پیاده سازی سنسور مجازی دما روی برد Arduino

پروژه پنجم درس مدارهای واسط

استاد درس: دکتر فصحتی

دستیار تدریس راهنما: مهندس توکلی

اعضای تیم:

رایا رضائی

امیرمحمد کوشکی

مهراد میلانلو

فهرست مطالب

شرح پروژه:

ما در این پروژه قصد پیاده‌سازی یک پیش‌بینی کننده دمای باطری برروی برد آردوئینو را داریم. این مدار دمای فعلی محیط، میزان ولتاژ و میزان جریان را به عنوان ورودی گرفته و دمای فعلی باطری را خروجی می‌دهد.

این مدار به ما این قابلیت را می‌دهد که نیاز به پیاده‌سازی سخت افزار اضافه و صرف هزینه برای اندازه‌گیری دمای باطری نباشیم و تنها با سنسورهای موجود برای اندازه‌گیری دمای محیط و داشتن ولتاژ و جریان به دمای فعلی باطری برسیم.

پیاده‌سازی پروژه:

برای train کردن مدل از فایل دیتاستی که در اختیارمان قرار داده شده استفاده می‌کنیم.

	Voltage_measured (Volts)	Current_measured (Amps)	Temperature_measured (C)	Time (secs)	Source_File	Ambient Temperature	Delta_Temperature	state	cycle_number
2	3.873017221	-0.001200661	24.65535783	0	Cycle_1_charge.csv	24	0.655357834	charge	1
3	3.479393559	-4.030268478	24.66647981	2.532	Cycle_1_charge.csv	24	0.666479812	charge	1
4	4.000587822	1.512730647	24.67539447	5.5	Cycle_1_charge.csv	24	0.67539447	charge	1
5	4.012395194	1.509063282	24.69386509	8.344	Cycle_1_charge.csv	24	0.693865094	charge	1
6	4.019708059	1.511318193	24.70506946	11.125	Cycle_1_charge.csv	24	0.70506946	charge	1
7	4.025409467	1.51277913	24.71813972	13.891	Cycle_1_charge.csv	24	0.718139717	charge	1
8	4.030636266	1.511838343	24.73114415	16.672	Cycle_1_charge.csv	24	0.731144147	charge	1
9	4.035348959	1.510245404	24.74128965	19.5	Cycle_1_charge.csv	24	0.741289654	charge	1
10	4.039716367	1.507795762	24.75901141	22.282	Cycle_1_charge.csv	24	0.759011414	charge	1
11	4.04354121	1.507322026	24.76689109	25.063	Cycle_1_charge.csv	24	0.766891095	charge	1
12	4.046724069	1.510225939	24.77833908	27.828	Cycle_1_charge.csv	24	0.778339076	charge	1
13	4.050320832	1.51185336	24.79565341	30.641	Cycle_1_charge.csv	24	0.795653409	charge	1
14	4.053477757	1.510139289	24.79560886	33.453	Cycle_1_charge.csv	24	0.795608863	charge	1
15	4.056879474	1.512817714	24.80759179	36.219	Cycle_1_charge.csv	24	0.807591786	charge	1
16	4.06020401	1.509577661	24.82664625	39.735	Cycle_1_charge.csv	24	0.826646253	charge	1
17	4.06309148	1.512772146	24.83742133	42.578	Cycle_1_charge.csv	24	0.837421333	charge	1
18	4.066063636	1.509769494	24.8467593	45.438	Cycle_1_charge.csv	24	0.846759303	charge	1
19	4.068105681	1.51136777	24.84916666	48.297	Cycle_1_charge.csv	24	0.849166658	charge	1
20	4.070910891	1.513397984	24.85294998	51.188	Cycle_1_charge.csv	24	0.852949982	charge	1
21	4.073140624	1.511197352	24.86507399	54.047	Cycle_1_charge.csv	24	0.865073985	charge	1
22	4.075311977	1.511244173	24.87506247	56.922	Cycle_1_charge.csv	24	0.875062473	charge	1
23	4.077986508	1.508112631	24.88648983	59.797	Cycle_1_charge.csv	24	0.886489825	charge	1
24	4.079760222	1.508852666	24.90653923	62.688	Cycle_1_charge.csv	24	0.906539228	charge	1
25	4.081802115	1.510345307	24.90930298	65.657	Cycle_1_charge.csv	24	0.909302977	charge	1

شکل ۱: تصویری از دیتاست

همان‌طور که در تصویر نیز می‌بینیم دمای محیط ۲۴ درجه سانتی‌گراد بوده و ولتاژ در بازه ی $[3.873,4.2]$ ولت بوده و جریان در بازه $[-0.002,1.51]$ آمپر می‌باشد.

برای پیدا کردن مدل ایده‌آل هم از روش خطی استفاده می‌کنیم و هم شبکه عصبی تا روشی که بهترین پیش‌بینی را دارد را روی برد منتقل کنیم.

در ادامه مراحل پیاده‌سازی و تست عملکرد دو مدل رگرسیون خطی و شبکه عصبی را مورد بررسی قرار می‌دهیم.

مراحل train کردن مدل

ابتدا فایل داده را می‌خوانیم و ۸۰ درصد داده‌ها را به عنوان داده training برای یادگیری مدل و ۲۰ درصد داده‌ها را به عنوان داده testing برای آزمون مدل استفاده می‌کنیم. و همچنین به منظور آموزش صحیح مدل داده ها را نرمالایز می‌کنیم

```
import pandas as pd


file_path = "data.csv"
df = pd.read_csv(file_path)
df.head()
```

	Voltage_measured (Volts)	Current_measured (Amps)	Temperature_measured (C)	Time (secs)	Source_File	Ambient Temperature	Delta_Temperature	state	cycle_number	Unnar
0	3.873017	-0.001201	24.655358	0.000	Cycle_1_charge.csv	24	0.655358	charge	1	
1	3.479394	-4.030268	24.666480	2.532	Cycle_1_charge.csv	24	0.666480	charge	1	
2	4.000588	1.512731	24.675394	5.500	Cycle_1_charge.csv	24	0.675394	charge	1	
3	4.012395	1.509063	24.693865	8.344	Cycle_1_charge.csv	24	0.693865	charge	1	
4	4.019708	1.511318	24.705069	11.125	Cycle_1_charge.csv	24	0.705069	charge	1	

شکل ۲ خواندن فایل دیتاست

```
[ ]
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler


features = ["Voltage_measured (Volts)", "Current_measured (Amps)", "Ambient Temperature"]
target = "Temperature_measured (C)"

X = df[features].values
y = df[target].values


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

شکل ۳ بررسی دیتاست و نرمالایز داده ها

مدل رگرسیون خطی

در ادامه با تابع LinearRegression() مدل خطی خود را می‌سازیم و مقداری که این مدل پیش‌بینی می‌کند را در y_pred_line نگهداری کرده و برای تست عملکرد آن مقدار R و میزان خطای میانگین مربعات آن را اندازه‌گیری کرده و خروجی می‌دهیم.

```python
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

lin_reg = LinearRegression()
lin_reg.fit(X_train, y_train)

y_pred_lin = lin_reg.predict(X_test)


mse_lin = mean_squared_error(y_test, y_pred_lin)
r2_lin = r2_score(y_test, y_pred_lin)

print("Linear Regression Results:")
print(f"Mean Squared Error (MSE): {mse_lin:.4f}")
print(f"R² Score: {r2_lin:.4f}")
```

```
Linear Regression Results:
Mean Squared Error (MSE): 0.3693
R² Score: 0.6186
```

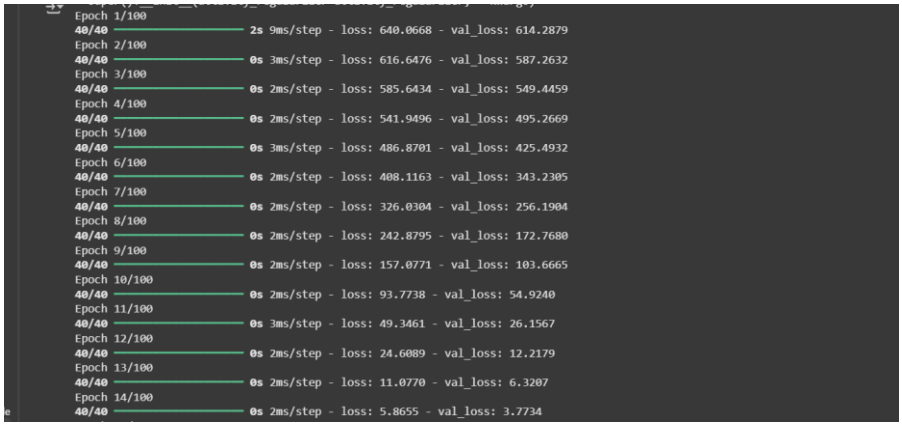
شکل ۴ خروجی رگرسیون خطی

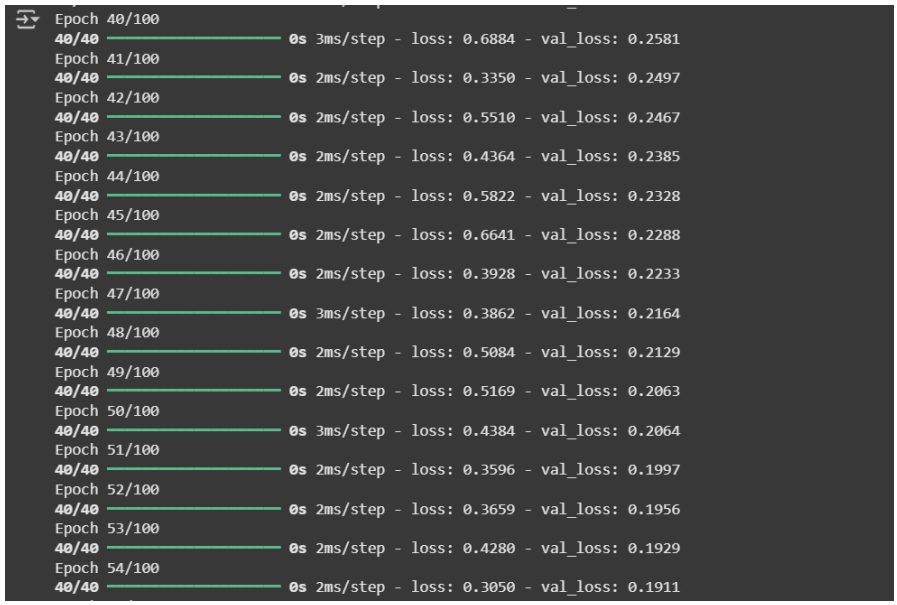
مدل شبکه عصبی

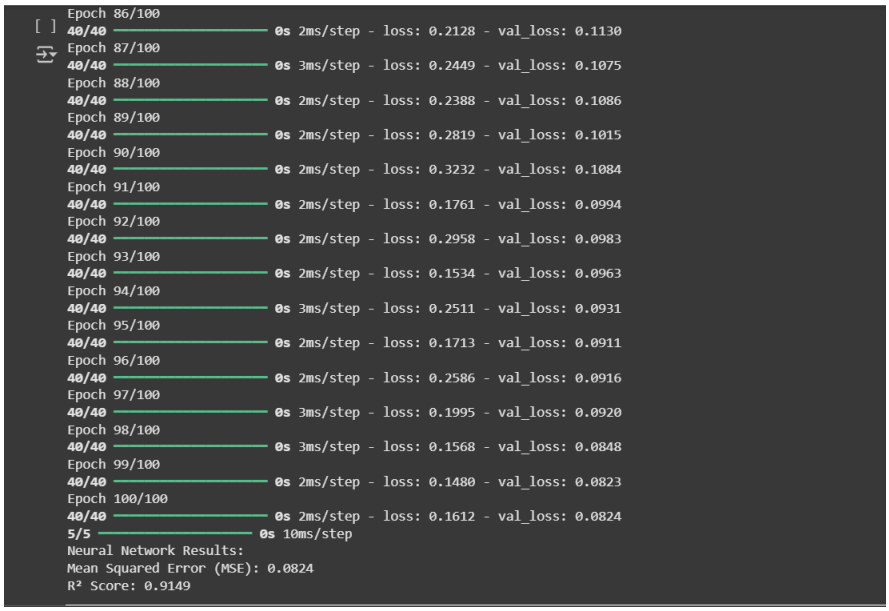
در ادامه با استفاده از کتابخانه tensorflow یک شبکه عصبی با دو input layer و یک hidden layer را پیاده‌سازی کرده و همچنین از relu به عنوان تابع فعال ساز آن استفاده می‌کنیم. و در نهایت میزان خطای میانگین مربعات و R آن را مورد بررسی قرار می‌دهیم.

```python
import tensorflow as tf
from tensorflow import keras

model = keras.Sequential([
    keras.layers.Dense(16, activation='relu', input_shape=(X_train_scaled.shape[1],)),
    keras.layers.Dense(8, activation='relu'),
    keras.layers.Dense(1)
])


model.compile(optimizer='adam', loss='mse')

# Train Model
history = model.fit(X_train_scaled, y_train, epochs=100, batch_size=16, verbose=1, validation_data=(X_test_scaled, y_test))


y_pred_nn = model.predict(X_test_scaled).flatten()

mse_nn = mean_squared_error(y_test, y_pred_nn)
r2_nn = r2_score(y_test, y_pred_nn)


print("Neural Network Results:")
print(f"Mean Squared Error (MSE): {mse_nn:.4f}")
print(f"R² Score: {r2_nn:.4f}")
```

شکل ۵ پیاده‌سازی شبکه عصبی







شکل ۶ خروجی تست شبکه عصبی

می‌بینیم که هر چه جلوتر می‌رویم میزان loss مدل کمتر شده میزان خطای میانگین مربعات و R آن نیز خروجی داده می‌شود.

مقایسه مدل ها و انتخاب مدل دقیق‌تر

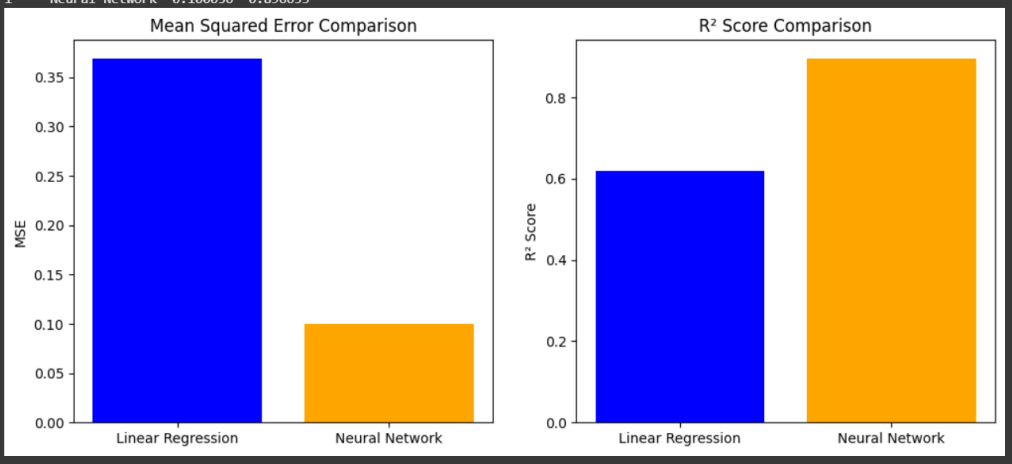
در این بخش با رسم نمودار برای مقدار خطای میانگین مربعات و R هر دو مدل دقت آنها را می‌سنجیم.

```python
import matplotlib.pyplot as plt
import pandas as pd


results = pd.DataFrame({
    "Model": ["Linear Regression", "Neural Network"],
    "MSE": [mse_lin, mse_nn],
    "R² Score": [r2_lin, r2_nn]
})


print(results)


fig, ax = plt.subplots(1, 2, figsize=(12, 5))


ax[0].bar(results["Model"], results["MSE"], color=['blue', 'orange'])
ax[0].set_title("Mean Squared Error Comparison")
ax[0].set_ylabel("MSE")


ax[1].bar(results["Model"], results["R² Score"], color=['blue', 'orange'])
ax[1].set_title("R² Score Comparison")
ax[1].set_ylabel("R² Score")

plt.show()
```

شکل ۷ مقایسه دقت دو مدل

در ادامه می‌بینیم که چون شبکه عصبی مقدار خطای میانگین مربعات کمتری داشته و R آن نیز به یک نزدیک تر است مدل دقیق‌تری است و آن را برای پیاده‌سازی رو برد انتخاب خواهیم کرد.

```
              Model       MSE  R² Score
0  Linear Regression  0.369266  0.618587
1     Neural Network  0.100056  0.896653
```



شکل ۸ نمودار مقایسه دو مدل

لازم است با کد زیر مقدارهای w و b آن را خروجی بگیریم.

```python
# Extract model weights and biases
weights = model.get_weights()

# Save them into a dictionary
model_weights = {
    "W1": weights[0].tolist(),
    "b1": weights[1].tolist(),
    "W2": weights[2].tolist(),
    "b2": weights[3].tolist(),
    "W3": weights[4].tolist(),
    "b3": weights[5].tolist()
}

# Extract scaler parameters
scaler_params = {
    "mean": scaler.mean_.tolist(),
    "std": scaler.scale_.tolist()
}

import json
with open("model_weights.json", "w") as f:
    json.dump(model_weights, f)

with open("scaler_params.json", "w") as f:
    json.dump(scaler_params, f)

print("Weights and scaler parameters saved!")
```

شکل ۹ خروجی گرفتن مقادیر پارامترهای شبکه عصبی

پیاده‌سازی روی برد

بعد از setup کردن و شناسایی برد ابتدا شبکه عصبی را براساس شبکه عصبی پیاده‌سازی شده در مرحله قبل پیاده کرده و در ادامه برای دادن داده به برد برای هر یک پارامتر ها یک عدد float در بازه اعداد موجود در dataset اولیه ساخته و به برد می‌دهیم و برد به ما دمای پیش‌بینی شده باطری را می‌دهد.

```
float W1[3][6] = {{-1.3599201440811157, 0.4730043113231659, 0.4874628782272339, 0.34043213725090027, 0.7712751626968384, 0.5910599827766418}, {2.1320865154266357, -0.26845452189445496,
float b1[6] = {2.945068359375, 2.1410326957702637, -0.26389631628990173, 2.271608352661133, 2.4415271282196045, 1.943440556526184};
float W2[6] = {2.1176624298095703, 2.2715976238250732, -0.41542890667915344, 1.6780105829238892, 2.148237466812134, 1.8572229146957397};
float b2 = 1.391572117805481;

float scaler_mean[3] = {4.187419504017744, 0.6434546691812421, 24.0};
float scaler_std[3] = {0.04576498913931391, 0.6254070302957698, 1.0};

float relu(float x) {
    return x > 0 ? x : 0;
}

float predict(float voltage, float current, float ambient_temp) {
    float input[3] = {voltage, current, ambient_temp};

    for (int i = 0; i < 3; i++) {
        input[i] = (input[i] - scaler_mean[i]) / scaler_std[i];
    }

    float hidden[6];
    for (int i = 0; i < 6; i++) {
        hidden[i] = b1[i];
        for (int j = 0; j < 3; j++) {
            hidden[i] += input[j] * W1[j][i];
        }
        hidden[i] = relu(hidden[i]);
    }

    float output = b2;
    for (int i = 0; i < 6; i++) {
        output += hidden[i] * W2[i];
    }
```

شکل ۱۰ پیاده‌سازی شبکه عصبی

```
float voltage, current, ambient_temp;

float randomFloat(float minVal, float maxVal) {
    return minVal + ((float)rand() / RAND_MAX) * (maxVal - minVal);
}

void updateValues() {
    voltage = randomFloat(3.873, 4.2);
    current = randomFloat(-0.001, 1.51);
    ambient_temp = randomFloat(24, 24);
}
```

شکل ۱۱ ایجاد عدد رندوم برای ورودی دادن به مدل

```
void loop() {
    unsigned long currentMillis = millis();
    if (currentMillis - previousMillis >= updateInterval) {
        previousMillis = currentMillis;

        updateValues();
        printValues();
        float predicted_temp = predict(voltage, current, ambient_temp);

    Serial.print("Predicted Temperature: ");
    Serial.println(predicted_temp, 6);
    }
}
```

شکل ۱۲ حلقه اصلی کد

مقدار دمای واقعی در دیتاست مقداری ۲۴ تا ۲۷ درجه سانتی گراد است و پیش‌بینی‌ها در برخی موارد خطا داشته اما به طور کلی در این بازه هستند.

```
Voltage: 4.14 | Current: 0.86 | Temperature: 24.00
Predicted Temperature: 24.030370
Voltage: 4.14 | Current: 0.95 | Temperature: 24.00
Predicted Temperature: 24.161140
Voltage: 4.20 | Current: 0.97 | Temperature: 24.00
Predicted Temperature: 26.194355
Voltage: 4.10 | Current: 0.92 | Temperature: 24.00
Predicted Temperature: 22.943763
Voltage: 4.12 | Current: 0.10 | Temperature: 24.00
Predicted Temperature: 21.612758
Voltage: 3.88 | Current: 0.77 | Temperature: 24.00
```

شکل ۱۳ خروجی کد