**Machine Learning Homework 1: ANN & KNN on SIFT Features**

**Assignment Goal**

The main objective of this assignment is to **build, implement, and compare Approximate Nearest Neighbor (ANN) and K-Nearest Neighbor (KNN) models** for tasks involving image keypoint matching and classification using SIFT features.

**Introduction**

In this assignment, you will work with datasets generated using the **SIFT (Scale-Invariant Feature Transform)** algorithm, a classical method for detecting and describing local features in images.

SIFT is used in image matching applications. It scans an image, detects distinctive keypoints, and describes the area around each keypoint using a 128-dimensional vector.

The goal of this assignment is to implement an algorithm based on an Approximate Nearest Neighbor (ANN) structure, which receives two images of roughly the same scene and attempts to find matching keypoints between them. The first image is used to build the ANN structure, and then each point in the second image is matched to its nearest neighbor from the first image based on descriptor similarity.

However, not every point in the second image has a true match in the first image. Reasons include non-overlapping image areas or non-distinctive points.

To filter matches, you will use the **ratio test**: find the two nearest neighbors, calculate the ratio of their distances, and only accept a match if the ratio is small (e.g., less than 0.8).

**Note:** The ratio provided (0.8) is only an example. It is recommended you try different values and pick the one that works best. Report your findings.

Example pseudocode:

nearest\_n, second\_nearest = kneighbors(sample, k=2)

ratio = nearest\_n\_distance / second\_nearest\_distance

if ratio < 0.8:

return nearest\_n, nearest\_n\_distance

return None, nearest\_n\_distance

This method helps reduce false matches caused by ambiguous descriptors. A small ratio suggests the closest neighbor is clearly better than the next closest one.

We evaluate the performance of our matching algorithms in two ways:

* **Runtime**: Compare the time it takes to run kneighbors() using ANN vs a linear KNN search.
* A black square with black text

  AI-generated content may be incorrect.**Accuracy**: Since ANN may return an approximate match, we use the following formula to compute the average distance ratio error:

Where:

* dNN(Pi) is the distance to the exact nearest neighbor (using brute-force KNN).
* dANN(Pi) is the distance to the ANN-estimated neighbor.
* Lower ε means better approximation.

Each keypoint also contains:

* X, Y: its sub-pixel position in the image
* Scale: the size of the region around the keypoint
* Angle: the orientation relative to the image
* Response: the distinctiveness of the keypoint (higher = more distinctive)

SIFT data is used in tasks like image matching, structure from motion, and object recognition. In this assignment, you will:

* Match keypoints between images
* Build and compare ANN and KNN models
* Perform image classification using keypoints

**Folder Structure & Dataset Format**

You are given a folder containing:

* .csv datasets for each image: includes keypoints and 128 SIFT descriptors
* Corresponding images (for visualization purposes only)

**Dataset format (CSV):**

Y, X, Scale, Angle, Response, Feature\_1, Feature\_2, ..., Feature\_128

For classification and points matching you would need to use only the 128 SIFT features. The rest of the columns are for visualization and you may use the Response columns to make your predicionts more accurate, though you will need to preprocess it and use it differently from the 128 SIFT features.

**Migdal Datasets:**

* migdal\_1\_sift\_dataset.csv and migdal\_2\_sift\_dataset.csv
* Corresponding images: migdal\_1, migdal\_2

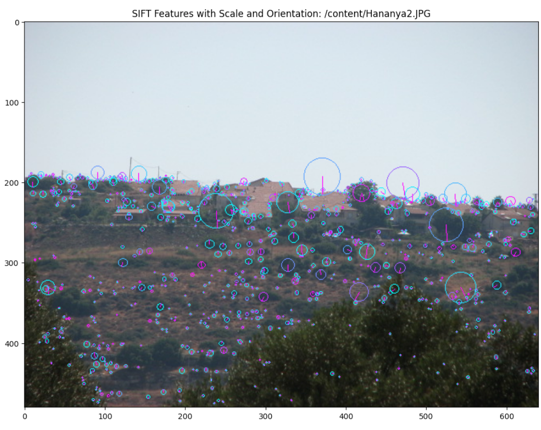
**Landmarks Datasets (for Part E):**

* Each landmark has two images and datasets:
  + e.g., statue\_of\_liberty\_1\_sift\_dataset.csv and statue\_of\_liberty\_2\_sift\_dataset.csv
* These datasets are in a separate folder.

**Data Exploration (5 pts)**

Provide a **short, concise** explanation

Before building models, perform a short data exploration on migdal\_1\_sift\_dataset.csv using methods we learned in class:

* Print .head() and .describe()
* Show range/distribution of Response, Scale, and some SIFT features
* Visualize the keypoints on migdal\_1.png using:
  + X and Y as position
  + Scale as radius
  + Angle as rotation/direction (use quiver or other methods)

For example:

**Part A: Model Implementation (35 pts)**

Provide a **concise** explanation of your approach and implementation.

Implement the following models:

1. **KNN** (linear search)
   * *Hint:* you can avoid Python loops by vectorizing distance computations via matrix multiplications.
2. **Abstract ANN class** (ANNBase) with required methods:
   * fit()
   * kneighbors()
3. **Two ANN models inheriting from the abstract class:**
   * LSH\_ANN
   * RKDT\_ANN

**Tip:** Handle cases where there are not enough near neighbors

**Constants**:

* For LSH: K = number of cuts, L = number of hash tables
* For RKDT: N0 = max points per leaf, L0 = number of trees

Use constants in your code to allow easy adjustments later. All models should support arbitrary k values for nearest neighbor search.

**Part B: KNN Nearest Neighbor Search (10 pts)**

Provide a **short, concise** explanation of your approach and results. Use migdal\_1\_sift\_dataset.csv as the training set and migdal\_2\_sift\_dataset.csv as the query/test set.

* Run KNN (k=1) to find the closest match for each test keypoint
* Store distances and matched indices
* Measure and print runtime of distance calculations
* Visualize the **top 10 best matches** (lowest distances) on the images using pillow library - see example below

For example:

**Part C: ANN Models + Grid Search (30 pts)**

Provide a **concise** explanation of your approach and results. Unless otherwise noted, tune and report the **best K** for this task.

Use both ANN models (LSH and RKDT) on the migdal\_1/2 datasets.

1. **Train both ANN models** using migdal\_1\_sift\_dataset.csv as the training set.
2. **Query** the models using migdal\_2\_sift\_dataset.csv to find the nearest neighbors for each test point.
3. Perform a **grid search**:
   * Try 10 different combinations of ANN configurations (each defined by the model type and its hyperparameters):
     + For LSH: pairs of K (cuts) and L (hash tables)
     + For RKDT: pairs of N0 (max points per leaf) and L0 (number of trees)
   * For each configuration:
     + Train the ANN model on migdal\_1
     + Use migdal\_2 as query
     + Measure runtime
     + Apply the ratio test (as explained in the Introduction) using k=2: only return the nearest match if the ratio d1 / d2 < 0.8 (or another value you’ve tried)
     + Compute the **average distance ratio error** for that configuration
     + Save the best results per model to use as comparison in Part D
   * Visualize how the hyperparameters affected runtime and accuracy
     + Visualize separately for runtime and for accuracy (at least two plots per model architecture)
     + Use a visualization method of your preference.
     + Hint: [How to create a 3d plot using matplotlib (link)](https://www.geeksforgeeks.org/three-dimensional-plotting-in-python-using-matplotlib/)
4. Choose the best-performing parameter set (lowest error):
   * Visualize the **top 10 best matches** using Pillow, overlaying the matched keypoints on the images
   * Mark corresponding keypoints with the same color
   * Based on the visualization, indicate which matches are valid and which are incorrect
5. Compare the performance of your two ANN models with the KNN baseline:
   * **Accuracy**: In this task, you do not have ground-truth labels for each keypoint, so accuracy is measured using the average distance ratio error formula shown above ( Introduction -> accuracy bullet).
   * **Runtime**: Time the search process
   * Present both comparisons graphically (bar charts, line plots, etc.)

Use only the methods and metrics we studied in class. Clear, concise visualizations are essential.

**Part D: Compare with sklearn NearestNeighbors (5 pts)**

Provide a **short, concise** explanation of your results.

* Use sklearn.neighbors.NearestNeighbors (KDTree backend)
* Train on migdal\_1, query migdal\_2
* Compare:
  + Accuracy
  + Runtime
* Visualize comparison to KNN, LSH, and RKDT with 2 plots we learned in class.
* Use previously saved results for comparison

**Part E: Place Recognition (30 pts)**

Provide a **short, concise** explanation of your approach and results. Unless otherwise noted, tune and report the **best K** for this task.

You are given a folder with 11 landmarks. Each has two images:

* One for training (e.g., statue\_of\_liberty\_1\_sift\_dataset.csv)
* One for testing (e.g., statue\_of\_liberty\_2\_sift\_dataset.csv)

Additionally, for this task, you must include migdal\_1\_sift\_dataset.csv as an **extra test image**, even though it does not belong to the original 11 landmark classes.

**Goal:** Classify each test image to its correct landmark by using SIFT features and nearest-neighbor voting.

**Note**: There is no corresponding training data for migdal\_1 — your model is expected to identify that it **does not belong to any known class**.

**What to do:**

* First, train both of your ANN models (LSH and RKDT) on all \_1 datasets.
* Then, for each \_2 test dataset, predict its class based on keypoint-level neighbor voting.

You must implement predict() method in both of your ANN models.

Example: How the predict() Function Could Be Implemented

* The function receives a test dataset (list of keypoints) which represent a landmark image from different angle and time your models trained on.
* For each keypoint:
  + Use kneighbors() to find its k nearest neighbors among all training points (from all \_1 datasets)
  + If the closest neighbor is farther than a given threshold you will decide, you may choose to ignore the point.
  + Otherwise, assign the keypoint the label of its closest neighbor (or majority label if k > 1)
* The predicted label for the image is the most frequent label among all valid keypoint predictions
  + In case the image does **not match confidently to any known landmark** (e.g., migdal\_1), your model should output a message such as: "No matching landmark found for this image" or similar.

You do **not** need to apply the ratio test in this task.

You may choose the value of k, and optionally perform a grid search to optimize it.

**Evaluation & Comparison:**

* Use both ANN models you have created (LSH and RKDT) to classify the test images
* For each prediction, also compute a **confidence score** (votes\_for\_prediction / total\_valid\_votes)
* Compare their results using:
  + Classification **accuracy** (image-level)
  + **Macro F1 score**
  + Confusion matrix (include a class for **"no match"**)
  + A **table** showing true vs predicted labels
  + Visuals showing prediction confidence (e.g., bar chart of keypoint votes)

Use only the techniques and metrics discussed in class.

**Bonus: Top-Performing Place Recognition**

* The 10 groups with the best classification (Part E) in both speed and accuracy (confidence score) will each earn +5 pts.  
  The next 10 best will get +3 pts, and the 10 after that +1 pt

**Submission Guidelines**

* Submission is in pairs only; for exceptional cases, please email the teaching assistants.
* Use only Jupyter Notebook
* You are encouraged to use any libraries we covered in class (e.g., pandas, matplotlib, numpy, etc.). Their use is highly recommended for data handling and visualization. If you use other libraries, explain their purpose and where they’re used.
* Submit both .ipynb and .html
* All code must run without errors or warnings
* Use markdown cells for explanations and conclusions — make them clear and concise. Explanations that are too short or too long will receive a lower score. Aim for balance: concise, informative, and to the point.
* Add comments and documentation to clarify the code
* Visualizations must include:
  + Axis labels
  + Titles
  + Legends (if needed)
  + A short explanation in markdown of what can be learned from the visualization
* Do not use ML libraries other than sklearn in Part D
* Use meaningful variable names and constants
* Questions may be asked in the Moodle forum. Questions asked by email will not be answered. Before you submit a question, make sure it hasn’t already been answered.

**Good Luck!**