

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ - ĐHQGHN

VIỆN TRÍ TUỆ NHÂN TẠO



BÁO CÁO DỰ ÁN

Đề tài
ỨNG DỤNG MÔ HÌNH LSTM VÀ
TRANSFORMER VÀO BÀI TOÁN DỰ ĐOÁN
NHIỆT ĐỘ

Giảng viên hướng dẫn: TS.Triệu Hải Long

Nhóm sinh viên thực hiện:

Họ tên	MSV
Vũ Minh Đức	22022587
Đào Duy Hưng	22022589
Thái Nguyễn Hoàng Bách	22022672
Quách Đắc Chính	22022518
Hoàng Ngọc Hào	22022668

Chương I: Lời nói đầu.....	3
1.1. Mở đầu.....	3
1.2. Mục tiêu của đề tài.....	3
Chương II: Thu thập và tiền xử lý dữ liệu.....	3
2.1 Thu thập dữ liệu nhiệt độ.....	3
2.2. Tiền xử lý dữ liệu.....	5
Chương III: Mô hình LSTM.....	6
3.1 Khái niệm và nguyên lý hoạt động của LSTM.....	6
3.2 Xây dựng mô hình LSTM.....	6
Chương IV: Mô hình Transformer.....	9
4.1 Cấu trúc và nguyên lý hoạt động của Transformer.....	9
4.2 Xây dựng mô hình Transformer.....	10
Chương V. Cài đặt giao diện người dùng.....	11
Chương VI. Ứng dụng và hướng phát triển của đề tài.....	14

Chương I: Lời nói đầu

1.1. Mở đầu

- Học sâu (Deep Learning) là lĩnh vực trí tuệ nhân tạo phát triển nhanh chóng trong những năm gần đây và được ứng dụng trong rất nhiều các lĩnh vực. Trong đó, mô hình mạng nơ-ron hồi quy (RNN) là cốt lõi trong việc xử lý chuỗi thời gian và dữ liệu tuần tự.
- Tuy nhiên, RNN gặp nhiều hạn chế với bài toán dài hạn dẫn đến tình trạng giảm hiệu suất huấn luyện. Để khắc phục điều này, mô hình LSTM (Long Short-Term Memory) đã ra đời, mang lại khả năng lưu trữ thông tin lâu hơn. Mô hình LSTM (Long Short-Term Memory) là một dạng đặc biệt của mạng nơ-ron học sâu, được thiết kế để xử lý vấn đề phụ thuộc dài hạn và biến mất đối với các mạng nơ-ron thông thường. Mô hình này có khả năng lưu giữ thông tin trong thời gian dài và giải quyết vấn đề triệt tiêu thông tin khi quá trình lan truyền ngược. Với cấu trúc đặc biệt và khả năng học được thông qua các cổng đặc biệt, LSTM đã trở thành một công cụ quan trọng trong việc dự đoán các chuỗi thời gian như dự đoán nhiệt độ. Điều này khiến cho việc tìm hiểu và áp dụng mô hình LSTM trở nên hết sức quan trọng trong lĩnh vực dự đoán nhiệt độ.

1.2. Mục tiêu của đề tài

- Tìm hiểu nguyên lý hoạt động của mô hình LSTM
- Triển khai mô hình LSTM cho bài toán dự báo nhiệt độ thời tiết
- Đánh giá kết quả mô hình và tìm kiếm phương hướng phát triển cho bài toán trong tương lai

Chương II: Thu thập và tiền xử lý dữ liệu

2.1 Thu thập dữ liệu nhiệt độ

- Sử dụng `get_api.py` để lấy dữ liệu từ web `weatherapi`

```
# Hàm lấy dữ liệu thời tiết từ API
def fetch_weather_data(location, date):
    date_str = date.strftime("%Y-%m-%d")
    url = f'http://api.weatherapi.com/v1/history.json?key={api_key}&q={location}&dt={date_str}'

    try:
        response = requests.get(url)
        response.raise_for_status()
        data = response.json()

        if 'forecast' in data and 'forecastday' in data['forecast']:
            return data
        else:
            print(f"Error: Unexpected data format for {location} on {date_str}")
            return None
    except requests.exceptions.RequestException as e:
        print(f"Error fetching data for {location} on {date_str}: {e}")
        return None
```

- Sau khi đã lấy dữ liệu thành công từ weatherapi, tiến hành lưu trữ các thuộc tính như city, date, temp_c, humidity, rainfall_mm, windspeed_kph vào file csv

```
# Hàm ghi dữ liệu vào CSV
def write_to_csv(data, filename="weather_data.csv"):
    file_exists = False
    try:
        with open(filename, mode='r', encoding='utf-8') as file:
            file_exists = True
    except FileNotFoundError:
        pass

    with open(filename, mode='a', newline='', encoding='utf-8') as file:
        writer = csv.writer(file)
        if not file_exists:
            # Ghi tiêu đề cột chỉ một lần
            writer.writerow(["city", "date", "temp_c", "humidity", "rainfall_mm", "wind_speed_kph"])
        for entry in data:
            writer.writerow([entry['city'], entry['date'], entry['temp'],
                             entry['humidity'], entry['rainfall'], entry['wind_speed']])
    print(f"Weather data has been written to {filename}")
```

- Dữ liệu được thu thập từ ngày 13/12/2023 đến ngày 12/12/2024 từ 3 thành phố lớn là Hà Nội, Thành phố Hồ Chí Minh và Đà Nẵng

```
# Chương trình chính
if __name__ == "__main__":
    try:
        cities = ['Hanoi', 'Ho Chi Minh City', 'Da Nang']

        end_date = datetime.today()
        start_date = end_date - timedelta(days=365) # 1 year

        filename = f"weather_data_{start_date.strftime('%d_%m_%Y')}_to_{end_date.strftime('%d_%m_%Y')}.csv"

        weather_data = collect_weather_data(cities, start_date, end_date)
        write_to_csv(weather_data, filename)

    except Exception as e:
        print(f"Unexpected error: {e}")
```

2.2. Tiền xử lý dữ liệu

- Bước 1: Loại bỏ các dữ liệu trống
- Bước 2: Tách dữ liệu thành tập X và y
- Bước 3: Chuẩn hóa dữ liệu về khoảng [0, 1] bằng MinMaxScaler để tăng tốc độ hội tụ của mô hình.
- Bước 4: Chia dữ liệu thành các tập huấn luyện, kiểm tra và kiểm định

```
# 1. Đọc dữ liệu từ file CSV
df = pd.read_csv(filename)

print("Dữ liệu ban đầu:")
print(df.head())

# 2. Kiểm tra dữ liệu thiếu và loại bỏ các hàng có giá trị thiếu
df = df.dropna()
if df.empty:
    print("\nLỗi: Dữ liệu sau khi loại bỏ giá trị thiếu bị trống. Kiểm tra lại file CSV.")
    return

print("\nDữ liệu sau khi loại bỏ giá trị thiếu:")
print(df.head())

# 3. Mã hóa dữ liệu phân loại (One-hot encoding cho cột 'city')
if 'city' in df.columns:
    df = pd.get_dummies(df, columns=['city'], drop_first=True)

# 4. Tách dữ liệu thành đầu vào (X) và đầu ra (y)
if 'date' in df.columns:
    df = df.drop(columns=['date']) # Loại bỏ cột ngày nếu không cần thiết

X = df.drop(columns=['temp_c']) # Dữ liệu đầu vào
y = df['temp_c'] # Dữ liệu đầu ra (mục tiêu dự đoán)
```

```
# 5. Chuẩn hóa dữ liệu đầu vào
if X.empty or y.empty:
    print("\nLỗi: Dữ liệu đầu vào hoặc đầu ra bị trống.")
    return

scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)

# 6. Chia dữ liệu thành tập huấn luyện, kiểm tra, và kiểm định
X_train, X_temp, y_train, y_temp = train_test_split(X_scaled, y, test_size=0.3, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)

# 7. Lưu các tập dữ liệu vào file CSV
train_data = pd.DataFrame(X_train, columns=X.columns)
train_data['temp_c'] = y_train.values
train_data.to_csv('train.csv', index=False)

val_data = pd.DataFrame(X_val, columns=X.columns)
val_data['temp_c'] = y_val.values
val_data.to_csv('validation.csv', index=False)

test_data = pd.DataFrame(X_test, columns=X.columns)
test_data['temp_c'] = y_test.values
test_data.to_csv('test.csv', index=False)

print("\nDữ liệu đã được chuẩn hóa và lưu vào các file: train.csv, validation.csv, test.csv")
```

Chương III: Mô hình LSTM

3.1 Khái niệm và nguyên lý hoạt động của LSTM

- Mô hình LSTM là phiên bản cải tiến của RNN, giải quyết vấn đề vanishing gradient nhờ vào các gate trong mô hình.
- LSTM bao gồm ba loại cửa chính:
 - + **Forget Gate**: Quyết định thông tin nào cần loại bỏ khỏi trạng thái bộ nhớ.
 - + **Input Gate**: Xác định thông tin nào cần thêm vào trạng thái bộ nhớ.
 - + **Output Gate**: Quyết định thông tin nào từ trạng thái bộ nhớ được sử dụng để tạo đầu ra.
- Các phương trình chính trong LSTM:
 - + Forget Gate: $f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$
 - + Input Gate:
 - Quyết định mức độ cập nhật thông tin: $i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$
 - Tạo giá trị thông tin mới: $C_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$
 - + Cập nhật trạng thái bộ nhớ: $C_t = f_t * C_{t-1} + i_t * C_t$
 - + Output Gate:
 - Quyết định mức độ đầu ra: $o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$
 - Tạo đầu ra: $h_t = o_t * \tanh(C_t)$

3.2 Xây dựng mô hình LSTM

- Cấu trúc:
 - Sử dụng hai lớp LSTM với 50 đơn vị ẩn mỗi lớp.
 - Một lớp Dense đầu ra với một đơn vị để dự đoán nhiệt độ.
- Sử dụng hàm mất mát Root Mean Squared Error (RMSE) để đánh giá model
- Tối ưu hóa: Adam với learning rate = 0.001.
- Huấn luyện mô hình :
 - Sử dụng EarlyStopping với patience=5 để tránh hiện tượng overfitting.
 - Số epoch tối đa: 50
 - Kích thước batch: 32
- Các bước thực hiện:
 - + Bước 1: Tạo chuỗi thời gian cho tập dữ liệu

```

# Hàm tạo chuỗi thời gian
def create_sequences(data, target, sequence_length):
    X, y = [], []
    for i in range(len(data) - sequence_length):
        X.append(data[i:i+sequence_length])
        y.append(target[i+sequence_length])
    return np.array(X), np.array(y)

# Đặt chiều dài chuỗi thời gian
sequence_length = 30

# Tạo chuỗi dữ liệu cho train, validation, và test
X_train_seq, y_train_seq = create_sequences(X_train_scaled, y_train, sequence_length)
X_val_seq, y_val_seq = create_sequences(X_val_scaled, y_val, sequence_length)
X_test_seq, y_test_seq = create_sequences(X_test_scaled, y_test, sequence_length)

print(f"Shape of X_train_seq: {X_train_seq.shape}")
print(f"Shape of y_train_seq: {y_train_seq.shape}")

```

[11]

```

... Shape of X_train_seq: (738, 30, 5)
Shape of y_train_seq: (738,)

```

+ Bước 2: Xây dựng và huấn luyện mô hình

```

# Xây dựng mô hình LSTM
model = Sequential([
    LSTM(units=50, return_sequences=True, input_shape=(X_train_seq.shape[1], X_train_seq.shape[2])),
    LSTM(units=50),
    Dense(units=1) # Dự đoán một giá trị
])

# Biên dịch mô hình
model.compile(optimizer= Adam(learning_rate = 0.001), loss='mean_squared_error')

# Huấn luyện mô hình
early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

history = model.fit(
    X_train_seq, y_train_seq,
    validation_data=(X_val_seq, y_val_seq),
    epochs=50,
    batch_size=32,
    callbacks=[early_stopping],
    verbose=1
)

```

2]

- + Bước 3: Sử dụng mô hình để dự đoán với dữ liệu là tập kiểm tra. Sau đó đưa ra kết quả RMSE để đánh giá mô hình. Kết quả cho thấy $RMSE = 3.63$. Điều này cho thấy trong các trường hợp bình thường mô hình này có thể sử dụng tuy nhiên trong một số trường hợp yêu cầu độ chính xác cao thì mô hình này chưa đáp ứng được điều đó

```

# Dự đoán trên tập kiểm tra
y_pred_seq = model.predict(X_test_seq)

# Chuyển giá trị dự đoán về khoảng giá trị ban đầu
y_pred_original = y_pred_seq.flatten()
y_test_original = y_test_seq

# Tính toán RMSE
from sklearn.metrics import mean_squared_error
rmse = np.sqrt(mean_squared_error(y_test_original, y_pred_original))
print(f"RMSE on test set: {rmse:.2f}")

```

5/5 ————— 0s 56ms/step
 RMSE on test set: 3.63

+ Bước 4: Xây dựng hàm dự đoán nhiệt độ trong tương lai

```

def forecast_future(data, model, steps, sequence_length):
    future = []
    current_sequence = data[-sequence_length:] # Lấy chuỗi cuối cùng

    for i in range(steps):
        # Đảm bảo dữ liệu có đúng kích thước
        current_sequence = current_sequence.reshape(1, sequence_length, -1)

        # Dự báo giá trị tiếp theo
        next_value = model.predict(current_sequence, verbose=0)

        # Lưu giá trị dự báo
        future.append(next_value[0, 0])
        print(f"Dự báo bước {i + 1}: {next_value[0, 0]}") # In ra từng bước dự báo

        # Tạo một giá trị đặc trưng đầy đủ
        next_features = current_sequence[:, -1, :].copy() # Sao chép hàng cuối cùng
        next_features[0, 0] = next_value[0, 0] # Thay giá trị nhiệt độ dự đoán

        # Cập nhật chuỗi dữ liệu với giá trị mới
        current_sequence = np.concatenate([current_sequence[:, 1:, :], next_features.reshape(1, 1, -1)], axis=1)

    return np.array(future)

```

```

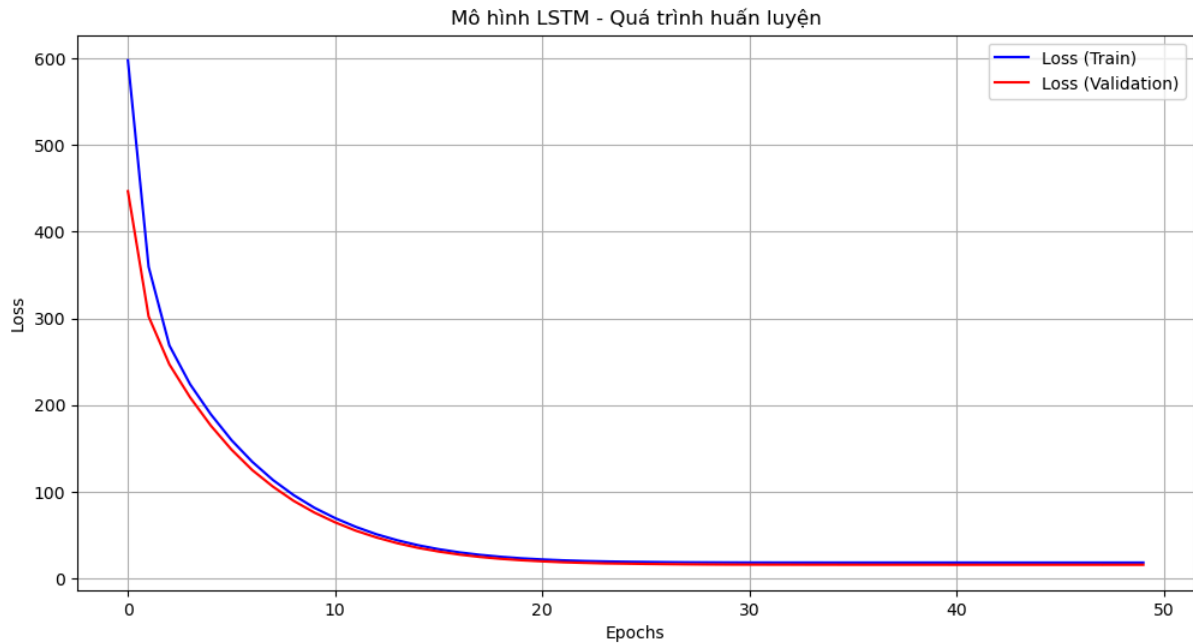
# Dự báo 7 ngày tiếp theo
print("Dự báo 7 ngày tiếp theo bắt đầu...")
future_forecast = forecast_future(X_test_scaled, model, steps=7, sequence_length=sequence_length)

# In ra kết quả dự báo
print("\nDự báo nhiệt độ 7 ngày tiếp theo:")
print(future_forecast)

```

Dự báo 7 ngày tiếp theo bắt đầu...
 Dự báo bước 1: 26.14470672607422
 Dự báo bước 2: 26.139970779418945
 Dự báo bước 3: 26.13783836364746
 Dự báo bước 4: 26.13606071472168
 Dự báo bước 5: 26.134122848510742
 Dự báo bước 6: 26.13214874267578
 Dự báo bước 7: 26.12973976135254

Dự báo nhiệt độ 7 ngày tiếp theo:
 [26.144707 26.13997 26.137838 26.13606 26.134123 26.132149 26.12974]



- Hình ảnh trực quan hóa quá trình huấn luyện model

Chương IV: Mô hình Transformer

4.1 Cấu trúc và nguyên lý hoạt động của Transformer

- Cấu trúc của Transformer
 - + Encoder: Bao gồm nhiều lớp Encoder Layer. Mỗi layer gồm:
 - Multi-Head Self-Attention: Tính Attention đồng thời trên nhiều không gian vector.
 - Feed-Forward Network: Là một mạng fully connected chuyển biểu diễn qua non-linear activation.
 - + Decoder (không áp dụng trong bài toán này): Thường dùng trong bài toán sinh ngôn ngữ.
 - + Positional Encoding: Thêm thông tin vị trí cho mỗi từ trong chuỗi do Transformer không xử lý theo thứ tự như RNN
- Nguyên lý hoạt động của Transformer
 - + **Embedding Dữ Liệu:** Dữ liệu đầu vào (chuỗi) được biểu diễn thành vector embedding.
 - + **Tính Attention:** Sử dụng ma trận Attention để tính tương quan giữa các từ trong chuỗi.
 - Self-Attention giúp mỗi từ "tìm hiểu" các từ khác trong ngữ cảnh toàn bộ chuỗi.
 - + **Feed-Forward:** Dữ liệu sau Attention được chuyển qua các lớp fully connected để biến đổi thành output cuối.

4.2 Xây dựng mô hình Transformer

- Trong vòng lặp qua các epoch:
 - + Forward pass: Mô hình nhận dữ liệu và sinh output dự đoán.
 - + Tính Loss: Sử dụng nnMSELoss()
 - + Backward pass: Tính gradient và cập nhật trọng số qua optimizer.step()
 - + In loss cứ 10 epoch để theo dõi.
- Các bước thực hiện:
 - + Bước 1: Tải và lấy các tham số từ cấu hình

```
# Tải các tham số cấu hình
config = load_config()

# Lấy các tham số từ cấu hình
model_name = config["model"]["name"]
input_size = config["model"]["input_size"]
hidden_size = config["model"]["hidden_size"]
output_size = config["model"]["output_size"]
num_layers = config["model"]["num_layers"]
num_heads = config["model"]["num_heads"]

batch_size = config["training"]["batch_size"]
epochs = config["training"]["epochs"]
learning_rate = config["training"]["learning_rate"]

file_path = config["data"]["file_path"]
target_columns = config["data"]["target_columns"]

global model, model_path
if model_name == 0:
    model = WeatherLSTM(input_size, hidden_size, output_size)
    model_path = "./model/LSTM.h5"
else:
    model = WeatherTransformer(input_size, hidden_size, output_size, num_layers, num_heads)
    model_path = "./model/Transformer.h5"
```

- + Bước 2: Huấn luyện mô hình

```

criterion = nn.MSELoss()
optimizer = optim.Adam(model.parameters(), lr=learning_rate)

X_train_loader, y_train_loader = data_loader('./data/train_data.csv', target_columns=['max', 'min'])
# X_test_loader, y_test_loader = data_loader('test_data.csv', target_columns=['max', 'min'])

for epoch in range(epochs):
    model.train()
    optimizer.zero_grad()

    # Forward pass
    y_pred = model(X_train_loader)

    # Tính toán loss
    loss = criterion(y_pred, y_train_loader)

    # Backward pass và cập nhật trọng số
    loss.backward()
    optimizer.step()

    if (epoch+1) % 10 == 0:
        print(f'Epoch [{epoch+1}/{epochs}], Loss: {loss.item():.4f}')

torch.save(model.state_dict(), model_path)

```

Chương V. Cài đặt giao diện người dùng

- Sử dụng thư viện streamlit để cài đặt giao diện người dùng
- Các bước thực hiện:
 - + Bước 1: Xây dựng hàm display_city_weather để hiển thị dữ liệu và dự báo

```

def display_city_weather(data, city_name, model, scaler, sequence_length=30):
    # Lọc dữ liệu cho thành phố được chọn
    city_data = data[data['city'] == city_name]

    # Trực quan hóa dữ liệu thời tiết
    st.subheader(f"Thời tiết tại {city_name}")
    st.write(city_data)

    # Biểu đồ nhiệt độ theo ngày
    plt.figure(figsize=(10, 5))
    plt.plot(city_data['date'], city_data['temp_c'], marker='o', label="Nhiệt độ (°C)")
    plt.xlabel("Ngày")
    plt.ylabel("Nhiệt độ (°C)")
    plt.title(f"Nhiệt độ tại {city_name}")
    plt.legend()
    st.pyplot(plt)

```

```

# Dự báo thời tiết
st.subheader(f"Dự báo thời tiết 7 ngày tới tại {city_name}")
forecast_data = forecast_with_model(model, city_data, scaler)
st.write(forecast_data)

# Biểu đồ dự báo
plt.figure(figsize=(10, 5))
plt.plot(forecast_data['date'], forecast_data['temp_c'], marker='o', linestyle='--', color='orange', 1
plt.xlabel("Ngày")
plt.ylabel("Nhiệt độ (°C)")
plt.title(f"Dự báo nhiệt độ tại {city_name}")
plt.legend()
st.pyplot(plt)

# Thông tin tổng quan
avg_temp = city_data['temp_c'].mean()
total_rainfall = city_data['rainfall_mm'].sum()
st.write(f"Nhiệt độ trung bình: {avg_temp:.2f}°C")
st.write(f"Tổng lượng mưa: {total_rainfall:.2f} mm")

```

+ Bước 2: Sử dụng thư viện streamlit để tải dữ liệu và mô hình.

```

# Tải dữ liệu từ file CSV
uploaded_file = st.file_uploader("Tải lên file CSV thời tiết", type="csv")
model_path = st.text_input("Nhập đường dẫn tới mô hình học sâu đã lưu (.h5):")

if uploaded_file and model_path:
    # Đọc dữ liệu
    data = load_and_process_data(uploaded_file)

    # Chuẩn hóa dữ liệu
    data_scaled, scaler = normalize_data(data)

    # Tạo chuỗi dữ liệu cho train, validation, và test
    sequence_length = 30 # Số ngày để tạo chuỗi
    X_data, y_data = create_sequences(data_scaled, data['temp_c'].values, sequence_length)

    # Tải mô hình
    try:
        model = load_model(model_path)
        st.success("Mô hình học sâu đã được tải thành công!")
    except Exception as e:
        st.error(f"Lỗi khi tải mô hình: {e}")
        st.stop()

```

+ Hiện thị các thông tin dự đoán

```

# Hiện thị các thành phố có trong dữ liệu
cities = data['city'].unique()
city_choice = st.selectbox("Chọn thành phố", cities)


# Hiện thị thông tin thời tiết và dự báo
display_city_weather(data, city_choice, model, scaler)

```

- Các hình ảnh giao diện người dùng

Thông tin thời tiết và dự báo

Tải lên file CSV thời tiết



Drag and drop file here
Limit 200MB per file • CSV

Browse files

 weather_data_13_12_2023_to_12_12_2024.csv 43.1KB ✕

Nhập đường dẫn tới mô hình học sâu đã lưu (.h5):

D:\Project\deep-learning\lstm_weather_model.h5

Mô hình học sâu đã được tải thành công!

Chọn thành phố

Hanoi ▼

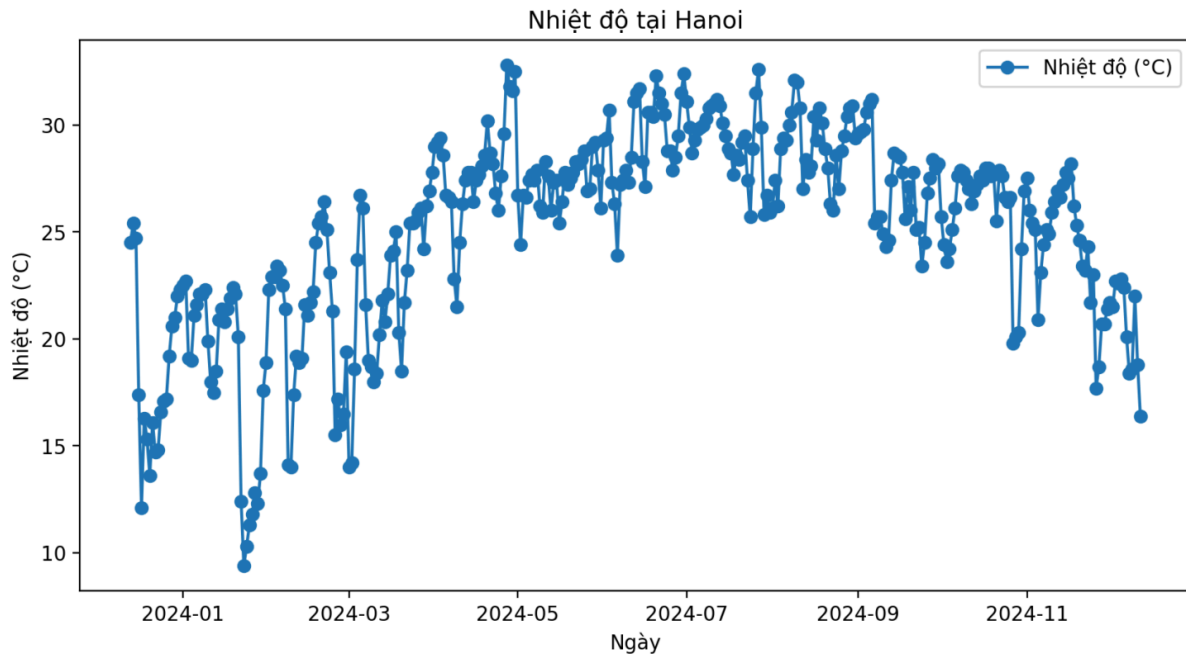
Thời tiết tại Hanoi

⬇

🔍

🗖

	city	date	temp_c	humidity	rainfall_mm	wind_speed_kph
6	Hanoi	2023-12-15 00:00:00	24.7	80	0.16	11.9
9	Hanoi	2023-12-16 00:00:00	17.4	81	10.31	32
12	Hanoi	2023-12-17 00:00:00	12.1	70	0.5	19.8
15	Hanoi	2023-12-18 00:00:00	16.3	66	0.03	10.8
18	Hanoi	2023-12-19 00:00:00	15.3	73	0.37	18.7
21	Hanoi	2023-12-20 00:00:00	13.6	62	0.08	18.7
24	Hanoi	2023-12-21 00:00:00	16.1	45	0	22
27	Hanoi	2023-12-22 00:00:00	14.7	38	0	24.1
30	Hanoi	2023-12-23 00:00:00	14.8	36	0	15.8



Chương VI. Ứng dụng và hướng phát triển của đề tài

- Ứng dụng thực tiễn:
 - Dự báo thời tiết ngắn hạn, hỗ trợ nông nghiệp và du lịch.
 - Cảnh báo sớm về các hiện tượng thời tiết cực đoan.
- Hướng phát triển:
 - Sử dụng mô hình xử lý dữ liệu lớn: Thêm các mô hình để xử lý dữ liệu lớn để thu thập và xử lý một lượng lớn dữ liệu từ nhiều nguồn khác nhau (cảm biến, vệ tinh, mô hình dự báo khí hậu toàn cầu).
 - Ứng dụng trong các hệ thống thông minh: Tích hợp mô hình dự đoán nhiệt độ vào các hệ thống thông minh như smart cities, nông nghiệp thông minh hay quản lý năng lượng, giúp tối ưu hóa quy trình và ra quyết định dựa trên dữ liệu thời tiết dự báo.