Sprint #3

Instrument Recognition Software

Aleks
Aner
Axel
Cuong
Joe
Thomas

BRD and Management Plan

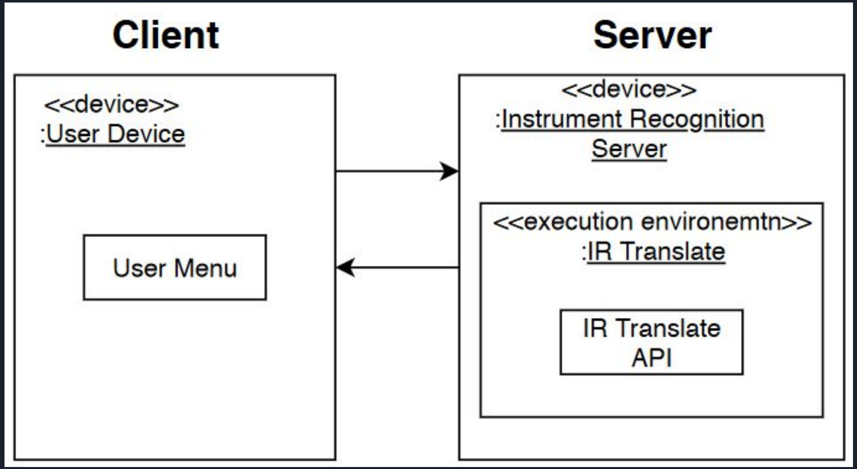
- Business Requirements Document
  - Our team did not have any significant update in our Business Requirements Document

- Management Plan
  - For our Management Plan, we updated the Sprint Board and the Burndown Chart

Architecture

- Client
    - The user's side of the software that contains a menu which allows them to interact with the software

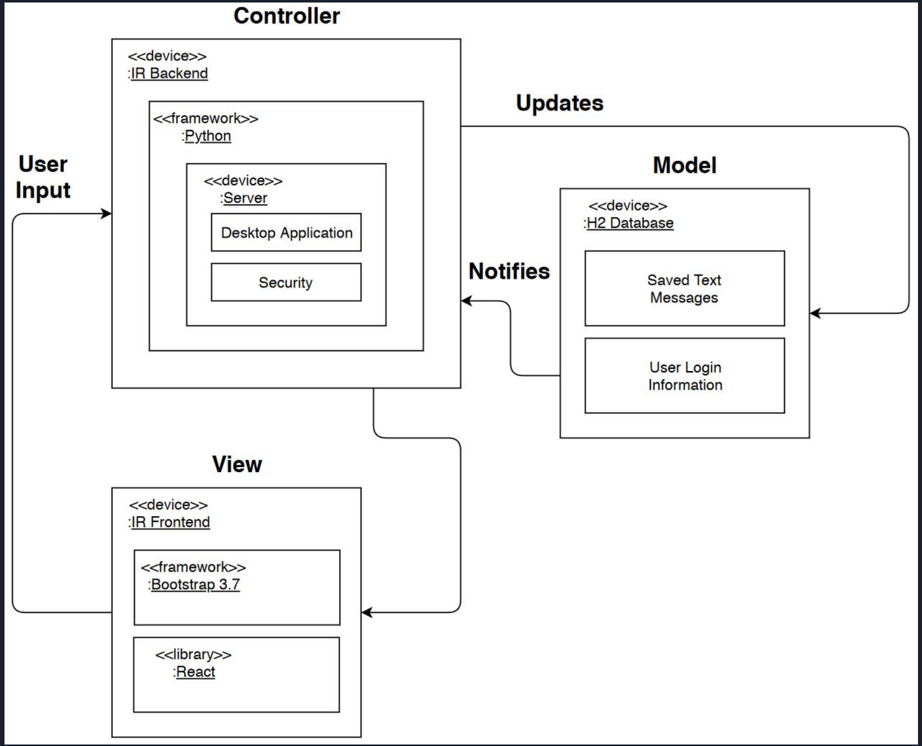
- Server
    - The server handles the user registration and login authentication, machine learning model, and handles the calculation of the algorithms.



Client-Server interaction

Architecture

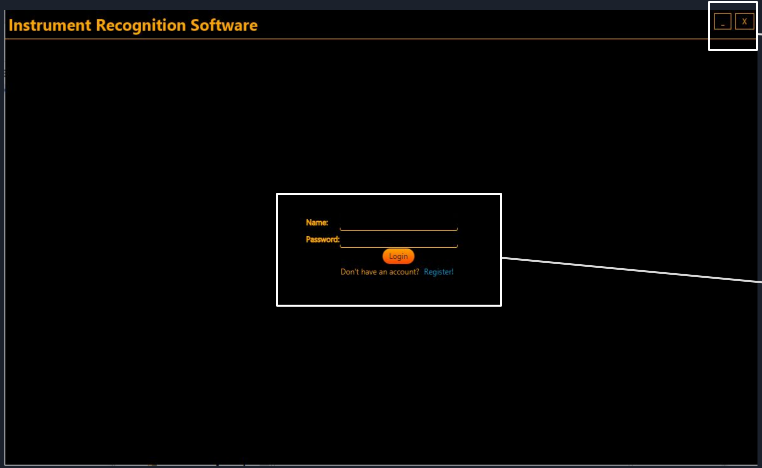
- Controller
  - The Controller manages the updates and serves as the main process for Model and View
- Model
  - The Model contains the login information and audio files
- View
  - The View provides the user with the final outcome

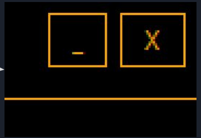


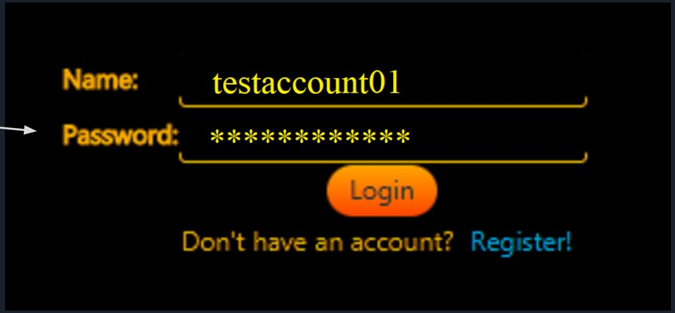
Controller-View-Model interaction

Architecture

- Front-end
  - JavaFX
    - Built using JavaFX SDK
    - Desktop Application







Prototype registration and login page

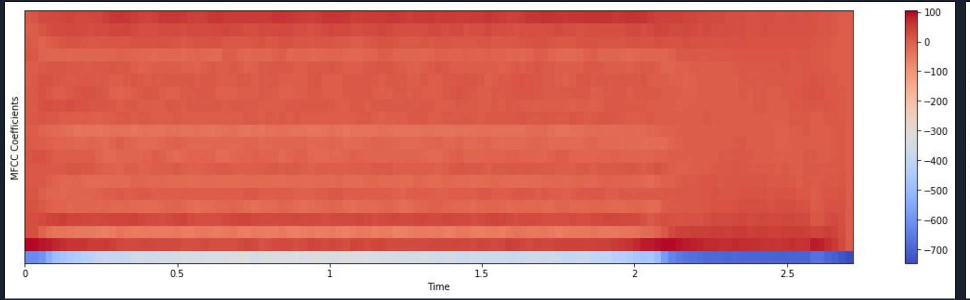
Login and Registration interface

Architecture

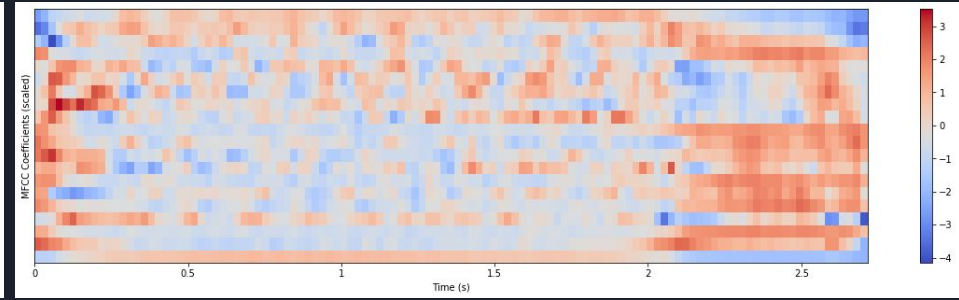
- Back-end
  - Server language: Python
  - Database: MongoDB (non-relational)
    - Connected using PyMongo
    - Stores the usernames and hashed passwords

Features

- Mel-Frequency Cepstral Coefficient (MFCC)
  - Commonly used as features in speech recognition systems
  - Increasingly finding uses in music information retrieval applications such as genre classification, audio similarity measures, etc.
  - Will be used to analyze the Timbre (the uniqueness of an instrument from another)
  - Commonly derived by using both Fast Fourier Transform (FFT) and Discrete Cosine Transformation (DCT)



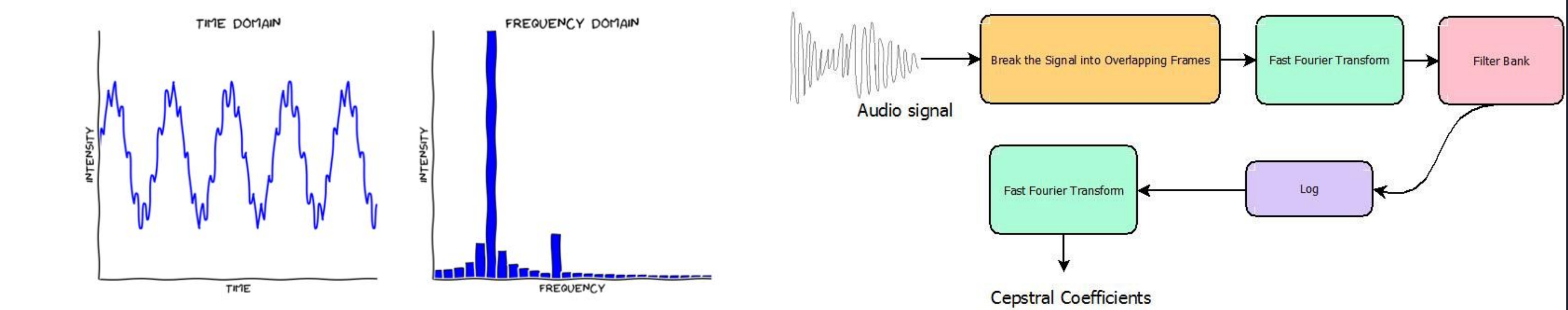
MFCC Spectrogram (unscaled)



MFCC Spectrogram (scaled)

What MFCC is doing

- Mel-Frequency Cepstral Coefficient (MFCC)
    - First it takes a FFT in the time signal
    - Then takes a log of that result and it takes a DCT
    - Finally it takes another FFT on the frequency spectrum.
    - Mel-spaced filterbank will be used after to specify what frequencies we want to look at



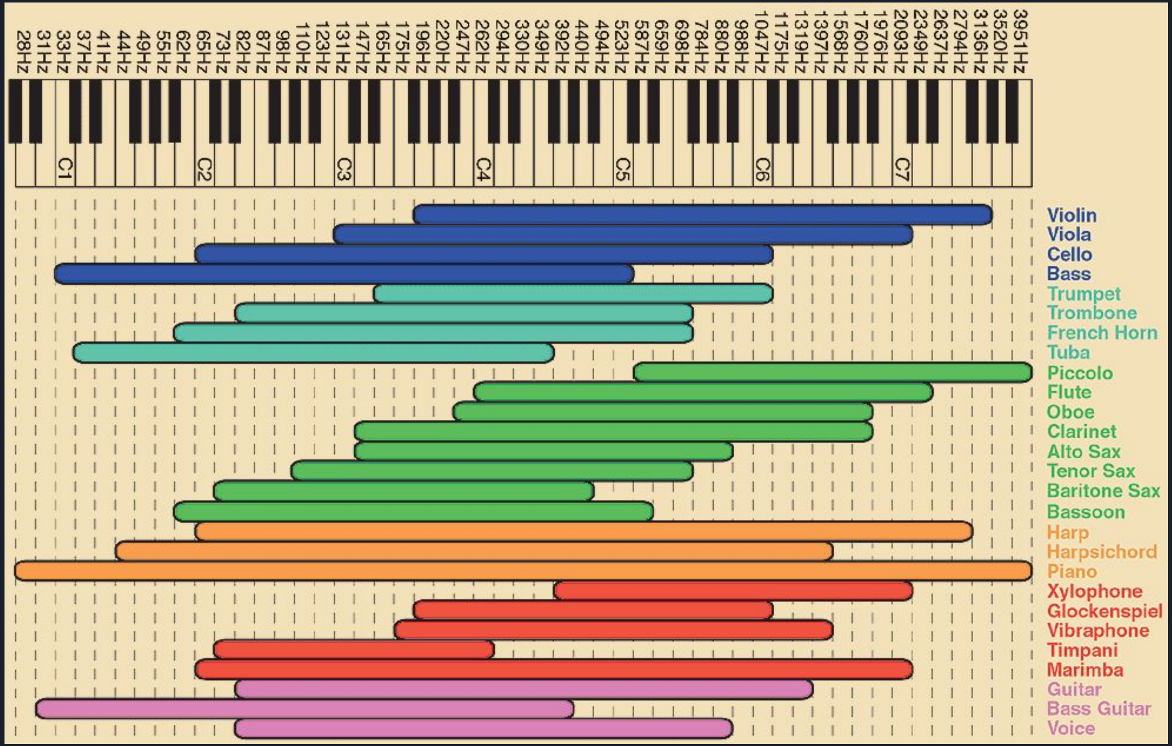
Features

- Tone
  - Can distinguish instruments based on their highest and lowest achievable frequency

Frequency Band	Frequency (Hz)
Low Bass	20 - 40
Middle Bass	40 - 80
Upper Bass	80 - 160
Lower Midrange	160 - 320
Middle Midrange	320 - 640
Upper Midrange	640 - 1280
Lower Treble	1280 - 2560
Middle Treble	2560 - 5120
Upper Treble	5120 - 10,200
Top Octave	10,200 - 20,400

Human Hearing Range

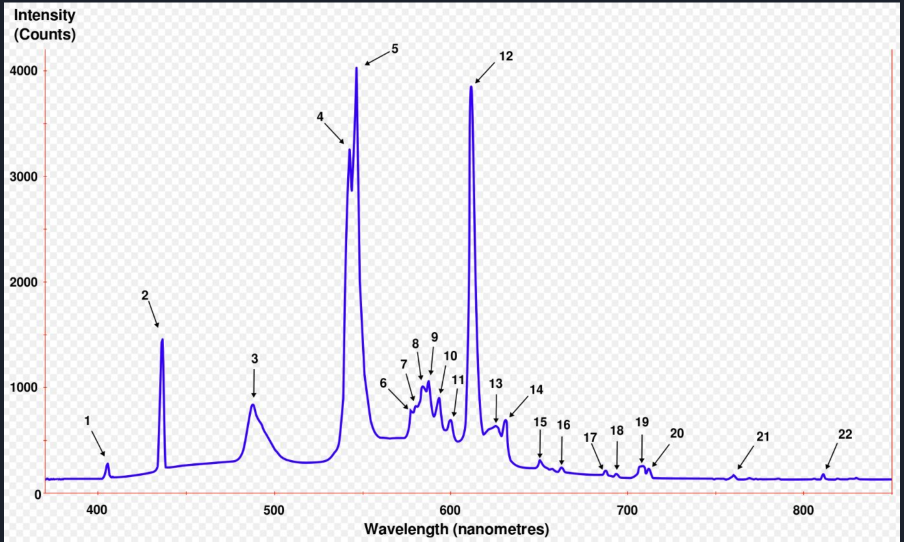
Features



Frequency Range for Instruments

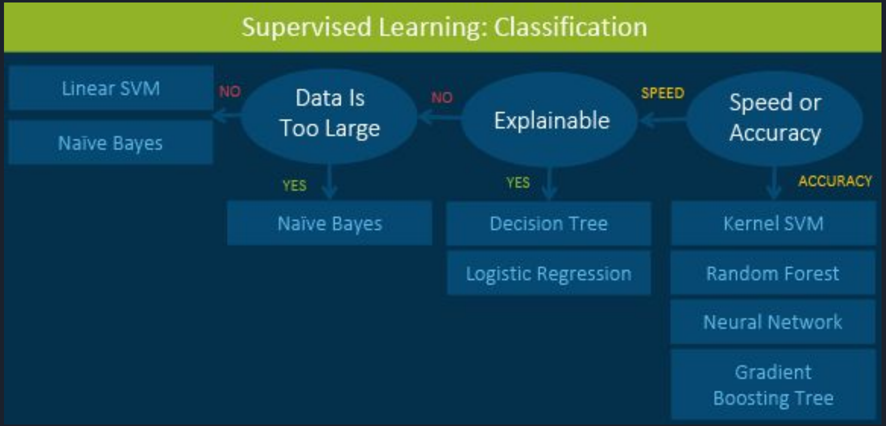
Features

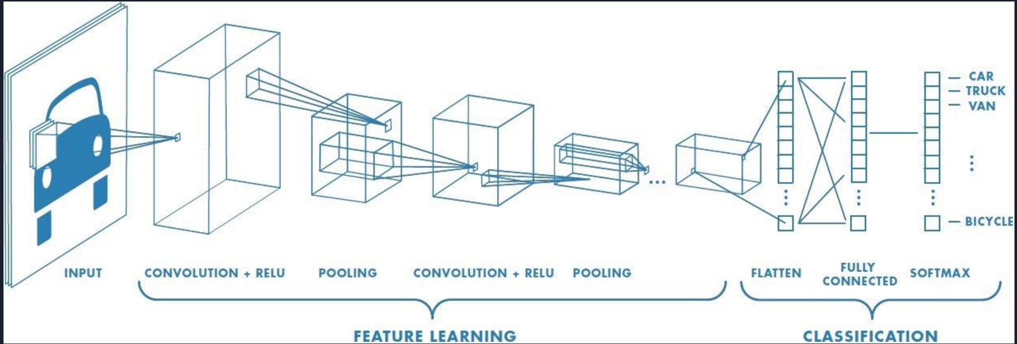
- Power Spectral Density
  - Shows the strength of the variations(energy) as a function of frequency
  - Unit of PSD is energy per frequency(width)
  - Can obtain energy within a specific frequency range by integrating PSD within that frequency range
    - Computation of PSD is done directly by the method called FFT or computing autocorrelation function and then transforming it

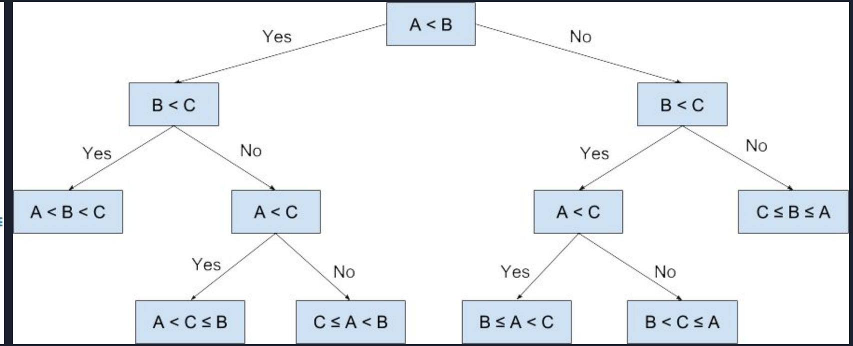


Candidate Models

- Label-focused machine learning:
  - Decision Tree
  - Random Forest
  - Support Vector Machine (SVM)
  - Convolutional Neural Network (CNN)







Model Decision: Convolutional Neural Network

What is CNN?
A CNN is a powerful classification model that compares large data to recognizable patterns to determine a label.

- Data Preparation
- Convolution
- Pooling
- Normalization
- Drop out
- Soft Max

Why CNN?

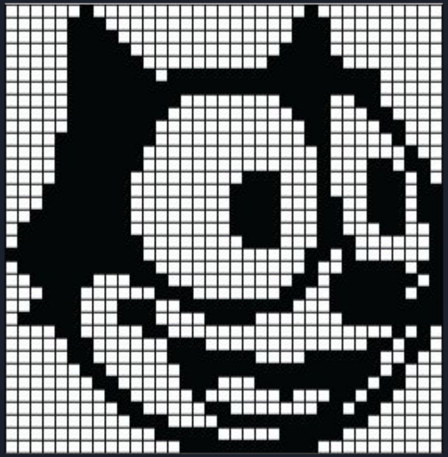
- Image processing
- Pattern recognition
- Large data
- Recommended by research

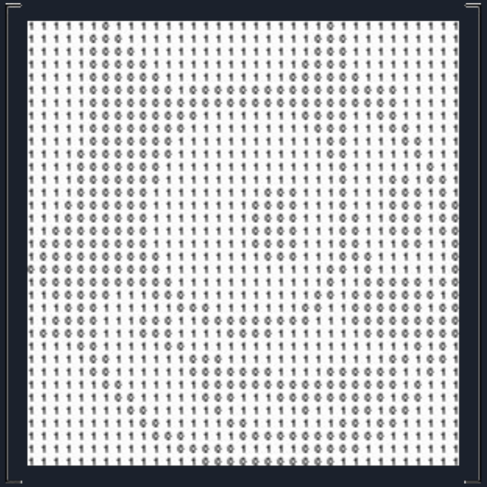
How was it constructed?

- Developed in PyCharm - Python IDE
- 96 Layers
- 3x3 Filters

Understanding the Model: Data Preparation

Image Transformation:





Layer/Kernel/Weight/Feature Matrix:

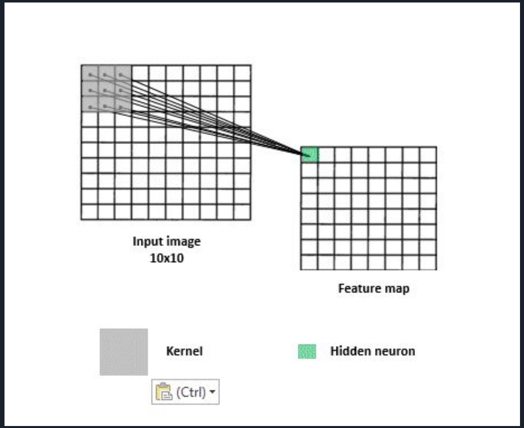
0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Visualization of a curve detector filter

Understanding the Model: Convolution

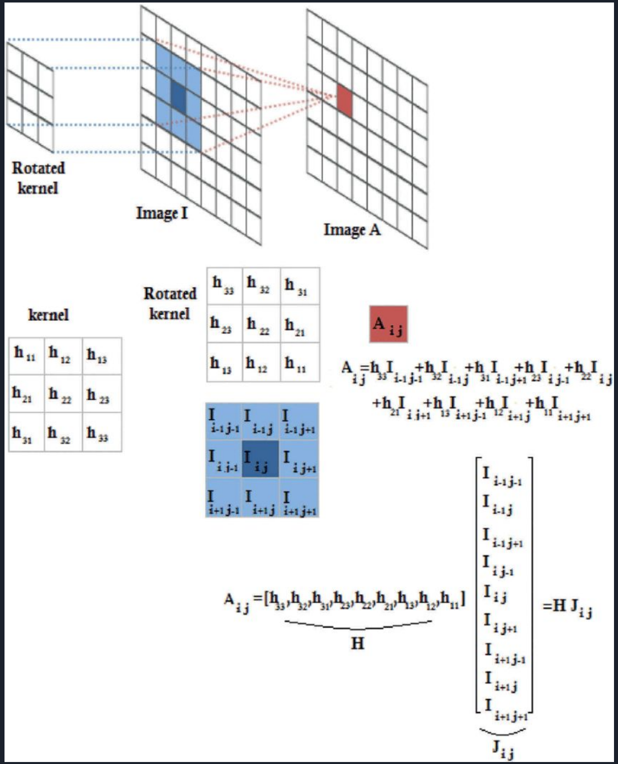
What it does:



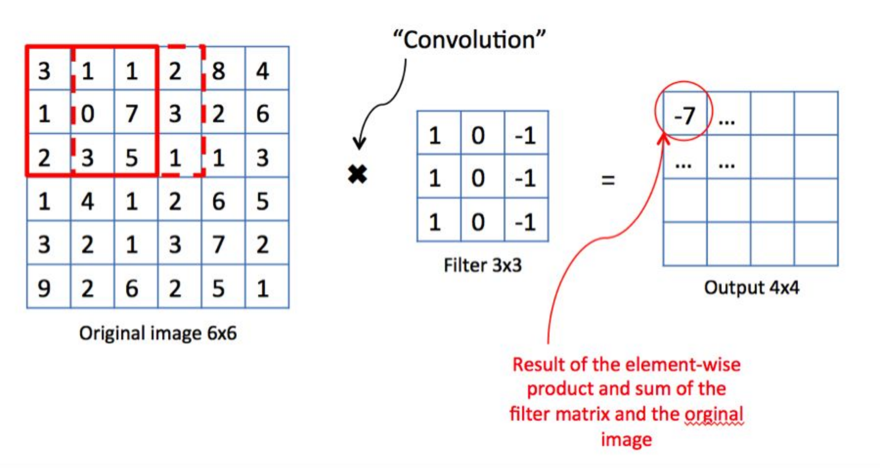
Above, a portion of the input matrix is condensed into a feature map by forming a dot product between a receptive field and a kernel

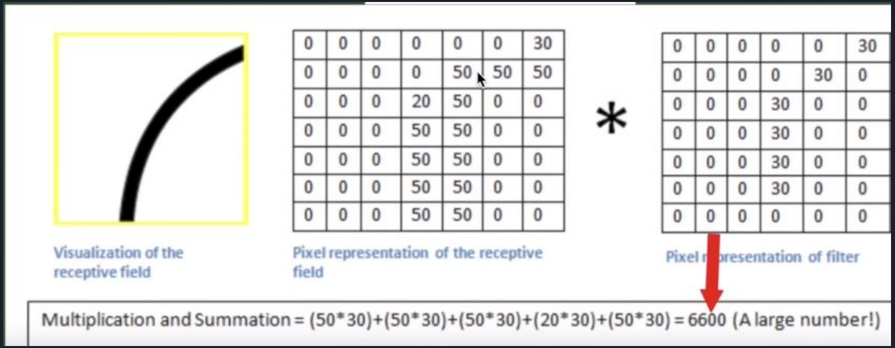
Why it works:

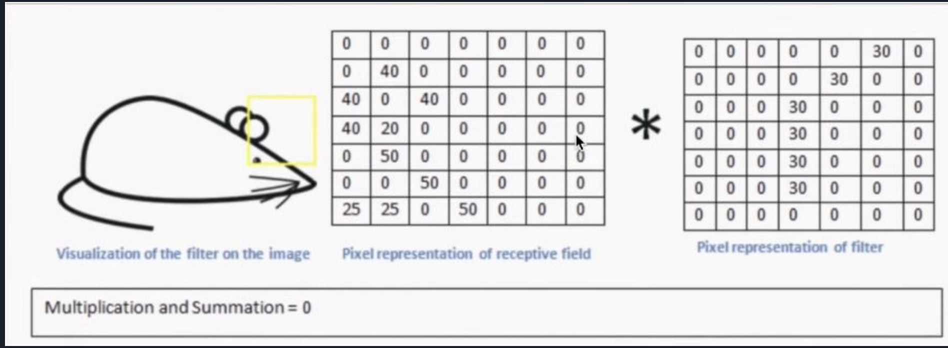
By taking the summation of associated products, a value can be obtained representing the similarity of the input area to the pattern given



Understanding the Model: Convolution Example

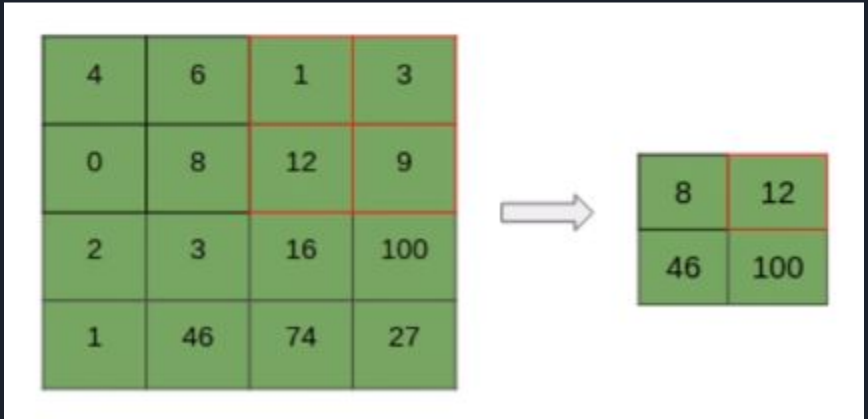






Understanding the Model: Pooling & Normalization

Pooling:
The process of taking the largest (most relevant) data from a field to condense a matrix



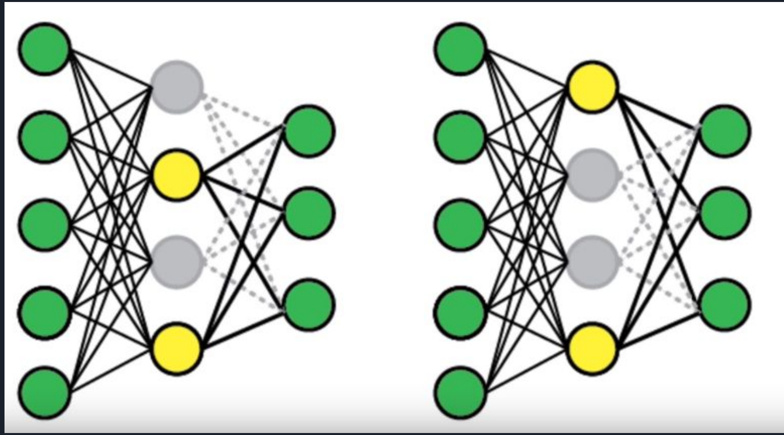
Normalization:
Prevents the math from breaking.
Turns all negative values into zero.

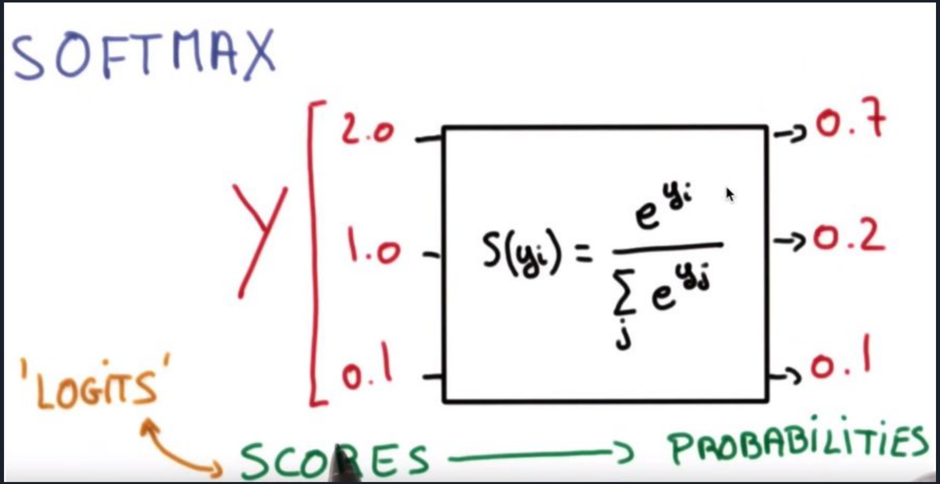
reLU

- A library process for normalization on a stack of matrices

Understanding the Model: Drop out & Soft Max

Drop out:
The random deactivation of neurons (entries in a convolved matrix) to reduce noise and combat over-fitting



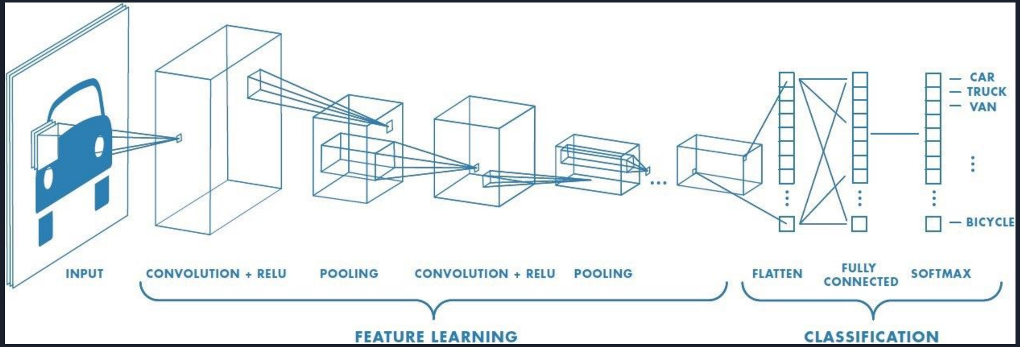


Soft Max:
Condenses matrices into a set of final probability values for each category. From here, the highest probability is taken and used as the label for the input.

Code Implementation

```
#model building
model = Sequential()
#convolutional model using layers
model.add(Conv2D(32, kernel_size=(3, 3),
                activation='relu',
                input_shape=input_shape))
#adds 32 convolution filters each of size 3x3
model.add(Conv2D(64, (3, 3), activation='relu'))
#64 additional convolution filters that are also 3x3

#choose the best features through pooling
model.add(MaxPooling2D(pool_size=(2, 2)))
#implements dropout at a 25% chance
model.add(Dropout(0.25))
#condenses the matrices
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
#uses a softmax to create probabilities
model.add(Dense(num_category, activation='softmax'))
```



Model Analysis and Reflection

Analysis:
The model provided an average accuracy of ~99.13% when given 60k training data and 10k for testing over 10 trials.

It is hard to say whether the model has been over or under fitted, but given the high dropout rates provided it is more likely under than over.

Reflection:
The model performed at an insanely high accuracy rate that is likely due to the simplicity of the given task. Unlikely to carry over to musical data but we can be optimistic.

We are in a very good spot for next semester as we already have a way to transform musical data into images the CNN can interpret.

Next ML Models

What are the next two machine learning models the team plans to code, test and demonstrate?

We plan to improve the CNN model and adapt it to our project. We may also implement a decision tree model if necessary but as of now we do not have plans for it.

Month we plan to complete the coding of a ML model from an actual piece of music

February/March - projected for sprint 1 completion

Demos

- Youtube sound conversion and analysis.
- Client-Server interaction.
- Sample CNN.

Sprint Goal

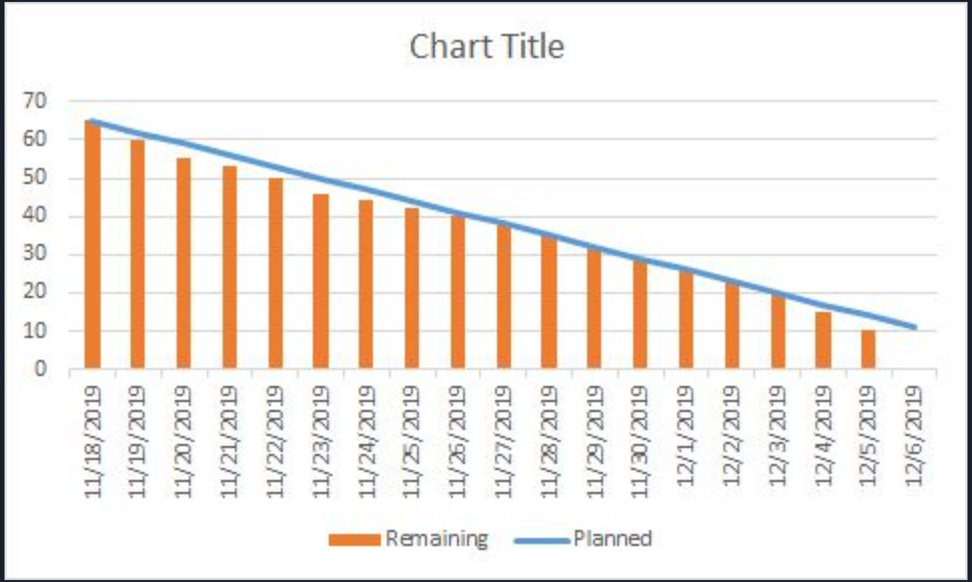
The goal of this sprint was to create samples of machine learning algorithms we had researched in previous sprints.

User Stories

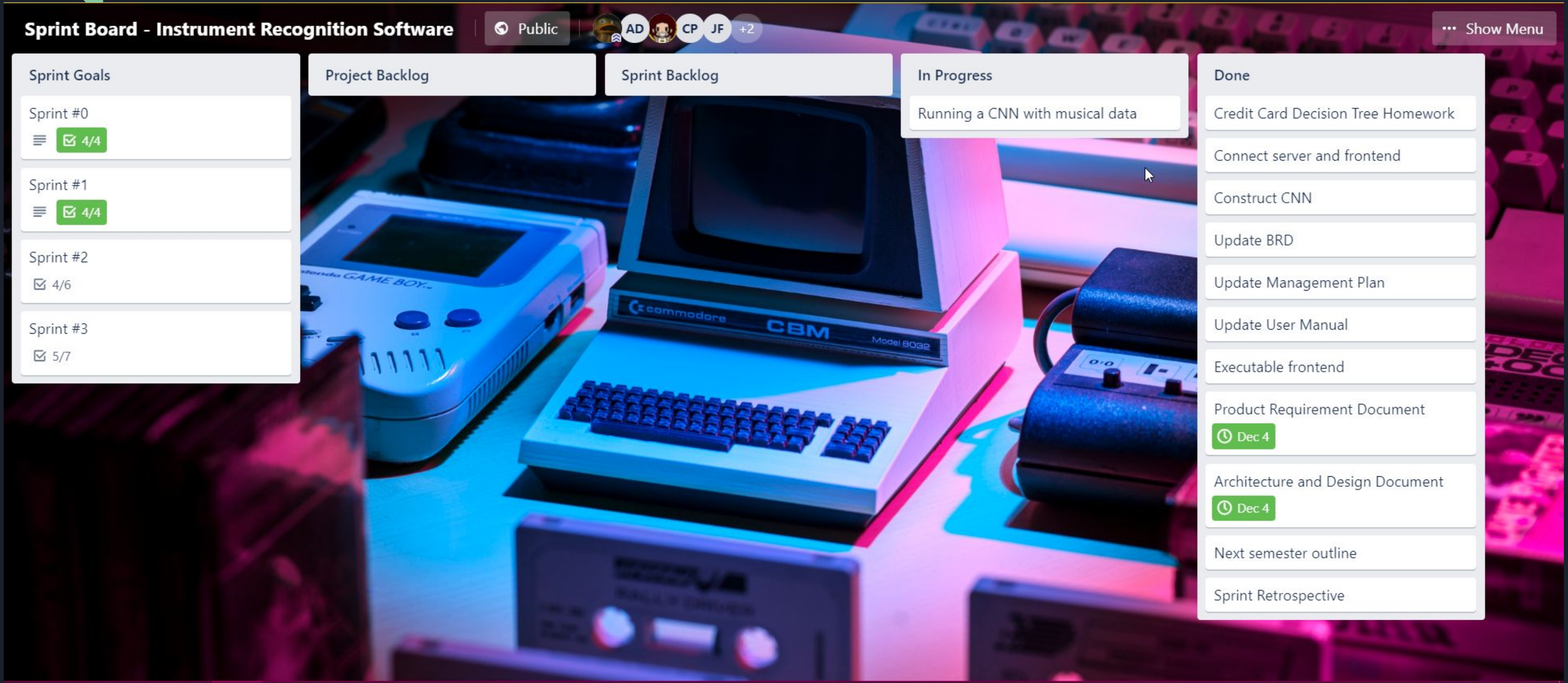
Planned: 10

Accomplished: 3

Burndown Chart



Sprint Board Goal



Sprint Retrospective

- Created and tested a sound converter
- Created and tested a CNN model
- Created and tested Client-Server-Database interactions