

stress-lstm-project

December 8, 2025

```
[2]: !pip install kagglehub
```

```
Requirement already satisfied: kagglehub in /opt/anaconda3/lib/python3.12/site-packages (0.3.13)
Requirement already satisfied: packaging in /opt/anaconda3/lib/python3.12/site-packages (from kagglehub) (24.1)
Requirement already satisfied: pyyaml in /opt/anaconda3/lib/python3.12/site-packages (from kagglehub) (6.0.1)
Requirement already satisfied: requests in /opt/anaconda3/lib/python3.12/site-packages (from kagglehub) (2.32.3)
Requirement already satisfied: tqdm in /opt/anaconda3/lib/python3.12/site-packages (from kagglehub) (4.66.5)
Requirement already satisfied: charset-normalizer<4,>=2 in /opt/anaconda3/lib/python3.12/site-packages (from requests->kagglehub) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /opt/anaconda3/lib/python3.12/site-packages (from requests->kagglehub) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in /opt/anaconda3/lib/python3.12/site-packages (from requests->kagglehub) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in /opt/anaconda3/lib/python3.12/site-packages (from requests->kagglehub) (2025.11.12)
```

```
[3]: import numpy as np
import pandas as pd
import torch
import torch.nn as nn
from torch.utils.data import Dataset, DataLoader
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report

import matplotlib.pyplot as plt

SEED = 42
np.random.seed(SEED)
torch.manual_seed(SEED)
```

```
[3]: <torch._C.Generator at 0x11ae7b050>
```

```
[4]: import kagglehub

# Download latest version
path = kagglehub.dataset_download("wafaaelhusseini/
↳worklife-balance-synthetic-daily-wellness-dataset")

print("Path to dataset files:", path)
```

Path to dataset files:
/Users/makayla/.cache/kagglehub/datasets/wafaaelhusseini/worklife-balance-synthetic-daily-wellness-dataset/versions/1

```
[5]: import os

print("Dataset folder:", path)
print("Contents:", os.listdir(path))
```

Dataset folder:
/Users/makayla/.cache/kagglehub/datasets/wafaaelhusseini/worklife-balance-synthetic-daily-wellness-dataset/versions/1
Contents: ['weekly_summaries.csv', 'users.csv', 'daily_all.csv',
'interventions.csv', 'daily_logs.csv']

```
[6]: csv_name = "Stress Level Detection Based on Daily Activities.csv" # <---↳
↳change this
csv_path = os.path.join(path, csv_name)
print("CSV path:", csv_path)
```

CSV path: /Users/makayla/.cache/kagglehub/datasets/wafaaelhusseini/worklife-balance-synthetic-daily-wellness-dataset/versions/1/Stress Level Detection Based on Daily Activities.csv

```
[7]: import pandas as pd
import os

for name in ["daily_all.csv", "daily_logs.csv"]:
    print("\n---", name, "---")
    df_temp = pd.read_csv(os.path.join(path, name))
    print(df_temp.head())
    print(df_temp.info())
```

```
--- daily_all.csv ---
  user_id      date  week_start  workday  profession  work_mode  chronotype \
0        1  2024-01-01  2024-01-01    True  operations    onsite    morning
1        1  2024-01-02  2024-01-01    True  operations    onsite    morning
2        1  2024-01-03  2024-01-01    True  operations    onsite    morning
3        1  2024-01-04  2024-01-01    True  operations    onsite    morning
4        1  2024-01-05  2024-01-01    True  operations    onsite    morning
```

	age	sex	height_cm	...	workouts_count	cheat_meals_count	\
0	27	female	174	...	10	1	
1	27	female	174	...	10	1	
2	27	female	174	...	10	1	
3	27	female	174	...	10	1	
4	27	female	174	...	10	1	

	has_intervention	intervention_diet_coaching	intervention_exercise_plan	\
0	False	False	False	
1	False	False	False	
2	False	False	False	
3	False	False	False	
4	False	False	False	

	intervention_meditation	intervention_sick_leave	intervention_therapy	\
0	False	False	False	
1	False	False	False	
2	False	False	False	
3	False	False	False	
4	False	False	False	

	intervention_vacation	intervention_workload_cap
0	False	False
1	False	False
2	False	False
3	False	False
4	False	False

[5 rows x 53 columns]

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 731000 entries, 0 to 730999

Data columns (total 53 columns):

#	Column	Non-Null Count	Dtype
---	-----	-----	-----
0	user_id	731000 non-null	int64
1	date	731000 non-null	object
2	week_start	731000 non-null	object
3	workday	731000 non-null	bool
4	profession	731000 non-null	object
5	work_mode	731000 non-null	object
6	chronotype	731000 non-null	object
7	age	731000 non-null	int64
8	sex	731000 non-null	object
9	height_cm	731000 non-null	int64
10	mental_health_history	731000 non-null	object
11	exercise_habit	731000 non-null	object
12	caffeine_sensitivity	731000 non-null	object

13	baseline_bmi	731000	non-null	float64
14	sleep_hours	731000	non-null	float64
15	sleep_quality	731000	non-null	int64
16	work_hours	731000	non-null	float64
17	meetings_count	731000	non-null	int64
18	tasks_completed	731000	non-null	int64
19	emails_received	731000	non-null	int64
20	commute_minutes	731000	non-null	int64
21	exercise_minutes	731000	non-null	int64
22	steps_count	731000	non-null	int64
23	caffeine_mg	731000	non-null	int64
24	alcohol_units	731000	non-null	float64
25	screen_time_hours	731000	non-null	float64
26	social_interactions	731000	non-null	int64
27	outdoor_time_minutes	731000	non-null	int64
28	diet_quality	731000	non-null	int64
29	calories_intake	731000	non-null	int64
30	stress_level	731000	non-null	int64
31	mood_score	731000	non-null	int64
32	energy_level	731000	non-null	int64
33	focus_score	731000	non-null	int64
34	work_pressure	731000	non-null	object
35	weather_mood_impact	731000	non-null	float64
36	weight_kg	731000	non-null	float64
37	job_satisfaction	731000	non-null	int64
38	perceived_stress_scale	731000	non-null	int64
39	anxiety_score	731000	non-null	int64
40	depression_score	731000	non-null	int64
41	sleep_debt_hours	731000	non-null	float64
42	avg_weight_kg_week	731000	non-null	float64
43	workouts_count	731000	non-null	int64
44	cheat_meals_count	731000	non-null	int64
45	has_intervention	731000	non-null	bool
46	intervention_diet_coaching	731000	non-null	bool
47	intervention_exercise_plan	731000	non-null	bool
48	intervention_meditation	731000	non-null	bool
49	intervention_sick_leave	731000	non-null	bool
50	intervention_therapy	731000	non-null	bool
51	intervention_vacation	731000	non-null	bool
52	intervention_workload_cap	731000	non-null	bool

dtypes: bool(9), float64(9), int64(25), object(10)

memory usage: 251.7+ MB

None

--- daily_logs.csv ---

	user_id	date	workday	sleep_hours	sleep_quality	work_hours	\
0	1	2024-01-01	True	7.14	6	6.99	
1	1	2024-01-02	True	7.61	5	8.42	

2	1	2024-01-03	True	8.00	7	7.21
3	1	2024-01-04	True	7.19	5	7.85
4	1	2024-01-05	True	7.95	8	7.17

	meetings_count	tasks_completed	emails_received	commute_minutes	...	\
0	3	6	51	16	...	
1	6	0	50	36	...	
2	4	10	57	35	...	
3	4	0	38	33	...	
4	1	7	38	30	...	

	outdoor_time_minutes	diet_quality	calories_intake	stress_level	\
0	7	4	2125	4	
1	23	6	2465	6	
2	22	3	2235	3	
3	34	5	2433	5	
4	24	7	1852	3	

	mood_score	energy_level	focus_score	work_pressure	weather_mood_impact	\
0	3	3	7	low	-0.5	
1	5	6	2	low	-0.5	
2	6	7	7	low	-0.5	
3	4	6	6	low	-0.5	
4	8	5	5	low	-0.5	

	weight_kg
0	73.25
1	73.23
2	73.20
3	73.18
4	73.09

[5 rows x 26 columns]

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 731000 entries, 0 to 730999

Data columns (total 26 columns):

#	Column	Non-Null Count	Dtype
0	user_id	731000 non-null	int64
1	date	731000 non-null	object
2	workday	731000 non-null	bool
3	sleep_hours	731000 non-null	float64
4	sleep_quality	731000 non-null	int64
5	work_hours	731000 non-null	float64
6	meetings_count	731000 non-null	int64
7	tasks_completed	731000 non-null	int64
8	emails_received	731000 non-null	int64
9	commute_minutes	731000 non-null	int64

```

10 exercise_minutes      731000 non-null int64
11 steps_count           731000 non-null int64
12 caffeine_mg           731000 non-null int64
13 alcohol_units         731000 non-null float64
14 screen_time_hours     731000 non-null float64
15 social_interactions    731000 non-null int64
16 outdoor_time_minutes  731000 non-null int64
17 diet_quality           731000 non-null int64
18 calories_intake        731000 non-null int64
19 stress_level           731000 non-null int64
20 mood_score             731000 non-null int64
21 energy_level           731000 non-null int64
22 focus_score            731000 non-null int64
23 work_pressure          731000 non-null object
24 weather_mood_impact    731000 non-null float64
25 weight_kg              731000 non-null float64
dtypes: bool(1), float64(6), int64(17), object(2)
memory usage: 140.1+ MB
None

```

```
[8]: df_all = pd.read_csv(os.path.join(path, "daily_all.csv"))
df_all.columns
```

```
[8]: Index(['user_id', 'date', 'week_start', 'workday', 'profession', 'work_mode',
'chronotype', 'age', 'sex', 'height_cm', 'mental_health_history',
'exercise_habit', 'caffeine_sensitivity', 'baseline_bmi', 'sleep_hours',
'sleep_quality', 'work_hours', 'meetings_count', 'tasks_completed',
'emails_received', 'commute_minutes', 'exercise_minutes', 'steps_count',
'caffeine_mg', 'alcohol_units', 'screen_time_hours',
'social_interactions', 'outdoor_time_minutes', 'diet_quality',
'calories_intake', 'stress_level', 'mood_score', 'energy_level',
'focus_score', 'work_pressure', 'weather_mood_impact', 'weight_kg',
'job_satisfaction', 'perceived_stress_scale', 'anxiety_score',
'depression_score', 'sleep_debt_hours', 'avg_weight_kg_week',
'workouts_count', 'cheat_meals_count', 'has_intervention',
'intervention_diet_coaching', 'intervention_exercise_plan',
'intervention_meditation', 'intervention_sick_leave',
'intervention_therapy', 'intervention_vacation',
'intervention_workload_cap'],
dtype='object')
```

```
[9]: #load and basic cleaning

import pandas as pd
import numpy as np

df = pd.read_csv(os.path.join(path, "daily_all.csv"))
```

```

# Convert date to real datetime
df['date'] = pd.to_datetime(df['date'])

# Sort by user then date
df = df.sort_values(['user_id', 'date']).reset_index(drop=True)

df.head()
df.info()

```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 731000 entries, 0 to 730999
Data columns (total 53 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   user_id                               731000 non-null  int64
1   date                                  731000 non-null  datetime64[ns]
2   week_start                            731000 non-null  object
3   weekday                               731000 non-null  bool
4   profession                            731000 non-null  object
5   work_mode                             731000 non-null  object
6   chronotype                            731000 non-null  object
7   age                                    731000 non-null  int64
8   sex                                    731000 non-null  object
9   height_cm                             731000 non-null  int64
10  mental_health_history                 731000 non-null  object
11  exercise_habit                        731000 non-null  object
12  caffeine_sensitivity                  731000 non-null  object
13  baseline_bmi                         731000 non-null  float64
14  sleep_hours                          731000 non-null  float64
15  sleep_quality                         731000 non-null  int64
16  work_hours                           731000 non-null  float64
17  meetings_count                       731000 non-null  int64
18  tasks_completed                      731000 non-null  int64
19  emails_received                      731000 non-null  int64
20  commute_minutes                     731000 non-null  int64
21  exercise_minutes                    731000 non-null  int64
22  steps_count                         731000 non-null  int64
23  caffeine_mg                         731000 non-null  int64
24  alcohol_units                       731000 non-null  float64
25  screen_time_hours                   731000 non-null  float64
26  social_interactions                  731000 non-null  int64
27  outdoor_time_minutes                731000 non-null  int64
28  diet_quality                        731000 non-null  int64
29  calories_intake                     731000 non-null  int64
30  stress_level                        731000 non-null  int64
31  mood_score                          731000 non-null  int64
32  energy_level                        731000 non-null  int64
33  focus_score                         731000 non-null  int64

```

```

34 work_pressure          731000 non-null object
35 weather_mood_impact    731000 non-null float64
36 weight_kg              731000 non-null float64
37 job_satisfaction       731000 non-null int64
38 perceived_stress_scale 731000 non-null int64
39 anxiety_score          731000 non-null int64
40 depression_score       731000 non-null int64
41 sleep_debt_hours       731000 non-null float64
42 avg_weight_kg_week     731000 non-null float64
43 workouts_count         731000 non-null int64
44 cheat_meals_count      731000 non-null int64
45 has_intervention       731000 non-null bool
46 intervention_diet_coaching 731000 non-null bool
47 intervention_exercise_plan 731000 non-null bool
48 intervention_meditation 731000 non-null bool
49 intervention_sick_leave 731000 non-null bool
50 intervention_therapy    731000 non-null bool
51 intervention_vacation   731000 non-null bool
52 intervention_workload_cap 731000 non-null bool
dtypes: bool(9), datetime64[ns](1), float64(9), int64(25), object(9)
memory usage: 251.7+ MB

```

```

[10]: target_col = "stress_level"
      id_col = "user_id"
      time_col = "date"

      drop_cols = [target_col, id_col, time_col]

      feature_cols = [c for c in df.columns if c not in drop_cols]
      feature_cols[:10], len(feature_cols)

```

```

[10]: ('week_start',
      'workday',
      'profession',
      'work_mode',
      'chronotype',
      'age',
      'sex',
      'height_cm',
      'mental_health_history',
      'exercise_habit'],
50)

```

```

[11]: cat_cols = df[feature_cols].select_dtypes(include=['object', 'bool']).columns.
      >tolist()
      num_cols = [c for c in feature_cols if c not in cat_cols]

```

```
cat_cols, len(cat_cols)
num_cols[:10]
```

```
[11]: ['age',
       'height_cm',
       'baseline_bmi',
       'sleep_hours',
       'sleep_quality',
       'work_hours',
       'meetings_count',
       'tasks_completed',
       'emails_received',
       'commute_minutes']
```

```
[50]: from sklearn.preprocessing import StandardScaler

       # One-hot encode categoricals
       df_cat = pd.get_dummies(df[cat_cols], drop_first=True)

       # Scale numerical columns
       scaler = StandardScaler()
       df_num = pd.DataFrame(
           scaler.fit_transform(df[num_cols]),
           columns=num_cols,
           index=df.index
       )

       # Combine processed features
       X_df = pd.concat([df_num, df_cat], axis=1)

       y = df[target_col].values
```

```
[51]: # Ensure no object dtypes remain
       print("Object columns:", X_df.select_dtypes(include='object').columns.tolist())

       # Convert booleans to integers
       bool_cols = X_df.select_dtypes(include='bool').columns
       X_df[bool_cols] = X_df[bool_cols].astype(int)

       # Convert everything to float32
       X_df = X_df.astype('float32')

       print("Final dtypes:", X_df.dtypes.unique())
```

```
Object columns: []
Final dtypes: [dtype('float32')]
```

```
[52]: df_all['stress_level'].describe()
df_all['stress_level'].unique()
```

```
[52]: array([4, 6, 3, 5, 2, 7, 1, 8, 9])
```

```
[53]: def bin_stress(x):
    if x <= 3:
        return 0    # low
    elif x <= 6:
        return 1    # medium
    else:
        return 2    # high

df['stress_class'] = df['stress_level'].apply(bin_stress)
df['stress_class'].unique()
```

```
[53]: array([1, 0, 2])
```

```
[54]: def build_sequences(df, X_df, seq_len, id_col='user_id',
    ↪target_col='stress_class'):

    X_list = []
    y_list = []

    # Group by user
    for user_id, group in df.groupby(id_col):
        group = group.sort_values('date')

        X_user = X_df.loc[group.index].values    # features
        y_user = group[target_col].values        # stress_class target

        T = len(group)

        # Create sliding windows
        for i in range(T - seq_len):
            X_list.append(X_user[i : i + seq_len])
            y_list.append(y_user[i + seq_len])    # next day's stress class

    X_seq = np.array(X_list)
    y_seq = np.array(y_list)

    return X_seq, y_seq
```

```
[55]: X5, y5 = build_sequences(df, X_df, seq_len=5)

X5.shape, y5.shape
```

```
[55]: ((726000, 5, 166), (726000,))
```

```
[56]: from sklearn.model_selection import train_test_split

seq_len = 5
X_seq, y_seq = build_sequences(df, X_df, seq_len)

X_train, X_temp, y_train, y_temp = train_test_split(
    X_seq, y_seq, test_size=0.3, random_state=42, stratify=y_seq
)
X_val, X_test, y_val, y_test = train_test_split(
    X_temp, y_temp, test_size=0.5, random_state=42, stratify=y_temp
)
```

```
[57]: import torch
from torch.utils.data import Dataset, DataLoader

class StressSequenceDataset(Dataset):
    def __init__(self, X, y):
        self.X = torch.tensor(X, dtype=torch.float32)
        self.y = torch.tensor(y, dtype=torch.long)

    def __len__(self):
        return len(self.X)

    def __getitem__(self, idx):
        return self.X[idx], self.y[idx]
```

```
[58]: #Feed-Forward Network BASELINE
```

```
[59]: import torch.nn as nn

class FFN(nn.Module):
    def __init__(self, seq_len, num_features, num_classes=3):
        super().__init__()

        input_dim = seq_len * num_features

        self.net = nn.Sequential(
            nn.Linear(input_dim, 128),
            nn.ReLU(),
            nn.Linear(128, 64),
            nn.ReLU(),
            nn.Linear(64, num_classes)
        )

    def forward(self, x):
```

```

    # x shape: (batch, seq_len, num_features)
    x = x.reshape(x.size(0), -1) # flatten
    return self.net(x)

```

```

[60]: #Build the LSTM model
class LSTMModel(nn.Module):
    def __init__(self, num_features, hidden_size=64, num_layers=1,
    ↪ num_classes=3):
        super().__init__()

        self.lstm = nn.LSTM(
            input_size=num_features,
            hidden_size=hidden_size,
            num_layers=num_layers,
            batch_first=True
        )

        self.fc = nn.Linear(hidden_size, num_classes)

    def forward(self, x):
        out, (h_n, c_n) = self.lstm(x)

        last_hidden = h_n[-1]

        return self.fc(last_hidden)

```

```

[61]: def accuracy(preds, labels):
    return (preds.argmax(dim=1) == labels).float().mean().item()

```

```

[62]: def train_one_epoch(model, loader, optimizer, criterion, device):
    model.train()
    total_loss, total_acc = 0, 0

    for X, y in loader:
        X, y = X.to(device), y.to(device)

        optimizer.zero_grad()
        out = model(X)
        loss = criterion(out, y)
        loss.backward()
        optimizer.step()

        total_loss += loss.item()
        total_acc += accuracy(out, y)

    return total_loss / len(loader), total_acc / len(loader)

```

```
[63]: def eval_model(model, loader, criterion, device):
    model.eval()
    total_loss, total_acc = 0, 0

    with torch.no_grad():
        for X, y in loader:
            X, y = X.to(device), y.to(device)

            out = model(X)
            loss = criterion(out, y)

            total_loss += loss.item()
            total_acc += accuracy(out, y)

    return total_loss / len(loader), total_acc / len(loader)
```

```
[64]: seq_len = 5
X_seq, y_seq = build_sequences(df, X_df, seq_len)

# Split
X_train, X_temp, y_train, y_temp = train_test_split(
    X_seq, y_seq, test_size=0.3, random_state=42, stratify=y_seq
)
X_val, X_test, y_val, y_test = train_test_split(
    X_temp, y_temp, test_size=0.5, random_state=42, stratify=y_temp
)

# Dataloaders
train_ds = StressSequenceDataset(X_train, y_train)
val_ds = StressSequenceDataset(X_val, y_val)
test_ds = StressSequenceDataset(X_test, y_test)

train_loader = DataLoader(train_ds, batch_size=64, shuffle=True)
val_loader = DataLoader(val_ds, batch_size=64)
test_loader = DataLoader(test_ds, batch_size=64)

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
num_features = X_seq.shape[2]
```

```
[65]: ffn = FFN(seq_len, num_features).to(device)
optimizer = torch.optim.Adam(ffn.parameters(), lr=1e-3)
criterion = nn.CrossEntropyLoss()

for epoch in range(10):
    train_loss, train_acc = train_one_epoch(ffn, train_loader, optimizer,
    ↪criterion, device)
    val_loss, val_acc = eval_model(ffn, val_loader, criterion, device)
```

```
print(f"Epoch {epoch+1}: train_acc={train_acc:.3f}, val_acc={val_acc:.3f}")
```

```
Epoch 1: train_acc=0.692, val_acc=0.693
Epoch 2: train_acc=0.697, val_acc=0.696
Epoch 3: train_acc=0.698, val_acc=0.698
Epoch 4: train_acc=0.700, val_acc=0.698
Epoch 5: train_acc=0.701, val_acc=0.698
Epoch 6: train_acc=0.703, val_acc=0.693
Epoch 7: train_acc=0.704, val_acc=0.696
Epoch 8: train_acc=0.705, val_acc=0.696
Epoch 9: train_acc=0.707, val_acc=0.694
Epoch 10: train_acc=0.708, val_acc=0.694
```

```
[66]: lstm = LSTMModel(num_features).to(device)
optimizer = torch.optim.Adam(lstm.parameters(), lr=1e-3)
criterion = nn.CrossEntropyLoss()

for epoch in range(10):
    train_loss, train_acc = train_one_epoch(lstm, train_loader, optimizer,
    ↪criterion, device)
    val_loss, val_acc = eval_model(lstm, val_loader, criterion, device)

    print(f"Epoch {epoch+1}: train_acc={train_acc:.3f}, val_acc={val_acc:.3f}")
```

```
Epoch 1: train_acc=0.695, val_acc=0.696
Epoch 2: train_acc=0.699, val_acc=0.699
Epoch 3: train_acc=0.701, val_acc=0.696
Epoch 4: train_acc=0.702, val_acc=0.698
Epoch 5: train_acc=0.703, val_acc=0.697
Epoch 6: train_acc=0.705, val_acc=0.697
Epoch 7: train_acc=0.707, val_acc=0.693
Epoch 8: train_acc=0.708, val_acc=0.694
Epoch 9: train_acc=0.710, val_acc=0.693
Epoch 10: train_acc=0.712, val_acc=0.689
```

```
[68]: sequence_lengths = [3, 5, 7, 10, 14]
```

```
[69]: results_ffn = {}
results_lstm = {}

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
criterion = nn.CrossEntropyLoss()

for seq_len in sequence_lengths:
    print(f"\n===== Testing sequence length {seq_len} =====")

    # Build sequences
```

```

X_seq, y_seq = build_sequences(df, X_df, seq_len)
print("Sequence data shape:", X_seq.shape)

# Train/val/test split
X_train, X_temp, y_train, y_temp = train_test_split(
    X_seq, y_seq, test_size=0.3, random_state=42, stratify=y_seq
)
X_val, X_test, y_val, y_test = train_test_split(
    X_temp, y_temp, test_size=0.5, random_state=42, stratify=y_temp
)

# Build Datasets
train_ds = StressSequenceDataset(X_train, y_train)
val_ds   = StressSequenceDataset(X_val, y_val)
test_ds  = StressSequenceDataset(X_test, y_test)

train_loader = DataLoader(train_ds, batch_size=64, shuffle=True)
val_loader   = DataLoader(val_ds, batch_size=64)
test_loader  = DataLoader(test_ds, batch_size=64)

num_features = X_seq.shape[2]

# -----
# Train FFN
# -----
ffn = FFN(seq_len, num_features).to(device)
optimizer = torch.optim.Adam(ffn.parameters(), lr=1e-3)

for epoch in range(8): # shorter training since repeated
    train_loss, train_acc = train_one_epoch(ffn, train_loader, optimizer,
↪criterion, device)
    val_loss, val_acc = eval_model(ffn, val_loader, criterion, device)

    results_ffn[seq_len] = val_acc
    print(f"FFN val accuracy @ len={seq_len}: {val_acc:.4f}")

# -----
# Train LSTM
# -----
lstm = LSTMModel(num_features).to(device)
optimizer = torch.optim.Adam(lstm.parameters(), lr=1e-3)

for epoch in range(8):
    train_loss, train_acc = train_one_epoch(lstm, train_loader, optimizer,
↪criterion, device)
    val_loss, val_acc = eval_model(lstm, val_loader, criterion, device)

```

```

results_lstm[seq_len] = val_acc
print(f"LSTM val accuracy @ len={seq_len}: {val_acc:.4f}")

```

```

===== Testing sequence length 3 =====
Sequence data shape: (728000, 3, 166)
FFN val accuracy @ len=3: 0.6919
LSTM val accuracy @ len=3: 0.6884

```

```

===== Testing sequence length 5 =====
Sequence data shape: (726000, 5, 166)
FFN val accuracy @ len=5: 0.6910
LSTM val accuracy @ len=5: 0.6881

```

```

===== Testing sequence length 7 =====
Sequence data shape: (724000, 7, 166)
FFN val accuracy @ len=7: 0.6945
LSTM val accuracy @ len=7: 0.6954

```

```

===== Testing sequence length 10 =====
Sequence data shape: (721000, 10, 166)
FFN val accuracy @ len=10: 0.6949
LSTM val accuracy @ len=10: 0.6969

```

```

===== Testing sequence length 14 =====
Sequence data shape: (717000, 14, 166)
FFN val accuracy @ len=14: 0.6930
LSTM val accuracy @ len=14: 0.6982

```

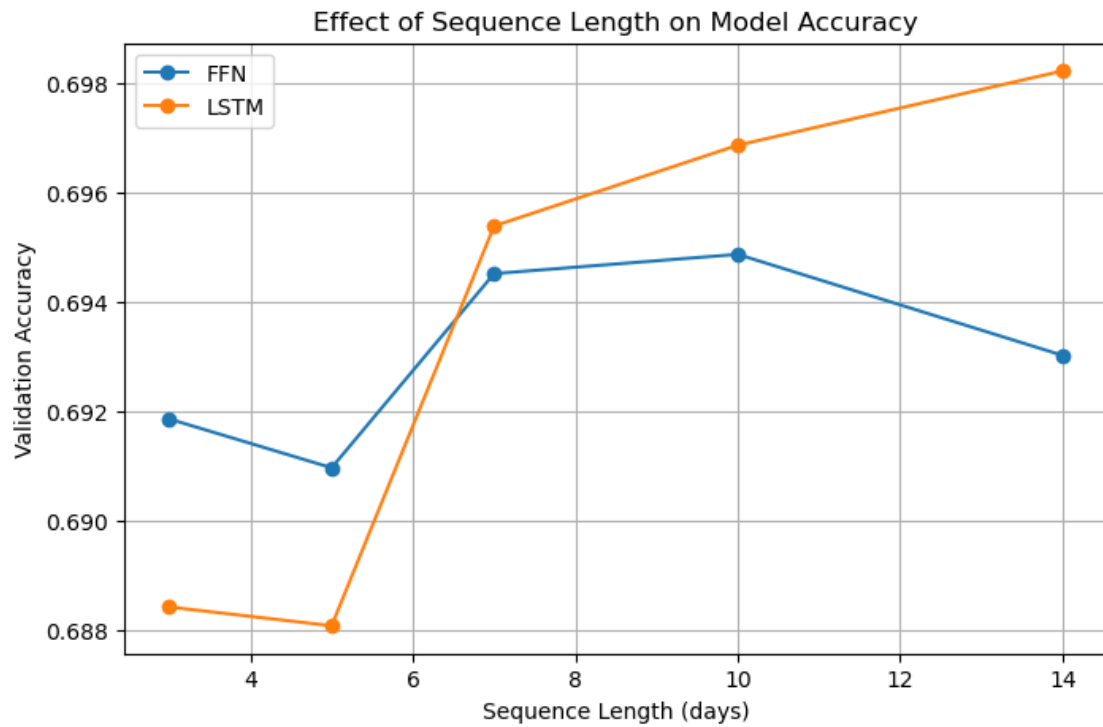
```

[70]: import matplotlib.pyplot as plt

plt.figure(figsize=(8,5))
plt.plot(sequence_lengths, [results_ffn[L] for L in sequence_lengths],
        ↪marker='o', label='FFN')
plt.plot(sequence_lengths, [results_lstm[L] for L in sequence_lengths],
        ↪marker='o', label='LSTM')

plt.xlabel("Sequence Length (days)")
plt.ylabel("Validation Accuracy")
plt.title("Effect of Sequence Length on Model Accuracy")
plt.legend()
plt.grid(True)
plt.show()

```



- []:
- #1. LSTM accuracy increases steadily as sequence length grows
 - #2. FFN accuracy is flatter and does NOT improve with longer sequences
 - #3. LSTM surpasses FFN once sequence length 7