Improving Soybean Disease Prediction by Performing Late-Stage Re-Training Using Fireworks Algorithm

Abstract— **This paper proposes a novel method of trend detection and visualization, more specifically, modeling the change in a topic over time. Whereas current models used for the identification and visualization of trends only convey the popularity of a singular word over time, our approach illustrates the popularity and direction that a topic is moving in. The direction in our case is a distinct subtopic within our corpus. We model the movement of a topic by using k-means clustering and cosine similarity to group the distances between clusters over time. In a convergent scenario, it is inferred that the topics are meshing (tokens between topics, are becoming interchangeable). On the contrary, a divergent scenario would imply that each topic's respective tokens would not be found in the same context. Our methodology was tested on millions of tweets automatically processed from a developer-based Twitter account, as well on a New York Times online data repository using an automated web scraping script. Our implementation produces trends so accurately that they are not detectable without thoroughly examining the text for specific relationships.**

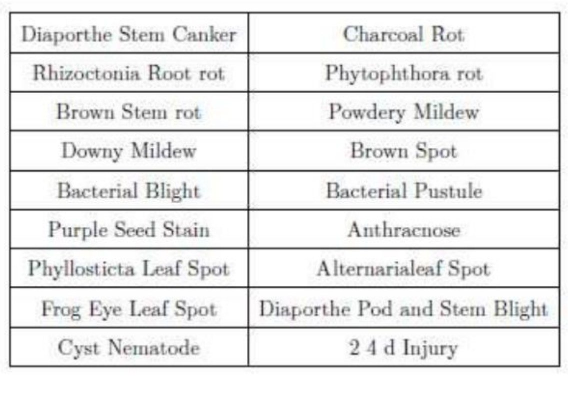
Keywords—**Optimizers, Fireworks, Adam, Soybean Classification, Artificial Neural Network**

I. Introduction

A. Soybean Disease Classification

The growing human population needs more and more food each year. This Growth demand instead means an increase in crop demand. Agricultural losses around the world are largely caused by plant diseases. Plant diseases, weeds, insects, and a decline in yields around the world have all been responsible for a decline in worldwide crop production. It is therefore of paramount importance that diseases are detected early to prevent their spread and reduce crop losses. Agriculture has for decades used cultural practices to protect the crops from pests and diseases, mostly in conjunction with solarization and rotation of crops to provide some protection from harmful pests, as well as the development of pesticide-resistant cultivars, in conjunction with the use of biological agents.

A healthy agroecosystem depends on the effective detection of diseases. With the development of molecular biology and biotechnology, plant diseases can now be detected more effectively. Preventing and controlling soybean diseases requires an automated diagnostic system. This is because it will minimize yield losses and economic losses caused by pesticide residues on the land and by improving the quality of the crops. To predict soybean diseases at an early stage, it is necessary to classify the diseases effectively. Several learning algorithms have also been applied to predict pest attacks and disease infestations in crops. An algorithm has been developed to compare aerial parts of healthy and diseased plants using spectral imaging data. In addition to monitoring morphological traits, there is also a substantial amount of evidence that ML methods are successful as well.



Diaporthe Stem Canker	Charcoal Rot
Rhizoctonia Root rot	Phytophthora rot
Brown Stem rot	Powdery Mildew
Downy Mildew	Brown Spot
Bacterial Blight	Bacterial Pustule
Purple Seed Stain	Anthracnose
Phyllosticta Leaf Spot	Alternarialeaf Spot
Frog Eye Leaf Spot	Diaporthe Pod and Stem Blight
Cyst Nematode	2 4 d Injury

Fig 1: Soybean Diseases

Nevertheless, the varying nature of plant changes may lead to inaccurate predictions due to changes in symptomatology. Thus, appearance-based disease identification cannot be relied on to identify diseases reliably, especially at the early stages of growth. To identify the causative agent of charcoal rot, it is important to have an appropriate detection method since the symptoms do not appear until midseason. Our method for predicting the occurrence of charcoal rot disease also includes morphological characteristics (including characteristics that are related to growth and yield) as well as physiological features. To train and assess machine learning algorithms, a hybrid set of features derived from healthy and diseased soybean plants is used. Experimental setups, as well as field conditions, are presented in the available dataset.

It uses machine learning to distinguish healthy plants from unhealthy ones. It has further been shown that supervised machine learning algorithms can be used to model disease in other diseases, based on mainly image datasets, and such algorithms can be used for disease prediction. It is the objective of this study to propose a set of features for enhancing the prediction of charcoal rot diseases, as well as to compare several machine learning techniques to enhance their accuracy. Agriculture crop yield is greatly affected by many pests and diseases. To improve the yield, early diagnosis and control of disease are very important.

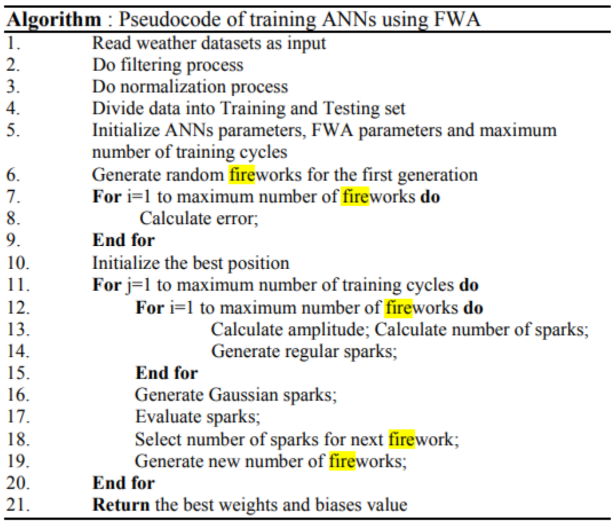
For a long time, Soybean is a rapidly growing crop in India, and it is considered a Kharif crop. The leading producer of soybean in India is Madhya Pradesh, followed by Maharashtra and Rajasthan [2]. Around the world, the Soybean is recognized as one of the outstanding crops. It is majorly used for protein, animal feed, and vegetable oil. Soybean is an important food commodity because of its high protein content (greater than 40%) and high oil content (greater than 20%). Since the Soya Protein supplies enough amino acids, it is called a complete protein. Soybean oil does not contain cholesterol. Low yield is the major problem with the soya industry in the country because diseases are one of the reasons.

Soybeans are affected by several diseases like downy mildew, pod, and stem blight, phytophthora root, and stem rot, brown spot, Cercosporin leaf blight, purple seed stain, frog eye leaf spot, and many more. The work deals with the classification of soybean diseases based on weather data, physic crop properties, plant properties, and crop management properties. The dataset is available in the UCI machine learning repository. We will use Machine learning techniques such as ANNs for this.

B. *Fireworks Algorithm*

Fireworks Algorithm (FWA) is a novel swarm intelligence algorithm introduced by Ying Tan [4]. FWA is one method of collaborated intelligence groups, which is a technique that simulates the behavior of fireworks in the night sky. The algorithm is generally used for solving optimization by searching for the optimum value inside the solution space of specific problems.

An explosion of fireworks occurs when fireworks are shot up into the sky, and they move to the appropriate position and explode into shining lights in various shapes, depending on the designs of the fireworks. Each firework has a specific explosion characteristic, for example, a narrow explosion with a lot of sparks, a wide explosion with fewer sparks, a wide explosion with so many sparks, and so on. Thus, each firework has a unique explosion and different positions to set off. In this research, FWA is used as an algorithm for training ANNs and determining the proper weights and biases. The proper values are chosen from the processing of the error values between the predicted results and the actual results. This algorithm has the characteristics of a repeating cycle until a suitable result is achieved. The work is divided into four main parts as described in the followings.



```
Algorithm : Pseudocode of training ANNs using FWA
1.   Read weather datasets as input
2.   Do filtering process
3.   Do normalization process
4.   Divide data into Training and Testing set
5.   Initialize ANNs parameters, FWA parameters and maximum
     number of training cycles
6.   Generate random fireworks for the first generation
7.   For i=1 to maximum number of fireworks do
8.       Calculate error;
9.   End for
10.  Initialize the best position
11.  For j=1 to maximum number of training cycles do
12.      For i=1 to maximum number of fireworks do
13.              Calculate amplitude; Calculate number of sparks;
14.              Generate regular sparks;
15.      End for
16.      Generate Gaussian sparks;
17.      Evaluate sparks;
18.      Select number of sparks for next firework;
19.      Generate new number of fireworks;
20.  End for
21.  Return the best weights and biases value
```

Fig 2: Fireworks Psudocode

II. Methodology

When deciding what algorithm to choose for the optimization step during the training of neural network models when training with large amounts of data where each epoch takes a considerable amount of time, we need to consider various factors regarding training time and the best fit in terms of our data. But it can also be intuitively understood that different sections of training can benefit most from different optimization techniques.

A. *Hypothesis*

Here we provide an example for the same, we first train our Soybean classifier using the Adam Optimizer that has been used in several implementations of this problem statement. But then as our algorithm gets trapped in a local minimum, we call upon the Firework algorithm to use built-in sparks to further train the model for better results, for both our training and testing data.

B. *Why Fireworks*

Fireworks is a swarm intelligence algorithm, that uses 'explosions' to semi-randomly find out the next optimal step in our solution space, is a great method to get out of any local minima that we might have gotten ourselves trapped into due to the linear descending structure of Adam.

C. *Limitations & Alternatives*

As we know, one of the drawbacks of the Adam optimizer is that in some scenarios Adam does not converge to the Optimal Solution. Also, slow convergence and low accuracy are the major issues with the Fireworks Algorithm. But if we use these operations in tandem, they proceed to rectify each other's drawbacks, i.e. The initial training is done using Adam provides us with a quick and accurate starting point, and further passes using fireworks take care of any local minimums and takes our solution to the global minimum.

This approach still has several issues, such as the Weight Decay problem in Adam and Explosion Tuning in Fireworks. These aforementioned problems can be circumvented using alternative versions of these algorithms themselves i.e., using the dynamic search firework algorithm (dynFWA) and AdamW or AMSGrad.

III. Experimental setup

A. *Dataset*

The dataset is retrieved using the pmbl library, where we have more than 30 columns (such as: date', 'plant-stand', 'precip', 'temp', 'hail', 'crop-hist'). We import this data, converting it into a dataframe and then making the necessary splits (X, y, test, train).

B. *Creating Custom Optimizer*

Using the built-in Optimizer class of TensorFlow, we created an implementation of the Fireworks Algorithm as a new class that can be called inside our main driver code. To create an optimizer function we need an Object Function to create and evaluate the sparks array inside the Fireworks class used to initialize new possible optimal values. The Object Function along with the number of dimensions and maximum iterations as parameters.

C. *Training Model*

Now as we begin creating and training our model. We create 2 instances of optimizers, one being Adam and the other being Fireworks. We create a sequential model with 3 Dense Layers with 56, 28, and 19 nodes respectively.

Then we create a new TensorFlow session followed by compiling, training, and finally evaluating

IV. Results and Discussion

Operation	Train Accuracy	Test Accuracy
1st Pass (Adam)	94.47%	90.58%
2nd Pass (Adam)	97.12%	89.69%
3rd Pass (Fireworks)	98.89%	92.03%

Fig 3: The results of the 3-step training

In Table I we can see that during training,

A. *First Pass (using Adam)*

We first train our model to start from around ~40% accuracy in the first epoch to nearly ~94.5% by the last epoch. Here we use speed and accuracy of Adam to get a well-trained model as our starting point.

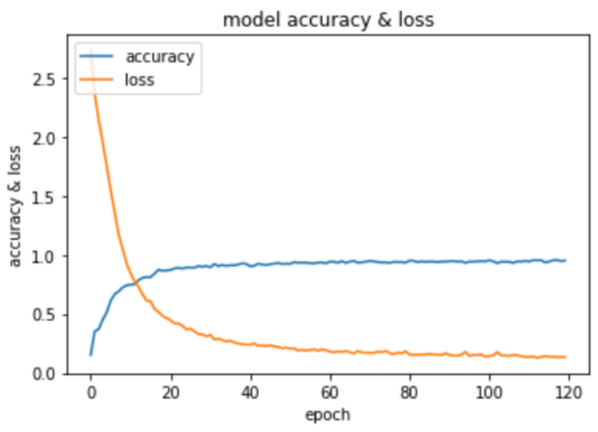
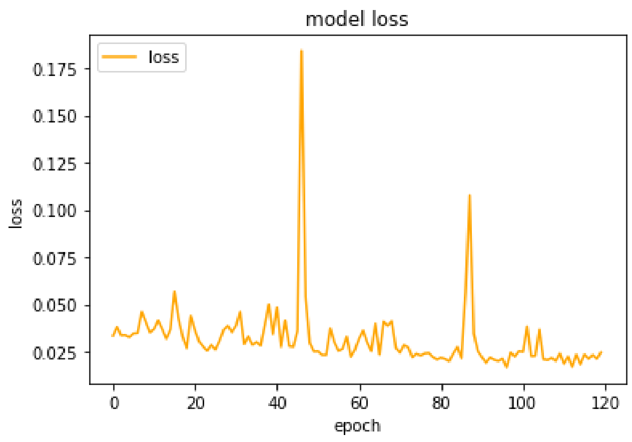


Fig 4: The results of 1st pass of the training

B. *Second Pass (using Adam)*



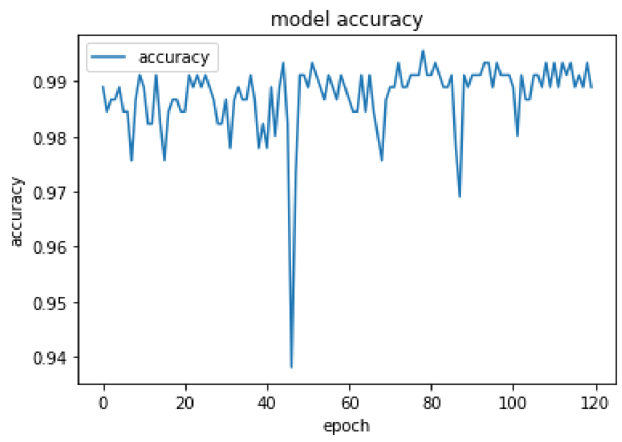
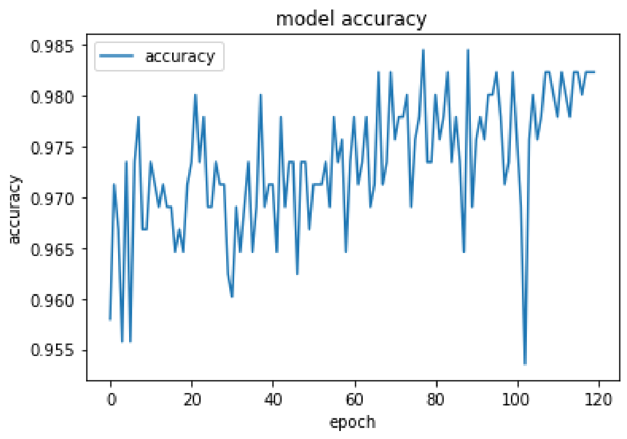


Fig 5: The results of the 2nd pass of the training

In the next phase of our training, we run the `model.fit()` function again to retrain our existing model on the dataset using Adam again. This operation gives us a slightly better train accuracy but nearly the same test accuracy (if not lower). Subsequently, no matter how many times we retrain, our test accuracy does not bulge.

C. *Third Pass (using Fireworks)*



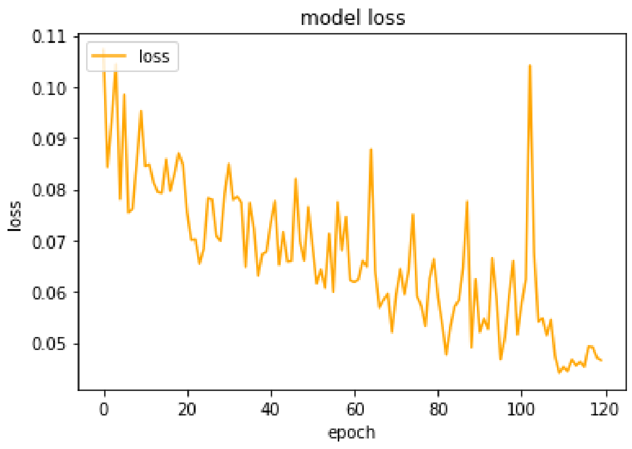


Fig 6: The results of the 3rd part of the training

In the next phase of our training, we now use fireworks as our optimizer and retrain our model. After training, we see an equivalent increase in both, the test and train accuracy of the dataset. Whereas with Adam our accuracy and loss were stagnant, Fireworks provides the last stage training to get better results.

V. Conclusion and Future work

In conclusion, we discovered a new way of getting better results on both our test and train data with the help of the hybrid implementation of optimizers.

In the future, we would focus on more hybrid optimization combinations and further research on consequent training.

Acknowledgment

MIT Manipal, Rajashree Krishna Ma'am.

REFERENCES

[1] Dr. Nanda Ashwin1, Uday Kumar Adusumilli2. A Machine Learning Approach to Prediction of Soybean Disease, ISSN: 2394-4099

[2] Rajashree Krishna, Prema K V. Soybean crop disease classification using machine learning techniques, INSPEC Accession Number: 2027-9158

[3] Saktaya Suksri, Warangkhana Kimpan. Neural Network Training Model for Weather Forecasting Using Fireworks Algorithm, INSPEC Accession Number: 16692911

TODO

1. Fix References
2. Write Acknowlegements
3. Write Abstrat