CIS 4190/5190: Applied Machine Learning Fall 2024

Course Projects

Release Date: October 29, 2024

1 Introduction

Welcome to the course project for CIS 4190/5190 Applied Machine Learning! This project is designed to be both a structured learning experience and an opportunity for exploration. The project specifications outlined here are **not intended to be fully prescriptive**. Instead, there will be open-ended components both to how you will go about achieving the objectives specified in the description (i.e., **core questions**), and also to the objectives that you set for yourselves beyond what is in the document (i.e., **exploratory questions**). We hope that this project offers you the chance to deepen your understanding of machine learning concepts while also challenging you to explore new directions.

Each team (2-3 students) will work on one of three project directions:

A: Image2GPS: Predicting camera location based on the photo.

B: News Source Classification: Classifying the source of a news headline.

C: Audio-Based Material Classification: Classifying materials based on audio input.

The rest of this section describe the general structure shared among all project directions, and Section 2, 3, and 4 provide project-specific instructions.

1.1 Core Questions

The core questions are designed to ensure that you meet the essential requirements of the project. Each project has specific tasks that must be completed, and these are considered the foundation of your work. This description includes:

1. **a specification of the machine learning problem**: what is the task, what are the performance metrics of interest, how the data is to be collected
2. **a representative data sample** to help you understand what kind of data to collect, and what kind of data will be in the leaderboard test set. You will be able to compare against other teams on a leaderboard over the course of the project.
3. **code for training a baseline model** that you will use both as a way to evaluate your training data as you collect it, and as a starter code for the more sophisticated models that you will train over the course of the project

You will **submit your models to a leaderboard** periodically (more details to follow) over the course of the project, allowing you to measure your progress against other teams. This is intended as a fun, competitive exercise. High positions on the leaderboard do not guarantee

good grades, and low positions do not necessarily imply poor performance.

1.2 Exploratory Questions

Beyond the core questions, the exploratory questions provide an opportunity for you to engage with the project in a more open-ended and creative way. Investigating an exploratory question can involve experimenting with different approaches, testing hypotheses, or pursuing new directions that build on the core components of the project.

As part of the project, you will submit a **5-page report** documenting your approach to the exploratory questions, the experiments you conducted, and the insights you gained.

For your exploration questions, you will describe the following in the report:

- **Question and motivation:** Clearly define the question you aim to investigate, such as "Does random hyperparameter sampling perform better than grid search?" and explain why this question is relevant to your project.
- **Prior work or course material:** Review any related research or course material, and describe your prior expectation of the answer (before conducting experiments). For example, you can search Google Scholar for relevant research papers and condense them into a paragraph.
- **Methods for investigation:** Describe the methods you will use in your investigation. For instance, which hyperparameters will you explore, and how will you design your experiments?
- **Results and updated beliefs:** Present your findings and explain how your results have influenced your beliefs about the problem.
- **Limitations:** Discuss any limitations of your investigation, and describe what additional steps you might take if you had more time or resources.

1.3 Deliverables and Dates

At the end of the course, on **December 16**, you will submit the following:

- **Dataset:** The dataset you collected and used in your experiments.
- **Best-performing model:** The model that achieved the best performance based on your evaluation metrics.
- **Project report:** A report (5 pages) that details your approach to the core questions, as well as the investigations you performed and the progress you made towards answering them. We will release a report template soon.

2 Image2GPS

2.1 Problem Definition and Motivation

In this project, you will train a regression model to predict GPS coordinates from input images. The first step involves building a dataset that pairs images with their corresponding GPS labels. You will then use this data to train supervised computer vision models capable of estimating the location where each image was captured. This project will provide you with hands-on experience in standard computer vision pipelines and the opportunity to develop practical CV models for everyday use cases.

Problem Definition and Scope: The goal of this project is to build a computer vision model that could predict the GPS locations from any images taken from campus. To make it more feasible, we define the testing region to be on Penn's campus in between the rectangular region from 33rd and Walnut to 38th and Spruce st, as shown in Figure 1.

2.1.1 Performance metrics

In this section we give some details on how will the dataset and the model that you submit be tested on our end. The dataset testing is easier. Basically, whatever dataset you collect and submit should perform similar (i.e. provide non-trivial gains) on the baseline model to the toy dataset that we release. To rephrase, the baseline model should be good enough to provide non-trivial performance on your dataset. For the model that you submit, we will perform inference on the hidden test data and calculate the RMSE loss of your model. The RMSE loss will indicate the mean distance in meters that your model predictions are far from the actual GPS coordinates. This loss will be a factor that decides your position on the leaderboard. Our test data will include cases that try to test your model for robustness against distribution shifts. So, you should try to think and come up with cases that your model might perform bad on and how can you mitigate them. Of course we will not test on images pointing towards the sky ;) Since, you will have the baseline model with you, we expect that the data you will collect will be of high quality and with proper testing you should be able to score well in the data collection component :)

2.2 Representative Data Sample

You can find some examples of test data that will be used for evaluation here. You can use it as a resource for validation, but we do not recommend including it in your training set.

2.3 Code for Training a Baseline Model

You can find how we build a naive baseline method that just slightly outperforms trivial solutions (by only outputting the mean of latitude and longitude) in this notebook.

Essentially, we resize the image to 224 by 224 (**you should also resize the image to 224 $\times$ 224**) and normalize it. More importantly, we normalize the outputs by calculating

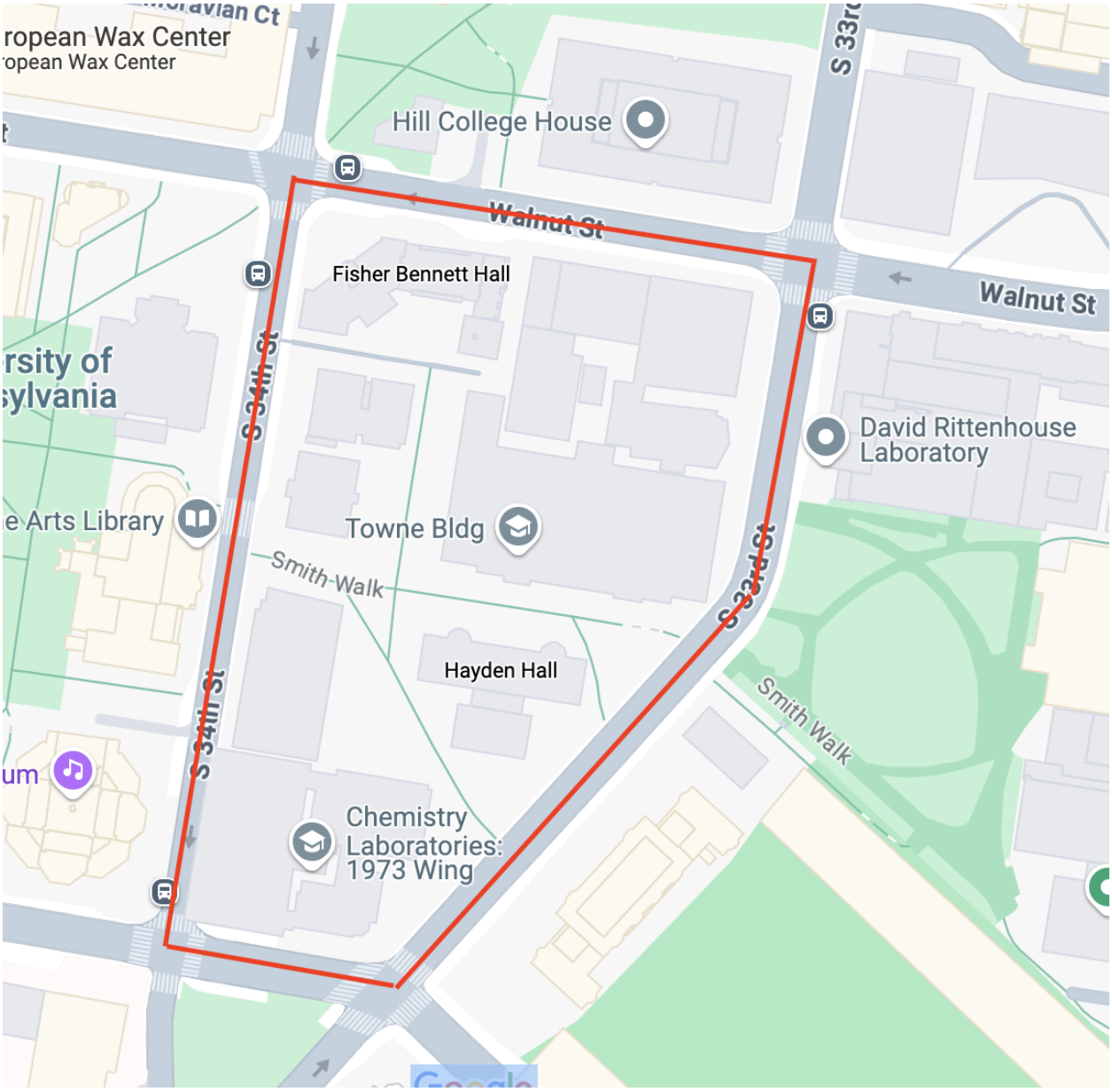


Figure 1: Test region for GPS prediction.

the mean and variance of the latitudes and longitudes (remember to scale it back after prediction).

We use a ResNet-18 as the backbone and modify the output of the last layer to have a dimension of two. We conduct full fine-tuning using MSE loss, the Adam optimizer with a learning rate of 0.001, and a scheduler that decreases the learning rate every few epochs. In total, we run 10-15 epochs to obtain baseline performance. Your model is expected to achieve performance at least as good as this baseline.

A reminder: make sure to set the Colab runtime to connect with GPUs. Note that you are also likely to run out of GPU allocation if you are using a free account.

2.4 Other Miscellaneous Resources

2.4.1 Data Collection

In modern computer vision tasks, models are often trained on large-scale image datasets available online, such as COCO and ImageNet. However, assembling a dataset that effectively supports your specific tasks can be more challenging than it appears. In the initial stage of this project, you will collect an image dataset with GPS labels. You will learn how to gather high-quality images suitable for this task, ensure consistency across different data collectors, and account for factors that may affect data distribution such as lighting and weather.
We will implement a standardized data collection procedure. First, the test dataset will consist exclusively of images taken along walkways, so you won't need to cover every inch of the campus. Second, it's important to note that each GPS location can correspond to multiple views. To ensure comprehensive coverage, you should capture several photos from different viewpoints at the same exact location. During data collection, we usually stand at a point and rotate to take eight pictures from various angles. When taking pictures, avoid zooming in or out, and always hold your phone upright.

2.4.2 Post Process and Extracting GPS Location

You can easily access the GPS location of the image that you captured through the EXIF data of the image. For this you will need to enable settings from your smartphone that allow you to capture/store the EXIF data of the picture that you take. We will give suggested approach for iPhone and Android (Samsung). Note that this may slightly differ for your smartphone but the general idea is the same. For iPhone, you will need to go to (Settings)—>(Privacy)—>(Location Services) and make sure that the application for Camera is given the access. For Android, the default camera application will have settings. You should open settings from inside the default camera app and make sure that the Location/Location tags option is selected. Note, you should always make sure that there is EXIF data collected for the picture that you capture first before starting to collect more images. To help you with this, please find the ipynb notebook. The notebook will extract the EXIF data (specifically, the latitude and longitude) from the image and store it in a CSV file. Feel free to create a copy of the notebook and change it to your requirements. This way you can create the

training labels for the image that you capture.

2.4.3 Uploading Data and Model Weights on Huggingface

After your team has collected data, the best practice is to upload your data to Huggingface, along with the images and their associated latitude and longitude. This way, everyone in your team, as well as the TAs, can access your data using a simple API call. After training your model using this data, you will also need to upload your model weights to Huggingface as part of your deliverables.

The first step is for one of your teammates (as the admin) to create an organization on Huggingface, and then invite your teammates and TAs to join the organization (go to Organization Settings → Members → Add New Member → Set Role to 'Write' → Fill in the username). After following the steps described in Section 2.4.2, you will need to follow the steps in this notebook to push your dataset and load it.

After training your model and when you are ready to submit it for the leaderboard, push your model weights to Huggingface by following the end of this notebook. Note that if you use a model from `TORCHVISION.MODELS`, additional steps are required to push it (though you can still follow the steps described in the notebook). Therefore, it is highly recommended to use models from Huggingface directly, as you will most likely find the model architecture there. Of course, you are also encouraged to build a vision model using PyTorch from scratch, following the steps described here.

After uploading your model, make sure to enable access requests and select "Manual request" for new requests. This ensures that people from other teams **will not** have access to your model weights.

It is **crucial to ensure that the TAs have the correct access**. It is your responsibility to verify that we have the proper access. Please remind us if we accidentally missed the invite.

2.4.4 TA Contact Points

- Harshwardhan Manoj Yadav (hyadav@seas.upenn.edu)
- Jianing Qian (jianingq@seas.upenn.edu)
- Guangyao Dou (gydou@seas.upenn.edu)

3 News Source Classification

3.1 Problem Definition and Motivation

In this topic, you will work to scrape the News Headlines dataset and train on a Binary Text Classification task using news headlines from two prominent news outlets: **Fox News** and **NBC News**. The goal is to create some machine-learning models that classify news based on their headlines. We will provide a baseline model with lower accuracies, 3815 URLs of 3815 news (so 2010 from FoxNews, 1805 from NBC) for you to start Headlines scraping, and a screenshot of the first few lines of hidden test data that we will use to test your models. Your task will be to collect a dataset, process it, and experiment with various models to improve classification performance as best as you can. To show and summarize your improvement, in the final report, you will submit one or several line charts of your models' metrics, including metrics of the baseline model.

Here is the dataset of links to articles from NBC News and Fox News: **Dataset**.

3.1.1 Performance metrics

For your final model performance, we will evaluate your models on news headlines scrapped from NBC News and Fox News after the deadline submission. As such, your model will not have access to the testing set when you are training/validating your model. In the sections below, we describe what the data looks like and give you tips on how to clean the data, as well as how to do the webscrapping.

3.2 Representative Data Sample

Figures 2 and 3 are the examples of the test data for NBC and Fox News that we will use.

D	E
alternative_headline	
Iranian President Raisi is killed in helicopter crash	
Kristen Cavallari and Jay Cutler to divorce after 10 years together	
Why Atlanta spa shooter's Asian 'acquaintances' can't tell us much about his racial biases	
The best TV streaming services in 2024	
Mike Johnson won't commit to bringing House back before the election for more hurricane relief	

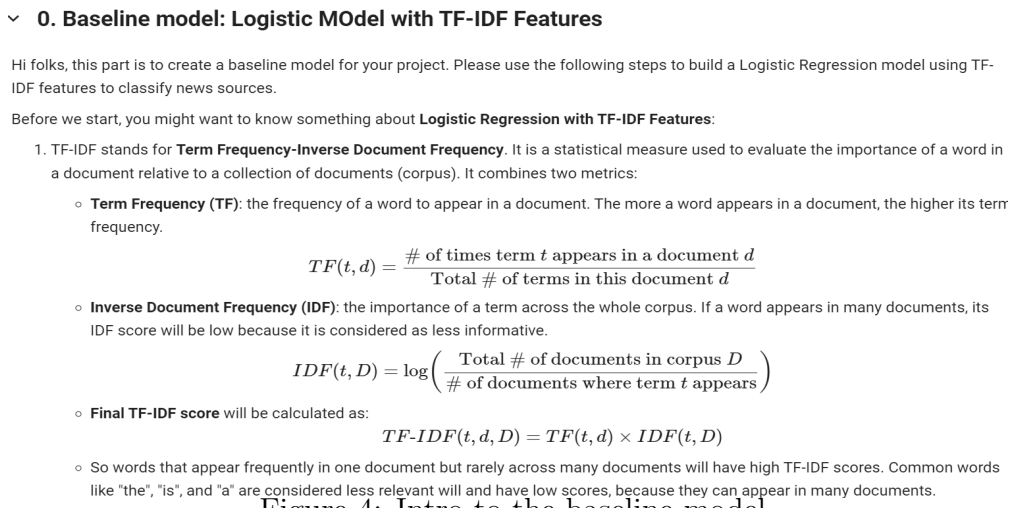
Figure 2: NBC Example Headlines

3.3 Code for Training a Baseline Model

In the following figures, please refer to the explanation and the code of the baseline model to reproduce this model. It should have an accuracy of 66.49% and all other metrics. You should build models than this baseline model.

C
scraped_headline
Wisconsin dairy farmer says 'no question' Trump admin was 'much better' than Biden-Harris
Apalachee High School shooting suspect Colt Gray and father appear in court for separate hearings
Bruce Willis' daughter says he's shown her 'to not take any moment for granted'
Most Irish cities in US and their beloved pubs
Harris shredded for resurfaced video of promising to close migrant detention centers

Figure 3: Fox News Example Headlines



0. Baseline model: Logistic MOdel with TF-IDF Features

Hi folks, this part is to create a baseline model for your project. Please use the following steps to build a Logistic Regression model using TF-IDF features to classify news sources.

Before we start, you might want to know something about **Logistic Regression with TF-IDF Features**:

1. TF-IDF stands for **Term Frequency-Inverse Document Frequency**. It is a statistical measure used to evaluate the importance of a word in a document relative to a collection of documents (corpus). It combines two metrics:
   - **Term Frequency (TF)**: the frequency of a word to appear in a document. The more a word appears in a document, the higher its term frequency.

$$TF(t,d) = \frac{\#\text{ of times term } t \text{ appears in a document } d}{\text{Total } \#\text{ of terms in this document } d}$$

   - **Inverse Document Frequency (IDF)**: the importance of a term across the whole corpus. If a word appears in many documents, its IDF score will be low because it is considered as less informative.

$$IDF(t,D) = \log\left(\frac{\text{Total } \#\text{ of documents in corpus } D}{\#\text{ of documents where term } t \text{ appears}}\right)$$

   - **Final TF-IDF score** will be calculated as:

$$TF\text{-}IDF(t,d,D) = TF(t,d) \times IDF(t,D)$$

   - So words that appear frequently in one document but rarely across many documents will have high TF-IDF scores. Common words like "the", "is", and "a" are considered less relevant will and have low scores, because they can appear in many documents.

Figure 4: Intro to the baseline model

```
# 1. Load the CSV files & Preprocess the data
# csv_file_path = '/merged_news_data.csv'
# news _df = pd.read_csv(csv_filePath)
# ... ...

# 2. Split the data into training and testing sets
# (80% train, 20% test)
# ... ...

# 3. Convert the labels to binary values (0 for 'FoxNews', 1 for 'NBC')
y_train = y_train.apply(lambda x: 1 if x == 'FoxNews' else 0)
y_test = y_test.apply(lambda x: 1 if x == 'FoxNews' else 0)

# 4. Convert the text data to TF-IDF features
vectorizer = TfidfVectorizer(stop_words='english', max_features=100)
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)

# 5. Train a Logistic Regression model
model = LogisticRegression(max_iter=100)
model.fit(X_train_tfidf, y_train)

```

```
# 6. Make predictiosn on the test set
y_pred = model.predict(X_test_tfidf)

# 7. Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.4f}")
print("Classification Report:\n", classification_report(y_test, y_pred)
    )

## Result
# Accuracy: 0.6649
# Classification Report:
#               precision    recall  f1-score   support
#            0       0.69       0.54      0.60       358
#            1       0.65       0.78      0.71       400
#
#     accuracy                           0.66       758
#    macro avg       0.67       0.66      0.66       758
# weighted avg       0.67       0.66      0.66       758
```

3.4 Other Miscellaneous Resources

This dataset is a .csv file that contains links to Fox News at the beginning and NBC News at the end. To read this dataset in Python, you can use the `pandas` library. Your task is to web scrap the headlines from these links. For those new to web scrapping, please follow the steps below. The Python libraries that are most helpful are `BeautifulSoup` and `request`. To web scrape, you identify the HTML tabs that contain the text that you want. In the example below, we are scrapping the headline which is in a `<h1>` tag with the class name `"headline speakable."` Here is a sample Python code to web scrape:

```
import requests
from bs4 import BeautifulSoup

url = https://www.foxnews.com/sports/juan-soto-sends-yankees-world-
    series-first-time-15-years # example website

response = requests.get(url)
if response.status_code != 200:
  raise Exception(f"Failed to load page:
  Status code {response.status_code}")

# Parse the HTML content with BeautifulSoup
soup = BeautifulSoup(response.text, "html.parser")

title = soup.find("h1", class_="headline speakable")

# title should be the following
```

```
17 # Juan Soto sends the Yankees to the World Series for the first time in
       15 years
```

You are also encouraged to webscrap and collect your own headlines from Fox News and NBC News.

3.4.1 Cleaning Process Suggestions

After scraping the headlines, consider further cleaning to make sure your data is ready for training. Some possible operations are listed, but they are not required or no guarantee of performance improvement. You are highly encouraged to explore on your own and incorporate more advanced techniques:

- **Data Cleaning**: Remove unwanted elements such as HTML tags, scripts, and special characters. Address unnecessary spaces, tabs, and newline characters to ensure consistency. Decide how to handle punctuation marks within the headlines.
- **Normalization**: Standardize the text by converting it to lowercase or uppercase. Remove common stopwords that may not add significant value. Apply stemming or lemmatization to reduce words to their base forms.
- **Handling Missing or Incomplete Data**: Detect any missing or incomplete headlines in your dataset. Choose methods to handle these gaps, such as removing incomplete entries or imputing missing values.
- **Consistency Checks**: Ensure all headlines follow a consistent format. Identify and address duplicate headlines to maintain the integrity of your dataset.
- **Data Validation**: Manually inspect a subset of processed headlines for quality assurance. Calculate metrics such as average headline length or word frequency distributions to understand your dataset better.
- **Documentation and Tracking**: Maintain detailed notes on the preprocessing steps applied. Use version control systems like Git to manage changes in your preprocessing workflows.

TA Contact Points: Zelong Wang, Haorui Li, Wendy Deng, Chandler Cheung

4 Audio-Based Material Classification

4.1 Project Overview

Sound-based material classification is an intriguing area of research. Audio sensors are generally more affordable, easier to install, and have lower power requirements than cameras, making them attractive for various applications. In industrial settings, this technology can be utilized for quality control and defect detection in manufacturing processes, material sorting and recycling in waste management, and detecting hazardous materials or contaminants. These real-world applications showcase the potential impact of sound-based classification systems and motivate the exploration of robust, accurate models in this domain. In this project, you will have some hands-on experience of building a simple audio-based classification system for materials in Levine Hall.

As a team of 2-3 people, you will collect (record) the dataset to explore the end-to-end data processing pipeline and understand how data quality and data drift can impact the model's performance. Additionally, you will have to conduct a literature review to learn the best approaches for audio-based classification and apply the knowledge gained from this class to develop and improve the model's performance.

4.2 Problem Definition and Motivation

We frame this as a multi-class classification problem, where a model uses audio input to classify the object. The dataset will consist of .wav audio files as inputs and object classes as labels. To make the problem tractable, we are restricting the classes to 7 different objects, all easily accesible on the Penn campus, as seen in figure 5. Even within this fixed distribution, there are many different ways to elicit sounds from objects, making this problem challenging.

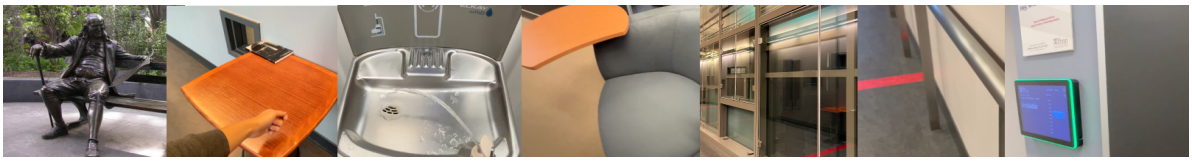


Figure 5: The 7 objects.

4.2.1 Performance metrics

We will evaluate the models based on their classification accuracy on held out test data. We will have two different test datasets. The first is the in-distribution test set, where the audio samples are collected according to section 4.3. The second dataset is an "in-the-wild" test datset, where audio samples are collected from objects that are within the same class but are not the exact objects mentioned in section 4.3. For example, we may record audio from a water fountain in another building, or use an audio recording from the internet.

4.3 Data Collection Instructions & Submission Requirements

We will provide some starting code, videos, and audio snippets to get you started. See link here for example audio snippets and videos showing how we knocked on the objects. Use these to make sure your data collection procedure is somewhat similar to ours, and that your model can at least classify the samples collected by us.

We required a standardized data collection procedure. First, in one hand, the user holds a mobile phone. Then with the other hand, the user knocks the object with their knuckles, which should produce an audible sound. Some objects that cannot be hit, *e.g.* a water fountain, can just be passively recorded. The length of the audio recording should be 1 second, and there should only be 1 knock.

Here are the 7 objects and details on where to find them.

- Ben Franklin statue - Google Map "Ben on the Bench"
- Blackboard (not white board) - Levine 4th floor bump space
- Table - Levine 4th floor near window (turn left from the elevator)
- Glass window - Levine 4th floor (turn left from the elevator)
- Handrail - Levine 4th floor (turn left from the elevator)
- Water fountain - Levine 4th floor
- Sofa - Levine 4th floor bump space

See the data collection videos for their exact locations and reach out on Ed if you are unclear where is the object.

Tips: You might be wondering, "What if the sound I record for the same object differs from the sound my teammates record, or from the sound that the CIS 5190 TA team records (which will be used for evaluation)?" This is exactly how real-world machine learning problems work! The dataset may be noisy and come from different distributions. Even if the sound is collected from the same person, different times of day may introduce varying levels of background noise. You will need to address this issue in your project and discuss your strategies for managing it roughly in project proposal and in greater detail in the final report. A key strategy is to ensure that your dataset is diverse enough to generalize well on hidden test set. While there is no specific requirement for the number of samples per class you have to collect, consider this from the perspective of the bias-variance trade-off, which will depend on the complexity of the model you choose.

We will release some example test data that will be used for evaluation, so use it as a resource for validation, but we do not recommend including it in your training set.

Submission Requirements You're required to submit a dataset of all the sounds you have collected.

The folder structure for your dataset should mimic the following structure.

```
ben/
    ben-1.wav
    ben-2.wav
    ...
blackboard/
    blackboard-1.wav
    blackboard-2.wav
    ...
glass/
    glass-1.wav
    glass-2.wav
    ...
railing/
    railing-1.wav
    railing-2.wav
    ...
sofa/
    sofa-1.wav
    sofa-2.wav
    ...
table/
    table-1.wav
    table-2.wav
    ...
water/
    water-1.wav
    water-2.wav
    ...
```

While there is no specific requirement for the number of data points, it should be sufficient to perform non-trivially on the baseline model we provide (make sure to test this, as we will also test it) and also on your own model. Non-trivially means better than random guessing. Please follow the folder structure to submit the final dataset.

4.4 Code for Training a Baseline Model & Model Submission Requirements

Next, we trained a simple baseline model (link here) using multi-class logistic regression. This baseline achieves around around 50% accuracy on unseen data, which is somewhat better than a random chance classifier ($\frac{1}{7}$). There remains ample room for improvement, and you are not required to build off of the starter code.

This submission will deal with the modelling side of machine learning. Feel free to explore as much as you can on this part by reading literatures or finding resources online to train a better model. You will submit a model capable of doing multi-class classification of audio inputs through a google colab file.

We will provide skeleton code for the Colab notebook (link here) from which you are expected to fill in. You are free to train your models elsewhere, as long as you can load the weights and do the evaluation in the provided colab notebook.

4.5 Other Miscellaneous Resources

Userful Resources:

1. Sample evaluation data and video of data collection process: link here.
2. Notebook of the baseline model the TA team trained: link here

3. Skeleton code: link here

TA Contact:

1. Daniel Alexander (alexdan@seas.upenn.edu)
2. Edward Hu (hued@seas.upenn.edu)
3. Tongtong Liu (liufrank@seas.upenn.edu)