CIS 4190/5190: Applied Machine Learning Spring 2025

Project Instructions

March 17, 2025

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, and also to the objectives that you set for yourselves beyond what is in the document. We hope 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, ideally 3) will work on 1 of 2 project directions:

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

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

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

1.1 Grouping Google Form

After you form your group of 2-3 members, your group will fill out one (and only one) submission of the Google Form to show:

- Your team members.
- Which of the two topics do you want to choose.

1.2 Project Report

As part of the project, you will submit a **report of 3 pages for the check-in and 5 pages for the final report** to document your approach, the experiments you conducted, and the insights you gained. Your report must address the following **core questions**:

- Describe the procedure and protocol for **data collection**, what considerations went into these, how you curated or cleaned up your datasets, how you split the datasets for internal evaluations (if you did), and what your final dataset ended up looking like.
- Describe the design considerations for the model, the iterative process through which you tried to improve those designs, and what your final **model design** ended up looking like.

- Describe the evaluation protocols and results (how you evaluated model iterations, what performance metrics you used, and how you chose models to submit to the leaderboard)

It must also address the following **exploratory questions**:

- **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. (It can be short descriptions of your expectations in the Proposal.)
- **Limitations:** Discuss any limitations of your investigation, and describe what additional steps you might take if you had more time or resources.

These questions are deliberately open-ended, and require you to engage with the project in a creative way. Investigating each question can involve experimenting with different approaches, testing hypotheses, or pursuing new directions that build on the core components of the project.

1.3 Milestones and Deadlines

Milestones and Deadlines

- **Project Milestone 1 - Grouping Google Form**: due Wednesday, Mar 26, 2025, 11:59pm
- **Project Milestone 2 - Status Check-In**: due Wednesday, Apr 23, 2025, 11:59pm (no late submissions)
- **Project Milestone 3 - Colab Notebook, Report & Summary Slides**: due Sunday, May 4, 2025, 11:59pm (no late submissions)
- **Project Extra Credit - Video Demo**: due Sunday, May 4, 2025, 11:59pm (no late submissions)

1.4 Grading Breakdown

- Grouping Google Form: 5%
- Status Check-in: 15%
- Final summary slides: 10%
- Final report and Colab notebook: 70%

1.5 Additional Information

You will also be required to **submit your Colab notebooks / Reports of models to Grade-scope**. 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 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 engineering block, as shown in Figure 1.

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 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. Additionally, the use of Hugging Face is completely optional, but highly recommended!

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 effec-

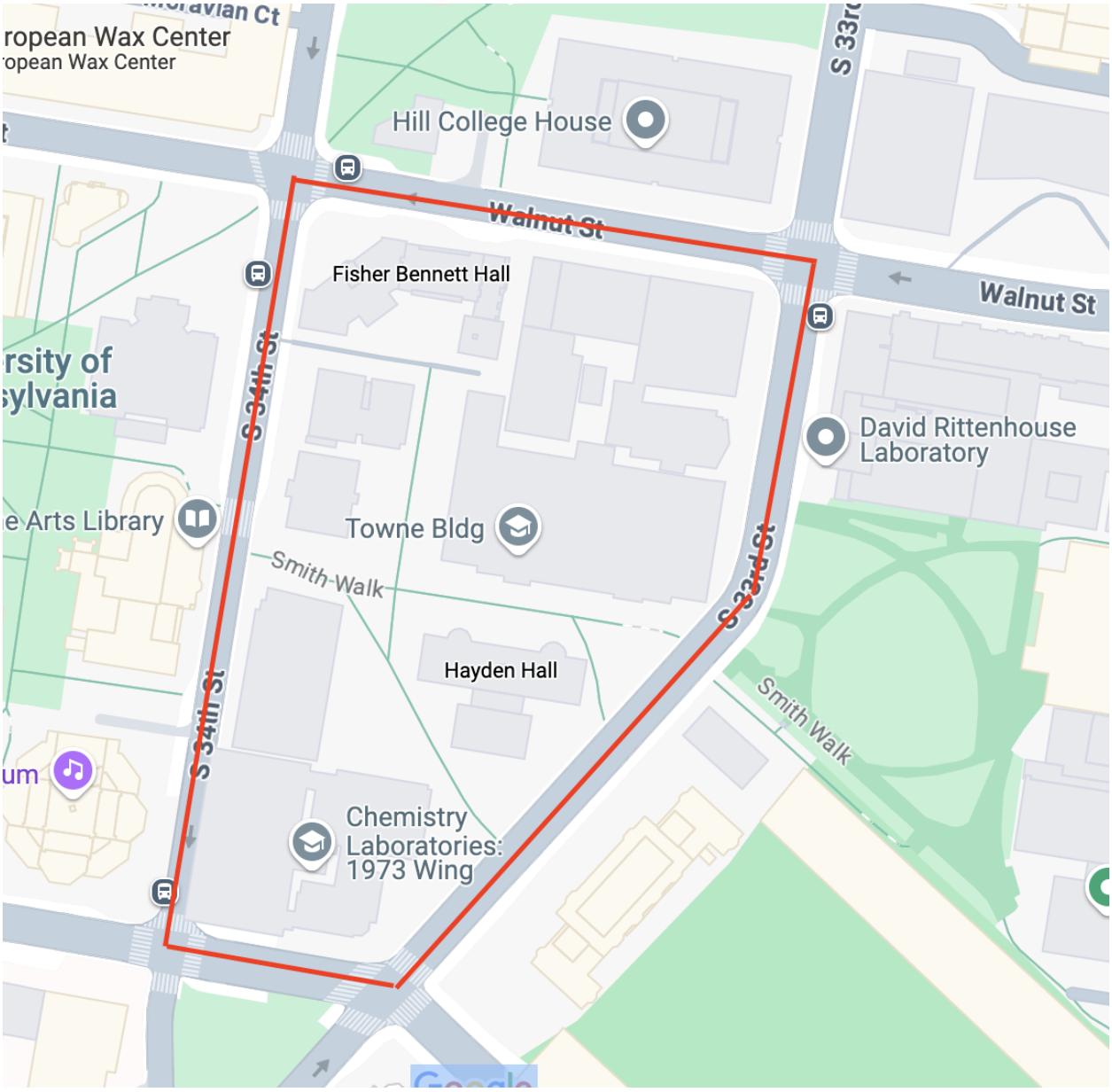


Figure 1: Test region for GPS prediction.

tively 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.

Your model must be able to run inference on this example dataset with 224×224 inputs and output the average distance for these images (This is the only requirement). See how you can compute the mean squared error loss and distance metrics here.

2.4.3 TA Contact Points

- Guangyao Dou (gydou@seas.upenn.edu)
- TBD

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, 3805 URLs of 3805 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 you will use later 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 URLs to articles from NBC and Fox News: **Dataset**. And this is a random subset of the hidden test set, with 10+10=20 news titles: **Sub-Testset**. Your model must be able to run inference on this subset so that it will also work on the hidden test set: **Example Test Colab**.

3.2 Representative Data Sample

Figures 2a and 2b are the examples of the test data for NBC and Fox News that we will use. For the test dataset, the text is directly from the news article, so there has been no preprocessing or cleaning of the data.

Note: You should name news outlets as 'FoxNews' & 'NBC', and set 1 for 'FoxNews' & 0 for 'NBC'.

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	

(a) NBC Example Headlines

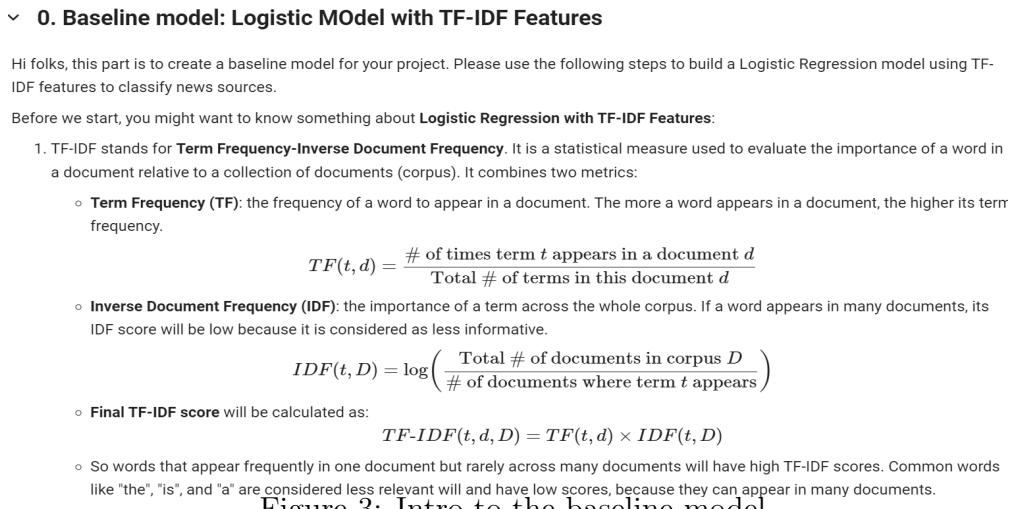
C
scraped_headline
Wisconsin dairy farmer says 'no question' Trump admin was 'much better' than Biden-Harris
Apalachee High School shooting suspect Colt Gray and father appear in court for separate hearings
Bruce Willis' daughter says he's shown her 'to not take any moment for granted'
Most Irish cities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

(b) Fox News Example Headlines

Figure 2: Example Headlines from NBC and Fox News

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 would have an accuracy around 66.49% and all other metrics. You should build models than this baseline model.



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 3: 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 (1 for 'FoxNews', 0 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
# Juan Soto sends the Yankees to the World Series for the first time in
    15 years
```

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.

For the test dataset, the text is directly from the news article, so there has been no pre-processing or cleaning of the data. For evaluation, we expect you to set up the pipeline for reading in the test data, pre-processing the data, and evaluating the model on the data. We did not give the preprocessing pipeline because you can have different ways to clean the data, and the model is trained specific to that method.

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.

3.4.2 Your Own Evaluation Part

Please make sure your evaluation code is included and clearly explained, or otherwise it would be hard for us to be able to evaluate.

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. Additionally, the use of **Hugging Face is completely optional, but highly recommended!**

You can cite this part in 3.1: The dataset of URLs: **Dataset**. The random subset of the hidden test set, with 10+10=20 news titles: **Sub-Testset**. Your model must be able to run inference on this subset to work on the hidden test set: **Example Test Colab**.

Huggingface Guideline (if you want to use Huggingface for saving your pre-trained models).

3.4.3 TA Contact Points

- Haorui Li (haoruili@seas.upenn.edu)
- Manurag Khullar (manurag@seas.upenn.edu)
- TBD