APPENDICES

Appendix 1. Hyperparameter Configuration Code

```python
class LSTMHyperModel(HyperModel):
    def build(self, hp):
        model = Sequential()

        # Input layer
        model.add(Input(shape=(X_train.shape[1], X_train.shape[2])))

        # First LSTM layer
        return_sequences_flag = hp.Choice('use_second_lstm', [True, False])
        model.add(LSTM(units=hp.Int('units_lstm_1', min_value=32, max_value=512, step=32),
                       return_sequences=return_sequences_flag,
                       kernel_regularizer=tf.keras.regularizers.l1_l2(
                           l1=hp.Choice('l1_lstm_1', [1e-3, 1e-5]),
                           l2=hp.Choice('l2_lstm_1', [1e-3, 1e-5]))))

        # Batch Normalization & Dropout
        model.add(BatchNormalization())
        model.add(Dropout(rate=hp.Float('dropout_lstm_1', min_value=0.2, max_value=0.5, step=0.1)))

        # Second LSTM layer (only if True)
        if return_sequences_flag:
            model.add(LSTM(units=hp.Int('units_lstm_2', min_value=32, max_value=512, step=32),
                           return_sequences=False,
                           kernel_regularizer=tf.keras.regularizers.l1_l2(
                               l1=hp.Choice('l1_lstm_2', [1e-3, 1e-5]),
                               l2=hp.Choice('l2_lstm_2', [1e-3, 1e-5]))))

            # Batch Normalization & Dropout for second LSTM
            model.add(BatchNormalization())
            model.add(Dropout(rate=hp.Float('dropout_lstm_2', min_value=0.2, max_value=0.5, step=0.1)))

        # Dense layer 1
        model.add(Dense(units=hp.Int('units_dense_1', min_value=32, max_value=128, step=32),
                        activation='relu',
                        kernel_regularizer=tf.keras.regularizers.l1_l2(
                            l1=hp.Choice('l1_dense_1', [1e-3, 1e-5]),
                            l2=hp.Choice('l2_dense_1', [1e-3, 1e-5]))))
        model.add(Dropout(rate=hp.Float('dropout_dense_1', min_value=0.2, max_value=0.5, step=0.1)))

        # Dense layer 2
        model.add(Dense(units=hp.Int('units_dense_2', min_value=32, max_value=128, step=32),
                        activation='relu',
                        kernel_regularizer=tf.keras.regularizers.l2(
                            hp.Choice('l2_dense_2', [1e-4, 1e-5]))))
        model.add(Dropout(rate=hp.Float('dropout_dense_2', min_value=0.2, max_value=0.5, step=0.1)))

        # Output layer
        model.add(Dense(1))

        # Optimizer
        optimizer = tf.keras.optimizers.Adam(
            learning_rate=hp.Float('learning_rate',  max_value=1e-5, min_value=1e-7, sampling='LOG'))
        model.compile(optimizer=optimizer, loss='mse', metrics=['mae'])

        return model

```

Appendix 2. Data and Source Code

The data and complete code utilized in this study are available through the author's GitHub repository, as provided below. For further inquiries or to request full access, please contact the author.

The repository: https://github.com/mellisadmyn/drought-forecasting

BIOGRAPHY

Mellisa Damayanti is the daughter of Noris Hidayat and Ni Nengah Suriati. She was born in Denpasar on August 4, 2003, and is a native Indonesian citizen residing in Kuta Utara, Badung, Bali. She completed her primary education in 2015 at SD No 1 Kerobokan Kaja, followed by junior high school at SMP Negeri 2 Kuta Utara, graduating in 2018. She then continued her senior high school education at SMA Negeri 1 Kuta Utara. In 2021, she began her undergraduate studies in Computer Science at Universitas Pendidikan Ganesha, where she is currently completing her undergraduate thesis.