Image Matching Challenge

Longbin Jin

Artificial Inteligence
& Computer Vision
Laboratory

Schedule of Lectures

⬜ offline 🟦 online

	Mon		Wen		
12주	Image matching		🟦 Hands-on CNN		
13주	CNN		CNN		
14주	CNN architecture		🟦 Semantic segmentation		
15주	🟦 Object Detection		**Image matching Challenge (40%)**		
16주	**Final exam (50%) 6.17**				

Recap: Image Matching System

Image dataset

Preprocessing
- Point processing
  - Gray scale / HSI
  - Contrast stretching
- Area processing
  - Noise filtering
  - Edge
  - Sharpening
  - Morphological
- Deep Learning

Feature extraction
- Color
  - Color histogram
- Texture
  - LBP
  - GLCM
  - Law's texture
- Shape
  - Harris corner
  - SIFT
  - HoG
- Deep Learning

Classification
- Similarity
  - Distance
  - Cosine similarity
- ML classifier
  - KNN
  - SVM
  - … …
- Deep Learning

Image Matching System Challenge 1 & 2

Image dataset

Preprocessing

- Point processing
  - Gray scale / HSI
  - Contrast stretching
- Area processing
  - Noise filtering
  - Edge
  - Sharpening
  - Morphological
- Deep Learning

Feature extraction

- **CHALLENGE 1**
- Low/mid-level
  - Color
  - Texture
  - Shape

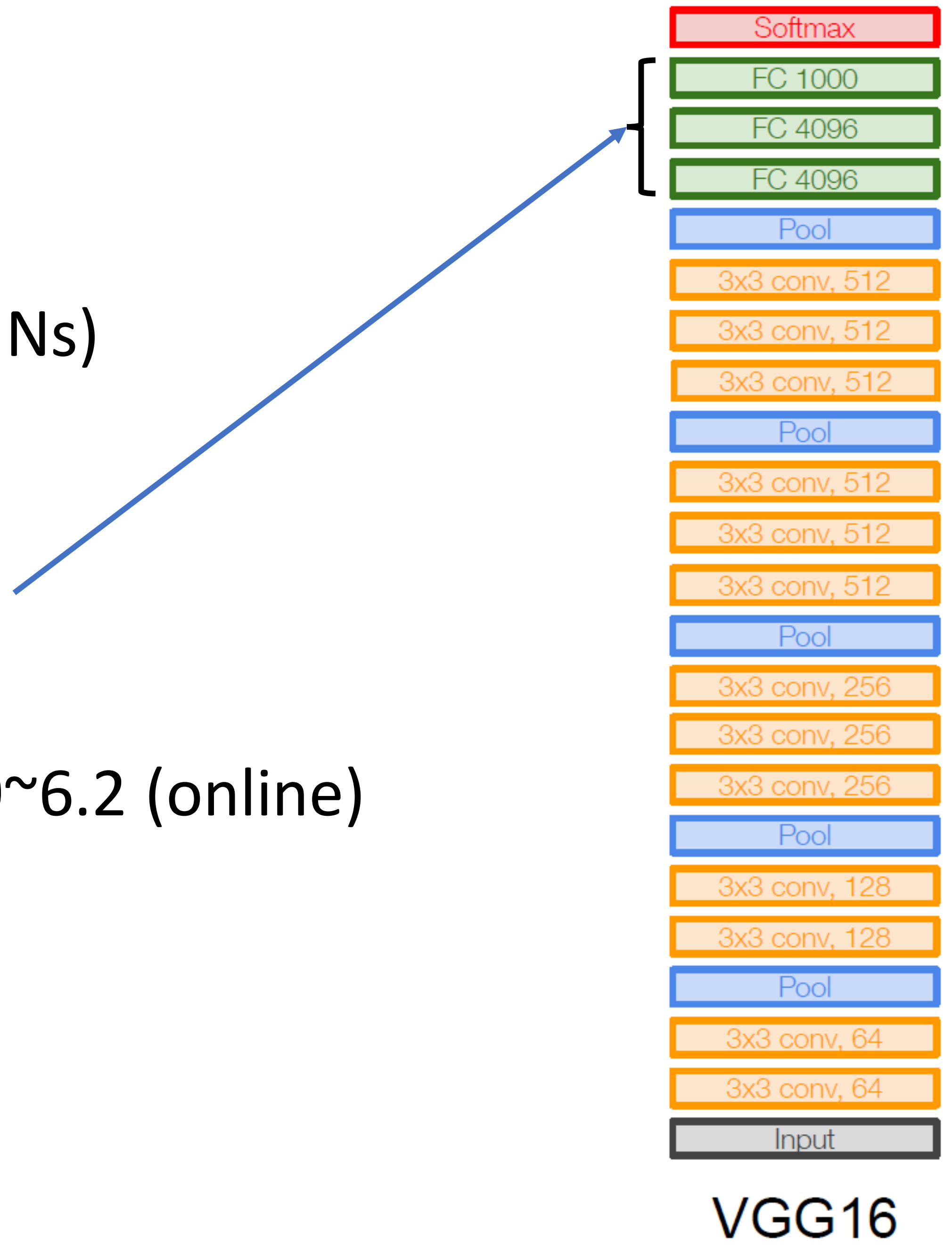
- **CHALLENGE 2**
- High-level
  - CNNs

Matching

- Similarity
  - Distance
  - Cosine similarity
- ML classifier
  - KNN

CHALLENGE 2

- Convolutional Neural Networks (CNNs)
  - will be covered in next class
- Extract CNN output feature vectors
- Pytorch-based CNN practice in 5.20~6.2 (online)
- Banning the use of transformers!
  - ViT, Swin, CLIP, …



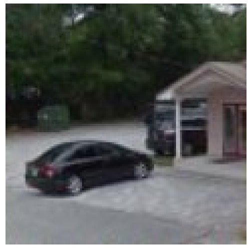
DB Images (train image)

- link
  - git clone https://github.com/folfcoder/recaptcha-dataset.git
- Classes (without mountain)

Bicycle (800)

Bridge (553)

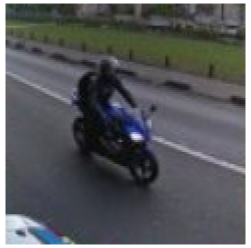
Bus (1229)


Car (3578)

Chimney (56)

Crosswalk (1260)

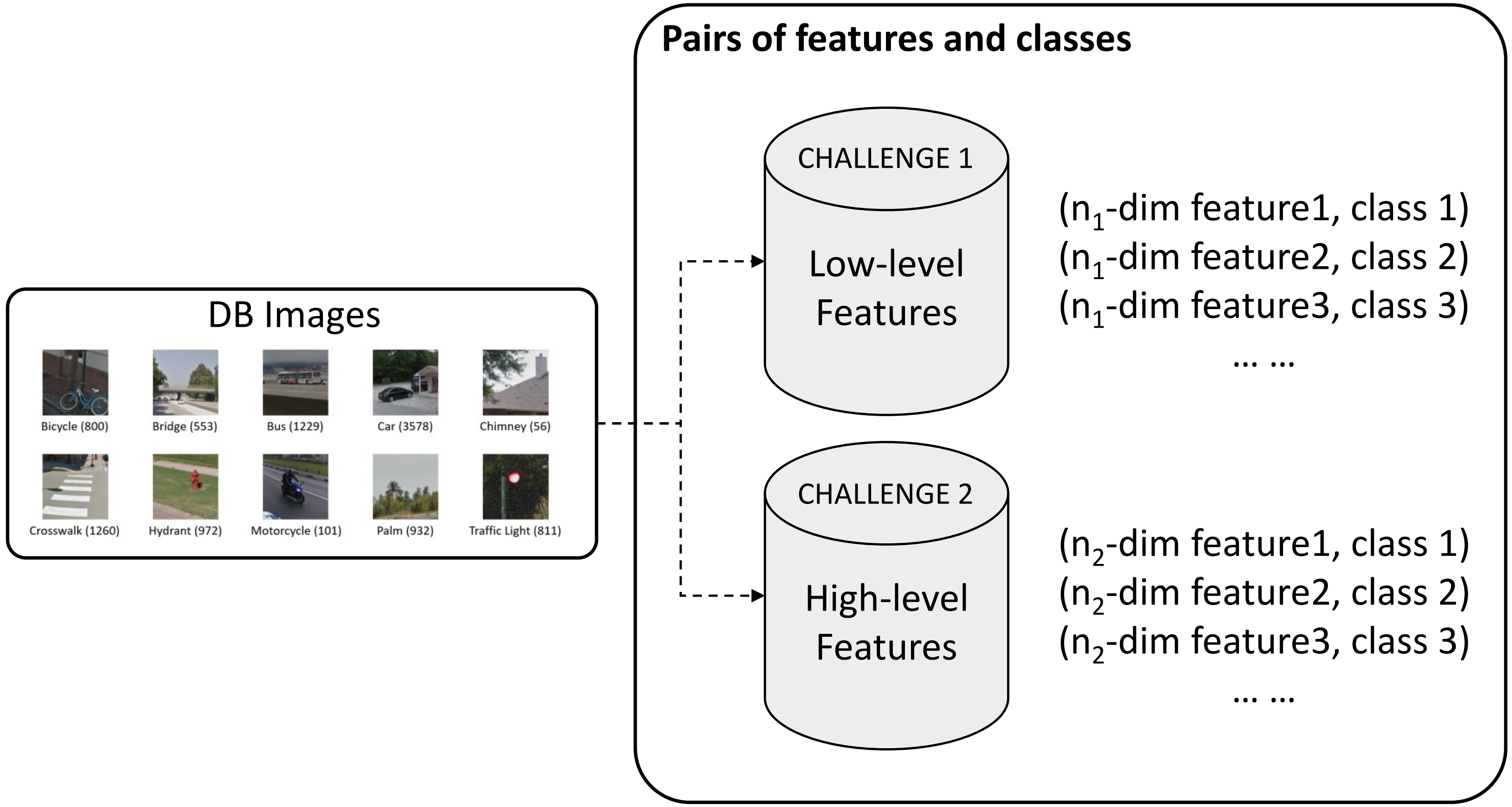
Hydrant (972)


Motorcycle (101)

Palm (932)

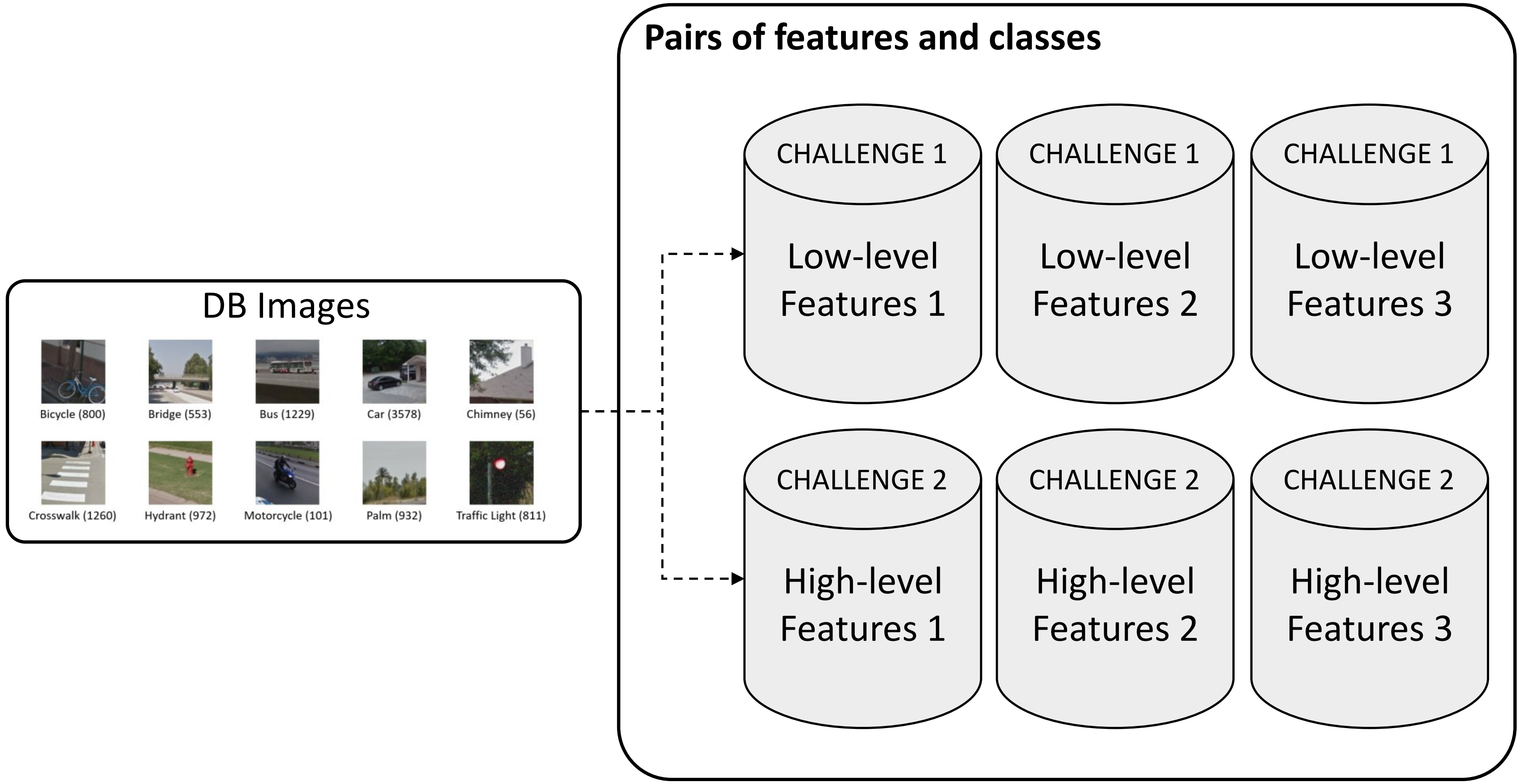
Traffic Light (811)

Representing DB images by feature vectors



Explore and save the best feature sets using cross-validation in DB image

Representing DB images by feature vectors

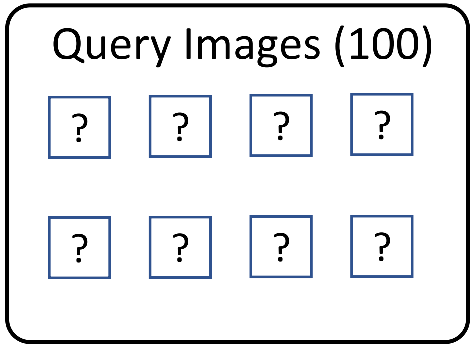


You can save up to <u>three feature sets</u> for each CHALLENGE

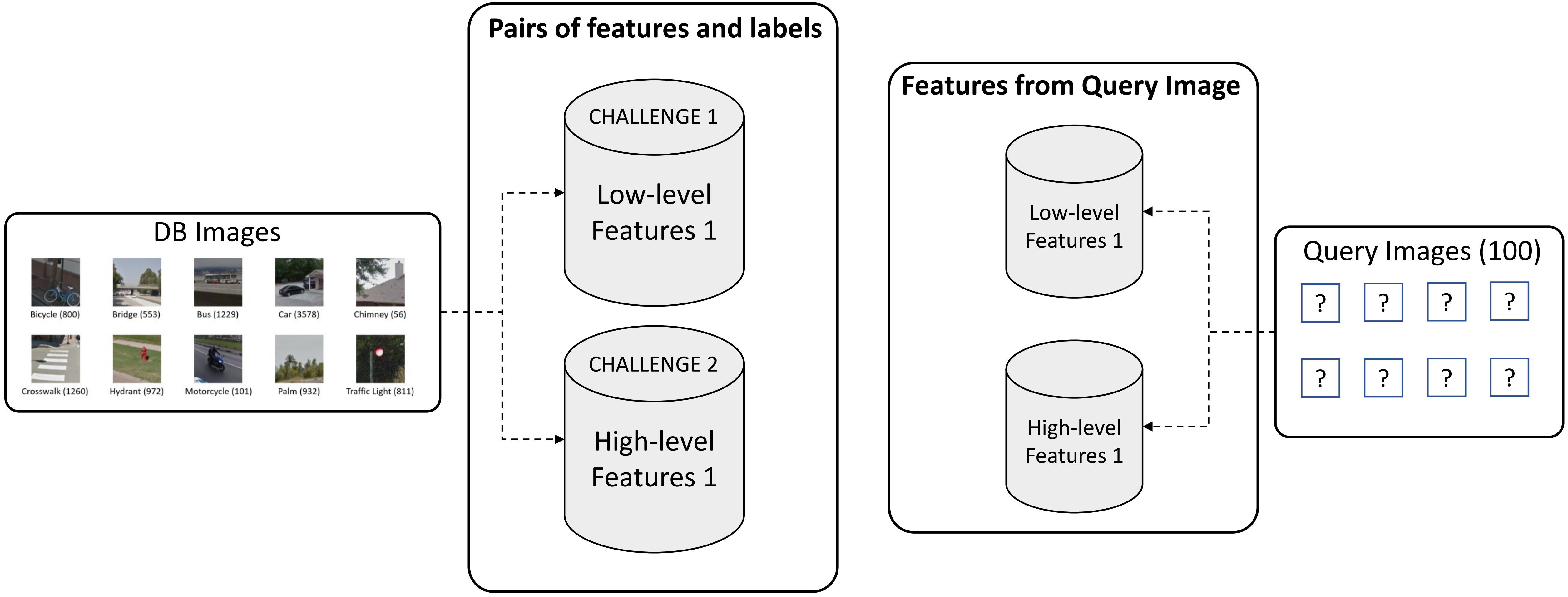
Query Images (test image)

- Total 100 images
  - 10 images for each class
- Query Images will be available on **12 June (Wen)**
  - We provide query images in class, without label
- Query folder structure

```
query
|----query001.png
|----query002.png
|----...
|----query100.png
```



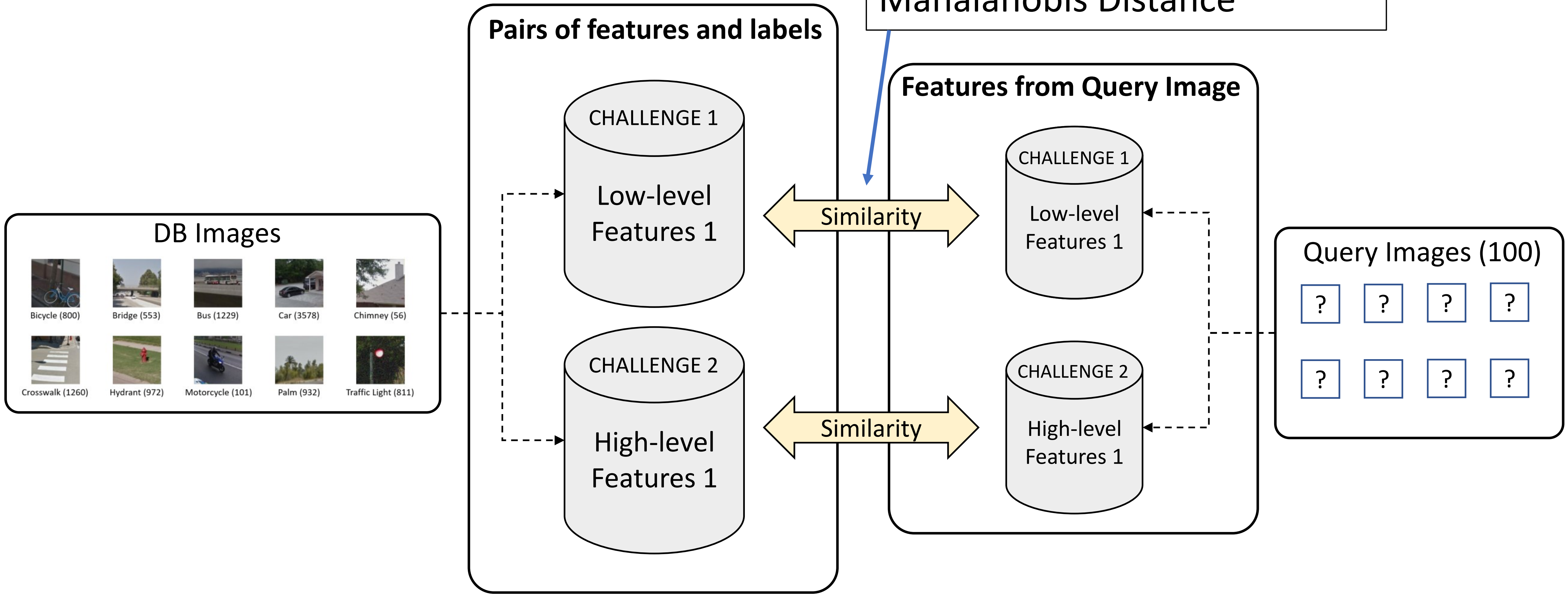
During the Challenge
12 Jun.



Step 1. representing query images by your own feature sets

During the Challenge
12 Jun.

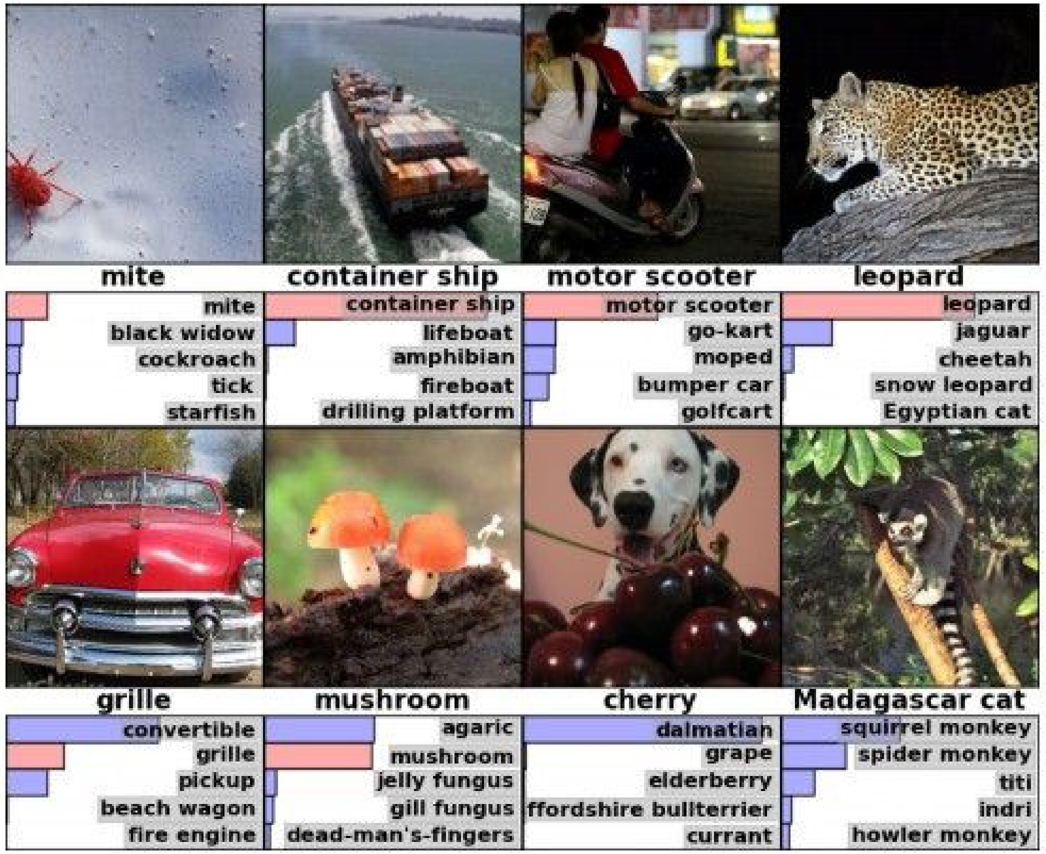
Cosine Similarity
Manhattan distance (L1 norm)
Euclidean distance (L2 norm)
Mahalanobis Distance



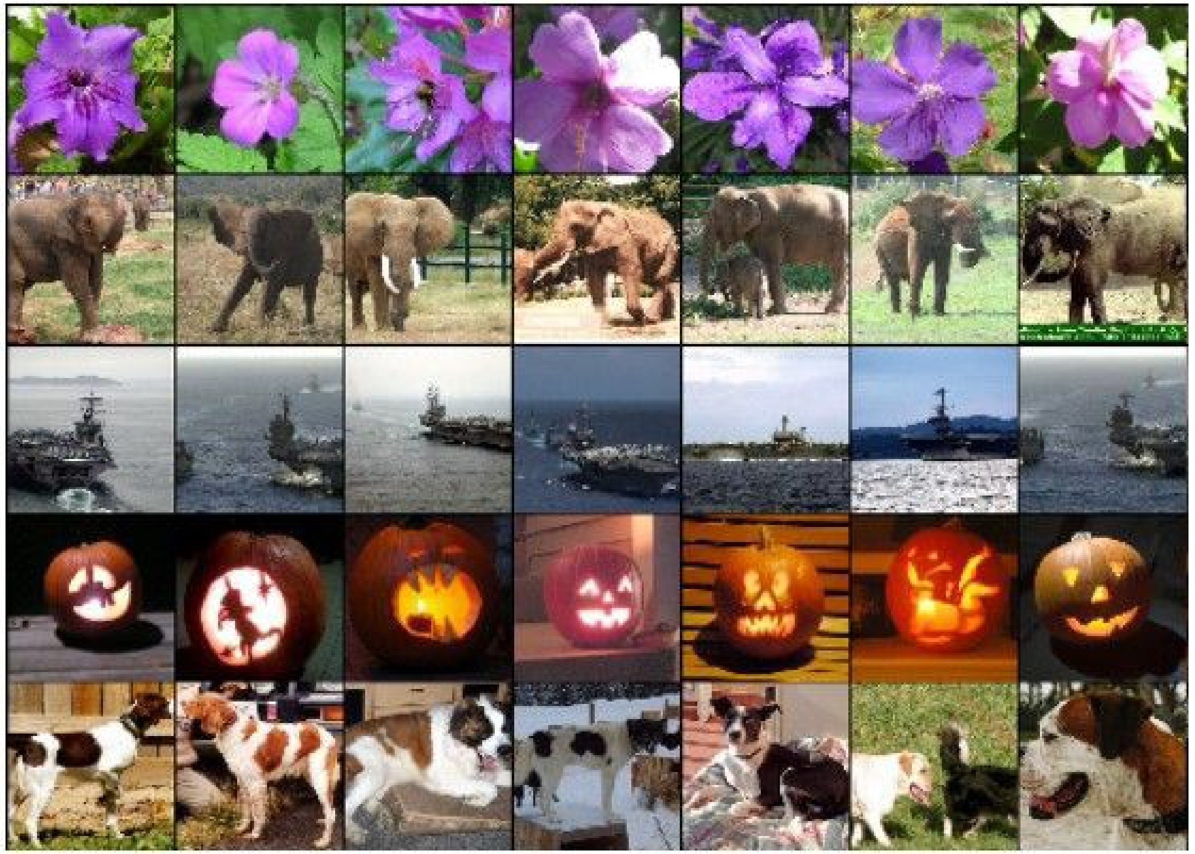
Step2. Calculate the similarity between features of all DB Images and given query Image

Tasks for the CHALLENGE

Task1: Classification

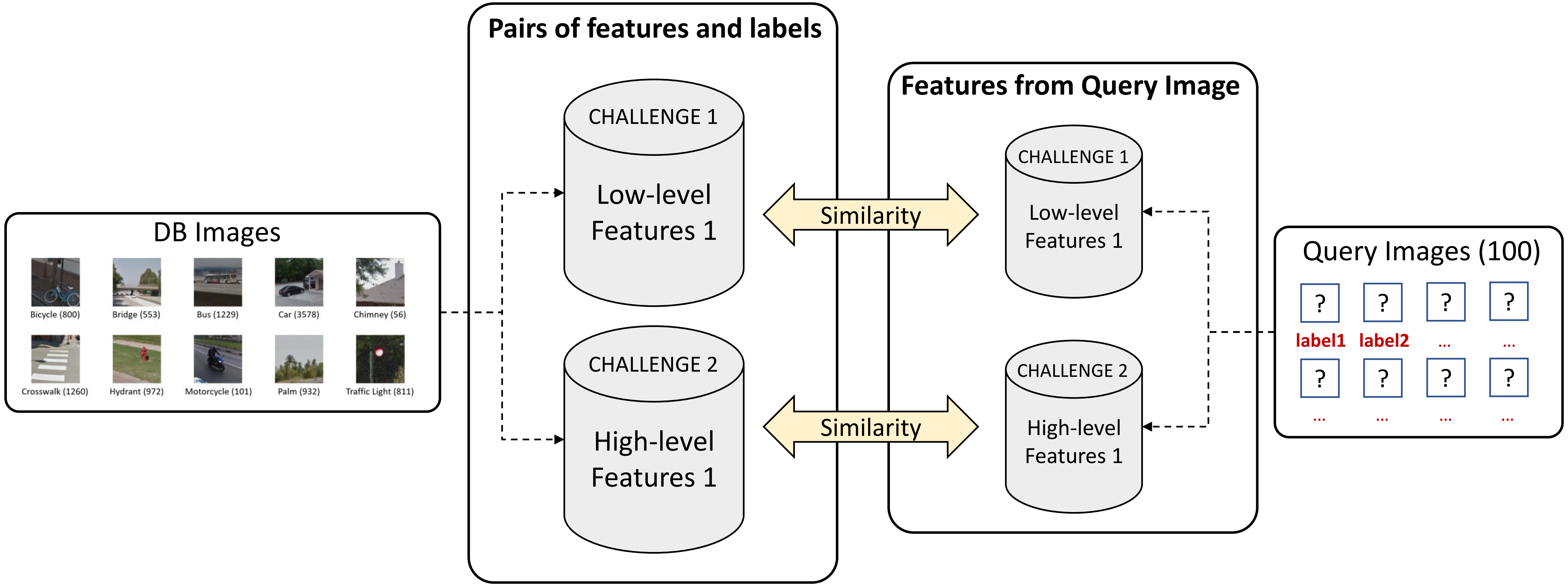


Task2: Retrieval



Figures copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012.

During the Challenge
12 Jun.



Step3. Classification Result (Task 1): Classify and tag the label for every query image
Step4. Retrieval Result (Task 2): Retrieve top ranked 10 images for every query image

Classification Result (Task 1)

- Outputs
  - 100 dimension array
  - Save the results as csv file for each attempt

```
predict_labels = classifier.predict(test_features)
print(predict_labels)    # array(100)
```

```
['Hydrant' 'Crosswalk' 'Bus' 'Motorcycle' 'Bicycle' 'Bicycle' 'Chimney'
 'Bicycle' 'Car' 'Bridge' 'Chimney' 'Chimney' 'Bicycle' 'Bridge' 'Bridge'
 'Hydrant' 'Bridge' 'Chimney' 'Hydrant' 'Bridge' 'Bus' 'Bus' 'Bridge'
 'Bus' 'Bridge' 'Chimney' 'Bridge' 'Bus' 'Bus' 'Bridge' 'Car' 'Car'
 'Motorcycle' 'Chimney' 'Bridge' 'Bus' 'Bicycle' 'Crosswalk' 'Hydrant'
 'Car' 'Chimney' 'Hydrant' 'Bus' 'Hydrant' 'Crosswalk' 'Car' 'Car'
 'Bridge' 'Chimney' 'Hydrant' 'Crosswalk' 'Bus' 'Bridge' 'Hydrant' 'Car'
 'Car' 'Hydrant' 'Car' 'Car' 'Bicycle' 'Hydrant' 'Chimney' 'Car'
 'Crosswalk' 'Car' 'Bus' 'Car' 'Bicycle' 'Car' 'Bicycle' 'Car' 'Chimney'
 'Bicycle' 'Chimney' 'Motorcycle' 'Chimney' 'Bus' 'Bus' 'Car' 'Bus' 'Bus'
 'Bicycle' 'Bridge' 'Bridge' 'Bus' 'Car' 'Hydrant' 'Car' 'Bicycle'
 'Hydrant' 'Bridge' 'Bicycle' 'Bus' 'Car' 'Crosswalk' 'Chimney' 'Bus'
 'Bus' 'Motorcycle' 'Motorcycle']
```

```
import csv

with open('c1_t1_a1.csv','w') as file :
    write = csv.writer(file)
    for i, predict_label in enumerate(predict_labels):
        write.writerow([f'query{i+1:03}.png', predict_label])
```

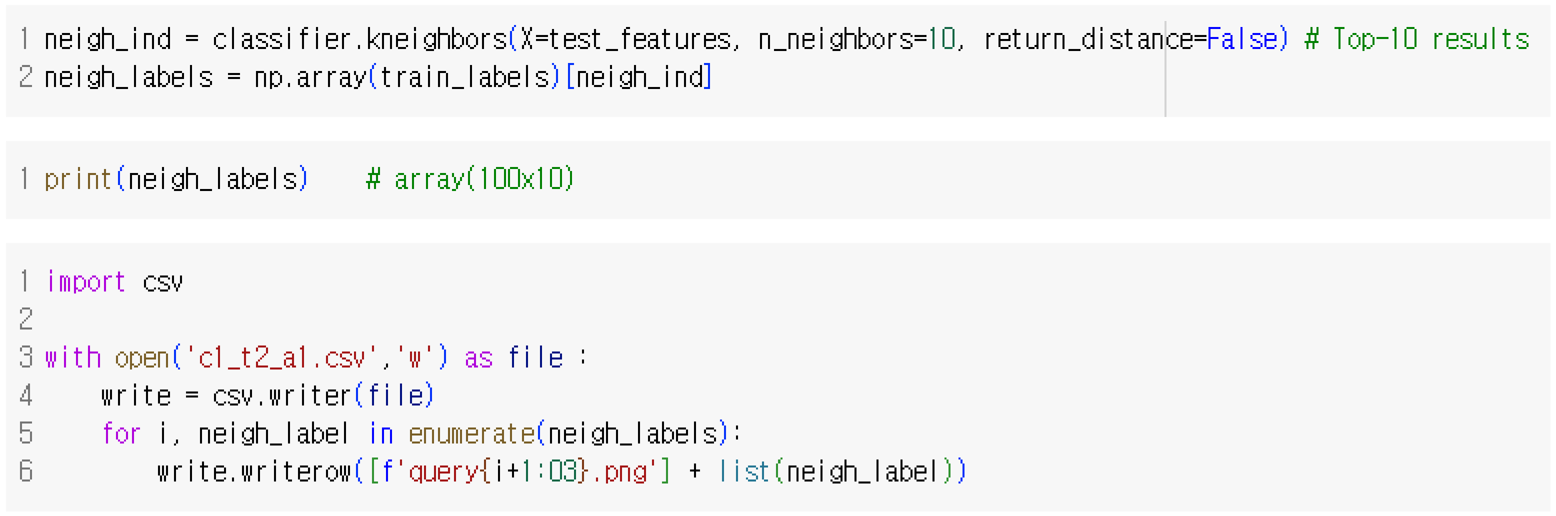
c1_t1_a1.csv

	A	B
1	query001.p	Hydrant
2	query002.p	Crosswalk
3	query003.p	Bus
4	query004.p	Motorcycle
5	query005.p	Bicycle
6	query006.p	Bicycle
7	query007.p	Chimney
8	query008.p	Bicycle
9	query009.p	Car
10	query010.p	Bridge
11	query011.p	Chimney
12	query012.p	Chimney
96	query096.p	Chimney
97	query097.p	Bus
98	query098.p	Bus
99	query099.p	Motorcycle
100	query100.p	Motorcycle

https://colab.research.google.com/drive/1nC_YMtHQn1GYMzBUGRS3esftKS8YKAZn?hl=ko#scrollTo=Ad4y-3QSh5mA

Retrieval Result (Task 2)

- Outputs
  - 100x10 dimension array
  - Save the results as csv file for each attempt



```python
neigh_ind = classifier.kneighbors(X=test_features, n_neighbors=10, return_distance=False) # Top-10 results
neigh_labels = np.array(train_labels)[neigh_ind]
```

```python
print(neigh_labels)    # array(100x10)
```

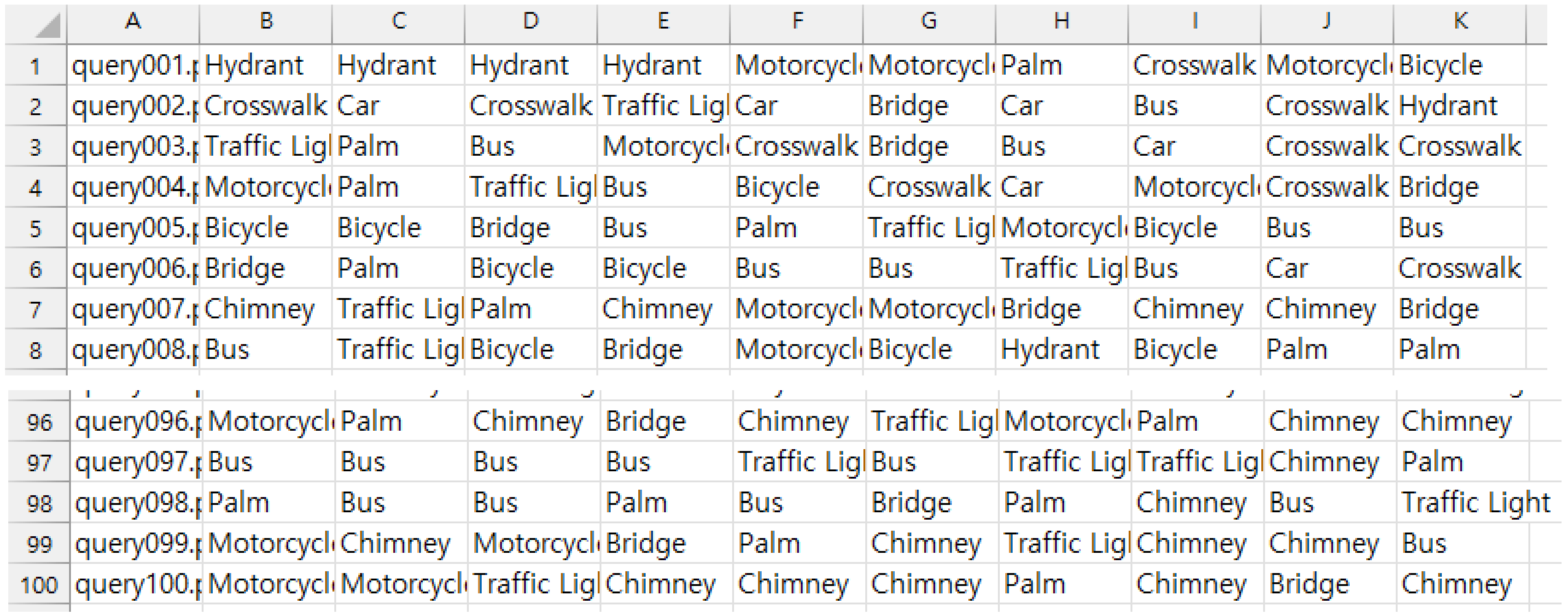
```python
import csv

with open('c1_t2_a1.csv','w') as file :
    write = csv.writer(file)
    for i, neigh_label in enumerate(neigh_labels):
        write.writerow([f'query{i+1:03}.png'] + list(neigh_label))
```

https://colab.research.google.com/drive/1nC_YMtHQn1GYMzBUGRS3esftKS8YKAZn?hl=ko#scrollTo=Ad4y-3QSh5mA

Retrieval Result (Task 2)

- Outputs
  - 100x10 dimension array
  - Save the results as csv file for each attempt

c1_t2_a1.csv



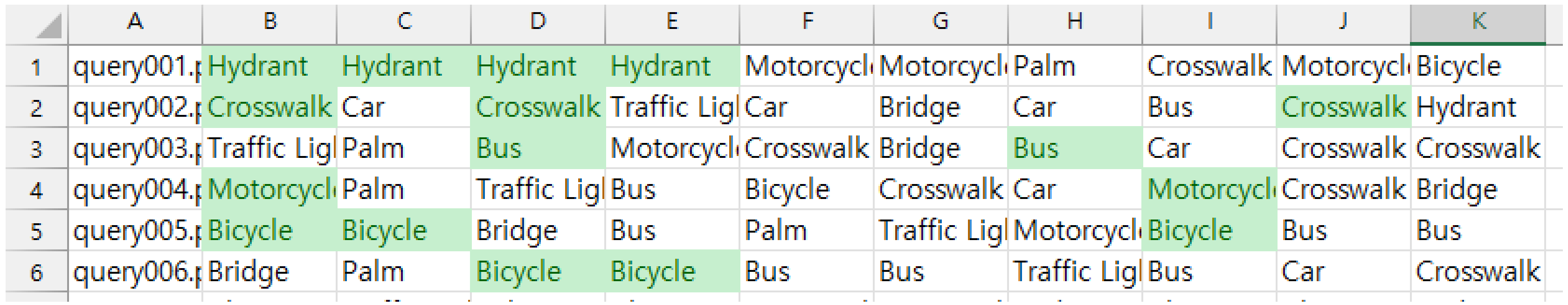
	A	B	C	D	E	F	G	H	I	J	K
1	query001.p	Hydrant	Hydrant	Hydrant	Hydrant	Motorcycl	Motorcycl	Palm	Crosswalk	Motorcycl	Bicycle
2	query002.p	Crosswalk	Car	Crosswalk	Traffic Lig	Car	Bridge	Car	Bus	Crosswalk	Hydrant
3	query003.p	Traffic Lig	Palm	Bus	Motorcycl	Crosswalk	Bridge	Bus	Car	Crosswalk	Crosswalk
4	query004.p	Motorcycl	Palm	Traffic Lig	Bus	Bicycle	Crosswalk	Car	Motorcycl	Crosswalk	Bridge
5	query005.p	Bicycle	Bicycle	Bridge	Bus	Palm	Traffic Lig	Motorcycl	Bicycle	Bus	Bus
6	query006.p	Bridge	Palm	Bicycle	Bicycle	Bus	Bus	Traffic Lig	Bus	Car	Crosswalk
7	query007.p	Chimney	Traffic Lig	Palm	Chimney	Motorcycl	Motorcycl	Bridge	Chimney	Chimney	Bridge
8	query008.p	Bus	Traffic Lig	Bicycle	Bridge	Motorcycl	Bicycle	Hydrant	Bicycle	Palm	Palm
96	query096.p	Motorcycl	Palm	Chimney	Bridge	Chimney	Traffic Lig	Motorcycl	Palm	Chimney	Chimney
97	query097.p	Bus	Bus	Bus	Bus	Traffic Lig	Bus	Traffic Lig	Traffic Lig	Chimney	Palm
98	query098.p	Palm	Bus	Bus	Palm	Bus	Bridge	Palm	Chimney	Bus	Traffic Light
99	query099.p	Motorcycl	Chimney	Motorcycl	Bridge	Palm	Chimney	Traffic Lig	Chimney	Chimney	Bus
100	query100.p	Motorcycl	Motorcycl	Traffic Lig	Chimney	Chimney	Chimney	Palm	Chimney	Bridge	Chimney

https://colab.research.google.com/drive/1nC_YMtHQn1GYMzBUGRS3esftKS8YKAZn?hl=ko#scrollTo=Ad4y-3QSh5mA

Ranking

- Task1: Classification
  1. Highest accuracy of all attempts
  2. Ties will be broken by F1 score

- Task2: Retrieval
  1. Highest (average) Top-10 accuracy of all attempts
  2. Ties will be broken by the ranking of correct label



	A	B	C	D	E	F	G	H	I	J	K
1	query001.p	Hydrant	Hydrant	Hydrant	Hydrant	Motorcycl	Motorcycl	Palm	Crosswalk	Motorcycl	Bicycle
2	query002.p	Crosswalk	Car	Crosswalk	Traffic Ligl	Car	Bridge	Car	Bus	Crosswalk	Hydrant
3	query003.p	Traffic Ligl	Palm	Bus	Motorcycl	Crosswalk	Bridge	Bus	Car	Crosswalk	Crosswalk
4	query004.p	Motorcycl	Palm	Traffic Ligl	Bus	Bicycle	Crosswalk	Car	Motorcycl	Crosswalk	Bridge
5	query005.p	Bicycle	Bicycle	Bridge	Bus	Palm	Traffic Ligl	Motorcycl	Bicycle	Bus	Bus
6	query006.p	Bridge	Palm	Bicycle	Bicycle	Bus	Bus	Traffic Ligl	Bus	Car	Crosswalk

Matching Results

- You should upload
  - Output files
    - c1_t1_a1.csv, c1_t1_a2.csv, c1_t1_a3.csv
    - c1_t2_a1.csv, c1_t2_a2.csv, c1_t2_a3.csv
    - c2_t1_a1.csv, c2_t1_a2.csv, c2_t1_a3.csv
    - c2_t2_a1.csv, c2_t2_a2.csv, c2_t2_a3.csv
  - Python code (.py or .ipynb)
    - c1_t1_a1.ipynb, c1_t1_a2.ipynb, c1_t1_a3.ipynb
    - c1_t2_a1.ipynb, c1_t2_a2.ipynb, c1_t2_a3.ipynb
    - c2_t1_a1.ipynb, c2_t1_a2.ipynb, c2_t1_a3.ipynb
    - c2_t2_a1.ipynb, c2_t2_a2.ipynb, c2_t2_a3.ipynb
  - DB features file (.npy or .pkl or others) if necessary

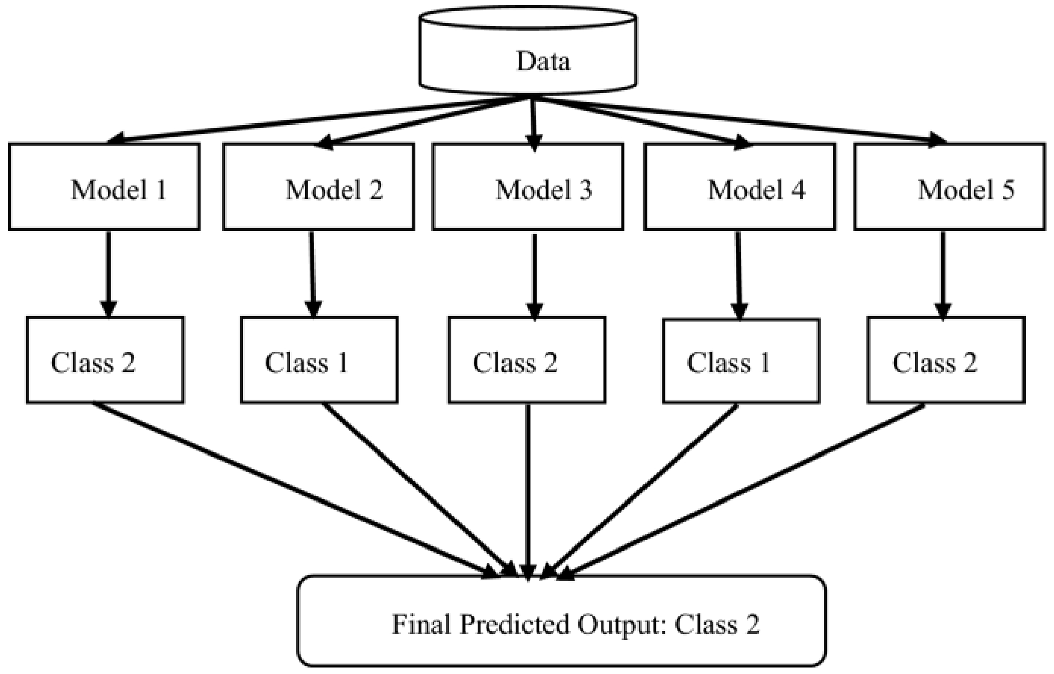
Tips & Tricks

- Normalize the features
  - Normalize the features to have zero mean and unit variance
  - e.g. Concatenation(Norm(GLCM), Norm(histogram))
- Consider using a dimensionality reduction technique
  - It works well on low-dimension data (compact features)
- Cross-validation
  - try different hyperparameter values and keep the values that lead to the best performance on the validation set

Tips & Tricks

- Ensemble methods
  - Majority Voting
  - Weighted Voting

  - Stable and improve accuracy
  - Pay computational cost at test time


Data
Model 1
Model 2
Model 3
Model 4
Model 5
Class 2
Class 1
Class 2
Class 1
Class 2
Final Predicted Output: Class 2