ECE 285 Assignment 1: KNN

For this part of assignment, you are tasked to implement KNN algorithm and test it on the a subset of CIFAR10 dataset.

You sould run the whole notebook and answer the question in the notebook.

TO SUBMIT: PDF of this notebook with all the required outputs and answers.

In [22]:

```
# Import Packages
import numpy as np
import matplotlib.pyplot as plt
```

Prepare Dataset

Since CIFAR10 is a relative large dataset, and KNN is quite time-consuming method, we only a small sub-set of CIFAR10 for KNN part

In [23]:

```
from ece285.utils.data_processing import get_cifar10_data

# Use a subset of CIFAR10 for KNN assignments
dataset = get_cifar10_data(subset_train=5000, subset_val=250, subset_test=500)

print(dataset.keys())
print("Training Set Data  Shape: ", dataset["x_train"].shape)
print("Training Set Label Shape: ", dataset["y_train"].shape)
```

```
dict_keys(['x_train', 'y_train', 'x_val', 'y_val', 'x_test', 'y_test'])
Training Set Data  Shape:  (5000, 3072)
Training Set Label Shape:  (5000,)
```

Implementation (60%)

You need to implement the KNN method in `algorithms/knn.py` . You need to fill in the prediction function(since the training of KNN is just remembering the training set).

For KNN implementation, you are tasked to implement two version of it.

- Two Loop Version: use one loop to iterate through training samples and one loop to iterate through test samples
- One Loop Version: use one loop to iterate through test samples and use broadcast feature of numpy to calculate all the distance at once

Note: It is possible to build a Fully Vectorized Version without explicit for loop to calculate the distance, but you do not have to do it in this assignment.

For distance function, in this assignment, we use Euclean distance between samples.

In [24]:

```
from ece285.algorithms import KNN

knn = KNN(num_class=10)
knn.train(
    x_train=dataset["x_train"],
    y_train=dataset["y_train"],
    k=5,
)
```

Compare the time consumption of different method

In this section, you will test your different implementation of KNN method, and compare their speed.

In [25]:
```
from ece285.utils.evaluation import get_classification_accuracy
```

Two Loop Version:

In [26]:
```
import time
c_t = time.time()
prediction = knn.predict(dataset["x_test"], loop_count=2)
print("Two Loop Prediction Time:", time.time() - c_t)

test_acc = get_classification_accuracy(prediction, dataset["y_test"])
print("Test Accuracy:", test_acc)
```

```
Two Loop Prediction Time: 761.9649147987366
Test Accuracy: 0.278
```

One Loop Version

In [27]:
```
import time

c_t = time.time()
prediction = knn.predict(dataset["x_test"], loop_count=1)
print("One Loop Prediction Time:", time.time() - c_t)

test_acc = get_classification_accuracy(prediction, dataset["y_test"])
print("Test Accuracy:", test_acc)
```

```
One Loop Prediction Time: 28.30700421333313
Test Accuracy: 0.278
```

Your different implementation should output the exact same result

Test different Hyper-parameter(20%)

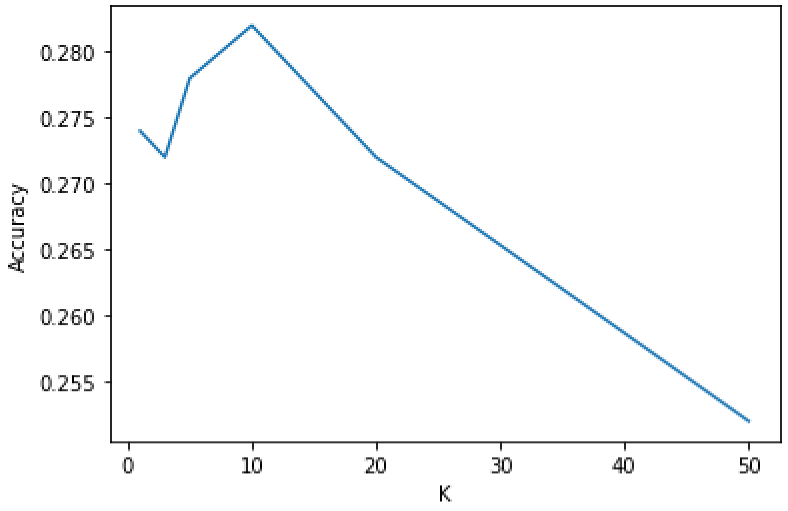
For KNN, there is only one hyper-parameter of the algorithm: How many nearest neighbour to use(**K**).

Here, you are provided the code to test different k for the same dataset.

In [32]:
```
accuracies = []

k_candidates = [1, 3, 5, 10, 20, 50]
for k_cand in k_candidates:
    prediction = knn.predict(x_test=dataset["x_test"], k=k_cand)
    acc = get_classification_accuracy(prediction, dataset["y_test"])
    accuracies.append(acc)
plt.ylabel("Accuracy")
plt.xlabel("K")
plt.plot(k_candidates, accuracies)
plt.show()

print(accuracies)
```



Inline Question 1:

Please describe the output result you get, and provide some explanation as well.

Your Answer:

First, the overall accuracy of KNN prediction is low, around 30%, the highest accuracy is at k=10.

Second, the 1 loop versionis much more faster that the two loop version. The one loop version takes around 25 seconds. The 2 loop version takes about 12 minutes (Maybe I have the wrong implementation).

Last but not least, the accuracy does not monotonically increase with K. It reaches a peak at k=10 (28.2%) and start decreasing. This confirms that KNN algorithm does not necessarily have better performance with larger K. The result may be infected by the proportion of each class.

Try different feature representation(20%)

Since machine learning method rely heavily on the feature extraction, you will see how different feature representation affect the performance of the algorithm in this section.

You are provided the code about using **HOG** descriptor to represent samples in the notebook.

In [29]:

```
from ece285.utils.data_processing import get_cifar10_data
from ece285.utils.data_processing import HOG_preprocess
from functools import partial

# Delete previous dataset to save memory
del dataset
del knn

# Use a subset of CIFAR10 for KNN assignments
hog_p_func = partial(
    HOG_preprocess,
    orientations=9,
    pixels_per_cell=(4, 4),
    cells_per_block=(1, 1),
    visualize=False,
    multichannel=True,
)
dataset = get_cifar10_data(
    feature_process=hog_p_func, subset_train=5000, subset_val=250, subset_test=500
)
```

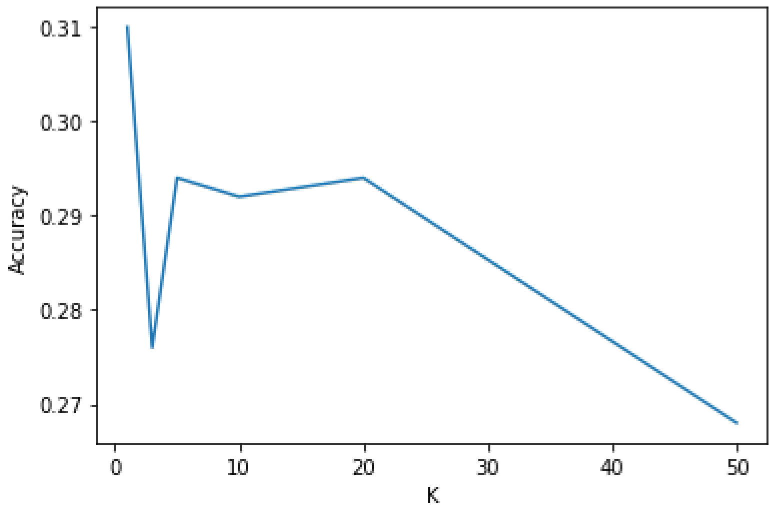
```
Start Processing
Processing Time: 8.607496500015259
```

In [30]:

```
knn = KNN(num_class=10)
knn.train(
    x_train=dataset["x_train"],
    y_train=dataset["y_train"],
    k=5,
)
accuracies = []

k_candidates = [1, 3, 5, 10, 20, 50]
for k_cand in k_candidates:
    prediction = knn.predict(x_test=dataset["x_test"], k=k_cand)
    acc = get_classification_accuracy(prediction, dataset["y_test"])
    accuracies.append(acc)

plt.ylabel("Accuracy")
plt.xlabel("K")
plt.plot(k_candidates, accuracies)
plt.show()
```



Inline Question 2:

Please describe the output result you get, compare with the result you get in the previous section, and provide some explanation as well.

Your Answer:

First, the overal accuracy of HOG representation is around 29%, which is higher than the pixel representation, but still in unsatisfying.

Second, the accuracy of algorithm varies with the K. The highest accuracy apperas at K=1 (31%), then it rise and fall around 29%. This also shows that the accuracy of KNN may not increase with K. The result can be affected by the proportion of the classes