National College of Ireland

Configuration Manual for Improving Crop Yield Forecasting with Hybrid CNN-GRU and ARIMA-GRU Models

MSc Research Project
MSc Data Analytics

Sameer Ahamed Sahul Hameed
Student ID: 23326751

School of Computing
National College of Ireland

Supervisor: Prof. John Kelly

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: …… SAMEER AHAMED SAHUL HAMEED ……

Student ID: …… 23326751 ……

Programme: … MSc Data Analytics …… **Year:** 2024 - 2025

Module: …… MSc Research Project ……

Lecturer: …… Prof. JOHN KELLY ……

Submission Due Date: …… 11/08/2025 ……

Project Title: Improving Crop Yield Forecasting with Hybrid CNN-GRU and ARIMA-GRU Models

Word Count: …… 661 …… **Page Count:** …… 7 ……

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: …… Sameer Ahamed Sahul Hameed ……

Date: …… 11/08/2025 ……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual for Improving Crop Yield Forecasting with Hybrid CNN-GRU and ARIMA-GRU Models

Sameer Ahamed Sahul Hameed
23326751

1. Introduction

This manual provides step-by-step instructions for setting up the environment, dependencies, and configurations needed to replicate and run the research study titled ***Improving Crop Yield Forecasting with Hybrid CNN-GRU and ARIMA-GRU Models***. The models implemented in this research include several machine learning models (Lasso Regression, KNN, Decision Tree), deep learning architectures (DNN, LSTM), and a hybrid CNN-GRU and ARIMA-GRU model.

The study utilizes crop yield data consisting of various agricultural features (rainfall, temperature, crop type, etc.) and aims to enhance prediction accuracy using advanced hybrid modelling techniques.

2. System Hardware Requirements

Before beginning the setup, ensure the system meets the following hardware requirements:

- **Operating System**: Ubuntu 20.04+ or Windows 10/11 (Linux-based systems preferred)
- **Processor**: Intel Core i5 or higher (Quad-core or better)
- **RAM**: Minimum 8 GB (16 GB or more recommended)
- **GPU**: NVIDIA GPU with CUDA support (optional, recommended for deep learning)
- **Disk Space**: Minimum 10 GB free (for dataset, models, and dependencies)

3. Software Requirements:

The following software and libraries are required:

- Python 3.8 or newer
- Anaconda (for managing environments and packages)
- Jupyter Notebook or VS Code (for development and experimentation)

Required Python Packages:

- Pandas – For data manipulation
- NumPy – For numerical operations
- Matplotlib / Seaborn – For visualizations and plots
- Scikit-learn – For machine learning models and evaluation metrics
- TensorFlow / Keras – For deep learning model building
- Statsmodels – For ARIMA modelling
- Warnings – For suppressing unnecessary warnings
- StandardScaler / MinMaxScaler – For feature scaling

4. Dataset Details

4.1 Dataset Source and Access

- Dataset Name: [Agriculture Crop Yield]
- Source: Available on Kaggle — samuelotiattakorah/agriculture-crop-yield
- Sample Size: Approximately 1,000,000 records, each representing a unique crop cultivation event, with the target variable being crop yield in tons per hectare.

4.2 Dataset Features

The dataset includes the following key variables:

```
# view the first five attributes of the dataset
dataFrame.head()
```

	Region	Soil_Type	Crop	Rainfall_mm	Temperature_Celsius	Fertilizer_Used	Irrigation_Used	Weather_Condition	Days_to_Harvest	Yield_tons_per_hectare
0	West	Sandy	Cotton	897.077239	27.676966	False	True	Cloudy	122	6.555816
1	South	Clay	Rice	992.673282	18.026142	True	True	Rainy	140	8.527341
2	North	Loam	Barley	147.998025	29.794042	False	False	Sunny	106	1.127443
3	North	Sandy	Soybean	986.866331	16.644190	False	True	Rainy	146	6.517573
4	South	Silt	Wheat	730.379174	31.620687	True	True	Cloudy	110	7.248251

```
# view the last five attributes of the dataset
dataFrame.tail()
```

	Region	Soil_Type	Crop	Rainfall_mm	Temperature_Celsius	Fertilizer_Used	Irrigation_Used	Weather_Condition	Days_to_Harvest	Yield_tons_per_hectare
999995	West	Silt	Rice	302.805345	27.987428	False	False	Sunny	76	1.347586
999996	South	Chalky	Barley	932.991383	39.661039	True	False	Rainy	93	7.311594
999997	North	Peaty	Cotton	867.362046	24.370042	True	False	Cloudy	108	5.763182
999998	West	Silt	Wheat	492.812857	33.045505	False	False	Sunny	102	2.070159
999999	West	Sandy	Maize	180.936180	27.298847	True	False	Sunny	76	2.937243

Feature	Description
Region	Geographical region of cultivation (e.g. North, East, South, West)
Soil_Type	Soil classification (e.g. Clay, Sandy, Loam, Silt, Peaty, Chalky)
Crop	Crop type (e.g. Wheat, Rice, Maize, Barley, Soybean, Cotton)
Rainfall_mm	Total rainfall (in mm) during the growing season
Temperature_Celsius	Average temperature during crop growth (°C)
Fertilizer_Used	Whether fertilizer was applied (True/False)
Irrigation_Used	Whether irrigation was used (True/False)
Weather_Condition	Primary weather during season (Sunny, Rainy, Cloudy)
Days_to_Harvest	Duration between planting and harvest (in days)
Yield_tons_per_hectare	Final yield outcome per hectare (target variable)

5. Model Configuration

This section outlines the configuration details for each model used in the research.

5.1. Baseline Machine Learning Models

Lasso Regression:

- Regularization: L1
- Used for feature selection and linear forecasting

```
from sklearn.linear_model import Lasso
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# Initialize Lasso regression model (L1 regularization)
lasso_model = Lasso()
```

```
# Train the model
lasso_model.fit(X_train, y_train)
```

Lasso
▸ Parameters

```
# Make predictions
lasso_preds = lasso_model.predict(X_test)
```

```
# Evaluate performance
print("Lasso Regression Performance:")
print("MAE:", mean_absolute_error(y_test, lasso_preds))
print("MSE:", mean_squared_error(y_test, lasso_preds))
print("R² Score:", r2_score(y_test, lasso_preds))
```

```
Lasso Regression Performance:
MAE: 0.8890853371879558
MSE: 1.1871711756343184
R² Score: 0.5867702148520592
```

K-Nearest Neighbours:

- Neighbours: 5 (default)
- Distance metric: Euclidean

```
# Initialize KNN Regressor
knn_model = KNeighborsRegressor()
```

```
# Fit the model
knn_model.fit(X_train, y_train)
```

KNeighborsRegressor
▸ Parameters

```
# Predict
knn_preds = knn_model.predict(X_test)
```

```
# Evaluate performance
print(" KNN Regression Performance:")
print("MAE:", mean_absolute_error(y_test, knn_preds))
print("MSE:", mean_squared_error(y_test, knn_preds))
print("R² Score:", r2_score(y_test, knn_preds))
```

```
 KNN Regression Performance:
MAE: 0.8701638776916835
MSE: 1.150232933207540 7
R² Score: 0.5996276547015436
```

Decision Tree Regressor:

- Default parameters used
- Suitable for capturing non-linear relationships

```
# Initialize and train model
dt_model = DecisionTreeRegressor(random_state=42)
dt_model.fit(X_train, y_train)
```

```
▾ DecisionTreeRegressor
▸ Parameters
```

```
# Predict
dt_preds = dt_model.predict(X_test)
```

```
# Evaluate
print(" Decision Tree Performance:")
print("MAE:", mean_absolute_error(y_test, dt_preds))
print("MSE:", mean_squared_error(y_test, dt_preds))
print("R² Score:", r2_score(y_test, dt_preds))
```

```
 Decision Tree Performance:
MAE: 0.5824835086521799
MSE: 0.5312821268412065
R² Score: 0.8150716563597248
```

5.2. Deep Learning Models

Deep Neural Network (DNN) Configuration:

- Input Layer: Normalized numeric features
- Hidden Layers: 128 → 64 → 32
- Dropout: 0.3 after each dense layer
- Output Layer: Single neuron (linear activation)
- Optimizer: Adam
- Loss Function: Mean Squared Error (MSE)
- Activation: ReLU for hidden layers

```
# ------------------ MODEL 1: DNN ------------------
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout

dnn_model = Sequential([
    Dense(128, activation='relu', input_shape=(X_train_dl.shape[1],)),
    Dropout(0.3),
    Dense(64, activation='relu'),
    Dropout(0.3),
    Dense(32, activation='relu'),
    Dense(1)
])
dnn_model.compile(optimizer='adam', loss='mse', metrics=['mae'])
history_dnn = dnn_model.fit(X_train_dl, y_train, validation_split=0.2, epochs=10, batch_size=64, verbose=2)

dnn_preds = dnn_model.predict(X_test_dl).flatten()
print(" DNN Performance — MAE: {:.4f}, MSE: {:.4f}, R²: {:.4f}".format(
    mean_absolute_error(y_test, dnn_preds),
    mean_squared_error(y_test, dnn_preds),
    r2_score(y_test, dnn_preds)
))
```

LSTM Configuration:

- Input Shape: (Samples, Time Step=1, Features)
- LSTM Layer: 64 units
- Dropout: 0.3
- Dense Layer: 32
- Output Layer: 1
- Optimizer: Adam
- Loss: MSE

```
# ------------------ MODEL 2: LSTM ------------------
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
import numpy as np

# Ensure data is reshaped: (samples, timesteps, features)
X_train_dl = X_train.copy()
X_test_dl = X_test.copy()
X_train_lstm = np.expand_dims(X_train_dl.values, axis=1)
X_test_lstm = np.expand_dims(X_test_dl.values, axis=1)

lstm_model = Sequential([
    LSTM(64, input_shape=(X_train_lstm.shape[1], X_train_lstm.shape[2]), return_sequences=False),
    Dropout(0.3),
    Dense(32, activation='relu'),
    Dense(1)
])

lstm_model.compile(optimizer='adam', loss='mse', metrics=['mae'])

history_lstm = lstm_model.fit(
    X_train_lstm, y_train,
    validation_split=0.2,
    epochs=10,
    batch_size=64,
    verbose=2
)

lstm_preds = lstm_model.predict(X_test_lstm).flatten()

print(" LSTM Performance — MAE: {:.4f}, MSE: {:.4f}, R²: {:.4f}".format(
    mean_absolute_error(y_test, lstm_preds),
    mean_squared_error(y_test, lstm_preds),
    r2_score(y_test, lstm_preds)
))
```

CNN-GRU Hybrid Model:

- Input Shape: (Features, 1)
- Conv1D Layer: 64 filters, kernel size = 2
- Batch Normalization
- Dropout: 0.3
- GRU Layer: 64 units
- Dense Layers: 32 → 1
- Optimizer: Adam
- Loss: MSE

```
# Build CNN-GRU model
def build_cnn_gru_model(input_shape):
    model = Sequential([
        Conv1D(filters=64, kernel_size=3, activation='relu', padding='same', input_shape=input_shape),
        BatchNormalization(),
        Dropout(0.3),
        GRU(64, return_sequences=False),
        Dropout(0.3),
        Dense(32, activation='relu'),
        Dense(1)
    ])
    model.compile(optimizer='adam', loss='mse', metrics=['mae'])
    return model

input_shape = X_train_cnn_gru.shape[1:]  # (features, 1)
cnn_gru_model = build_cnn_gru_model(input_shape)
```

```
# Train CNN-GRU
history = cnn_gru_model.fit(
    X_train_cnn_gru, y_train,
    validation_split=0.2,
    epochs=10,
    batch_size=64,
    verbose=2
)
```

5.3. ARIMA-GRU Hybrid Model

Step 1: ARIMA

- ARIMA Order: (2,1,2)
- Use ARIMA to model the main trend of yield data

```
# ARIMA Model
arima_model = ARIMA(yield_series_noisy, order=(2, 1, 2))
arima_result = arima_model.fit()
arima_pred = arima_result.predict(start=0, end=len(yield_series_noisy)-1, typ='levels')

# Residuals
residuals = yield_series_noisy - arima_pred
residuals = residuals.values.reshape(-1, 1)

# Normalize residuals
scaler_resid = StandardScaler()
residuals_scaled = scaler_resid.fit_transform(residuals)
```

```
# Create Sequences
def create_sequences(data, seq_length):
    X, y = [], []
    for i in range(len(data) - seq_length):
        X.append(data[i:i+seq_length])
        y.append(data[i+seq_length])
    return np.array(X), np.array(y)
```

Step 2: GRU on Residuals

- Sequence length: 10
- GRU Layers: [128 → 64]
- Batch Normalization + Dropout (0.2)
- Dense Layers: [32 → 1]

```
# GRU Model
model_residuals = Sequential([
    GRU(128, input_shape=(SEQ_LEN, 1), return_sequences=True),
    BatchNormalization(),
    Dropout(0.2),
    GRU(64, return_sequences=False),
    Dense(32, activation='relu'),
    Dense(1)
])

optimizer = Adam(learning_rate=0.001)
model_residuals.compile(optimizer=optimizer, loss='mse', metrics=['mae'])

# Early Stopping
early_stop = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)
```

Final output is computed by combining ARIMA output and GRU prediction of residuals.

```
# Train
history_model = model_residuals.fit(
    X_seq, y_seq,
    epochs=10,
    batch_size=64,
    validation_split=0.2,
    callbacks=[early_stop],
    verbose=2
)

# Predict Residuals
residual_preds = model_residuals.predict(X_seq).flatten()
residual_preds_trimmed = residual_preds[:len(arima_pred) - SEQ_LEN]

# Final prediction = ARIMA + model(residual)
final_preds = arima_pred[SEQ_LEN:] + scaler_resid.inverse_transform(residual_preds_trimmed.reshape(-1, 1)).flatten()

# True values
true_yield = yield_series_noisy[SEQ_LEN:].values
```

6. Conclusion

This configuration manual provides a structured guide for replicating the crop yield forecasting research. Follow the steps for setting up the environment, preparing the dataset, configuring each model, and evaluating performance. Adjust model parameters as needed for further optimization and experiment replication.

For issues during setup, refer to official documentation for the respective libraries or consult error logs in Jupyter Notebook.

References

Python: https://www.python.org
TensorFlow: https://www.tensorflow.org
Keras: https://keras.io
Scikit-learn: https://scikit-learn.org
Dataset: Agriculture Crop Yield