TENACIOUS FISH SWARM OPTIMIZATION BASED HIDDEN MARKOV MODEL (TFSO-HMM) FOR AUGMENTED ACCURATE COTTON LEAF DISEASE IDENTIFICATION AND YIELD PREDICTION

S.GOVINDASAMY[1], D.JAYARAJ[2]

[1]Research Scholar, Department of Computer & Information Science, Annamalai University, Chidambaram,
[2]Assistant Professor / Programmer, Department of Computer Science & Engineering,
Annamalai University, Chidambaram
E-mail: [1]govindasamy1412@gmail.com, [2]jayarajvnr@gmail.com

ABSTRACT

This research presents an innovative approach called Tenacious Fish Swarm Optimization based Hidden Markov Model (TFSO-HMM) for augmented accurate cotton leaf disease identification and yield prediction. Cotton leaf diseases significantly threaten crop productivity, requiring timely detection and precise prediction for effective disease management. The proposed TFSO-HMM framework combines the strengths of Tenacious Fish Swarm Optimization (TFSO) and the Hidden Markov Model (HMM) to address the challenges associated with disease identification and yield prediction in cotton plants. TFSO, a nature-inspired optimization algorithm, optimizes the classification process, enhancing the accuracy of disease identification. By harnessing the collective intelligence of fish swarms, TFSO intelligently explores the search space to identify the optimal solution. The selected information is then incorporated into the HMM framework, which captures the temporal dependencies in disease progression and yield prediction. HMM's sequential modelling approach facilitates understanding the dynamic behaviour of cotton leaf diseases over time, leading to more accurate predictions. Experimental results on a comprehensive dataset demonstrate the superior performance of the TFSO-HMM method over existing approaches in terms of accuracy and predictive capability. The augmented accuracy achieved through TFSO-HMM enables early detection and precise prediction of cotton leaf diseases, enabling timely interventions for disease management and maximizing crop yield.

Keywords: *Tenacious Fish Swarm Optimization, Hidden Markov Model, Cotton Leaf Disease, Yield Prediction, Disease Identification, Augmented Accuracy.*

1. INTRODUCTION

Identifying and classifying leaf diseases is paramount for early detection and effective plant health management. The process commences with data collection, amassing a comprehensive dataset encompassing healthy and diseased leaf images. The collected data then undergoes preprocessing, wherein various techniques are applied to improve image quality and prepare the images for subsequent analysis. Feature extraction follows, where distinctive characteristics are extracted from the preprocessed leaf images. A model is selected to perform the classification task, and its hyperparameters are fine-tuned for optimal performance. The model is trained using the available data, and its effectiveness is evaluated and validated. In addition, error analysis and diagnosis aid in understanding misclassifications and improving overall accuracy. The identified diseases are ultimately presented as the output result, enabling effective decision-making in plant disease management. The general steps involved in lead disease identification are given in Figure 1.

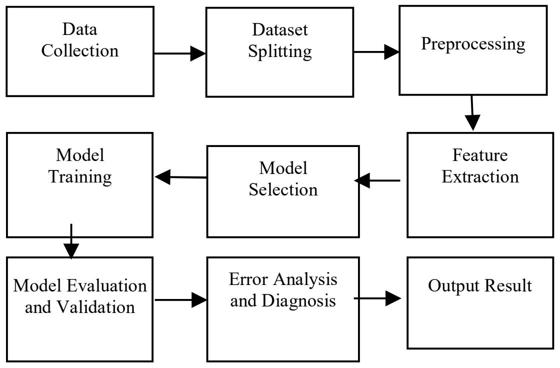

Figure 1. Image Processing in Leaf Disease Identification

Leaf diseases are conditions that affect the health and growth of plants by causing damage to their leaves. These diseases can be caused by bacteria, fungi, viruses, and environmental stressors such as extreme temperatures or insufficient water. One of the most common types of leaf diseases is fungal infections. These infections can cause leaf spots, blight, and rust [1]. Symptoms may include discoloration of the leaves, yellowing, wilting, and leaf drop. The fungus can spread quickly throughout the plant and, if left untreated, cause significant damage or even death. Bacterial infections can also cause leaf diseases, such as bacterial leaf spot. This condition is characterized by brown or black spots on the leaves, which can eventually spread and cause the leaves to die. Viruses can also cause leaf diseases, which can be more challenging to diagnose and treat. Symptoms may include yellowing or mottling of the leaves, stunted growth, and distorted leaf shapes [2]. Leaf diseases can have a significant impact on plant health and productivity. Diseased leaves are less able to photosynthesize, which can reduce the amount of energy available for growth and fruit production. In severe cases, leaf diseases can cause defoliation, leading to plant death. Preventing leaf diseases involves a combination of good cultural practices, such as proper watering and fertilization and using fungicides, bactericides, or other treatments as necessary. Early detection and prompt treatment are also crucial to minimizing damage and preventing the spread of disease [3].

Cotton leaf diseases can significantly impact the overall yield of cotton crops. In addition to the reduced photosynthesis and growth of diseased leaves, conditions can lead to premature plant defoliation, where leaves drop off before they should [4]. This reduces the plant's ability to produce bolls, where the cotton fibres are harvested. In severe cases, cotton diseases can cause complete crop failure, resulting in significant economic losses for farmers and impacting the global cotton supply. For instance, Fusarium wilt can cause up to 100% yield loss in susceptible cotton varieties. Cotton farmers use preventative and curative measures to control crop diseases [5]. Preventive measures include planting disease-resistant varieties, crop rotation, and maintaining healthy soil with proper irrigation, fertilization, and tillage. Curative measures involve using fungicides, bactericides, or other chemicals to treat diseased plants and prevent the spread of disease. However, chemical treatments can have negative environmental impacts and may lead to the development of resistant strains of pathogens over time. Therefore, many farmers are adopting alternative disease control methods, such as biocontrol agents, genetic engineering, and precision farming techniques [6].

A subfield of AI, machine learning is teaching computers to infer meaning from data without being told what to look for. In essence, it is a way for machines to learn from examples and experience and use this knowledge to make decisions or predictions about new data. In the context of cotton leaf disease identification, machine learning can be used to analyze data from images of diseased cotton leaves to identify the type of disease and predict its severity accurately [7]. An extensive collection of tagged photos is used to train a machine-learning model, where each image is tagged with the corresponding disease type and severity level. The trained model may then be used to classify new cotton leaf photos, accurately identifying the type of disease and predicting its severity. This can be especially useful for early detection and rapid response to disease outbreaks and for developing effective treatment strategies [8].

There are several approaches to machine learning that can be used for cotton leaf disease identification, including [9, 10]:

- In supervised learning, a model is trained using data that has already been annotated, where each image is tagged with the corresponding disease type and severity level. The model can then classify new ideas based on the patterns it learned from the training data.
- Unsupervised learning: Includes training a model on data that hasn't been labelled and letting it figure out patterns and connections on its own. This approach can be helpful when there is no prior knowledge about the types of diseases or their severity levels.
- In deep learning, neural networks discover hidden connections and patterns in data. Algorithms based on deep learning can assess photos quickly and accurately, as they can learn to recognize patterns at multiple levels of abstraction.

1.1. Motivation

Cotton is a vital cash crop for many countries, and its production significantly contributes to food security and economic stability. Cotton leaf diseases substantially threaten global agricultural resilience and food production. Farmers and agricultural organizations can take timely actions to minimize the impact of diseases on cotton production by accurately classifying these diseases and predicting crop yields. The availability of a reliable disease classification and yield prediction system can empower farmers to implement appropriate preventive measures, adopt disease-resistant varieties, and improve crop management practices to enhance overall productivity. This, in turn, ensures a steady supply of cotton for the textile industry and mitigates the risk of food shortages. By developing an effective cotton leaf disease classification and yield prediction system, we can bolster global agricultural resilience, contribute to food security, and support sustainable development goals related to poverty reduction and economic stability in cotton-producing regions.

Bio-inspired optimization, a computational approach inspired by biological systems, has the potential to address diverse research issues across multiple domains [11–24], [25], [26], [27]. These algorithms offer versatile and practical solutions by emulating adaptive behaviors observed in nature. They have been successfully applied in engineering design, data mining, scheduling, image processing, robotics, and more.

1.2. Problem Statement

Cotton leaf diseases significantly threaten global agricultural resilience and food security. The lack of an accurate and efficient cotton leaf disease classification and yield prediction system hinders farmers' ability to identify and manage these diseases effectively, resulting in substantial yield losses and compromising the availability of cotton for the textile industry and other related sectors. Farmers face difficulties identifying specific diseases affecting their cotton crops without a reliable classification system, leading to delayed or inadequate interventions. Consequently, the spread of diseases goes unchecked, resulting in reduced crop productivity and lower yields. The inability to accurately predict yield based on disease profiles further exacerbates the challenge, as it hampers farmers' capacity to plan for future production levels and take proactive measures to mitigate potential food shortages. To address these issues, there is an urgent need to develop a robust cotton leaf disease classification system that can accurately identify and classify diseases affecting cotton plants. Farmers and agricultural organizations can make informed decisions regarding disease management strategies, resource allocation, and overall production planning by integrating a yield prediction component. The development of such a system would contribute to global agricultural resilience by reducing crop losses, ensuring food security, and supporting sustainable development goals related to poverty reduction and economic stability in cotton-producing regions.

1.3. Research Objective

The objective of this research titled "TFSO-HMM: A Novel Method for Accurate Cotton Leaf Disease Identification and Yield Prediction" is to develop and evaluate a novel method that combines Hidden Markov Models (HMM) with the enhanced version of Fish Swarm Optimization (FSO) for accurate cotton leaf disease identification and reliable yield prediction. Based on the problem statement related to Motivation 3, which highlights the significance of accurate cotton leaf disease classification and yield prediction for ensuring global agricultural resilience and food security, the research objective aims to address this challenge by leveraging the capabilities of HMM and FSO. The specific objectives are as follows:

- **Develop an enhanced cotton leaf disease identification model using HMM:** Design a Hidden Markov Model that leverages the sequential nature of symptom development in cotton leaf diseases. The model will learn and capture the underlying patterns and transitions between disease states, enabling accurate identification of specific diseases. Train the HMM using a labelled dataset of cotton leaf disease samples to learn the disease-specific emission and transition probabilities.

- **Enhance Fish Swarm Optimization (FSO) for model parameter optimization:** Integrate the enhanced version of the FSO algorithm (namely TFSO) with the HMM framework to optimize the model parameters. TFSO will be employed to fine-tune the emission and transition probabilities of the HMM, maximizing the model's accuracy and robustness in cotton leaf disease identification. The TFSO algorithm will simulate the movement and behaviour of a fish swarm, enabling efficient parameter optimization.

- **Evaluate the performance of the TFSO-HMM method for disease identification and yield prediction:** Assess the accuracy and effectiveness of the TFSO-HMM method in accurately identifying cotton leaf diseases. Compare the performance of the proposed method with other existing classification techniques and traditional HMM models.

Additionally, incorporate the disease identification results into a yield prediction model, enabling reliable estimation of crop productivity based on disease profiles.

- **Analyze the impact of the TFSO-HMM method on global agricultural resilience and food security:** Evaluate the potential benefits of the TFSO-HMM approach for enhancing global agricultural resilience and ensuring food security. Analyze the implications of accurate disease identification and reliable yield prediction on optimizing disease management strategies, resource allocation, and overall cotton production. Assess the potential economic and societal impacts of improved agricultural practices enabled by the TFSO-HMM method.

2. LITERATURE REVIEW

"Learning Aided System for Agriculture Monitoring" [28] combines image processing techniques with IoT-CNN architecture to enable efficient and accurate monitoring of agricultural activities. The system can detect and classify agricultural parameters such as crop health, weed infestation, and pest presence by capturing and analyzing images from IoT devices deployed in the fields. An extensive collection of labelled photos teaches the IoT-CNN model, allowing it to learn and recognize patterns associated with different agricultural conditions. This integrated approach enhances the monitoring capabilities and enables timely decision-making for farmers. "Tomato Leaf Disease Recognition Systems" [29] explores using machine learning algorithms to develop efficient and accurate models for detecting and recognizing tomato leaf diseases. These systems can analyze leaf images and classify them into different disease categories by leveraging advanced machine learning techniques, such as convolutional neural networks (CNNs) or support vector machines (SVMs). Machine learning enables rapid and early detection of diseases, allowing farmers to take timely action to prevent the spread of infections and optimize crop yields. Implementing such systems in developing countries can significantly benefit small-scale farmers by providing them with affordable and accessible tools for disease management.

"Tomato Leaf Disease Classification" [30] proposes a methodology that leverages transfer learning techniques to utilize pre-trained models and extract high-level features from tomato leaf images. Combining the learned features with handcrafted features through feature concatenation improves the classification model's accuracy and robustness in identifying various tomato leaf diseases. Transfer learning enables the model to benefit from knowledge gained from large-scale datasets, while feature concatenation ensures the integration of learned and manually designed features. This approach enhances the performance of the classification system, enabling accurate and reliable disease identification. "Hybrid Convolutional Neural Network-based Classification" [31] proposes a methodology that combines different CNN architectures to classify diseases effectively based on the type of pathogen affecting the tomato plants. By training the hybrid CNN model on a diverse dataset of tomato leaf images, the system learns to distinguish between bacterial, viral, and fungal diseases with high accuracy. The hybrid approach leverages the strengths of multiple CNN architectures to capture both low-level and high-level features, enhancing the discriminative power of the model. This methodology contributes to plant disease classification by providing a comprehensive and accurate solution for identifying different diseases affecting tomato plants.

"AdaBoostSVM Classifier" [32] proposes a methodology that combines the AdaBoost algorithm with support vector machines (SVM) for disease classification in rice plants. The AdaBoost algorithm is used to enhance the performance of the SVM classifier by iteratively adjusting the weights of training samples to focus on difficult-to-classify instances. The system learns to properly categorize and detect different illnesses affecting rice plants by being trained on a dataset of tagged photos of rice plants using the AdaBoostSVM model. The combination of AdaBoost and SVM offers improved accuracy and robustness in disease detection. "Cassava Disease Recognition" [33] proposes a methodology that addresses the challenge of limited data and low-quality images commonly encountered in cassava disease recognition. The enhanced data augmentation model generates additional training samples by applying various transformations and image processing techniques, expanding the dataset and improving the model's generalization ability. Deep learning techniques, such as convolutional neural networks (CNNs), are then employed to learn and extract meaningful features from the augmented images. The system achieves accurate and robust cassava disease recognition by training the deep learning model on the enhanced dataset, even from low-quality images.

"Attention Deep Learning-based Large-Scale Learning Classifier" [34] proposes a methodology that leverages attention mechanisms within deep learning architectures to enhance the classifier's ability to focus on relevant features within the Cassava leaf images. The model can selectively attend to important regions and patterns by incorporating attention mechanisms, improving classification accuracy. This approach benefits large-scale learning scenarios where the dataset consists of many images. The attention-based deep learning classifier can efficiently process and classify Cassava leaf disease images, enabling rapid and accurate identification of various diseases affecting Cassava plants. "ResNet with YOLO Classifier" [35] proposes a methodology that combines the strengths of both ResNet and YOLO architectures to improve the accuracy and efficiency of disease recognition in paddy leaf images. ResNet is utilized for its deep feature extraction capabilities, enabling the model to capture intricate details and patterns associated with different diseases. YOLO, known for its real-time object detection capabilities, is integrated to accurately localize and classify disease regions within the paddy leaf images. Combining these two architectures, the hybridized model achieves enhanced accuracy and computational efficiency.

"Weighted Ensemble Learning" [36] proposes a methodology that combines image processing algorithms with ensemble learning to improve the accuracy of disease classification in tomato leaves. Image processing techniques extract relevant features and characteristics from the leaf images, providing valuable information for disease identification. Weighted ensemble learning is then applied to combine the predictions of multiple classifiers, assigning higher weights to more accurate classifiers. This approach enhances the overall classification performance by leveraging the strengths of different models and reducing the impact of individual classifier errors. The weighted ensemble learning technique ensures robust and reliable disease classification in tomato plants. "Multiscale Voting Mechanism" [37] proposes a methodology that leverages multiscale analysis to capture features at different resolutions, allowing for more comprehensive disease recognition. The voting mechanism combines the predictions of multiple classifiers operating at different scales, using their collective decisions to make the final classification. This approach enhances the robustness and accuracy of disease recognition by considering various perspectives and avoiding the limitations of single-scale analysis. The multiscale voting mechanism is designed to handle the challenges posed by natural field conditions, where variations in lighting, background, and other environmental factors can affect disease appearance. By incorporating this mechanism, the system performs better in identifying rice leaf diseases under realistic field conditions.

"Plant Disease Detection" [38] utilizes machine learning algorithms to analyze plant images and identify diseases accurately. One commonly used technique is the application of convolutional neural networks (CNNs), designed to learn and extract meaningful features from images. Convolutional neural network (CNN) models are trained on massive annotated datasets of plant photos to recognize patterns and traits associated with different illnesses. Another approach is using decision trees or random forests, which employ a set of rules to classify plant images based on their visual features. Support vector machines (SVMs) are also utilized, where they create a hyperplane to separate different disease classes. Additionally, deep learning techniques like recurrent neural networks (RNNs) and long short-term memory (LSTM) networks have been applied to analyze sequential data, such as time series data from sensors monitoring plant health. These machine learning approaches enable early and accurate detection of plant diseases, facilitating timely intervention and management strategies. "Self-Adaptive-Deer Hunting Optimization" [39] proposes an optimal weighted feature selection mechanism that utilizes the SADHO algorithm to identify the most informative and discriminative features from the plant leaf images. The classification model is optimized for accurate disease detection by assigning appropriate weights to these selected features. The approach also incorporates a hybrid classifier that combines the strengths of multiple classification algorithms, such as support vector machines (SVM), random forests (RF), or artificial neural networks (ANN), to improve the overall classification performance. This hybrid classifier leverages the complementary nature of different algorithms, enhancing the robustness and accuracy of disease detection. By integrating the SADHO algorithm, weighted feature selection, and hybrid classification, this research provides an effective solution for automated disease detection in plant leaves.

"Random Forest (RF)" [40] is a robust machine-learning algorithm that has gained significant attention in cotton leaf disease identification. Its working mechanism involves creating an ensemble of decision trees, each trained on a different subset of the training data. During the classification process, the algorithm combines the predictions of all trees to make the final decision. This ensemble approach offers several advantages, including improved accuracy, robustness to noise, and reduce overfitting. In the context of cotton leaf disease identification, RF has demonstrated excellent performance in accurately classifying various disease types based on their symptom patterns. Its ability to handle high-dimensional data, deal with missing values, and capture complex relationships makes it well-suited for this task. Using RF, researchers make significant strides in automating cotton leaf disease identification, early detection, and supporting effective agricultural disease management.

"Support Vector Machines (SVM)" [41] have become a popular machine-learning algorithm for cotton leaf disease identification due to their effective working mechanism. SVM works by transforming the input data into a higher-dimensional feature space using a kernel function, where it aims to find an optimal hyperplane that separates different disease classes. SVM aims to maximize the margin between the decision boundary, the support vectors, and the data points closest to the boundary. This margin maximization approach allows SVM to achieve good generalization and robustness in disease classification. In the context of cotton leaf disease identification, SVM has shown promising results in accurately classifying different disease types based on their symptom patterns. Its ability to handle high-dimensional data, handle nonlinearity through kernel functions, and find optimal decision boundaries makes it a valuable tool for automated disease identification and supporting sustainable agricultural practices.

3. TENACIOUS FISH SWARM OPTIMIZATION BASED HIDDEN MARKOV MODEL

3.1. Hidden Markov Model

The hidden Markov Model (HMM) belongs to the statistical models used for extracting concealed information from observed sequences of symbols in various applications. HMM uses an unconstrained Markov model to represent the system. Separating the unknown parameters from the known ones is HMM's fundamental difficulty. HMM precisely models real-time data and can simulate the source data. Multiple machine learning strategies have been developed based on HMM, which find significant applications in computational biology, optical character recognition, and speech recognition. HMM is an indispensable tool in bioinformatics due to its robustness, manipulability, simplicity, and adaptability to handle multiclass problems. A discrete stochastic process describes the input sequence as an output sequence during the HMM procedure. The system continuously transitions between states not immediately apparent to the user. Each hidden state represents a fundamental element of the modelled data, such as the level of amino acid present in a protein sequence.

3.1.1. Hidden markov model in image classification

Hidden Markov models (HMMs) are widespread because these probabilistic models are effective in many contexts, such as voice recognition, NLP, and time series analysis. They can also be utilized for image classification tasks. An HMM treats an image as a sequence of observed symbols or features in image classification. These symbols can be derived from various image characteristics, such as color, texture, or shape. The HMM can learn the underlying patterns and structure in the image data by capturing the spatial or temporal dependencies between these symbols. The key idea behind using HMMs for image classification is that underlying hidden states influence the observed symbols. These hidden states represent the latent factors or classes that the image belongs to. For example, in a binary classification problem of distinguishing between cats and dogs, the hidden states could represent the "cat" and "dog" classes. During the training phase, the HMM learns the statistical properties of the observed symbols and their relationship to the hidden states. Calculating the chance of detecting a specific symbol given a hidden state requires calculating the distribution of the starting state and the probability of transitioning between different concealed states.

Once the HMM is trained, it can be applied to classify new images. This involves inferring probable hidden-states-sequence (Viterbi algorithm) or estimating the posterior probabilities of hidden states given the observed symbols. The image is assigned to a specific class or category based on the inference results. By leveraging the temporal or spatial dependencies captured by the HMM, image classification systems based on HMMs can effectively model the relationships between different parts or features of an image. This can lead to improved classification accuracy and the ability to handle variations within image datasets.

HMMs can be used for image classification by treating the image as a sequence of observed symbols and leveraging the probabilistic modelling capabilities of HMMs. The steps involved in HMMs when applied for image classification are:

- **Image Representation:** Convert each image into a sequence of symbols or features that capture relevant information. For example, images can be divided into smaller regions or patches, and features like colour histograms, texture descriptors, or local binary patterns can be extracted from each patch.
- **Symbolization:** Map the extracted features to discrete symbols or discrete levels to create an observed symbol sequence. This step is essential because HMMs operate on discrete symbols. The symbols should capture discriminative information about the image patches.
- **HMM Training:** Train an HMM model using a labelled dataset of images. Each picture in the training set is mapped to a series of symbols. The HMM learns the underlying statistical patterns and dependencies between the observed symbols and hidden states.
- **Hidden States:** Define a set of hidden states that capture the underlying characteristics or classes of images. These hidden states represent the latent or unobservable factors contributing to the observed symbols. For example, if classifying images like "cat" or "dog," the hidden states can represent different visual patterns associated with each class.
- **Model Parameters:** Calculate the HMM's transition probabilities (A), emission probability (B), and starting state distribution (). Given a set of concealed states, the likelihood of being in each beginning state is represented by a distribution, and state transition probabilities capture the probability of moving between initial states, and emission probabilities model the probability of witnessing each symbol.
- **Inference:** Given a new image to classify, apply the trained HMM model to perform inference. This involves computing the most likely sequence of hidden states (Viterbi algorithm) or estimating the posterior probabilities of hidden states given the observed symbols.
- **Classification:** Assign the image to a class based on the inference results. The assigned class can be determined by considering the most likely hidden state sequence or using a decision rule based on the posterior probabilities.
- **Evaluation and Iteration:** Metrics like precision, recall, accuracy, and F1-score may be used to evaluate the efficacy of an HMM-based image categorization system. If necessary, iterate and refine the model and feature representation to improve classification performance.

3.1.2. Stochastic process

The fundamental principle behind modelling a process is collecting instances derived from it. This allows us to understand different aspects of the procedure and make predictions based on historical data. Stochastic models serve three primary purposes:

(i) Elaborate process details for improved outcomes,

(ii) Forecast results, and

(iii) Perform classification by predicting a single variable k based on input values from a limited and unordered set, where the input data x is defined as $(x_1, x_2, x_3, \dots., x_n)$.

In a stochastic model, events can be predicted with certainty. A stochastic model introduces a state of non-determinism, meaning it performs its process randomly. The term "stochastic" derives from "random." On the other hand, a deterministic model makes only one prediction for each set of inputs. The stochastic model is driven by a probabilistic event sequence in which different numbers determine each step's result. To rephrase, the probabilistic outcomes predicted by the stochastic model consider both the probability value and the weighted likelihood. Probability distributions for outcomes may be predicted using stochastic process modelling, which accounts for random changes in input across time. One definition of a stochastic process in the field of joint probability is a random variable $X = X_t$; while the prediction of a deterministic model is always the same given a specific set of inputs. In the stochastic model, each step in the process has a different likelihood of success, and the whole thing is predicated on that. To rephrase, the probabilistic outcomes predicted by the stochastic model consider both the probability value and the weighted likelihood.

Stochastic process modelling is a tool to predict probability distributions for potential outcomes, accommodating random variations in input over a specified period. A stochastic process is characterized by a random variable $X = \{X_t; t \in T\}$ in the space of joint probability. It takes a state space S (i.e., a standard set) and is indexed by T, representing discrete-time intervals.

3.1.3. Markov processes and markov chains

There are many stochastic processes, but the most prominent are Markov chains & Markov processes. These models represent the behaviour of systems whose next state is determined purely by the existing state, making them effectively memoryless.

Markov Process

A Markov process is characterized by its ability to reduce memory utilization. It is a stochastic process in which the dispersion of the following state depends solely on the present state, disregarding previous states or events. This property allows for efficient modelling and analysis, as a finite number of states can describe the process. The conditions for a stochastic process X(t) to be considered a Markov process are as follows:

- **Finite State Count:** A Markov process must have a finite number of possible states or outcomes. This means the system can exist in a limited and well-defined set of states, and transitions between states occur according to specific rules.
- **Constant Probabilities over Time:** The probabilities associated with transitioning between states remain constant as time progresses. This implies that the transition probabilities are independent of time, ensuring the system's dynamics are consistent throughout the process.
- **Memorylessness:** One of the critical characteristics of a Markov process is its property of memorylessness. According to this characteristic, past events have no bearing on the likelihood of an inevitable future state occurring. No extrinsic factors influence the conditional likelihood of a future state transition.

By satisfying these conditions, a stochastic process can be classified as a Markov process, enabling the application of specific mathematical techniques to analyze and model the system's behaviour.

Markov Chain

Markov chains are a special kind of Markov process that adheres to memoryless conditions and has a discrete state space. It consists of a sequence of states, where transition probabilities determine the transition from one state to the next. The chain moves from state to state based on probabilistic transitions, forming a stochastic process with Markovian properties. Markov chains are widely used in various fields, including mathematics, physics, economics, and computer science, for modeling systems with probabilistic dynamics and predicting future states based on current observations.

3.1.4. Hidden markov models

HMM is a probability-based model that employs two simultaneous stochastic processes: a state transition process governed by the Markov property and an output process that produces random sequences. The Markov model describes the sequence of variables using initial and transformation probabilities. In contrast, the output model generates characters from a given alphabet with only one element. The state transition sequence is hidden, meaning the variables' states are not directly observed but inferred from the output symbols within the sequence. Therefore, HMM can be characterized by its states, state, transition, and output probabilities. HMM serves as an architecture for processing inputs, and a quintuple represents its formal definition (S, V, π, A, B), It includes the following components:

- $S = \{S_1, S_2, S_3, \dots, S_n\}$: Represents the set of states, with N, indicating the total number of states.
- The Markov chain is denoted as a triplet (S, π, A), where the states are hidden and not directly observed.
- Vocabulary is defined as a set $V = \{v_1, v_2, v_3, \dots \dots, v_m,\}$.
- $\pi: S \to [0,1] = \{\pi_1, \pi_2, \pi_3, \dots, \pi_n\}$: Represents the initial stage of the probability distribution, indicating the probability of each state. It satisfies the condition $\Sigma(s \in S)\pi(s) = \Sigma(i = 1)^n \pi_i = 1$.
- The probability of transitioning from one state S_i to another state S_j is represented by

$A = (a_{ij})_{ins}$, where $a_{ij} \in [0,1]$ for every individual transition between S_i and S_j, and it satisfies the condition $\Sigma(i \in S)a_{ij} = 1$.

- $B = (b_{ij})_i^v s$ indicates the output probability, where v_i corresponds to the state S_i.

HMM proves useful for modelling processes when the system's state is unknown. Its fundamental concept involves generating a sequence of numbers randomly. Typically, when observing output events, HMM is considered a generative model used to generate sequences for observation. Algorithmically, an observed sequence $O = o_1, o_2, o_3, \dots, o_T$, where $o_t \in V$ can be generated by an HMM.

3.1.5. Forward algorithm

Hidden Markov Models (HMMs) rely on the forward algorithm, which is essential in many fields. The probability of transitioning between states at every step in an observable sequence is computed. The forward algorithm provides valuable insights into the likelihood of different hidden state sequences, enabling us to make inferences and perform tasks such as decoding and parameter estimation. In HMMs, the underlying system is modelled as a probabilistic framework comprising two stochastic processes: a Markov process governing the state transitions and an output process determining the emitted symbols. Given the observed sequence, the forward algorithm allows us to efficiently compute the probabilities of being in different states at each time step. By calculating the emission and transition probability based on the starting state distribution, the algorithm recursively calculates the forward variables, representing the probabilities of being in each state at a specific time. By employing the forward algorithm, we can analyze and understand the dynamics of the hidden states in an HMM. It lays the groundwork for tasks like estimating states, predicting sequences, and finding the most likely order of hidden states given an observational sequence. Additionally, the forward algorithm serves as a building block for other algorithms in HMMs, such as the backward algorithm, the Baum-Welch algorithm for parameter estimation, and the Viterbi algorithm for decoding. Here's a step-by-step representation of the forward algorithm:

Algorithm 1. Forward Algorithm

```
Step 1: Initialize the forward variable $\alpha$ at time $t = 1$
    a) For each state $i$:
    b) Set $\alpha_1(i) = \pi(i) * B(i, o_1)$, where
       $\pi(i)$ is the initial state distribution, and
       $B(i, o_1)$ is the probability of emitting
       the first observed symbol from state $i$.
Step 2: Recursion
    a) For each time step $t$ from 2 to $T$:
    b) For each state $j$:
    c) Calculate the forward variable $\alpha_{tj}$
       using $\alpha_{tj} = 0$.
    d) For each state $i$:
    e) Add $\alpha_{tj} += \alpha_{t-1}(i) * B(j, o_t)$, where
       $A(i,j)$ is the transition probability
       from state $i$ to state $j$, and $B(j, o_t)$ is
       the probability of emitting a symbol $o_t$
       from state $j$.
Step 3: Termination
    a) Figure out the forward chance at the
       last time step $T$:
    b) Set $P(O|\lambda) = 0$.
    c) For each state i:
    d) Add $(O|\lambda) += \alpha T(i)$, where $\alpha T(i)$
       represents the forward variable at time
       $T$ for state $i$.
Step 4: Output
    a) The prospective possibilities Given the
       sequence $O$, the probabilities of being
       in state $j$ at time $t$ are represented by
       $\alpha_{tj}$.
    b) The forward probability $P(O|\lambda)$
       represents the overall likelihood of the
       observed sequence $O$ given the HMM
       parameters $\lambda$.
```

3.1.6. Backward algorithm

The backward algorithm is an essential component of HMMs and complements the forward algorithm in providing a complete understanding of the underlying system. It allows us to compute the probabilities of future observations given a particular state at each time step. The backward algorithm is crucial in bioinformatics parameter estimation, decoding, and sequence alignment. In HMMs, the backward algorithm operates in tandem with the forward algorithm, providing a complementary perspective on the observed sequence. At each time step, the forward algorithm determines the likelihood of several possible states. The backward algorithm computes the probabilities of future observations given a specific state at each time step. This information is valuable in deciphering the significance and impact of different hidden states on future observations. The backward algorithm starts by initializing the backward variable at the last time step to capture the probabilities of the final observations given each state. It then recursively calculates the backward variables at earlier time steps, considering the transition and emission probabilities. By utilizing these probabilities, the backward algorithm effectively propagates the information from future observations to the present, allowing us to estimate the likelihood of observing specific sequences of symbols.

By employing the backward algorithm, we comprehensively understand the HMM by considering past and future information. This information is crucial for decoding tasks, where the sequence seen is used to infer the most likely sequence of concealed states. Additionally, the backward algorithm plays a pivotal role in parameter estimation techniques such as the Baum-Welch algorithm, which relies on the backward probabilities to update the model parameters and improve their accuracy iteratively.

Algorithm 2. Backward Algorithm

```
• Initialization
• Set $\beta_t(i) = 1$ for every state $i$ at the last time step $T$.
• Recursion
• For every time step $t$ from $T - 1$ to 1:
• For every state $i$:
• Calculate the backward variable $\beta_t(i)$ using the following steps:
• Set $\beta_t(i) = 0$.
• For each state $j$:
• Calculate the backward probability for state $i$ at time step $t$:
• Multiply the transition probability $A(i,j)$ from state $i$ to state $j$ by the emission probability $B(j,o_{t+1})$ of emitting symbol $o_{t+1}$ from state $j$.
• Multiply the result by the backward variable $\beta_{t+1}(j)$ at time step $t + 1$ for state $j$.
• Add the product to $\beta_t(i)$.
• Termination
• Calculate the backward probability at the first time step $t = 1$:
• Set $P(O|\lambda) = 0$.
• For each state $i$:
• Calculate the contribution of state $i$ to the overall backward probability:
• Multiply the initial state distribution $\pi(i)$ by the emission probability $B(i,o_1)$ of emitting the first observed symbol from state $i$.
• Multiply the result by the backward variable $\beta_1(i)$ at time step 1 for state $i$.
• Add the product to $P(O|\lambda)$.
• Output
• The backward probabilities $\beta_t(i)$ represent the probability of being in state $i$ at time $t$, given the observed sequence $O$.
• The backward probability $P(O|\lambda)$ represents the overall likelihood of the observed sequence $O$ given the HMM parameters $\lambda$.
```

3.1.7. Forward-Backward Algorithm

The Forward-Backward (FB) algorithm estimates the posterior & pairwise probability of a Hidden Markov Model (HMM) by fusing the forward and backward methods. Based on the observed sequence, it provides valuable information about the underlying states and their transitions.

Algorithm 3. Forward-Backward Algorithm

```
Step 1: Initialize
    a) Set the forward variable $\alpha_t(i) = \pi(i) * B(i,o_1)$ for each state $i$ at time step $t = 1$.
    b) Set the backward variable $\beta_t(i) = 1$ for each state $i$ at the last time step $T$.
Step 2: Forward Procedure
    a) For every time step $t$ from 2 to $T$:
    b) For every state $i$:
    c) Calculate the forward variable $\alpha_t(i)$ using the following steps:
    d) Set $\alpha_t(i) = 0$.
    e) For each state $j$:
    f) Multiply the transition probability $A(j,i)$ from state $j$ to state $i$ by the emission probability $B(i,o_t)$ of emitting symbol $o_t$ from state $i$.
    g) Multiply the result by the forward variable $\alpha_{t-1}(j)$ at time step $t - 1$ for state $j$.
    h) Add the product to $\alpha_t(i)$.
Step 3: Backward Procedure
    a) For each time step $t$ from $T - 1$ to 1:
    b) For each state $i$:
    c) Calculate the backward variable $\beta_t(i)$ using the following steps:
    d) Set $\beta_t(i) = 0$.
    e) For each state $j$:
    f) Multiply the transition probability $A(i,j)$ from state $i$ to state $j$ by the emission probability $B(j,o_{t+1})$ of emitting symbol $o_{t+1}$ from state $j$.
    g) Multiply the result by the backward variable $\beta_{t+1}(j)$ at time step $t + 1$ for state $j$.
    h) Add the product to $\beta_t(i)$.
```

```
Step 4: Calculate Posterior Probabilities
    a) For every time step $t$ from 1 to $T$:
    b) For every state $i$:
    c) Calculate the posterior probability $\gamma_t(i)$ using the following formula:
    d) $\gamma_t(i) = (\alpha_t(i) * \beta_t(i)) / P(O|\lambda)$, where $P(O|\lambda)$ is the overall likelihood of the observed sequence.
Step 5: Calculate Pairwise Probabilities
    a) For each time step $t$ from 1 to $T-1$:
    b) For each state $i$ and state $j$:
    c) Calculate the pairwise probability $\xi_t(i,j)$ using the following formula:
    d) $\xi_t(i,j) = \left(\alpha_t(i) * A(i,j) * B(j, o_{t+1}) * \beta_{f+1}(j)\right) / P(O|\lambda)$, where $A(i,j)$ is the transition probability from state $i$ to state $j$, and $B(j, o_{t+1})$ is the emission probability of emitting symbol $o_{t+1}$ from state $j$.
Step 6: Output
    a) The posterior probabilities $\gamma_t(i)$ represent the probability of being in state $i$ at time $t$, given the observed sequence $O$.
    b) The pairwise probabilities $\xi_t(i,j)$ represent the probability of transitioning from state $i$ to state $j$ at time $t$, given the observed sequence $O$.
```

3.1.5. Baum-welch algorithm

The Baum-Welch Algorithm, also known as the expectation-maximization (EM) algorithm, can be applied to image classification tasks using HMMs. Although HMMs are not the most common approach for image classification in modern deep learning, they can still provide a probabilistic framework for specific scenarios. In image classification, the Baum-Welch Algorithm can be used to estimate the parameters of an HMM that captures the underlying patterns in image data. The chance of viewing particular picture characteristics provided the hidden states is defined by these parameters, which include the distribution of the beginning states, the probabilities of transitioning between states, and the emission probabilities. The steps involved in using the Baum-Welch Algorithm for image classification are given in Algorithm 4:

Algorithm 4: Baum-Welch Algorithm

```
Step 1: Data Representation:
    Images must be preprocessed and represented as sequences of observed symbols or features. These symbols can be derived from various image characteristics such as colour histograms, texture descriptors, or local features.
Step 2: HMM Initialization
    The initial parameters of the HMM, the probability of transition between states and emission probabilities, and the distribution of the beginning state are initialized either randomly or based on prior knowledge.
Step 3: FB Algorithm
    The FB algorithm is applied to compute the posterior probabilities of the hidden states given the observed image features. This step involves calculating the forward probabilities (the likelihood of existing in a specific condition at a specific time instant based on the characteristics already seen) and the backward probabilities.
Step 4: Parameter Update
    The computed posterior probabilities are used to update the model parameters using the Baum-Welch update formulas. These equations find the values for the model parameters that maximize the likelihood of the observable picture characteristics.
Step 5: Iterative Process
    Steps 3 and 4 are repeated until convergence is achieved in an iterative process, or a predefined number of iterations is completed. The iterative process allows the HMM to refine its parameters to fit the observed image data better.
Step 6: Classification
    Once the HMM is trained using the Baum-Welch Algorithm, it can be used for image classification. Given a new image, the HMM can infer the expected progression of covert states using the Viterbi Algorithm or estimate the posterior probabilities of the hidden states given the observed features.
```

3.1.6. Modified Viterbi Algorithm

Dynamic programming algorithm Viterbi can be used for HMM-based image classification problems. While HMMs are not commonly used for image classification in modern deep learning, the Viterbi Algorithm can still provide insights into the process. The Viterbi Algorithm may determine the most probable concealed state sequence in a hidden Markov model (HMM) using a seen picture. Each hidden state represents a specific class or category, while the observed image corresponds to a sequence of symbols or features.

Algorithm 5: Modified Viterbi Algorithm

```
Step 1:  Initialization
    a) Initialize the trellis structure $V$ with dimensions $(N \times T)$, where $N$ is the number of states, and $T$ is the number of time steps.
    b) Initialize the backpointer matrix BP with the exact dimensions.
    c) Set the initial probabilities for the first time step:
    d) $V[i,1] = \pi[i] * b[i,1]$, for all states $Si$.
Step 2:  Recursion
    a) For each time step $t$ from 2 to $T$:
    • For each state $Si$ from 1 to $N$:
    • Compute the probability of reaching state $Si$ at time $t$:
    • $V[i,t] = max(V[j,t-1] * a[j,i] * b[i,t])$, for all states $Sj$.
    • This calculation involves multiplying the probability of being in state $Sj$ at the previous time step ($V[j,t-1]$), the transition probability from state $Sj$ to $Si(a[j,i])$, and the emission probability of observing the feature at time $t$ given state $Si(b[j,i])$.
    b) Update the backpointer matrix:
    • $BP[i,t] = argmax(V[j,t-1] * a[j,i] * b[i,t])$, for all states $Sj$.
    • The backpointer matrix stores the state $Sj$ index corresponding to the maximum probability calculation, indicating the most likely previous state leading to state $Si$ at time $t$.
Step 3:  Termination
    a) Find the maximum probability among the final time step probabilities:
    • $P^* = max(V[i,T])$, for all states $Si$.
    b) Set the final state as the one with the highest probability:
    • $S^* = argmax(V[i,T])$, for all states $Si$.
Step 4:  Backtracking
    a) Starting from step $T$'s final time, To discover the expected order of concealed states, one should follow the backpointers.
    b) Initialize an empty sequence of states.
    c) Append the state $S^*$ (the final state with the highest probability) to the sequence.
    d) For each time step $t$ from $T-1$ to 1:
    • Retrieve the index of the following state from the backpointer matrix:
    • $S^* = BP[S^*, t+1]$.
    • Append $S^*$ to the sequence.
    • Reverse the sequence to obtain the correct order of hidden states.
```

3.2. Tenacious Fish Swarm Optimization (TFSO)

The Standardized Artificial Fish Swarm Algorithm (TFSO) is an innovative optimization approach that creates a cohesive school of artificial fish exhibiting behaviour reminiscent of real fish. This algorithm showcases complex and intelligent behaviour on a macro level, achieved through the simple actions and interactions of individual fish within the group. In algorithm optimization, it demonstrates global optimum approximation through swarm behaviour, in contrast to the local optimization performed by individual algorithms. The TFSO operates as a population-based random search algorithm, commencing with generating an initial population through random selection. Subsequently, it iteratively explores the solution space to find the most optimal answer. The algorithm's strength lies in its ability to collectively move towards a global optimum by leveraging the collective intelligence of the fish school instead of relying solely on the efforts of individual fish. By emulating the behaviour of real fish, the TFSO offers a unique and practical approach to optimization problems. Each fish within the school represents an individual solution candidate, and their interactions with one another facilitate the exploration of the solution space. Through continuous adaptation and learning, the fish swarm intelligently navigates the optimization landscape, gradually converging towards the global optimum.

3.2.1. Meaning of TFSO

The state vector comprehensively represents a fish's characteristics and properties relevant to the optimization process. It encapsulates various attributes contributing to the fish's behaviour and decision-making within the swarm. These attributes may include position, velocity, fitness value, search space boundaries, and additional parameters specific to the optimization problem. TFSO create a structured representation that facilitates the algorithm's operations by expressing a fish's condition through a state vector. This vector allows for the tracking and manipulation of individual fish properties during the optimization process. It enables the algorithm to assess the fitness of each fish's solution and determine their contribution to the collective intelligence of the swarm. The initialization of the state vector involves assigning initial values to its components for each fish in the population. The specific values depend on the problem's requirements and the algorithm's implementation. These initial values can be randomly generated or based on prior problem domain knowledge. By considering the Y-dimensional nature of the problem, the state vector accommodates the necessary dimensions to capture the characteristics relevant to the optimization task. It ensures that each fish's state is adequately represented, enabling them to interact with other fish and adapt their behaviour to achieve the desired global optimum.

In the context of TFSO, let us consider that the initial fish population is represented by the variable T. This population comprises a group of artificial fish aiming to solve a Y-dimensional problem. This research work utilizes a state vector to track the condition or state of each fish within the swarm, and the same is expressed as Eq.(1).

$$P_s = (p_1, p_2, \ldots\ldots p_Y) \qquad (1)$$

$Q_s = g(P_s)$ indicates the fitness level (i.e., satisfaction with food). For every pair of fish, the $Y_{s,w} = \|P_s - P_w\|$. Euclidean distance is used to express the relationship. In the case when $s, w = 1,2,\ldots.N,$ and the various states of the fish are represented as T, P_s and P_w respectively.

The efficiency of the TFSO is influenced by three crucial factors: the fish's visual field, the step size, and the crowding level. These factors are represented by the Visual, Step, and Crowded factors, respectively. The three factors are described below:

- **Visual Factor (Visual):** The visual field of a fish refers to its perception range or the distance within which it can detect and interact with other fish. In the TFSO, the Visual factor determines the neighbourhood of influence for each fish. A larger visual field allows fish to perceive a broader range of their surroundings, facilitating the exchange of information and collective decision-making within the swarm. On the other hand, a smaller visual field restricts interactions to closer proximity, potentially limiting the exploration and convergence capabilities of the algorithm.

- **Step Factor (Step):** The step size represents the distance a fish can move in each algorithm iteration. It influences the exploration and exploitation trade-off of the swarm. A more significant step size enables fish to cover more ground in the search space, facilitating exploration and increasing the chances of finding a global optimum. However, a more significant step size may increase the likelihood of overshooting or missing optimal solutions. Conversely, a minor step enhances exploitation by allowing fine-grained adjustments and precise refinement around promising solutions. Still, it may lead to slower convergence and potential stagnation in local optima.

- **Crowded Factor (Crowded):** The crowded level denotes the density or proximity of fish within the swarm. It determines the level of competition and congestion within the population. A higher crowded level implies a more congested swarm with intense competition among fish for resources and space. This can lead to a higher likelihood of premature convergence towards suboptimal solutions. Conversely, a lower crowded level promotes more diverse exploration as fish have more space to manoeuvre and interact with a broader range of neighbours. However, crowded deficient levels may result in a scattered or disorganized swarm, potentially hindering efficient information sharing and convergence.

In the domain of leaf disease classification, this research draws an analogy between the exploration behaviour of artificial fish and the process of classifying data. Artificial fish exhibit four primary behaviours: Random, Preying, Swarming, and Following. These behaviours are employed to identify the most favourable feeding locations, which can be likened to identifying optimal classification outcomes.

- **Random Behavior:** It explores the search space without any specific pattern or predefined strategy. In this, fish randomly move and investigate different areas in the search space, allowing for broad exploration.
- **Preying Behavior:** It targets and focuses on specific areas or regions in the search space likely to contain favourable solutions. They concentrate on those areas to discover and exploit potential optimal solutions.
- **Swarming Behavior:** It involves coordination and collaboration among artificial fish. They collectively move and communicate with each other, sharing information and insights. The artificial fish can enhance their search efficiency and effectiveness by working together as a swarm.
- **Following Behavior:** It learns from the experiences and decisions of other fish in the swarm. They observe and adapt their actions based on the success and behaviour of well-performing fish. This adaptive behaviour helps improve the overall performance of the swarm.

3.2.2. Random behavior

Random behaviour is a fundamental aspect of the Artificial Fish Swarm Algorithm (TFSO) that mimics the exploration process of real fish. In TFSO, random behaviour refers to the strategy employed by artificial fish to explore the search space without any predefined pattern or specific direction. This behaviour allows the artificial fish to survey various possibilities and discover solutions unbiasedly. The random Behavior in TFSO serves as a means of broad exploration, helping the algorithm to escape local optima and find better solutions. The artificial fish can uncover new regions and evaluate their fitness by randomly selecting positions or movements within the search space. This exploration process is crucial for discovering promising areas with optimal solutions, particularly in complex and high-dimensional problem spaces.

During the random behaviour phase, artificial fish may probe various regions, sample different attributes, or explore different combinations of parameters. The randomness in their movement introduces diversity into the search process, preventing the algorithm from getting trapped in suboptimal solutions. By continuously exploring different regions, the artificial fish have a chance to encounter more favourable areas, leading to improved convergence towards the global optimum. It is important to note that random behaviour alone is not sufficient for efficient optimization. It needs to be complemented by other behaviours, such as preying, swarming, and following, to guide the search process towards optimal solutions collectively. The balance between random exploration and exploitation of known reasonable solutions is essential for achieving an effective trade-off between exploration and exploitation in TFSO. By incorporating random behaviour, TFSO introduces an element of stochasticity that enhances its robustness and adaptability. This behaviour allows the algorithm to handle uncertainties in the problem space and discover diverse solutions. The random exploration process and other behaviours contribute to the algorithm's ability to converge towards global optima and achieve effective optimization results.

Random Position Selection

An artificial fish selects a random position within the search space at each iteration. This can be expressed as Eq.(2):

$$x = x_{min} + (x_{max} - x_{min}) * rand\ () \qquad (2)$$

Where x is the randomly selected position within the search space, x_{min} and x_{max} represent the lower and upper bounds of the search space, respectively, and $rand()$ generates a random number between 0 and 1.

Algorithm 6. Random Position Selection

```
Step 1: Set the minimum value of the search space as $x_{min}$.
Step 2: Set the maximum value of the search space as $x_{max}$.
Step 3: Generate a random number between 0 and 1, and store it as random_number.
Step 4: Calculate the random position within the search space using Eq.(16)
Step 5: Return the randomly selected position $x$.
```

Random Movement

After selecting a random position, the artificial fish performs a random movement in the search space. This movement can be represented by adjusting the current position using a random displacement. Mathematically, it can be expressed as Eq.(3):

$$x_{new} = x + rand() * step_{size} \qquad (3)$$

Where x_{new} is the new position after the random movement, x is the current position, rand()

generates a random number between 0 and 1, and $step_{size}$ represents the magnitude of the random movement.

Algorithm 7. Random Movement

```
Step 1: Set $x$ as the current position of the artificial fish.
Step 2: Generate a random number between 0 and 1 using $rand()$.
Step 3: Set $step_{size}$ as the magnitude of the random movement.
Step 4: Compute the new position after the random movement using Eq.(3).
Step 5: Return to the new position $x_{new}$.
```

3.2.3. Preying behavior

The predatory behaviour of real fish inspires the Preying behaviour in TFSO. It involves the artificial fish targeting and concentrating their search efforts on regions of the search space with higher fitness values. The objective is to exploit these promising areas and increase the chances of finding optimal solutions. During the Preying behaviour, the artificial fish evaluate their fitness values at their current positions in the search space. These fitness values represent the quality or suitability of the solutions found by the fish. The fish then identify the locations with the highest fitness values among all the individuals in the swarm. These locations are considered potential prey locations, likely to contain optimal solutions.

The artificial fish update their positions to move towards the prey locations using a predefined strategy. This strategy typically involves moving towards the selected prey location and introducing a random perturbation to diversify the search. The difference between the fish's current position and the selected prey's position guides the movement towards the prey's location. The random perturbation helps explore the search space beyond the immediate vicinity of the prey location. By concentrating their search efforts on regions with higher fitness values, the Preying behaviour allows the artificial fish to focus on promising areas and increase the chances of finding optimal solutions. This behaviour enhances the exploitation of the search space while still maintaining an element of exploration through random perturbations.

Fitness Evaluation

Each artificial fish evaluates the fitness function at its current position in the search space. Let the fitness value be denoted as $f(x)$, where x represents the fish's position.

Identification of Prey Locations

Identify the locations with the highest fitness values among all the artificial fish. Let $prey_{locations} = \{x_1, x_2, \dots. x_k\}$ represent the positions of these prey locations.

Position Update

Move each artificial fish towards one of the prey locations based on a predefined strategy. The position update equation can be written as Eq.(4):

$$x_{new} = x + step_{size} * (x_{prey} - x) + random_{displacement} \quad (4)$$

where x_{new} represents the new position of the fish, x is the current position of the fish, $step_{size}$ is a parameter controlling the magnitude of the movement, x_{prey} is the position of the selected prey location, and $random_{displacement}$ represents a random perturbation or noise introduced to diversify the search.

Iteration

Repeat the above steps for a certain number of iterations or until a termination criterion is met.

Algorithm 8. Preying Behavior

```
Step 1: Initialize the swarm of artificial fish with their initial positions.
Step 2: Evaluate the fitness value for each fish's position using the fitness function.
Step 3: Set the current iteration counter to 1.
Step 4: While the termination criterion is not met or the specified number of iterations is not reached, repeat Step 5 to Step 8.
Step 5: Fitness Evaluation
    a) For each fish in the swarm, Evaluate the fitness value of the fish's current position using the fitness function.
Step 6: Identification of Prey Locations
    a) Identify the positions of the fish with the highest fitness values as the prey locations.
Step 7: Position Update
    a) Select one of the prey locations randomly from prey_locations for each fish in the swarm.
    b) Calculate the new position for the fish using Eq.(4).
```

```
      c)  Update the position of the fish to
          the new position.
Step 8:   Increment the iteration counter.
Step 9:   Return the best solution found by the
          swarm of artificial fish.
```

3.2.4. Swarming Behavior

The Swarming behaviour is a crucial characteristic of the TFSO, inspired by the collective behaviour observed in swarms of real fish. It involves the artificial fish in the swarm aligning their movements and converging towards a common position or direction. In TFSO, the Swarming behaviour promotes cooperation and information sharing among the fish, allowing them to explore the search space and find optimal solutions collectively. It leverages the principle that the collective behaviour of a group can lead to better results than individuals' isolated behaviour. The following general characteristics can describe the Swarming behaviour in TFSO:

- **Information Sharing:** The fish in the swarm share information about their positions, fitness values, and other relevant attributes. This information exchange allows the fish to gain insights into the behaviour and movements of other fish in the swarm.
- **Alignment of Movements:** The fish align their movements towards a common position or direction, which is typically determined based on the collective behaviour of the swarm. This alignment encourages cooperation and helps the fish converge towards better solutions.
- **Emergent Behavior:** Through the Swarming Behavior, the fish exhibit emergent behaviour, meaning that the collective behaviour of the swarm arises from the interactions and coordination among individual fish. The emergent behaviour can lead to complex patterns and self-organization within the swarm.
- **Exploration and Exploitation:** The Swarming behaviour balances exploration and exploitation of the search space. The fish explore new regions by following the movements of other fish and exploit promising areas by converging towards positions with higher fitness values.
- **Adaptability:** The Swarming behaviour allows the fish to adapt their movements and responses to changes in the environment or the positions of other fish. This adaptability helps the swarm to adjust its exploration and exploitation strategies dynamically.

Information Sharing

Information sharing is crucial in facilitating cooperation and coordination among the fish in the swarm. Mathematically, TFSO can represent this information sharing using matrices or sets of variables. Let's consider a swarm of N fish indexed from 1 to N. Each fish i in the swarm shares its position vector P_i and fitness value F_i with the other fish. The shared position matrix, denoted as P, is a matrix that contains the position vectors of all fish in the swarm. It can be represented as Eq.(5):

$$P = [P_1, P_2, \ldots, P_N] \qquad (5)$$

Here, P_i represents the position vector of fish s, and it typically consists of coordinates or attributes that define the position in the problem space. The matrix P represents the shared information about the positions of all fish in the swarm.

The shared fitness values can be represented using a fitness matrix or a vector. Let's denote the fitness values of the fish as $F_1, F_2, \ldots, F_N$. TFSO can represent the shared fitness values as Eq.(6):

$$F = [F_1, F_2, \ldots, F_N] \qquad (6)$$

Here, F_i represents the fitness value of fish i, which measures its performance or quality in the problem domain. The matrix F represents the shared information about the fitness values of all fish in the swarm.

By sharing this information, each fish in the swarm can learn about other fish's positions and fitness values. This information exchange enables the fish to make informed decisions, such as aligning their movements or evaluating the quality of potential solutions.

Algorithm 9: Information Sharing

```
Input:
•  Position matrix $P = [P_1, P_2, \ldots, P_N]$
•  Fitness matrix $F = [F_1, F_2, \ldots, F_N]$
Output:
•  Updated position matrix $P$
•  Updated fitness matrix $F$
Procedure:
Step 1:   Initialize an empty matrix for shared
          positions, $P_{share}$, and an empty
          matrix for shared fitness values,
          $F_{shared}$.
```

```
Step 2:  For each fish $i$ in the swarm:
         a) Create a neighbourhood set $N_i$ containing the indices of neighbouring fish based on a distance criterion.
         b) Construct a submatrix $P_i$ containing the positions of fish $i$ and its neighbours from matrix $P$.
         c) Append $P_i$ to $P_{shared}$.
         d) Construct a submatrix $F_i$ containing the fitness values of fish $i$ and its neighbours from matrix $F$.
         e) Append $F_i$ to $F_{shared}$.
Step 3:  Update the position matrix $P$ by replacing the original positions with the shared positions, i.e., $P = P_{shared}$
Step 4:  Update the fitness matrix $F$ by replacing the original fitness values with the shared fitness values, i.e., $F = F_{shared}$
Step 5:  Return the updated position matrix $P$ and fitness matrix $F$.
```

Perception of Surroundings

The perception of surroundings is an essential step that allows each fish to evaluate the positions of its neighbouring fish. This evaluation helps the fish determine its spatial relationships and enables them to make informed decisions based on the proximity of other fish. Mathematically, TFSO can represent this perception using a set-based approach. Let's consider a fish in the swarm, denoted as fish i. The fish i can perceive its surroundings by evaluating the positions of neighbouring fish within a specific distance r. TFSO denote the set of neighbouring fish as N_i, which contains the indices of fish j within the distance of r from fish i. Mathematically, TFSO can represent the neighbouring fish set N_i as Eq.(7):

$$N_i = \{j | dist(P_i, P_j) <= r\} \tag{7}$$

Here, P_i represents the position vector of fish i, and P_j represents the position vector of fish j. The function dist(.,.) represents the distance function between two positions. The condition $dist(P_i, P_j) <= r$ ensures that fish j is within the defined distance r from fish i.

In this representation, the neighbouring fish set N_i captures the fish close to fish i. The distance criterion r defines the range the fish can perceive its surroundings. By evaluating the positions of the neighbouring fish, each fish can gather information about the spatial distribution of the swarm and make decisions based on this perceived information.

Alignment of Movement

The alignment of movements is a crucial step where each fish adjusts its movement direction towards the centre of mass of its neighbouring fish. This alignment helps the fish collectively move in a coordinated manner. Mathematically, TFSO can represent this alignment using the concept of the centre of mass. Let's consider a fish in the swarm, denoted as fish i. The goal is for fish i to align its movement direction with the collective behaviour of its neighbouring fish. To achieve this, TFSO calculates the centre of mass of the neighbouring fish positions. Mathematically, the centre of mass, denoted as C, is calculated using Eq.(8).

$$C = (1/|N_i|) * \sum_{\{j \in N_i\}} P_j \tag{8}$$

Here, $|N_i|$ represents the cardinality of the neighbouring fish set N_i, which is the number of fish in the set. $\sum_{\{j \in N_i\}} P_j$ represents the sum of the neighbouring fish's position vectors.

To calculate the centre of mass, TFSO sums up the position vectors of all the neighbouring fish and divides them by the cardinality of the set. This normalization ensures that the centre of mass represents the average position of the neighbouring fish. Once the centre of mass C is determined, fish i aligns its movement direction towards this point. By doing so, each fish in the swarm aligns its movements with the overall collective behaviour, promoting cohesion and coordination among the fish.

Movement Update

The movement update step involves updating the position of each fish based on a predefined movement rule or equation. This update allows the fish to adjust its position towards the desired direction, typically towards the neighbouring fish's centre of mass. Mathematically, TFSO can represent this movement update using a specific equation. Let's consider a fish in the swarm, denoted as fish i. To update its position, fish i utilizes the following Eq.(9).

$$P_{i'} = P_i + step_{size} * (C - P_i) \tag{9}$$

Here, P_i represents the current position vector of fish i. $P_{i'}$ represents the updated position vector of fish i after the movement update. C represents the centre of mass, which is the desired direction towards which the fish wants to move. The $step_{size}$ determines the magnitude or distance of the movement towards the centre of mass.

The movement update equation calculates the difference between the current position P_i and the centre of mass C. This difference is then multiplied by the $step_{size}$ and added to the current position, resulting in the updated position P_i'. By applying this equation, each fish adjusts its position to move towards the centre of mass while considering the magnitude of movement determined by the $step_{size}$. This movement update aligns the fish with the collective behaviour and enhances the overall coordination within the swarm.

Iteration and exploration

This process involves iterating through the steps of information sharing, perception of surroundings, alignment of movements, and movement update. This iteration loop allows the swarm to explore and refine their collective behaviour over several iterations or until a termination criterion is met. Mathematically, TFSO can represent the iteration and exploration process as follows:

Algorithm 10: Iteration and Exploration

```
Input:
    • Number of iterations: $T$
Output:
    • Final swarm behaviour
Procedure:
    Step 1: Initialize the swarm of artificial fish with their initial positions and fitness values.
    Step 2: Iterate through the following steps for $t = 1$ to $T$:
        a) Perform information sharing among the fish in the swarm.
        b) Calculate the perception of surroundings for each fish, evaluating the positions of neighbouring fish within a certain distance.
        c) Align the movements of each fish towards the centre of mass of the neighbouring fish positions.
        d) Update the positions of each fish based on a predefined movement rule or equation.
    Step 3: Check the termination criterion. If the criterion is met, terminate the iteration loop and proceed to the next step. Otherwise, continue with the next iteration.
    Step 4: Return the final swarm behaviour, which includes the updated positions and fitness values of the fish after the completion of the iteration loop.
```

The iteration loop allows the swarm to explore the problem space, exchange information, and adapt their movements based on the collective behaviour of neighbouring fish. This iterative process enables the swarm to refine their solutions over time and converge towards optimal or near-optimal solutions.

3.2.5. Following behaviour

The following behaviour is a crucial aspect where fish learn from the experiences and decisions of other fish in the swarm. It allows them to observe and adapt their actions based on the success and behaviour of well-performing fish. This adaptive behaviour plays a significant role in improving the overall performance of the swarm. Mathematically, the Following Behavior can be described as utilizing the knowledge gained from labelled instances to guide the classification process for new, unseen instances. This involves leveraging the information in the labelled examples to make informed decisions for unlabeled instances. The following steps are involved in the Following Behavior:

- **Data Collection:** Gather a set of labelled instances, denoted as D, where each labelled instance is represented as (x_1, y_1). Here, x_1 represents the input features of the labelled instance, and y_1 represents its corresponding class label.

- **Learning Phase:** Utilize the labelled instances to learn and build a model or acquire knowledge about the underlying patterns and relationships between the input features and their corresponding class labels. This learning phase can involve various techniques, such as training a classifier using supervised learning algorithms.

- **Classification of Unlabeled Instances:** Once the learning phase is complete, apply the acquired knowledge to classify new, unlabeled instances.Given an unlabeled instance x_u, the goal is to assign a class label based on the learned information from the labelled instances.

- **Decision-making:** Utilize the learned model or knowledge to predict the unlabeled instances. This involves applying decision rules or mathematical functions to determine each unlabeled instance's most likely class label.

3.2.6. Digital noticeboard

A digital noticeboard is utilized in the optimization process to track and share the health status of the top synthetic fish. After each iteration, every fish checks the noticeboard to evaluate its performance relative to others in the swarm. The goal is always to maintain the best possible result on the noticeboard. This is achieved by comparing each fish's state's value with the objective function's stored value on the noticeboard. If the current state's value is higher, the corresponding entry on the noticeboard is updated to reflect the new state. The algorithm continues running iterations until all possible iterations have been completed. At this point, the state displayed on the noticeboard represents the best possible solution achieved by the swarm.

Algorithm 11: Digital Noticeboard

```
Step 1:  Initialize the noticeboard with an initial state and the corresponding value of the objective function.
Step 2:  Initialize the swarm of synthetic fish with their initial states.
Step 3:  Run iterations of the optimization process until a termination condition is met.
Step 4:  For each fish in the swarm:
         a) Evaluate the fish's current state by computing the value of the objective function.
         b) Compare the value of the current state with the value stored on the noticeboard.
         c) If the current state's value is higher:
         d) Update the noticeboard with the new state and its corresponding value.
         e) Otherwise, no changes are made to the noticeboard.
Step 5:  Check the termination condition:
         a) If the termination condition is satisfied (e.g., reaching a maximum number of iterations), proceed to the next step.
         b) Otherwise, go back to step 4 and continue with the next iteration.
Step 6:  Termination
         a) At this point, the state displayed on the noticeboard represents the best possible solution achieved by the swarm.
```

3.2.7. Varying step size

Various factors influence the optimization outcome of the algorithm, and in this research, TFSO focuses on examining the impact of the Step parameter on the convergence speed and accuracy. By adjusting the value of the Step parameter, TFSO can control how quickly the algorithm converges and the precision of the obtained solution. Increasing the Step parameter allows the artificial fish to take more significant steps and cover more ground in each iteration. This accelerates the convergence rate as the fish explores a more extensive search space. However, it is crucial to find the right balance, as excessively large or small step sizes can impede convergence. When the step size is too large, especially in the later stages of the algorithm, the artificial fish may traverse a wide range of the search space. This can result in the fish oscillating around the extreme points and struggling to reach an accurate solution.

On the other hand, using tiny step sizes can lead to low-frequency oscillations and promote high-accuracy solutions. However, convergence may be slowed down due to the limited field of view of the artificial fish, making it prone to getting trapped in local extrema. This research applies an adaptive step-size approach to address the challenges mentioned above. This adaptive step size technique helps mitigate the onset of vibrations, enhances convergence speed, and improves the accuracy of the optimization process. Specifically, the algorithm's preying behaviour, swarming behaviour, and the following behaviour incorporate different adaptive step size strategies tailored to their respective tasks.

Algorithm 12: Varying Step Size

```
Step 1:  Initialize the algorithm parameters, including the Step parameter.
Step 2:  Generate an initial population of artificial fish.
Step 3:  Evaluate the fitness of each fish in the population.
Step 4:  While the termination criterion is
```

```
          not met, do the following:
          a) Apply the preying behaviour to update the fish positions and adjust the step size adaptively.
          b) Apply the swarming behaviour to update the fish positions and adjust the step size adaptively.
          c) Apply the following behaviour to update the fish positions and adjust the step size adaptively.
          d) Evaluate the fitness of each fish in the updated population.
          e) Update the best solution found so far.
Step 5:   Terminate the algorithm and output the best solution obtained.
```

The overall process of TFSO is provided in Algorithm 13.

Algorithm 13: TFSO

```
Step 1:   Initialize Parameters:
          • Set the number of artificial fish ($N$).
          • Set the maximum number of iterations.
          • Set the search space boundaries ($x_{min}$ and $x_{max}$).
          • Set the step size for random movement.
          • Set the visual field (visual) to determine the neighborhood.
          • Set the crowded level (crowded), determining the fish density.
          • Initialize the state vector for each fish with random positions within the search space.
          • Evaluate the fitness value for each fish's initial position.
Step 2:   Start Iterations:
          • Set the current iteration counter to 1.
          • While the termination criterion is not met or the maximum number of iterations is not reached, do the following steps:
          Step 3: Preying Behavior:
          • Evaluate the fitness value for each fish's current position.
          • Identify the positions of the fish with the highest fitness values as prey locations.
          • For each fish, select one of the prey locations randomly.
          • Calculate the new position for each fish using a predefined strategy:
            ✓ Update the position based on the difference between the fish's current and the selected prey's positions.
            ✓ Introduce a random perturbation to diversify the search.
            ✓ Ensure the new position is within the search space boundaries.
          • Evaluate the fitness value for each fish's new position.
Step 4:   Swarming Behavior
          • Share information among the fish in the swarm:
            ✓ Update the shared position matrix (P) with the new positions of all fish.
          • For each fish, calculate the average position of its neighbours within the visual field.
          • Update the fish's position towards the average position:
            ✓ Adjust the fish's position based on its current and average positions.
            ✓ Ensure the new position is within the search space boundaries.
          • Evaluate the fitness value for each fish's updated position.
Step 5:   Random Behavior
          • For each fish, perform the random movement:
            ✓ Select a random position within the search space.
            ✓ Update the fish's position based on random movement.
            ✓ Ensure the new position is within the search space boundaries.
          • Evaluate the fitness value for each fish's randomly moved position.
Step 6:   Check Termination Criterion
          • If the termination criterion is met (e.g., the desired fitness value is achieved), stop the iterations and return the best solution found.
          • Otherwise, increment the iteration counter and go to Step 2.
```

3.3. Fusion of TFSO and HMM

The fusion of TFSO and HMMs offers a promising approach to enhance image classification tasks. TFSO, as a nature-inspired optimization algorithm, exhibits excellent exploration and exploitation capabilities. By leveraging the collective intelligence of a fish swarm, TFSO can effectively navigate complex search spaces and find optimal solutions. On the other hand, HMMs are well-suited for modelling sequential data and capturing temporal dependencies, making them suitable for analyzing image sequences. However, HMMs rely on accurate parameter estimation, feature selection, and model structure optimization to achieve optimal performance. By merging TFSO with HMMs, the fusion approach allows for optimizing HMM parameters, feature selection, and model structure using TFSO's optimization capabilities. TFSO can explore the parameter and feature spaces to find optimal parameter values, select informative features, and determine the optimal model structure. This fusion enables HMMs to capture the underlying patterns in image data more accurately, improving classification accuracy. The integration of TFSO and HMMs offers a synergistic combination of optimization and modelling techniques, enhancing the capabilities of both approaches and providing a robust framework for image classification tasks.

Algorithm 14: TFSO-HMM

```
Step 1:  Initialize HMM parameters (transition probabilities, emission
         probabilities)
Step 2:  Initialize TFSO parameters (swarm size, maximum iterations, etc.).
Step 3:  Generate an initial swarm of fish, each representing a potential
         solution (parameter set) for the HMM.
Step 4:  Evaluate the fitness of each fish in the swarm based on their
         corresponding HMM parameters (classification accuracy on a
         training/validation dataset).
Step 5:  Set each fish's unique best positions and fitness values to their
         initial positions and fitness values.
Step 6:  Determine the fish's best fitness as the global best position and
         fitness.
Step 7:  Repeat the following steps until convergence or a maximum number
         of iterations is reached:
      a) Update the velocity and position of each fish based on personal
         and global best positions and social interactions within the swarm.
      b) Evaluate the fitness of each fish based on the updated HMM
         parameters.
      c) Update personal best positions and fitness values if improved.
      d) Update the global best position and fitness if a fish with better
         fitness is found.
Step 8:  Extract the optimized parameter set corresponding to the fish with
         the best fitness.
Step 9:  Update the HMM parameters with the optimized parameter set.
Step 10: Classify new or unseen images using the optimized HMM parameters.
Step 11: Evaluate the classification accuracy of the optimized HMM.
```

4. ABOUT THE DATASET

The “Cotton Plant Disease Dataset” is a comprehensive and focused collection of images centred around diseases commonly affecting cotton plants. This dataset consists of a substantial number of images, totalling 26,100, providing a wealth of visual information for researchers and practitioners in plant pathology and agriculture. With a dataset size of approximately 4GB, the “Cotton Plant Disease Dataset” offers an extensive representation of four significant diseases impacting cotton plants: Aphids, Armyworms, Bacterial Blight, and Powdery Mildew. These diseases primarily manifest on the leaves of cotton plants, making the dataset highly relevant for studying leaf-based cotton plant diseases.

Additionally, the dataset includes a valuable subset of images featuring healthy cotton leaves. This inclusion allows for comparative analysis and serves as a reference point for accurately distinguishing diseased plants from healthy ones. It is important to note that the dataset does not cover diseases affecting other parts of cotton plants, such as the stem, buds, flowers, or bolls. The primary focus of this dataset is to provide a rich collection of images explicitly targeting diseases occurring on the leaves of cotton plants. Researchers and practitioners can leverage the “Cotton Plant Disease Dataset” to develop and validate advanced algorithms, machine learning models, and image recognition techniques. This dataset can enhance disease identification and classification accuracy and efficiency in cotton plants. This, in turn, enables more effective disease management strategies, leading to improved crop health and increased yield.

The availability of a substantial number of images in this dataset allows for comprehensive analysis and exploration of different disease manifestations, stages, and variations. Researchers can delve into each disease’s visual patterns and characteristics, contributing to developing robust diagnostic tools and decision support systems for cotton plant disease management. The “Cotton

Plant Disease Dataset" available at https://www.kaggle.com/datasets/dhamur/cotton-plant-disease

5. PERFORMANCE METRICS

- **Classification Accuracy (CA)** represents the effectiveness of a classification model in accurately distinguishing between healthy and diseased cotton plant leaves. It measures the correctness of the model's predictions, expressed as the ratio of correctly classified leaves to the total number of leaves in the dataset.
- **F-Measure (FM),** in the context of cotton leaf disease identification, assesses the overall performance of a classification model by combining precision and recall. It provides a single score representing the harmonic mean of precision and recall, thus capturing the model's ability to accurately identify diseased cotton leaves and minimize false positives and negatives.
- **Fowlkes-Mallows Index (FMI)** in cotton leaf disease identification quantifies the level of agreement or similarity between two techniques or algorithms in correctly identifying and classifying different diseases in cotton plant leaves. It measures the agreement between the pairwise relationships of the disease identification results obtained from the two methods, providing a single value that indicates the degree of similarity or agreement between them.
- **Matthews Correlation Coefficient (MCC)** is a measurement that quantifies the quality of a classification model in correctly identifying and classifying different types of diseases affecting cotton plant leaves. It considers true positives, false positives, and false negatives, providing a single value representing the model's overall performance.

6. RESULTS AND DISCUSSION

6.1. Assessment of Classifiers using CA and FM Performance Metrics

Figure 1 presents the CA and FM analysis of three classification algorithms: RF, SVM, and TFSO-HMM. The graph showcases the performance of these algorithms in terms of their ability to classify instances accurately and provide a balanced evaluation of precision and recall.

Classification Accuracy (CA) calculates the proportion of correct predictions out of the total predictions made, providing an understanding of how accurate the model is in its classifications. RM is an ensemble learning algorithm that combines multiple decision trees to make predictions. Each decision tree is constructed on a random subset of features and generates its prediction. The final prediction is determined by aggregating the predictions of all the trees through a voting mechanism. In the analysis, RF achieves a CA of 50.487%. This relatively low accuracy can be attributed to the limitations of individual decision trees and the possible presence of irrelevant or redundant features in the dataset. RF may struggle to capture the underlying patterns and relationships in the data, resulting in lower classification accuracy. SVM is a robust supervised learning algorithm that aims to find an optimal hyperplane in the feature space. The hyperplane separates different classes while maximizing the margin between them. SVM achieves a CA of 64.333% in the analysis. SVM's ability to identify the optimal decision boundary allows it to achieve better classification accuracy than RF. By maximizing the margin, SVM can effectively handle complex decision boundaries and capture intricate patterns in the data, leading to improved classification accuracy. TFSO-HMM is a proposed algorithm combining Hidden Markov Models (HMM) principles and Tenacious Fish Swarm Optimization (TFSO). HMM is a statistical model that can capture the underlying dynamics of sequential data. TFSO is an optimization algorithm inspired by the collective behavior of fish swarms, which emphasizes persistence and adaptability. In the analysis, TFSO-HMM achieves the highest CA of 95.379%. The integration of TFSO with HMM enables TFSO-HMM to effectively optimize the HMM parameters and capture the complex temporal patterns in the data, resulting in significantly higher classification accuracy.

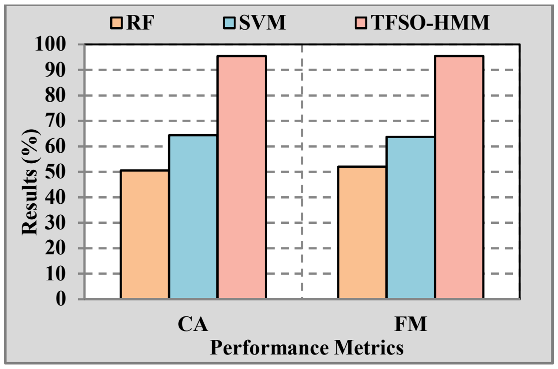

Figure 1. CA and FM

F-Measure (FM) provides a single score that balances the model's ability to make accurate positive predictions and effectively identify true positive instances. RF achieves an FM of 52.013% in the analysis. This relatively low FM score indicates that RF may have lower precision and recall than the other algorithms. RF's ensemble of decision trees might struggle with accurately identifying both positive and negative instances, resulting in a suboptimal F-measure. SVM achieves an FM of 63.757% in the analysis. The higher FM score indicates that SVM exhibits better precision and recall than RF. SVM's ability to find an optimal hyperplane allows it to effectively separate different classes and reduce false positives and negatives, resulting in a higher F-measure. TFSO-HMM achieves the highest FM of 95.487% in the analysis. Integrating TFSO with HMM in TFSO-HMM enhances precision and recall by optimizing the HMM parameters and capturing complex temporal patterns. This results in significantly better class discrimination, leading to a higher F-measure than RF and SVM.

Table 1.CA and FM Results

Classification Algorithms	CA	FM
RF	50.487	52.013
SVM	64.333	63.757
TFSO-HMM	95.379	95.487

The working mechanisms of the classification algorithms contribute to the observed CA and FM results in Figure 1. RF's ensemble learning approach may result in lower classification accuracy and F-measure due to limitations in individual decision trees. SVM's optimal hyperplane search enhances its accuracy and F-measure compared to RF. TFSO-HMM leverages the optimization process of TFSO and the modeling capabilities of HMM to achieve significantly higher classification accuracy and F-measure.

6.2. Assessment of Classifiers using FMI and MCC Performance Metrics

Figure 2 presents the analysis of two evaluation metrics, namely the Fowlkes-Mallows Index (FMI) and Matthews Correlation Coefficient (MCC), for three classification algorithms: Random Forest (RF), Support Vector Machine (SVM), and TFSO-HMM (Tenacious Fish Swarm Optimization-Based Hidden Markov Model). These metrics provide insights into the algorithms' performance regarding clustering quality and overall correlation between predicted and actual classifications.

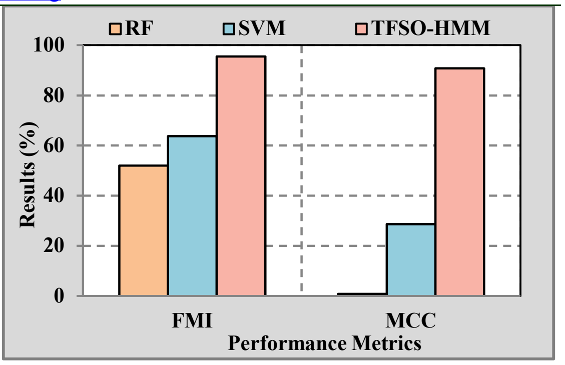

Figure 2. FMI and MCC

The FMI measures the similarity between the obtained and ground truth clusters. It quantifies the degree of agreement between predicted and true labels. Higher FMI scores indicate better clustering performance. In Figure 2, RF achieves an FMI of 52.016, SVM achieves 63.763, and TFSO-HMM achieves the highest FMI of 95.488. These results suggest that TFSO-HMM performs superior clustering by effectively capturing the underlying patterns and clustering the instances more accurately.

The MCC is a correlation coefficient that considers true positives, true negatives, false positives, and false negatives. It comprehensively measures the overall classification performance, considering both binary classification results and class imbalance. Higher MCC values indicate better classification performance. In Figure 2, RF achieves an MCC of 0.887, SVM achieves 28.669, and TFSO-HMM achieves the highest MCC of 90.757. These results indicate that TFSO-HMM outperforms both RF and SVM regarding classification accuracy and handling class imbalance.

The superior performance of TFSO-HMM in terms of FMI and MCC can be attributed to its unique working mechanism. TFSO-HMM combines the optimization capabilities of Tenacious Fish Swarm Optimization (TFSO) with the modeling capabilities of Hidden Markov Models (HMM). TFSO allows for efficient exploration of the search space, optimizing the parameters of the HMM model. This optimization process enables TFSO-HMM to accurately capture the underlying patterns in the data, leading to improved clustering quality and overall classification performance.

Figure 2 demonstrates the FMI and MCC analysis of three classification algorithms: Random Forest, Support Vector Machine, and TFSO-HMM. The results highlight the superior clustering quality and classification performance achieved by TFSO-HMM. The unique working mechanism of TFSO-HMM, combining TFSO optimization with HMM modeling, enables it to effectively capture underlying patterns and optimize the clustering process, leading to higher FMI and MCC scores. These findings showcase the potential of TFSO-HMM as a promising approach for clustering and classification tasks, providing valuable insights for researchers and practitioners in machine learning.

Table 2.FMI and MCC Results

Classification Algorithms	FMI	MCC
RF	52.016	0.887
SVM	63.763	28.669
TFSO-HMM	95.488	90.757

7. CONCLUSION

The Tenacious Fish Swarm Optimization-based Hidden Markov Model (TFSO-HMM) approach presented in this research offers a robust and innovative solution for augmented and accurate cotton leaf disease identification and yield prediction. The proposed method was evaluated on the widely recognized and comprehensive "Cotton Plant Disease Dataset," which served as a benchmark for assessing the performance of the TFSO-HMM framework and existing classification algorithms. Through rigorous experimentation on the "Cotton Plant Disease Dataset," the superior performance of the TFSO-HMM method was demonstrated, surpassing existing approaches in terms of augmented accuracy and reliable predictions. By leveraging the unique characteristics of TFSO and HMM, the proposed framework exhibited enhanced capabilities in optimizing the classification process and capturing temporal dependencies in disease progression and yield prediction. Utilizing the "Cotton Plant Disease Dataset" for evaluation enhances the credibility of the research findings and highlights the practical applicability of the TFSO-HMM approach in real-world scenarios. By leveraging this dataset, which contains diverse instances of cotton leaf diseases, the proposed method was subjected to thorough analysis and comparison with existing algorithms, providing valuable insights into its effectiveness and potential impact on disease management and yield optimization. The outcomes of this research contribute significantly to the advancement of agricultural practices, empowering farmers, agronomists, and decision-makers with a reliable tool for early disease detection and accurate yield prediction in cotton plants. The proposed TFSO-HMM framework can potentially mitigate losses and promote sustainable agriculture by maximizing crop yield through timely interventions and informed decision-making.

REFERENCES

[1] Fallahi, A., Mahnam, M., Niaki, S.T.A.: A discrete differential evolution with local search particle swarm optimization to direct angle and aperture optimization in IMRT treatment planning problem. Appl. Soft Comput. 131, 109798 (2022). https://doi.org/10.1016/j.asoc.2022.109798.

[2] Venkatesh, J., Ramasamy, K.K., Aruna, M., Praveen Kumar Rao, K., Sasikala, N., Nasani, K.: EAgri: Smart Agriculture Monitoring Scheme using Machine Learning Strategies. In: Proceedings of the 2022 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems, ICSES 2022 (2022). https://doi.org/10.1109/ICSES55317.2022.9914216.

[3] Vijayalakshmi, B., Ramkumar, C., Niveda, S., Pandian, S.C.: Smart Pest Control System in Agriculture. In: IEEE International Conference on Intelligent Techniques in Control, Optimization and Signal Processing, INCOS 2019 (2019). https://doi.org/10.1109/INCOS45849.2019.8951351.

[4] Jumat, M.H., Nazmudeen, M.S., Wan, A.T.: Smart farm prototype for plant disease detection, diagnosis & treatment using IoT device in a greenhouse. In: IET Conference Publications (2018). https://doi.org/10.1049/cp.2018.1545.

[5] Ivliev, E., Demchenko, V., Obukhov, P.: Automatic Monitoring of Smart Greenhouse Parameters and Detection of Plant Diseases by Neural Networks, https://www.scopus.com/inward/record.uri?eid=2-s2.0-85117457142&doi=10.1007%2F978-981-16-3844-2_4&partnerID=40&md5=d352869110dacb5b20dbb305ccd5d8ff, (2022). https://doi.org/10.1007/978-981-16-3844-2_4.

[6] Marcu, I., Suciu, G., Bălăceanu, C., Drăgulinescu, A.M., Dobrea, M.A.: IoT Solution for Plant Monitoring in Smart Agriculture. In: SIITME 2019 - 2019 IEEE 25th International Symposium for Design and Technology in Electronic Packaging, Proceedings. pp. 194–197 (2019). https://doi.org/10.1109/SIITME47687.2019.8990798.

[7] Spyroglou, I., Rybka, K., Czembor, P., Piaskowska, D., Pernisová, M., Matysik, P.: Higher alterations in leaf fluorescence parameters of wheat cultivars predict more extensive necrosis in response to Zymoseptoria tritici. Plant Pathol. 71, 1454–1466 (2022). https://doi.org/10.1111/ppa.13569.

[8] Balram, G., Kumar, K.K.: Crop Field Monitoring and Disease Detection of Plants in Smart Agriculture using Internet of Things. Int. J. Adv. Comput. Sci. Appl. 13, 819–826 (2022). https://doi.org/10.14569/IJACSA.2022.0130795.

[9] Saberi Anari, M.: A Hybrid Model for Leaf Diseases Classification Based on the Modified Deep Transfer Learning and Ensemble Approach for Agricultural AIoT-Based Monitoring. Comput. Intell. Neurosci. 2022, (2022). https://doi.org/10.1155/2022/6504616.

[10] Gupta, D., Sharma, P., Choudhary, K., Gupta, K., Chawla, R., Khanna, A., Albuquerque, V.H.C. d.: Artificial plant optimization algorithm to detect infected leaves using machine learning. Expert Syst. 38, e12501 (2021). https://doi.org/10.1111/exsy.12501.

[11] Ramkumar, J., Vadivel, R.: Improved frog leap inspired protocol (IFLIP) – for routing in cognitive radio ad hoc networks (CRAHN). World J. Eng. 15, 306–311 (2018). https://doi.org/10.1108/WJE-08-2017-0260.

[12] Ramkumar, J., Vadivel, R.: Performance Modeling of Bio-Inspired Routing Protocols in Cognitive Radio Ad Hoc Network to Reduce End-to-End Delay. Int. J. Intell. Eng. Syst. 12, 221–231 (2019). https://doi.org/10.22266/ijies2019.0228.22.

[13] Jaganathan, R., Ramasamy, V.: Performance modeling of bio-inspired routing protocols in Cognitive Radio Ad Hoc Network to reduce end-to-end delay. Int. J. Intell. Eng. Syst. 12, 221–231 (2019). https://doi.org/10.22266/IJIES2019.0228.22.

[14] Jaganathan, R., Vadivel, R.: Intelligent Fish Swarm Inspired Protocol (IFSIP) for Dynamic Ideal Routing in Cognitive Radio Ad-Hoc Networks. Int. J. Comput. Digit. Syst. 10, 1063–1074 (2021). https://doi.org/10.12785/ijcds/100196.

[15] Vadivel, R., Ramkumar, J.: QoS-enabled improved cuckoo search-inspired protocol (ICSIP) for IoT-based healthcare applications. Inc. Internet Things Healthc. Appl. Wearable Devices. 109–121 (2019). https://doi.org/10.4018/978-1-7998-1090-2.ch006.

[16] Ramkumar, J., Vadivel, R.: CSIP—cuckoo search inspired protocol for routing in cognitive radio ad hoc networks. In: Advances in Intelligent Systems and Computing. pp. 145–153. Springer Verlag (2017). https://doi.org/10.1007/978-981-10-3874-7_14.

[17] Lingaraj, M., Sugumar, T.N., Felix, C.S., Ramkumar, J.: Query aware routing protocol for mobility enabled wireless sensor network. Int. J. Comput. Networks Appl. 8, 258–267 (2021). https://doi.org/10.22247/ijcna/2021/209192.

[18] Ramkumar, J., Vadivel, R.: Whale optimization routing protocol for minimizing energy consumption in cognitive radio wireless sensor network. Int. J. Comput. Networks Appl. 8, 455–464 (2021). https://doi.org/10.22247/ijcna/2021/209711.

[19] Ramkumar, J., Samson Dinakaran, S., Lingaraj, M., Boopalan, S., Narasimhan, B.: IoT-Based Kalman Filtering and Particle Swarm Optimization for Detecting Skin Lesion. Presented at the (2023). https://doi.org/10.1007/978-981-19-8353-5_2.

[20] J, R.: Meticulous Elephant Herding Optimization based Protocol for Detecting Intrusions in Cognitive Radio Ad Hoc Networks. Int. J. Emerg. Trends Eng. Res. 8, 4548–4554 (2020). https://doi.org/10.30534/ijeter/2020/82882020.

[21] Ramkumar, J., Vadivel, R.: Multi-Adaptive Routing Protocol for Internet of Things based Ad-hoc Networks. Wirel. Pers. Commun. 120, 887–909 (2021). https://doi.org/10.1007/s11277-021-08495-z.

[22] Ramkumar, J.: Bee inspired secured protocol for routing in cognitive radio ad hoc networks. Indian J. Sci. Technol. 13, 2159–2169 (2020). https://doi.org/10.17485/ijst/v13i30.1152.

[23] Ramkumar, J., Kumuthini, C., Narasimhan, B., Boopalan, S.: Energy Consumption Minimization in Cognitive Radio Mobile Ad-Hoc Networks using Enriched Ad-hoc On-demand Distance Vector Protocol. 2022 Int. Conf. Adv. Comput. Technol. Appl. ICACTA 2022. 1–6 (2022). https://doi.org/10.1109/ICACTA54488.2022.9752899.

[24] Menakadevi, P., Ramkumar, J.: Robust Optimization Based Extreme Learning Machine for Sentiment Analysis in Big Data. 2022 Int. Conf. Adv. Comput. Technol. Appl. ICACTA 2022. 1–5 (2022). https://doi.org/10.1109/ICACTA54488.2022.9

753203.
[25] Ramkumar, J., Jeen Marseline, K.S., Medhunhashini, D.R.: Relentless Firefly Optimization-Based Routing Protocol (RFORP) for Securing Fintech Data in IoT-Based Ad-Hoc Networks. Int. J. Comput. Networks Appl. 10, 668 (2023). https://doi.org/10.22247/IJCNA/2023/223319.
[26] Jayaraj, D., Ramkumar, J., Lingaraj, M., Sureshkumar, B.: AFSORP: Adaptive Fish Swarm Optimization-Based Routing Protocol for Mobility Enabled Wireless Sensor Network. Int. J. Comput. Networks Appl. 10, 119–129 (2023). https://doi.org/10.22247/ijcna/2023/218516.
[27] Mani, L., Arumugam, S., Jaganathan, R.: Performance Enhancement of Wireless Sensor Network Using Feisty Particle Swarm Optimization Protocol. ACM Int. Conf. Proceeding Ser. 1–5 (2022). https://doi.org/10.1145/3590837.3590907.
[28] Sarma, K.K., Das, K.K., Mishra, V., Bhuiya, S., Kaplun, D.: Learning Aided System for Agriculture Monitoring Designed Using Image Processing and IoT-CNN. IEEE Access. 10, 41525–41536 (2022). https://doi.org/10.1109/ACCESS.2022.3167061.
[29] Kovvuri, R.R., Kaushik, A., Yadav, S.: Disruptive technologies for smart farming in developing countries: Tomato leaf disease recognition systems based on machine learning. Electron. J. Inf. Syst. Dev. Ctries. n/a, e12276 (2023). https://doi.org/10.1002/isd2.12276.
[30] Al-gaashani, M.S.A.M., Shang, F., Muthanna, M.S.A., Khayyat, M., Abd El-Latif, A.A.: Tomato leaf disease classification by exploiting transfer learning and feature concatenation. IET Image Process. 16, 913–925 (2022). https://doi.org/10.1049/ipr2.12397.
[31] Cengil, E., Çınar, A.: Hybrid convolutional neural network based classification of bacterial, viral, and fungal diseases on tomato leaf images. Concurr. Comput. Pract. Exp. 34, e6617 (2022). https://doi.org/10.1002/cpe.6617.
[32] Kumar K, K., Kannan, E.: Detection of rice plant disease using AdaBoostSVM classifier. Agron. J. 114, 2213–2229 (2022). https://doi.org/10.1002/agj2.21070.
[33] Abayomi-Alli, O.O., Damaševičius, R., Misra, S., Maskeliūnas, R.: Cassava disease recognition from low-quality images using enhanced data augmentation model and deep learning. Expert Syst. 38, e12746 (2021). https://doi.org/10.1111/exsy.12746.
[34] Ravi, V., Acharya, V., Pham, T.D.: Attention deep learning-based large-scale learning classifier for Cassava leaf disease classification. Expert Syst. 39, e12862 (2022). https://doi.org/10.1111/exsy.12862.
[35] Ganesan, G., Chinnappan, J.: Hybridization of ResNet with YOLO classifier for automated paddy leaf disease recognition: An optimized model. J. F. Robot. 39, 1087–1111 (2022). https://doi.org/10.1002/rob.22089.
[36] Javidan, S.M., Banakar, A., Vakilian, K.A., Ampatzidis, Y.: Tomato leaf diseases classification using image processing and weighted ensemble learning. Agron. J. n/a, (2023). https://doi.org/10.1002/agj2.21293.
[37] Tang, Y., Zhao, J., Huang, H., Zhuang, J., Tan, Z., Hou, C., Chen, W., Ren, J.: Multiscale voting mechanism for rice leaf disease recognition under natural field conditions. Int. J. Intell. Syst. 37, 12169–12191 (2022). https://doi.org/10.1002/int.23081.
[38] Ahmed, I., Yadav, P.K.: Plant disease detection using machine learning approaches. Expert Syst. 40, e13136 (2022). https://doi.org/10.1111/exsy.13136.
[39] Sahu, K., Minz, S.: Self-adaptive-deer hunting optimization-based optimal weighted features and hybrid classifier for automated disease detection in plant leaves. Expert Syst. 39, e12982 (2022). https://doi.org/10.1111/exsy.12982.
[40] Pavlov, Y.L.: Random forests. Random For. 45, 1–122 (2019). https://doi.org/10.4324/9781003109396-5.
[41] Cortes, C., Vapnik, V.: Support-vector networks. Mach. Learn. 20, 273–297 (1995). https://doi.org/10.1007/bf00994018.