Using Transfer Learning, Spectrogram Audio Classification, and MIT App Inventor to Facilitate Machine Learning Understanding

by

Nikhil Bhatia

S.B., Massachusetts Institute of Technology (2020)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2020

Author . . . .
Department of Electrical Engineering and Computer Science
May 18, 2020

Certified by . . . .
Harold Abelson
Class of 1922 Professor of Computer Science and Engineering
Thesis Supervisor

Accepted by . . . .
Katrina LaCurts
Chair, Master of Engineering Thesis Committee

Using Transfer Learning, Spectrogram Audio Classification, and MIT App Inventor to Facilitate Machine Learning Understanding

by

Nikhil Bhatia

Submitted to the Department of Electrical Engineering and Computer Science
on May 18, 2020, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Recent advancements in deep learning have brought machine learning and its many applications to the forefront of our everyday lives. As technology has become more and more integrated into our educational curriculum, researchers have focused on creating deep learning tools that allow students to interact with machine learning in a way that incites curiosity and teaches important concepts. My research contribution focuses on applying transfer learning and spectrogram audio classification methods to teach basic machine learning concepts to students. I introduce the Personal Audio Classifier (PAC), a web application that allows users to train and test custom audio classification models that can classify 1-2 second sound bites recorded by the user. Alongside this in-browser machine learning tool, I provide a set of best practices for spectrogram audio conversion in machine learning applications. I also contribute a custom App Inventor extension that allows users to use the output of the web interface to create App Inventor applications that rely on their trained custom audio classification model. Finally, I provide a high school workshop curriculum based on PAC and the App Inventor extension and detail the results of running workshops with three high school classes at the Boston Latin Academy. My results show that high school students with no prior machine learning knowledge are able to grasp important technical concepts related to machine learning and its applications, as well as build and explore custom machine learning models in the browser through the exploration of a hands-on curriculum based around PAC.

Thesis Supervisor: Harold Abelson
Title: Class of 1922 Professor of Computer Science and Engineering

Acknowledgments

I'm grateful for the support of the MIT App Inventor team in pursuing this project. Specifically, I thank my advisor Hal Abelson for pushing me to go above and beyond in my exploration; my PhD advisor Natalie Lao for inspiring my project and providing me with constant support and direction; Danny Tang for paving the path before me with the Personal Image Classifier; Evan Patton for his patience and assistance with development and debugging; Jeff Schiller for his technical expertise; Selim Tezel and Karen Lang for their expertise in educational research and assistance with curriculum design.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

From personal voice assistants to self-driving cars, machine learning applications have permeated every aspect of our daily lives. Many of these advances are thanks to the subfield of machine learning known as deep learning, a field primarily concerned with building large neural networks to perform specialized tasks. Deep learning architectures such as deep neural networks, recurrent neural networks, and convolutional neural networks have been widely applied to fields such as audio recognition, computer vision, and natural language processing. In these architectures, each layer of a network is responsible for learning a transformation from its input data to a more abstract representation. In an image classification problem, this input data might consist of a 3-dimensional matrix of pixels, and the first layer might compose a composite representation of edge pixels, while a later layer might encode features like a nose or mouth. A key feature of deep neural networks is their ability to learn this representation without the need for human input. Even so, architectural decisions, including the number of layers or even the number of nodes in each layer, are vital to the ultimate success of a deep learning model.

In recent years, the research field of machine learning has developed a better understanding of how to tailor deep neural networks to specific problems that hitherto could not be tackled using standard algorithms. Yet as researchers have continued to make significant advancements in deep learning during the past decade, it has become clear that computational complexity, training time, and esoteric development tools could pose as a deterrent to the widespread development of deep learning applications. As machine learning continues to permeate various aspects of our daily lives through applications such as facial recognition and even autonomous vehicles, it is vital that those without access to complex machine learning technology stacks can still gain an understanding of the underlying technical concepts behind these tools so that they feel comfortable utilizing them. In doing so, it is important to build powerful machine learning tools and education curriculum targeting students and those without significant machine learning experience. I present the following contributions in this thesis:

1. The Personal Audio Classifier (PAC), a machine learning tool that allows users to build, analyze, and export custom audio-classification models in their web browser without the need for expensive technology stacks.

2. A framework for performing on-the-fly spectrogram audio conversion that can be used to translate audio classification to the image classification problem space in the browser.

3. An MIT App Inventor extension that allows users to use audio-classification models exported from PAC to build custom mobile applications.

4. A high school workshop curriculum that utilizes PAC and the App Inventor extension to explore custom audio-classifiers and teach basic machine learning concepts.

5. An analysis of results collected from running two workshops across three Boston Latin Academy AP Computer Science classes.

The structure of this thesis is as follows: Chapter 1 continues to provide the motivation and background for the Personal Audio Classifier and introduces a number of machine learning tools that are instrumental to its development. Chapter 2 discusses prior work related to the development of in-browser machine learning tools and provides an understanding of the technical concepts that are important for performing audio classification using convolutional neural networks. Chapter 3 introduces the Personal Audio Classifier and details the design choices, implementation, and functionality behind the web interface. Chapter 4 introduces MIT App Inventor and explains the implementation and use cases for the PAC extension. Chapter 5 delves into the workshop curriculum and explains how PAC is used to teach machine learning concepts. Chapter 6 gives a high-level view and provides analysis of the results collected during the Boston Latin Academy workshops. Chapter 7 concludes the thesis and re-summarizes my contributions. Chapter 8 discusses potential extensions of my work and future development of related machine learning curriculum.

1.1 Applications of Transfer Learning to K-12 education

In the past decade, colleges and higher education institutions around the world have embraced machine learning and artificial intelligence as an integral aspect of advanced computer science curriculum. Academic and industry professionals have joined forces to build out a complex set of programming tools that allow us to push the boundaries of machine learning understanding while developing real-life applications of artificial intelligence that continue to permeate our every day lives. Frameworks like Tensorflow [12], Pytorch [11], and Keras [13] have given researchers and engineers unprecedented ability to build and tune complex neural networks. However, the development of these complex technology stacks coupled with the increasing computational complexity and training time for deep neural networks has made machine learning less accessible to those below the college level. While some might argue this is a necessary side effect of advancement in a complex research field and that machine learning curriculum should be restricted to post K-12 education, studies such as [1] have shown that interacting with machine learning tools at a younger age provides an important introduction to the development of computational thinking skills. Thus, the motivation of this thesis work is to contribute to making machine learning concepts accessible to K-12 students through in-browser tools that do not require users to have experience with esoteric programming frameworks.

1.2 Transfer Learning

In order to make machine learning models more accessible to a broader audience, I draw on the large body of work related to transfer learning, a concept born out of the aforementioned deficiencies, and first introduced by Yosinki's 2014 paper [4] on transferable features in deep neural networks. Transfer learning is a machine learning method where an existing deep learning model is used as the starting point to train a new model specialized for a slightly different task. The ability to start with a pre-trained model allows students to apply deep learning to solve novel problems without the vast compute and time resources normally needed to train neural networks from scratch. To take an example, we can look at one recent notable application from the field of deep learning: the development of robust deep convolutional neural networks used for image classification and object recognition. In 2012, Hinton [3] first showed that a deep neural network with over 60 million parameters and 650,000 neurons could be trained to classify ImageNet images with startling accuracy. However, developing this complex model required a uniquely efficient GPU implementation of the convolution operation, tens of thousands of dollars in state-of-the-art GPU hardware, and months of training and testing. While this development process is likely only accessible to researchers or institutions with deep pockets, the result is one that should be available to developers of all levels and students of any age. Transfer learning has allowed for just this, giving machine learning enthusiasts around the world the ability to build their own models using complex models like Hinton's as a starting point.

1.3 Tensorflow.js and MIT App Inventor

By using transfer learning as a starting point, the possibilities for novices to experiment and understand machine learning are endless. However, if we want to create effective machine learning engagement tools for students and machine learning novices, we have to think beyond the neural network. To remain engaged, students must interact with machine learning in a way that precipitates curiosity while demonstrating the power of deep learning models.

Suppose Oli, a high school student, has recently been introduced to machine learning. Oli has only a surface-level understanding of machine learning and computer science concepts, but has learned to identify instances of machine learning in his everyday life: things like FaceID on his iPhone, or interacting with Siri and Alexa. Recently, Oli has become especially fascinated by the power of personal assistants, and has noticed that Siri and Google Assistant can be trained to only respond to his own voice. Motivated by the ubiquity of privacy and technology in today's media headlines, Oli wants to combine his interest in machine learning with an attempt to build his own privacy-conscious application. He comes up with the following goals:

1. Learn how audio-classification works, and train his own custom audio-classifier that can detect features like speaker, emotion, language, etc.

2. Leverage this audio-classifier to create a private voice diary on his own phone that would only unlock at the sound of his voice.

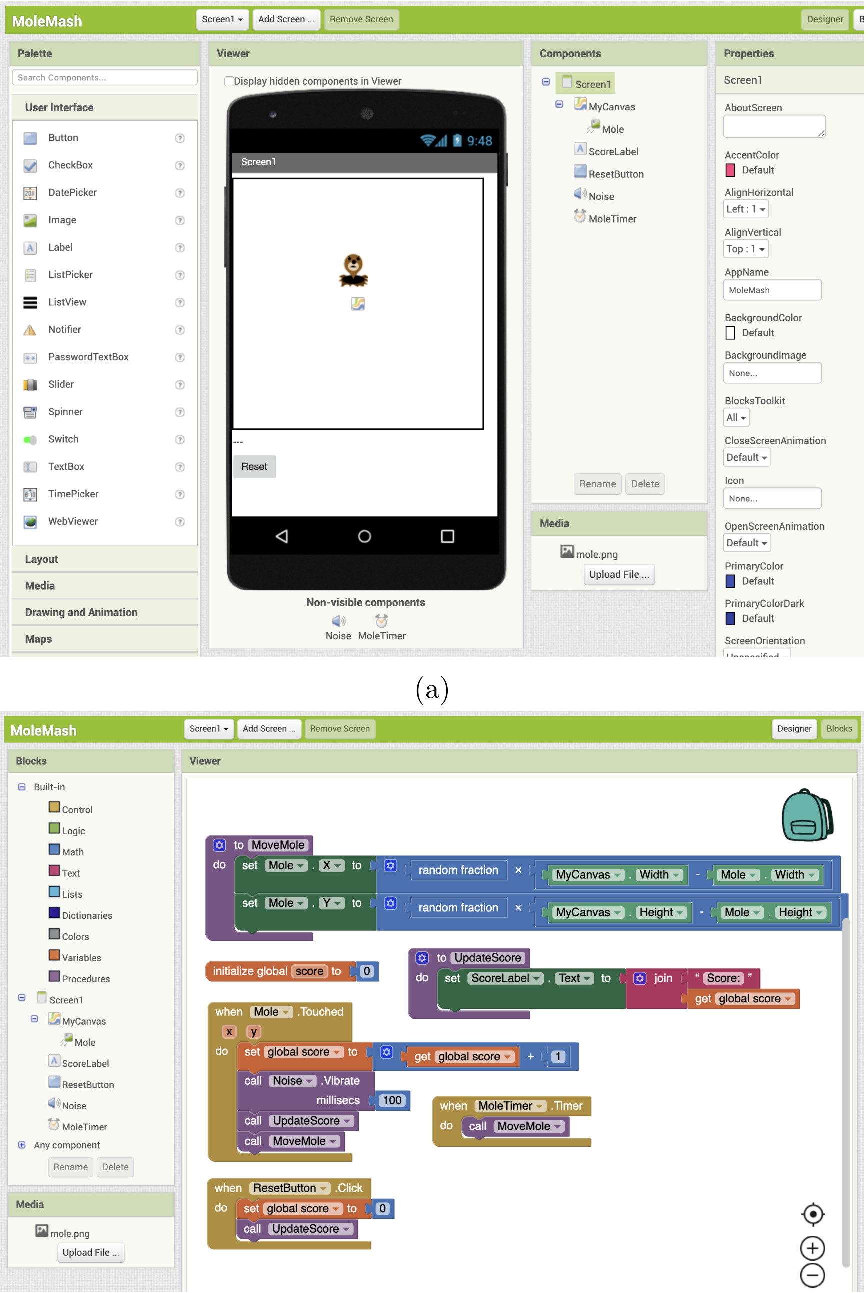
To help Oli achieve this goal, we can build him a flexible audio-classification tool that uses transfer learning to retrain a model to classify unique types of audio. With such a tool, Oli could easily train a model to identify different types of emotion, or even the sound of his own voice. However, Oli does not come from a computer science background and would likely be unable to use machine learning libraries like PyTorch [11] or TensorFlow [12]. Furthermore, Oli has little experience developing mobile applications and hopes to find a way to achieve his goals without needing to learn an entire mobile development stack.

Thus I introduce two important technologies, Tensorflow.js [14] and MIT App Inventor [2], that this project utilizes to help students like Oli develop exposure to machine learning concepts without requiring a deep computer science background. Tensorflow.js is a Javascript machine learning library that has recently found success in the niche bridging machine learning implementation and educational tools. It allows for deep learning models to be trained and run right in the browser, and when combined with a well-designed web GUI, can hide the complexities of programming syntax while still allowing users to interface with machine learning models. Similarly, MIT App Inventor (Figure 1-1) is a free open-source web platform that allows users to create mobile applications via a drag-and-drop interface, requiring little to no programming experience while still offering rich application functionality. App Inventor also offers the ability to add custom extensions to any app, allowing us to build an audio classification extension that Oli could upload and use to help build his private diary app.

1.4 Project Tools Overview

This section will provide a brief overview of the tools introduced in this thesis and explain their relationship in the context of developing machine learning understanding. The primary tool described in this thesis is the Personal Audio Classifier (PAC). PAC is a web application application built in React that allows users to train custom audio classifiers in their browsers. Oli could specify custom audio labels, corresponding to his own voice as well as the voices of a few friends, and record a series of audio clips that correspond to each voice (Figure 1-2). The PAC backend passes all recorded audio through a series of mathematical operations to both selectively remove sequences of quiet or anomalistic audio, and convert the raw audio WAV form into a spectrogram that can be processed via modern image recognition techniques. Oli is then prompted to specify the layers and parameters of a transfer model, pre-trained on ImageNet, that is then re-trained to distinguish between the various voice labels Oli specified. Finally, Oli is prompted to record test audio clips to see how successful his trained model is at distinguishing between the various voices (Figure 1-3). At this point Oli can download his trained model from PAC for use in the MIT App Inventor PAC Extension.

The PAC Extension was created for use in MIT App Inventor and allows for Oli



AboutScreen
AccentColor Default
AlignHorizontal Left : 1
AlignVertical Top : 1
AppName MoleMash
BackgroundColor Default
BackgroundImage None...
BlocksToolkit All
CloseScreenAnimation Default
Icon None...
OpenScreenAnimation Default
PrimaryColor Default
PrimaryColorDark Default
ScreenOrientation

(a)

```
to MoveMole
do  set Mole . X to   random fraction × MyCanvas . Width - Mole . Width
    set Mole . Y to   random fraction × MyCanvas . Height - Mole . Height

initialize global score to 0

to UpdateScore
do  set ScoreLabel . Text to   join  " Score: "
                                      get global score

when Mole .Touched
  x  y
do  set global score to   get global score + 1
    call Noise .Vibrate
                millisecs 100
    call UpdateScore
    call MoveMole

when MoleTimer .Timer
do  call MoveMole

when ResetButton .Click
do  set global score to 0
    call UpdateScore
```

(b)

Figure 1-1: (a) The App Inventor designer interface allows users drag and drop components to build an app's visual functionality. (b) The block interface allows users to program blocks and define interactions between various components.

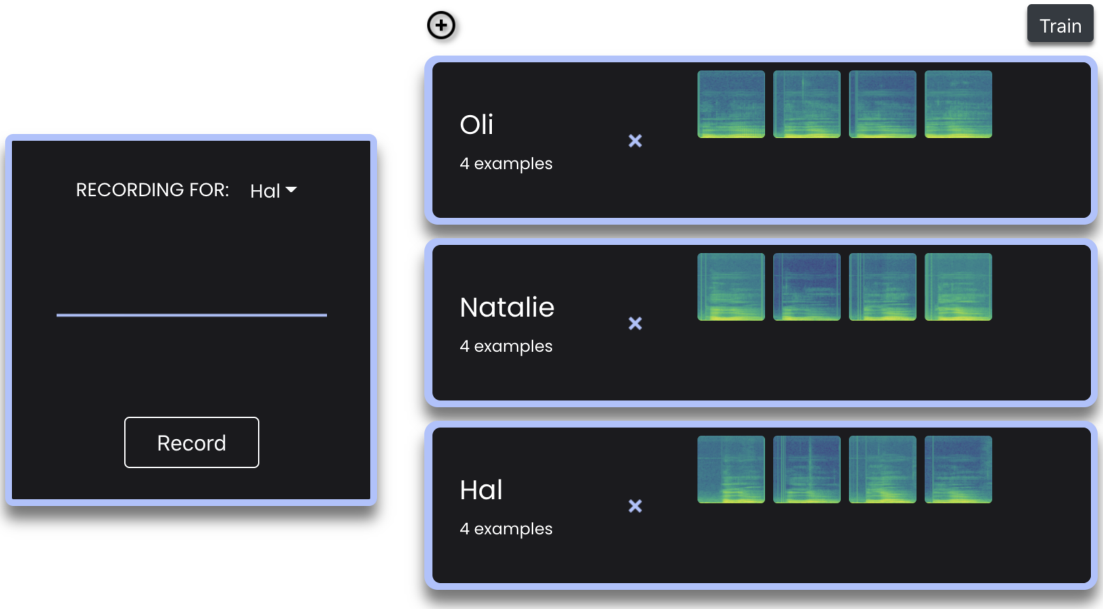
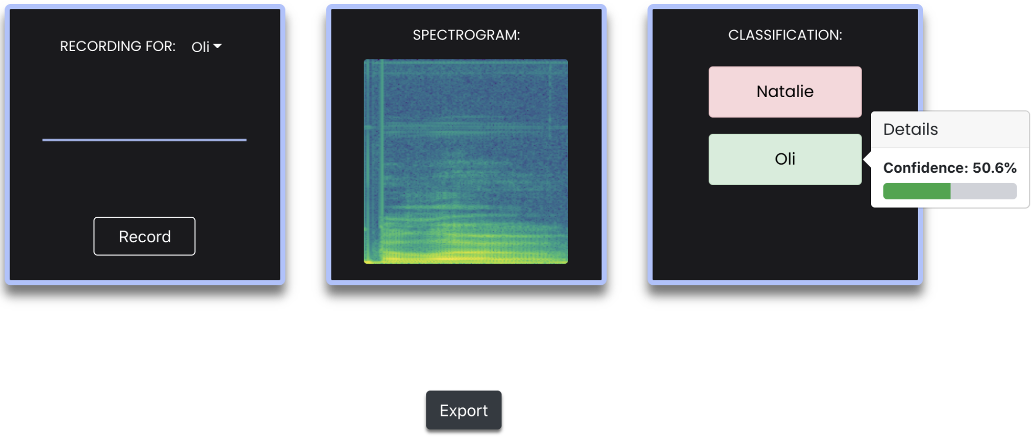

RECORDING FOR: Hal

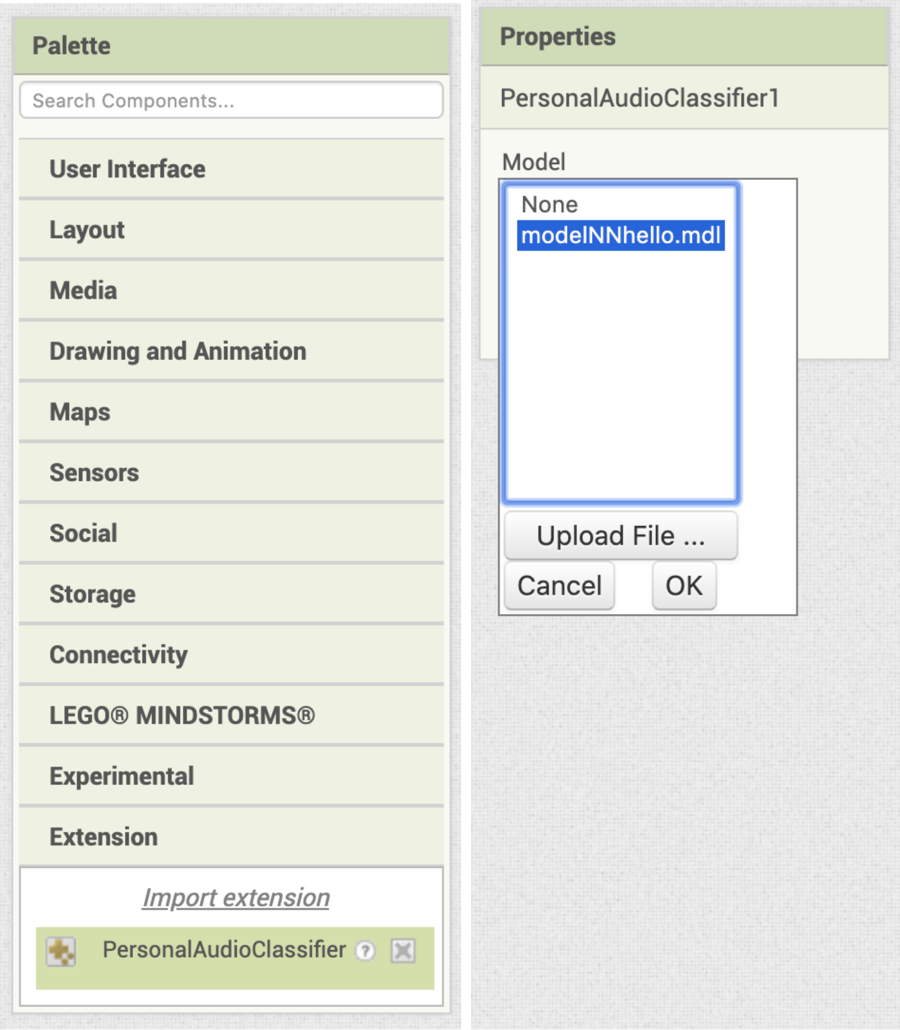
Figure 1-2: The PAC training interface. Users specify custom labels and record training examples for each label.

to import his custom audio classification model directly through the App Inventor interface (Figure 1-4). After uploading his model, Oli can now create an application in App Inventor that records audio clips to provide as inputs to his PAC model, and executes specific actions upon successful classification of these recorded inputs. Oli can use the PAC Extension to build his voice authentication app, only allowing entry to his diary if the recorded voice matches his own (Figure 1-5).



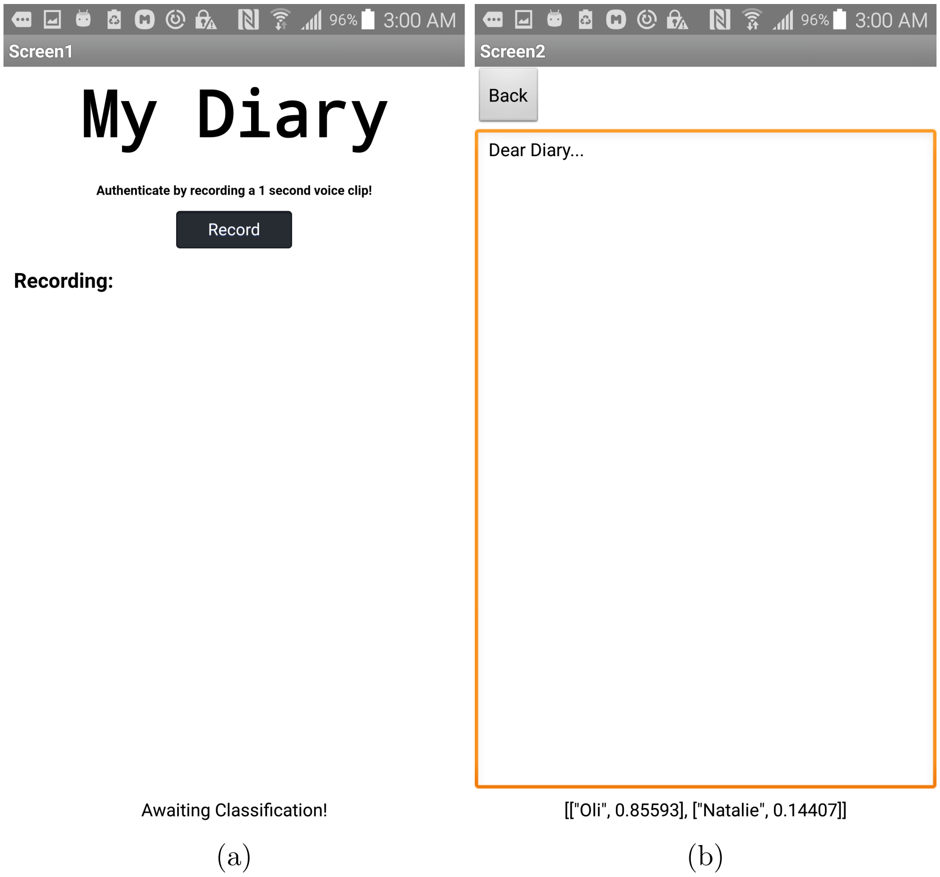
Confidence: 50.6%

Figure 1-3: The PAC testing interface. Users record test audio clips and PAC classifies each recording, showing classification confidence for each label.



(a) (b)

Figure 1-4: (a) App Inventor offers a variety of components and allows for users to build and upload their own extensions (b) Each extension can receive a set of custom properties, allowing users to upload their trained model to be used in their application.



My Diary

Authenticate by recording a 1 second voice clip!

Recording:

Awaiting Classification!

Dear Diary...

```
[["Oli", 0.85593], ["Natalie", 0.14407]]
```

(a)

(b)

Figure 1-5: (a) Oli's voice authentication app allows him to record a 1 second voice clip to authenticate a user. (b) Upon classifying a recording as Oli's voice, the app allows entry to Oli's diary.

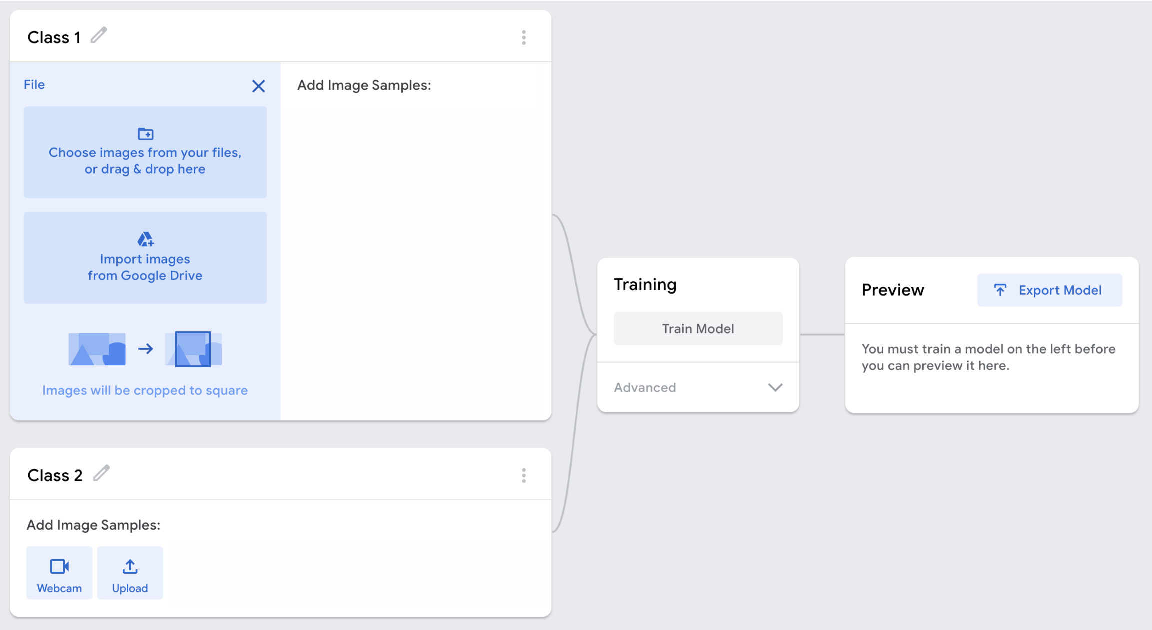
Chapter 2

Prior Work

My work in this thesis draws upon prior work in the fields of transfer learning, image classification, educational machine learning tools, and spectrogram audio classification. This section will first introduce Teachable Machine [5], an educational machine learning tool developed by Google that builds on important concepts including transfer learning and in-browser image classification. Then, I'll introduce the Personal Image Classifier (PIC) developed by Danny Tang in the MIT App Inventor lab [2] from which this project takes significant inspiration. Lastly, I will introduce tools specific to spectrogram audio classification that are important to understanding the theory behind the conversion of an audio classification problem to an image classification problem.

2.1 Teachable Machine

Teachable Machine [5] is a tool created by Google that utilizes TensorFlow.js [14] to increase machine learning accessibility for students. As described in the prior chapter, existing machine learning research is primarily focused on development in frameworks like Pytorch and TensorFlow. Unfortunately, these technologies require significant knowledge and computing resources. Using Teachable Machine, users can build custom image classifiers with nothing more than a Chrome browser and a webcam. In order to provide this functionality, Teachable Machine uses Tensorflow.js to



Choose images from your files, or drag & drop here

Images will be cropped to square

You must train a model on the left before you can preview it here.

Figure 2-1: The Teachable Machine training interface. Users specify labels and record or upload images for each class.

implement a transfer learning model that allows users to specify custom image labels and corresponding images as inputs. It then trains the model and classifies inputs as one of the provided image labels. The user experience is as follows:

1. Training: Users first specify classification labels and either upload images or record images with their webcam, as seen in (Figure 2-1). After populating each label with image examples, Teachable Machine prompts users to train their custom transfer model.

2. Testing: After training the custom model, Teachable Machine provides an interface for webcam or file inputs and classifies the input clips between the user-provided labels (Figure 2-2).

3. Export: After testing the Teachable Machine model, users are prompted to export their models for use in Tensorflow or Tensorflow.js. This allows for applications to be built using these custom machine learning models.

Much of the PAC approach described in this thesis takes after key components of Teachable Machine. PAC utilizes a transfer learning model to build machine learning

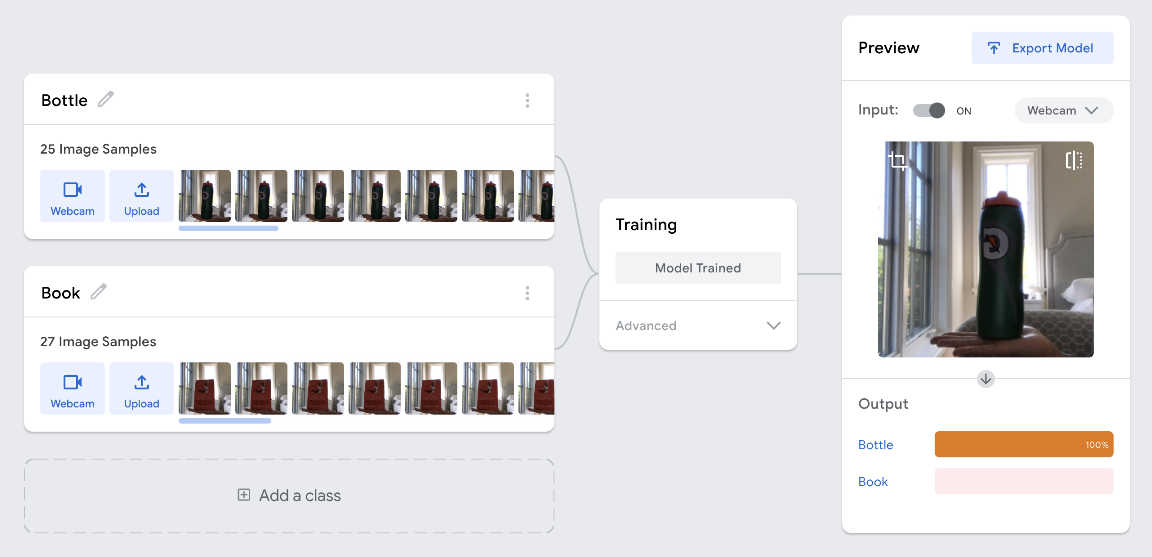


Figure 2-2: The Teachable Machine testing interface. The trained model shows webcam classification results live in the web browser.

functionality into a web application. Similarly, the ability to test and export a custom machine learning model allows for students to build a wide variety of applications.

2.2 Personal Image Classifier

The Personal Image Classifier (PIC) is a educational machine learning tool built by Danny Tang in the MIT App Inventor lab. The Personal Image Classifier is an important addition to the field of educational machine learning curriculum as it bridges the gap between Teachable Machine and MIT App Inventor while allowing users more machine learning customization. The user experience is as follows:

1. Training: Just like in Teachable Machine, users specify classification labels and upload or record images for each label, as seen in (Figure 2-3). After populating each label with image examples, PIC prompts users to train a custom transfer model, allowing them to specify hyperparameters and customize model layers (Figure 2-4).

2. Testing: After training the custom model, PIC provides an interface for webcam inputs and classifies the test images, showing the user classification confidences

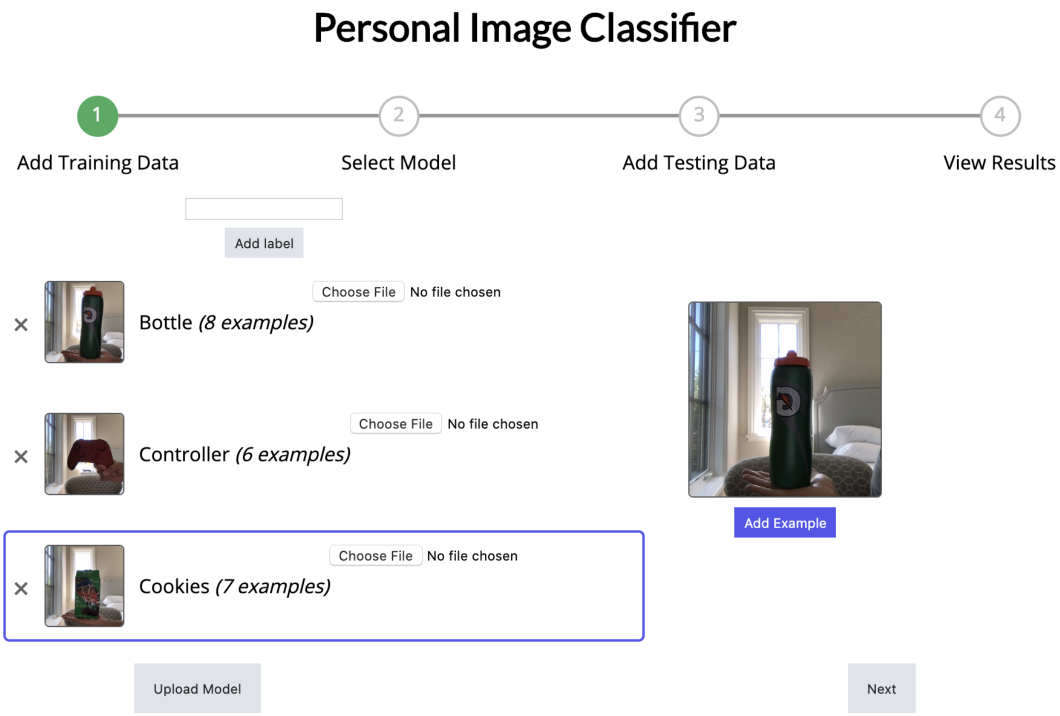


Figure 2-3: The PIC training interface. Users specify custom labels and record corresponding examples.

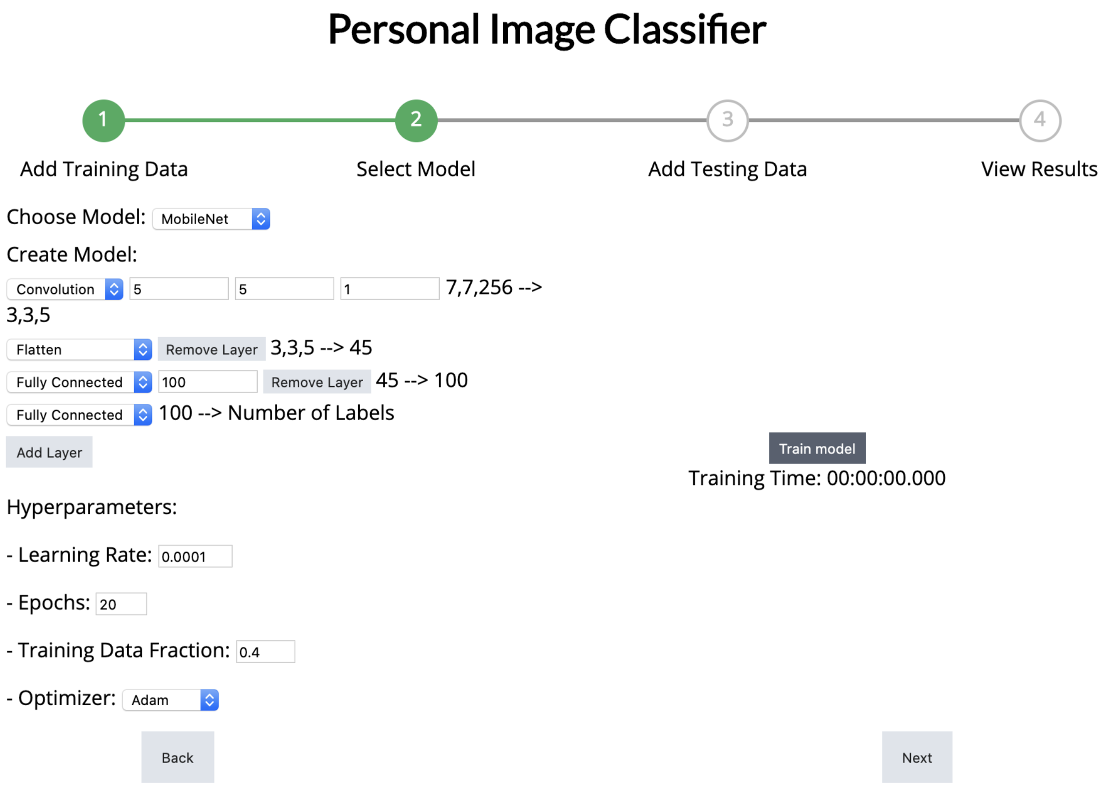
and test error metrics (Figure 2-5).

3. Export: After testing a custom PIC model, users can export their models for use in the MIT App Inventor PIC Extension. This allows for App Inventor applications to provide inputs to a pre-trained PIC model, and perform actions based on the model's output classifications.

In following this structure, PAC was built to provide users with a similar user experience for building custom audio classifiers, with an improved user interface and important input processing features prompted by the inherently inconsistent nature of audio recordings.

2.3 Spectrogram Audio Classification

In recent years, image classification has become one of the most prevalent applications of deep learning across industry and academia. The reasons for this include the prac-



Personal Image Classifier

Choose Model: MobileNet

Create Model:

Convolution 5 5 1 7,7,256 --> 3,3,5

Flatten Remove Layer 3,3,5 --> 45

Fully Connected 100 Remove Layer 45 --> 100

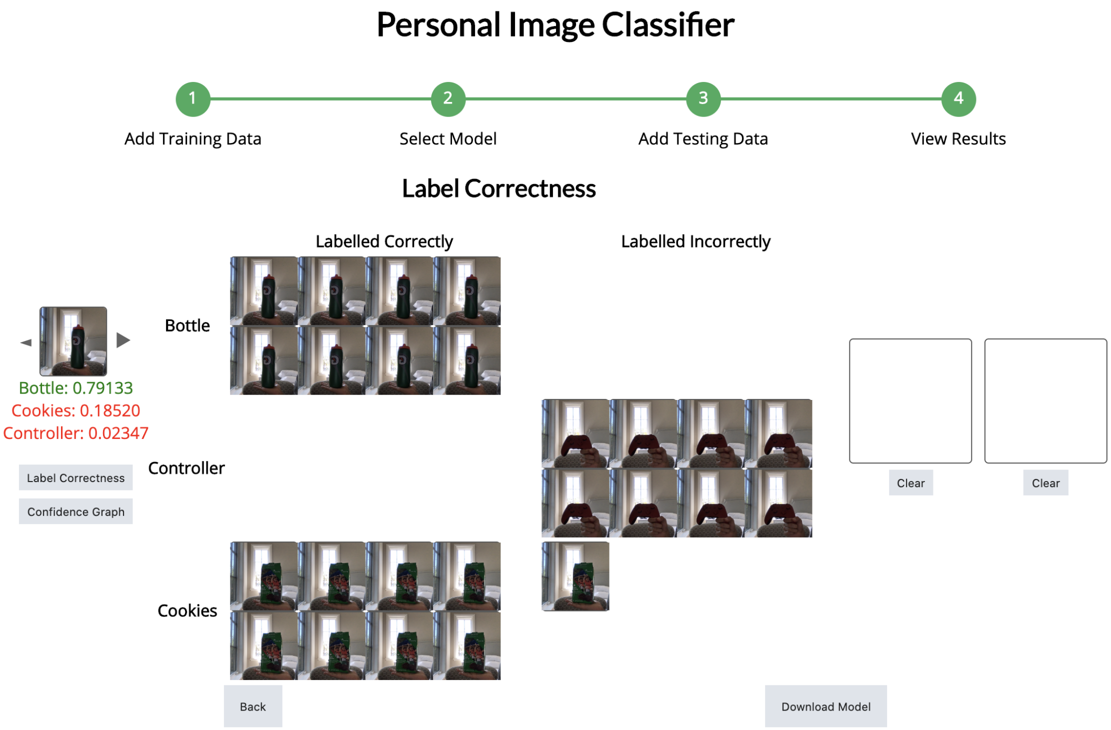
Fully Connected 100 --> Number of Labels

Training Time: 00:00:00.000

Hyperparameters:

- Learning Rate: 0.0001
- Epochs: 20
- Training Data Fraction: 0.4
- Optimizer: Adam

Figure 2-4: The PIC model interface. Users customize a variety of model hyperparameters, including layer details, training epochs, and training data fraction.



Bottle: 0.79133
Cookies: 0.18520
Controller: 0.02347

Figure 2-5: The PIC testing interface. Users can add images for each class, test their trained model, and view label correctness in the same interface.

tical applications of image classification algorithms in our every day life, from facial recognition to self driving cars, as well as the development of a clear representational understanding of images. Images are two-dimensional, with a possible third dimension for RGBA channels. Understanding how adjacent pixels relate to each other forms the basis for understanding an image. This insight has spurred the development of machine learning tools like filters and convolutional neural networks that provide researchers with the tools to build powerful image classification models. At first glance, it might seem that the audio format would provide audio classification models with similar advantages: there is a clear relationship between adjacent portions of an audio clip, and understanding these relationships can provide a broader understanding of the audio content. Unfortunately, the difficulties surrounding the audio format primarily lie in representation. The information that humans extract from audio clips is complex in that it spans a variety of audio characteristics, including amplitude, frequency, and time. This complexity is exacerbated by the fact that there is no obvious "winner" in terms of a representational format that can encompass all of these characteristics. In recent years, researchers have found that image representations of audio can incorporate many of these important characteristics, while allowing them to apply existing image classification tools to a novel classification problem. In the remainder of this section I will delve into the shortcomings of various audio representations and motivate the use of spectrogram images as an optimal representation of audio for the purposes of audio classification.

Take for example a typical WAV audio recording. Any such audio file has an associated sample rate, conveying the number of audio sample taken per second. In Figure 2-6 [7], we see a 3 second audio clip with a sample rate of $44,100$ Hz, implying $44,100 * 3 = 132,300$ consecutive changes in air pressure. This commonly utilized audio visualization, plotted using Python, depicts amplitude against time. Although this representation gives us important information about how loud or quiet an audio clip is, it tells us nothing about the actual audio content, thereby giving us very little representational value in the context of deep learning. In order to extract information about the audio content, we can compute the Fast Fourier Transform (FFT) of our audio clip, providing us with a visualization of the frequencies present in our audio. The FFT operation takes in a frequency bin size as input and produces a graph depicting how often each frequency occurs in the audio clip (Figure 2-7) [7]. While this new representation now includes information about audio frequencies, it introduces a new problem: the FFT operation deliberately converts an input from the time dimension to the frequency dimension, but the audio classification problem requires that our representation encapsulate some understanding of time; just as the human ear would not equate an audio clip played forwards with the same clip played backwards, our representation must inherently capture when the frequencies occur in our audio clip. To solve this, we can add a time dimension to our representation by repeatedly computing our FFT over short overlapping sample windows, and sliding this sample window across the audio file. This allows us to visualize how the frequencies in our audio file change over time. As a final addition to our audio visualization, seen in Figure 2-8 [7], we represent the amplitude of a particular frequency at a particular time

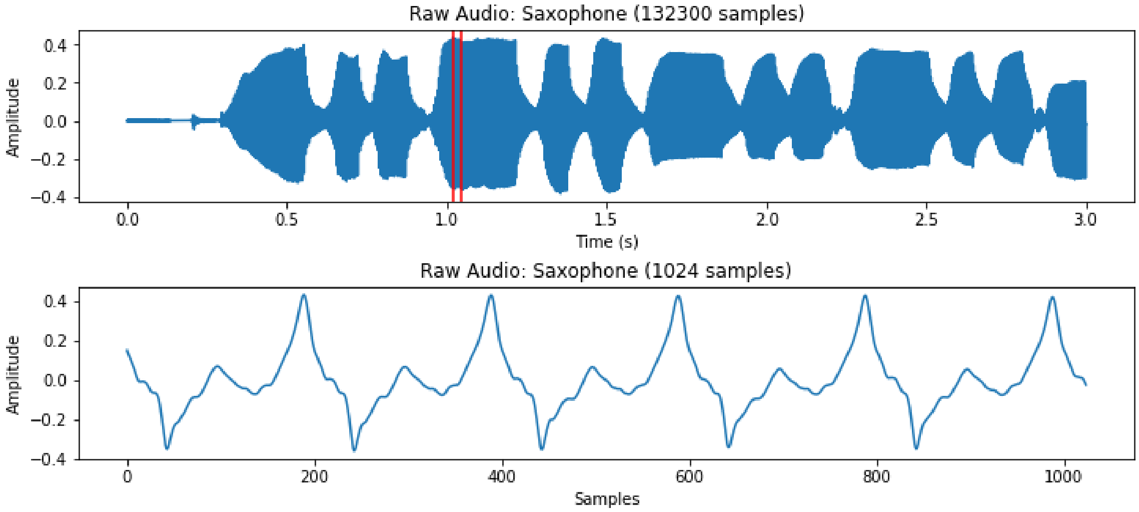


Figure 2-6: A standard audio visualization plotting amplitude against time for a three second saxophone clip.

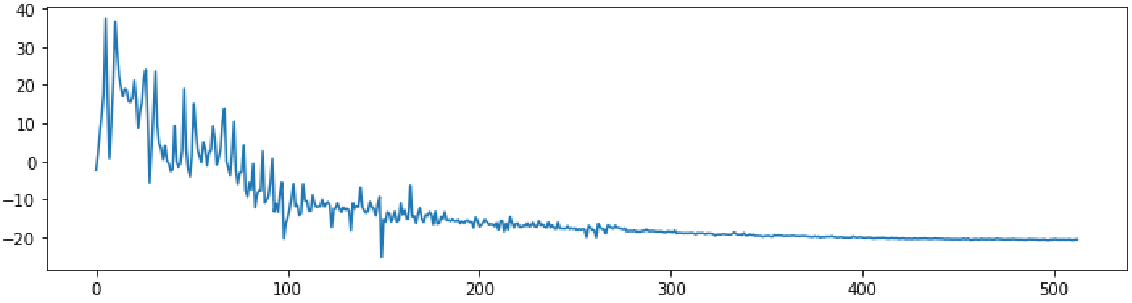


Figure 2-7: The frequency distribution from a Fast Fourier Transform for a three second saxophone clip.

using a third dimension, color, with darker colors corresponding to low amplitudes and lighter colors corresponding to progressively stronger amplitudes.

Our resulting audio representation, also known as a spectrogram, provides us with a comprehensive visualization that encapsulates frequency, time, and amplitude. Khamparia [8] and Boddapati [9] have showed that existing image classification techniques can achieve impressive results on spectrogram audio classification tasks with very few changes to typical convolutional model architectures. These results provide the basis for building an in-browser audio classifier using spectrograms as the audio representation.

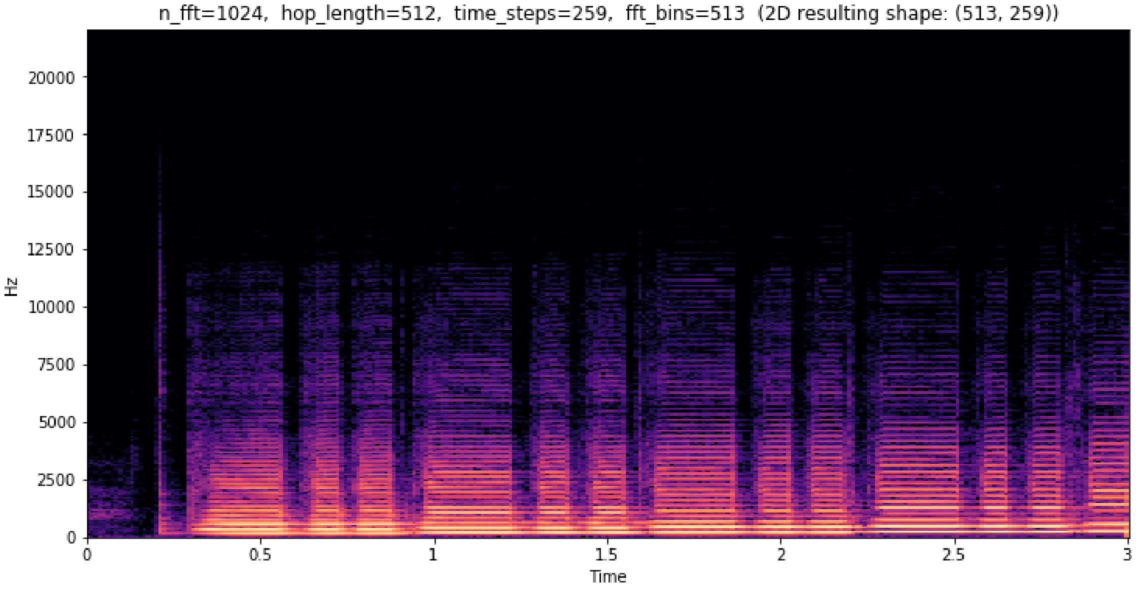


Figure 2-8: The final spectrogram output depicting frequency, amplitude, and time over a three second saxophone clip.

Chapter 3

The Personal Audio Classifier

While many existing in-browser tools have targeted the widely popular domain of image classification, less work has been done to make the field of audio classification more accessible to students and machine learning novices. PAC was built to offer a simple and accessible tool to empower those without significant machine learning knowledge to understand and utilize audio classification in their own research, projects, or applications. In this section I will delve into the design methodology, functionality, and UI behind PAC, detailing each user interaction step and delving into a variety of challenges I faced in building this system.

3.1 Technology Stack

The PAC front-end is built primarily in React, with custom components written in JavaScript. The front-end server is hosted via Express and Node. The spectrogram back-end is written in Python, and the back-end server is hosted via Gunicorn and Flask. Both servers are hosted on a Linux virtual machine owned by MIT CSAIL.

3.2 Design Tenants

In designing a flexible and intuitive interface for PAC, it was important to follow a number of important design tenants, inspired by interfaces like Teachable Machine and PIC. These include:

1. The interface should require no coding experience or prior machine learning knowledge.
2. The application should be modularized, with various steps of the machine learning process abstracted into different steps of the interface.
3. The application should still allow for significant customizability (from classification labels, to model hyperparameters), in order to cater to a variety of use cases.
4. The interface should have an easy learning curve, and be intuitive enough to allow users to start building their classifier immediately.
5. The application should be visually appealing and grab the attention of students and teachers.

3.3 Front-end Functionality

The front-end is broken up into four important pieces of functionality: training, model customization, testing, results. Each section provides users with a means of interacting with the machine learning development process without requiring any prior machine learning knowledge. In this section I will describe PAC's front-end functionality and user interface.

3.3.1 Training

On the initial landing page for PAC, users have the opportunity to begin building a custom audio classifier. The UI first shows the recording component in the left column where users will be able to record audio clips (Figure 3-1). All recording features are disabled until the user adds classification labels by clicking the animated "+" symbol. Upon clicking the "+" symbol, a popup prompts users to provide a name for the new label and creates an empty label in the right column of the UI (Figure 3-1). In terms of implementation, each label corresponds to an instance of a "Label" React component, with its user-specified name, and an initially empty dataset of training images. Each label has a number of functions that allow for spectrograms to be added to and removed from the dataset.

After adding at least one label, the recording component becomes enabled, allowing users to record audio clips for a label via a drop-down in the recording component (Figure 3-2). After clicking the record button, a waveform displays above the button, showing the device's input audio stream (Figure 3-2). After the recording button is pressed, the following steps occur:

1. A custom React library opens the device microphone for one second, recording the device's input audio as a WAV file.

2. The audio file is passed to the PAC back-end via an API call, where the file is converted to a spectrogram image.

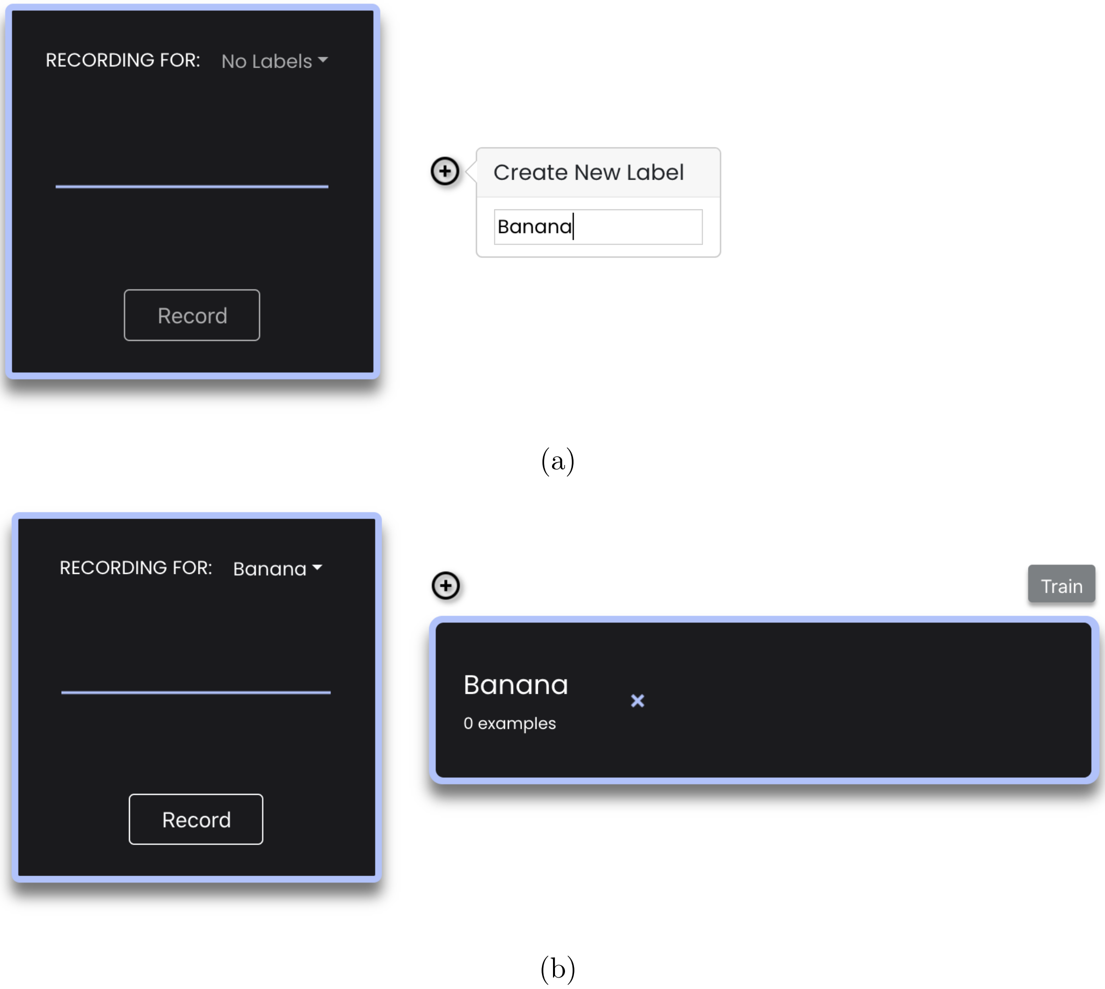
3. The spectrogram PNG is returned to the front-end and stored in memory in the label's training dataset via a temporary data url.

4. A preview of the audio spectrogram appears in the label's UI component and the indicator for the number of training samples is updated.

After recording at least two audio clips for each classification label, a "Train" button becomes enabled above the label view (Figure 3-2). Clicking this button displays the model customization UI, as detailed in the following section.

3.3.2 Model Customization

After clicking the "Train" button, a popup appears to users, allowing them to modify the hyper-parameters of a custom transfer model that will train to distinguish between the user-specified audio classes (FIG). PAC allows users to tweak their model in the following ways:



(a)

(b)

Figure 3-1: (a) The initial PAC web interface provides a recording interface and a button to add custom classification labels. (b) Upon entering a label name, a label is added to the right column of the PAC UI. Each label is associated with an initial empty dataset of training images.

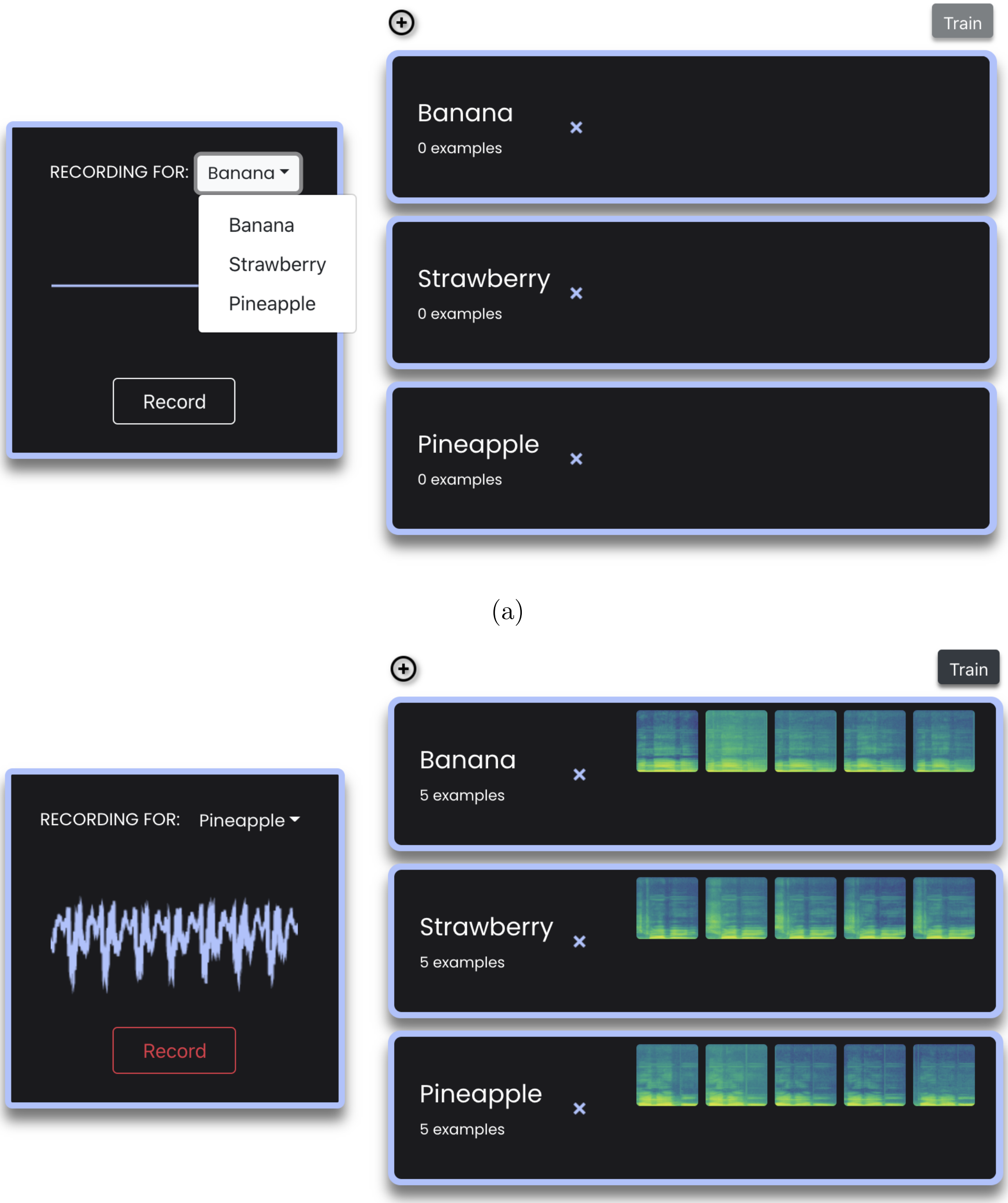


Figure 3-2: (a) After entering multiple classification labels, users can select a label from the dropdown for which the recording component will record audio clips. (b) After the "Record" button is clicked, the waveform visualizes device microphone input and records the next one second of audio. Audio clips are passed to the backend, returned as spectrogram images, and added the corresponding label training dataset. After at least two training examples are added to each label, the "Train" button becomes active, allowing users to train a neural network with their training data.

1. Learning Rate: a hyper-parameter that controls how much the weights of the neural network are adjusted with respect to the loss gradient.

2. Optimizer: a hyper-parameter that ties together the loss function and model parameters by updating the model in response to the output of the loss function in a specific way.

3. Epochs: the number of complete passes made through the training dataset.

4. Training Data Fraction: the fraction of the training dataset to use as the batch size.

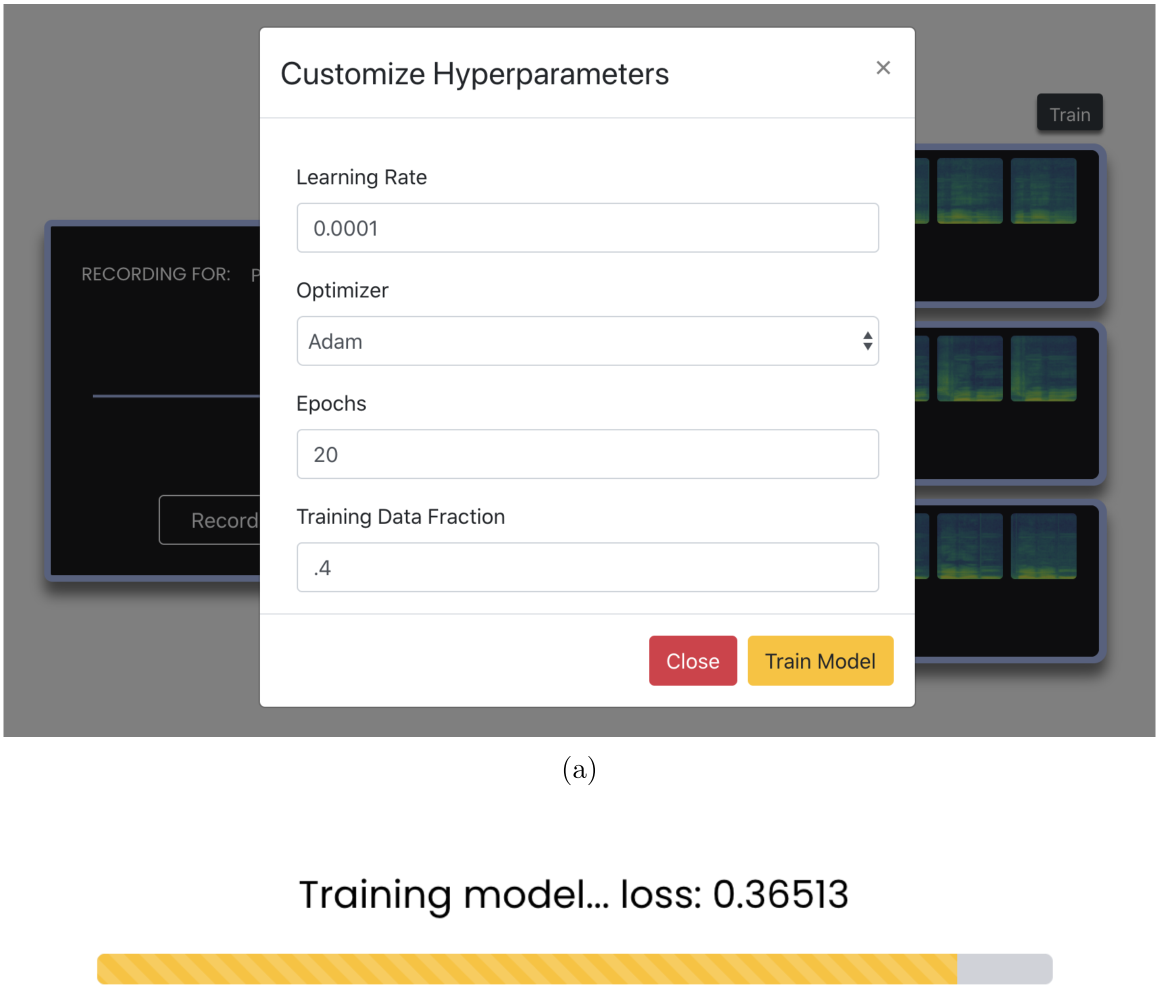
After users customize hyperparameters and press "Train Model," the training datasets is passed through a pre-trained MobileNet model. Then, the output is used to train a custom five-layer image classification model with the user-specified hyperparameters. The steps for this process are as follows:

1. Each spectrogram in each label dataset is retrieved via data url and converted to a tensor for use in TensorFlow.js.

2. The tensor is passed through a modified, pre-trained MobileNet neural network, and the image tensors are replaced by the corresponding activations from MobileNet's second-to-last layer.

3. A custom five-layer model is trained to the specifications supplied by the user, with the MobileNet activations as the input. This model is composed of a single convolutional layer, followed by a flatten layer, two fully connected layers, and a single softmax layer. This architecture is designed to output a standard tensor with probabilities distributed across the various user-specified labels, where a given probability corresponds to the probability that the input spectrogram belongs to that label.

While the model begins to train, the PAC UI shows an interim training screen with continuously updating model loss values. After the training is completed, users are immediately brought to the testing screen.



Customize Hyperparameters ×

Learning Rate: 0.0001

Optimizer: Adam

Epochs: 20

Training Data Fraction: .4

(a)

Training model... loss: 0.36513

(b)

Figure 3-3: (a) The model customization popup. Users can modify hyper-parameters for their custom model as they see fit. (b) An interim loading screen shows training progress as well as the current model loss.

3.3.3 Testing

On the testing screen, users can record test audio clips and see how their trained model classifies recordings. The UI displays the same recording component in the left column, a spectrogram visualizer in the middle column, and a list of the user-specified labels in the right column (Figure 3-4). Users interact with the "Record" button in the same way as they did on the training screen, recording one second audio clips that are converted to spectrograms on the backend and displayed in the spectrogram viewer. After the front-end receives the corresponding audio spectrogram from the back-end, the following steps occur:

1. The spectrogram is passed through the same pre-trained MobileNet model used in the training process.

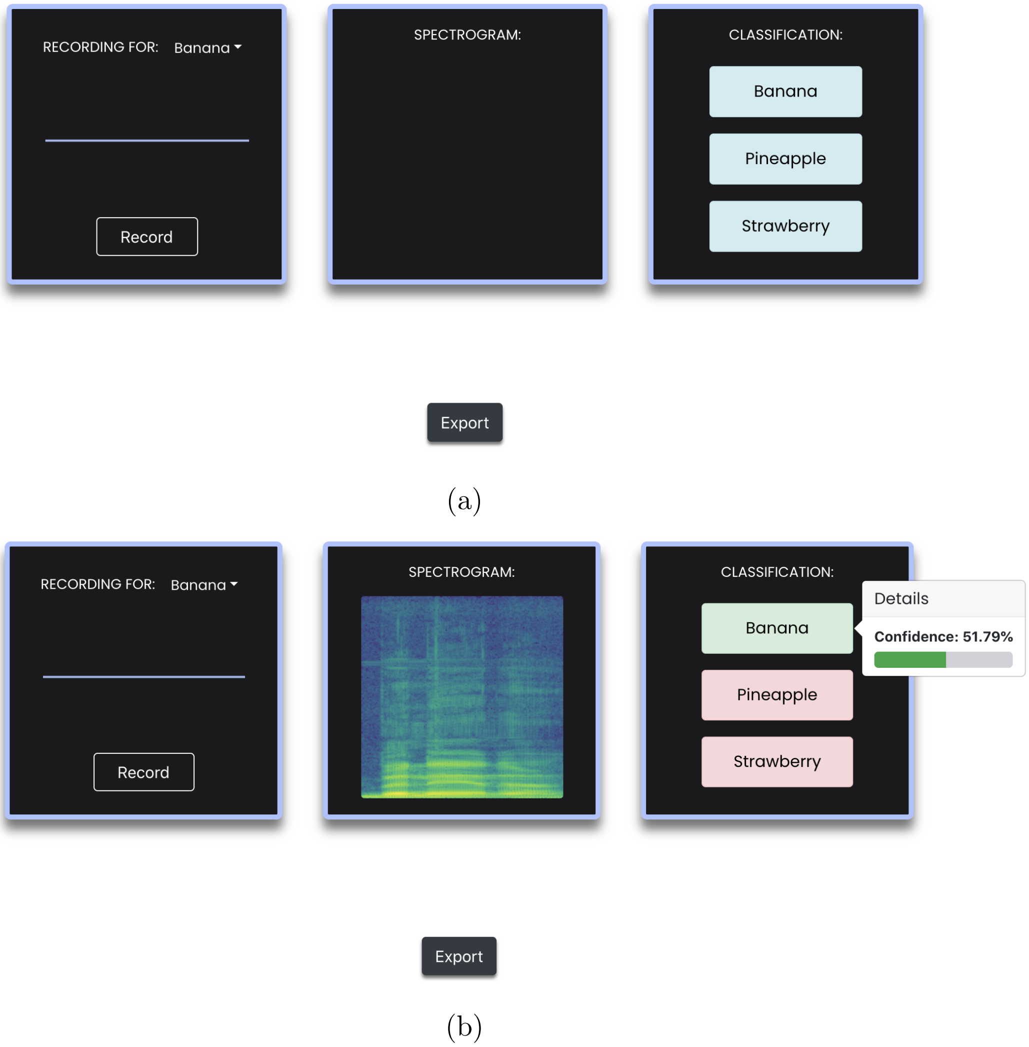
2. The resulting activations are passed through the trained user model; the softmax layer outputs an array of probabilities that correspond to the array of labels, such that the probabilities sum to one.

3. The output probabilities are assigned to the labels and the label with highest probability is chosen as the "winning" classification.

The UI proceeds to highlight the "winning" label in green, with all other labels highlighted in red. Users can hover over each label to view the model's corresponding output probability (Figure 3-4). Scrolling down below the main view brings the user to the "Test Results" view.

3.3.4 Results and Exporting

In the "Test Results" view, users can visualize the results of their test audio clips. In this view, spectrograms are organized by model classification and their corresponding confidence is shown under each image. Correctly classified images are shown with green confidence bars, while incorrectly classified images are shown with red confidence bars. Users can use these results to gain a better understanding of the features



(a)

(b)

Figure 3-4: (a) The initial testing view is made up of a recording component, a spectrogram visualizer component, and a label view component. (b) Users record audio clips that are then passed through the trained model. The model outputs probabilities associated with each label and visually highlights the "winning" classification. Users can hover over labels to see classification confidences.

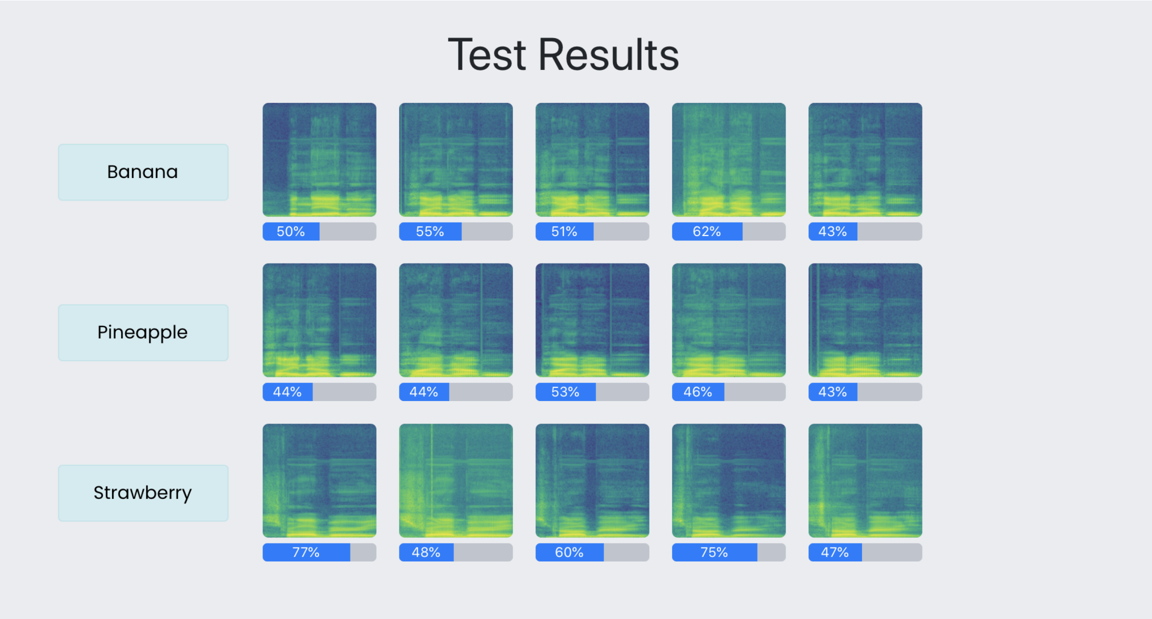


Figure 3-5: The PAC test results interface. Test audio clips are organized by their corresponding classification.

in spectrograms that might lead to correct and incorrect classifications, and distinguish between higher and lower confidence spectrogram classifications. Users have the ability to return to the training screen at any time via the navigation bar in order to improve their model, either by removing suspect training examples, or by adding new training examples based on the insights they glean from the test results.

As seen in Figure 3-4, the PAC testing interface also allows users to export their trained model for use in external applications. Users can click the "Export" button to download a zip file with the following contents:

1. A model topology file detailing the architecture of the pre-trained MobileNet model.

2. A weights file detailing the weights of the pre-trained MobileNet model.

3. A JSON file mapping the model's predictions indices to their corresponding label names.

3.4 Back-end Functionality

The PAC back-end is responsible for receiving WAV audio files through a Flask API, performing a variety of audio modifications, and creating a spectrogram image from the audio recording. This section will detail the audio manipulation techniques used as well as the spectrogram conversion process.

3.4.1 Flask API

The back-end is accessible via a Python Flask API, served from a Linux virtual machine running on an MIT CSAIL server. The server listens via a "/spectrogram" POST endpoint on the 5000 port. This endpoint receives requests from the front-end, unpacks the request body into a local WAV file, and calls a helper function to perform the audio modification and spectrogram conversion.

3.4.2 Spectrogram Conversion

After the request body is unpacked into a local WAV file, the back-end extracts the left audio channel from the stereo signal. Then, a helper function removes beginning or ending silence in the audio clip by iterating over all audio chunks in the recording and removing the first and last continuous audio segments that register below -20 decibels. This feature was added after extensive user testing showed that user input recordings were rarely centered in the recorded audio clip. If a user were to record two identical word samples, the resulting spectrograms might look different if the user were to start speaking the word during different segments of the recording. By removing any leading and lagging silence in the input recording, we ensure that the user's input audio is centered in the recording.

Next, the back-end splits the audio recording into 10ms chunks and removes any remaining chunks that register audio below -50dB. User testing showed that this feature made PAC models far more flexible and accurate across a variety of use cases. The intuition for this is that various users might record the same phrase, word, or number, but differences in pronunciation speed might lead to small differences in the spectrogram output. By removing naturally occurring silence in input audio clips, we can better standardize recordings to contain only audio content that will reflect meaningful frequency information in our spectrogram outputs. It is important to note that we apply a more strict silence requirement for chunks in the middle of the audio recording to ensure that important audio content is not significantly altered.

Lastly, the processed audio recording is passed through a spectrogram conversion function. This function splits the audio data into segments of length equal to the audio recording's sample rate, and computes the frequency spectrum for each section. A windowing function is then applied to each audio segment to connect the audio segments into a time dependent color-map (FIG).

3.5 Challenges

During the development of PAC, I encountered a number of unique challenges related to the world of audio processing and spectrogram audio conversion. In this section I will detail these challenges and how they influenced PAC's implementation.

3.5.1 Online vs Offline

One of the challenges of implementing PAC was thinking about whether the spectrogram conversion process should be online or offline. My prior discussion of PAC has described an online implementation: PAC records audio clips and makes requests to a back-end server that handles the audio to spectrogram conversion process. The side-effect of an online implementation is that any application that wants to use an exported PAC model must also use the same spectrogram conversion process, thereby requiring the application have access to the internet so that requests can be made to the PAC back-end. An offline implementation of PAC would necessitate that the spectrogram conversion process occur entirely client-side (in JavaScript). Thus, any application that utilizes a model exported from PAC can run offline, making use of the same client-side spectrogram conversion process used by PAC.

Even though the final implementation of PAC was online, I heavily explored an

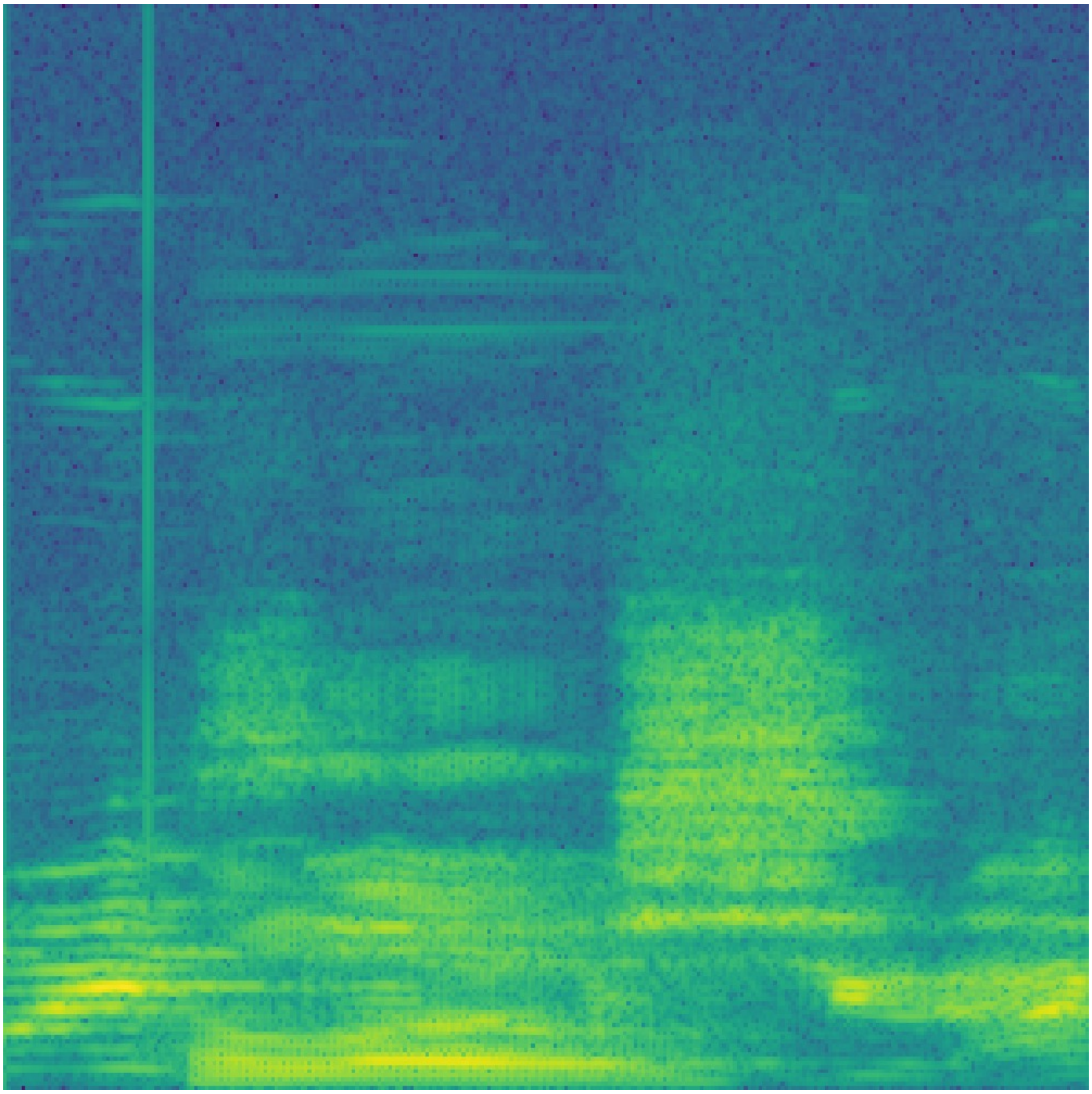


Figure 3-6: An example spectrogram generated in Python for a one second audio recording.

offline implementation to provide application developers with more flexible offline use cases for PAC models. In building out a spectrogram conversion process in JavaScript, I had to implement complicated audio processing functions in a language with very limited audio support. Most audio techniques in JavaScript process audio in buffers, giving you only temporary access to a stream of the audio input. This provides limited ability to generate a spectrogram image in a single operation. In order to implement spectrogram conversion in this form, I modified an existing streaming spectrogram generation library to perform the required FFT operations on each element of the audio buffer, iteratively generating a spectrogram output pixel by pixel. Due to JavaScript's limitations on sampling rate and lack of support for more complex spectrogram libraries, the resulting spectrograms were far less detailed and conveyed a more limited spectrum of frequencies (Figure 3-8). In analyzing the viability of these offline spectrograms, I compare them to their online equivalents across three use cases in order to analyze their effectiveness:

1. Word Classification: In this use case, PAC models are trained to distinguish between different spoken words. The offline spectrograms performance is comparable to that of the online spectrograms, however I discovered that the offline models are far less flexible, requiring that the same voice speak every training example.

2. Voice Classification: In this use case, PAC models are trained to distinguish between various human voices by training on different users speaking the word "Hello." I found that offline spectrograms were almost entirely capable of supporting this use case, where as online spectrograms could guarantee 80%+ classification accuracy for a well-trained voice classification model.

3. Mobile Classification: In this use case, PAC models are trained on the prior two use cases on a mobile device. The challenge of this use case is that mobile hardware encompasses a wide range of low to medium quality microphones, often providing much lower sampling rates for audio recordings. It is in this use cases that the limitations of offline spectrograms become apparent. While these

spectrograms provide enough unique information to build passable classifiers when using high quality computer microphones, any such model tested on a mobile device has a near random success rate.

After performing this analysis, I decided to pursue an online PAC implementation. In following my design tenants, I wanted to give users the ability to build powerful audio classification models across the widest variety of use cases. This would facilitate more opportunities to build interesting projects, curricula, and classroom discussions for the small price of requiring that devices have internet connection.

3.5.2 Silence Thresholds

Another important challenge that influenced the development of audio processing features was dealing with inevitable human error when recording audio clips. While webcam interfaces like PIC can show the user the exact frame they will capture, an audio interface like PAC must give the user a window during which they must record their audio content. This results in unavoidable variability when interacting with the recording interface, including situations where:

1. Users begin to speak prior to the microphone recording audio.
2. Users continue to speak after the microphone has finished recording.
3. Users record audio content in different periods of the recording window.
4. Users speed up or slow down pronunciation in an attempt to change an incorrect classification.
5. Different voices are used to train the same model.

In combating these many sources of error, I explored a number of sound processing techniques to mitigate the variability present in recorded audio clips. After extensive experimentation, it became clear that removing silence at the beginning and end of audio clips could help combat variability resulting from users speaking at different

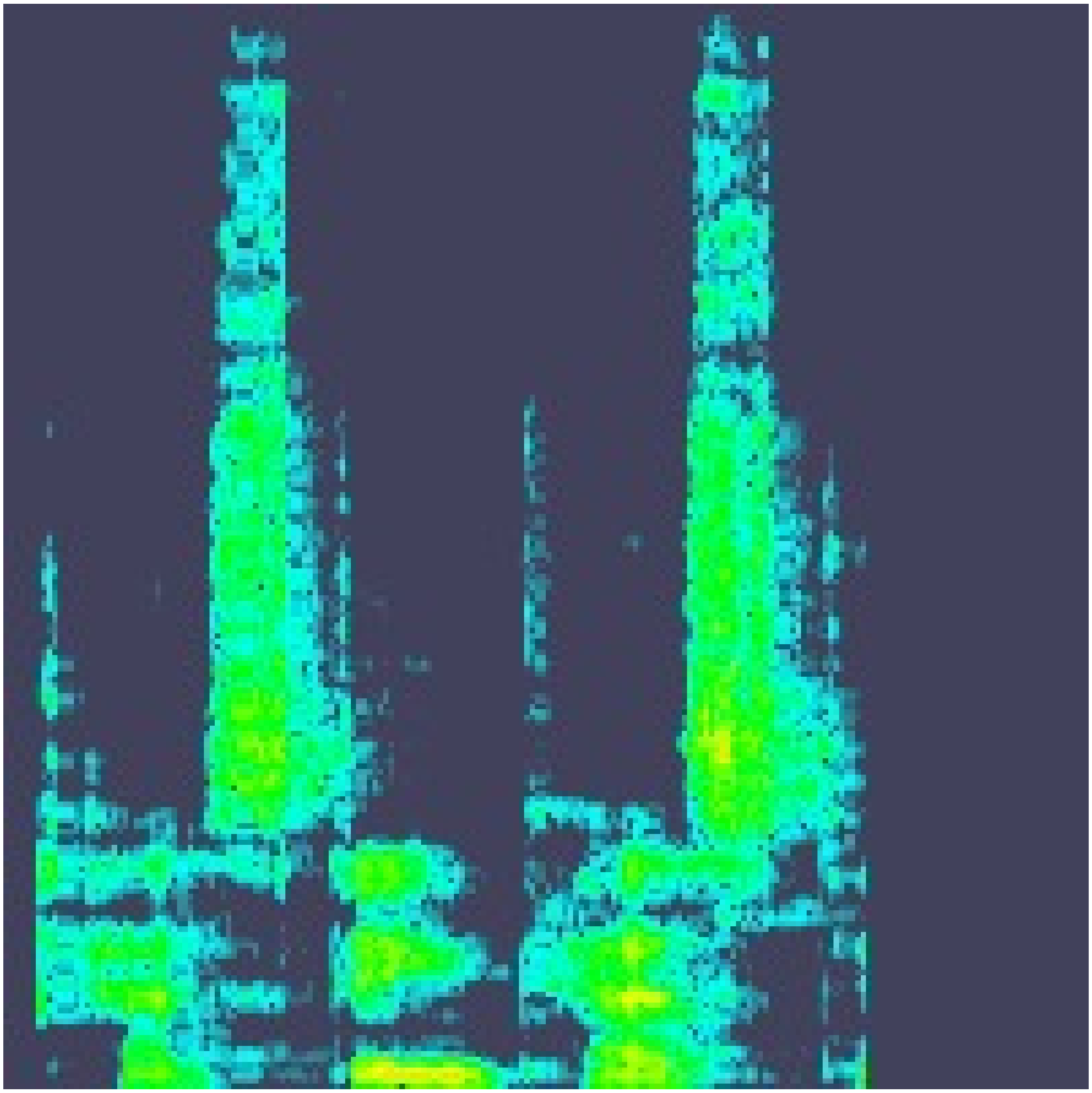


Figure 3-7: An example spectrogram generated in JavaScript for a one second audio recording.

times during the recording window. This would center the important sound information and lead to similar spectrograms for similar audio clips. To help counter the other sources of error, I implemented a similar silence removal technique for the remaining centered audio content. This algorithm searches for 10ms windows of silence and removes them to reduce variability in user intonation and pronunciation speed.

In implementing both these techniques, it became clear that the threshold for silence would be a vital parameter to get right in order to ensure that I was maximizing audio processing impact without fundamentally changing the audio content. Below are the three silence threshold options I explored, each expressed in relativity to full scale (0 dB, or the loudest possible volume in the audio clip):

1. -50dB: This is the most stringent silence threshold. Given the logarithmic decibel scale, this is equivalent to $1/10^5$th of full volume.
2. -20dB: A soft silence threshold, equivalent to 1/100th of full volume.
3. -10dB: The most lenient threshold, equivalent to 1/10th of full volume.

In this analysis, "fringe" silence refers to silence found at the beginning and end of a recorded clip, while "central" silence refers to silence present in the user's actual audio input. Initially, it was not obvious that different silence thresholds might be required for these different types of silence. A deeper analysis of audio clips recorded from a number of different models showed that the audio chunks that preceded and followed the clip's central audio content were often completely silent. This meant the most strict silence threshold of -50dB would still be able to catch fringe silence while ensuring no important audio content was accidentally removed by a more lenient silence threshold. Unfortunately, this same threshold did very little to remove central silence. An analysis of 50 recorded audio clips, processed with a -50dB silence threshold, showed a mere 2% reduction in total central audio length; there was clearly still silence present in the central audio content that was not being removed at the -50dB threshold. By separating the two silence removal use cases, I was able to specify unique thresholds for fringe and central silence. Increasing the threshold for central silence to -10dB led to a 18% reduction in central audio content across the same 50 recorded audio clips, and noticeable qualitative improvements in model classification accuracy for word and mobile classification use cases. Unfortunately, a -10dB threshold led to incredibly poor performance on voice classification models, likely due to the removal of all unique voice characteristics under an incredibly lenient silence threshold. By decreasing this threshold to -20dB, we see a 6% reduction in central audio content across the same 50 test audio clips, and better performance across every type of audio classification model. The final silence removal algorithm used a -50dB threshold for fringe silence, and -20dB for central silence. By exploring unique audio processing techniques and comparing their effectiveness, I was able to overcome a number of human and input error challenges that are unique to audio classification.

Chapter 4

The Personal Audio Classifier Extension

PAC allows machine learning novices to explore audio classifiers and basic machine learning concepts in their web browser. While students' exploration of PAC can already spawn a number of interesting discussions about how neural networks perform impressive image and audio classification tasks, I hope to give students the tools to build custom applications so that they can further explore the real-life applications of this machine learning functionality. This section will introduce the PAC App Inventor extension, a tool that integrates with MIT App Inventor to provide users with the ability to perform on-device audio classification by integrating pre-trained PAC models into their own applications.

4.1 Design Requirements

All MIT App Inventor extensions are composed of a core Java file, and any number of optional external assets. The Java file defines the main functionality of the extension and defines the properties and blocks that users can interact with. The external assets often include JavaScript files to allow for TensorFlow.js code to run in a custom App Inventor Web Viewer. Extensions can also include other ancillary machine learning files and make external requests assuming the device is connected to the internet.

The goal of the PAC extension is to allow users to build App Inventor applications that can take in a pre-trained PAC model and make predictions using audio recorded from the device. The important feature requirements for the PAC extension are as follows:

1. Allow users to load in a pre-trained, exported PAC model.
2. Load in MobileNet to provide activations prior to making predictions.
3. Provide an interface for users to record audio clips.
4. Display model predictions and execute user-defined application functionality based on model results.

4.2 Machine Learning Functionality

In order to load a MobileNet and PAC model into our extension, we can utilize TensorFlow.js to implement the same in-browser machine learning functionality found in PAC. Yet rather than running this JavaScript functionality in a mobile device's web browser, App Inventor offers a WebViewer component that can run JavaScript code from within an App Inventor application.

To provide our extension's TensorFlow.js code with the same MobileNet model used by PAC, I shard this model across a number of weights files and include these files in the extension's assets for offline access. To allow users to import their own models for use in the extension, I provide support for reading in ".mdl" files exported from the PAC interface. After the file is uploaded by the user via the PAC extension properties, the file path is read using a Java zip reader.

With the MobileNet and PAC models now accessible within the extension, we make a number of TensorFlow.js requests from the WebViewer. Each request is handled in the core Java client, allowing JavaScript code to access assets provided by the user as well as those included in the extension backend. The following requests are required before the PAC extension is ready to make predictions:

1. Load the sharded pieces of the MobileNet transfer model from the assets folder.
2. Retrieve the model topology and weights file for the user's PAC model.
3. Retrieve the mapping of model outputs to label names included in the exported PAC model.

After retrieving these resources, the extension is ready to classify audio clips. But first, we must provide App Inventor applications with the ability to record custom audio clips without extra libraries or components. In order to do so, we once again utilize the powerful WebViewer component. The WebViewer is important not only for making back-end TensorFlow.js requests, but also for providing a programmable visual interface that users can interact with. With a simple HTML/JavaScript/CSS implementation, we provide an audio recording interface for users to record and play back audio clips (FIG). After an audio clip is recorded, the extension makes another POST request to the PAC backend. The returned result is a spectrogram that is ready to be classified. The spectrogram is automatically passed through the MobileNet and PAC models, and the resulting classification is exposed to the application via the App Inventor Blocks interface (FIG).

With the aforementioned features, users can integrate the PAC extension into any existing application to record audio clips and build application functionality based on the model results.

4.3 Extension Blocks and Properties

The Personal Audio Classifier Extension has two designer properties and four blocks. The designer properties allow users to specify inputs to the extension and are accessible via the "Designer" tab in App Inventor. Blocks allow users to specify interactions and functionality for their application and are available within the "Blocks" tab in App Inventor. The available properties and blocks are listed below (Table 4-1).

	Block	Description

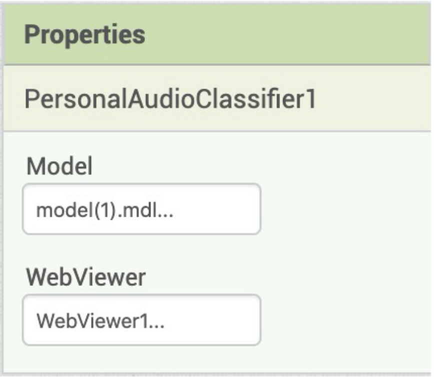
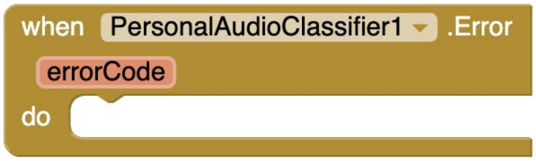
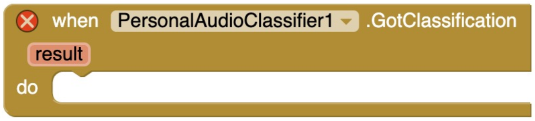
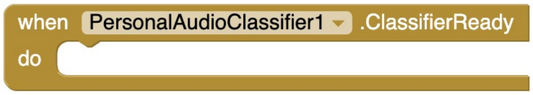
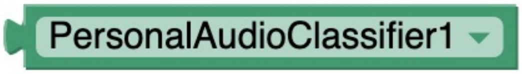
1.	 Properties PersonalAudioClassifier1 Model model(1).mdl... WebViewer WebViewer1...	The *Model* designer property allows users to upload and select a model exported from the PAC interface to use in making predictions. The *WebViewer* property allows users to specify a WebViewer component to use with the extension.
2.	 when PersonalAudioClassifier1 .Error errorCode do	The *Error* block triggers when an error is encountered in the extension, and returns the code associated with that error.
3.	 when PersonalAudioClassifier1 .GotClassification result do	The *GotClassification* block is triggered after an audio clip is recorded and passed through the PAC model. The resulting classification output (the *sound* parameter) is returned in a list of lists with the top three prediction results.
4.	 when PersonalAudioClassifier1 .ClassifierReady do	The *ClassifierReady* block is triggered when the extension has finished loading the MobileNet and PAC models.
5.	 PersonalAudioClassifier1	The *PersonalAudioClassifier* block returns a specific extension instance.

Table 4.1: A description of the designer properties and blocks available in the PAC extension.

Chapter 5

PAC Workshop

5.1 Purpose

In order to evaluate the effectiveness of PAC and the PAC extension as a learning tool for machine learning novices, it was important that I utilize the tools in a classroom of students with limited machine learning experience. In pursuing this goal, I designed and led six workshop sessions with three AP Computer Science Principles (CSP) students at the Boston Latin Academy. Each class period lasted 55 minutes, and each CSP student engaged with the workshop material across two sessions. The Boston Latin Academy classes provided a perfect workshop environment to test PAC and its related tools as student had prior experience using App Inventor tools, but had limited exposure to machine learning. My curriculum was designed around PAC and the PAC extension and focused on conveying important concepts related to artificial intelligence, image classification, and audio classification. Hands-on activities were designed to engage students with in-browser neural networks and discussion topics were chosen to encourage students to think about the real-life implications of machine learning.

Class	Total Students	18+ Students
APCSP1	21	8
APCSP2	29	9
APCSP3	30	11
Total	80	28

Table 5.1: Student breakdown for all three Boston Latin Academy AP CSP classes.

5.2 Data Collection

The workshop study was conducted through the MIT Committee on on the Use of Humans as Experimental Subjects (COUHES). Prior to the workshop, MIT App Inventor and the Boston Latin Academy Research Team reached an agreement for me to collect and report anonymized data for all 18+ members of the AP CSP classes. Data from younger students was used to provide general context for the workshop's results, and all personalized data was discarded after the study's completion. An age breakdown of the three Boston Latin Academy AP CSP classes can be found in Table 5-1.

Data from the workshop was gathered via a pre-workshop and post-workshop questionnaire. The pre-workshop questionnaire was composed of 15 questions, targeted at gaining an understanding of students' current level of familiarity with workshop concepts, and was given to students to fill out prior to the start of the workshop. The post-workshop questionnaire was composed of 29 questions, evaluating changes in students' machine learning understanding and the effectiveness of the teaching tools, and was given to students to fill out after the completion of the workshop. Both questionnaires posed questions that fit into one of the following categories.

1. Basic understanding of fundamental machine learning concepts.

2. Awareness surrounding machine learning applications in the real world.

3. Effectiveness of the in-browser PAC machine learning tool.

4. Effectiveness of the PAC extension and MIT App Inventor applications.

5.3 Curriculum

Each of the three AP CSP classes participated in the study over two workshop session. In the first workshop session, the primary goal is to introduce students to fundamental machine learning ideas, and encourage them to think critically about the implications of machine learning in their every day lives. Students start by discussing the features of an intelligent being, and agree as a class on a definition for artificial intelligence. Students then test this definition and discuss whether various pieces of technology fall under the classification of artificially intelligent. The first workshop also introduces the first in-browser machine learning tool, Google Quick Draw, and gives students the opportunity to explore doodle classification and learn more about how models utilize large data-sets to recognize important patterns. Workshop exercises are accompanied by class discussions, slideshows, and small breakout activities. The full first workshop lesson plan can be found in Table 5-2, and the accompanying workshops materials can be found in Appendix B.

In the second workshop, we recap the lessons taken from Google Quick Draw, and pivot to understanding the building blocks for image and audio classification. Students learn the basic inputs and outputs of machine learning models, and understand the differences between training and testing. Much of the second workshop focuses on students exploring PAC and using what they've learned to build effective audio classification models. After exploring three unique types of PAC models, students export their models and use the PAC extension to plug them into an App Inventor application. This pre-built Voice Authentication Diary allows students to import their PAC models and add custom block logic. The second workshop concludes with a discussion of students' successes and failures during the workshop, as well as a demonstration of any students' unique application logic. The full workshop curriculum can be found in Table 5-3, and the accompanying workshop materials can be found in Appendix B.

Time	Activity
10 min	Introductions, explanation of MIT App Inventor and purpose of study, pre-workshop questionnaire.
10 min	Introduce CSP students to Artificial Intelligence. What is it? What does it mean? Exercises: • Explore what it means to be artificial. • Deciding what makes a human, animal, or object intelligent. • Defining artificial intelligence as a class.
15 min	Explore the ubiquity of machine learning tools in our everyday lives. Exercises: • Brainstorm examples of AI in our lives. • Segment examples by understanding vs. learning vs. planning. • Play an "Is this AI?" game as a class.
20 min	Interactive introduction to practical machine learning using Google Quick Draw. Exercises: • Explore the interactive Quick Draw game. • Discuss how Quick Draw uses data and AI. • Discuss sources of bias in Quick Draw and how they manifest.

Table 5.2: First workshop curriculum.

Time	Activity
10 min	Understand the building blocks of machine learning. How exactly does it work? Exercises: • Understand the inputs and outputs in machine learning. • Use Quick Draw to discuss training and testing. • Introduce other relevant types of learning: image classification, audio classification.
20 min	Explore audio classification as a learning tool through the Personal Audio Classifier. Exercises: • Number classification: build a model that can distinguish between students speaking three different numbers. • Free classification: explore PAC, build a unique model that can distinguish between a variety of words or sounds. • Voice classification: students pair up, train a model to recognize each student's voice.
15 min	Build and explore the Voice Diary App Inventor application. Exercises: • Export PAC voice classification model for use in the Voice Diary app. • Download the Voice Diary app, set up the PAC extension using a custom PAC model. • Test model and application functionality, add custom block logic.
10 min	Concluding remarks, questions, post-workshop questionnaire.

Table 5.3: Second workshop curriculum.

Chapter 6

Results and Discussion

During the six workshop sessions, I attempted to use PAC and App Inventor to introduce students to basic concepts related to artificial intelligence and machine learning. In order to evaluate the success of these workshops, the pre-workshop and post-workshop questionnaire aimed to assess the following questions:

1. Did the workshops improve students understanding of basic concepts related to artificial intelligence and machine learning?

2. Did the workshops help students better identify examples of artificial intelligence in their every day life as well as envision impactful applications of machine learning?

3. Were PAC, the PAC extension, and MIT App Inventor effective teaching tools?

Both questionnaires posed a mixture of multiple choice and free response prompts to provide insight into these guiding questions. The full questionaires can be found in Appendix B. While questionnaire responses were collected from all 80 students, only responses and data collected from the 28 18+ students will be analyzed and reported in this thesis. The remaining 52 students' responses were used to provide general sentiment around workshop and teaching success.

6.1 Workshop Results

6.1.1 Prior Machine Learning Experience

In the pre-workshop questionnaire, we asked the students the following question:

> *"What exposure have you had with machine learning in the past?"*
>
> 1. *I've heard of it.*
> 2. *I am reasonably confident that I know what it is.*
> 3. *I've used something that involves machine learning.*
> 4. *I've created something that involves machine learning.*

Students were allowed to select all answers that applied. Out of the 28 18+ students, 25 selected choice (1), 8 selected choice (2), 12 selected choice (3), and 1 selected choice (4). These responses were in line with what we expected for the workshop demographic, with the majority of students having heard of machine learning, but less than a third of students confident that they know what it is. Out of the 12 students that selected choice (3), only 6 of them also selected choices (1) and (2). This seems to reflect the idea that students are aware that machine learning has permeated many aspects of their every day lives, but have a limited understanding of how machine learning works and why it continues to appear in new technology.

6.1.2 Basic Understanding of Machine Learning

In order to evaluate students' understanding of machine learning before and after the workshop, I pose the following open-ended question in both the pre-workshop and post-workshop questionnaire.

> *"How would you explain machine learning to a friend who has never heard of it before?"*

In the pre-workshop questionnaire, students took a stab at describing machine learning based on their limited interactions with the concept. Answers primarily described machine learning as a method of emulating human intelligence, with 18 student responses falling into this category. The 6 remaining non-empty responses generally characterize machine learning as teaching machines to learn with the use of extensive amounts of data. Within these 6 responses, only 2 answers seemed to show understanding of how data is used by machine learning models to learn: *"Training a machine to recognize patterns by feeding it lots of data"*, and *"Using data that is provided to machines in a repeated way to teach it to do specific tasks*. Besides these two responses, the majority of students demonstrate a limited understanding of what machine learning is and how it works prior to the start of the workshop.

The post-workshop questionnaire posed the same question, with significantly different results. Only 3 responses mentioned human intelligence, while 19 responses mentioned the use of data in machine learning. The remaining 6 responses discussed training, testing, or the iterative nature of machine learning. In general, responses from the pre-workshop questionnaire described machine learning in a more general, all-encompassing way, while responses from the post-workshop questionnaire commonly cited more specific tasks like classification, training, and testing. 9 of the 29 post-workshop responses referenced either Google's Quick Draw, Teachable Machine, or PAC to describe how a machine learning model works, using examples like *"imagine teaching a computer to recognize doodles by showing it millions of doodle examples"*, or *"like learning to recognize voices from audio clips"*. This seems to reflect an improved understanding of what machine learning is and how it works, as students cite specific concepts and use them to describe machine learning functionality.

Analyzing the differences between the pre-workshop and post-workshop responses shows an improvement in understanding of basic machine learning concepts. Every student who either couldn't define machine learning or defined it in terms of emulating human intelligence was able to generally describe the process by which models use data to create representations and learn from patterns. Almost a third of the class was able to cite specific exercises from the workshop to help them define machine learning to a friend, and well over half of students mentioned the importance of data to machine learning in the post-workshop questionnaire.

6.1.3 Ability to Discuss Machine Learning

In order the evaluate students' confidence in their own ability to describe machine learning to others, I posed two multiple choice questions in both the pre-workshop and post-workshop questionnaires:

> *"How confident do you feel about explaining or discussing machine learning with a non-technical person? (Please check one)"*
>
> - *0 = No confidence*
> - *1 = Slight confidence*
> - *2 = Moderate confidence*
> - *3 = High confidence*

> *"How confident do you feel about explaining or discussing machine learning with a machine learning expert? (Please check one)"*
>
> - *0 = No confidence*
> - *1 = Slight confidence*
> - *2 = Moderate confidence*
> - *3 = High confidence*

Prior to the workshop, student responses leaned heavily towards "No confidence" for both questions (Table 6-1). The pre-workshop mean and median confidence regarding discussions with non-technical peers were 0.55 and 1, respectively, with 0 representing no confidence and 3 representing high confidence. For discussing machine learning with an expert, the pre-workshop mean was 0.23, and the median was 0. Only 5 students reported high confidence in explaining machine learning to non-technical peers, and 1 student reported high confidence in explaining machine learning to a machine learning expert. These pre-workshop results reflected the students' written responses regarding their machine learning understanding; some students have an idea of what machine learning is, but most have only a cursory understanding of the

Question	Mean	Median	Range
How confident do you feel about explaining or discussing machine learning with a non-technical person?	0.55	1	0-3
How confident do you feel about explaining or discussing machine learning with a machine learning expert?	0.23	0	0-2

Table 6.1: Pre-workshop student response breakdown for questions related to ability to discuss machine learning with others.

concept and would not be comfortable discussing it with non-technical or technical individuals.

After the workshop, student confidence in discussing machine learning concepts with non-technical individuals improved, with the mean and median both increasing to 2 (Table 6-2). Every student felt at least slightly confident discussing machine learning concepts with non-technical individuals, with 18 students reporting moderate confidence, and 5 even reporting high confidence. Students also felt more confident in discussing machine learning with experts after the workshop, with the mean and median confidence increasing to 1.25 and 1, respectively. Most students who reported no confidence discussing with experts felt at least slightly confident after the workshop, and a few students moved from moderate confidence to high confidence. During smaller discussions after the workshops, students expressed that just learning the basic building blocks of machine learning models helped demystify the scary term, and interacting with accessible workshop tools gave them more confidence to discuss machine learning concepts. These results show that the workshops were successful at improving student confidence in discussing machine learning with individuals of all backgrounds.

6.1.4 Applications of Machine Learning

In order to evaluate students' familiarity with machine learning applications in their every day lives, we asked them the following free-response question in the pre-workshop and post-workshop questionnaire:

Question	Mean	Median	Range
How confident do you feel about explaining or discussing machine learning with a non-technical person?	2	2	1-3
How confident do you feel about explaining or discussing machine learning with a machine learning expert?	1.25	1	0-3

Table 6.2: Post-workshop student response breakdown for questions related to ability to discuss machine learning with others.

> *"What are some common things you see or use everyday that have machine learning in them?"*

Answers to this question fell into one of three categories: hardware products, software applications, and specific algorithms. Hardware products included answers like "*iPhones*," "*laptops*," "*Alexa*," and "*Google Home*." Many of the responses in this category showed an understanding of machine learning's prevalence in modern technology, but showed limited knowledge of how machine learning augments core hardware functionality. Responses in the software applications category included "*Youtube*," "*Google search*," and "*Netflix*." These answers showed a deeper recognition of machine learning applications in the software products that we interact with on a daily basis. The last category of responses referenced specific machine learning algorithms and demonstrated an understanding of how machine learning yields unique functionality. These responses included "*iPhone FaceID*," "*Youtube recommendations*," "*Google's targeted ads*," "*Netflix recommendation algorithm*," and "*Instagram face filters*."

For the pre-workshop questionnaire, 23 responses fell under the hardware products category, 10 responses fell under the software applications category, and 6 responses fell under the specific algorithms category. Students were allowed to provide multiple answers to the free-response question, and thus many fell into more than one of the three categories. For the post-workshop questionnaire, the responses shifted away from general hardware products, with 15 responses related to hardware products, 18 responses related to software applications, and 13 responses related to specific machine learning algorithms. This shift was likely due to the first workshop's focus on defining machine learning and discussing a number of specific machine learning algorithms that students may not have already be aware of. The post-workshop questionnaire had far more references to these applications, including answers like "*image classification*," and "*voice recognition*," as well as applications that we did not explicitly discuss, such as "*facial recognition software*," and "*deep fakes*". The additional specificity of students' post-workshop questionnaire responses demonstrated an improved understanding of the specific ways in which machine learning is used across popular hardware and software products. This result shows that giving students even a limited understanding of machine learning can help them better identify areas where machine learning is applied in their own daily interactions with technology.

6.1.5 Machine Learning Sentiment

While it is accepted that machine learning has revolutionized the future of computer science and artificial intelligence, it is important to understand whether students believe machine learning will positively or negatively affect their lives. The dangers of machine learning misuse have recently become a focus of machine learning research, with adversarial machine learning, AI surveillance, and deep fakes making headlines over the last few years. As machine learning education continues to expand, it is important to consider these more controversial machine learning applications and discuss whether they pose a danger to technology and our everyday lives. While the workshop did not explicitly address these more controversial topics, I posed the following two questions in both the pre-workshop and post-workshop questionnaire to evaluate general student sentiment around the advantages and dangers of machine learning:

> *"How much do you agree with the following statement: Machine learning will make everyone's life better. (Please check one)"*
>
> - *0 = Strongly disagree*
> - *1 = Disagree*

Question	Mean	Median	Range
How much do you agree with the following statement: Machine learning will make everyone's life better.	2.12	2	1-3
How much do you agree with the following statement: Machine learning is dangerous.	1.32	1	0-2

Table 6.3: Pre-workshop student response breakdown for questions related to machine learning sentiment.

- *2 = Agree*
- *3 = Strongly Agree*

> *"How much do you agree with the following statement: Machine learning is dangerous. (Please check one)"*

- *0 = Strongly disagree*
- *1 = Disagree*
- *2 = Agree*
- *3 = Strongly Agree*

Prior to the workshop, the mean student response for whether machine learning improves our lives was 2.12, the median was 2, and the range was 1-3, where 0 represents strongly disagree, and 3 represents strongly agree (Table 6-3). After the workshop the mean and median both improved to 2.57 and 2.5, respectively (Table 6-4). While the workshop did not specifically discuss improving our lives with machine learning, students likely saw the potential positive effects of the machine learning applications discussed in the workshop. In terms of sentiment regarding the dangers of machine learning, the pre-workshop mean was 1.32, the median was 1, and the range was 0-2. After the workshop, the mean decreased slightly to 1.25, and the median and range remain the same. This change was not significant, but it is worth noting that students do not think of machine learning as a dangerous addition to their everyday lives. Even so, it is important to prompt students to think about the potential negative implications of increased machine learning prevalence.

Question	Mean	Median	Range
How much do you agree with the following statement: Machine learning will make everyone's life better.	2.46	2.5	1-3
How much do you agree with the following statement: Machine learning is dangerous.	1.25	1	0-2

Table 6.4: Post-workshop student response breakdown for questions related to machine learning sentiment.

6.1.6 Machine Learning Applicability

Throughout the workshop, students were introduced to a variety of machine learning applications. We delved into many of these applications via hands on exercises (Quick Draw, Teachable Machine, PAC, App Inventor), but also discussed a number of other applications from a high level. Students were introduced to the motivation behind image and audio classification, and we discussed a range of other ways machine learning has been applied across industry and research fields. One of the goals of introducing so many applications to students was to give them a better sense of when machine learning can be applied effectively. To gauge this, I posed the following free-response question in the pre-workshop and post-workshop questionnaire:

> *"How do you determine whether or not machine learning should be used to solve a problem?"*

Responses to this question fell into one of three categories. The first type of response proposed that machine learning can be used to attack problems where current solutions are too slow or incapable. Many of these responses also referenced phrases like "*a machine can solve it faster*," or "*humans can't solve it.*" 10 pre-workshop responses fell into this category. The second category of responses specifically referenced the importance of data in building effective machine learning solutions. These responses included phrases like "*require enough data*," and "*if you can learn from a dataset.*" 9 pre-workshop responses fell under this category. The final response category referenced the type and complexity of the problem in question, often specifying a specific task that machine learning excels at. These responses included phrases like, "*classification problems*" or "*pattern recognition problems.*" The remaining 5 non-empty responses fell under this category. After the workshop, this distribution changed significantly, with 4 responses referencing effectiveness of current solutions, 16 responses referencing data, and 2 responses citing specific machine learning problems.

6.1.7 Effectiveness of Teaching Tools

The Boston Latin Academy students engaged with a number of teaching tools throughout the six workshops. These tools included Google's Quick Draw, Google's Teachable Machine, the Personal Audio Classifier, and MIT App Inventor. In order to gauge the effectiveness of these teaching tools, I posed the following questions to students in the post-workshop questionnaire, each with the standard 0-3 strongly agree to strongly disagree scale.

> *"How much do you agree with the following statement: I enjoyed using the Quick Draw web tool. (Please check one)"*
>
> *"How much do you agree with the following statement: I enjoyed using the Teachable Machine web tool. (Please check one)"*
>
> *"How much do you agree with the following statement: I enjoyed using the Personal Audio Classifier web tool. (Please check one)"*
>
> *"How much do you agree with the following statement: I enjoyed using the Voice Authentication app. (Please check one)"*

Results from these four multiple choice questions can be seen in Table 6.5. Overall, student sentiment around teaching tools was quite positive, with Google's Quick Draw reporting the highest average score of 2.75. This reflected the student excitement I saw during the Quick Draw exercises in class. The combination of an easy to use interface with the novelty of a powerful doodle classifier built directly into the browser kept students engaged throughout the exercise. The Personal Audio Classifier and

Teachable Machine web tools both scored averages of 2.35. The Voice Authentication app saw the lowest average of 2.1, though this was still a high score on the sentiment scale. In order to get feedback on the most enjoyable and the most confusing aspects of the workshop, I also posed the following free response questions:

> *"What were the most fun parts of the workshop?"*

> *"What were the most confusing parts of the workshop?"*

Out of the 23 non-empty responses to the first question, 12 responses referenced Quick Draw, 5 responses referenced Teachable Machine, 8 responses referenced the Personal Audio Classifier, and 9 responses referenced the Voice Authentication app. Many of the responses referenced the customizability of both PAC and the App Inventor application, implying that students enjoyed exploring and building their own applications of audio classification. For the second question, 13 responses stated there were no confusing parts of the workshop, and 8 responses mentioned struggles with the Voice Authentication app. Out of the 8 responses related to the Voice Authentication app, 4 directly referenced the exporting process being confusing, and 4 responses suggested struggles with getting the Voice Authentication app to perform well on the phone. Due to the limited functionality of student tablets, some models trained using the Personal Audio Classifier did not work as well on the tablets.

6.1.8 PAC Customizability and Improvements

In order to get further feedback on the Personal Audio Classifier interface and how successful students were at exploring various applications of audio classification, I posed the following free response questions:

> *"What was the most confusing aspect of the Personal Audio Classifier?"*

> *"What types of audio classification models did you explore in the Personal Audio Classifier?"*

Question	Mean	Median	Range
How much do you agree with the following statement: I enjoyed using the Quick Draw web tool.	2.75	2.5	2-3
How much do you agree with the following statement: I enjoyed using the Teachable Machine web tool.	2.35	2	2-3
How much do you agree with the following statement: I enjoyed using the Personal Audio Classifier web tool.	2.35	2	2-3
How much do you agree with the following statement: I enjoyed using the Voice Authentication app.	2.1	2	1-3

Table 6.5: Student response breakdown for questions related to teaching tool effectiveness.

> *"Please provide any other general feedback on the Personal Audio Classifier interface."*

While many students did not note any confusing aspects of the Personal Audio Classifier, 5 responses specified pain points. 3 of these responses mentioned the inability to play back recorded voice clips as a source of confusion when building models. The remaining 2 responses discussed the export process and the Voice Authentication app. For future iterations of this workshop, I hope to include additional features to resolve these pain points and allow for more fluid integration between the Personal Audio Classifier and MIT App Inventor.

For the second question, student responses covered a wide range of audio classification models. Many responses referenced either word or number classification, where a model distinguishes between users speaking different words or numbers. A majority of students also explored voice classifiers, pairing up with one or more students to build a model that could distinguish between different voices. One group of students even built a classifier using music and sounds played from their phones. The range of responses to this question demonstrated that students were successfully able to build a wide range of audio classification models using the Personal Audio Classifier.

Responses to the final question were generally positive in nature. Students explicitly mentioned the ease of use of the interface and the aesthetic elements of the PAC GUI. Some students provided suggestions for new feature additions, including the ability to play back audio after users record clips. Student responses to these three questions showed that they were able to quickly grasp the PAC interface and build impactful audio classification models. Furthermore, feedback provided by the students will help the App Inventor team further improve future workshop content.

Chapter 7

Conclusion

7.1 Contributions

The Personal Audio Classifier and the PAC Extension have been important additions to a suite of MIT App Inventor tools that bring machine learning to the forefront of computer science curriculum. In this thesis, I detail a number of important contributions related to audio classification, spectrogram analysis, and educational machine learning tools that were achieved throughout this process. These contributions are listed below in the order they were presented.

1. The Personal Audio Classifier (PAC), a machine learning tool that allows users to build, analyze, and export custom audio-classification models in their web browser without the need for expensive technology stacks.

2. A framework for performing on-the-fly spectrogram audio conversion that can be used to translate audio classification to the image classification problem space in the browser.

3. An MIT App Inventor extension that allows users to use audio-classification models exported from PAC to build custom mobile applications.

4. A high school workshop curriculum that utilizes PAC and the App Inventor extension to explore custom audio-classifiers and teach basic machine learning

concepts.

5. An analysis of results collected from running two workshops across three Boston Latin Academy AP Computer Science classes.

7.2 Future Work

While the Personal Audio Classifier and PAC Extension received positive feedback throughout the Boston Latin Academy workshops, students and test users in the MIT community recommended a number of great features additions that could improve PAC. Additional features like audio clip uploading capabilities would allow for a more standardized approach to building custom classifiers. This feature would also allow for more robust testing capabilities as each type of audio classifier could be trained using identical audio clips. Furthermore, students suggested adding the ability to play back recorded audio clips in the browser interface.

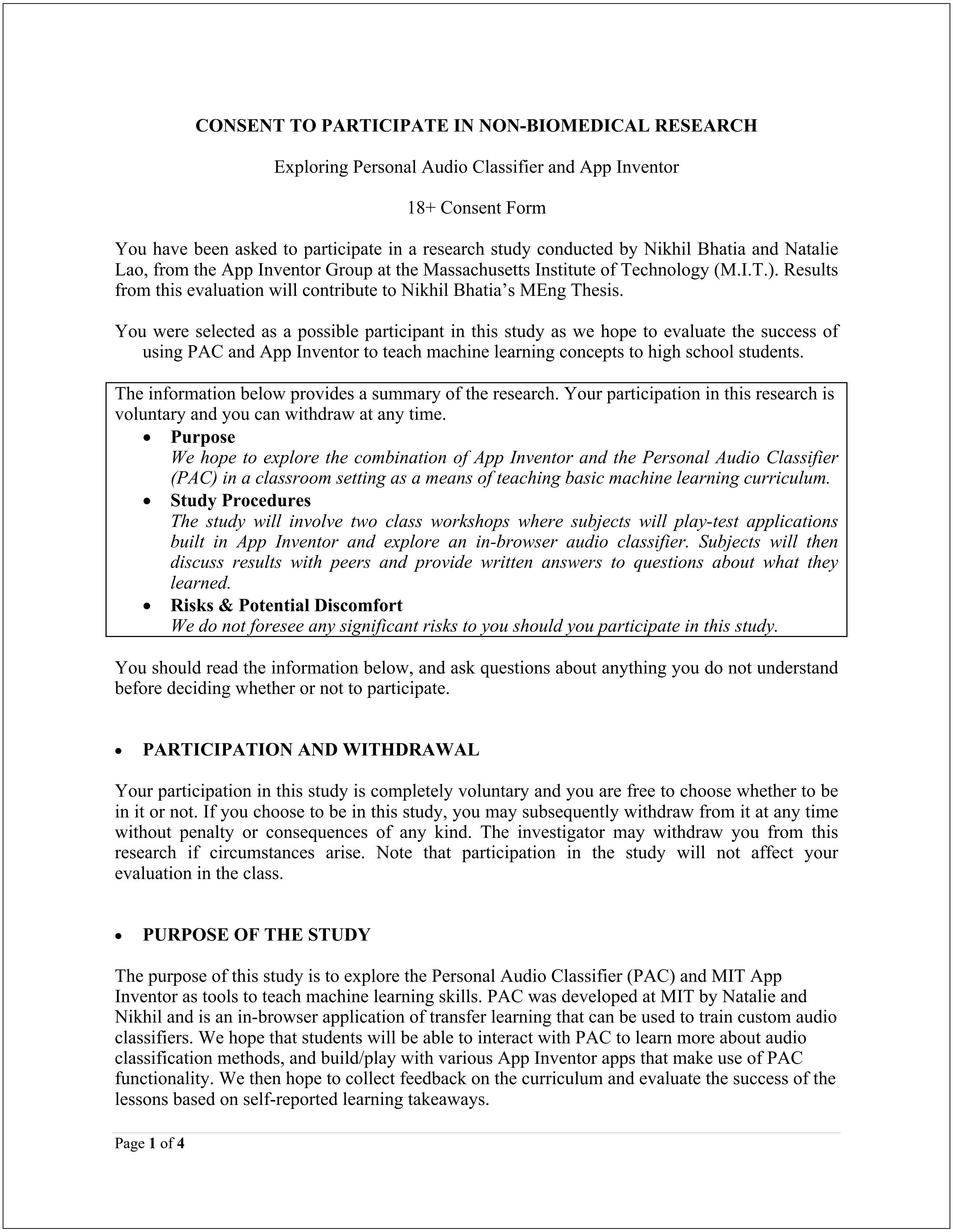
Other future projects could build on the PIC and PAC tools to bring other machine learning capabilities to an in-browser interface. The MIT App Inventor team has been working on integrating PoseNet [10] functionality into App Inventor applications, and an accompanying in-browser PoseNet exploration tool would have significant success in a workshop setting.

Lastly, more work could be done to build on the educational curriculum that I designed for the PAC workshop. Feedback from the workshop surveys could be integrated into future workshop curriculum. The Boston Latin Academy has also expressed interest in porting my workshop curriculum to a Mobile CSP approved curriculum that would be used in AP Computer Science classes across the country.

Appendix A

Consent forms for data release to research

A.1 Student Consent Form



CONSENT TO PARTICIPATE IN NON-BIOMEDICAL RESEARCH

Exploring Personal Audio Classifier and App Inventor

18+ Consent Form

You have been asked to participate in a research study conducted by Nikhil Bhatia and Natalie Lao, from the App Inventor Group at the Massachusetts Institute of Technology (M.I.T.). Results from this evaluation will contribute to Nikhil Bhatia's MEng Thesis.

You were selected as a possible participant in this study as we hope to evaluate the success of using PAC and App Inventor to teach machine learning concepts to high school students.

> The information below provides a summary of the research. Your participation in this research is voluntary and you can withdraw at any time.
>
> - **Purpose**
>   *We hope to explore the combination of App Inventor and the Personal Audio Classifier (PAC) in a classroom setting as a means of teaching basic machine learning curriculum.*
> - **Study Procedures**
>   *The study will involve two class workshops where subjects will play-test applications built in App Inventor and explore an in-browser audio classifier. Subjects will then discuss results with peers and provide written answers to questions about what they learned.*
> - **Risks & Potential Discomfort**
>   *We do not foresee any significant risks to you should you participate in this study.*

You should read the information below, and ask questions about anything you do not understand before deciding whether or not to participate.

- **PARTICIPATION AND WITHDRAWAL**

Your participation in this study is completely voluntary and you are free to choose whether to be in it or not. If you choose to be in this study, you may subsequently withdraw from it at any time without penalty or consequences of any kind. The investigator may withdraw you from this research if circumstances arise. Note that participation in the study will not affect your evaluation in the class.

- **PURPOSE OF THE STUDY**

The purpose of this study is to explore the Personal Audio Classifier (PAC) and MIT App Inventor as tools to teach machine learning skills. PAC was developed at MIT by Natalie and Nikhil and is an in-browser application of transfer learning that can be used to train custom audio classifiers. We hope that students will be able to interact with PAC to learn more about audio classification methods, and build/play with various App Inventor apps that make use of PAC functionality. We then hope to collect feedback on the curriculum and evaluate the success of the lessons based on self-reported learning takeaways.

- **PROCEDURES**

If you volunteer to participate in this study, we would ask you to do the following things:

1. Participate in two 48 minute class sections led by Nikhil and Natalie.
2. Explore the Personal Audio Classifier (PAC) and break out into smaller groups to train custom audio classification models (i.e. models that can learn to classify short audio clips into various categories).
3. Re-convene to discuss the successes and failures of the PAC model: what trained well, what didn't?
4. Explore pre-built App Inventor mobile apps that make use of PAC in various ways.
5. Answer written questions about the class, PAC, and App Inventor, and reflect on major takeaways from the workshop.

What is the Personal Audio Classifier? The Personal Audio Classifier (PAC) is a web application that allows you to build an in-browser audio classifier! You can specify labels and record corresponding audio clips to train your own custom machine learning model.

What is MIT App Inventor? App Inventor lets you build applications for Android phones using a simple block language. We'll play around with some simple pre-built applications that allow you to import your own PAC models!

- **POTENTIAL RISKS AND DISCOMFORTS**

We do not believe that there are any significant risks to you should you participate in the study. We will make sure that your personal information is not shared with others, and will never use your name when we talk about this research study.

- **POTENTIAL BENEFITS**

Benefits of participating in this study include getting to use novel machine learning applications built by the MIT App Inventor team, and getting exposure to basic machine learning concepts that will help subjects deepen their knowledge and interests in computer science.

- **PAYMENT FOR PARTICIPATION**

Subjects in this study will not receive any payment.

- **PRIVACY AND CONFIDENTIALITY**

Any information obtained in connection with this study and that can be identified with you will remain confidential and will be disclosed only with your permission or as required by law. In addition, your information may be reviewed by authorized MIT representatives to ensure compliance with MIT policies and procedures.

Subjects will be asked whether they agree to being video and audio recorded during the workshop. If consent is obtained, recorded photos and videos will not be used for purposes outside of MIT App Inventor.

- **IDENTIFICATION OF INVESTIGATORS**

If you have any questions or concerns about the research, please feel free to contact Nikhil Bhatia (650-823-5187, nwbhatia@mit.edu), or Natalie Lao (617-866-8304, natalie@mit.edu).

- **EMERGENCY CARE AND COMPENSATION FOR INJURY**

If you feel you have suffered an injury, which may include emotional trauma, as a result of participating in this study, please contact the person in charge of the study as soon as possible.

In the event you suffer such an injury, M.I.T. may provide itself, or arrange for the provision of, emergency transport or medical treatment, including emergency treatment and follow-up care, as needed, or reimbursement for such medical services. M.I.T. does not provide any other form of compensation for injury. In any case, neither the offer to provide medical assistance, nor the actual provision of medical services shall be considered an admission of fault or acceptance of liability. Questions regarding this policy may be directed to MIT's Insurance Office, (617) 253-2823. Your insurance carrier may be billed for the cost of emergency transport or medical treatment, if such services are determined not to be directly related to your participation in this study.

- **RIGHTS OF RESEARCH SUBJECTS**

You are not waiving any legal claims, rights or remedies because of your participation in this research study. If you feel you have been treated unfairly, or you have questions regarding your rights as a research subject, you may contact the Chairman of the Committee on the Use of Humans as Experimental Subjects, M.I.T., Room E25-143B, 77 Massachusetts Ave, Cambridge, MA 02139, phone 1-617-253 6787.

SIGNATURE OF RESEARCH SUBJECT OR LEGAL REPRESENTATIVE

I understand the procedures described above. My questions have been answered to my satisfaction, and I agree to participate in this study. I have been given a copy of this form.

____
Name of Subject

____
Name of Legal Representative (if applicable)

____________________________________ ______________
Signature of Subject Date

______________________________________ ______________
Legal Representative (if applicable) Date

SIGNATURE OF PERSON OBTAINING INFORMED CONSENT

In my judgment the subject is voluntarily and knowingly giving informed consent and possesses the legal capacity to give informed consent to participate in this research study.

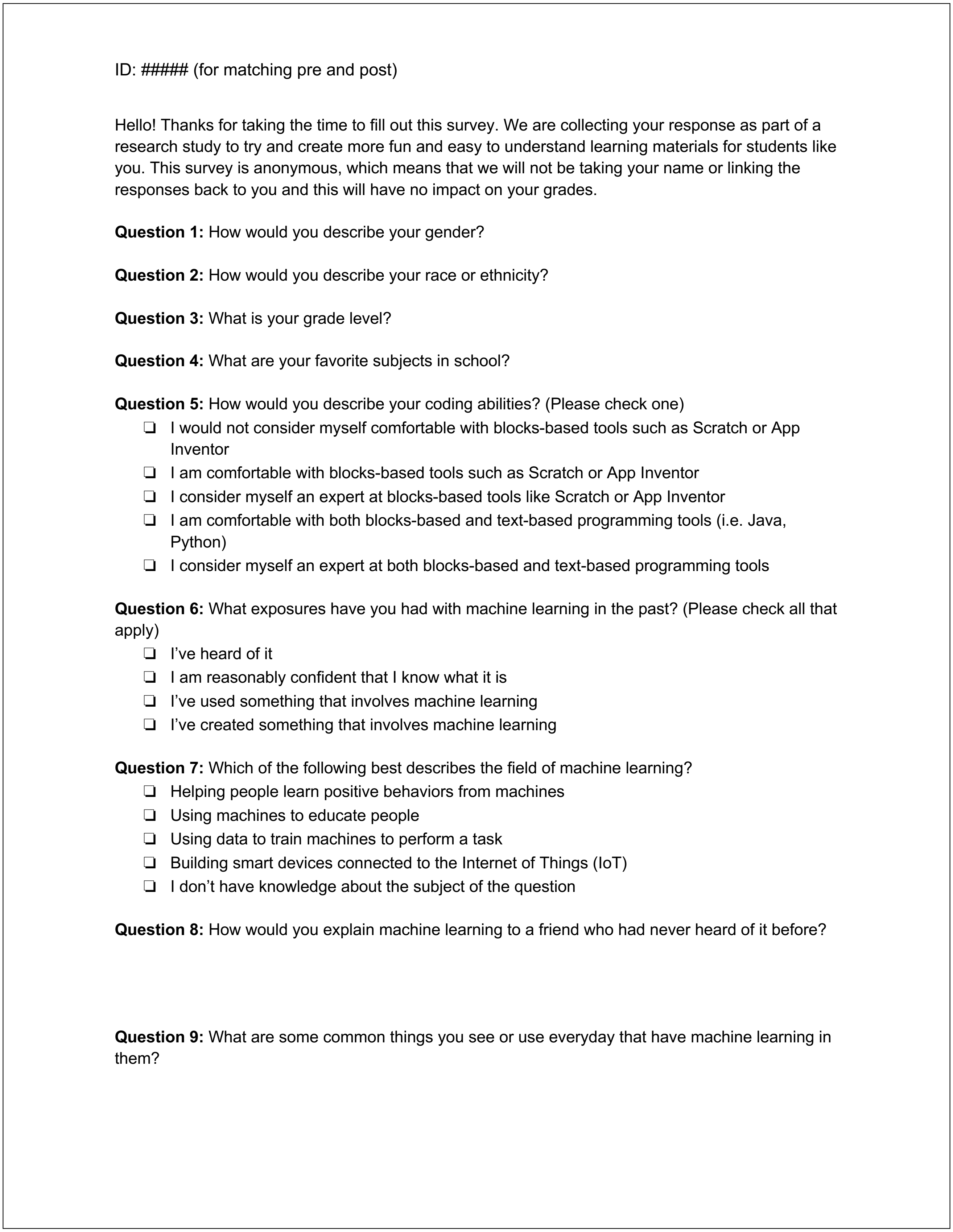
____
Name of Person Obtaining Informed Consent

______________________________________ ______________
Signature of Person Obtaining Informed Consent Date

Appendix B

Supplemental Workshop Handouts

B.1 Pre-workshop Questionnaire



ID: ##### (for matching pre and post)

Hello! Thanks for taking the time to fill out this survey. We are collecting your response as part of a research study to try and create more fun and easy to understand learning materials for students like you. This survey is anonymous, which means that we will not be taking your name or linking the responses back to you and this will have no impact on your grades.

Question 1: How would you describe your gender?

Question 2: How would you describe your race or ethnicity?

Question 3: What is your grade level?

Question 4: What are your favorite subjects in school?

Question 5: How would you describe your coding abilities? (Please check one)

- □ I would not consider myself comfortable with blocks-based tools such as Scratch or App Inventor
- □ I am comfortable with blocks-based tools such as Scratch or App Inventor
- □ I consider myself an expert at blocks-based tools like Scratch or App Inventor
- □ I am comfortable with both blocks-based and text-based programming tools (i.e. Java, Python)
- □ I consider myself an expert at both blocks-based and text-based programming tools

Question 6: What exposures have you had with machine learning in the past? (Please check all that apply)

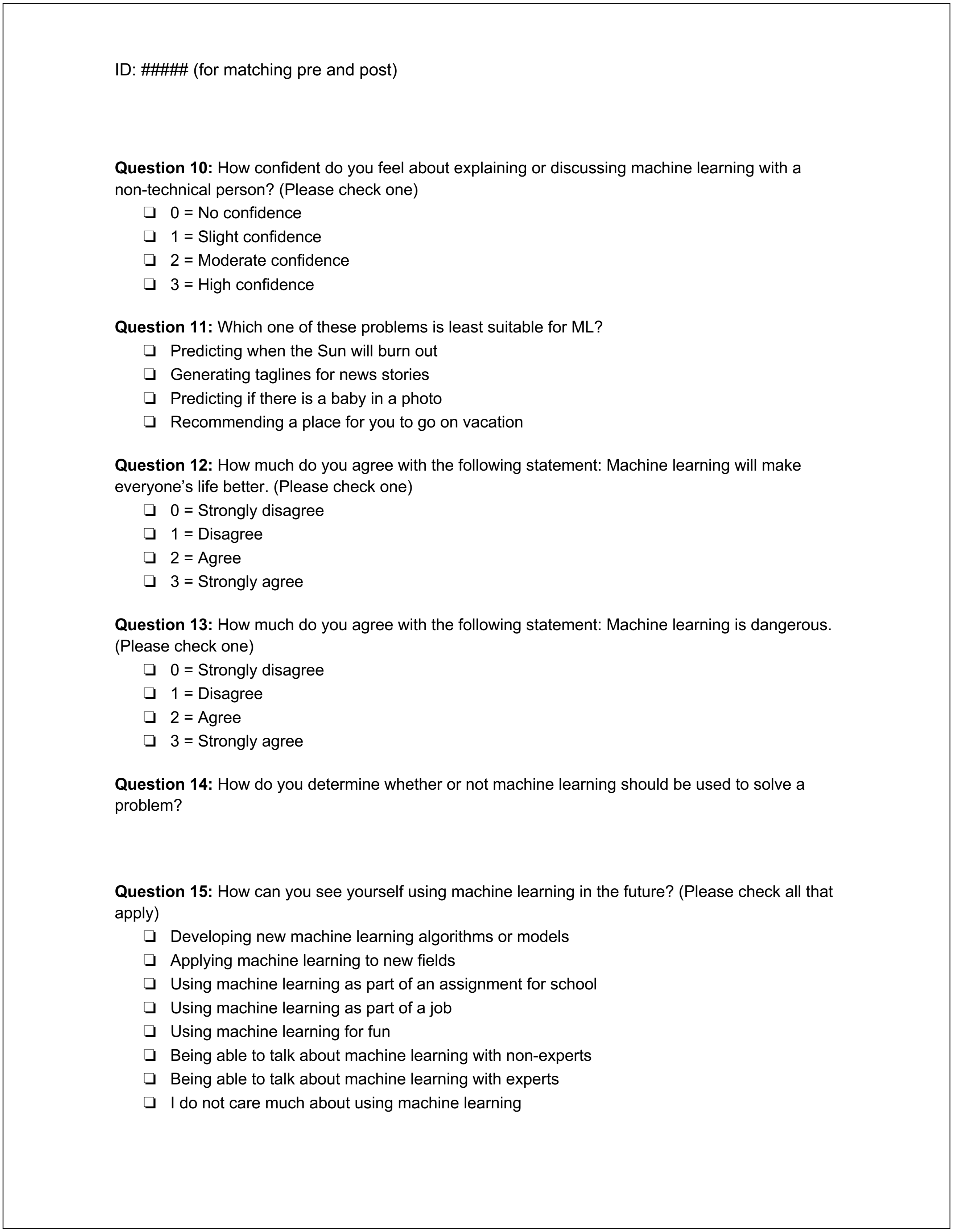
- □ I've heard of it
- □ I am reasonably confident that I know what it is
- □ I've used something that involves machine learning
- □ I've created something that involves machine learning

Question 7: Which of the following best describes the field of machine learning?

- □ Helping people learn positive behaviors from machines
- □ Using machines to educate people
- □ Using data to train machines to perform a task
- □ Building smart devices connected to the Internet of Things (IoT)
- □ I don't have knowledge about the subject of the question

Question 8: How would you explain machine learning to a friend who had never heard of it before?

Question 9: What are some common things you see or use everyday that have machine learning in them?



ID: ##### (for matching pre and post)

Question 10: How confident do you feel about explaining or discussing machine learning with a non-technical person? (Please check one)

- ❑ 0 = No confidence
- ❑ 1 = Slight confidence
- ❑ 2 = Moderate confidence
- ❑ 3 = High confidence

Question 11: Which one of these problems is least suitable for ML?

- ❑ Predicting when the Sun will burn out
- ❑ Generating taglines for news stories
- ❑ Predicting if there is a baby in a photo
- ❑ Recommending a place for you to go on vacation

Question 12: How much do you agree with the following statement: Machine learning will make everyone's life better. (Please check one)

- ❑ 0 = Strongly disagree
- ❑ 1 = Disagree
- ❑ 2 = Agree
- ❑ 3 = Strongly agree

Question 13: How much do you agree with the following statement: Machine learning is dangerous. (Please check one)

- ❑ 0 = Strongly disagree
- ❑ 1 = Disagree
- ❑ 2 = Agree
- ❑ 3 = Strongly agree

Question 14: How do you determine whether or not machine learning should be used to solve a problem?

Question 15: How can you see yourself using machine learning in the future? (Please check all that apply)

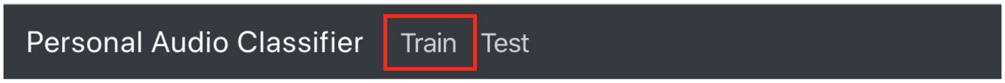
- ❑ Developing new machine learning algorithms or models
- ❑ Applying machine learning to new fields
- ❑ Using machine learning as part of an assignment for school
- ❑ Using machine learning as part of a job
- ❑ Using machine learning for fun
- ❑ Being able to talk about machine learning with non-experts
- ❑ Being able to talk about machine learning with experts
- ❑ I do not care much about using machine learning

B.2 Personal Audio Classifier Guide

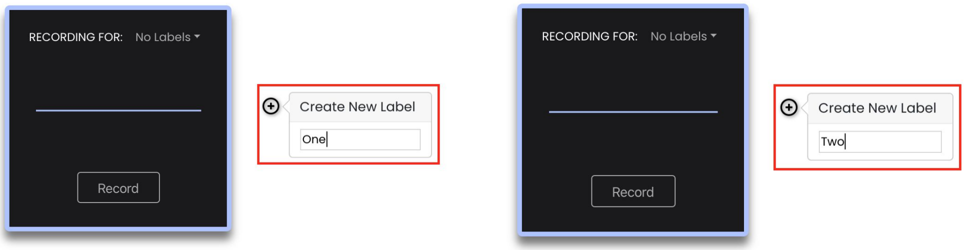
Personal Audio Classifier (Student Guide)

1. Navigate browser to https://c1.appinventor.mit.edu/

We're going to train an audio classifier from scratch! First, we have to teach it what we want to classify. This is the "training" phase.



2. Add labels corresponding to three different numbers (ex: "One," "Two," "Eleven")

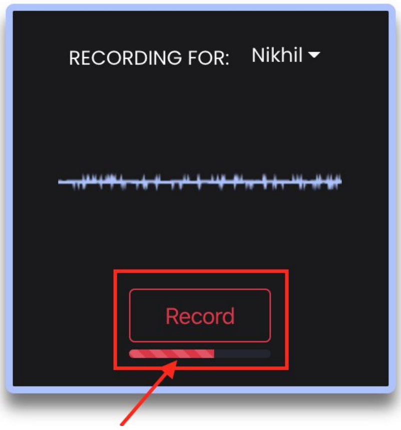


We're going to train a simple model that can distinguish between speaking these three different numbers.

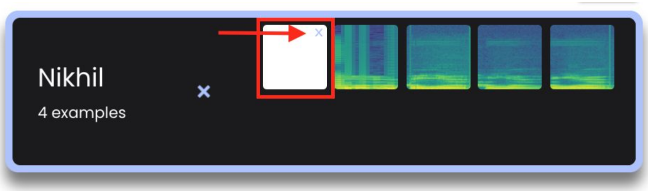
3. Test your computer's microphone and get a sense of how long each voice clip will be.

Practice by pressing record and saying "Hello" loudly and clearly (you may need to grant microphone access).

Make sure you only speak while the red recording bar is active.



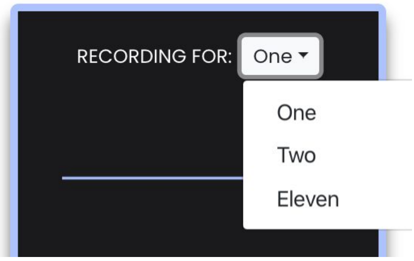
RECORDING FOR: Nikhil



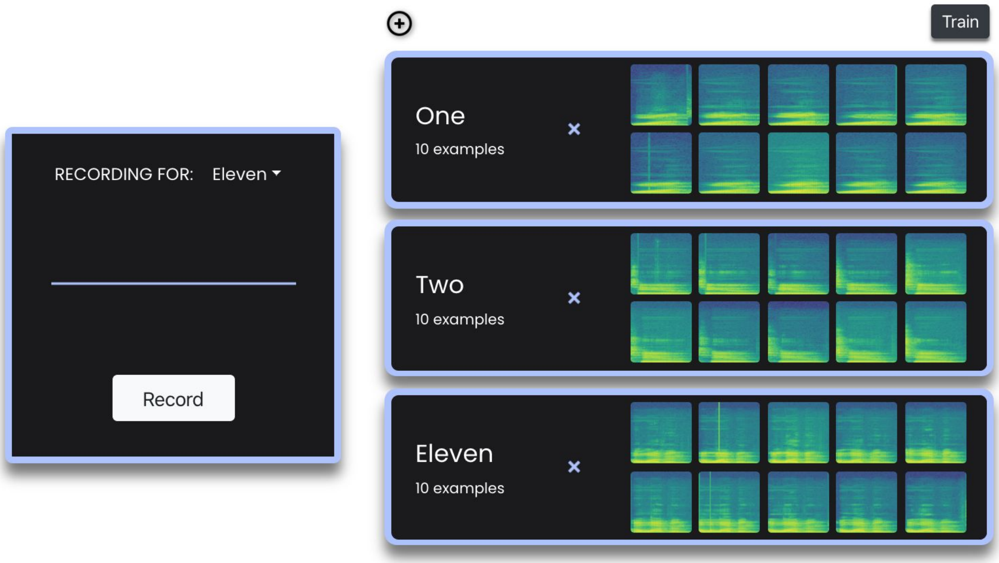
As your microphone warms up, you may need to remove some bad recordings by pressing the blue "x" in the top right corner of each image.

4. Let’s train our model! Clear any practice recordings and record 5-10 good examples for each number.

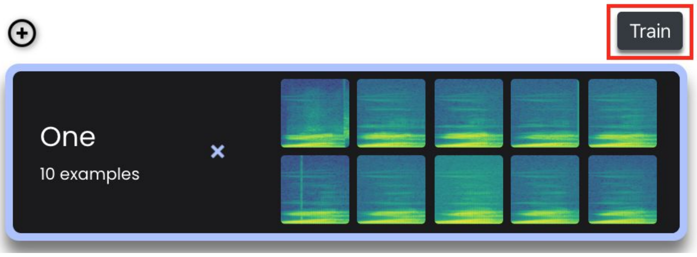
Click the “RECORDING FOR” button to switch to a different label.

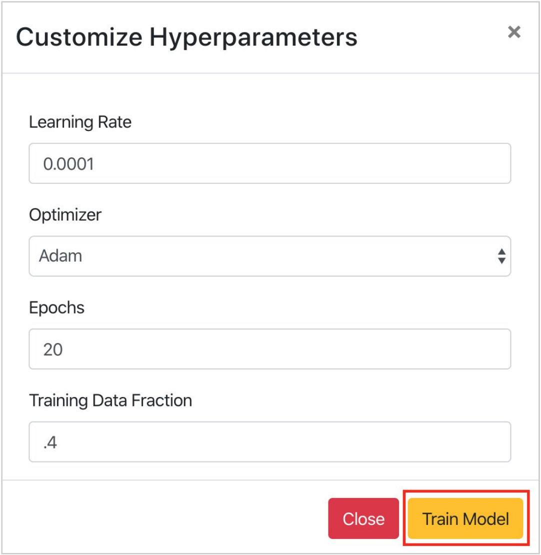


You should now have 5-10 recorded examples for each of your three numbers (remember you can remove any odd-looking examples and record them again).



4. Let's train our model! Press the "train" button, and then press "train model" using the default parameters.




Customize Hyperparameters

Learning Rate

0.0001

Optimizer

Adam

Epochs

20

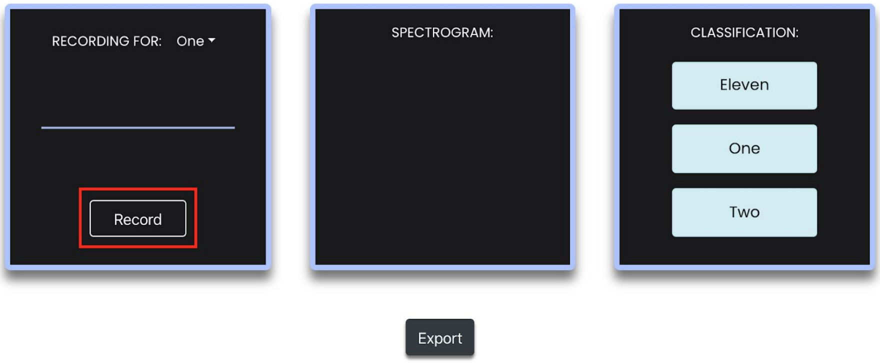
Training Data Fraction

.4

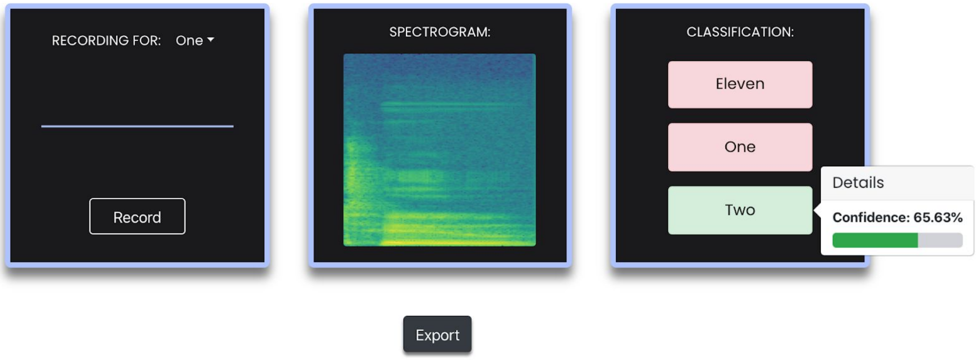
4. Now that our model is trained, it's time to test it!

Press record to record a voice clip for a number and see how the model classifies the clip!

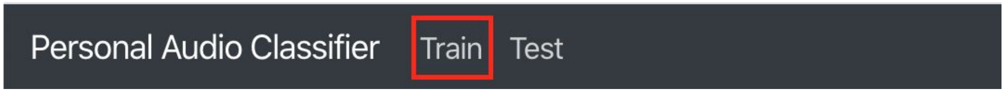
(Don't worry about who the "RECORDING FOR" button is set to). Just see how the model performs for each of the three numbers.



Hover over each of the labels to see the model's confidence in its classification.

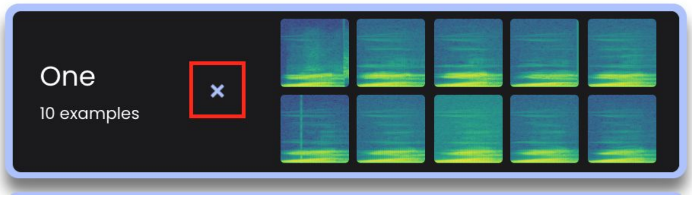


You can press the "train" button in the navbar to return to the training screen and make changes to the model (then retrain your updated model following step 4).



That’s it! You’ve just built a custom audio classifier. Now go back to the “Train” screen and train a new model.

Press the blue “x” shown below to remove a label from the classifier.



Here are some exercises we recommend you try with your partner:

1. *Try building a classifier that can distinguish between different names or words. How different do the words have to be for the classifier to work well? Does syllable count matter? Length?*

2. *Try building a classifier that can distinguish between you and a partner saying “hello”. Train using two labels, each with 5-10 examples of you/your partner saying “hello.” First test the model with “hello” only, and see it can distinguish between you and your partner’s voices. Then see if the model can correctly classify phrases or words that aren’t “hello”.*

3. *Now try training a model that can distinguish more generally between you and your partner’s voice. Try training on multiple words (not just “hello”), or try reading an online passage or part of a book! Does this work better or worse than the “hello” model? Do you and your partner have to train on the same words for the model to work well?*

4. *Can you train a model to classify emotions? Try angry or sad or happy! Does this work well? If not, why do you think that is?*

5. *As you evaluate your models, do you notice any patterns regarding wrong classifications? Is your model bad at classifying specific numbers, words, or even voices? Why do you think that is?*

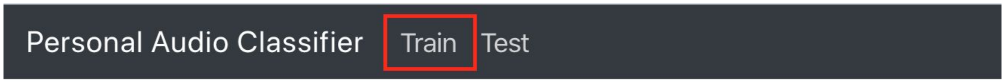
6. *When the model performs well, do you think you could do what it’s doing just by looking at the spectrograms? Do you and your partner saying “hello” look different to the human eye? Why or why not?*

B.3 Voice Authentication App Guide

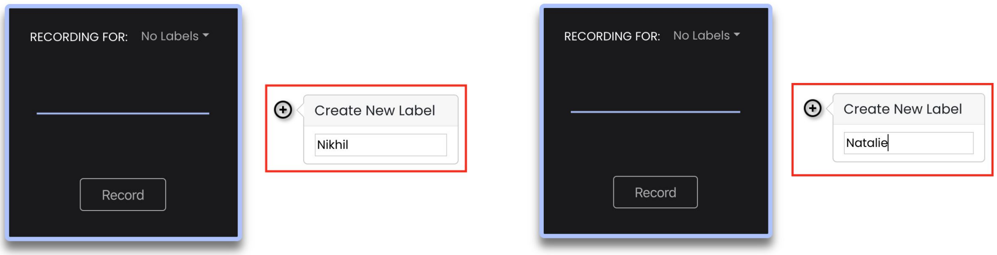
Voice Authentication App (Student Guide)

1. Navigate browser to https://c1.appinventor.mit.edu/

We're going to train an audio classifier from scratch! First, we have to teach it what we want to classify. This is the "training" phase.



2. Add labels corresponding to your voice and your partner's voice.



RECORDING FOR: No Labels

Nikhil

RECORDING FOR: No Labels

Natalie

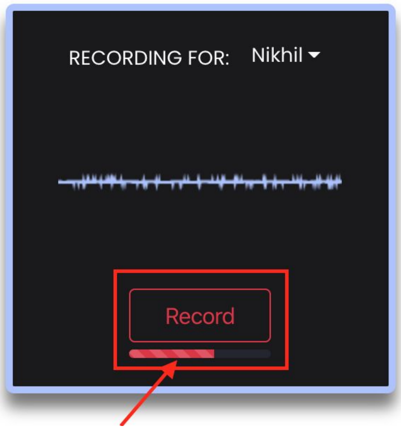
Now we have two possible classifications for our model: Nikhil and Natalie.

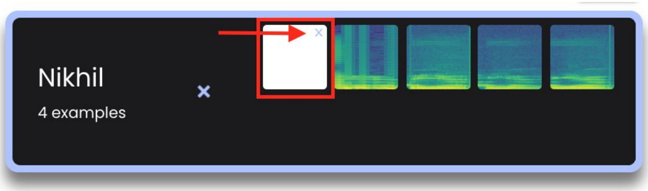
Your model is going to learn to distinguish between you and your partner saying the word "Hello." Let's record some voice clips for the model to train on.

3. Test your computer's microphone and get a sense of how long each voice clip will be.

Practice by pressing record and saying "Hello" loudly and clearly (you may need to grant microphone access).

Make sure you only speak while the red recording bar is active.

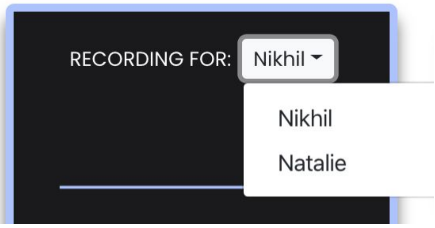




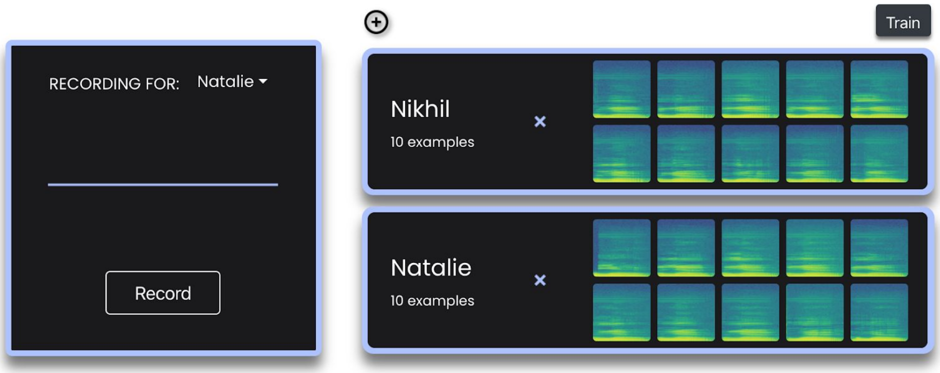
As your microphone warms up, you may need to remove some bad recordings by pressing the blue "x" in the top right corner of each image.

4. Let’s train our model! Clear any practice recordings and ecord 5-10 good examples of you and your partner saying “Hello.”

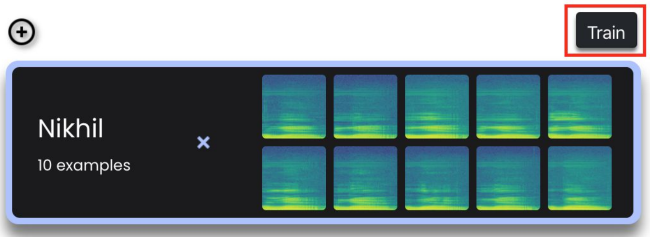
Click the “RECORDING FOR” button to record for different labels.

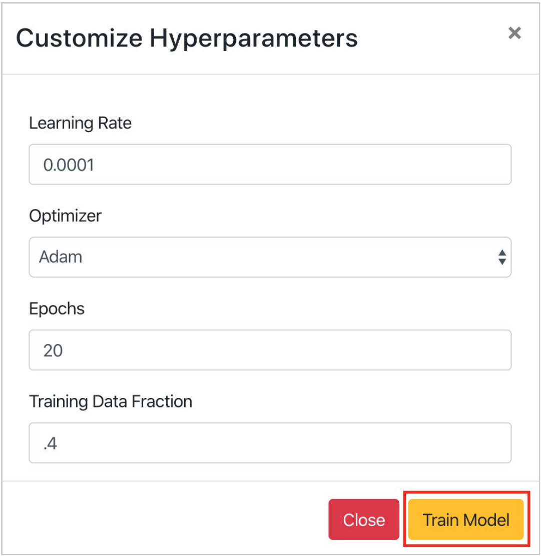


Double check that your recorded examples look similar to the ones shown below (remember you can remove any odd-looking examples and record them again).



4. Let’s train our model! Press the “train” button, and then press “train model” using the default parameters.




Customize Hyperparameters

Learning Rate: 0.0001

Optimizer: Adam

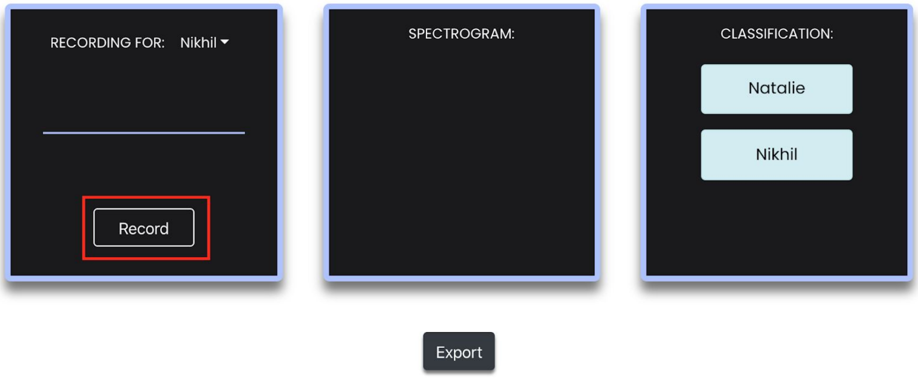
Epochs: 20

Training Data Fraction: .4

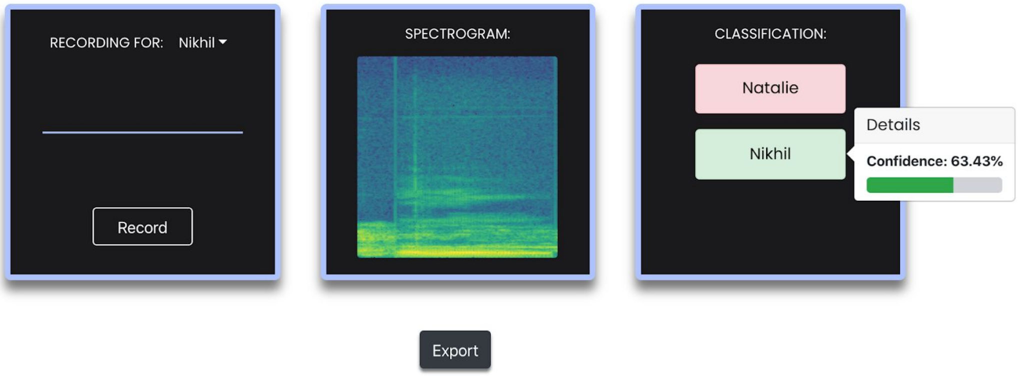
4. Now that our model is trained, it's time to test it!

Press record to record a voice clip and see how the model classifies the clip!

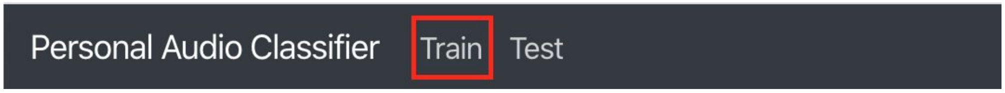
(Don't worry about who the "RECORDING FOR" button is set to). Just see how the model performs for you and your partner's voices.



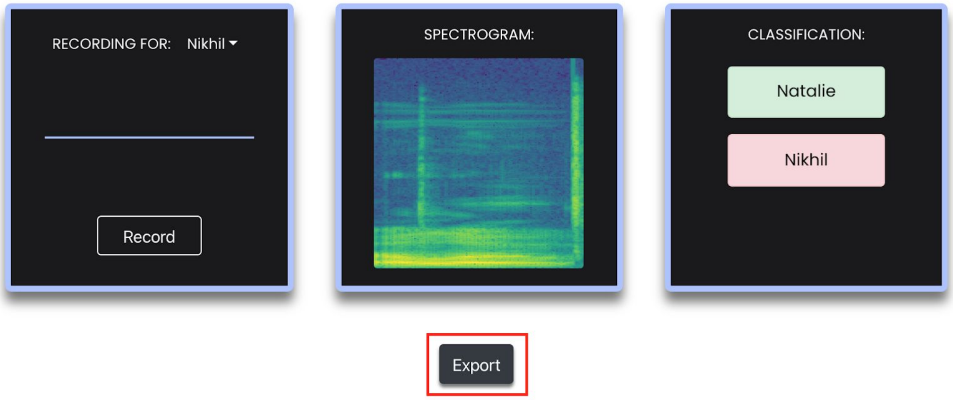
Hover over each of the labels to see the model's confidence in its classification.



You can press the "train" button in the navbar to return to the training screen and make changes to the model (then retrain your updated model following step 4).



4. Once you’re confident in your model, press the “Export” button to export your new model!

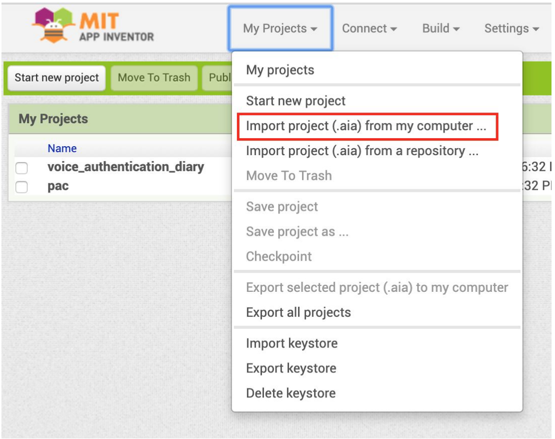


5. Now let’s build our Voice Authentication App using this exported model!

Download the Voice Authentication Diary from this link:

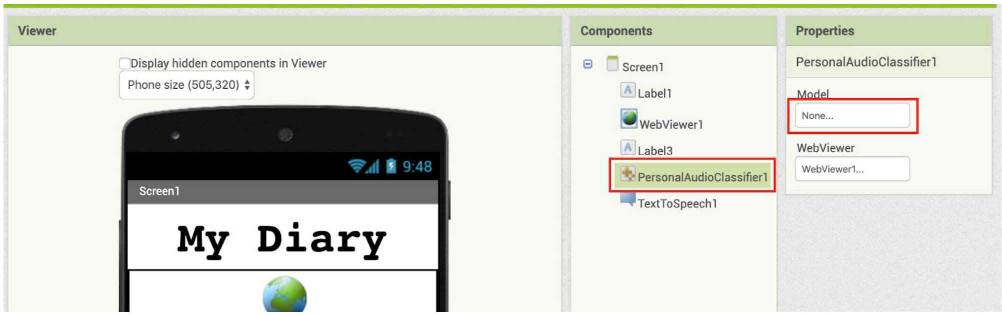
http://bit.ly/voice_auth_diary

Navigate to http://ai2.appinventor.mit.edu/, go to My Projects, and import the file you just downloaded.



6. Now let’s upload your custom model to be used in the app.

Open up the “voice_authentication_diary” app you just imported, click on the PersonalAudioClassifier1 Component, and upload your exported model.



Viewer

□ Display hidden components in Viewer

Phone size (505,320)

Screen1

My Diary

Components

Screen1

Label1

WebViewer1

Label3

PersonalAudioClassifier1

TextToSpeech1

Properties

PersonalAudioClassifier1

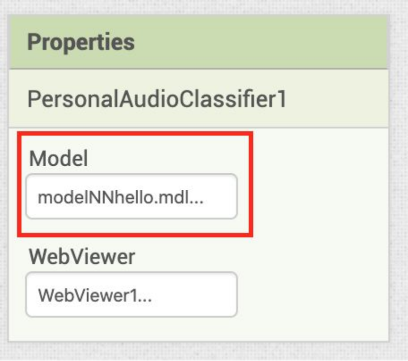
Model

None...

WebViewer

WebViewer1...

After uploading your model.mdl file, make sure it shows up in the “Model” box as shown below.



Properties

PersonalAudioClassifier1

Model

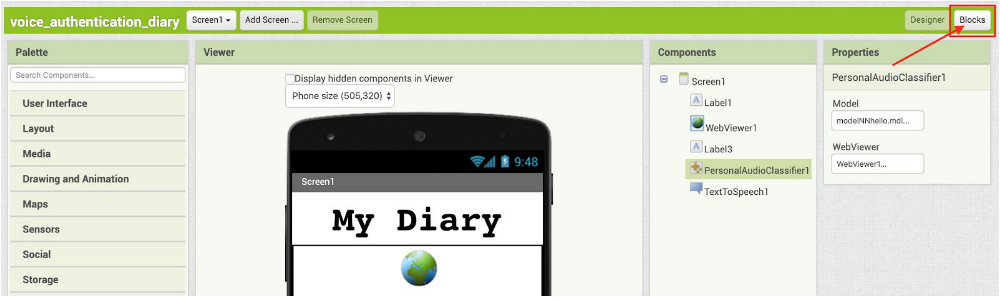
modelNNhello.mdl...

WebViewer

WebViewer1...

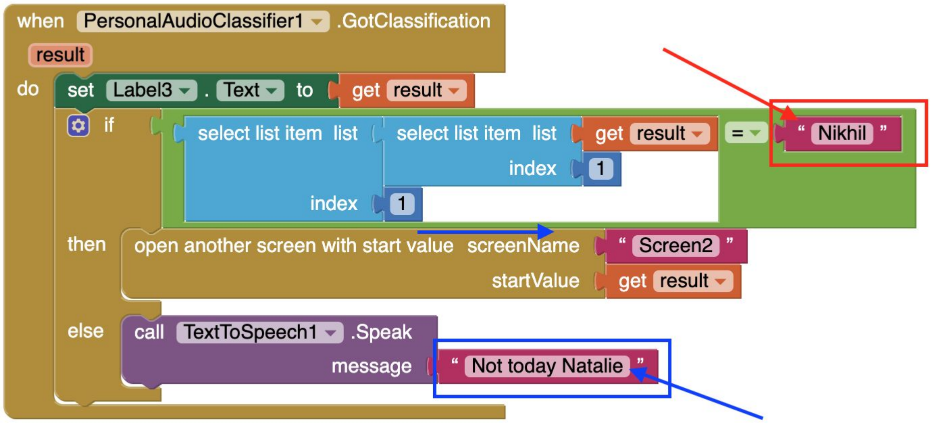
7. Now that your model is uploaded, let's edit the blocks to tell the diary who is allowed in!

Click on the blocks button to navigate to the blocks view.



In the red box, input the name of the label that corresponds to your voice. If the model classifies the voice as this person, it will let you in.

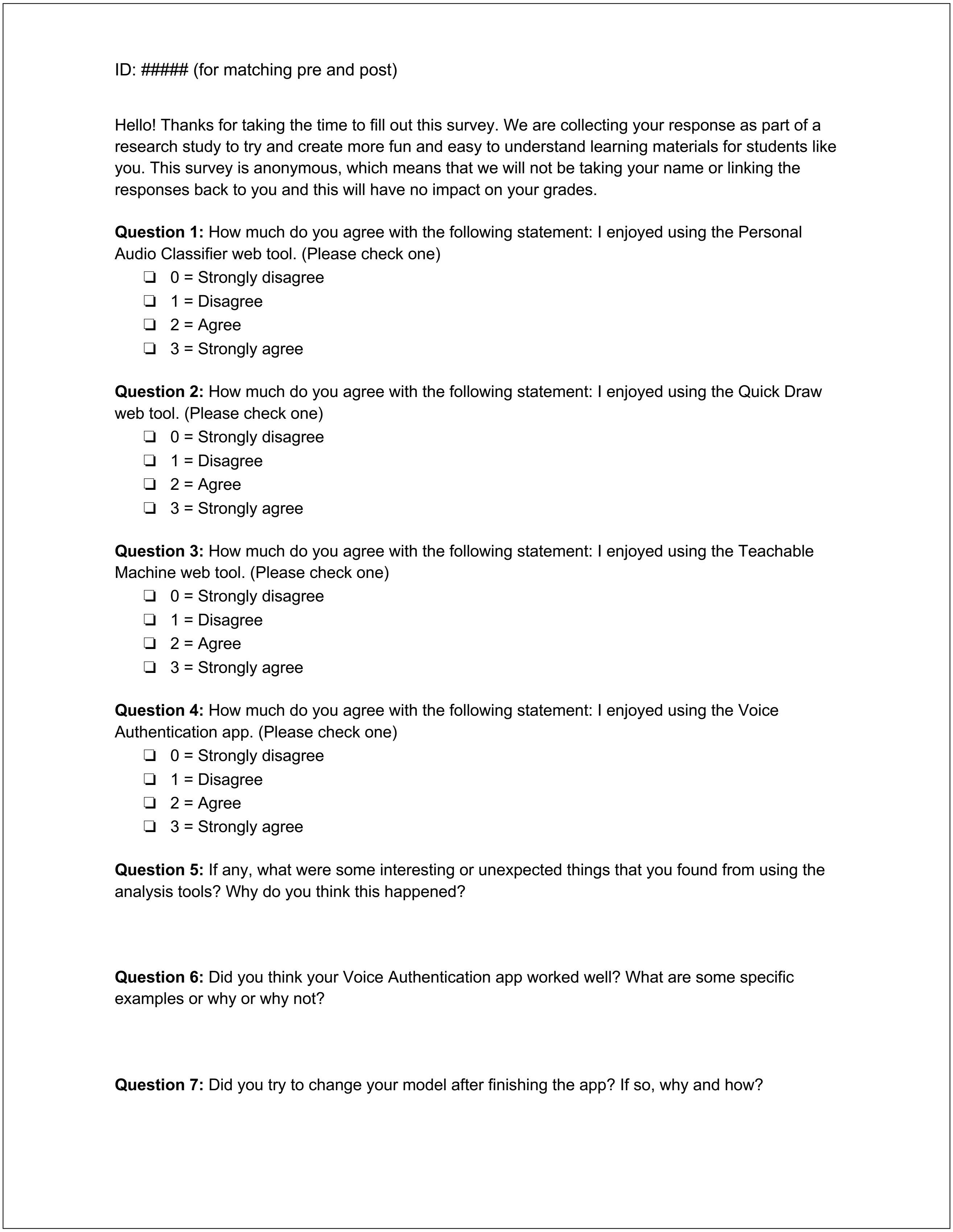
In the blue box, input an access denied phrase that the app will speak out loud if it doesn't think the voice is you!



```
when PersonalAudioClassifier1 .GotClassification
  result
do  set Label3 . Text to  get result
    if    select list item list  select list item list  get result  = " Nikhil "
                                 index 1
          index 1
    then  open another screen with start value  screenName " Screen2 "
                                                startValue get result
    else  call TextToSpeech1 .Speak
               message " Not today Natalie "
```

8. And we're done! Build the application to your phone and test it out. Raise your hand if the app is not working as expected!

B.4 Post-workshop Questionnaire



ID: ##### (for matching pre and post)

Hello! Thanks for taking the time to fill out this survey. We are collecting your response as part of a research study to try and create more fun and easy to understand learning materials for students like you. This survey is anonymous, which means that we will not be taking your name or linking the responses back to you and this will have no impact on your grades.

Question 1: How much do you agree with the following statement: I enjoyed using the Personal Audio Classifier web tool. (Please check one)

□ 0 = Strongly disagree
□ 1 = Disagree
□ 2 = Agree
□ 3 = Strongly agree

Question 2: How much do you agree with the following statement: I enjoyed using the Quick Draw web tool. (Please check one)

□ 0 = Strongly disagree
□ 1 = Disagree
□ 2 = Agree
□ 3 = Strongly agree

Question 3: How much do you agree with the following statement: I enjoyed using the Teachable Machine web tool. (Please check one)

□ 0 = Strongly disagree
□ 1 = Disagree
□ 2 = Agree
□ 3 = Strongly agree

Question 4: How much do you agree with the following statement: I enjoyed using the Voice Authentication app. (Please check one)

□ 0 = Strongly disagree
□ 1 = Disagree
□ 2 = Agree
□ 3 = Strongly agree

Question 5: If any, what were some interesting or unexpected things that you found from using the analysis tools? Why do you think this happened?

Question 6: Did you think your Voice Authentication app worked well? What are some specific examples or why or why not?

Question 7: Did you try to change your model after finishing the app? If so, why and how?

ID: ##### (for matching pre and post)

Question 8: How confident do you feel about describing what you did for the Voice Authentication project to a non-technical person?

- ❏ 0 = No confidence
- ❏ 1 = Slight confidence
- ❏ 2 = Moderate confidence
- ❏ 3 = High confidence

Question 9: How confident do you feel about describing what you did for the Voice Authentication project to a machine learning expert?

- ❏ 0 = No confidence
- ❏ 1 = Slight confidence
- ❏ 2 = Moderate confidence
- ❏ 3 = High confidence

Question 10: What were the most fun parts of this workshop?

Question 11: What were the most confusing parts of this workshop?

Question 12: Please check all that apply.

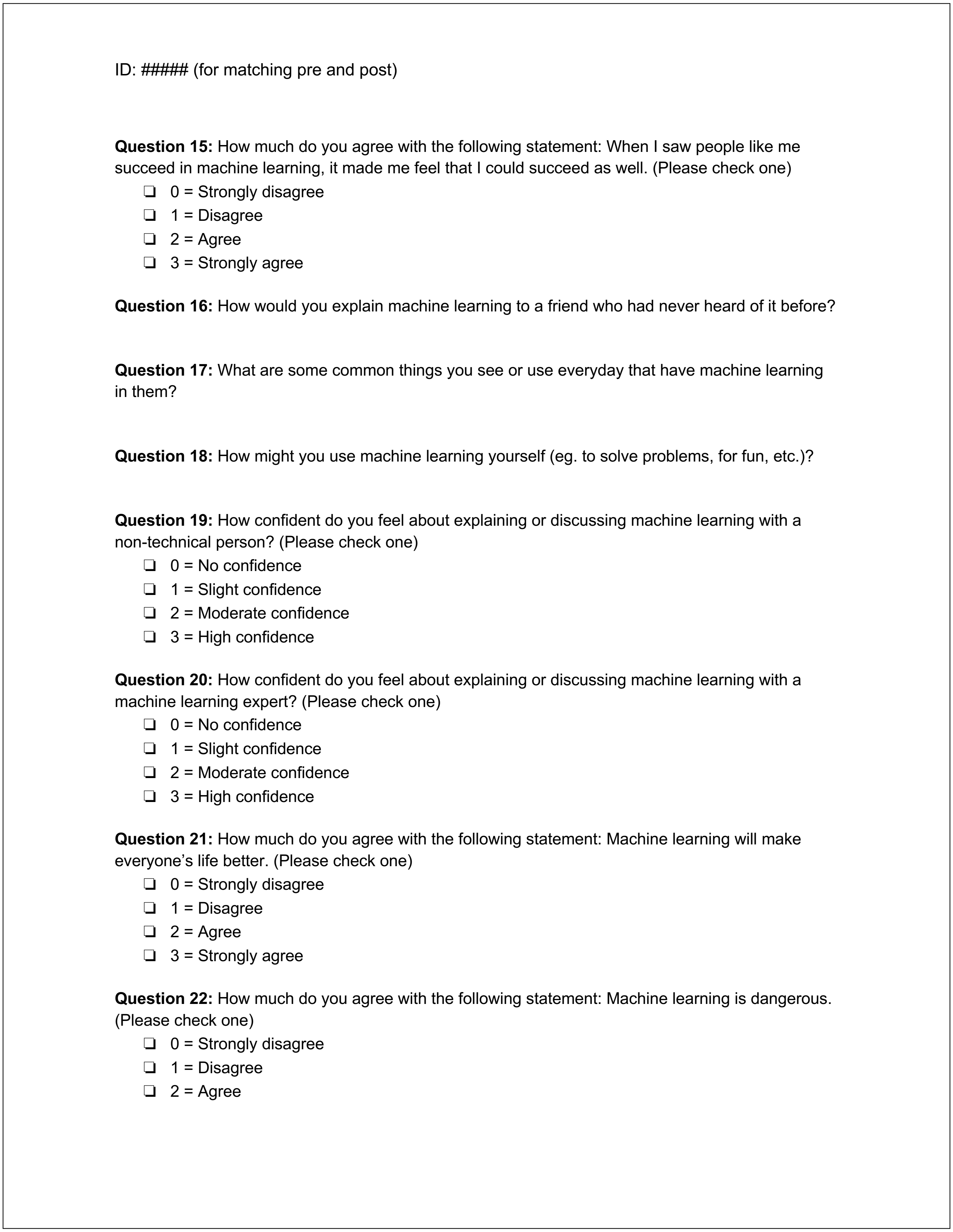
- ❏ I was successful in understanding the concepts taught by the teacher
- ❏ I was successful in using the Personal Audio Classifier to complete the assigned tasks
- ❏ I was successful in building a working Voice Identification app

Question 13: How much do you agree with the following statement: Other students in the workshop helped me learn the technical material. (Please check one)

- ❏ 0 = Strongly disagree
- ❏ 1 = Disagree
- ❏ 2 = Agree
- ❏ 3 = Strongly agree

Question 14: How much do you agree with the following statement: In this workshop, I saw people like me succeed at learning machine learning. (Please check one)

- ❏ 0 = Strongly disagree
- ❏ 1 = Disagree
- ❏ 2 = Agree
- ❏ 3 = Strongly agree



ID: ##### (for matching pre and post)

Question 15: How much do you agree with the following statement: When I saw people like me succeed in machine learning, it made me feel that I could succeed as well. (Please check one)

- ❑ 0 = Strongly disagree
- ❑ 1 = Disagree
- ❑ 2 = Agree
- ❑ 3 = Strongly agree

Question 16: How would you explain machine learning to a friend who had never heard of it before?

Question 17: What are some common things you see or use everyday that have machine learning in them?

Question 18: How might you use machine learning yourself (eg. to solve problems, for fun, etc.)?

Question 19: How confident do you feel about explaining or discussing machine learning with a non-technical person? (Please check one)

- ❑ 0 = No confidence
- ❑ 1 = Slight confidence
- ❑ 2 = Moderate confidence
- ❑ 3 = High confidence

Question 20: How confident do you feel about explaining or discussing machine learning with a machine learning expert? (Please check one)

- ❑ 0 = No confidence
- ❑ 1 = Slight confidence
- ❑ 2 = Moderate confidence
- ❑ 3 = High confidence

Question 21: How much do you agree with the following statement: Machine learning will make everyone’s life better. (Please check one)

- ❑ 0 = Strongly disagree
- ❑ 1 = Disagree
- ❑ 2 = Agree
- ❑ 3 = Strongly agree

Question 22: How much do you agree with the following statement: Machine learning is dangerous. (Please check one)

- ❑ 0 = Strongly disagree
- ❑ 1 = Disagree
- ❑ 2 = Agree

ID: ##### (for matching pre and post)

- ❏ 3 = Strongly agree

Question 23: How do you determine whether or not machine learning should be used to solve a problem?

Question 24: What do you think are the most important parts of a machine learning project?

Question 25: Would you recommend this workshop to any of the following groups? (Please check all that apply)

- ❏ Elementary school students
- ❏ Middle school students
- ❏ High school students
- ❏ Adults

Question 26: How can you see yourself using machine learning in the future? (Please check all that apply)

- ❏ Developing new machine learning algorithms or models
- ❏ Applying machine learning to new fields
- ❏ Using machine learning as part of an assignment for school
- ❏ Using machine learning as part of a job
- ❏ Using machine learning for fun
- ❏ Developing my Voice Identification project further
- ❏ Creating a new app using the Personal Audio Classifier
- ❏ Being able to talk about machine learning with non-experts
- ❏ Being able to talk about machine learning with experts
- ❏ I do not care much about using machine learning

Question 27: What types of audio classification models did you explore in the Personal Audio Classifier?

Question 28: Please provide any other general feedback on the Personal Audio Classifier interface.

Question 29: (Optional) Do you have any comments about this workshop? (What you liked, what you didn’t like, how you felt...etc.)

Bibliography

[1] Valerie Barr and Chris Stephenson. 2011. Bringing computational thinking to K-12: what is Involved and what is the role of the computer science education community? In *ACM Inroads 2, 1* (February 2011), 48-54. DOI=http://dx.doi.org/10.1145/1929887.1929905

[2] MIT App Inventor. Massachusetts Institute of Technology, 2015. Web. 03 Aug. 2016. <http://appinventor.mit.edu/explore/>.

[3] Krizhevsky, A. E., Sutskever, I. E., Hinton, G. E. (1970, January 1). ImageNet Classification with Deep Convolutional Neural Networks. Retrieved from https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks

[4] Yosinski, Jason, Clune, Jeff, Bengio, Yoshua, ... Hod. (2014, November 6). How transferable are features in deep neural networks? Retrieved from https://arxiv.org/abs/1411.1792

[5] Google. (n.d.). Teachable Machine. Retrieved from https://teachablemachine.withgoogle.com/

[6] Tang, D. (n.d.). Personal Image Classifier. Retrieved from https://classifier.appinventor.mit.edu/

[7] Hartquist, J. (2018, November 29). Audio Classification using FastAI and On-the-Fly Frequency Transforms. Retrieved from

https://towardsdatascience.com/audio-classification-using-fastai-and-on-the-fly-frequency-transforms-4dbe1b540f89

[8] Gupta, D. K., Nguyen, N. G., Khanna, A., Pandey, B., Tiwari, P. (1970, January 1). Sound Classification Using Convolutional Neural Network and Tensor Deep Stacking Network: Semantic Scholar. Retrieved from https://www.semanticscholar.org/paper/Sound-Classification-Using-Convolutional-Neural-and-Khamparia-Gupta/0b6cd03d1cc608aaeb8122ed25780d0e9a2e372f

[9] Boddapati, V., Petef, A., Rasmusson, J., Lundberg, L. (2017, September 1). Classifying environmental sounds using image recognition networks. Retrieved from https://www.sciencedirect.com/science/article/pii/S1877050917316599

[10] TensorFlow. (2018, September 27). Real-time Human Pose Estimation in the Browser with TensorFlow.js. Retrieved from https://medium.com/tensorflow/real-time-human-pose-estimation-in-the-browser-with-tensorflow-js-7dd0bc881cd5

[11] PyTorch. (n.d.). Retrieved from https://pytorch.org/

[12] TensorFlow. (n.d.). Retrieved from https://www.tensorflow.org/

[13] Microsoft Corporation (n.d.). Keras Retrieved from https://keras.io/

[14] TensorFlow.js: Machine Learning for Javascript Developers. (n.d.). Retrieved from https://www.tensorflow.org/js