tall Fescue seed as possible. As of now, this is a binary classifier. This means the classifier will identify a seed as either a Tall Fescue seed or an off-type. The classifier does not have to identify each of the individual off-types, so we will hold off on training other types of images. We have discussed identifying the off-types with the client but decided it was beyond the scope of this project and will only be pursued if we have time at the end of this project.

We plan on commencing this stage as early as possible as to hit our target goal of finishing initial training by February

5th. It is important we give ourselves ample time to test as it is a long process.

5.5 Data Testing

After the training phase has been completed, our program should be able to recognize whether a seed is an off-type. The testing phase has two parts: early model testing and final model testing.

The technology that will be used to move the seeds and automate the setup is a vibrating table apparatus. The vibrating table has a suspended camera attachment, funnel, light source and dividers for sorting the seed. We will be using our Jettson TX2 processor to run the tests on accuracy.

5.5.1 Early Model Test

The early model test will be used to determine whether the program was properly trained on the dataset of images. The purpose of the early model test is to do a quick, preliminary test to see if the software will be able to differentiate an off-type from the target seed, which is Tall Fescue. If the early model testing stage is successful, we can move on to the final model testing stage.

This type of test works by manually inputting a set of 10 images of Tall Fescue seed and a set of 10 images of the more common off-types we may encounter. If we manage to get a Top-1 accuracy of 80%, we can commence testing by entering the final model testing stage. The reason for having a preliminary stage before commencing final testing is to ensure we do not waste time testing a large batch only to discover that there is an anomaly or bug that is preventing us from reaching our desired Top-1 accuracy. The preliminary stage will allow us to quickly figure out if something is wrong. If we do not get our desired Top-1 accuracy, it means we may need to retrain more images, need better image preprocessing, or need to modify parameters.

The Early Model Test can be used to determine if there are bugs in the program. However, it cannot be used as a means to test accuracy because the sample size is too small. We plan on commencing the early model test by February 10th.

5.5.2 Final Model Test

Our final model testing stage is the test used to determine if our classifer works and can accurately identify off-type seeds in real-time. The goal of the final model test is to ensure our program reaches our target Top-1 accuracy rate, which is 80%. This test will utilize the same approach as our data collection with regards to capturing images of the seeds.

As mentioned before, we are working in tandem with a mechanical engineering group. They have designed a funnel system that is connected to a vibrating table. To run our final model tests, we insert our seeds through the funnel onto the vibrating table. The vibrating table will then take the seeds to the image capture area. The image capture area is the surface area that is within the suspended camera's field of view. When a batch of seeds passes through the image capture area, the camera will constantly be taking pictures and inputting them into the computer via a USB connection and tethering software. It is important to note that it will take approximately 3-8 seconds for an image to be transferred onto the computer. As the camera is constantly feeding images of seeds in real-time to the computer, the classifier will output the probability of a seed matching Tall Fescue by running the testing script on each image. If the Top-1 accuracy of an image is less than 80%, we can discard that batch of seeds. The constraint we must work around is we need to minimize false positives as even a few can impact the quality of the seed batch.

When the classifier outputs a probability lower than 80%, that means there are likely off-types within the batch. The classifier will return false and send a signal from the computer to the testing setup. This signal will indicate that there is an off-type and that batch of seeds will be trashed. The reason behind trashing the seeds if there is even one off-type in the batch is because our client stated he would prefer false negatives over false positives. False positives can ruin the integrity of the batch and make it unsellable.

This stage will likely be the most difficult and extensive testing on timing in particular is needed. Our testing script for our model should run on batches of images in real time. Since it takes 3-8 seconds for an image to be transferred onto a computer via tethered shooting, we will need to implement a wait timer before running the script on the image. If the test script runs before the image has actually made it to the machine, then it will output errors. Furthermore, the vibrating table will likely need a stop button when processing the images. We are not yet sure if the Jettson TX2 will be fast enough to test images in real time at the speed of the vibrating table.

This testing phase will commence once we have our first prototype. Our estimated date of completion is February 25th.

5.6 Conclusion

In conclusion, this project can be split into four distinct parts: data collection, data preprocessing, data training, and data testing. It is important to note that while these steps are in sequence, we will very likely find bugs that require us to jump between the phases.

6 Design Viewpoints

6.1 Context Viewpoint

It is important that we can create a suitable abstraction for the user experience when designing the components for this project. Potential users will likely come from a variety of different backgrounds with ample knowledge in their field of expertise but varied degrees of knowledge regarding operating technology. It is important we account for the fact that users with different levels of computer fluency will be using this product.

6.2 Structure Viewpoint

The structure of this project follows a four phase plan. It is sequential because the phases must follow It is also important to note that each of these roles is dependent on the previous role being completed. The phases must be completed sequentially. We will likely switch

6.3 Interaction Viewpoint

The system requires interaction between components as each component is dependent on the prior one. This is unavoidable because there is simply no way to test the validity of our product without going through the necessary data collection and processing steps. We will constantly jump between phases as needed to iron out bugs and fine-tune parameters.

6.4 Algorithm Viewpoint

The model that is going to be used is the inception_resnet_v2, which is based on a deep convolutional neural network. The reason we are using this model is because of it's state of the art performance on the ImageNet dataset. It achieved a Top-1 percentage of 80.4%, which is considered to be cutting edge [3].

6.5 Stakeholder & Concerns

Our client requested that our project be designed in such away that it will allow other engineers to add further functionality to it. As of now, the scope of the project is to simply identify if there is an off-type seed in a batch. If there is, it will send a signal to the vibrating table indicating that the seed should be trashed. Our client expressed a desire for an interface that will allow a user to train a seed of their choice. Our project will only identify Tall Fescue seeds but adding this interface will allow users to identify different kinds of seeds. The client also expressed interest in identifying the off-type seeds. This would require us to produce datasets of each individual off-type and train them individually. It is highly unlikely we have the time to train more than one dataset. However, the plan is to implement our software in such a way that it makes the addition of these potential new features plausible.

7 References

[1] Tf.image.per_image_standardization. per_image_standardization, TensorFlow, https://www.tensorflow.org/api_docs/python/tf
.

[2] GitHub. (2017). tensorflow/models. [online] Available at: https://github.com/tensorflow/models/tree/master/research/slim
[Accessed 1 Dec. 2017].

[3] Improving Inception and Image Classification in TensorFlow. Research Blog, 31 Aug. 2016, research.googleblog.com/2016/08/i
inception-and-image.html.