Air Guitar: Machine Learning and Arduino based Virtual Guitar Playing Experience

Bingan Chen[1] Shiyu Chen[1] Samantha Reksosamudra[1]

[1]Dept. of Electrical and Computer Engineering
University of Washington

EE 400: TinyML, Final Project Presentation, May 2024

Introduction

- We are creating a virtual guitar playing experience using an Arduino and machine learning.
- Dataset: Data is collected by ourselves (without a guitar in hand).
- Team member and there roles:

Name	Role
Shiyu Chen	Data Collection; ML Models; Demo
Bingan Chen	Hardware; Algorithms; Slides; ~~Aesthetic~~
Samantha Reksosamudra	Data Collection; Validation; Materials

Table: Team Members and Roles*

*Most of the work is completed collaboratively, so the roles are not strictly defined.

We used *Edge Impulse* to train our machine learning models.

Figure: Edge Impulse Logo

Data Collection

Shiyu: collected data using a flex sensors.

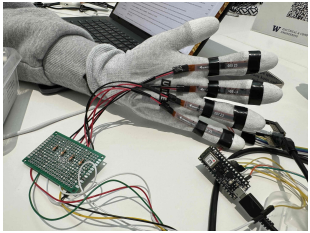

Figure: Flex Sensors

Samantha: collected data using accelerometer.

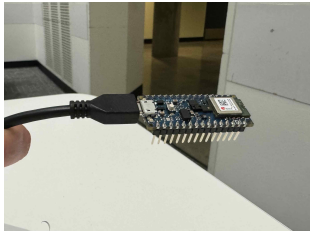

Figure: Accelerometer (built-in)

Full Setup

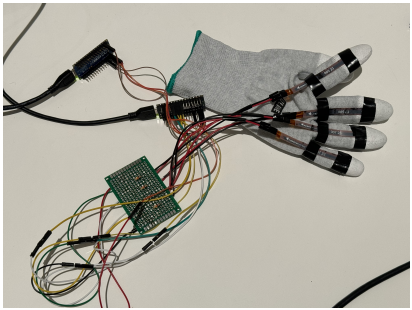

Figure: Full Setup

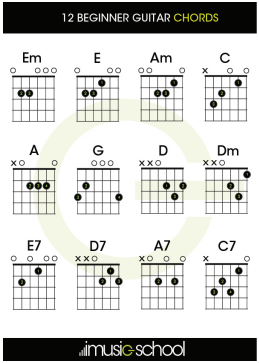

Figure: Chords Gestures

Pipeline

1. Collect data using flex sensors and accelerometer.
2. Train the models using Edge Impulse.
3. Load the models onto **two separate** Arduinos Nano 33 BLE Sense.
4. Communicate between the two Arduinos *Flex & Acc* using BLE using **UART (Universal Asynchronous Receiver-Transmitter)**:
    1. *Acc* predict the strumming.
    2. *Acc* sends the prediction [idle, strum] to *Flex*.
    3. Once *Flex* receives the signal for strumming, it predicts the chord.
    4. *Flex* sends the prediction to the computer.
    5. The computer plays the sound of the chord.
5. Play the guitar by strumming.

Flex Sensors Data

- The flex sensors data is used to determine the notes being played.
- Input: 4 axes of flex sensors data in voltage.
- Labels: [A, C, C7, D, E, Em, G, relax] by file name.
- Output: 8 classes of guitar chords.
- Data Loading:
  Flex sensors →Arduino Analog Read →script monitoring serial →CSV file.
- Architecture: `[400 epochs, learning rate 0.001, batch size 32]`

Input layer (4 features)
Dense layer (20 neurons)
Dense layer (10 neurons)
Dropout (rate 0.5)
Output layer (8 classes)

Accelerometer Data

- The accelerometer data is used to determine whether we want to play the note.
- Input: 3 axes of accelerometer data.
- Labels: [idle, strum]
- Output: 2 classes of strumming.
- Data Loading:
  `edge-impulse-cli`
- Architecture: `[30 epochs, learning rate 0.0005, batch size 32]`

Input layer (39 features)
Dense layer (20 neurons)
Dense layer (10 neurons)
Output layer (2 classes)

Confusion Matrix

We achieved an **testing accuracy** of 91.21% for the flex sensors model and 99.55% for the accelerometer model.

	A	C	C7	D	E	EM	G	RELAX	UNCER
A	100%	0%	0%	0%	0%	0%	0%	0%	0%
C	0%	82.9%	0%	0%	7.1%	0%	0%	0%	10%
C7	0%	0%	91.2%	0%	0%	0%	0%	0%	8.8%
D	0%	0%	0%	85.7%	0%	0%	0%	0%	14.3%
E	0%	3.4%	0%	0%	79.5%	0%	0%	0%	17.0%
EM	0%	0%	0%	0%	0%	92.7%	6.1%	0%	1.2%
G	0%	0%	0%	0%	0%	0%	99.0%	0%	1.0%
RELAX	0%	0%	0%	0%	0%	0%	0%	100%	0%
F1 SC	1.00	0.89	0.95	0.92	0.86	0.96	0.97	1.00	

Figure: Flex Sensors Confusion Matrix

	IDLE	STRUM	UNCERTAIN
IDLE	100%	0%	0%
STRUM	0%	97.7%	2.3%
F1 SCORE	1.00	0.99	

Figure: Accelerometer Confusion Matrix

According to the confusion matrix, notes C, D, E, are a little bit harder to predict.

Features Explorer

They are clustered well in the feature space.

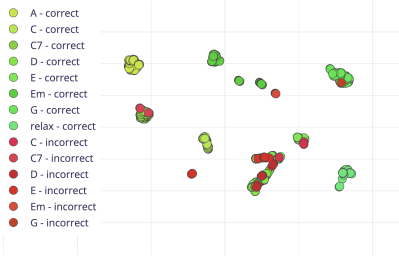

Figure: Flex Sensors Features Explorer

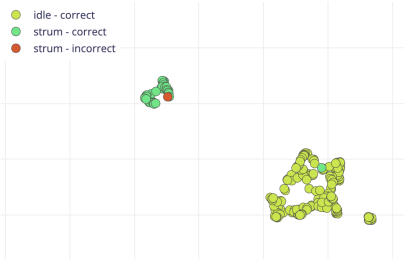

Figure: Accelerometer Features Explorer

Learning Curve

The learning curve shows that the model is trained properly.

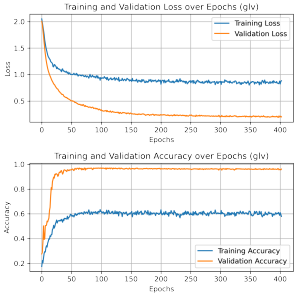

Figure: Flex Sensors Learning Curve

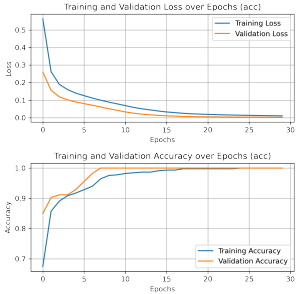

Figure: Accelerometer Learning Curve

Note: Both models converge well before the max epochs, but full training is shown for better analysis.

Demo

Link to Demo Video

Observations & Insights

- The flex sensors are hard to work with if we use the fabric glove. Rubber gloves may fit better but lack of reusability and sanitization.
- We tried bluetooth communication first but it was too laggy and hard to implement. UART (wired) is more reliable.
- The uncertain time required for both Arduino to predict and communicate add aditional difficulty to synchronize the strumming and chord playing.
- Python script is a good tool to serve as a bridge between the two Arduino and the computer through serial communication: (`pyserial`) for serial monitoring and `playsound` for playing the sound.
- Edge Impulse is a powerful tool for training models.

Conclusion

- We successfully created a virtual guitar playing experience using machine learning and Arduino.
- We achieved a testing accuracy of 91.21% for the flex sensors model and 99.55% for the accelerometer model.
- The project has potential applications in music education and entertainment.
- The project also demonstrates the potential of TinyML in creating interactive experiences.
- A 3D-printed mechanical hand may be used to improve the accuracy of the flex sensors model due to its stability.
- Data from more people can be collected to improve the model's generalization based on different hand dimensions & shapes.