Phoneme Recognition using fEMG

NeurotechSC 2023
UC Santa Cruz

Why Phoneme Recognition?

FUTURE GOALS

Word recognition from facial EMG signals

COMMUNICATION

Allows for subvocal speech transcription

DISABILITIES

Not all people are capable of producing vocalized speech

INTERACTION

Computer-Human Interfaces; technological progress and exploration

Neuroethics

- Data Ownership: Who owns the data and how is it used?
  - Data is only recorded with the consent of the user; kept securely and free of personal information, data is only used for training the model

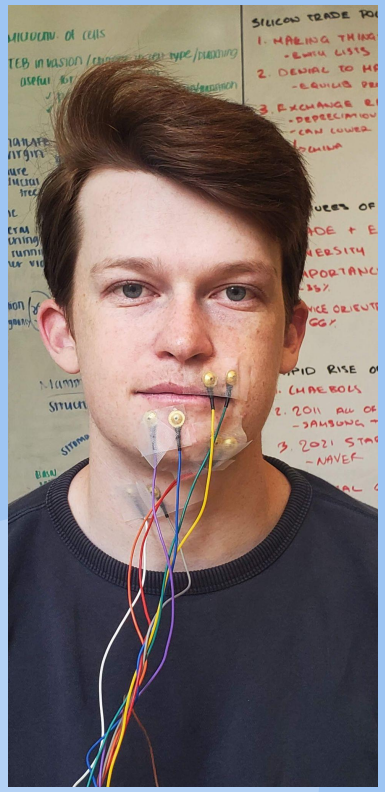
- Data Acquisition & Bias: What sources of bias exist in the data?
  - The data included recordings from both a 20 year old male and female subject, but the final model was only trained on the former
  - May struggle to generalize to more people

- Technological Use: How can this technology be used to ensure a better future?
  - This technology will be applicable in various fields, including silent communication for military or emergency personnel, assistive devices for people with speech impairments, and hands-free control of computers and other electronic devices.

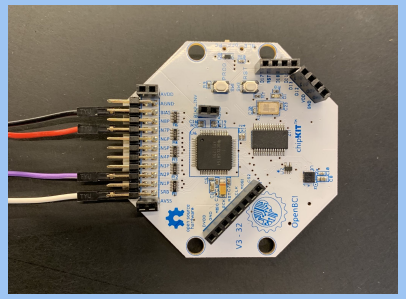
- Ethical Considerations: What are some ethical risks/challenges and how are they addressed?
  - The project code and development documentation is open source, so our methodologies and results can be replicated/verified.

Overview

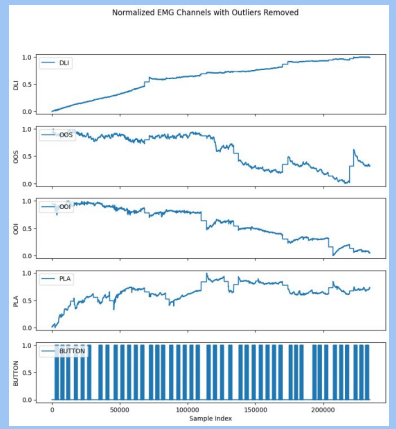
FACIAL EMG

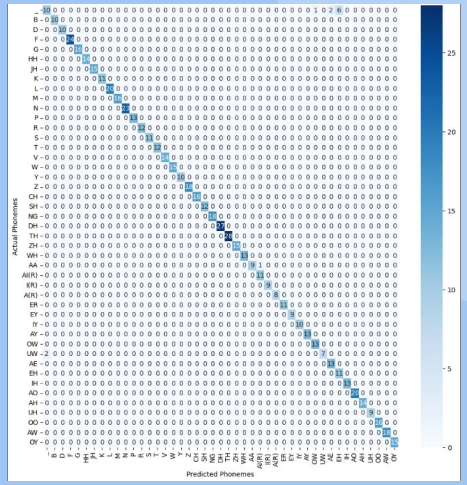


HARDWARE

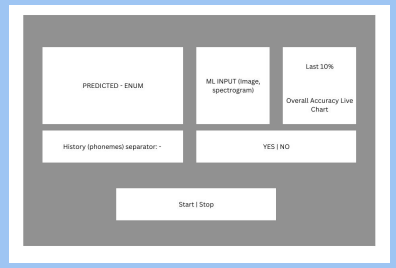


DATA & ML

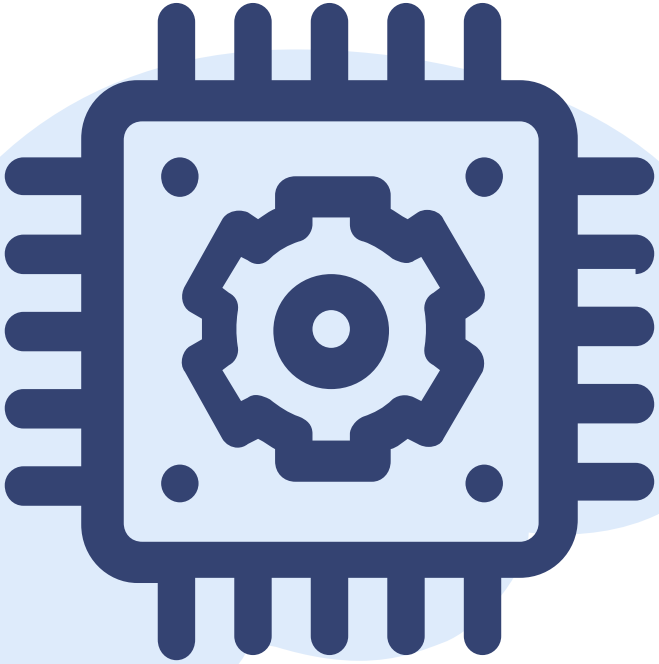




SOFTWARE

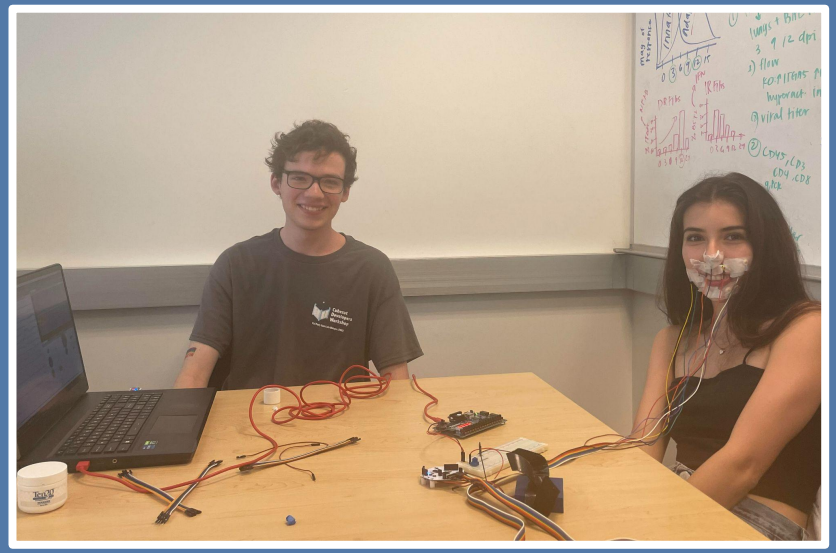


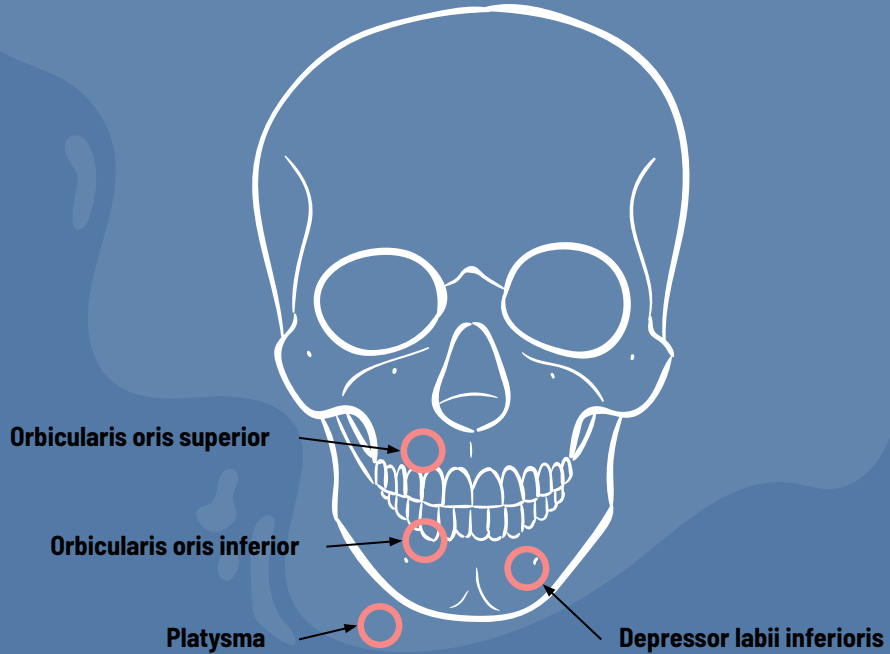
PREDICTED PHONEME



01
Hardware

Data Acquisition

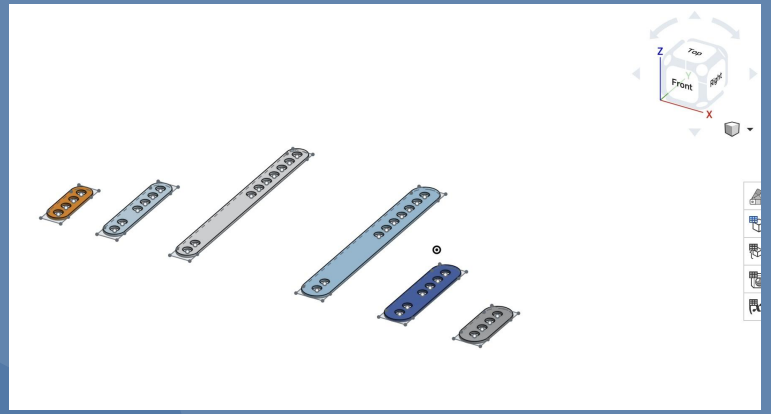


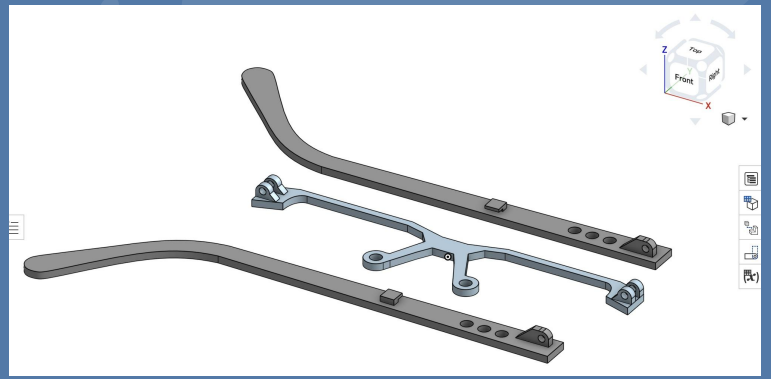


Recorded by 9 Surface Dry Electrodes connected to an OpenBCI Cyton Board using OpenBCI GUI. Data was collected at an amplification factor of 1170, 16 bits A/D conversion, a resolution of 0.298 microvolt/ bit, and a frequency range of 0.9-295 Hz. EMG sampled at 250 Hz. With recordings performed in a push-to-talk setting in quiet rooms, but without electrical shielding: We expect this to be closer to real-life usage than using a specialized recording room. Used a hardware button to mark phoneme pronunciation time segments.

OnShape Mask

- Utilizing OnShape, we made a mask to conduct proper and accurate placement for the electrodes
- Bars of varying sizes would be used to sit on the face and place the electrodes into
- Design was not finished and tested in time for completion and implementation





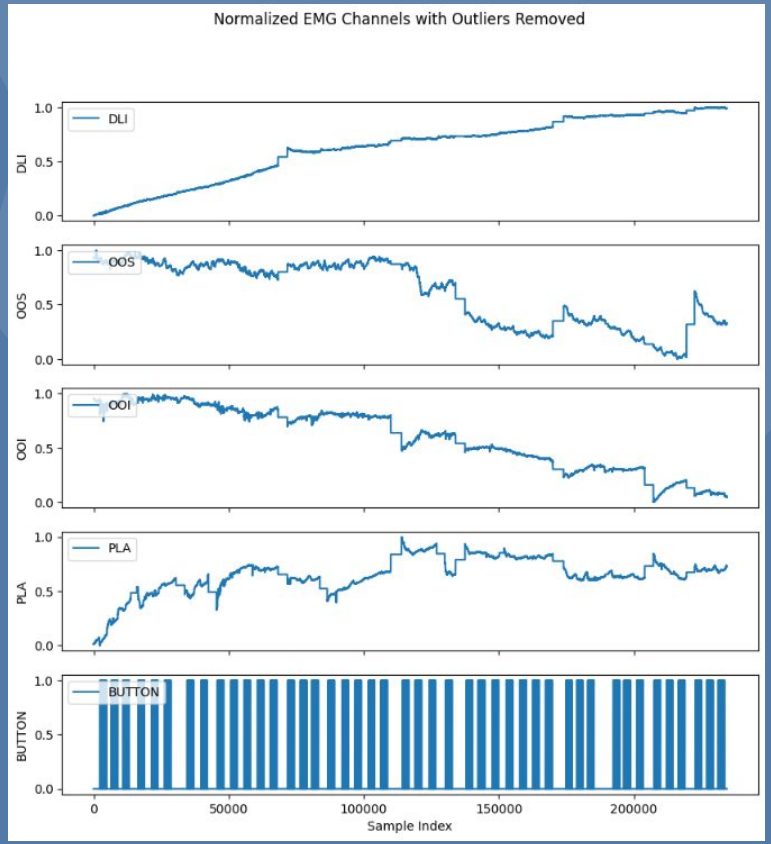
02
Data & ML

Data Analysis and Processing

- Initial steps included assessing which phonemes, muscle groups, and data processing techniques made sense to focus on
- The project began by collecting and analyzing data for 5 phonemes
- Phoneme data set was expanded from an initial 5 to a diverse set of 22
- Eventually recorded all 44 phonemes with the four best muscle groups

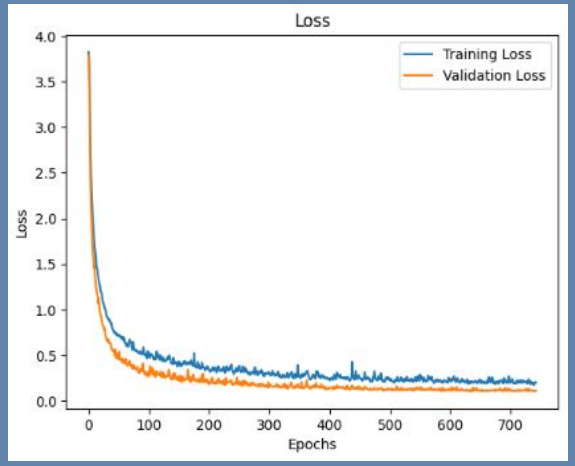
  Got phonemes, letters, words, phrases, and paragraphs
- Basic outline of data processing pipeline: filter columns, remove outliers, min-max normalization, clean up BUTTON channel, identify segments, chunk data into segments, generate training examples from those chunks, and create .npy files for training examples/labels
- Concerns and challenges: outliers, normalization, background noise/trends, lack of data, training data doesn't fit real-life use case

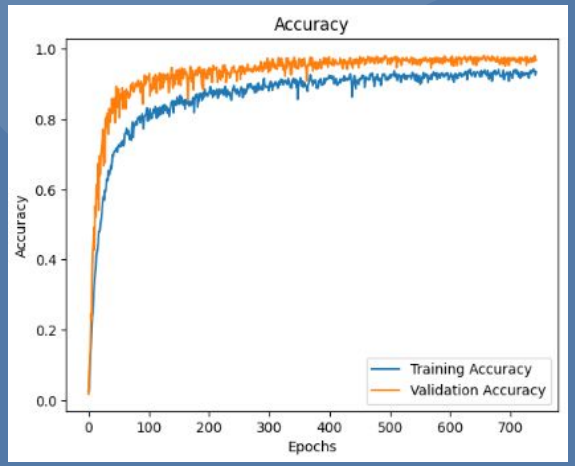
```
Number of segments: 440
Average segment length: 37.5 values
Minimum segment length: 19 values
Maximum segment length: 66 values
Standard Deviation of segment length: 8.36 values
Average segment length (ms): 150.01 ms
Minimum segment length (ms): 76 ms
Maximum segment length (ms): 264 ms
Standard Deviation of segment length (ms): 33.46 ms
```



Machine Learning (DNNs)

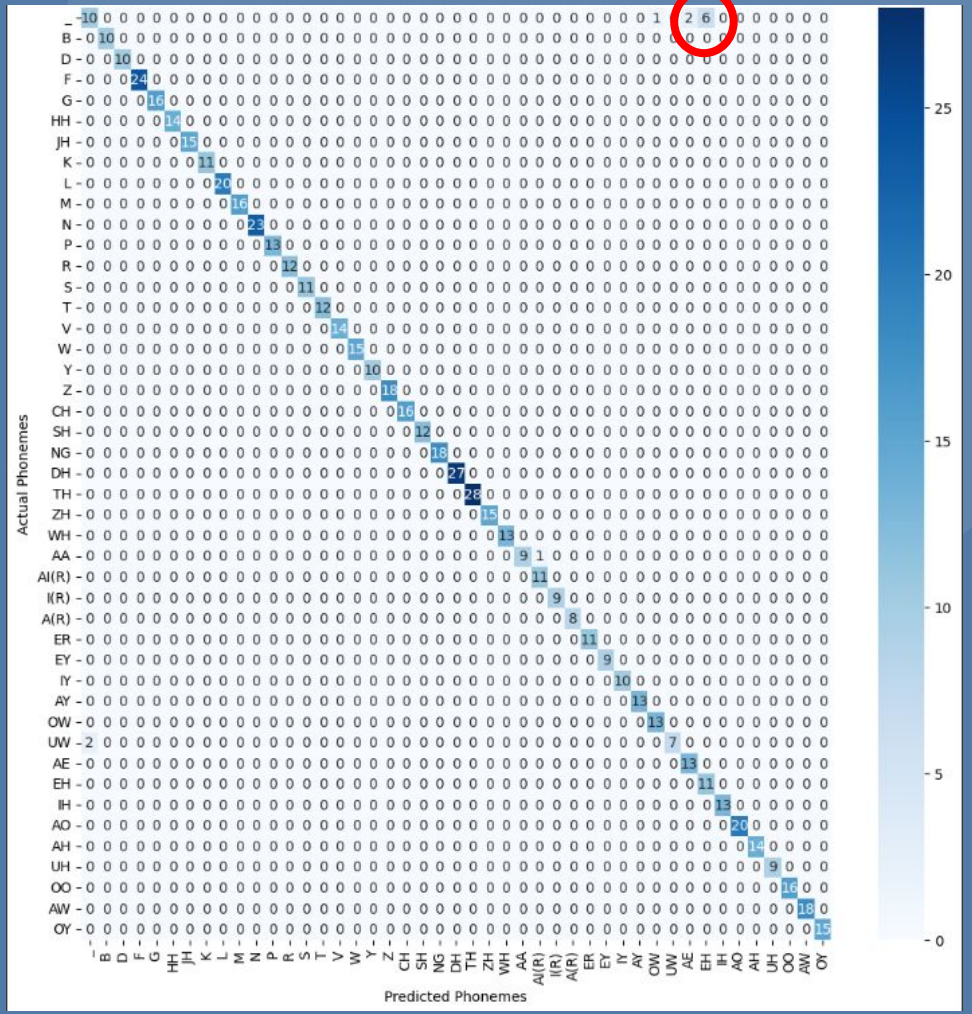
- Initially, the project used a TMC-ViT model, a deep learning architecture that combines vision transformers and temporal multi-channel features, which seemed a good fit
- After a series of experiments, the team switched to an LSTM/RNN model, which proved to be more effective (and simpler) for phoneme recognition
- The model's capabilities were progressively enhanced, expanding from recognizing 5 phonemes to a diverse set of 22, and eventually all 44
- Kept most hyperparameters similar between trainings, model concerns were typically due to data quality/amount rather than parameter count or other model failings
- Used standard 80/20 test split to see performance on a validation set that isn't used for training
- Used a Keras sequential model with Conv1D, LSTM, Dense, and Dropout layers, compiled with Adam optimizer and sparse_categorical_crossentropy
- Experimented with a variety of network architectures/layer designs, tested models ranging from ~100k to ~2 million parameters, eventually achieved ~98.1% accuracy on test data with only 223,533 parameters
- Loss, accuracy, and confusion matrix graphs gave insight on issues
- Concerns and challenges: overfitting, data leakage, trained only on single phoneme pronunciations





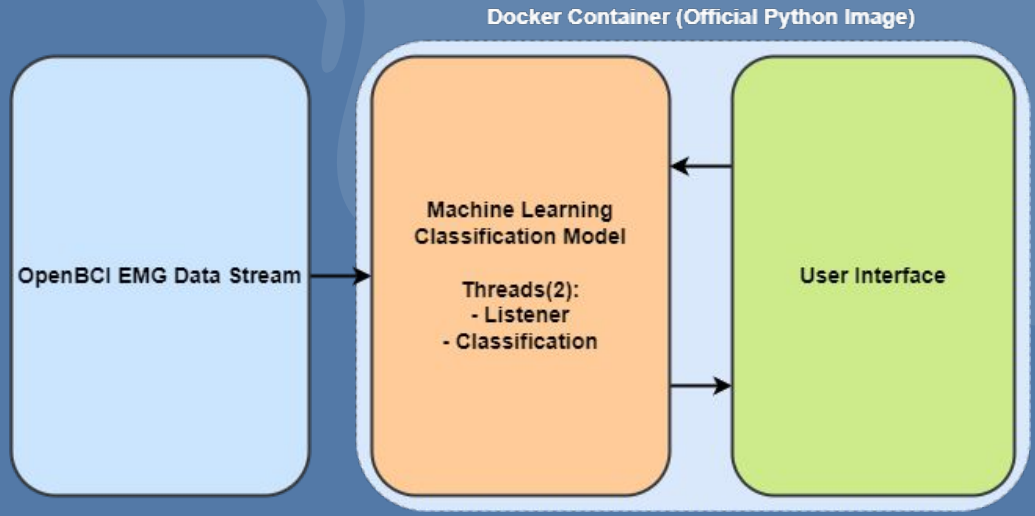
Confusion Matrix

- A multiclass confusion matrix is a table that helps us understand how well a classification model is performing when there are more than two classes. It shows the model's predictions compared to the actual outcomes for each class
- Each row in the confusion matrix represents the true class, and each column represents the predicted class
- The **diagonal** elements represent the **true positive predictions** for each class
- **Off-diagonal** elements represent **false predictions**
- The confusion matrix helps us understand how well the model is performing for each class
- Here, the model predicted 6 of the silence examples as 'EH', giving insight into possible data/model issues

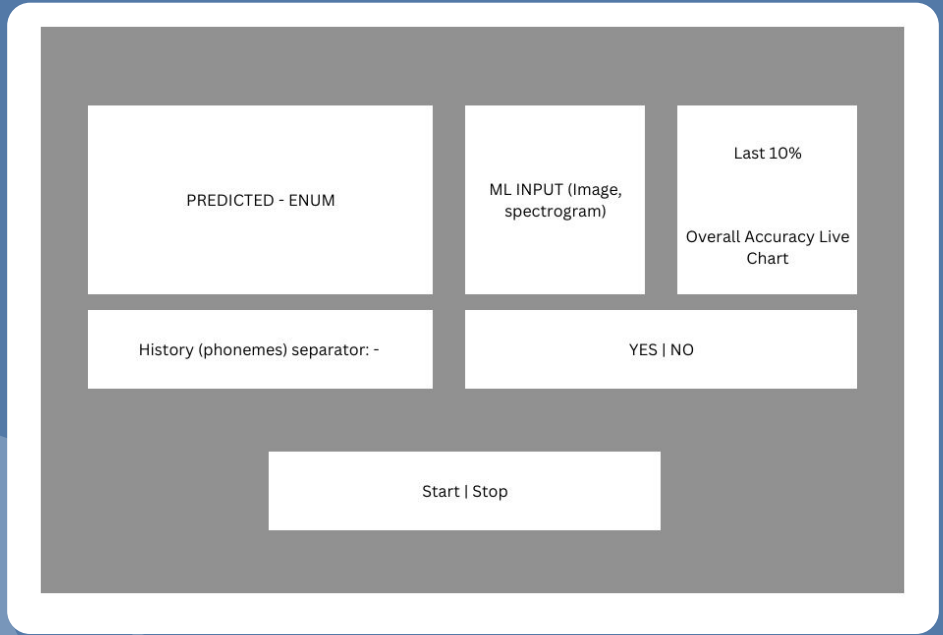


03
Software

Proposed Architecture



Proposed User Interface



- Show a live phoneme transcription over 10 second intervals
- User will then classify the prediction as correct/incorrect
- Future Work:
  - Pass phoneme predictions to another machine learning model to predict text

User Interface: What We Tried

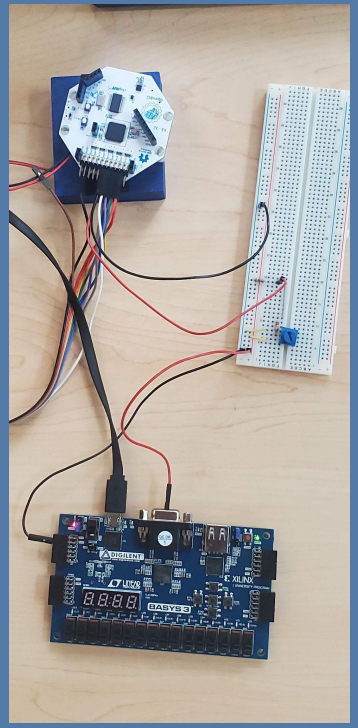
First Attempt

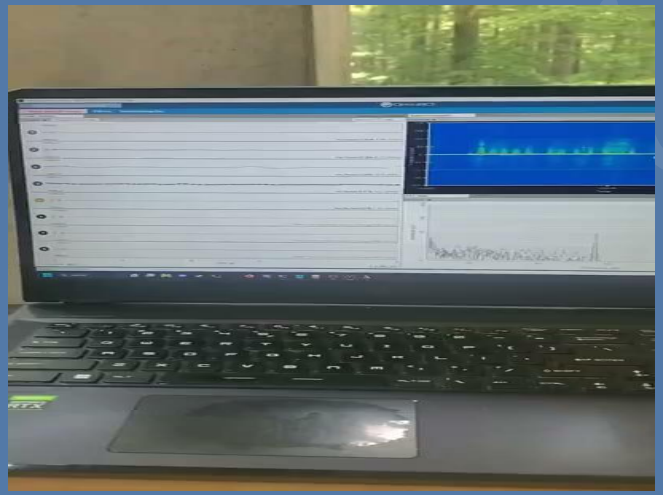
- Web App with Python (Flask), ML model and GUI in separate containers(processes)
  - Websocket to send data back and forth
  - Not everyone was familiar with creating an API
  - Unnecessary for our use case
- Database to store data
  - MongoDB

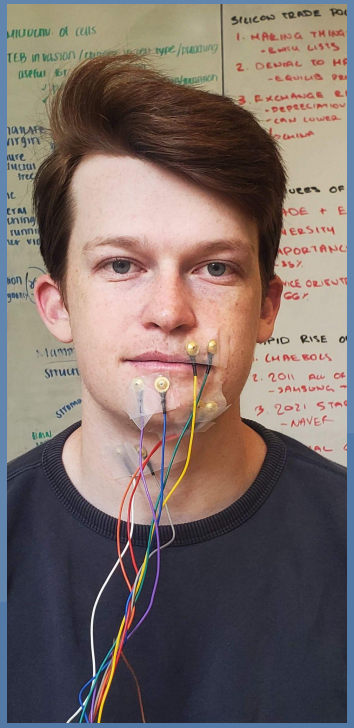
Second Attempt

- Python (tkinter), ML model and GUI in same process.
  - Comes with python standard library
  - Less overhead
  - Shared variables instead of sending messages through a websocket
  - Simpler and easier to learn
  - Cannot containerize tkinter
- Database to store data
  - MongoDB

04
Demo


DIGILENT
XILINX
BASYS 3





Limitations and Future Steps

- Limitations
  - Background noise/trends in recordings
  - Electrode placement issues (consistency, adjusting causes spikes)
  - Not enough data/wrong type of data
- Future Steps
  - Building/Acquiring new hardware for more accurate and efficient data collection, such as: Wet Electrodes, 3D printed mask, FPGA board
  - More data collection for better model performance/robustness
  - Streamlined recording protocol for faster and more efficient/accurate recordings.
  - Finish integrating model/data processing into live transcription app

Thank you

THANKS!

Do you have any questions?
neurotechsc@gmail.com
neurotech.ucsc.edu

CREDITS: This presentation template was created by Slidesgo, and includes icons by Flaticon, and infographics & images by Freepik