Deep Learning Portfolio Optimization

Library imports & Data Loading

```
import yfinance as yf
import pandas as pd
import numpy as np
import torch
import warnings
warnings.filterwarnings("ignore")
print("Libraries imported successfully.")

tickers = ['SPY', 'QQQ', 'IWM', 'RSP', 'MSFT', 'AAPL', 'TSLA', 'NVDA']
start_date = '2010-01-01'
end_date = '2025-01-01'
print(f"Tickers defined: {tickers}")
print(f"Start date: {start_date}, End date: {end_date}")

data = yf.download(tickers, start=start_date, end=end_date)
print("Data downloaded successfully.")

close_prices = data['Close']
daily_simple_returns = close_prices.pct_change()
daily_simple_returns.columns = [f'{col}_simple_returns' for col in
daily_simple_returns.columns]
print("Daily simple returns calculated successfully.")

weekly_prices = data['Close'].resample('W').last()
print("Daily prices resampled to weekly frequency successfully.")

weekly_log_returns = np.log(weekly_prices / weekly_prices.shift(1))
weekly_log_returns.columns = [f'{col}_returns' for col in
weekly_log_returns.columns]
print("Weekly log-returns calculated successfully.")

daily_simple_returns = daily_simple_returns.iloc[1:]
print("Daily simple returns DataFrame aligned by dropping the first
row.")

short_ma_daily = pd.DataFrame()
long_ma_daily = pd.DataFrame()

for ticker in tickers:
    short_ma_daily[f'{ticker}_short_ma'] =
close_prices[ticker].rolling(window=5).mean()
    long_ma_daily[f'{ticker}_long_ma'] =
close_prices[ticker].rolling(window=20).mean()

print("Daily short and long moving averages calculated successfully.")
```

```
volatility_daily = pd.DataFrame()

for ticker in tickers:
    volatility_daily[f'{ticker}_volatility'] =
daily_simple_returns[f'{ticker}_simple_returns'].rolling(window=20).st
d()

print("Daily volatilities calculated successfully.")

short_ma_daily_aligned = short_ma_daily.dropna()
long_ma_daily_aligned = long_ma_daily.dropna()
volatility_daily_aligned = volatility_daily.dropna()

print("Daily moving averages and volatilities DataFrames aligned by
dropping NaN values.")

weekly_short_ma_features = short_ma_daily_aligned.resample('W').last()
weekly_long_ma_features = long_ma_daily_aligned.resample('W').last()
weekly_volatility_features =
volatility_daily_aligned.resample('W').last()

print("Daily features resampled to weekly frequency successfully.")

combined_weekly_features = pd.concat([
    weekly_log_returns,
    weekly_short_ma_features,
    weekly_long_ma_features,
    weekly_volatility_features
], axis=1)

print("Weekly features concatenated successfully.")

combined_weekly_features = combined_weekly_features.dropna()
print("NaN values removed from combined_weekly_features.")
```

```
 [                       0%                       ]

Libraries imported successfully.
Tickers defined: ['SPY', 'QQQ', 'IWM', 'RSP', 'MSFT', 'AAPL', 'TSLA',
'NVDA']
Start date: 2010-01-01, End date: 2025-01-01

[**********************100%***********************]  8 of 8 completed

Data downloaded successfully.
Daily simple returns calculated successfully.
Daily prices resampled to weekly frequency successfully.
Weekly log-returns calculated successfully.
Daily simple returns DataFrame aligned by dropping the first row.
Daily short and long moving averages calculated successfully.
```

```
Daily volatilities calculated successfully.
Daily moving averages and volatilities DataFrames aligned by dropping
NaN values.
Daily features resampled to weekly frequency successfully.
Weekly features concatenated successfully.
NaN values removed from combined_weekly_features.
```

Feature Engineering & Data Preprocessing

```python
from sklearn.preprocessing import StandardScaler
from torch.utils.data import TensorDataset, DataLoader

def make_sequences(dataframe, sequence_length, feature_cols,
target_cols):
    X, y = [], []
    features = dataframe[feature_cols].values
    targets = dataframe[target_cols].values
    for i in range(len(dataframe) - sequence_length):
        X.append(features[i : i + sequence_length])
        y.append(targets[i + sequence_length])
    return np.array(X), np.array(y)

# 2. Split the combined_weekly_features DataFrame into training,
validation, and test sets.
total_len = len(combined_weekly_features)
train_size = int(0.7 * total_len)
val_size = int(0.15 * total_len)
test_size = total_len - train_size - val_size

train_df = combined_weekly_features.iloc[:train_size]
val_df = combined_weekly_features.iloc[train_size : train_size +
val_size]
test_df = combined_weekly_features.iloc[train_size + val_size :]

print(f"Train size: {len(train_df)}, Validation size: {len(val_df)},
Test size: {len(test_df)}")

# 3. Identify feature columns and target columns.
feature_cols = [col for col in combined_weekly_features.columns if not
col.endswith('_returns')]
target_cols = [col for col in combined_weekly_features.columns if
col.endswith('_returns')]

# 4. Initialize two StandardScaler objects: feature_scaler for
features and target_scaler for target returns.
feature_scaler = StandardScaler()
target_scaler = StandardScaler()

# 5. Fit feature_scaler and target_scaler on the training data and
then transform the training data.
```

```
X_train_scaled = feature_scaler.fit_transform(train_df[feature_cols])
y_train_scaled = target_scaler.fit_transform(train_df[target_cols])

X_train_scaled_df = pd.DataFrame(X_train_scaled, columns=feature_cols,
index=train_df.index)
y_train_scaled_df = pd.DataFrame(y_train_scaled, columns=target_cols,
index=train_df.index)

# 6. Transform the validation and test data using the *fitted*
feature_scaler and target_scaler.
X_val_scaled = feature_scaler.transform(val_df[feature_cols])
y_val_scaled = target_scaler.transform(val_df[target_cols])
X_test_scaled = feature_scaler.transform(test_df[feature_cols])
y_test_scaled = target_scaler.transform(test_df[target_cols])

X_val_scaled_df = pd.DataFrame(X_val_scaled, columns=feature_cols,
index=val_df.index)
y_val_scaled_df = pd.DataFrame(y_val_scaled, columns=target_cols,
index=val_df.index)
X_test_scaled_df = pd.DataFrame(X_test_scaled, columns=feature_cols,
index=test_df.index)
y_test_scaled_df = pd.DataFrame(y_test_scaled, columns=target_cols,
index=test_df.index)

print("Data scaled successfully.")

# 7. Concatenate the scaled feature and target DataFrames for each
split for sequence creation
train_combined_scaled_df = pd.concat([X_train_scaled_df,
y_train_scaled_df], axis=1)
val_combined_scaled_df = pd.concat([X_val_scaled_df, y_val_scaled_df],
axis=1)
test_combined_scaled_df = pd.concat([X_test_scaled_df,
y_test_scaled_df], axis=1)

# 8. Apply the make_sequences function to create sequential data.
sequence_length = 12 # Define sequence length

X_train_seq, y_train_seq = make_sequences(train_combined_scaled_df,
sequence_length, feature_cols, target_cols)
X_val_seq, y_val_seq = make_sequences(val_combined_scaled_df,
sequence_length, feature_cols, target_cols)
X_test_seq, y_test_seq = make_sequences(test_combined_scaled_df,
sequence_length, feature_cols, target_cols)

print(f"Sequential data created with sequence length:
{sequence_length}.")

# 9. Convert the sequential NumPy arrays into PyTorch tensors.
X_train_tensor = torch.tensor(X_train_seq, dtype=torch.float32)
```

```
y_train_tensor = torch.tensor(y_train_seq, dtype=torch.float32)
X_val_tensor = torch.tensor(X_val_seq, dtype=torch.float32)
y_val_tensor = torch.tensor(y_val_seq, dtype=torch.float32)
X_test_tensor = torch.tensor(X_test_seq, dtype=torch.float32)
y_test_tensor = torch.tensor(y_test_seq, dtype=torch.float32)

print("Sequential data converted to PyTorch tensors.")

# 10. Create TensorDataset objects.
train_dataset = TensorDataset(X_train_tensor, y_train_tensor)
val_dataset = TensorDataset(X_val_tensor, y_val_tensor)
test_dataset = TensorDataset(X_test_tensor, y_test_tensor)

print("TensorDatasets created.")

# 11. Create DataLoader objects.
batch_size = 32 # Define batch size
train_loader = DataLoader(train_dataset, batch_size=batch_size,
shuffle=True)
val_loader = DataLoader(val_dataset, batch_size=batch_size,
shuffle=False)
test_loader = DataLoader(test_dataset, batch_size=batch_size,
shuffle=False)

print(f"DataLoaders created with batch size: {batch_size}.")
```

```
Train size: 527, Validation size: 113, Test size: 114
Data scaled successfully.
Sequential data created with sequence length: 12.
Sequential data converted to PyTorch tensors.
TensorDatasets created.
DataLoaders created with batch size: 32.
```

Transformer Model Definitions

```
import torch.nn as nn
import math

class PositionalEncoding(nn.Module):
    """Injects some information about the relative or absolute
position of the tokens in the sequence."""
    def __init__(self, d_model, max_len=5000):
        super(PositionalEncoding, self).__init__()
        pe = torch.zeros(max_len, d_model)
        position = torch.arange(0, max_len,
dtype=torch.float).unsqueeze(1)
        div_term = torch.exp(torch.arange(0, d_model, 2).float() * (-
math.log(10000.0) / d_model))
        pe[:, 0::2] = torch.sin(position * div_term)
        pe[:, 1::2] = torch.cos(position * div_term)
```

```python
        pe = pe.unsqueeze(0).transpose(0, 1)
        self.register_buffer('pe', pe)

    def forward(self, x):
        """Adds positional encoding to the input tensor."""
        # x is expected to be of shape (seq_len, batch_size, d_model)
        x = x + self.pe[:x.size(0), :]
        return x

class PortfolioTransformer(nn.Module):
    """Transformer model for portfolio allocation."""
    def __init__(self, input_dim, d_model, n_heads, num_layers,
dim_feedforward, output_dim, dropout=0.1, max_len=5000):
        super(PortfolioTransformer, self).__init__()
        self.model_type = 'Transformer'
        self.d_model = d_model

        # 3a. Linear layer to project input features to d_model
        self.input_linear = nn.Linear(input_dim, d_model)

        # 3b. Positional Encoding layer
        self.pos_encoder = PositionalEncoding(d_model, max_len)

        # 3c. Transformer Encoder Layer
        encoder_layers = nn.TransformerEncoderLayer(
            d_model=d_model,
            nhead=n_heads,
            dim_feedforward=dim_feedforward,
            dropout=dropout,
            batch_first=False # Input is (seq_len, batch_size,
feature_dim)
        )

        # 3d. Transformer Encoder
        self.transformer_encoder =
nn.TransformerEncoder(encoder_layers, num_layers)

        # 3e. Final linear layer to map Transformer's output to target
assets
        self.output_linear = nn.Linear(d_model, output_dim)

        # 3f. Softmax activation for normalized portfolio weights
        self.softmax = nn.Softmax(dim=-1)

    def forward(self, x):
        """Forward pass for the PortfolioTransformer model."""
        # x shape: (batch_size, sequence_length, input_dim)
        # Transformer expects (sequence_length, batch_size,
feature_dim)
        x = x.permute(1, 0, 2) # (sequence_length, batch_size,
```

```
input_dim)

        # 4a. Pass input through initial linear projection
        x = self.input_linear(x) * math.sqrt(self.d_model)

        # 4b. Add positional encodings
        x = self.pos_encoder(x)

        # 4c. Pass through TransformerEncoder
        output = self.transformer_encoder(x)

        # 4d. Apply the final linear layer (taking the last time
step's output)
        # output shape is (sequence_length, batch_size, d_model)
        # We take the output of the last time step for prediction
        output = self.output_linear(output[-1, :, :]) # (batch_size,
output_dim)

        # 4e. Apply softmax for normalized portfolio weights
        weights = self.softmax(output)

        return weights

print("PositionalEncoding and PortfolioTransformer classes defined
successfully.")
```

```
PositionalEncoding and PortfolioTransformer classes defined
successfully.
```

LSTM Model Definitions

```
import torch.nn as nn
import torch

class PortfolioLSTM(nn.Module):
    """LSTM model for portfolio allocation."""
    def __init__(self, input_size, hidden_size, num_layers,
output_dim, dropout=0.1):
        super(PortfolioLSTM, self).__init__()
        self.hidden_size = hidden_size
        self.num_layers = num_layers

        # 2a. LSTM layer
        self.lstm = nn.LSTM(
            input_size,
            hidden_size,
            num_layers,
            batch_first=True, # Input and output tensors are provided
as (batch, seq, feature)
            dropout=dropout
```

```
        )

        # 2b. Final linear layer to map LSTM's output to target assets
        self.fc = nn.Linear(hidden_size, output_dim)

        # 2c. Softmax activation for normalized portfolio weights
        self.softmax = nn.Softmax(dim=-1)

    def forward(self, x):
        """Forward pass for the PortfolioLSTM model."""
        # x shape: (batch_size, sequence_length, input_size)

        # Initialize hidden and cell states
        h0 = torch.zeros(self.num_layers, x.size(0),
self.hidden_size).to(x.device)
        c0 = torch.zeros(self.num_layers, x.size(0),
self.hidden_size).to(x.device)

        # 3a. Pass input through LSTM layer
        # out: tensor of shape (batch_size, seq_length, hidden_size)
        # hn: tensor of shape (num_layers, batch_size, hidden_size)
        # cn: tensor of shape (num_layers, batch_size, hidden_size)
        out, (hn, cn) = self.lstm(x, (h0.detach(), c0.detach()))

        # 3b. Take the output from the last time step of the LSTM
sequence
        # For batch_first=True, the last time step is out[:, -1, :]
        last_step_output = out[:, -1, :]

        # 3c. Pass this last-step output through the final linear
layer
        linear_output = self.fc(last_step_output)

        # 3d. Apply softmax for normalized portfolio weights
        weights = self.softmax(linear_output)

        return weights

print("PortfolioLSTM class defined successfully.")
```

```
PortfolioLSTM class defined successfully.
```

Define Model Evaluation Function & Loss Function (Sharpe Ratio)

```
import torch
import numpy as np

def evaluate_model(model, data_loader, y_scaler):
    # 2. Set the model to evaluation mode and disable gradient
```

```
calculation.
    model.eval()
    all_weights = []
    all_returns = []

    with torch.no_grad():
        for X_batch, y_batch in data_loader:
            # 6. Move X_batch and y_batch to the appropriate device
(CPU or GPU).
            device = next(model.parameters()).device
            X_batch = X_batch.to(device)
            y_batch = y_batch.to(device)

            # 7. Pass X_batch through the model to obtain
predicted_weights.
            predicted_weights = model(X_batch)

            # 8. Convert y_batch to a NumPy array and then denormalize
it.
            y_batch_np = y_batch.cpu().numpy()
            denormalized_returns_batch =
y_scaler.inverse_transform(y_batch_np)

            # 9. Append the predicted_weights and the
denormalized_returns_batch to their respective lists.
            all_weights.append(predicted_weights.cpu().numpy())
            all_returns.append(denormalized_returns_batch)

    # 10. Concatenate all collected predicted_weights and all_returns
arrays horizontally.
    all_weights = np.concatenate(all_weights, axis=0)
    all_returns = np.concatenate(all_returns, axis=0)

    # 11. Calculate portfolio_returns
    portfolio_returns = np.sum(all_weights * all_returns, axis=1)

    # 12. Calculate the cumulative_return
    cumulative_return = np.prod(1 + portfolio_returns) - 1

    # 13. Calculate the volatility
    volatility = np.std(portfolio_returns)

    # 14. Calculate the sharpe_ratio with numerical stability
    sharpe_ratio = np.mean(portfolio_returns) / (volatility + 1e-6)

    # 15. Return the calculated cumulative_return, volatility, and
sharpe_ratio.
    return cumulative_return, volatility, sharpe_ratio
```

```
print("Model evaluation function `evaluate_model` defined
successfully.")
```

```
Model evaluation function `evaluate_model` defined successfully.
```

Execute Reduced Hyperparameter Search (Transformer with Sharpe Ratio Loss)

```
import torch.nn as nn
import torch
import numpy as np

def sharpe_ratio_loss(weights, returns):
    # Ensure returns are 2D for batch processing if they come as 1D
    if returns.dim() == 1:
        returns = returns.unsqueeze(0) # Make it (1, num_assets)

    # Ensure weights are 2D for batch processing if they come as 1D
    if weights.dim() == 1:
        weights = weights.unsqueeze(0) # Make it (1, num_assets)

    # Calculate portfolio returns: (batch_size, num_assets) *
(batch_size, num_assets) -> (batch_size,)
    portfolio_returns = torch.sum(weights * returns, dim=1)

    # Calculate the mean of portfolio returns
    mean_portfolio_return = torch.mean(portfolio_returns)

    # Calculate the standard deviation of portfolio returns with
numerical stability
    std_portfolio_return = torch.std(portfolio_returns) + 1e-6

    # Calculate Sharpe Ratio
    sharpe_ratio = mean_portfolio_return / std_portfolio_return

    # Return the negative Sharpe Ratio for minimization
    return -sharpe_ratio

def evaluate_model(model, data_loader, y_scaler):
    model.eval() # Set the model to evaluation mode
    all_weights = []
    all_returns = []

    with torch.no_grad(): # Disable gradient calculation for inference
        for X_batch, y_batch in data_loader:
            # Move data to the same device as the model
            device = next(model.parameters()).device
            X_batch = X_batch.to(device)
            y_batch = y_batch.to(device)
```

```
            predicted_weights = model(X_batch)

            y_batch_np = y_batch.cpu().numpy()
            denormalized_returns_batch =
y_scaler.inverse_transform(y_batch_np)

            all_weights.append(predicted_weights.cpu().numpy())
            all_returns.append(denormalized_returns_batch)

    # Concatenate all predicted weights and actual denormalized
returns
    all_weights = np.concatenate(all_weights, axis=0)
    all_returns = np.concatenate(all_returns, axis=0)

    # Calculate portfolio returns (dot product of weights and returns)
    portfolio_returns = np.sum(all_weights * all_returns, axis=1)

    # Calculate the cumulative returns
    # Add 1 to portfolio_returns before cumulative product
    cumulative_return = np.prod(1 + portfolio_returns) - 1

    # Calculate the volatility
    volatility = np.std(portfolio_returns)

    # Calculate the Sharpe Ratio with numerical stability
    sharpe_ratio = np.mean(portfolio_returns) / (volatility + 1e-6)

    # Return portfolio_returns along with the other metrics
    return cumulative_return, volatility, sharpe_ratio,
portfolio_returns

import itertools
import torch.optim as optim
import copy # For deep copying model states
import torch # Ensure torch is imported

# 1. Set the device for training
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
print(f"Using device: {device}")

# 2. Define fixed hyperparameters for the Transformer model
fixed_transformer_hps = {
    'd_model': 64,
    'n_heads': 4,
    'num_layers': 2,
    'dim_feedforward': 128,
    'dropout': 0.1
}
```

```
# 3. Define tuneable hyperparameters
hyperparams_reduced = {
    'lr': [0.0001, 0.001, 0.01],
    'weight_decay': [1e-5, 1e-4, 1e-3],
    'patience': [5, 7, 10]
}

# 4. Generate all possible combinations of the tuneable
hyperparameters.
hyparam_names_reduced = list(hyperparams_reduced.keys())
hyparam_combinations_reduced =
list(itertools.product(*hyperparams_reduced.values()))

# Get input and output dimensions from the prepared tensors
# These should be available from the consolidated data preprocessing
step.
# Assuming X_train_tensor and y_train_tensor are defined globally.
input_dim = X_train_tensor.shape[-1]
output_dim = y_train_tensor.shape[-1]

# 5. Initialize an empty list results_sharpe to store the results
results_sharpe = []
# Variables to track the best validation Sharpe Ratio for overall best
config
best_test_sharpe_overall = -float('inf')
best_config_overall = None

print(f"Total hyperparameter combinations to test:
{len(hyparam_combinations_reduced)}")
print(f"Input Dimension: {input_dim}, Output Dimension: {output_dim}")


num_epochs = 50 # Define a fixed number of epochs for training

# 6. Loop through each hyperparameter combination:
for i, combo in enumerate(hyparam_combinations_reduced):
    current_hps = dict(zip(hyparam_names_reduced, combo))
    print(f"\n--- Testing Combination
{i+1}/{len(hyparam_combinations_reduced)}: {current_hps} ---")

    # Initialize the PortfolioTransformer model
    model = PortfolioTransformer(
        input_dim=input_dim,
        d_model=fixed_transformer_hps['d_model'],
        n_heads=fixed_transformer_hps['n_heads'],
        num_layers=fixed_transformer_hps['num_layers'],
        dim_feedforward=fixed_transformer_hps['dim_feedforward'],
        output_dim=output_dim,
        dropout=fixed_transformer_hps['dropout']
    ).to(device)
```

```
    # Initialize the Adam optimizer
    optimizer = optim.Adam(model.parameters(), lr=current_hps['lr'],
weight_decay=current_hps['weight_decay'])

    # Initialize best_val_sharpe for the current run and
patience_counter
    best_val_sharpe = -float('inf')
    patience_counter = 0
    best_model_state = None # To save the best model state during
validation

    # Training loop for a fixed number of epochs
    for epoch in range(num_epochs):
        model.train() # Set the model to training mode
        total_loss = 0
        for X_batch, y_batch in train_loader:
            X_batch, y_batch = X_batch.to(device), y_batch.to(device)

            optimizer.zero_grad() # Zero the gradients

            predicted_weights = model(X_batch) # Get model predictions
(weights)

            loss = sharpe_ratio_loss(predicted_weights, y_batch) #
Calculate the sharpe_ratio_loss
            loss.backward() # Perform backpropagation
            optimizer.step() # Update model weights

            total_loss += loss.item()

        avg_train_loss = total_loss / len(train_loader) # Calculate
the average training loss for the epoch

        # Evaluate the model on the validation set
        _, _, val_sharpe, _ = evaluate_model(model, val_loader,
target_scaler)

        print(f"Epoch {epoch+1}/{num_epochs}, Train Loss:
{avg_train_loss:.4f}, Val Sharpe: {val_sharpe:.4f}")

        # Implement early stopping
        if val_sharpe > best_val_sharpe:
            best_val_sharpe = val_sharpe
            patience_counter = 0
            best_model_state = copy.deepcopy(model.state_dict()) #
Save the best model state
        else:
            patience_counter += 1
            if patience_counter >= current_hps['patience']:
```

```
                print(f"Early stopping triggered after {epoch+1}
epochs due to no improvement for {current_hps['patience']} epochs.")
                break

    # Load the best model state before testing
    if best_model_state:
        model.load_state_dict(best_model_state)

    # Evaluate the final model on the test set AND capture portfolio
returns
    # The evaluate_model function already returns
test_portfolio_returns
    test_cumulative_return, test_volatility, test_sharpe_ratio,
test_portfolio_returns = evaluate_model(model, test_loader,
target_scaler)

    # Store the results, including the portfolio_returns array
    results_sharpe.append({
        'hyperparams': current_hps,
        'cumulative_return': test_cumulative_return,
        'volatility': test_volatility,
        'sharpe_ratio': test_sharpe_ratio,
        'portfolio_returns': test_portfolio_returns # Store the actual
portfolio returns for plotting
    })

    # Update overall best configuration
    # Deep copy the best result dictionary to avoid issues with
mutable objects
    if test_sharpe_ratio > best_test_sharpe_overall:
        best_test_sharpe_overall = test_sharpe_ratio
        best_config_overall = copy.deepcopy(results_sharpe[-1]) #
Store the last appended result (which includes portfolio_returns)

# 7. Convert results_sharpe into a pandas DataFrame
results_sharpe_df = pd.DataFrame(results_sharpe)

# 8. Print the best_config_overall
print("\n--- Best Transformer Model Configuration (Sharpe Ratio Loss)
---")
print(best_config_overall)
```

```
Using device: cpu
Total hyperparameter combinations to test: 27
Input Dimension: 24, Output Dimension: 8

--- Testing Combination 1/27: {'lr': 0.0001, 'weight_decay': 1e-05,
'patience': 5} ---
Epoch 1/50, Train Loss: 0.0152, Val Sharpe: 0.0328
Epoch 2/50, Train Loss: -0.0970, Val Sharpe: 0.0333
```

```
Epoch 3/50, Train Loss: -0.0066, Val Sharpe: 0.0334
Epoch 4/50, Train Loss: 0.0676, Val Sharpe: 0.0344
Epoch 5/50, Train Loss: -0.0690, Val Sharpe: 0.0361
Epoch 6/50, Train Loss: -0.0288, Val Sharpe: 0.0368
Epoch 7/50, Train Loss: -0.0622, Val Sharpe: 0.0375
Epoch 8/50, Train Loss: -0.0917, Val Sharpe: 0.0385
Epoch 9/50, Train Loss: -0.0167, Val Sharpe: 0.0387
Epoch 10/50, Train Loss: -0.1101, Val Sharpe: 0.0387
Epoch 11/50, Train Loss: -0.0349, Val Sharpe: 0.0381
Epoch 12/50, Train Loss: -0.0424, Val Sharpe: 0.0377
Epoch 13/50, Train Loss: -0.0992, Val Sharpe: 0.0376
Epoch 14/50, Train Loss: 0.0091, Val Sharpe: 0.0373
Epoch 15/50, Train Loss: 1.0612, Val Sharpe: 0.0368
Early stopping triggered after 15 epochs due to no improvement for 5
epochs.

--- Testing Combination 2/27: {'lr': 0.0001, 'weight_decay': 1e-05,
'patience': 7} ---
Epoch 1/50, Train Loss: -0.2040, Val Sharpe: 0.0558
Epoch 2/50, Train Loss: -0.0372, Val Sharpe: 0.0544
Epoch 3/50, Train Loss: -0.2171, Val Sharpe: 0.0541
Epoch 4/50, Train Loss: -0.0424, Val Sharpe: 0.0535
Epoch 5/50, Train Loss: -0.0356, Val Sharpe: 0.0535
Epoch 6/50, Train Loss: -0.1476, Val Sharpe: 0.0539
Epoch 7/50, Train Loss: -0.0642, Val Sharpe: 0.0543
Epoch 8/50, Train Loss: -0.0313, Val Sharpe: 0.0545
Early stopping triggered after 8 epochs due to no improvement for 7
epochs.

--- Testing Combination 3/27: {'lr': 0.0001, 'weight_decay': 1e-05,
'patience': 10} ---
Epoch 1/50, Train Loss: -0.0006, Val Sharpe: 0.0344
Epoch 2/50, Train Loss: 0.0125, Val Sharpe: 0.0347
Epoch 3/50, Train Loss: -0.0282, Val Sharpe: 0.0356
Epoch 4/50, Train Loss: -0.1548, Val Sharpe: 0.0362
Epoch 5/50, Train Loss: -0.0926, Val Sharpe: 0.0366
Epoch 6/50, Train Loss: -0.0859, Val Sharpe: 0.0370
Epoch 7/50, Train Loss: 0.0083, Val Sharpe: 0.0379
Epoch 8/50, Train Loss: -0.0727, Val Sharpe: 0.0384
Epoch 9/50, Train Loss: -0.0380, Val Sharpe: 0.0390
Epoch 10/50, Train Loss: -0.0996, Val Sharpe: 0.0396
Epoch 11/50, Train Loss: -0.0584, Val Sharpe: 0.0397
Epoch 12/50, Train Loss: -0.0106, Val Sharpe: 0.0407
Epoch 13/50, Train Loss: -0.0087, Val Sharpe: 0.0414
Epoch 14/50, Train Loss: -0.0472, Val Sharpe: 0.0423
Epoch 15/50, Train Loss: -0.0742, Val Sharpe: 0.0424
Epoch 16/50, Train Loss: -0.0198, Val Sharpe: 0.0431
Epoch 17/50, Train Loss: -0.0683, Val Sharpe: 0.0429
Epoch 18/50, Train Loss: -0.1338, Val Sharpe: 0.0440
```

```
Epoch 19/50, Train Loss: -0.1478, Val Sharpe: 0.0453
Epoch 20/50, Train Loss: -0.0335, Val Sharpe: 0.0443
Epoch 21/50, Train Loss: -0.1006, Val Sharpe: 0.0428
Epoch 22/50, Train Loss: -0.1461, Val Sharpe: 0.0438
Epoch 23/50, Train Loss: -0.1061, Val Sharpe: 0.0443
Epoch 24/50, Train Loss: -0.0480, Val Sharpe: 0.0474
Epoch 25/50, Train Loss: -0.0896, Val Sharpe: 0.0540
Epoch 26/50, Train Loss: -0.0766, Val Sharpe: 0.0555
Epoch 27/50, Train Loss: -0.1480, Val Sharpe: 0.0559
Epoch 28/50, Train Loss: -0.0946, Val Sharpe: 0.0511
Epoch 29/50, Train Loss: -0.0600, Val Sharpe: 0.0496
Epoch 30/50, Train Loss: -0.0099, Val Sharpe: 0.0461
Epoch 31/50, Train Loss: -0.0811, Val Sharpe: 0.0480
Epoch 32/50, Train Loss: -0.1326, Val Sharpe: 0.0487
Epoch 33/50, Train Loss: -0.0492, Val Sharpe: 0.0494
Epoch 34/50, Train Loss: -0.1547, Val Sharpe: 0.0508
Epoch 35/50, Train Loss: -0.2181, Val Sharpe: 0.0517
Epoch 36/50, Train Loss: -0.1798, Val Sharpe: 0.0541
Epoch 37/50, Train Loss: -0.1015, Val Sharpe: 0.0558
Early stopping triggered after 37 epochs due to no improvement for 10
epochs.

--- Testing Combination 4/27: {'lr': 0.0001, 'weight_decay': 0.0001,
'patience': 5} ---
Epoch 1/50, Train Loss: -0.0146, Val Sharpe: 0.0326
Epoch 2/50, Train Loss: -0.0237, Val Sharpe: 0.0339
Epoch 3/50, Train Loss: -0.0374, Val Sharpe: 0.0341
Epoch 4/50, Train Loss: -0.0325, Val Sharpe: 0.0338
Epoch 5/50, Train Loss: -0.0699, Val Sharpe: 0.0339
Epoch 6/50, Train Loss: -0.0397, Val Sharpe: 0.0348
Epoch 7/50, Train Loss: -0.0664, Val Sharpe: 0.0333
Epoch 8/50, Train Loss: -0.0690, Val Sharpe: 0.0337
Epoch 9/50, Train Loss: 0.0199, Val Sharpe: 0.0343
Epoch 10/50, Train Loss: -0.0345, Val Sharpe: 0.0357
Epoch 11/50, Train Loss: -0.0405, Val Sharpe: 0.0371
Epoch 12/50, Train Loss: 0.0020, Val Sharpe: 0.0374
Epoch 13/50, Train Loss: -0.0840, Val Sharpe: 0.0393
Epoch 14/50, Train Loss: -0.0732, Val Sharpe: 0.0425
Epoch 15/50, Train Loss: -0.1037, Val Sharpe: 0.0435
Epoch 16/50, Train Loss: -0.0527, Val Sharpe: 0.0444
Epoch 17/50, Train Loss: -0.0377, Val Sharpe: 0.0444
Epoch 18/50, Train Loss: -0.0604, Val Sharpe: 0.0443
Epoch 19/50, Train Loss: -0.1202, Val Sharpe: 0.0443
Epoch 20/50, Train Loss: -0.1079, Val Sharpe: 0.0440
Epoch 21/50, Train Loss: -0.0765, Val Sharpe: 0.0433
Epoch 22/50, Train Loss: -0.0030, Val Sharpe: 0.0439
Early stopping triggered after 22 epochs due to no improvement for 5
epochs.
```

```
--- Testing Combination 5/27: {'lr': 0.0001, 'weight_decay': 0.0001,
'patience': 7} ---
Epoch 1/50, Train Loss: 0.0929, Val Sharpe: 0.0542
Epoch 2/50, Train Loss: -0.0084, Val Sharpe: 0.0532
Epoch 3/50, Train Loss: -0.0290, Val Sharpe: 0.0526
Epoch 4/50, Train Loss: 0.1080, Val Sharpe: 0.0522
Epoch 5/50, Train Loss: -0.0754, Val Sharpe: 0.0517
Epoch 6/50, Train Loss: -0.0559, Val Sharpe: 0.0515
Epoch 7/50, Train Loss: -0.0177, Val Sharpe: 0.0514
Epoch 8/50, Train Loss: -0.0862, Val Sharpe: 0.0510
Early stopping triggered after 8 epochs due to no improvement for 7
epochs.

--- Testing Combination 6/27: {'lr': 0.0001, 'weight_decay': 0.0001,
'patience': 10} ---
Epoch 1/50, Train Loss: -0.0205, Val Sharpe: 0.0376
Epoch 2/50, Train Loss: 0.0397, Val Sharpe: 0.0376
Epoch 3/50, Train Loss: -0.0097, Val Sharpe: 0.0371
Epoch 4/50, Train Loss: -0.0308, Val Sharpe: 0.0370
Epoch 5/50, Train Loss: -0.0363, Val Sharpe: 0.0378
Epoch 6/50, Train Loss: -0.0599, Val Sharpe: 0.0396
Epoch 7/50, Train Loss: 0.0097, Val Sharpe: 0.0413
Epoch 8/50, Train Loss: -0.1001, Val Sharpe: 0.0428
Epoch 9/50, Train Loss: -0.0118, Val Sharpe: 0.0427
Epoch 10/50, Train Loss: 0.0483, Val Sharpe: 0.0428
Epoch 11/50, Train Loss: -0.0742, Val Sharpe: 0.0439
Epoch 12/50, Train Loss: -0.0740, Val Sharpe: 0.0456
Epoch 13/50, Train Loss: 0.0236, Val Sharpe: 0.0479
Epoch 14/50, Train Loss: -0.0613, Val Sharpe: 0.0495
Epoch 15/50, Train Loss: -0.0110, Val Sharpe: 0.0497
Epoch 16/50, Train Loss: -0.0886, Val Sharpe: 0.0491
Epoch 17/50, Train Loss: -0.0535, Val Sharpe: 0.0499
Epoch 18/50, Train Loss: -0.0265, Val Sharpe: 0.0498
Epoch 19/50, Train Loss: -0.0495, Val Sharpe: 0.0502
Epoch 20/50, Train Loss: -0.0384, Val Sharpe: 0.0518
Epoch 21/50, Train Loss: -0.0367, Val Sharpe: 0.0536
Epoch 22/50, Train Loss: -0.0211, Val Sharpe: 0.0542
Epoch 23/50, Train Loss: -0.0413, Val Sharpe: 0.0546
Epoch 24/50, Train Loss: -0.0614, Val Sharpe: 0.0555
Epoch 25/50, Train Loss: -0.0647, Val Sharpe: 0.0564
Epoch 26/50, Train Loss: -0.1101, Val Sharpe: 0.0594
Epoch 27/50, Train Loss: -0.0257, Val Sharpe: 0.0627
Epoch 28/50, Train Loss: -0.1150, Val Sharpe: 0.0647
Epoch 29/50, Train Loss: -0.1017, Val Sharpe: 0.0640
Epoch 30/50, Train Loss: -0.0549, Val Sharpe: 0.0635
Epoch 31/50, Train Loss: -0.0990, Val Sharpe: 0.0634
Epoch 32/50, Train Loss: -0.0819, Val Sharpe: 0.0635
Epoch 33/50, Train Loss: -0.0869, Val Sharpe: 0.0630
Epoch 34/50, Train Loss: -0.0779, Val Sharpe: 0.0638
```

```
Epoch 35/50, Train Loss: -0.0876, Val Sharpe: 0.0649
Epoch 36/50, Train Loss: -0.0699, Val Sharpe: 0.0666
Epoch 37/50, Train Loss: -0.1110, Val Sharpe: 0.0667
Epoch 38/50, Train Loss: -0.0849, Val Sharpe: 0.0669
Epoch 39/50, Train Loss: -0.1948, Val Sharpe: 0.0670
Epoch 40/50, Train Loss: -0.1138, Val Sharpe: 0.0621
Epoch 41/50, Train Loss: -0.0820, Val Sharpe: 0.0628
Epoch 42/50, Train Loss: -0.0832, Val Sharpe: 0.0641
Epoch 43/50, Train Loss: -0.1224, Val Sharpe: 0.0649
Epoch 44/50, Train Loss: -0.1392, Val Sharpe: 0.0656
Epoch 45/50, Train Loss: -0.1343, Val Sharpe: 0.0639
Epoch 46/50, Train Loss: -0.1295, Val Sharpe: 0.0640
Epoch 47/50, Train Loss: -0.1902, Val Sharpe: 0.0651
Epoch 48/50, Train Loss: -0.1602, Val Sharpe: 0.0664
Epoch 49/50, Train Loss: -0.0971, Val Sharpe: 0.0672
Epoch 50/50, Train Loss: -0.1907, Val Sharpe: 0.0679

--- Testing Combination 7/27: {'lr': 0.0001, 'weight_decay': 0.001,
'patience': 5} ---
Epoch 1/50, Train Loss: 0.0068, Val Sharpe: 0.0429
Epoch 2/50, Train Loss: -0.0654, Val Sharpe: 0.0448
Epoch 3/50, Train Loss: -0.0787, Val Sharpe: 0.0469
Epoch 4/50, Train Loss: -0.0482, Val Sharpe: 0.0454
Epoch 5/50, Train Loss: -0.1044, Val Sharpe: 0.0456
Epoch 6/50, Train Loss: -0.0549, Val Sharpe: 0.0460
Epoch 7/50, Train Loss: -0.0061, Val Sharpe: 0.0468
Epoch 8/50, Train Loss: -0.0071, Val Sharpe: 0.0488
Epoch 9/50, Train Loss: -0.0206, Val Sharpe: 0.0493
Epoch 10/50, Train Loss: 0.0066, Val Sharpe: 0.0502
Epoch 11/50, Train Loss: 0.5192, Val Sharpe: 0.0511
Epoch 12/50, Train Loss: 0.0180, Val Sharpe: 0.0508
Epoch 13/50, Train Loss: -0.0419, Val Sharpe: 0.0510
Epoch 14/50, Train Loss: -0.0471, Val Sharpe: 0.0515
Epoch 15/50, Train Loss: -0.0315, Val Sharpe: 0.0514
Epoch 16/50, Train Loss: -0.1240, Val Sharpe: 0.0515
Epoch 17/50, Train Loss: -0.0528, Val Sharpe: 0.0518
Epoch 18/50, Train Loss: -0.1190, Val Sharpe: 0.0521
Epoch 19/50, Train Loss: -0.0583, Val Sharpe: 0.0527
Epoch 20/50, Train Loss: -0.0973, Val Sharpe: 0.0530
Epoch 21/50, Train Loss: -0.0425, Val Sharpe: 0.0534
Epoch 22/50, Train Loss: -0.0586, Val Sharpe: 0.0542
Epoch 23/50, Train Loss: -0.0586, Val Sharpe: 0.0544
Epoch 24/50, Train Loss: -0.0517, Val Sharpe: 0.0545
Epoch 25/50, Train Loss: -0.0279, Val Sharpe: 0.0548
Epoch 26/50, Train Loss: -0.0816, Val Sharpe: 0.0549
Epoch 27/50, Train Loss: -0.1081, Val Sharpe: 0.0552
Epoch 28/50, Train Loss: -0.1082, Val Sharpe: 0.0558
Epoch 29/50, Train Loss: -0.1202, Val Sharpe: 0.0552
Epoch 30/50, Train Loss: -0.1869, Val Sharpe: 0.0541
```

```
Epoch 31/50, Train Loss: -0.0751, Val Sharpe: 0.0514
Epoch 32/50, Train Loss: -0.0345, Val Sharpe: 0.0510
Epoch 33/50, Train Loss: -0.0647, Val Sharpe: 0.0517
Early stopping triggered after 33 epochs due to no improvement for 5
epochs.

--- Testing Combination 8/27: {'lr': 0.0001, 'weight_decay': 0.001,
'patience': 7} ---
Epoch 1/50, Train Loss: -0.1547, Val Sharpe: 0.0460
Epoch 2/50, Train Loss: 0.0091, Val Sharpe: 0.0467
Epoch 3/50, Train Loss: 0.0545, Val Sharpe: 0.0474
Epoch 4/50, Train Loss: -0.0212, Val Sharpe: 0.0487
Epoch 5/50, Train Loss: -0.0002, Val Sharpe: 0.0496
Epoch 6/50, Train Loss: 0.0705, Val Sharpe: 0.0499
Epoch 7/50, Train Loss: -0.0213, Val Sharpe: 0.0501
Epoch 8/50, Train Loss: -0.0176, Val Sharpe: 0.0502
Epoch 9/50, Train Loss: -0.0047, Val Sharpe: 0.0508
Epoch 10/50, Train Loss: -0.1048, Val Sharpe: 0.0514
Epoch 11/50, Train Loss: -0.1551, Val Sharpe: 0.0519
Epoch 12/50, Train Loss: -0.0126, Val Sharpe: 0.0526
Epoch 13/50, Train Loss: 0.1200, Val Sharpe: 0.0527
Epoch 14/50, Train Loss: 0.0418, Val Sharpe: 0.0526
Epoch 15/50, Train Loss: -0.0642, Val Sharpe: 0.0530
Epoch 16/50, Train Loss: -0.0543, Val Sharpe: 0.0535
Epoch 17/50, Train Loss: -0.2419, Val Sharpe: 0.0534
Epoch 18/50, Train Loss: -0.1027, Val Sharpe: 0.0518
Epoch 19/50, Train Loss: -0.0160, Val Sharpe: 0.0520
Epoch 20/50, Train Loss: -0.0389, Val Sharpe: 0.0525
Epoch 21/50, Train Loss: -0.0551, Val Sharpe: 0.0523
Epoch 22/50, Train Loss: -0.0440, Val Sharpe: 0.0525
Epoch 23/50, Train Loss: -0.0649, Val Sharpe: 0.0527
Early stopping triggered after 23 epochs due to no improvement for 7
epochs.

--- Testing Combination 9/27: {'lr': 0.0001, 'weight_decay': 0.001,
'patience': 10} ---
Epoch 1/50, Train Loss: -0.1034, Val Sharpe: 0.0275
Epoch 2/50, Train Loss: -0.0059, Val Sharpe: 0.0276
Epoch 3/50, Train Loss: -0.0522, Val Sharpe: 0.0282
Epoch 4/50, Train Loss: 0.0298, Val Sharpe: 0.0294
Epoch 5/50, Train Loss: 0.0212, Val Sharpe: 0.0300
Epoch 6/50, Train Loss: -0.1211, Val Sharpe: 0.0323
Epoch 7/50, Train Loss: -0.0711, Val Sharpe: 0.0327
Epoch 8/50, Train Loss: -0.0965, Val Sharpe: 0.0330
Epoch 9/50, Train Loss: -0.0177, Val Sharpe: 0.0335
Epoch 10/50, Train Loss: -0.0557, Val Sharpe: 0.0346
Epoch 11/50, Train Loss: -0.0642, Val Sharpe: 0.0357
Epoch 12/50, Train Loss: -0.0126, Val Sharpe: 0.0370
Epoch 13/50, Train Loss: -0.0731, Val Sharpe: 0.0383
```

```
Epoch 14/50, Train Loss: -0.0180, Val Sharpe: 0.0396
Epoch 15/50, Train Loss: -0.0147, Val Sharpe: 0.0397
Epoch 16/50, Train Loss: -0.0533, Val Sharpe: 0.0400
Epoch 17/50, Train Loss: 0.0057, Val Sharpe: 0.0407
Epoch 18/50, Train Loss: -0.0635, Val Sharpe: 0.0415
Epoch 19/50, Train Loss: 0.0430, Val Sharpe: 0.0427
Epoch 20/50, Train Loss: -0.0417, Val Sharpe: 0.0455
Epoch 21/50, Train Loss: -0.1442, Val Sharpe: 0.0454
Epoch 22/50, Train Loss: -0.1943, Val Sharpe: 0.0439
Epoch 23/50, Train Loss: -0.0952, Val Sharpe: 0.0445
Epoch 24/50, Train Loss: -0.0359, Val Sharpe: 0.0448
Epoch 25/50, Train Loss: -0.0109, Val Sharpe: 0.0450
Epoch 26/50, Train Loss: -0.0721, Val Sharpe: 0.0450
Epoch 27/50, Train Loss: -0.0503, Val Sharpe: 0.0450
Epoch 28/50, Train Loss: -0.0680, Val Sharpe: 0.0451
Epoch 29/50, Train Loss: -0.0386, Val Sharpe: 0.0451
Epoch 30/50, Train Loss: -0.0854, Val Sharpe: 0.0450
Early stopping triggered after 30 epochs due to no improvement for 10
epochs.

--- Testing Combination 10/27: {'lr': 0.001, 'weight_decay': 1e-05,
'patience': 5} ---
Epoch 1/50, Train Loss: -0.0029, Val Sharpe: 0.0507
Epoch 2/50, Train Loss: -0.2676, Val Sharpe: 0.0653
Epoch 3/50, Train Loss: -0.0661, Val Sharpe: 0.0649
Epoch 4/50, Train Loss: -0.0286, Val Sharpe: 0.0691
Epoch 5/50, Train Loss: -0.0869, Val Sharpe: 0.0664
Epoch 6/50, Train Loss: -0.1541, Val Sharpe: 0.0705
Epoch 7/50, Train Loss: 0.0831, Val Sharpe: 0.0713
Epoch 8/50, Train Loss: -0.0753, Val Sharpe: 0.0665
Epoch 9/50, Train Loss: -0.1251, Val Sharpe: 0.0717
Epoch 10/50, Train Loss: -0.0851, Val Sharpe: 0.0704
Epoch 11/50, Train Loss: -0.4728, Val Sharpe: 0.0718
Epoch 12/50, Train Loss: 0.0321, Val Sharpe: 0.0657
Epoch 13/50, Train Loss: -0.0320, Val Sharpe: 0.0678
Epoch 14/50, Train Loss: -0.0368, Val Sharpe: 0.0675
Epoch 15/50, Train Loss: -0.0597, Val Sharpe: 0.0661
Epoch 16/50, Train Loss: -0.0765, Val Sharpe: 0.0647
Early stopping triggered after 16 epochs due to no improvement for 5
epochs.

--- Testing Combination 11/27: {'lr': 0.001, 'weight_decay': 1e-05,
'patience': 7} ---
Epoch 1/50, Train Loss: 0.0171, Val Sharpe: 0.0367
Epoch 2/50, Train Loss: -0.0791, Val Sharpe: 0.0396
Epoch 3/50, Train Loss: -0.0422, Val Sharpe: 0.0394
Epoch 4/50, Train Loss: -0.0654, Val Sharpe: 0.0402
Epoch 5/50, Train Loss: 0.2071, Val Sharpe: 0.0422
Epoch 6/50, Train Loss: -0.0984, Val Sharpe: 0.0452
```

```
Epoch 7/50, Train Loss: -0.0358, Val Sharpe: 0.0618
Epoch 8/50, Train Loss: -0.1017, Val Sharpe: 0.0695
Epoch 9/50, Train Loss: -0.0717, Val Sharpe: 0.0722
Epoch 10/50, Train Loss: -0.1165, Val Sharpe: 0.0721
Epoch 11/50, Train Loss: -0.2634, Val Sharpe: 0.0750
Epoch 12/50, Train Loss: -0.0615, Val Sharpe: 0.0768
Epoch 13/50, Train Loss: -0.1296, Val Sharpe: 0.0776
Epoch 14/50, Train Loss: -0.0625, Val Sharpe: 0.0775
Epoch 15/50, Train Loss: -0.2206, Val Sharpe: 0.0770
Epoch 16/50, Train Loss: -0.1353, Val Sharpe: 0.0778
Epoch 17/50, Train Loss: -0.0798, Val Sharpe: 0.0228
Epoch 18/50, Train Loss: -0.1774, Val Sharpe: 0.0568
Epoch 19/50, Train Loss: -0.1148, Val Sharpe: 0.0609
Epoch 20/50, Train Loss: -0.0952, Val Sharpe: 0.0140
Epoch 21/50, Train Loss: -0.1451, Val Sharpe: 0.0134
Epoch 22/50, Train Loss: -0.1639, Val Sharpe: 0.0105
Epoch 23/50, Train Loss: -0.1117, Val Sharpe: 0.0104
Early stopping triggered after 23 epochs due to no improvement for 7
epochs.

--- Testing Combination 12/27: {'lr': 0.001, 'weight_decay': 1e-05,
'patience': 10} ---
Epoch 1/50, Train Loss: 0.0429, Val Sharpe: 0.0465
Epoch 2/50, Train Loss: -0.0737, Val Sharpe: 0.0412
Epoch 3/50, Train Loss: -0.0492, Val Sharpe: 0.0459
Epoch 4/50, Train Loss: -0.0957, Val Sharpe: 0.0471
Epoch 5/50, Train Loss: -0.1017, Val Sharpe: 0.0334
Epoch 6/50, Train Loss: -0.1526, Val Sharpe: 0.0311
Epoch 7/50, Train Loss: -0.1852, Val Sharpe: 0.0477
Epoch 8/50, Train Loss: -0.1243, Val Sharpe: 0.0284
Epoch 9/50, Train Loss: 0.0003, Val Sharpe: 0.0308
Epoch 10/50, Train Loss: -0.1459, Val Sharpe: 0.0745
Epoch 11/50, Train Loss: -0.0767, Val Sharpe: 0.0756
Epoch 12/50, Train Loss: -0.1570, Val Sharpe: 0.0745
Epoch 13/50, Train Loss: -0.1254, Val Sharpe: 0.0729
Epoch 14/50, Train Loss: -0.1091, Val Sharpe: 0.0724
Epoch 15/50, Train Loss: -0.1534, Val Sharpe: 0.0703
Epoch 16/50, Train Loss: -0.1678, Val Sharpe: 0.0725
Epoch 17/50, Train Loss: -0.1719, Val Sharpe: 0.0709
Epoch 18/50, Train Loss: -0.1491, Val Sharpe: 0.0733
Epoch 19/50, Train Loss: -0.1965, Val Sharpe: 0.0699
Epoch 20/50, Train Loss: -0.1716, Val Sharpe: 0.0687
Epoch 21/50, Train Loss: -0.2010, Val Sharpe: 0.0682
Early stopping triggered after 21 epochs due to no improvement for 10
epochs.

--- Testing Combination 13/27: {'lr': 0.001, 'weight_decay': 0.0001,
'patience': 5} ---
Epoch 1/50, Train Loss: -0.0145, Val Sharpe: 0.0369
```

```
Epoch 2/50, Train Loss: -0.0830, Val Sharpe: 0.0391
Epoch 3/50, Train Loss: -0.0958, Val Sharpe: 0.0416
Epoch 4/50, Train Loss: -0.1051, Val Sharpe: 0.0427
Epoch 5/50, Train Loss: -0.0906, Val Sharpe: 0.0406
Epoch 6/50, Train Loss: -0.0597, Val Sharpe: 0.0405
Epoch 7/50, Train Loss: -0.1164, Val Sharpe: 0.0409
Epoch 8/50, Train Loss: -0.1281, Val Sharpe: 0.0409
Epoch 9/50, Train Loss: -0.1030, Val Sharpe: 0.0408
Early stopping triggered after 9 epochs due to no improvement for 5
epochs.

--- Testing Combination 14/27: {'lr': 0.001, 'weight_decay': 0.0001,
'patience': 7} ---
Epoch 1/50, Train Loss: 0.0058, Val Sharpe: 0.0326
Epoch 2/50, Train Loss: -0.0042, Val Sharpe: 0.0385
Epoch 3/50, Train Loss: -0.0720, Val Sharpe: 0.0398
Epoch 4/50, Train Loss: -0.0819, Val Sharpe: 0.0345
Epoch 5/50, Train Loss: -0.0994, Val Sharpe: 0.0483
Epoch 6/50, Train Loss: -0.0343, Val Sharpe: 0.0244
Epoch 7/50, Train Loss: -0.3376, Val Sharpe: 0.0308
Epoch 8/50, Train Loss: -0.1322, Val Sharpe: 0.0319
Epoch 9/50, Train Loss: -0.0627, Val Sharpe: 0.0270
Epoch 10/50, Train Loss: -0.1465, Val Sharpe: 0.0234
Epoch 11/50, Train Loss: -0.3499, Val Sharpe: 0.0264
Epoch 12/50, Train Loss: -0.1474, Val Sharpe: 0.0407
Early stopping triggered after 12 epochs due to no improvement for 7
epochs.

--- Testing Combination 15/27: {'lr': 0.001, 'weight_decay': 0.0001,
'patience': 10} ---
Epoch 1/50, Train Loss: 0.0268, Val Sharpe: 0.0225
Epoch 2/50, Train Loss: -0.0386, Val Sharpe: 0.0265
Epoch 3/50, Train Loss: 0.0071, Val Sharpe: 0.0243
Epoch 4/50, Train Loss: -0.0595, Val Sharpe: 0.0360
Epoch 5/50, Train Loss: -0.0764, Val Sharpe: 0.0355
Epoch 6/50, Train Loss: -0.0377, Val Sharpe: 0.0320
Epoch 7/50, Train Loss: -0.1642, Val Sharpe: 0.0529
Epoch 8/50, Train Loss: -0.3320, Val Sharpe: 0.0637
Epoch 9/50, Train Loss: -0.0747, Val Sharpe: 0.0732
Epoch 10/50, Train Loss: -0.1270, Val Sharpe: 0.0746
Epoch 11/50, Train Loss: -0.0829, Val Sharpe: 0.0753
Epoch 12/50, Train Loss: -0.0606, Val Sharpe: 0.0740
Epoch 13/50, Train Loss: -0.1461, Val Sharpe: 0.0712
Epoch 14/50, Train Loss: -0.0516, Val Sharpe: 0.0554
Epoch 15/50, Train Loss: -0.2478, Val Sharpe: 0.0462
Epoch 16/50, Train Loss: -0.1553, Val Sharpe: 0.0479
Epoch 17/50, Train Loss: -0.1112, Val Sharpe: 0.0633
Epoch 18/50, Train Loss: -0.1230, Val Sharpe: 0.0635
Epoch 19/50, Train Loss: -0.1510, Val Sharpe: 0.0620
```

```
Epoch 20/50, Train Loss: -0.1997, Val Sharpe: 0.0650
Epoch 21/50, Train Loss: -0.1722, Val Sharpe: 0.0699
Early stopping triggered after 21 epochs due to no improvement for 10
epochs.

--- Testing Combination 16/27: {'lr': 0.001, 'weight_decay': 0.001,
'patience': 5} ---
Epoch 1/50, Train Loss: 0.0551, Val Sharpe: 0.0435
Epoch 2/50, Train Loss: -0.0934, Val Sharpe: 0.0502
Epoch 3/50, Train Loss: -0.0623, Val Sharpe: 0.0545
Epoch 4/50, Train Loss: -0.0443, Val Sharpe: 0.0551
Epoch 5/50, Train Loss: -0.0970, Val Sharpe: 0.0632
Epoch 6/50, Train Loss: -0.1120, Val Sharpe: 0.0699
Epoch 7/50, Train Loss: -0.0673, Val Sharpe: 0.0560
Epoch 8/50, Train Loss: -0.0154, Val Sharpe: 0.0696
Epoch 9/50, Train Loss: -0.1142, Val Sharpe: 0.0668
Epoch 10/50, Train Loss: -0.1446, Val Sharpe: 0.0720
Epoch 11/50, Train Loss: -0.1228, Val Sharpe: 0.0787
Epoch 12/50, Train Loss: -0.1356, Val Sharpe: 0.0712
Epoch 13/50, Train Loss: -0.1626, Val Sharpe: 0.0759
Epoch 14/50, Train Loss: -0.1562, Val Sharpe: 0.0748
Epoch 15/50, Train Loss: -0.1656, Val Sharpe: 0.0777
Epoch 16/50, Train Loss: -0.0706, Val Sharpe: 0.0761
Early stopping triggered after 16 epochs due to no improvement for 5
epochs.

--- Testing Combination 17/27: {'lr': 0.001, 'weight_decay': 0.001,
'patience': 7} ---
Epoch 1/50, Train Loss: 0.0714, Val Sharpe: 0.0507
Epoch 2/50, Train Loss: -0.0921, Val Sharpe: 0.0434
Epoch 3/50, Train Loss: -0.1095, Val Sharpe: 0.0451
Epoch 4/50, Train Loss: -0.1205, Val Sharpe: 0.0341
Epoch 5/50, Train Loss: -0.0688, Val Sharpe: 0.0418
Epoch 6/50, Train Loss: -0.1366, Val Sharpe: 0.0479
Epoch 7/50, Train Loss: -0.1392, Val Sharpe: 0.0282
Epoch 8/50, Train Loss: -0.2491, Val Sharpe: 0.0199
Early stopping triggered after 8 epochs due to no improvement for 7
epochs.

--- Testing Combination 18/27: {'lr': 0.001, 'weight_decay': 0.001,
'patience': 10} ---
Epoch 1/50, Train Loss: -0.6239, Val Sharpe: 0.0369
Epoch 2/50, Train Loss: -0.0555, Val Sharpe: 0.0414
Epoch 3/50, Train Loss: -0.0447, Val Sharpe: 0.0431
Epoch 4/50, Train Loss: -0.0273, Val Sharpe: 0.0444
Epoch 5/50, Train Loss: -0.0578, Val Sharpe: 0.0446
Epoch 6/50, Train Loss: 0.0142, Val Sharpe: 0.0456
Epoch 7/50, Train Loss: -0.1098, Val Sharpe: 0.0478
Epoch 8/50, Train Loss: -0.1276, Val Sharpe: 0.0488
Epoch 9/50, Train Loss: -0.0181, Val Sharpe: 0.0465
```

```
Epoch 10/50, Train Loss: -0.0071, Val Sharpe: 0.0455
Epoch 11/50, Train Loss: -0.0430, Val Sharpe: 0.0438
Epoch 12/50, Train Loss: -0.0432, Val Sharpe: 0.0444
Epoch 13/50, Train Loss: -0.0743, Val Sharpe: 0.0431
Epoch 14/50, Train Loss: -0.1731, Val Sharpe: 0.0456
Epoch 15/50, Train Loss: -0.2156, Val Sharpe: 0.0522
Epoch 16/50, Train Loss: -0.0394, Val Sharpe: 0.0636
Epoch 17/50, Train Loss: -0.0636, Val Sharpe: 0.0608
Epoch 18/50, Train Loss: -0.0538, Val Sharpe: 0.0565
Epoch 19/50, Train Loss: -0.0674, Val Sharpe: 0.0571
Epoch 20/50, Train Loss: 0.0556, Val Sharpe: 0.0555
Epoch 21/50, Train Loss: -0.0933, Val Sharpe: 0.0539
Epoch 22/50, Train Loss: -0.0933, Val Sharpe: 0.0577
Epoch 23/50, Train Loss: -0.0393, Val Sharpe: 0.0636
Epoch 24/50, Train Loss: -0.1483, Val Sharpe: 0.0664
Epoch 25/50, Train Loss: -0.2036, Val Sharpe: 0.0666
Epoch 26/50, Train Loss: -0.0808, Val Sharpe: 0.0677
Epoch 27/50, Train Loss: -0.1716, Val Sharpe: 0.0613
Epoch 28/50, Train Loss: -0.0679, Val Sharpe: 0.0460
Epoch 29/50, Train Loss: -0.1800, Val Sharpe: 0.0497
Epoch 30/50, Train Loss: -0.0878, Val Sharpe: 0.0515
Epoch 31/50, Train Loss: -0.1407, Val Sharpe: 0.0573
Epoch 32/50, Train Loss: -0.1184, Val Sharpe: 0.0643
Epoch 33/50, Train Loss: -0.1701, Val Sharpe: 0.0639
Epoch 34/50, Train Loss: -0.1771, Val Sharpe: 0.0590
Epoch 35/50, Train Loss: -0.1717, Val Sharpe: 0.0612
Epoch 36/50, Train Loss: -0.1586, Val Sharpe: 0.0586
Early stopping triggered after 36 epochs due to no improvement for 10
epochs.

--- Testing Combination 19/27: {'lr': 0.01, 'weight_decay': 1e-05,
'patience': 5} ---
Epoch 1/50, Train Loss: -0.0544, Val Sharpe: -0.0027
Epoch 2/50, Train Loss: 0.0386, Val Sharpe: 0.0721
Epoch 3/50, Train Loss: 0.0272, Val Sharpe: -0.0060
Epoch 4/50, Train Loss: -0.0450, Val Sharpe: -0.0060
Epoch 5/50, Train Loss: -0.0384, Val Sharpe: -0.0057
Epoch 6/50, Train Loss: -0.0324, Val Sharpe: -0.0059
Epoch 7/50, Train Loss: -0.0317, Val Sharpe: 0.0721
Epoch 8/50, Train Loss: -0.1019, Val Sharpe: 0.0721
Epoch 9/50, Train Loss: 0.0191, Val Sharpe: 0.0721
Epoch 10/50, Train Loss: 0.0085, Val Sharpe: 0.0721
Epoch 11/50, Train Loss: 0.0449, Val Sharpe: 0.0721
Epoch 12/50, Train Loss: 0.0522, Val Sharpe: 0.0721
Epoch 13/50, Train Loss: 0.0144, Val Sharpe: 0.0721
Epoch 14/50, Train Loss: 0.0861, Val Sharpe: 0.0721
Epoch 15/50, Train Loss: 0.0358, Val Sharpe: 0.0721
Early stopping triggered after 15 epochs due to no improvement for 5
epochs.
```

```
--- Testing Combination 20/27: {'lr': 0.01, 'weight_decay': 1e-05,
'patience': 7} ---
Epoch 1/50, Train Loss: 0.0039, Val Sharpe: 0.0385
Epoch 2/50, Train Loss: -0.0268, Val Sharpe: 0.0376
Epoch 3/50, Train Loss: -0.0302, Val Sharpe: 0.0375
Epoch 4/50, Train Loss: -0.0300, Val Sharpe: 0.0376
Epoch 5/50, Train Loss: -0.0880, Val Sharpe: 0.0374
Epoch 6/50, Train Loss: -0.0192, Val Sharpe: 0.0374
Epoch 7/50, Train Loss: -0.0510, Val Sharpe: 0.0374
Epoch 8/50, Train Loss: -0.0853, Val Sharpe: 0.0374
Early stopping triggered after 8 epochs due to no improvement for 7
epochs.

--- Testing Combination 21/27: {'lr': 0.01, 'weight_decay': 1e-05,
'patience': 10} ---
Epoch 1/50, Train Loss: 0.0756, Val Sharpe: -0.0000
Epoch 2/50, Train Loss: -0.0216, Val Sharpe: -0.0060
Epoch 3/50, Train Loss: -0.0197, Val Sharpe: 0.0248
Epoch 4/50, Train Loss: -0.0064, Val Sharpe: 0.0166
Epoch 5/50, Train Loss: -0.0221, Val Sharpe: 0.0166
Epoch 6/50, Train Loss: 0.0038, Val Sharpe: 0.0166
Epoch 7/50, Train Loss: -0.0113, Val Sharpe: 0.0166
Epoch 8/50, Train Loss: -0.0628, Val Sharpe: 0.0166
Epoch 9/50, Train Loss: 0.0250, Val Sharpe: 0.0166
Epoch 10/50, Train Loss: -0.0621, Val Sharpe: 0.0166
Epoch 11/50, Train Loss: -0.0365, Val Sharpe: 0.0166
Epoch 12/50, Train Loss: -0.0152, Val Sharpe: 0.0166
Epoch 13/50, Train Loss: -0.0103, Val Sharpe: 0.0166
Early stopping triggered after 13 epochs due to no improvement for 10
epochs.

--- Testing Combination 22/27: {'lr': 0.01, 'weight_decay': 0.0001,
'patience': 5} ---
Epoch 1/50, Train Loss: -0.0716, Val Sharpe: 0.0373
Epoch 2/50, Train Loss: -0.0366, Val Sharpe: 0.0373
Epoch 3/50, Train Loss: -0.0505, Val Sharpe: 0.0373
Epoch 4/50, Train Loss: -0.0014, Val Sharpe: 0.0373
Epoch 5/50, Train Loss: -0.1248, Val Sharpe: 0.0373
Epoch 6/50, Train Loss: -0.0485, Val Sharpe: 0.0373
Early stopping triggered after 6 epochs due to no improvement for 5
epochs.

--- Testing Combination 23/27: {'lr': 0.01, 'weight_decay': 0.0001,
'patience': 7} ---
Epoch 1/50, Train Loss: 0.0036, Val Sharpe: 0.0770
Epoch 2/50, Train Loss: -0.0214, Val Sharpe: 0.0720
Epoch 3/50, Train Loss: -0.0297, Val Sharpe: 0.0721
Epoch 4/50, Train Loss: -0.0059, Val Sharpe: 0.0720
Epoch 5/50, Train Loss: -0.0202, Val Sharpe: 0.0720
Epoch 6/50, Train Loss: -0.0720, Val Sharpe: 0.0720
```

```
Epoch 7/50, Train Loss: -0.0165, Val Sharpe: 0.0720
Epoch 8/50, Train Loss: 0.0332, Val Sharpe: 0.0719
Early stopping triggered after 8 epochs due to no improvement for 7
epochs.

--- Testing Combination 24/27: {'lr': 0.01, 'weight_decay': 0.0001,
'patience': 10} ---
Epoch 1/50, Train Loss: 0.0471, Val Sharpe: 0.0174
Epoch 2/50, Train Loss: -0.0916, Val Sharpe: 0.0001
Epoch 3/50, Train Loss: -0.1068, Val Sharpe: 0.0091
Epoch 4/50, Train Loss: -0.1321, Val Sharpe: 0.0045
Epoch 5/50, Train Loss: -0.0490, Val Sharpe: 0.0365
Epoch 6/50, Train Loss: 0.0511, Val Sharpe: 0.0620
Epoch 7/50, Train Loss: -0.0582, Val Sharpe: 0.0772
Epoch 8/50, Train Loss: -0.0098, Val Sharpe: 0.0772
Epoch 9/50, Train Loss: -0.0210, Val Sharpe: 0.0771
Epoch 10/50, Train Loss: -0.0925, Val Sharpe: 0.0771
Epoch 11/50, Train Loss: -0.0615, Val Sharpe: 0.0771
Epoch 12/50, Train Loss: -0.0399, Val Sharpe: 0.0771
Epoch 13/50, Train Loss: 0.0208, Val Sharpe: 0.0770
Epoch 14/50, Train Loss: -0.0890, Val Sharpe: 0.0770
Epoch 15/50, Train Loss: 0.0909, Val Sharpe: 0.0772
Epoch 16/50, Train Loss: 0.0082, Val Sharpe: 0.0773
Epoch 17/50, Train Loss: -0.0595, Val Sharpe: 0.0773
Epoch 18/50, Train Loss: -0.0687, Val Sharpe: 0.0718
Epoch 19/50, Train Loss: -0.0467, Val Sharpe: 0.0720
Epoch 20/50, Train Loss: -0.0030, Val Sharpe: 0.0721
Epoch 21/50, Train Loss: -0.0133, Val Sharpe: 0.0721
Epoch 22/50, Train Loss: 0.0748, Val Sharpe: 0.0721
Epoch 23/50, Train Loss: -0.0378, Val Sharpe: 0.0720
Epoch 24/50, Train Loss: -0.0356, Val Sharpe: 0.0720
Epoch 25/50, Train Loss: -0.0356, Val Sharpe: 0.0720
Epoch 26/50, Train Loss: 0.0896, Val Sharpe: 0.0721
Epoch 27/50, Train Loss: 0.0083, Val Sharpe: 0.0721
Early stopping triggered after 27 epochs due to no improvement for 10
epochs.

--- Testing Combination 25/27: {'lr': 0.01, 'weight_decay': 0.001,
'patience': 5} ---
Epoch 1/50, Train Loss: -0.0119, Val Sharpe: -0.0038
Epoch 2/50, Train Loss: -0.0465, Val Sharpe: -0.0044
Epoch 3/50, Train Loss: -0.0517, Val Sharpe: -0.0009
Epoch 4/50, Train Loss: 0.1032, Val Sharpe: 0.0376
Epoch 5/50, Train Loss: 0.0099, Val Sharpe: 0.0371
Epoch 6/50, Train Loss: -0.0418, Val Sharpe: 0.0373
Epoch 7/50, Train Loss: -0.0333, Val Sharpe: 0.0373
Epoch 8/50, Train Loss: -0.0660, Val Sharpe: 0.0720
Epoch 9/50, Train Loss: 0.0878, Val Sharpe: 0.0376
Epoch 10/50, Train Loss: -0.0259, Val Sharpe: 0.0375
```

```
Epoch 11/50, Train Loss: -0.0310, Val Sharpe: 0.0706
Epoch 12/50, Train Loss: -0.0336, Val Sharpe: 0.0669
Epoch 13/50, Train Loss: 0.1214, Val Sharpe: 0.0629
Early stopping triggered after 13 epochs due to no improvement for 5
epochs.

--- Testing Combination 26/27: {'lr': 0.01, 'weight_decay': 0.001,
'patience': 7} ---
Epoch 1/50, Train Loss: 0.0279, Val Sharpe: 0.0771
Epoch 2/50, Train Loss: -0.0120, Val Sharpe: 0.0771
Epoch 3/50, Train Loss: -0.0433, Val Sharpe: 0.0771
Epoch 4/50, Train Loss: -0.0960, Val Sharpe: 0.0770
Epoch 5/50, Train Loss: -0.0756, Val Sharpe: 0.0759
Epoch 6/50, Train Loss: -0.0915, Val Sharpe: 0.0771
Epoch 7/50, Train Loss: -0.2111, Val Sharpe: 0.0770
Epoch 8/50, Train Loss: -0.0684, Val Sharpe: 0.0762
Epoch 9/50, Train Loss: -0.0191, Val Sharpe: 0.0759
Epoch 10/50, Train Loss: 0.0270, Val Sharpe: 0.0755
Epoch 11/50, Train Loss: 0.0138, Val Sharpe: 0.0747
Epoch 12/50, Train Loss: -0.0830, Val Sharpe: 0.0764
Epoch 13/50, Train Loss: -0.0292, Val Sharpe: 0.0769
Early stopping triggered after 13 epochs due to no improvement for 7
epochs.

--- Testing Combination 27/27: {'lr': 0.01, 'weight_decay': 0.001,
'patience': 10} ---
Epoch 1/50, Train Loss: -0.1820, Val Sharpe: 0.0145
Epoch 2/50, Train Loss: 0.0349, Val Sharpe: 0.0708
Epoch 3/50, Train Loss: -0.0500, Val Sharpe: 0.0744
Epoch 4/50, Train Loss: 0.0437, Val Sharpe: 0.0721
Epoch 5/50, Train Loss: -0.3497, Val Sharpe: 0.0720
Epoch 6/50, Train Loss: -0.0144, Val Sharpe: 0.0544
Epoch 7/50, Train Loss: 0.0359, Val Sharpe: 0.0170
Epoch 8/50, Train Loss: -0.0296, Val Sharpe: 0.0169
Epoch 9/50, Train Loss: -0.0095, Val Sharpe: 0.0169
Epoch 10/50, Train Loss: 0.0109, Val Sharpe: 0.0174
Epoch 11/50, Train Loss: -0.0701, Val Sharpe: 0.0216
Epoch 12/50, Train Loss: 0.0000, Val Sharpe: 0.0466
Epoch 13/50, Train Loss: -0.2243, Val Sharpe: 0.0440
Early stopping triggered after 13 epochs due to no improvement for 10
epochs.

--- Best Transformer Model Configuration (Sharpe Ratio Loss) ---
{'hyperparams': {'lr': 0.0001, 'weight_decay': 1e-05, 'patience': 10},
'cumulative_return': np.float32(1.3941593), 'volatility':
np.float32(0.03362132), 'sharpe_ratio': np.float32(0.272315),
'portfolio_returns': array([ 1.0204110e-01,  4.7160026e-02, -
9.3734171e-04,  1.2711523e-02,
       -2.0980490e-02,  2.2203580e-02, -4.9219288e-02,  6.0252380e-02,
        2.9139016e-02,  4.1846741e-02, -2.5870904e-02, -2.1145605e-03,
```

```
        -1.8740749e-02,  2.2429865e-02,  1.8941950e-02, -6.3207904e-03,
         4.6641663e-02,  6.9575161e-02,  3.2570556e-02,  1.7614955e-02,
         5.2421197e-02, -1.2062775e-02,  1.9319557e-02,  2.6149163e-04,
         3.1444129e-02, -1.5303785e-02,  2.2550218e-02, -4.3022364e-02,
        -3.8078867e-02, -1.3589873e-02,  3.8446594e-02,  3.9467312e-02,
        -2.8144825e-02, -1.0848138e-03, -4.4279419e-02,  8.7388009e-03,
         3.0764360e-02, -5.0834483e-03, -6.1576325e-02, -2.0097792e-02,
         7.0607975e-02,  3.2745302e-02,  3.0935958e-02, -5.7681277e-04,
         1.2375086e-03,  1.2341385e-02,  2.4038747e-02, -5.8131409e-05,
         1.5944545e-03, -3.0163275e-02,  2.9188883e-02,  2.7096733e-02,
        -1.5713558e-02,  2.2839081e-02,  3.4570657e-02, -2.7463557e-03,
         1.4629170e-02,  2.5891207e-02, -1.9794807e-02, -1.1338816e-02,
         3.6724959e-02, -4.8277350e-03, -2.2400815e-02,  3.8892683e-03,
        -8.7216020e-02,  7.2514236e-02,  2.7752463e-02, -5.8567156e-03,
         2.8520163e-02,  3.2897111e-02, -2.0114102e-03,  2.7884683e-02,
         3.5137471e-02,  7.7930279e-05,  9.5262509e-03,  6.3662328e-02,
         8.0774836e-03, -4.0338960e-02, -3.1619605e-02, -3.6968820e-02,
        -1.3904866e-02,  7.2806813e-02,  1.8228192e-02, -2.1502195e-02,
        -5.9846949e-02,  7.3650002e-02,  9.9545624e-03,  2.9202536e-02,
        -4.8760897e-03, -2.0952765e-03,  1.3403019e-02,  3.7114210e-02,
        -3.5859838e-02,  8.8509381e-02, -2.3206444e-02,  2.7243013e-02,
        -1.1148741e-03,  3.4446947e-02,  1.4151447e-04, -1.6108949e-02,
         8.9859031e-03, -2.4055995e-02], dtype=float32)}
```

Execute Reduced Hyperparameter Search (LSTM with Sharpe Ratio Loss)

```
import itertools
import torch.optim as optim
import copy # For deep copying model states
import pandas as pd
import torch
import torch.nn as nn
import numpy as np

# --- PortfolioLSTM class definition (re-included for self-
containment) ---
class PortfolioLSTM(nn.Module):
    """LSTM model for portfolio allocation."""
    def __init__(self, input_size, hidden_size, num_layers,
output_dim, dropout=0.1):
        super(PortfolioLSTM, self).__init__()
        self.hidden_size = hidden_size
        self.num_layers = num_layers

        # 2a. LSTM layer
        self.lstm = nn.LSTM(
            input_size,
            hidden_size,
```

```
            num_layers,
            batch_first=True, # Input and output tensors are provided
as (batch, seq, feature)
            dropout=dropout
        )

        # 2b. Final linear layer to map LSTM's output to target assets
        self.fc = nn.Linear(hidden_size, output_dim)

        # 2c. Softmax activation for normalized portfolio weights
        self.softmax = nn.Softmax(dim=-1)

    def forward(self, x):
        """Forward pass for the PortfolioLSTM model."""
        # x shape: (batch_size, sequence_length, input_size)

        # Initialize hidden and cell states
        h0 = torch.zeros(self.num_layers, x.size(0),
self.hidden_size).to(x.device)
        c0 = torch.zeros(self.num_layers, x.size(0),
self.hidden_size).to(x.device)

        # 3a. Pass input through LSTM layer
        # out: tensor of shape (batch_size, seq_length, hidden_size)
        # hn: tensor of shape (num_layers, batch_size, hidden_size)
        # cn: tensor of shape (num_layers, batch_size, hidden_size)
        out, (hn, cn) = self.lstm(x, (h0.detach(), c0.detach()))

        # 3b. Take the output from the last time step of the LSTM
sequence
        # For batch_first=True, the last time step is out[:, -1, :]
        last_step_output = out[:, -1, :]

        # 3c. Pass this last-step output through the final linear
layer
        linear_output = self.fc(last_step_output)

        # 3d. Apply softmax for normalized portfolio weights
        weights = self.softmax(linear_output)

        return weights
# --- End PortfolioLSTM class definition ---

# sharpe_ratio_loss function definition
def sharpe_ratio_loss(weights, returns):
    # Ensure returns are 2D for batch processing if they come as 1D
    if returns.dim() == 1:
        returns = returns.unsqueeze(0) # Make it (1, num_assets)

    # Ensure weights are 2D for batch processing if they come as 1D
```

```
    if weights.dim() == 1:
        weights = weights.unsqueeze(0) # Make it (1, num_assets)

    # Calculate portfolio returns: (batch_size, num_assets) *
(batch_size, num_assets) -> (batch_size,)
    portfolio_returns = torch.sum(weights * returns, dim=1)

    # Calculate the mean of portfolio returns
    mean_portfolio_return = torch.mean(portfolio_returns)

    # Calculate the standard deviation of portfolio returns with
numerical stability
    std_portfolio_return = torch.std(portfolio_returns) + 1e-6

    # Calculate Sharpe Ratio
    sharpe_ratio = mean_portfolio_return / std_portfolio_return

    # Return the negative Sharpe Ratio for minimization
    return -sharpe_ratio

# evaluate_model function definition
def evaluate_model(model, data_loader, y_scaler):
    # Set the model to evaluation mode and disable gradient
calculation.
    model.eval()
    all_weights = []
    all_returns = []

    with torch.no_grad():
        for X_batch, y_batch in data_loader:
            # Move data to the same device as the model
            device = next(model.parameters()).device
            X_batch = X_batch.to(device)
            y_batch = y_batch.to(device)

            # Pass the input features through the model to get
predicted weights.
            predicted_weights = model(X_batch)

            # Convert y_batch (scaled returns) to NumPy for
inverse_transform
            y_batch_np = y_batch.cpu().numpy()

            # Denormalize the actual target returns
            denormalized_returns_batch =
y_scaler.inverse_transform(y_batch_np)

            # Append to lists
            all_weights.append(predicted_weights.cpu().numpy())
            all_returns.append(denormalized_returns_batch)
```

```
    # Concatenate all predicted weights and denormalized actual
returns
    all_weights = np.concatenate(all_weights, axis=0)
    all_returns = np.concatenate(all_returns, axis=0)

    # Calculate portfolio returns (dot product of weights and returns)
    portfolio_returns = np.sum(all_weights * all_returns, axis=1)

    # Calculate the cumulative returns
    # Add 1 to portfolio_returns before cumulative product
    cumulative_return = np.prod(1 + portfolio_returns) - 1

    # Calculate the volatility
    volatility = np.std(portfolio_returns)

    # Calculate the Sharpe Ratio with numerical stability
    sharpe_ratio = np.mean(portfolio_returns) / (volatility + 1e-6)

    # Return the cumulative_return, volatility, and sharpe_ratio.
    return cumulative_return, volatility, sharpe_ratio,
portfolio_returns


# 1. Set the device for training
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
print(f"Using device: {device}")

# 2. Define fixed hyperparameters for the LSTM model
fixed_lstm_hps = {
    'hidden_size': 64,
    'num_layers': 2,
    'dropout': 0.1
}

# 3. Define tuneable hyperparameters (same as Transformer)
hyperparams_lstm_reduced = {
    'lr': [0.0001, 0.001, 0.01],
    'weight_decay': [1e-5, 1e-4, 1e-3],
    'patience': [5, 7, 10]
}

# 4. Generate all possible combinations of the tuneable
hyperparameters.
hyparam_names_lstm_reduced = list(hyperparams_lstm_reduced.keys())
hyparam_combinations_lstm_reduced =
list(itertools.product(*hyperparams_lstm_reduced.values()))

# Get input and output dimensions from the prepared tensors (already
defined globally)
```

```
# input_dim = X_train_tensor.shape[-1]
# output_dim = y_train_tensor.shape[-1]

# 5. Initialize an empty list results_sharpe_lstm to store the results
results_sharpe_lstm = []
# Variables to track the best validation Sharpe Ratio for overall best
config
best_test_sharpe_overall_lstm = -float('inf')
best_config_overall_lstm = None

print(f"Total LSTM hyperparameter combinations to test:
{len(hyparam_combinations_lstm_reduced)}")
print(f"Input Dimension: {input_dim}, Output Dimension: {output_dim}")

# 6. Define a fixed number of epochs for training
num_epochs = 50 # Re-using the same num_epochs as Transformer

# Loop through each hyperparameter combination:
for i, combo in enumerate(hyparam_combinations_lstm_reduced):
    current_hps_lstm = dict(zip(hyparam_names_lstm_reduced, combo))
    print(f"\n--- Testing LSTM Combination
{i+1}/{len(hyparam_combinations_lstm_reduced)}: {current_hps_lstm}
---")

    # Instantiate the PortfolioLSTM model
    model_lstm = PortfolioLSTM(
        input_size=input_dim,
        hidden_size=fixed_lstm_hps['hidden_size'],
        num_layers=fixed_lstm_hps['num_layers'],
        output_dim=output_dim,
        dropout=fixed_lstm_hps['dropout']
    ).to(device)

    # Initialize the Adam optimizer
    optimizer_lstm = optim.Adam(model_lstm.parameters(),
lr=current_hps_lstm['lr'],
weight_decay=current_hps_lstm['weight_decay'])

    # Initialize best_val_sharpe_lstm for the current run and
patience_counter_lstm
    best_val_sharpe_lstm = -float('inf')
    patience_counter_lstm = 0
    best_model_state_lstm = None # To save the best model state during
validation

    # Training loop for a fixed number of epochs
    for epoch in range(num_epochs):
        model_lstm.train() # Set the model to training mode
        total_loss_lstm = 0
        for X_batch, y_batch in train_loader:
```

```
            X_batch, y_batch = X_batch.to(device), y_batch.to(device)

            optimizer_lstm.zero_grad() # Zero the gradients

            predicted_weights_lstm = model_lstm(X_batch) # Get model
predictions (weights)

            loss_lstm = sharpe_ratio_loss(predicted_weights_lstm,
y_batch) # Calculate the sharpe_ratio_loss
            loss_lstm.backward() # Perform backpropagation
            optimizer_lstm.step() # Update model weights

            total_loss_lstm += loss_lstm.item()

        avg_train_loss_lstm = total_loss_lstm / len(train_loader) #
Calculate the average training loss for the epoch

        # Evaluate the model on the validation set
        _, _, val_sharpe_lstm, _ = evaluate_model(model_lstm,
val_loader, target_scaler)

        print(f"Epoch {epoch+1}/{num_epochs}, Train Loss:
{avg_train_loss_lstm:.4f}, Val Sharpe: {val_sharpe_lstm:.4f}")

        # Implement early stopping
        if val_sharpe_lstm > best_val_sharpe_lstm:
            best_val_sharpe_lstm = val_sharpe_lstm
            patience_counter_lstm = 0
            best_model_state_lstm =
copy.deepcopy(model_lstm.state_dict()) # Save the best model state
        else:
            patience_counter_lstm += 1
            if patience_counter_lstm >= current_hps_lstm['patience']:
                print(f"Early stopping triggered after {epoch+1}
epochs due to no improvement for {current_hps_lstm['patience']}
epochs.")
                break

    # Load the best model state before testing
    if best_model_state_lstm:
        model_lstm.load_state_dict(best_model_state_lstm)

    # Evaluate the final model on the test set AND capture portfolio
returns
    test_cumulative_return_lstm, test_volatility_lstm,
test_sharpe_ratio_lstm, test_portfolio_returns_lstm =
evaluate_model(model_lstm, test_loader, target_scaler)

    # Store the results
    results_sharpe_lstm.append({
```

```
        'hyperparams': current_hps_lstm,
        'cumulative_return': test_cumulative_return_lstm,
        'volatility': test_volatility_lstm,
        'sharpe_ratio': test_sharpe_ratio_lstm,
        'portfolio_returns': test_portfolio_returns_lstm # Store the
actual portfolio returns for plotting
    })

    # Update overall best configuration
    if test_sharpe_ratio_lstm > best_test_sharpe_overall_lstm:
        best_test_sharpe_overall_lstm = test_sharpe_ratio_lstm
        best_config_overall_lstm = copy.deepcopy(results_sharpe_lstm[-
1]) # Store the last appended result (which includes
portfolio_returns)

# 7. Convert results_sharpe_lstm into a pandas DataFrame
results_sharpe_lstm_df = pd.DataFrame(results_sharpe_lstm)

# 8. Print the best_config_overall_lstm
print("\n--- Best LSTM Model Configuration (Sharpe Ratio Loss) ---")
print(best_config_overall_lstm)
```

```
Using device: cpu
Total LSTM hyperparameter combinations to test: 27
Input Dimension: 24, Output Dimension: 8

--- Testing LSTM Combination 1/27: {'lr': 0.0001, 'weight_decay': 1e-
05, 'patience': 5} ---
Epoch 1/50, Train Loss: 0.0345, Val Sharpe: 0.0398
Epoch 2/50, Train Loss: 0.0057, Val Sharpe: 0.0398
Epoch 3/50, Train Loss: 0.0519, Val Sharpe: 0.0398
Epoch 4/50, Train Loss: -0.0187, Val Sharpe: 0.0399
Epoch 5/50, Train Loss: -0.0549, Val Sharpe: 0.0399
Epoch 6/50, Train Loss: 0.0513, Val Sharpe: 0.0398
Epoch 7/50, Train Loss: -0.0054, Val Sharpe: 0.0399
Epoch 8/50, Train Loss: 0.0146, Val Sharpe: 0.0399
Epoch 9/50, Train Loss: -0.0028, Val Sharpe: 0.0400
Epoch 10/50, Train Loss: 0.0280, Val Sharpe: 0.0399
Epoch 11/50, Train Loss: -0.0956, Val Sharpe: 0.0399
Epoch 12/50, Train Loss: 0.0100, Val Sharpe: 0.0399
Epoch 13/50, Train Loss: -0.0088, Val Sharpe: 0.0400
Epoch 14/50, Train Loss: -0.0243, Val Sharpe: 0.0400
Epoch 15/50, Train Loss: 0.0144, Val Sharpe: 0.0402
Epoch 16/50, Train Loss: -0.0360, Val Sharpe: 0.0404
Epoch 17/50, Train Loss: -0.0774, Val Sharpe: 0.0405
Epoch 18/50, Train Loss: -0.0374, Val Sharpe: 0.0406
Epoch 19/50, Train Loss: -0.0183, Val Sharpe: 0.0407
Epoch 20/50, Train Loss: 0.0201, Val Sharpe: 0.0409
Epoch 21/50, Train Loss: -0.0832, Val Sharpe: 0.0410
Epoch 22/50, Train Loss: -0.0234, Val Sharpe: 0.0412
```

```
Epoch 23/50, Train Loss: 0.0088, Val Sharpe: 0.0414
Epoch 24/50, Train Loss: -0.0427, Val Sharpe: 0.0416
Epoch 25/50, Train Loss: 0.0387, Val Sharpe: 0.0417
Epoch 26/50, Train Loss: -0.0316, Val Sharpe: 0.0419
Epoch 27/50, Train Loss: -0.0186, Val Sharpe: 0.0421
Epoch 28/50, Train Loss: 0.0268, Val Sharpe: 0.0424
Epoch 29/50, Train Loss: -0.0517, Val Sharpe: 0.0428
Epoch 30/50, Train Loss: -0.0209, Val Sharpe: 0.0432
Epoch 31/50, Train Loss: -0.0085, Val Sharpe: 0.0437
Epoch 32/50, Train Loss: -0.2234, Val Sharpe: 0.0440
Epoch 33/50, Train Loss: -0.0028, Val Sharpe: 0.0444
Epoch 34/50, Train Loss: -0.0538, Val Sharpe: 0.0448
Epoch 35/50, Train Loss: -0.0186, Val Sharpe: 0.0450
Epoch 36/50, Train Loss: -0.0124, Val Sharpe: 0.0453
Epoch 37/50, Train Loss: 0.0062, Val Sharpe: 0.0456
Epoch 38/50, Train Loss: 0.2212, Val Sharpe: 0.0459
Epoch 39/50, Train Loss: 0.0006, Val Sharpe: 0.0464
Epoch 40/50, Train Loss: -0.0623, Val Sharpe: 0.0466
Epoch 41/50, Train Loss: -0.0597, Val Sharpe: 0.0468
Epoch 42/50, Train Loss: -0.2014, Val Sharpe: 0.0470
Epoch 43/50, Train Loss: -0.0831, Val Sharpe: 0.0473
Epoch 44/50, Train Loss: -0.0410, Val Sharpe: 0.0476
Epoch 45/50, Train Loss: 0.0126, Val Sharpe: 0.0477
Epoch 46/50, Train Loss: -0.0880, Val Sharpe: 0.0479
Epoch 47/50, Train Loss: -0.0050, Val Sharpe: 0.0481
Epoch 48/50, Train Loss: -0.0734, Val Sharpe: 0.0483
Epoch 49/50, Train Loss: -0.0153, Val Sharpe: 0.0485
Epoch 50/50, Train Loss: -0.0357, Val Sharpe: 0.0486

--- Testing LSTM Combination 2/27: {'lr': 0.0001, 'weight_decay': 1e-
05, 'patience': 7} ---
Epoch 1/50, Train Loss: -0.0176, Val Sharpe: 0.0399
Epoch 2/50, Train Loss: -0.0221, Val Sharpe: 0.0399
Epoch 3/50, Train Loss: -0.0473, Val Sharpe: 0.0399
Epoch 4/50, Train Loss: -0.0175, Val Sharpe: 0.0399
Epoch 5/50, Train Loss: -0.0499, Val Sharpe: 0.0399
Epoch 6/50, Train Loss: -0.0614, Val Sharpe: 0.0399
Epoch 7/50, Train Loss: -0.0446, Val Sharpe: 0.0400
Epoch 8/50, Train Loss: -0.0551, Val Sharpe: 0.0400
Epoch 9/50, Train Loss: 0.0017, Val Sharpe: 0.0401
Epoch 10/50, Train Loss: -0.0346, Val Sharpe: 0.0401
Epoch 11/50, Train Loss: -0.0205, Val Sharpe: 0.0402
Epoch 12/50, Train Loss: -0.0179, Val Sharpe: 0.0403
Epoch 13/50, Train Loss: -0.1277, Val Sharpe: 0.0403
Epoch 14/50, Train Loss: 0.1714, Val Sharpe: 0.0406
Epoch 15/50, Train Loss: -0.2012, Val Sharpe: 0.0408
Epoch 16/50, Train Loss: 0.0145, Val Sharpe: 0.0409
Epoch 17/50, Train Loss: -0.0275, Val Sharpe: 0.0410
Epoch 18/50, Train Loss: -0.0284, Val Sharpe: 0.0411
```

```
Epoch 19/50, Train Loss: 0.0592, Val Sharpe: 0.0411
Epoch 20/50, Train Loss: 0.0266, Val Sharpe: 0.0411
Epoch 21/50, Train Loss: -0.0181, Val Sharpe: 0.0412
Epoch 22/50, Train Loss: -0.0132, Val Sharpe: 0.0413
Epoch 23/50, Train Loss: -0.0534, Val Sharpe: 0.0413
Epoch 24/50, Train Loss: -0.0202, Val Sharpe: 0.0412
Epoch 25/50, Train Loss: -0.0995, Val Sharpe: 0.0413
Epoch 26/50, Train Loss: -0.0307, Val Sharpe: 0.0415
Epoch 27/50, Train Loss: -0.0142, Val Sharpe: 0.0416
Epoch 28/50, Train Loss: 0.0155, Val Sharpe: 0.0418
Epoch 29/50, Train Loss: -0.0999, Val Sharpe: 0.0420
Epoch 30/50, Train Loss: -0.1004, Val Sharpe: 0.0422
Epoch 31/50, Train Loss: 0.0147, Val Sharpe: 0.0426
Epoch 32/50, Train Loss: -0.0149, Val Sharpe: 0.0431
Epoch 33/50, Train Loss: -0.0063, Val Sharpe: 0.0435
Epoch 34/50, Train Loss: -0.0811, Val Sharpe: 0.0438
Epoch 35/50, Train Loss: -0.0074, Val Sharpe: 0.0442
Epoch 36/50, Train Loss: -0.1079, Val Sharpe: 0.0445
Epoch 37/50, Train Loss: -0.0258, Val Sharpe: 0.0452
Epoch 38/50, Train Loss: -0.0528, Val Sharpe: 0.0455
Epoch 39/50, Train Loss: 0.0148, Val Sharpe: 0.0456
Epoch 40/50, Train Loss: -0.1265, Val Sharpe: 0.0457
Epoch 41/50, Train Loss: -0.0615, Val Sharpe: 0.0458
Epoch 42/50, Train Loss: -0.0920, Val Sharpe: 0.0458
Epoch 43/50, Train Loss: -0.0331, Val Sharpe: 0.0466
Epoch 44/50, Train Loss: -0.0761, Val Sharpe: 0.0460
Epoch 45/50, Train Loss: -0.0315, Val Sharpe: 0.0482
Epoch 46/50, Train Loss: -0.1193, Val Sharpe: 0.0458
Epoch 47/50, Train Loss: -0.0110, Val Sharpe: 0.0478
Epoch 48/50, Train Loss: -0.0588, Val Sharpe: 0.0511
Epoch 49/50, Train Loss: 0.0842, Val Sharpe: 0.0474
Epoch 50/50, Train Loss: -0.1074, Val Sharpe: 0.0468

--- Testing LSTM Combination 3/27: {'lr': 0.0001, 'weight_decay': 1e-
05, 'patience': 10} ---
Epoch 1/50, Train Loss: 0.0286, Val Sharpe: 0.0402
Epoch 2/50, Train Loss: -0.0625, Val Sharpe: 0.0402
Epoch 3/50, Train Loss: 0.1142, Val Sharpe: 0.0403
Epoch 4/50, Train Loss: -0.0034, Val Sharpe: 0.0404
Epoch 5/50, Train Loss: -0.0298, Val Sharpe: 0.0404
Epoch 6/50, Train Loss: -0.0390, Val Sharpe: 0.0404
Epoch 7/50, Train Loss: 0.0332, Val Sharpe: 0.0405
Epoch 8/50, Train Loss: 0.1642, Val Sharpe: 0.0406
Epoch 9/50, Train Loss: -0.0696, Val Sharpe: 0.0406
Epoch 10/50, Train Loss: -0.0457, Val Sharpe: 0.0406
Epoch 11/50, Train Loss: 0.0116, Val Sharpe: 0.0407
Epoch 12/50, Train Loss: -0.0680, Val Sharpe: 0.0408
Epoch 13/50, Train Loss: -0.0815, Val Sharpe: 0.0409
Epoch 14/50, Train Loss: 0.0267, Val Sharpe: 0.0410
```

```
Epoch 15/50, Train Loss: -0.1631, Val Sharpe: 0.0410
Epoch 16/50, Train Loss: 0.0455, Val Sharpe: 0.0410
Epoch 17/50, Train Loss: 0.1131, Val Sharpe: 0.0410
Epoch 18/50, Train Loss: 0.0091, Val Sharpe: 0.0409
Epoch 19/50, Train Loss: -0.0606, Val Sharpe: 0.0409
Epoch 20/50, Train Loss: -0.0165, Val Sharpe: 0.0409
Epoch 21/50, Train Loss: -0.0213, Val Sharpe: 0.0411
Epoch 22/50, Train Loss: 0.0331, Val Sharpe: 0.0413
Epoch 23/50, Train Loss: -0.1072, Val Sharpe: 0.0414
Epoch 24/50, Train Loss: -0.0152, Val Sharpe: 0.0416
Epoch 25/50, Train Loss: -0.0618, Val Sharpe: 0.0418
Epoch 26/50, Train Loss: -0.0418, Val Sharpe: 0.0420
Epoch 27/50, Train Loss: -0.0965, Val Sharpe: 0.0423
Epoch 28/50, Train Loss: -0.0163, Val Sharpe: 0.0426
Epoch 29/50, Train Loss: -0.0803, Val Sharpe: 0.0430
Epoch 30/50, Train Loss: -0.0379, Val Sharpe: 0.0436
Epoch 31/50, Train Loss: -0.1333, Val Sharpe: 0.0442
Epoch 32/50, Train Loss: -0.0073, Val Sharpe: 0.0447
Epoch 33/50, Train Loss: -0.0673, Val Sharpe: 0.0453
Epoch 34/50, Train Loss: 0.0048, Val Sharpe: 0.0459
Epoch 35/50, Train Loss: 0.0281, Val Sharpe: 0.0466
Epoch 36/50, Train Loss: 0.0160, Val Sharpe: 0.0474
Epoch 37/50, Train Loss: -0.0686, Val Sharpe: 0.0482
Epoch 38/50, Train Loss: -0.0429, Val Sharpe: 0.0491
Epoch 39/50, Train Loss: -0.0379, Val Sharpe: 0.0497
Epoch 40/50, Train Loss: -0.1132, Val Sharpe: 0.0503
Epoch 41/50, Train Loss: -0.0076, Val Sharpe: 0.0498
Epoch 42/50, Train Loss: 0.0021, Val Sharpe: 0.0498
Epoch 43/50, Train Loss: -0.1061, Val Sharpe: 0.0497
Epoch 44/50, Train Loss: -0.0352, Val Sharpe: 0.0499
Epoch 45/50, Train Loss: -0.0345, Val Sharpe: 0.0502
Epoch 46/50, Train Loss: -0.0653, Val Sharpe: 0.0515
Epoch 47/50, Train Loss: 0.0322, Val Sharpe: 0.0513
Epoch 48/50, Train Loss: -0.0150, Val Sharpe: 0.0529
Epoch 49/50, Train Loss: -0.0567, Val Sharpe: 0.0537
Epoch 50/50, Train Loss: -0.0353, Val Sharpe: 0.0541

--- Testing LSTM Combination 4/27: {'lr': 0.0001, 'weight_decay':
0.0001, 'patience': 5} ---
Epoch 1/50, Train Loss: -0.1455, Val Sharpe: 0.0400
Epoch 2/50, Train Loss: -0.0530, Val Sharpe: 0.0398
Epoch 3/50, Train Loss: 0.0486, Val Sharpe: 0.0398
Epoch 4/50, Train Loss: 0.1690, Val Sharpe: 0.0399
Epoch 5/50, Train Loss: 0.3410, Val Sharpe: 0.0399
Epoch 6/50, Train Loss: 0.0170, Val Sharpe: 0.0400
Epoch 7/50, Train Loss: 0.0359, Val Sharpe: 0.0400
Epoch 8/50, Train Loss: -0.0284, Val Sharpe: 0.0401
Epoch 9/50, Train Loss: -0.0382, Val Sharpe: 0.0401
Epoch 10/50, Train Loss: -0.0236, Val Sharpe: 0.0402
```

```
Epoch 11/50, Train Loss: 0.0735, Val Sharpe: 0.0403
Epoch 12/50, Train Loss: -0.0108, Val Sharpe: 0.0406
Epoch 13/50, Train Loss: -0.0013, Val Sharpe: 0.0407
Epoch 14/50, Train Loss: 0.0139, Val Sharpe: 0.0409
Epoch 15/50, Train Loss: -0.0352, Val Sharpe: 0.0410
Epoch 16/50, Train Loss: 0.0556, Val Sharpe: 0.0411
Epoch 17/50, Train Loss: -0.0435, Val Sharpe: 0.0412
Epoch 18/50, Train Loss: -0.0241, Val Sharpe: 0.0413
Epoch 19/50, Train Loss: -0.0248, Val Sharpe: 0.0414
Epoch 20/50, Train Loss: -0.0029, Val Sharpe: 0.0417
Epoch 21/50, Train Loss: -0.0467, Val Sharpe: 0.0418
Epoch 22/50, Train Loss: -0.3414, Val Sharpe: 0.0421
Epoch 23/50, Train Loss: 0.0311, Val Sharpe: 0.0431
Epoch 24/50, Train Loss: -0.0343, Val Sharpe: 0.0433
Epoch 25/50, Train Loss: -0.0205, Val Sharpe: 0.0435
Epoch 26/50, Train Loss: -0.0957, Val Sharpe: 0.0436
Epoch 27/50, Train Loss: -0.0075, Val Sharpe: 0.0438
Epoch 28/50, Train Loss: 0.0200, Val Sharpe: 0.0439
Epoch 29/50, Train Loss: -0.0342, Val Sharpe: 0.0439
Epoch 30/50, Train Loss: 0.0350, Val Sharpe: 0.0439
Epoch 31/50, Train Loss: -0.0456, Val Sharpe: 0.0437
Epoch 32/50, Train Loss: -0.0282, Val Sharpe: 0.0438
Epoch 33/50, Train Loss: -0.0010, Val Sharpe: 0.0440
Epoch 34/50, Train Loss: -0.0615, Val Sharpe: 0.0440
Epoch 35/50, Train Loss: -0.0245, Val Sharpe: 0.0439
Epoch 36/50, Train Loss: -0.0414, Val Sharpe: 0.0440
Epoch 37/50, Train Loss: -0.0018, Val Sharpe: 0.0440
Epoch 38/50, Train Loss: -0.0234, Val Sharpe: 0.0441
Epoch 39/50, Train Loss: 0.0047, Val Sharpe: 0.0442
Epoch 40/50, Train Loss: -0.0372, Val Sharpe: 0.0443
Epoch 41/50, Train Loss: -0.0287, Val Sharpe: 0.0443
Epoch 42/50, Train Loss: -0.1027, Val Sharpe: 0.0444
Epoch 43/50, Train Loss: 0.0158, Val Sharpe: 0.0445
Epoch 44/50, Train Loss: -0.0584, Val Sharpe: 0.0445
Epoch 45/50, Train Loss: -0.0126, Val Sharpe: 0.0446
Epoch 46/50, Train Loss: -0.0083, Val Sharpe: 0.0447
Epoch 47/50, Train Loss: -0.0353, Val Sharpe: 0.0448
Epoch 48/50, Train Loss: -0.0189, Val Sharpe: 0.0452
Epoch 49/50, Train Loss: 0.0198, Val Sharpe: 0.0453
Epoch 50/50, Train Loss: -0.0602, Val Sharpe: 0.0454

--- Testing LSTM Combination 5/27: {'lr': 0.0001, 'weight_decay':
0.0001, 'patience': 7} ---
Epoch 1/50, Train Loss: -0.0316, Val Sharpe: 0.0416
Epoch 2/50, Train Loss: -0.0416, Val Sharpe: 0.0417
Epoch 3/50, Train Loss: -0.0223, Val Sharpe: 0.0418
Epoch 4/50, Train Loss: -0.0375, Val Sharpe: 0.0419
Epoch 5/50, Train Loss: 0.1798, Val Sharpe: 0.0419
Epoch 6/50, Train Loss: -0.0334, Val Sharpe: 0.0419
```

```
Epoch 7/50, Train Loss: -0.0269, Val Sharpe: 0.0420
Epoch 8/50, Train Loss: -0.0087, Val Sharpe: 0.0420
Epoch 9/50, Train Loss: -0.0044, Val Sharpe: 0.0420
Epoch 10/50, Train Loss: -0.0701, Val Sharpe: 0.0421
Epoch 11/50, Train Loss: 0.1075, Val Sharpe: 0.0422
Epoch 12/50, Train Loss: 0.0266, Val Sharpe: 0.0422
Epoch 13/50, Train Loss: -0.0117, Val Sharpe: 0.0423
Epoch 14/50, Train Loss: -0.0471, Val Sharpe: 0.0425
Epoch 15/50, Train Loss: 0.0551, Val Sharpe: 0.0426
Epoch 16/50, Train Loss: -0.0032, Val Sharpe: 0.0427
Epoch 17/50, Train Loss: 0.0167, Val Sharpe: 0.0427
Epoch 18/50, Train Loss: -0.0441, Val Sharpe: 0.0429
Epoch 19/50, Train Loss: 0.0068, Val Sharpe: 0.0429
Epoch 20/50, Train Loss: -0.0738, Val Sharpe: 0.0429
Epoch 21/50, Train Loss: 0.0006, Val Sharpe: 0.0432
Epoch 22/50, Train Loss: -0.0430, Val Sharpe: 0.0433
Epoch 23/50, Train Loss: -0.0409, Val Sharpe: 0.0434
Epoch 24/50, Train Loss: -0.0135, Val Sharpe: 0.0436
Epoch 25/50, Train Loss: -0.0688, Val Sharpe: 0.0439
Epoch 26/50, Train Loss: -0.0383, Val Sharpe: 0.0442
Epoch 27/50, Train Loss: 0.0336, Val Sharpe: 0.0444
Epoch 28/50, Train Loss: -0.0075, Val Sharpe: 0.0448
Epoch 29/50, Train Loss: -0.1105, Val Sharpe: 0.0452
Epoch 30/50, Train Loss: -0.0433, Val Sharpe: 0.0455
Epoch 31/50, Train Loss: 0.0323, Val Sharpe: 0.0462
Epoch 32/50, Train Loss: -0.0712, Val Sharpe: 0.0466
Epoch 33/50, Train Loss: -0.0546, Val Sharpe: 0.0471
Epoch 34/50, Train Loss: -0.0201, Val Sharpe: 0.0483
Epoch 35/50, Train Loss: -0.0557, Val Sharpe: 0.0496
Epoch 36/50, Train Loss: 0.4701, Val Sharpe: 0.0504
Epoch 37/50, Train Loss: -0.0487, Val Sharpe: 0.0504
Epoch 38/50, Train Loss: 0.0072, Val Sharpe: 0.0506
Epoch 39/50, Train Loss: -0.0195, Val Sharpe: 0.0511
Epoch 40/50, Train Loss: -0.0344, Val Sharpe: 0.0516
Epoch 41/50, Train Loss: -0.0256, Val Sharpe: 0.0519
Epoch 42/50, Train Loss: -0.0379, Val Sharpe: 0.0521
Epoch 43/50, Train Loss: 0.0407, Val Sharpe: 0.0526
Epoch 44/50, Train Loss: -0.0370, Val Sharpe: 0.0530
Epoch 45/50, Train Loss: 0.0086, Val Sharpe: 0.0534
Epoch 46/50, Train Loss: 0.0147, Val Sharpe: 0.0539
Epoch 47/50, Train Loss: -0.0648, Val Sharpe: 0.0545
Epoch 48/50, Train Loss: -0.0140, Val Sharpe: 0.0548
Epoch 49/50, Train Loss: -0.3293, Val Sharpe: 0.0553
Epoch 50/50, Train Loss: -0.1072, Val Sharpe: 0.0558

--- Testing LSTM Combination 6/27: {'lr': 0.0001, 'weight_decay':
0.0001, 'patience': 10} ---
Epoch 1/50, Train Loss: 0.0113, Val Sharpe: 0.0403
Epoch 2/50, Train Loss: -0.0203, Val Sharpe: 0.0406
```

```
Epoch 3/50, Train Loss: -0.0458, Val Sharpe: 0.0408
Epoch 4/50, Train Loss: 0.0017, Val Sharpe: 0.0410
Epoch 5/50, Train Loss: -0.1029, Val Sharpe: 0.0412
Epoch 6/50, Train Loss: -0.0498, Val Sharpe: 0.0410
Epoch 7/50, Train Loss: -0.0286, Val Sharpe: 0.0412
Epoch 8/50, Train Loss: -0.0660, Val Sharpe: 0.0415
Epoch 9/50, Train Loss: -0.0221, Val Sharpe: 0.0416
Epoch 10/50, Train Loss: -0.0474, Val Sharpe: 0.0417
Epoch 11/50, Train Loss: 0.0564, Val Sharpe: 0.0419
Epoch 12/50, Train Loss: -0.0171, Val Sharpe: 0.0421
Epoch 13/50, Train Loss: -0.0422, Val Sharpe: 0.0424
Epoch 14/50, Train Loss: 0.0176, Val Sharpe: 0.0425
Epoch 15/50, Train Loss: -0.0098, Val Sharpe: 0.0427
Epoch 16/50, Train Loss: -0.0029, Val Sharpe: 0.0428
Epoch 17/50, Train Loss: -0.0153, Val Sharpe: 0.0429
Epoch 18/50, Train Loss: -0.0235, Val Sharpe: 0.0431
Epoch 19/50, Train Loss: -0.0278, Val Sharpe: 0.0433
Epoch 20/50, Train Loss: -0.0349, Val Sharpe: 0.0435
Epoch 21/50, Train Loss: -0.2982, Val Sharpe: 0.0438
Epoch 22/50, Train Loss: -0.0777, Val Sharpe: 0.0440
Epoch 23/50, Train Loss: 0.0143, Val Sharpe: 0.0442
Epoch 24/50, Train Loss: -0.0307, Val Sharpe: 0.0442
Epoch 25/50, Train Loss: -0.0283, Val Sharpe: 0.0441
Epoch 26/50, Train Loss: -0.0053, Val Sharpe: 0.0443
Epoch 27/50, Train Loss: -0.0033, Val Sharpe: 0.0446
Epoch 28/50, Train Loss: -0.0207, Val Sharpe: 0.0448
Epoch 29/50, Train Loss: 0.0922, Val Sharpe: 0.0451
Epoch 30/50, Train Loss: -0.0215, Val Sharpe: 0.0453
Epoch 31/50, Train Loss: -0.0715, Val Sharpe: 0.0458
Epoch 32/50, Train Loss: -0.0063, Val Sharpe: 0.0463
Epoch 33/50, Train Loss: -0.0768, Val Sharpe: 0.0467
Epoch 34/50, Train Loss: -0.0036, Val Sharpe: 0.0472
Epoch 35/50, Train Loss: -0.1057, Val Sharpe: 0.0475
Epoch 36/50, Train Loss: -0.0457, Val Sharpe: 0.0474
Epoch 37/50, Train Loss: -0.0467, Val Sharpe: 0.0479
Epoch 38/50, Train Loss: 0.3105, Val Sharpe: 0.0479
Epoch 39/50, Train Loss: -0.1007, Val Sharpe: 0.0464
Epoch 40/50, Train Loss: 0.0075, Val Sharpe: 0.0462
Epoch 41/50, Train Loss: -0.0190, Val Sharpe: 0.0462
Epoch 42/50, Train Loss: 0.0008, Val Sharpe: 0.0462
Epoch 43/50, Train Loss: 0.0276, Val Sharpe: 0.0463
Epoch 44/50, Train Loss: -0.0419, Val Sharpe: 0.0465
Epoch 45/50, Train Loss: -0.0314, Val Sharpe: 0.0466
Epoch 46/50, Train Loss: 0.0704, Val Sharpe: 0.0467
Epoch 47/50, Train Loss: -0.0121, Val Sharpe: 0.0468
Epoch 48/50, Train Loss: 0.0042, Val Sharpe: 0.0470
Early stopping triggered after 48 epochs due to no improvement for 10
epochs.
```

```
--- Testing LSTM Combination 7/27: {'lr': 0.0001, 'weight_decay': 
0.001, 'patience': 5} ---
Epoch 1/50, Train Loss: 0.0140, Val Sharpe: 0.0417
Epoch 2/50, Train Loss: -0.0740, Val Sharpe: 0.0418
Epoch 3/50, Train Loss: 0.0349, Val Sharpe: 0.0419
Epoch 4/50, Train Loss: -0.1290, Val Sharpe: 0.0419
Epoch 5/50, Train Loss: 0.0535, Val Sharpe: 0.0420
Epoch 6/50, Train Loss: -0.0995, Val Sharpe: 0.0422
Epoch 7/50, Train Loss: -0.1838, Val Sharpe: 0.0424
Epoch 8/50, Train Loss: 0.0610, Val Sharpe: 0.0424
Epoch 9/50, Train Loss: -0.0234, Val Sharpe: 0.0424
Epoch 10/50, Train Loss: 0.0084, Val Sharpe: 0.0425
Epoch 11/50, Train Loss: -0.0010, Val Sharpe: 0.0425
Epoch 12/50, Train Loss: -0.0352, Val Sharpe: 0.0426
Epoch 13/50, Train Loss: 0.0100, Val Sharpe: 0.0426
Epoch 14/50, Train Loss: 0.0388, Val Sharpe: 0.0426
Epoch 15/50, Train Loss: -0.0014, Val Sharpe: 0.0427
Epoch 16/50, Train Loss: 0.0424, Val Sharpe: 0.0427
Epoch 17/50, Train Loss: -0.0125, Val Sharpe: 0.0428
Epoch 18/50, Train Loss: -0.3597, Val Sharpe: 0.0428
Epoch 19/50, Train Loss: -0.0205, Val Sharpe: 0.0428
Epoch 20/50, Train Loss: 0.0144, Val Sharpe: 0.0428
Epoch 21/50, Train Loss: -0.0249, Val Sharpe: 0.0428
Epoch 22/50, Train Loss: 0.0180, Val Sharpe: 0.0428
Epoch 23/50, Train Loss: -0.0264, Val Sharpe: 0.0428
Epoch 24/50, Train Loss: -0.0053, Val Sharpe: 0.0428
Epoch 25/50, Train Loss: -0.0211, Val Sharpe: 0.0429
Epoch 26/50, Train Loss: -0.0508, Val Sharpe: 0.0429
Epoch 27/50, Train Loss: 0.0118, Val Sharpe: 0.0430
Epoch 28/50, Train Loss: -0.0648, Val Sharpe: 0.0430
Epoch 29/50, Train Loss: -0.1600, Val Sharpe: 0.0430
Epoch 30/50, Train Loss: 0.0091, Val Sharpe: 0.0428
Epoch 31/50, Train Loss: 0.1957, Val Sharpe: 0.0429
Epoch 32/50, Train Loss: -0.0515, Val Sharpe: 0.0428
Epoch 33/50, Train Loss: 0.0466, Val Sharpe: 0.0429
Early stopping triggered after 33 epochs due to no improvement for 5 
epochs.

--- Testing LSTM Combination 8/27: {'lr': 0.0001, 'weight_decay': 
0.001, 'patience': 7} ---
Epoch 1/50, Train Loss: -0.0103, Val Sharpe: 0.0402
Epoch 2/50, Train Loss: -0.0370, Val Sharpe: 0.0403
Epoch 3/50, Train Loss: 0.0191, Val Sharpe: 0.0403
Epoch 4/50, Train Loss: -0.0178, Val Sharpe: 0.0403
Epoch 5/50, Train Loss: -0.0388, Val Sharpe: 0.0403
Epoch 6/50, Train Loss: 0.0698, Val Sharpe: 0.0405
Epoch 7/50, Train Loss: 0.0872, Val Sharpe: 0.0406
Epoch 8/50, Train Loss: -0.0224, Val Sharpe: 0.0407
Epoch 9/50, Train Loss: -0.0112, Val Sharpe: 0.0409
```

```
Epoch 10/50, Train Loss: -0.0080, Val Sharpe: 0.0410
Epoch 11/50, Train Loss: 0.1188, Val Sharpe: 0.0412
Epoch 12/50, Train Loss: -0.0268, Val Sharpe: 0.0413
Epoch 13/50, Train Loss: -0.0597, Val Sharpe: 0.0413
Epoch 14/50, Train Loss: -0.0047, Val Sharpe: 0.0413
Epoch 15/50, Train Loss: -0.0310, Val Sharpe: 0.0414
Epoch 16/50, Train Loss: -0.0430, Val Sharpe: 0.0416
Epoch 17/50, Train Loss: -0.0385, Val Sharpe: 0.0417
Epoch 18/50, Train Loss: -0.0274, Val Sharpe: 0.0418
Epoch 19/50, Train Loss: -0.0083, Val Sharpe: 0.0418
Epoch 20/50, Train Loss: -0.0462, Val Sharpe: 0.0419
Epoch 21/50, Train Loss: -0.0606, Val Sharpe: 0.0419
Epoch 22/50, Train Loss: -0.0435, Val Sharpe: 0.0420
Epoch 23/50, Train Loss: -0.0861, Val Sharpe: 0.0422
Epoch 24/50, Train Loss: -0.0387, Val Sharpe: 0.0423
Epoch 25/50, Train Loss: 0.0069, Val Sharpe: 0.0424
Epoch 26/50, Train Loss: -0.0994, Val Sharpe: 0.0426
Epoch 27/50, Train Loss: 0.0223, Val Sharpe: 0.0436
Epoch 28/50, Train Loss: -0.0539, Val Sharpe: 0.0438
Epoch 29/50, Train Loss: -0.0421, Val Sharpe: 0.0439
Epoch 30/50, Train Loss: -0.0032, Val Sharpe: 0.0440
Epoch 31/50, Train Loss: -0.0336, Val Sharpe: 0.0441
Epoch 32/50, Train Loss: 0.0253, Val Sharpe: 0.0443
Epoch 33/50, Train Loss: -0.0358, Val Sharpe: 0.0444
Epoch 34/50, Train Loss: -0.0056, Val Sharpe: 0.0445
Epoch 35/50, Train Loss: -0.0110, Val Sharpe: 0.0447
Epoch 36/50, Train Loss: -0.0296, Val Sharpe: 0.0448
Epoch 37/50, Train Loss: 0.0714, Val Sharpe: 0.0450
Epoch 38/50, Train Loss: -0.0011, Val Sharpe: 0.0452
Epoch 39/50, Train Loss: -0.0588, Val Sharpe: 0.0455
Epoch 40/50, Train Loss: -0.0112, Val Sharpe: 0.0457
Epoch 41/50, Train Loss: -0.0552, Val Sharpe: 0.0460
Epoch 42/50, Train Loss: 0.0075, Val Sharpe: 0.0461
Epoch 43/50, Train Loss: -0.0255, Val Sharpe: 0.0465
Epoch 44/50, Train Loss: -0.0031, Val Sharpe: 0.0468
Epoch 45/50, Train Loss: -0.0257, Val Sharpe: 0.0471
Epoch 46/50, Train Loss: 0.0955, Val Sharpe: 0.0476
Epoch 47/50, Train Loss: -0.0491, Val Sharpe: 0.0481
Epoch 48/50, Train Loss: 0.0265, Val Sharpe: 0.0487
Epoch 49/50, Train Loss: -0.0085, Val Sharpe: 0.0490
Epoch 50/50, Train Loss: -0.0410, Val Sharpe: 0.0494

--- Testing LSTM Combination 9/27: {'lr': 0.0001, 'weight_decay':
0.001, 'patience': 10} ---
Epoch 1/50, Train Loss: -0.0304, Val Sharpe: 0.0405
Epoch 2/50, Train Loss: -0.0531, Val Sharpe: 0.0406
Epoch 3/50, Train Loss: -0.0094, Val Sharpe: 0.0407
Epoch 4/50, Train Loss: 0.0532, Val Sharpe: 0.0407
Epoch 5/50, Train Loss: 0.0245, Val Sharpe: 0.0407
```

```
Epoch 6/50, Train Loss: -0.0821, Val Sharpe: 0.0406
Epoch 7/50, Train Loss: -0.0312, Val Sharpe: 0.0405
Epoch 8/50, Train Loss: 0.0498, Val Sharpe: 0.0404
Epoch 9/50, Train Loss: 0.0463, Val Sharpe: 0.0404
Epoch 10/50, Train Loss: -0.0270, Val Sharpe: 0.0404
Epoch 11/50, Train Loss: -0.0207, Val Sharpe: 0.0405
Epoch 12/50, Train Loss: -0.1620, Val Sharpe: 0.0405
Epoch 13/50, Train Loss: -0.0081, Val Sharpe: 0.0405
Epoch 14/50, Train Loss: -0.0138, Val Sharpe: 0.0406
Early stopping triggered after 14 epochs due to no improvement for 10
epochs.

--- Testing LSTM Combination 10/27: {'lr': 0.001, 'weight_decay': 1e-
05, 'patience': 5} ---
Epoch 1/50, Train Loss: -0.0393, Val Sharpe: 0.0403
Epoch 2/50, Train Loss: 0.0062, Val Sharpe: 0.0413
Epoch 3/50, Train Loss: -0.0071, Val Sharpe: 0.0425
Epoch 4/50, Train Loss: 0.0239, Val Sharpe: 0.0465
Epoch 5/50, Train Loss: -0.0311, Val Sharpe: 0.0420
Epoch 6/50, Train Loss: -0.0575, Val Sharpe: 0.0432
Epoch 7/50, Train Loss: -0.0586, Val Sharpe: 0.0422
Epoch 8/50, Train Loss: -0.0367, Val Sharpe: 0.0417
Epoch 9/50, Train Loss: -0.0235, Val Sharpe: 0.0428
Early stopping triggered after 9 epochs due to no improvement for 5
epochs.

--- Testing LSTM Combination 11/27: {'lr': 0.001, 'weight_decay': 1e-
05, 'patience': 7} ---
Epoch 1/50, Train Loss: -0.0124, Val Sharpe: 0.0418
Epoch 2/50, Train Loss: -0.0281, Val Sharpe: 0.0442
Epoch 3/50, Train Loss: -0.0178, Val Sharpe: 0.0475
Epoch 4/50, Train Loss: -0.0366, Val Sharpe: 0.0542
Epoch 5/50, Train Loss: 0.0033, Val Sharpe: 0.0534
Epoch 6/50, Train Loss: -0.0538, Val Sharpe: 0.0571
Epoch 7/50, Train Loss: -0.0935, Val Sharpe: 0.0537
Epoch 8/50, Train Loss: 0.0777, Val Sharpe: 0.0493
Epoch 9/50, Train Loss: -0.1250, Val Sharpe: 0.0540
Epoch 10/50, Train Loss: -0.4156, Val Sharpe: 0.0470
Epoch 11/50, Train Loss: -0.0566, Val Sharpe: 0.0448
Epoch 12/50, Train Loss: -0.5415, Val Sharpe: 0.0458
Epoch 13/50, Train Loss: 0.0303, Val Sharpe: 0.0599
Epoch 14/50, Train Loss: -0.0624, Val Sharpe: 0.0601
Epoch 15/50, Train Loss: -0.0448, Val Sharpe: 0.0587
Epoch 16/50, Train Loss: -0.0436, Val Sharpe: 0.0583
Epoch 17/50, Train Loss: -0.0507, Val Sharpe: 0.0555
Epoch 18/50, Train Loss: 0.0824, Val Sharpe: 0.0526
Epoch 19/50, Train Loss: -0.0598, Val Sharpe: 0.0505
Epoch 20/50, Train Loss: -0.0201, Val Sharpe: 0.0493
Epoch 21/50, Train Loss: -0.1234, Val Sharpe: 0.0470
```

```
Early stopping triggered after 21 epochs due to no improvement for 7
epochs.

--- Testing LSTM Combination 12/27: {'lr': 0.001, 'weight_decay': 1e-
05, 'patience': 10} ---
Epoch 1/50, Train Loss: -0.0262, Val Sharpe: 0.0392
Epoch 2/50, Train Loss: 0.1054, Val Sharpe: 0.0406
Epoch 3/50, Train Loss: -0.0001, Val Sharpe: 0.0399
Epoch 4/50, Train Loss: -0.0641, Val Sharpe: 0.0400
Epoch 5/50, Train Loss: -0.1534, Val Sharpe: 0.0414
Epoch 6/50, Train Loss: -0.0000, Val Sharpe: 0.0425
Epoch 7/50, Train Loss: 0.0147, Val Sharpe: 0.0427
Epoch 8/50, Train Loss: 0.0230, Val Sharpe: 0.0449
Epoch 9/50, Train Loss: -0.0082, Val Sharpe: 0.0455
Epoch 10/50, Train Loss: -0.0102, Val Sharpe: 0.0450
Epoch 11/50, Train Loss: -0.0419, Val Sharpe: 0.0435
Epoch 12/50, Train Loss: -0.0395, Val Sharpe: 0.0435
Epoch 13/50, Train Loss: -0.0570, Val Sharpe: 0.0440
Epoch 14/50, Train Loss: -0.0850, Val Sharpe: 0.0424
Epoch 15/50, Train Loss: -0.1050, Val Sharpe: 0.0491
Epoch 16/50, Train Loss: -0.0777, Val Sharpe: 0.0419
Epoch 17/50, Train Loss: -0.1223, Val Sharpe: 0.0417
Epoch 18/50, Train Loss: -0.1032, Val Sharpe: 0.0417
Epoch 19/50, Train Loss: -0.1802, Val Sharpe: 0.0424
Epoch 20/50, Train Loss: -0.1807, Val Sharpe: 0.0418
Epoch 21/50, Train Loss: -0.0881, Val Sharpe: 0.0449
Epoch 22/50, Train Loss: -0.0900, Val Sharpe: 0.1178
Epoch 23/50, Train Loss: -0.2170, Val Sharpe: 0.0891
Epoch 24/50, Train Loss: -0.1461, Val Sharpe: 0.0904
Epoch 25/50, Train Loss: -0.0775, Val Sharpe: 0.1034
Epoch 26/50, Train Loss: -0.0595, Val Sharpe: 0.1048
Epoch 27/50, Train Loss: -0.0925, Val Sharpe: 0.1037
Epoch 28/50, Train Loss: -0.1566, Val Sharpe: 0.1070
Epoch 29/50, Train Loss: -0.1279, Val Sharpe: 0.1231
Epoch 30/50, Train Loss: -0.1139, Val Sharpe: 0.1228
Epoch 31/50, Train Loss: -0.1412, Val Sharpe: 0.1189
Epoch 32/50, Train Loss: -0.1130, Val Sharpe: 0.0371
Epoch 33/50, Train Loss: -0.1049, Val Sharpe: 0.1015
Epoch 34/50, Train Loss: -0.0587, Val Sharpe: 0.1182
Epoch 35/50, Train Loss: -0.1825, Val Sharpe: 0.0370
Epoch 36/50, Train Loss: -0.2793, Val Sharpe: 0.0370
Epoch 37/50, Train Loss: -0.0318, Val Sharpe: 0.0370
Epoch 38/50, Train Loss: -0.0079, Val Sharpe: 0.0371
Epoch 39/50, Train Loss: -0.0788, Val Sharpe: 0.0373
Early stopping triggered after 39 epochs due to no improvement for 10
epochs.

--- Testing LSTM Combination 13/27: {'lr': 0.001, 'weight_decay':
0.0001, 'patience': 5} ---
Epoch 1/50, Train Loss: -0.0031, Val Sharpe: 0.0438
```

```
Epoch 2/50, Train Loss: -0.0851, Val Sharpe: 0.0442
Epoch 3/50, Train Loss: 0.0006, Val Sharpe: 0.0502
Epoch 4/50, Train Loss: -0.0610, Val Sharpe: 0.0525
Epoch 5/50, Train Loss: 0.0949, Val Sharpe: 0.0502
Epoch 6/50, Train Loss: -0.0754, Val Sharpe: 0.0530
Epoch 7/50, Train Loss: -0.1497, Val Sharpe: 0.0499
Epoch 8/50, Train Loss: -0.1325, Val Sharpe: 0.0436
Epoch 9/50, Train Loss: -0.0702, Val Sharpe: 0.0421
Epoch 10/50, Train Loss: -0.1105, Val Sharpe: 0.0415
Epoch 11/50, Train Loss: -0.1200, Val Sharpe: 0.0413
Early stopping triggered after 11 epochs due to no improvement for 5
epochs.

--- Testing LSTM Combination 14/27: {'lr': 0.001, 'weight_decay':
0.0001, 'patience': 7} ---
Epoch 1/50, Train Loss: -0.0447, Val Sharpe: 0.0404
Epoch 2/50, Train Loss: 0.4162, Val Sharpe: 0.0420
Epoch 3/50, Train Loss: -0.0347, Val Sharpe: 0.0406
Epoch 4/50, Train Loss: -0.1062, Val Sharpe: 0.0404
Epoch 5/50, Train Loss: -0.0065, Val Sharpe: 0.0404
Epoch 6/50, Train Loss: 0.0145, Val Sharpe: 0.0404
Epoch 7/50, Train Loss: -0.0262, Val Sharpe: 0.0405
Epoch 8/50, Train Loss: 0.0205, Val Sharpe: 0.0405
Epoch 9/50, Train Loss: -0.1011, Val Sharpe: 0.0406
Early stopping triggered after 9 epochs due to no improvement for 7
epochs.

--- Testing LSTM Combination 15/27: {'lr': 0.001, 'weight_decay':
0.0001, 'patience': 10} ---
Epoch 1/50, Train Loss: -0.4902, Val Sharpe: 0.0428
Epoch 2/50, Train Loss: -0.0233, Val Sharpe: 0.0432
Epoch 3/50, Train Loss: -0.0135, Val Sharpe: 0.0436
Epoch 4/50, Train Loss: 0.0088, Val Sharpe: 0.0440
Epoch 5/50, Train Loss: 0.0138, Val Sharpe: 0.0448
Epoch 6/50, Train Loss: -0.0205, Val Sharpe: 0.0462
Epoch 7/50, Train Loss: -0.0417, Val Sharpe: 0.0479
Epoch 8/50, Train Loss: -0.0352, Val Sharpe: 0.0492
Epoch 9/50, Train Loss: -0.0588, Val Sharpe: 0.0498
Epoch 10/50, Train Loss: -0.0835, Val Sharpe: 0.0506
Epoch 11/50, Train Loss: -0.0183, Val Sharpe: 0.0528
Epoch 12/50, Train Loss: -0.0111, Val Sharpe: 0.0532
Epoch 13/50, Train Loss: -0.0376, Val Sharpe: 0.0521
Epoch 14/50, Train Loss: -0.0406, Val Sharpe: 0.0541
Epoch 15/50, Train Loss: -0.0761, Val Sharpe: 0.0554
Epoch 16/50, Train Loss: -0.0684, Val Sharpe: 0.0632
Epoch 17/50, Train Loss: 0.0210, Val Sharpe: 0.0542
Epoch 18/50, Train Loss: -0.0423, Val Sharpe: 0.0548
Epoch 19/50, Train Loss: -0.0422, Val Sharpe: 0.0433
Epoch 20/50, Train Loss: -0.0757, Val Sharpe: 0.0876
```

```
Epoch 21/50, Train Loss: 0.0095, Val Sharpe: 0.0434
Epoch 22/50, Train Loss: -0.1413, Val Sharpe: 0.0584
Epoch 23/50, Train Loss: -0.1086, Val Sharpe: 0.0783
Epoch 24/50, Train Loss: -0.0978, Val Sharpe: 0.0521
Epoch 25/50, Train Loss: -0.1466, Val Sharpe: 0.0967
Epoch 26/50, Train Loss: -0.0790, Val Sharpe: 0.0929
Epoch 27/50, Train Loss: -0.0735, Val Sharpe: 0.1259
Epoch 28/50, Train Loss: -0.5242, Val Sharpe: 0.0965
Epoch 29/50, Train Loss: -0.1261, Val Sharpe: 0.0881
Epoch 30/50, Train Loss: -0.1056, Val Sharpe: 0.1134
Epoch 31/50, Train Loss: 0.0689, Val Sharpe: 0.1282
Epoch 32/50, Train Loss: -0.1173, Val Sharpe: 0.0771
Epoch 33/50, Train Loss: -0.0866, Val Sharpe: 0.0862
Epoch 34/50, Train Loss: -0.1342, Val Sharpe: 0.0441
Epoch 35/50, Train Loss: -0.0603, Val Sharpe: 0.0447
Epoch 36/50, Train Loss: -0.0682, Val Sharpe: 0.0459
Epoch 37/50, Train Loss: -0.1148, Val Sharpe: 0.0445
Epoch 38/50, Train Loss: -0.0456, Val Sharpe: 0.0445
Epoch 39/50, Train Loss: -0.1038, Val Sharpe: 0.0450
Epoch 40/50, Train Loss: 0.0025, Val Sharpe: 0.0728
Epoch 41/50, Train Loss: -0.1045, Val Sharpe: 0.0893
Early stopping triggered after 41 epochs due to no improvement for 10
epochs.

--- Testing LSTM Combination 16/27: {'lr': 0.001, 'weight_decay':
0.001, 'patience': 5} ---
Epoch 1/50, Train Loss: 0.0366, Val Sharpe: 0.0378
Epoch 2/50, Train Loss: 0.0153, Val Sharpe: 0.0393
Epoch 3/50, Train Loss: 0.0305, Val Sharpe: 0.0404
Epoch 4/50, Train Loss: -0.0097, Val Sharpe: 0.0420
Epoch 5/50, Train Loss: -0.0179, Val Sharpe: 0.0439
Epoch 6/50, Train Loss: -0.0485, Val Sharpe: 0.0460
Epoch 7/50, Train Loss: 0.0939, Val Sharpe: 0.0465
Epoch 8/50, Train Loss: -0.0240, Val Sharpe: 0.0472
Epoch 9/50, Train Loss: -0.1563, Val Sharpe: 0.0467
Epoch 10/50, Train Loss: -0.0708, Val Sharpe: 0.0616
Epoch 11/50, Train Loss: 0.0240, Val Sharpe: 0.0602
Epoch 12/50, Train Loss: -0.0300, Val Sharpe: 0.0526
Epoch 13/50, Train Loss: -0.0602, Val Sharpe: 0.0526
Epoch 14/50, Train Loss: -0.0873, Val Sharpe: 0.0473
Epoch 15/50, Train Loss: -0.0723, Val Sharpe: 0.0787
Epoch 16/50, Train Loss: -0.0573, Val Sharpe: 0.0454
Epoch 17/50, Train Loss: -0.1059, Val Sharpe: 0.0444
Epoch 18/50, Train Loss: -0.1231, Val Sharpe: 0.0467
Epoch 19/50, Train Loss: -0.0924, Val Sharpe: 0.0505
Epoch 20/50, Train Loss: -0.0735, Val Sharpe: 0.0443
Early stopping triggered after 20 epochs due to no improvement for 5
epochs.
```

```
--- Testing LSTM Combination 17/27: {'lr': 0.001, 'weight_decay':
0.001, 'patience': 7} ---
Epoch 1/50, Train Loss: 0.0353, Val Sharpe: 0.0441
Epoch 2/50, Train Loss: -0.0530, Val Sharpe: 0.0448
Epoch 3/50, Train Loss: 0.0334, Val Sharpe: 0.0473
Epoch 4/50, Train Loss: -0.0161, Val Sharpe: 0.0497
Epoch 5/50, Train Loss: 0.0054, Val Sharpe: 0.0524
Epoch 6/50, Train Loss: -0.0248, Val Sharpe: 0.0511
Epoch 7/50, Train Loss: -0.0269, Val Sharpe: 0.0476
Epoch 8/50, Train Loss: 0.0110, Val Sharpe: 0.0511
Epoch 9/50, Train Loss: -0.0380, Val Sharpe: 0.0417
Epoch 10/50, Train Loss: -0.0547, Val Sharpe: 0.0483
Epoch 11/50, Train Loss: 0.0404, Val Sharpe: 0.0705
Epoch 12/50, Train Loss: -0.0144, Val Sharpe: 0.0800
Epoch 13/50, Train Loss: -0.0281, Val Sharpe: 0.0628
Epoch 14/50, Train Loss: -0.2318, Val Sharpe: 0.0572
Epoch 15/50, Train Loss: -0.1353, Val Sharpe: 0.0468
Epoch 16/50, Train Loss: -0.0448, Val Sharpe: 0.0517
Epoch 17/50, Train Loss: -0.0908, Val Sharpe: 0.0470
Epoch 18/50, Train Loss: -0.0869, Val Sharpe: 0.0437
Epoch 19/50, Train Loss: -0.0506, Val Sharpe: 0.0448
Early stopping triggered after 19 epochs due to no improvement for 7
epochs.

--- Testing LSTM Combination 18/27: {'lr': 0.001, 'weight_decay':
0.001, 'patience': 10} ---
Epoch 1/50, Train Loss: -0.0230, Val Sharpe: 0.0405
Epoch 2/50, Train Loss: -0.0106, Val Sharpe: 0.0416
Epoch 3/50, Train Loss: -0.0014, Val Sharpe: 0.0446
Epoch 4/50, Train Loss: -0.0627, Val Sharpe: 0.0468
Epoch 5/50, Train Loss: 0.0099, Val Sharpe: 0.0495
Epoch 6/50, Train Loss: -0.0115, Val Sharpe: 0.0529
Epoch 7/50, Train Loss: -0.0369, Val Sharpe: 0.0514
Epoch 8/50, Train Loss: -0.0776, Val Sharpe: 0.0514
Epoch 9/50, Train Loss: 0.0074, Val Sharpe: 0.0601
Epoch 10/50, Train Loss: -0.0868, Val Sharpe: 0.0517
Epoch 11/50, Train Loss: -0.0104, Val Sharpe: 0.0495
Epoch 12/50, Train Loss: -0.0813, Val Sharpe: 0.0479
Epoch 13/50, Train Loss: -0.0770, Val Sharpe: 0.0457
Epoch 14/50, Train Loss: -0.0747, Val Sharpe: 0.0444
Epoch 15/50, Train Loss: -0.1179, Val Sharpe: 0.0426
Epoch 16/50, Train Loss: -0.1270, Val Sharpe: 0.0418
Epoch 17/50, Train Loss: 0.0365, Val Sharpe: 0.0417
Epoch 18/50, Train Loss: -0.0894, Val Sharpe: 0.1092
Epoch 19/50, Train Loss: -0.2932, Val Sharpe: 0.0769
Epoch 20/50, Train Loss: -0.0731, Val Sharpe: 0.0877
Epoch 21/50, Train Loss: -0.0232, Val Sharpe: 0.0988
Epoch 22/50, Train Loss: -0.0156, Val Sharpe: 0.1036
Epoch 23/50, Train Loss: 0.0432, Val Sharpe: 0.1035
```

```
Epoch 24/50, Train Loss: -0.1073, Val Sharpe: 0.1065
Epoch 25/50, Train Loss: -0.0393, Val Sharpe: 0.1012
Epoch 26/50, Train Loss: 0.0181, Val Sharpe: 0.1034
Epoch 27/50, Train Loss: -0.0826, Val Sharpe: 0.1063
Epoch 28/50, Train Loss: -0.1190, Val Sharpe: 0.1096
Epoch 29/50, Train Loss: -0.0360, Val Sharpe: 0.1087
Epoch 30/50, Train Loss: -0.0575, Val Sharpe: 0.0942
Epoch 31/50, Train Loss: -0.0032, Val Sharpe: 0.0877
Epoch 32/50, Train Loss: -0.0447, Val Sharpe: 0.0825
Epoch 33/50, Train Loss: -0.0693, Val Sharpe: 0.0535
Epoch 34/50, Train Loss: -0.0493, Val Sharpe: 0.0487
Epoch 35/50, Train Loss: -0.1569, Val Sharpe: 0.0447
Epoch 36/50, Train Loss: -0.0783, Val Sharpe: 0.0427
Epoch 37/50, Train Loss: -0.0762, Val Sharpe: 0.0462
Epoch 38/50, Train Loss: -0.0244, Val Sharpe: 0.0441
Early stopping triggered after 38 epochs due to no improvement for 10
epochs.

--- Testing LSTM Combination 19/27: {'lr': 0.01, 'weight_decay': 1e-
05, 'patience': 5} ---
Epoch 1/50, Train Loss: 0.0535, Val Sharpe: 0.0427
Epoch 2/50, Train Loss: 0.0936, Val Sharpe: 0.0594
Epoch 3/50, Train Loss: -0.0200, Val Sharpe: 0.0347
Epoch 4/50, Train Loss: -0.0236, Val Sharpe: 0.0633
Epoch 5/50, Train Loss: -0.0361, Val Sharpe: 0.0751
Epoch 6/50, Train Loss: -0.0973, Val Sharpe: 0.0516
Epoch 7/50, Train Loss: -0.0439, Val Sharpe: 0.0733
Epoch 8/50, Train Loss: -0.1180, Val Sharpe: 0.0719
Epoch 9/50, Train Loss: -0.1174, Val Sharpe: 0.0750
Epoch 10/50, Train Loss: -0.0808, Val Sharpe: 0.0746
Early stopping triggered after 10 epochs due to no improvement for 5
epochs.

--- Testing LSTM Combination 20/27: {'lr': 0.01, 'weight_decay': 1e-
05, 'patience': 7} ---
Epoch 1/50, Train Loss: -0.0276, Val Sharpe: 0.0505
Epoch 2/50, Train Loss: 0.0216, Val Sharpe: 0.0495
Epoch 3/50, Train Loss: -0.0458, Val Sharpe: 0.0396
Epoch 4/50, Train Loss: 0.0772, Val Sharpe: 0.0425
Epoch 5/50, Train Loss: -0.0269, Val Sharpe: 0.0461
Epoch 6/50, Train Loss: -0.0266, Val Sharpe: 0.0484
Epoch 7/50, Train Loss: -0.0139, Val Sharpe: 0.0506
Epoch 8/50, Train Loss: -0.2197, Val Sharpe: 0.0515
Epoch 9/50, Train Loss: -0.0737, Val Sharpe: 0.0467
Epoch 10/50, Train Loss: -0.0950, Val Sharpe: 0.0482
Epoch 11/50, Train Loss: -0.1907, Val Sharpe: 0.0424
Epoch 12/50, Train Loss: -0.0510, Val Sharpe: 0.0409
Epoch 13/50, Train Loss: 0.0450, Val Sharpe: 0.0425
Epoch 14/50, Train Loss: -0.0378, Val Sharpe: 0.0451
```

```
Epoch 15/50, Train Loss: -0.0434, Val Sharpe: 0.0458
Early stopping triggered after 15 epochs due to no improvement for 7
epochs.

--- Testing LSTM Combination 21/27: {'lr': 0.01, 'weight_decay': 1e-
05, 'patience': 10} ---
Epoch 1/50, Train Loss: 0.0197, Val Sharpe: 0.0750
Epoch 2/50, Train Loss: -0.0666, Val Sharpe: 0.0710
Epoch 3/50, Train Loss: -0.1152, Val Sharpe: 0.0667
Epoch 4/50, Train Loss: -0.0145, Val Sharpe: 0.0435
Epoch 5/50, Train Loss: -0.0319, Val Sharpe: 0.0403
Epoch 6/50, Train Loss: -0.0820, Val Sharpe: 0.0174
Epoch 7/50, Train Loss: -0.0677, Val Sharpe: 0.0208
Epoch 8/50, Train Loss: 0.1157, Val Sharpe: 0.0422
Epoch 9/50, Train Loss: -0.0365, Val Sharpe: 0.0429
Epoch 10/50, Train Loss: 0.0049, Val Sharpe: 0.0420
Epoch 11/50, Train Loss: 0.0251, Val Sharpe: 0.0493
Early stopping triggered after 11 epochs due to no improvement for 10
epochs.

--- Testing LSTM Combination 22/27: {'lr': 0.01, 'weight_decay':
0.0001, 'patience': 5} ---
Epoch 1/50, Train Loss: -0.1182, Val Sharpe: 0.0377
Epoch 2/50, Train Loss: -0.1188, Val Sharpe: 0.0373
Epoch 3/50, Train Loss: -0.0897, Val Sharpe: 0.0374
Epoch 4/50, Train Loss: -0.0148, Val Sharpe: 0.0373
Epoch 5/50, Train Loss: -0.0675, Val Sharpe: 0.0377
Epoch 6/50, Train Loss: -0.0783, Val Sharpe: 0.0409
Epoch 7/50, Train Loss: -0.0209, Val Sharpe: 0.0560
Epoch 8/50, Train Loss: -0.0363, Val Sharpe: 0.0384
Epoch 9/50, Train Loss: -0.0491, Val Sharpe: 0.0359
Epoch 10/50, Train Loss: -0.0180, Val Sharpe: 0.0446
Epoch 11/50, Train Loss: -0.0676, Val Sharpe: 0.0388
Epoch 12/50, Train Loss: -0.1298, Val Sharpe: 0.0420
Early stopping triggered after 12 epochs due to no improvement for 5
epochs.

--- Testing LSTM Combination 23/27: {'lr': 0.01, 'weight_decay':
0.0001, 'patience': 7} ---
Epoch 1/50, Train Loss: 0.0048, Val Sharpe: 0.0550
Epoch 2/50, Train Loss: -0.0538, Val Sharpe: 0.0488
Epoch 3/50, Train Loss: -0.0380, Val Sharpe: 0.0480
Epoch 4/50, Train Loss: -0.0121, Val Sharpe: 0.0692
Epoch 5/50, Train Loss: -0.0351, Val Sharpe: 0.0595
Epoch 6/50, Train Loss: -0.0551, Val Sharpe: 0.0471
Epoch 7/50, Train Loss: -0.0413, Val Sharpe: 0.0470
Epoch 8/50, Train Loss: 0.0055, Val Sharpe: 0.0467
Epoch 9/50, Train Loss: -0.0443, Val Sharpe: 0.0584
Epoch 10/50, Train Loss: -0.0035, Val Sharpe: 0.0442
Epoch 11/50, Train Loss: -0.1241, Val Sharpe: 0.0479
```

```
Early stopping triggered after 11 epochs due to no improvement for 7
epochs.

--- Testing LSTM Combination 24/27: {'lr': 0.01, 'weight_decay':
0.0001, 'patience': 10} ---
Epoch 1/50, Train Loss: -0.0083, Val Sharpe: 0.0447
Epoch 2/50, Train Loss: -0.0496, Val Sharpe: 0.0469
Epoch 3/50, Train Loss: -0.0661, Val Sharpe: 0.0426
Epoch 4/50, Train Loss: 0.0067, Val Sharpe: 0.0534
Epoch 5/50, Train Loss: -0.0366, Val Sharpe: 0.0303
Epoch 6/50, Train Loss: -0.1657, Val Sharpe: 0.0276
Epoch 7/50, Train Loss: -0.0607, Val Sharpe: 0.0221
Epoch 8/50, Train Loss: 0.0007, Val Sharpe: 0.0215
Epoch 9/50, Train Loss: 0.0117, Val Sharpe: 0.0452
Epoch 10/50, Train Loss: -0.0284, Val Sharpe: 0.0506
Epoch 11/50, Train Loss: -0.1105, Val Sharpe: 0.0411
Epoch 12/50, Train Loss: -0.0734, Val Sharpe: 0.0411
Epoch 13/50, Train Loss: -0.3793, Val Sharpe: 0.0428
Epoch 14/50, Train Loss: -0.0617, Val Sharpe: 0.0344
Early stopping triggered after 14 epochs due to no improvement for 10
epochs.

--- Testing LSTM Combination 25/27: {'lr': 0.01, 'weight_decay':
0.001, 'patience': 5} ---
Epoch 1/50, Train Loss: -0.0366, Val Sharpe: 0.0403
Epoch 2/50, Train Loss: 0.0075, Val Sharpe: 0.0462
Epoch 3/50, Train Loss: 0.0274, Val Sharpe: 0.0618
Epoch 4/50, Train Loss: -0.0033, Val Sharpe: 0.0463
Epoch 5/50, Train Loss: -0.0395, Val Sharpe: 0.0395
Epoch 6/50, Train Loss: -0.0625, Val Sharpe: 0.0401
Epoch 7/50, Train Loss: -0.0059, Val Sharpe: 0.0214
Epoch 8/50, Train Loss: -0.0274, Val Sharpe: 0.0085
Early stopping triggered after 8 epochs due to no improvement for 5
epochs.

--- Testing LSTM Combination 26/27: {'lr': 0.01, 'weight_decay':
0.001, 'patience': 7} ---
Epoch 1/50, Train Loss: 0.0183, Val Sharpe: 0.0365
Epoch 2/50, Train Loss: -0.0212, Val Sharpe: 0.0420
Epoch 3/50, Train Loss: 0.0516, Val Sharpe: 0.0426
Epoch 4/50, Train Loss: -0.0350, Val Sharpe: 0.0595
Epoch 5/50, Train Loss: -0.0932, Val Sharpe: 0.0380
Epoch 6/50, Train Loss: -0.0299, Val Sharpe: 0.0681
Epoch 7/50, Train Loss: -0.0467, Val Sharpe: 0.0421
Epoch 8/50, Train Loss: 0.0426, Val Sharpe: 0.0417
Epoch 9/50, Train Loss: -0.0145, Val Sharpe: 0.0414
Epoch 10/50, Train Loss: -0.0481, Val Sharpe: 0.0378
Epoch 11/50, Train Loss: -0.0433, Val Sharpe: 0.0375
Epoch 12/50, Train Loss: -0.1137, Val Sharpe: 0.0373
Epoch 13/50, Train Loss: 0.0025, Val Sharpe: 0.0392
```

```
Early stopping triggered after 13 epochs due to no improvement for 7
epochs.

--- Testing LSTM Combination 27/27: {'lr': 0.01, 'weight_decay':
0.001, 'patience': 10} ---
Epoch 1/50, Train Loss: -0.0293, Val Sharpe: 0.0429
Epoch 2/50, Train Loss: 0.0037, Val Sharpe: 0.0396
Epoch 3/50, Train Loss: -0.0640, Val Sharpe: 0.0374
Epoch 4/50, Train Loss: -0.0294, Val Sharpe: 0.0372
Epoch 5/50, Train Loss: -0.0703, Val Sharpe: 0.0372
Epoch 6/50, Train Loss: -0.0406, Val Sharpe: 0.0408
Epoch 7/50, Train Loss: -0.0515, Val Sharpe: 0.0425
Epoch 8/50, Train Loss: -0.0906, Val Sharpe: 0.0433
Epoch 9/50, Train Loss: -0.0692, Val Sharpe: 0.0377
Epoch 10/50, Train Loss: -0.1249, Val Sharpe: 0.0373
Epoch 11/50, Train Loss: -0.0844, Val Sharpe: 0.0373
Epoch 12/50, Train Loss: -0.0194, Val Sharpe: 0.0377
Epoch 13/50, Train Loss: -0.0549, Val Sharpe: 0.0386
Epoch 14/50, Train Loss: -0.0585, Val Sharpe: 0.0376
Epoch 15/50, Train Loss: -0.2076, Val Sharpe: 0.0378
Epoch 16/50, Train Loss: -0.0417, Val Sharpe: 0.0379
Epoch 17/50, Train Loss: -0.0855, Val Sharpe: 0.0384
Epoch 18/50, Train Loss: -0.0021, Val Sharpe: 0.0382
Early stopping triggered after 18 epochs due to no improvement for 10
epochs.

--- Best LSTM Model Configuration (Sharpe Ratio Loss) ---
{'hyperparams': {'lr': 0.001, 'weight_decay': 0.0001, 'patience': 7},
'cumulative_return': np.float32(1.1225703), 'volatility':
np.float32(0.029699596), 'sharpe_ratio': np.float32(0.26411006),
'portfolio_returns': array([ 0.08521871,  0.04092786, -0.00227703,
0.00898712, -0.01767445,
        0.02185285, -0.05075501,  0.05387014,  0.02532765,
0.03983331,
       -0.02151876, -0.00128867, -0.0133384 ,  0.02244584,
0.01388565,
       -0.00645931,  0.04027934,  0.05459628,  0.02968456,
0.01352162,
        0.04414675, -0.01247472,  0.02254893, -0.00143895,
0.02834682,
       -0.01211677,  0.01678579, -0.04055452, -0.03109585, -
0.02105609,
        0.03307104,  0.0366002 , -0.02360087,  0.00099474, -
0.04218084,
        0.00453414,  0.02591551, -0.00474577, -0.05308568, -
0.01883734,
        0.06708606,  0.02761858,  0.02934824,  0.00252134,
0.00370979,
        0.01110455,  0.02254425,  0.00165762,  0.00041262, -
```

```
0.03140058,
        0.02555228,  0.02259774, -0.00856001,  0.01473518,
0.03007751,
       -0.00816082,  0.01343178,  0.01947761, -0.02346167, -
0.00602581,
        0.02941811, -0.00371456, -0.01810715,  0.00367932, -
0.07586797,
        0.05745942,  0.02802274,  0.00179924,  0.02626478,
0.02641253,
       -0.00245426,  0.02353588,  0.0370275 , -0.00108694,
0.0083405 ,
        0.05747322,  0.00882682, -0.03449386, -0.02676218, -
0.03449415,
       -0.01289473,  0.06422274,  0.01599209, -0.01463458, -
0.0559933 ,
        0.06366709,  0.01240799,  0.01864223, -0.0065406 , -
0.00201122,
        0.01489782,  0.02649604, -0.03394334,  0.0772232 , -
0.02189577,
        0.0269857 ,  0.00469742,  0.03405093,  0.00745737, -
0.01566924,
        0.0065786 , -0.02404652], dtype=float32)}
```

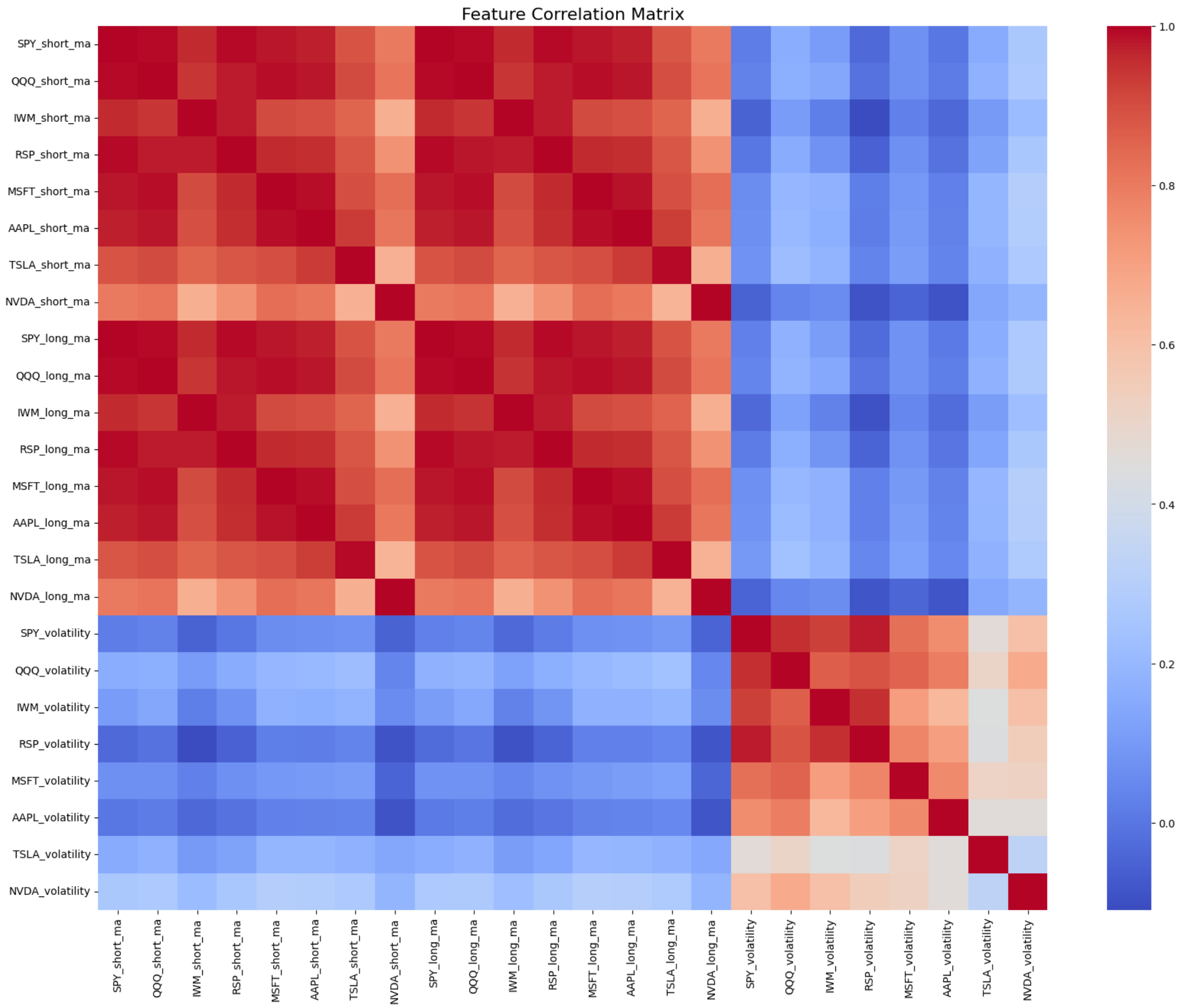
Feature Correlations

```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestRegressor

# 1. Create a new DataFrame, features_only_df, by dropping all columns
ending with '_returns' from `combined_weekly_features`.
# This ensures we are only correlating input features against each
other.
features_only_df = combined_weekly_features.loc[:,
~combined_weekly_features.columns.str.endswith('_returns')]
print("DataFrame 'features_only_df' created successfully.")

# 2. Calculate the correlation matrix of features_only_df.
correlation_matrix = features_only_df.corr()
print("Correlation matrix calculated successfully.")

# 3. Create and display a heatmap of the correlation matrix for input
features.
plt.figure(figsize=(20, 15)) # Adjust figure size for better
readability
sns.heatmap(correlation_matrix, annot=False, cmap='coolwarm',
fmt=".2f") # annot=False for large matrix
plt.title('Feature Correlation Matrix', fontsize=16)
plt.xticks(rotation=90)
```

```
plt.yticks(rotation=0)
plt.show()
print("Feature correlation matrix heatmap displayed successfully.")

# 4. Prepare a dictionary named feature_importances to store feature
importances for each asset.
feature_importances = {}
print("Dictionary 'feature_importances' initialized.")

# 5. Iterate through each target asset column (defined by
`target_cols` from preprocessing) in `combined_weekly_features`.
# `target_cols` should be globally available from the preprocessing
step.
for asset_return_col in target_cols:
    # Identify the current asset's return column as y_asset.
    y_asset = combined_weekly_features[asset_return_col]

    # Use features_only_df as the feature set X_asset for the
RandomForestRegressor.
    # `X_asset` and `y_asset` should already be aligned by index and
have no NaNs due to prior preprocessing steps.
    X_asset = features_only_df.copy()

    # Initialize and train a RandomForestRegressor model.
    # n_estimators=100 for robust importance, random_state for
reproducibility, n_jobs=-1 for parallel processing.
    rf_model = RandomForestRegressor(n_estimators=100,
random_state=42, n_jobs=-1)
    rf_model.fit(X_asset, y_asset)

    # Extract feature importances and store them as a pandas Series,
indexed by feature names.
    feature_importances[asset_return_col.replace('_returns', '')] =
pd.Series(rf_model.feature_importances_, index=X_asset.columns)

print("Feature importances calculated for each asset successfully.")

# 6. Convert the feature_importances dictionary into a pandas
DataFrame named feature_importances_df.
feature_importances_df = pd.DataFrame(feature_importances)
print("DataFrame 'feature_importances_df' created successfully.")

# 7. Aggregate feature importances across all assets by summing their
importances.
aggregated_feature_importances =
feature_importances_df.sum(axis=1).sort_values(ascending=False)
print("Top 15 aggregated features:")
print(aggregated_feature_importances.head(15))

# 8. Plot the feature importances for each asset using a bar plot for
```

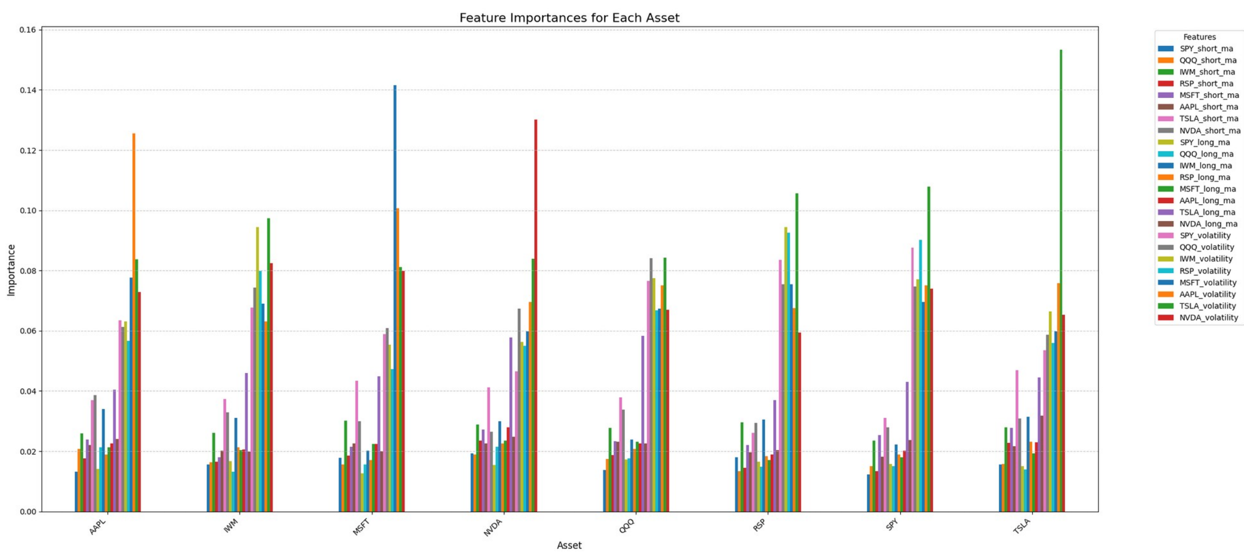
```
visual comparison.
plt.figure(figsize=(20, 10)) # Adjust figure size for better
readability
# Transpose the DataFrame to have assets on the x-axis and features in
the legend.
feature_importances_df.T.plot(kind='bar', figsize=(20, 10))
plt.title('Feature Importances for Each Asset', fontsize=16)
plt.xlabel('Asset', fontsize=12)
plt.ylabel('Importance', fontsize=12)
plt.xticks(rotation=45, ha='right') # Rotate x-axis labels for
readability
plt.tight_layout() # Adjust layout to prevent labels from overlapping
plt.legend(title='Features', bbox_to_anchor=(1.05, 1), loc='upper
left') # Place legend outside the plot area
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
print("Feature importances bar plot for each asset displayed
successfully.")
```

```
DataFrame 'features_only_df' created successfully.
Correlation matrix calculated successfully.
```




```
Feature correlation matrix heatmap displayed successfully.
Dictionary 'feature_importances' initialized.
Feature importances calculated for each asset successfully.
DataFrame 'feature_importances_df' created successfully.
Top 15 aggregated features:
TSLA_volatility    0.797298
AAPL_volatility    0.652182
NVDA_volatility    0.631022
MSFT_volatility    0.620087
IWM_volatility     0.584644
QQQ_volatility     0.556660
RSP_volatility     0.544005
SPY_volatility     0.537693
TSLA_long_ma       0.371963
TSLA_short_ma      0.300706
NVDA_short_ma      0.249905
IWM_long_ma        0.223535
```

```
IWM_short_ma        0.219706
MSFT_short_ma       0.189108
NVDA_long_ma        0.186971
dtype: float64

<Figure size 2000x1000 with 0 Axes>
```




```
Feature importances bar plot for each asset displayed successfully.
```

Summary & Visualizations

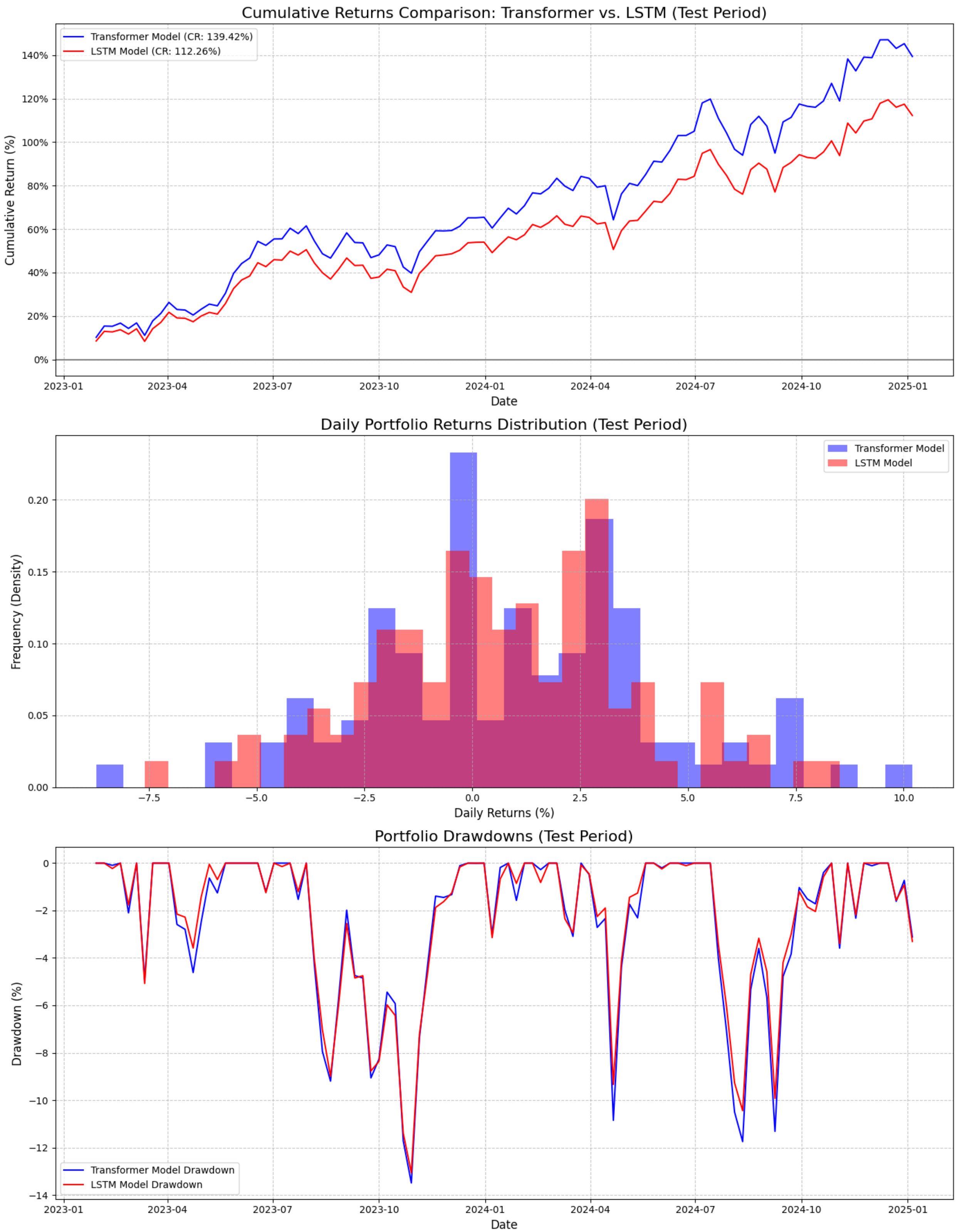
```python
import matplotlib.pyplot as plt
import matplotlib.ticker as mticker

# Redefine the evaluate_model function to return portfolio_returns
def evaluate_model(model, data_loader, y_scaler):
    model.eval() # Set the model to evaluation mode
    all_weights = []
    all_returns = []

    with torch.no_grad(): # Disable gradient calculation for inference
        for X_batch, y_batch in data_loader:
            # Move data to the same device as the model
            device = next(model.parameters()).device
            X_batch = X_batch.to(device)
            y_batch = y_batch.to(device)

            predicted_weights = model(X_batch)

            y_batch_np = y_batch.cpu().numpy()
            denormalized_returns_batch = 
y_scaler.inverse_transform(y_batch_np)
```

```
            all_weights.append(predicted_weights.cpu().numpy())
            all_returns.append(denormalized_returns_batch)

    # Concatenate all predicted weights and actual denormalized
returns
    all_weights = np.concatenate(all_weights, axis=0)
    all_returns = np.concatenate(all_returns, axis=0)

    # Calculate portfolio returns (dot product of weights and returns)
    portfolio_returns = np.sum(all_weights * all_returns, axis=1)

    # Calculate the cumulative returns
    # Add 1 to portfolio_returns before cumulative product
    cumulative_return = np.prod(1 + portfolio_returns) - 1

    # Calculate the volatility
    volatility = np.std(portfolio_returns)

    # Calculate the Sharpe Ratio with numerical stability
    sharpe_ratio = np.mean(portfolio_returns) / (volatility + 1e-6)

    # Return portfolio_returns along with the other metrics
    return cumulative_return, volatility, sharpe_ratio,
portfolio_returns

print("Model evaluation function `evaluate_model` redefined
successfully to include portfolio returns.")


# Extract portfolio returns directly from the stored best
configurations
transformer_test_portfolio_returns =
best_config_overall['portfolio_returns']
lstm_test_portfolio_returns =
best_config_overall_lstm['portfolio_returns']

print("Transformer and LSTM models portfolio returns loaded from best
configurations.")


# --- Calculate and plot cumulative returns and drawdowns ---
if transformer_test_portfolio_returns is not None and
lstm_test_portfolio_returns is not None:
    transformer_cumulative_returns = (1 +
transformer_test_portfolio_returns).cumprod() - 1
    lstm_cumulative_returns = (1 +
lstm_test_portfolio_returns).cumprod() - 1

    # Get the dates for the test set for plotting
    # `train_size`, `val_size`, `sequence_length` are assumed to be
```

```
available from data preprocessing
    test_start_index = train_size + val_size + sequence_length

    # Assuming `combined_weekly_features` is still available from
initial data loading/preprocessing.
    if 'combined_weekly_features' in globals():
        # Ensure the length of test_dates matches the length of
cumulative_returns_series
        test_dates = combined_weekly_features.index[test_start_index:
test_start_index + len(transformer_cumulative_returns)]
    else:
        print("Warning: combined_weekly_features not found. Plotting
dates might be incorrect.")
        test_dates =
pd.to_datetime(np.arange(len(transformer_cumulative_returns))) #
Fallback

    plt.figure(figsize=(14, 18)) # Increased figure height for more
subplots

    # Subplot 1: Cumulative Returns
    plt.subplot(3, 1, 1)
    plt.plot(test_dates, transformer_cumulative_returns * 100,
label=f"Transformer Model (CR:
{best_config_overall['cumulative_return']:.2%})", color='blue')
    plt.plot(test_dates, lstm_cumulative_returns * 100, label=f"LSTM
Model (CR: {best_config_overall_lstm['cumulative_return']:.2%})",
color='red')
    plt.title('Cumulative Returns Comparison: Transformer vs. LSTM
(Test Period)', fontsize=16)
    plt.xlabel('Date', fontsize=12)
    plt.ylabel('Cumulative Return (%)', fontsize=12) # Changed label
    plt.gca().yaxis.set_major_formatter(mticker.PercentFormatter()) #
Format as percentage
    plt.legend(fontsize=10)
    plt.grid(True, linestyle='--', alpha=0.7)
    plt.axhline(y=0, color='grey', linestyle='-')

    # Subplot 2: Daily Portfolio Returns Distribution
    plt.subplot(3, 1, 2)
    plt.hist(transformer_test_portfolio_returns * 100, bins=30,
alpha=0.5, label='Transformer Model', color='blue', density=True) #
Changed to percentage
    plt.hist(lstm_test_portfolio_returns * 100, bins=30, alpha=0.5,
label='LSTM Model', color='red', density=True) # Changed to percentage
    plt.title('Daily Portfolio Returns Distribution (Test Period)',
fontsize=16)
    plt.xlabel('Daily Returns (%)', fontsize=12) # Changed label
    plt.ylabel('Frequency (Density)', fontsize=12)
    plt.legend(fontsize=10)
```

```
    plt.grid(True, linestyle='--', alpha=0.7)

    # Subplot 3: Drawdown Plots
    def calculate_drawdowns(cumulative_returns_series):
        if len(cumulative_returns_series) == 0: # Handle empty series
            return np.array([])
        # Corrected: cumulative_returns_series are already compounded
- 1. Need to add 1 back.
        compounded_returns = 1 + cumulative_returns_series
        peak = np.maximum.accumulate(compounded_returns)
        # Ensure peak is used instead of undefined running_max
        drawdowns = (compounded_returns - peak) / (peak + 1e-9)
        return drawdowns

    transformer_drawdowns =
calculate_drawdowns(transformer_cumulative_returns)
    lstm_drawdowns = calculate_drawdowns(lstm_cumulative_returns)

    plt.subplot(3, 1, 3)
    plt.plot(test_dates, transformer_drawdowns * 100,
label='Transformer Model Drawdown', color='blue') # Changed to
percentage
    plt.plot(test_dates, lstm_drawdowns * 100, label='LSTM Model
Drawdown', color='red') # Changed to percentage
    plt.title('Portfolio Drawdowns (Test Period)', fontsize=16)
    plt.xlabel('Date', fontsize=12)
    plt.ylabel('Drawdown (%)', fontsize=12) # Changed label
    plt.legend(fontsize=10)
    plt.grid(True, linestyle='--', alpha=0.7)

    plt.tight_layout()
    plt.show()
    print("\nCumulative returns, returns distribution, and drawdown
plots generated successfully.")
else:
    print("\nError: Could not retrieve portfolio returns for
plotting.")
```

```
Model evaluation function `evaluate_model` redefined successfully to
include portfolio returns.
Transformer and LSTM models portfolio returns loaded from best
configurations.
```



Cumulative Returns Comparison: Transformer vs. LSTM (Test Period)
Transformer Model (CR: 139.42%)
LSTM Model (CR: 112.26%)
Cumulative Return (%)
Date
Daily Portfolio Returns Distribution (Test Period)
Transformer Model
LSTM Model
Frequency (Density)
Daily Returns (%)
Portfolio Drawdowns (Test Period)
Transformer Model Drawdown
LSTM Model Drawdown
Drawdown (%)
Date

```
Cumulative returns, returns distribution, and drawdown plots generated successfully.
```