# DEEP LEARNING MODELS DOCUMENTATION

---

## Fruit Detection Model

**Introduction:**

Objective: The primary objective of the model is to automate the classification of fruit images into specific categories. This model addresses the challenge of efficiently and accurately categorizing fruit types based on visual characteristics, which is useful for applications such as quality control in agriculture, inventory management in supermarkets, and automated sorting in food processing industries.

**Scope:**

The model is designed to classify images of fruits into a set of predefined categories. It handles images resized to 100x100 pixels and aims to achieve high accuracy in classification tasks. However, the model's performance may be limited by factors such as variations in lighting, background noise, and the diversity of fruit types in the dataset. It may also struggle with images of poor quality or those containing multiple overlapping objects.

**Model Overview:**

**Architecture:**

The model is a Convolutional Neural Network (CNN) specifically designed for image classification tasks.

It includes:

Input Layer: Takes input images of size 100x100 pixels with 3 color channels (RGB).

Preprocessing Layer: Applies resizing and normalization to the images.

Convolutional Layers: Consists of five convolutional layers that extract features from the images. The filter sizes are 32, 32, 64, 64, and 128, with each layer followed by a ReLU activation function to introduce non-linearity and improve feature representation.

MaxPooling Layers: Used after each convolutional layer to reduce the spatial dimensions of the feature maps, helping to down-sample and retain important features.

Flattening Layer: Converts the 2D feature maps into a 1D vector to feed into fully connected layers.

Dense Layers: Includes a dense layer with 512 units and ReLU activation, followed by a dropout layer (with a dropout rate of 0.25) to prevent overfitting. The final dense layer uses softmax activation to output class probabilities.

**Data:**

Dataset Description: The dataset consists of images collected from a fruit classification dataset, which includes various types of fruits. It is divided into three subsets: training, validation, and testing. The dataset features a diverse range of fruit images to ensure the model generalizes well across different fruit types and conditions.

Preprocessing: The images are resized to 100x100 pixels to ensure consistency in input size. Normalization is applied to scale pixel values to a range between 0 and 1. Data augmentation techniques, such as random rotations, flips, and brightness adjustments, are employed to enhance model robustness. The dataset is split into training (70%), validation (15%), and test (15%) sets to evaluate the model's performance accurately.

**Training:**

Hyperparameters: The model is trained using the following hyperparameters:

Learning Rate: Adjusted dynamically using the Adam optimizer.

Batch Size: Set to 32 images per batch.

Number of Epochs: The model is trained for 10 epochs.

Optimizer: The Adam optimizer is used, which adapts the learning rate during training to improve convergence.

Loss Function: Sparse categorical cross-entropy is utilized, suitable for multi-class classification problems where labels are provided as integers.

Training Procedure: The training involves feeding the training dataset into the model and validating its performance on a separate validation set. Techniques such as dropout and data augmentation are used to prevent overfitting and enhance generalization.

**Evaluation:**

Metrics: The model's performance is evaluated using accuracy, which measures the proportion of correctly classified images out of the total images.

Results: Evaluation results include the model's accuracy on training, validation, and test sets, demonstrating its ability to generalize to unseen data.

Comparison: The model's performance is compared with baseline models or other existing methods to highlight its effectiveness and improvements.

**Implementation:**

**Code:**

https://github.com/jimitchavdadev/SIH_Deep_Learning_Models/blob/main/multilabel-fruits-classification-cnn-keras.ipynb

Dependencies: The model is built using TensorFlow and Keras libraries. The specific versions used are noted in the repository to ensure compatibility.

**Usage:**

Inference: To use the model for inference, images should be resized to 100x100 pixels and normalized. The model will output class probabilities, with the highest probability indicating the predicted class.

Example: A simple example of how to use the model for inference involves loading an image, preprocessing it, and running it through the model to obtain and interpret the prediction.

**Limitations and Considerations:**

Assumptions: The model assumes consistent image quality and resolution during inference. It also presumes that the fruit types in the test dataset are represented in the training dataset.

Limitations: The model may encounter difficulties with images that have significant variations in lighting, background, or occlusions. It may also struggle with fruit types not included in the training dataset.

**Future Work:**

Improvements: Future work could focus on expanding the dataset to include more diverse fruit types, experimenting with advanced architectures (such as deeper CNNs or transfer learning), and incorporating additional data augmentation techniques to further improve model performance.

Open Questions: Areas for further research include exploring methods to enhance the model's robustness to variations in image quality and environmental conditions, as well as addressing potential challenges with unseen fruit types.

**References:**

https://www.mdpi.com/2078-2489/15/9/517 - RMN

---

# Crop Classification Model

**Introduction:**

The objective of the model is to classify spectral data from agricultural samples into distinct categories, such as different crop types and growth stages. This classification aids in precise agricultural management by providing insights into crop conditions and helping with yield predictions. The model addresses the problem of efficiently analyzing complex spectral data to make informed decisions in crop management and research.

The model is designed to handle spectral data from agricultural contexts and is capable of performing dimensionality reduction, feature extraction, and classification. It can categorize different crop types and growth stages based on the spectral features provided. However, the model may have limitations in generalizing to unseen data or different types of spectral data not included in the training set. It is also dependent on the quality and representativeness of the input data.

**Model Overview:**

Architecture: The model architecture comprises several key components:

Data Preprocessing: Involves normalization of spectral data to bring all measurements to a consistent scale. Non-Negative Matrix Factorization (NMF) is used for dimensionality reduction, which helps in extracting the most significant features from the data.

Feature Extraction: Principal Component Analysis (PCA) or similar techniques might be used on the components obtained from NMF to further reduce dimensionality and enhance feature representation.

Classification: Machine learning classifiers such as Random Forest and XGBoost are employed for predicting crop types and growth stages based on the processed spectral features.

Statistical Analysis: Techniques like ANOVA are used for assessing the significance of features, and confusion matrices are used for evaluating model performance.

**Data:**

Dataset Description: The dataset consists of spectral measurements obtained from agricultural samples, including various crops at different growth stages.

Key characteristics of the dataset include:

Source: Collected from agricultural research institutions or field studies.

Size: Comprising thousands of samples with spectral readings at various wavelengths.

Key Characteristics: Features include wavelength measurements, crop type, growth stage, Agro-Ecological Zone (AEZ), month, and weight (wgt).

Preprocessing: Data preprocessing involves several steps:

Normalization: Scaling the spectral data to a uniform range to ensure consistency across samples.

Dimensionality Reduction: Applying NMF to reduce the number of features while retaining significant information.

Splitting: Dividing the data into training, validation, and test sets to evaluate the model's performance.

**Training:**

**Hyperparameters:**

Learning Rate: For optimizing the training process.

Batch Size: Number of samples processed before the model's internal parameters are updated.

Number of Epochs: Total number of iterations over the entire training dataset.

**Optimizer:**

Random Forest: Implicit optimization during training.

XGBoost: Uses gradient boosting with parameters like learning rate and number of estimators.

**Loss Function:**

Classification: Cross-entropy loss is used to measure the performance of the classifier by comparing predicted probabilities to actual class labels.

**Training Procedure:**

Cross-Validation: Applied to tune hyperparameters and avoid overfitting.

Transfer Learning: Not utilized, but fine-tuning of hyperparameters is performed based on validation results.

**8. Limitations and Considerations**

**Assumptions:**

Assumes that the spectral data provided during inference is similar to the data used for training.

**Limitations:**

The model may not generalize well to new or unseen data outside the scope of the training dataset. It also depends on the quality of input data and the representativeness of training samples.

**Code:**

https://github.com/jimitchavdadev/SIH_Deep_Learning_Models/blob/main/crop-classification-xgboost-and-nmf.ipynb

**Future Work**

**Improvements:**

Consider integrating more diverse datasets, exploring deep learning models for potentially better performance, and enhancing preprocessing techniques.

**References:**

https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7780460 – YoLO

https://arxiv.org/abs/1608.06993 – CNN

---

# Apple Disease Model

**Introduction:**

The primary objective of this deep learning model is to classify apple images based on the presence of various diseases. By leveraging a convolutional neural network (CNN) architecture, the model aims to provide an automated and accurate method for diagnosing apple diseases, which can significantly aid farmers and agricultural professionals in monitoring and managing apple orchards. Accurate disease classification enables timely interventions, reducing crop loss and improving yield.

This model is designed to classify apple images into predefined disease categories using a pre-trained EfficientNetB3 base model, followed by custom classification layers. The capabilities of the model include handling image classification tasks and providing disease predictions based on visual features. However, its limitations include reliance on the quality and diversity of the training data, potential challenges in generalizing to unseen disease types or images with substantial variations, and sensitivity to image quality and conditions.

**Model Overview:**

Architecture: The model is based on a convolutional neural network (CNN) architecture, specifically using EfficientNetB3 as the backbone. EfficientNetB3 is a highly efficient and scalable network designed for image classification tasks. Key components of the architecture include:

Base Model: EfficientNetB3, which is pre-trained on the ImageNet dataset, consists of multiple convolutional layers, batch normalization, and activation functions (e.g., ReLU). The base model extracts high-level features from input images.

Batch Normalization: Applied after the base model output to normalize activations and improve training stability.

Dense Layers: Two fully connected layers with 512 and 128 units, respectively, using sigmoid activation functions. These layers are designed to learn complex features and patterns from the base model's output.

Dropout Layers: Introduced with dropout rates of 0.5 and 0.7 to reduce overfitting by randomly dropping units during training.

Output Layer: A Dense layer with softmax activation to produce probabilities for each disease class.

Input Layer: Image input of size 300x300x3.

EfficientNetB3 Base Model: Feature extraction with convolutional layers.

Batch Normalization Layer: Stabilization of feature maps.

Dense Layers: Fully connected layers with activation functions.

Dropout Layers: Regularization to prevent overfitting.

Output Layer: Classification into disease categories using softmax.

**Data:**

Dataset Description: The dataset used consists of apple images categorized into various disease classes. The dataset is sourced from Kaggle and includes both training and testing subsets. It comprises thousands of images with different apple diseases, providing a diverse representation of the target classes. Key characteristics include image size (300x300 pixels), resolution, and the diversity of disease types.

Preprocessing: Data preprocessing involves several steps:

Resizing: Images are resized to 300x300 pixels to match the input requirements of the EfficientNetB3 model.

Normalization: Pixel values are scaled to the range [0, 1] by dividing by 255 to standardize input data.

Augmentation: Although the dataset does not use extensive augmentation, basic preprocessing includes splitting the data into training and validation subsets using an 80-20 split to evaluate model performance.

**Training:**

**Hyperparameters:**

Learning Rate: 0.0001

Batch Size: 32

Number of Epochs: 100

Optimizer: The Adam optimizer is used with a learning rate of 0.0001. Adam is chosen for its adaptive learning rate capabilities and effective handling of sparse gradients.

Loss Function: Categorical cross-entropy is used as the loss function, appropriate for multi-class classification problems where each image belongs to one of several categories.

Training Procedure: The model is trained using transfer learning with the EfficientNetB3 base model. The process involves:

Loading Pre-trained Weights: EfficientNetB3 weights are initialized from the ImageNet dataset.

Adding Custom Layers: New layers are appended for classification.

Callbacks: ModelCheckpoint and EarlyStopping callbacks are used to save the best model based on validation accuracy and stop training early if no improvement is observed.

**Evaluation:**

Metrics: The model's performance is evaluated using accuracy, precision, recall, F1 score, and a confusion matrix. These metrics provide insights into the model's ability to classify apple diseases correctly.

Results: Evaluation results include performance metrics on training, validation, and test sets. Key findings are presented in the form of accuracy scores, precision-recall trade-offs, and a confusion matrix showing classification performance across different disease classes.

Comparison: If applicable, the model's performance is compared with other models or baseline methods to assess its effectiveness in disease classification.

**Implementation:**

Code:

https://github.com/jimitchavdadev/SIH_Deep_Learning_Models/blob/main/apple-disease-classification-effb3.ipynb

Dependencies: TensorFlow, Keras

**Limitations and Considerations**

Assumptions: The model assumes that the input images are properly preprocessed and belong to the predefined categories in the training data. It also assumes that the dataset used for training is representative of the real-world scenarios in which the model will be deployed.

Limitations: The model may struggle with images that are not well-represented in the training dataset or with significant variations in image quality. It may also face challenges in generalizing to new or rare disease types not present in the training data.

**Future Work**

Improvements: Future work could involve expanding the dataset to include more diverse images and disease types, experimenting with other CNN architectures or advanced techniques such as attention mechanisms, and optimizing the model for real-time deployment in agricultural settings.

Open Questions: Research could focus on improving the model's ability to generalize to new disease types, handling variations in image quality, and integrating the model into practical applications for disease detection in the field.

**References**

https://arxiv.org/abs/1905.11946 - CNN

# FERTILIZER PREDICTION MODEL

**Introduction:**

The objective of this model is to automate the prediction of appropriate fertilizers based on environmental and crop-related parameters. By leveraging machine learning models, this solution aims to assist farmers and agronomists in choosing the right fertilizer for their crops, optimizing yields, and ensuring efficient use of resources. This system has applications in precision agriculture, automated farm management systems, and decision-support systems in the agricultural industry.

**Scope:**

The model predicts the fertilizer type based on features like soil type, crop type, temperature, humidity, and nutrient content (Nitrogen, Phosphorus, Potassium). It processes structured data, aiming for high accuracy in predicting the most suitable fertilizer for different crops under various environmental conditions. However, the model's effectiveness may vary based on the quality and variety of input data, and it may struggle with extreme conditions or rare combinations of features that are not well-represented in the dataset.

**Model Overview:**

**Architecture:**

This system includes various machine learning models (Support Vector Machines, Random Forest, XGBoost, MLP, and a custom neural network) to classify the type of fertilizer based on input features. The models are compared for performance on both test data and the full dataset.

Key layers in the pipeline:

1. **Preprocessing**: The data is cleaned, and categorical variables (Soil Type, Crop Type, and Fertilizer) are label-encoded. Continuous features (Temperature, Humidity, etc.) are standardized using StandardScaler for certain models.
2. **Models**:
    - **SVM Pipeline**: A Support Vector Classifier (SVC) with probability estimation and standardized inputs.
    - **Random Forest Pipeline**: Random Forest Classifier with tree-based feature selection.
    - **XGBoost Pipeline**: XGBoost Classifier, optimized for gradient-boosting trees.

- **MLP (Multi-layer Perceptron)**: Implemented using MLPRegressor from sklearn, with two hidden layers (64 and 32 neurons).
- **Custom Neural Network**: A basic two-layer neural network implemented from scratch using numpy. It includes ReLU activation for the hidden layer and softmax for the output layer.

**Data:**

Dataset Description:

The dataset consists of various environmental and crop-related features (e.g., temperature, humidity, crop type, soil type, and nutrient content like nitrogen, phosphorus, and potassium) and the target variable is the type of fertilizer to be recommended. The dataset is balanced and includes both categorical and continuous features.

Preprocessing:

**Rescaling**: Features like temperature, humidity, and nutrient content are scaled between 0 and 1 using standardization techniques.

**Encoding**: Label encoding is applied to categorical features such as soil type, crop type, and fertilizer type.

**Train-Test Split**: The dataset is split into 80% for training and 20% for testing to ensure accurate performance evaluation on unseen data.

Training:

Hyperparameters:

**Learning Rate**: Managed dynamically through optimizers like Adam (for the MLP and custom neural network models).

**Batch Size**: Set to 32 for training models.

**Number of Epochs**: Models like MLP and custom neural networks are trained for 10 epochs.

**Optimizer**:

**Adam**: Used for MLP and custom neural network models, which adjusts the learning rate dynamically to improve convergence.

**Random Forest and SVM**: Default optimizers are used as they do not require gradient-based optimization.

**Loss Function**: Sparse categorical cross-entropy is used for classification tasks. Mean squared error is used for the MLP regressor model.

Training Procedure:

**Data Augmentation**: Not applicable for this structured data. However, SMOTE (Synthetic Minority Oversampling Technique) could be used if the dataset were imbalanced, but is commented out for this case.

**Regularization**: Dropout layers are applied in the MLP model to prevent overfitting.

**Model Training**: Models are trained on the training data with validation metrics evaluated periodically to monitor overfitting.

Evaluation:

Metrics:

**Accuracy**: The primary metric used to evaluate the performance of models. It is calculated as the percentage of correct predictions over the total predictions made.

**Confusion Matrix**: Used to visualize classification performance for each class.

**Loss**: For MLP and the custom neural network, the cross-entropy loss and accuracy are monitored during training.

**Results:**

**Test Data Accuracy**: Accuracy is calculated for each model on a 20% test set.

**Overall Data Accuracy**: Accuracy is calculated on the entire dataset to understand the model's generalization.

Model Performance:

**SVM Model**:

Achieved test accuracy of ~95% on the test data.

Confusion matrix shows the distribution of true vs predicted classes.

**Random Forest Model**:

Achieved test accuracy of ~97%.

Robust feature importance and accurate predictions.

**XGBoost Model**:

Achieved test accuracy of ~98%.

Performance enhanced with gradient-boosting techniques.

**MLP Model**:

MLP model with two hidden layers (64 and 32 units).

Trained with adam optimizer, dropout, and ReLU activation.

Achieved test accuracy comparable to tree-based models.

**Custom Neural Network**:

A basic 2-layer network implemented using numpy.

Achieved reasonable accuracy but requires more hyperparameter tuning and optimization.

Model Comparison:

XGBoost performed best on both the test data and overall dataset, followed by Random Forest and SVM.

The MLP model achieved competitive accuracy, while the custom neural network showed promise but required further optimization.

**Conclusion:**

The fertilizer prediction model successfully predicts fertilizer types based on crop, soil, and environmental conditions with high accuracy. The use of multiple models allows for comparison, and XGBoost proves to be the most effective in this case. Further tuning, especially in neural networks, could improve performance in real-world scenarios. The model can be extended with additional features and more diverse data for improved accuracy and generalization across different regions and farming conditions.

**Implementation:**

https://github.com/jimitchavdadev/SIH_Deep_Learning_Models/blob/main/fertilizer.ipynb