Homework laboratory 1

# Exercise 1

The size of the four vectors generated after loading CIFAR-10 dataset:

    print(x\_train.shape)

    print(y\_train.shape)

    print(x\_test.shape)

    print(y\_test.shape)

x\_train: (50000, 32, 32, 3)

y\_train: (50000, 1)

x\_test: (10000, 32, 32, 3)

y\_test: (10000, 1)

# Exercise 2

Visualize the first 10 images from the testing dataset with their associated labels

    for i in range(10):

        # define subplot

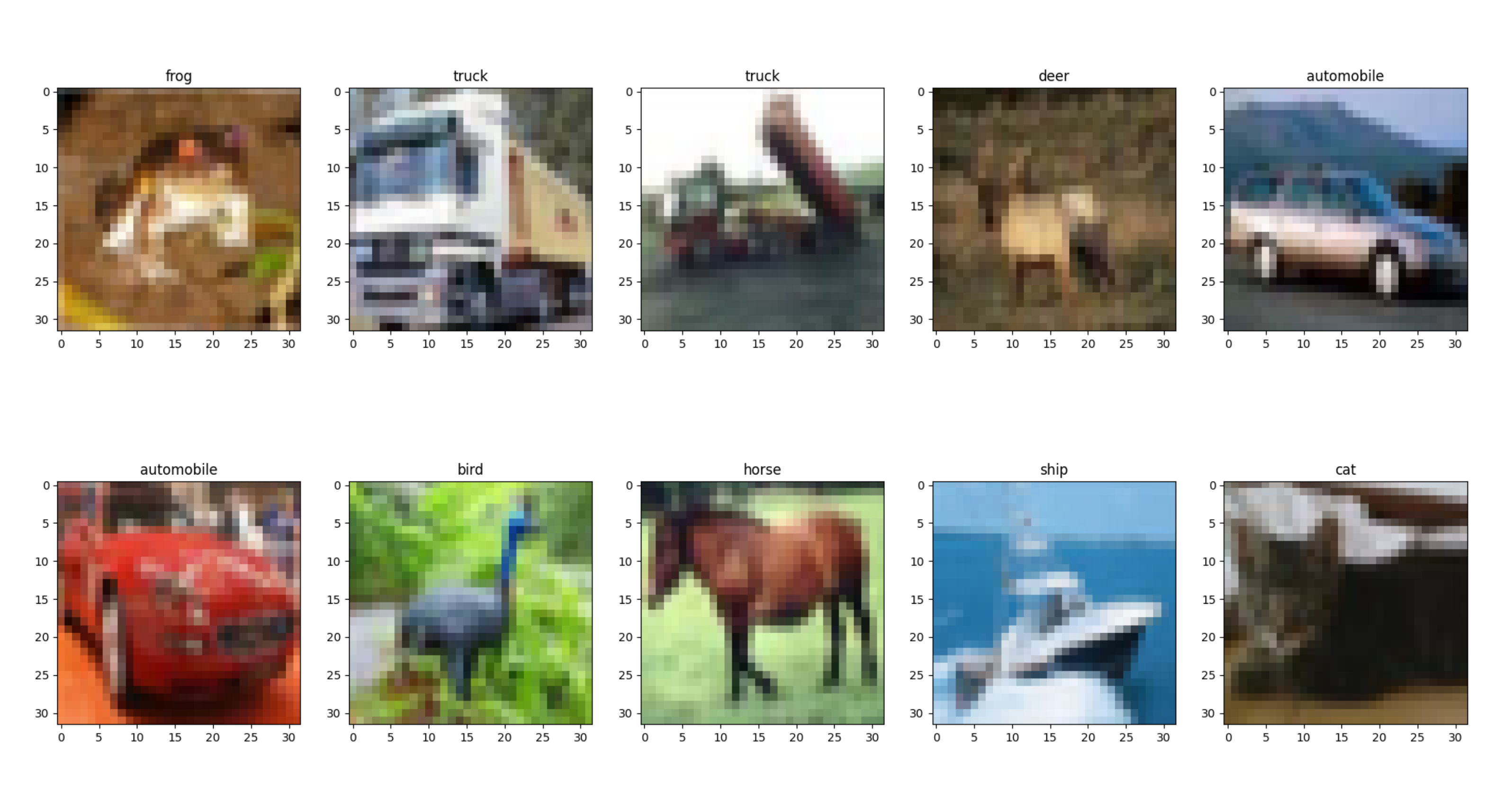
        ax = plt.subplot(2, 5, 1 + i)

        # plot raw pixel data

        plt.imshow(x\_train[i], cmap=plt.get\_cmap('gray'))

        ax.set\_title(dict\_classes[y\_train[i][0]])

    plt.show()



# Exercise 3

accuracy: 29.0%

def predictLabelNN(x\_train\_flatten, y\_train, img):

    predicted\_label\_index = -1

    scoreMin = 100000000

    for idx, imgT in enumerate(x\_train\_flatten):

        difference = abs(np.subtract(img, imgT))

        difference\_L2 = difference \*\* 2

        score = np.sqrt(np.sum(difference\_L2))

        if score < scoreMin:

            predicted\_label\_index = idx

            scoreMin = score

    return y\_train[predicted\_label\_index]

We will use this form of predict function. It uses a L2 difference for training.

# Exercise 4 and 5

def most\_frequent(l):

    count\_labels\_dict = {}

    for item in l:

        if item[1] not in count\_labels\_dict:

            count\_labels\_dict[item[1]] = 1

        else:

            count\_labels\_dict[item[1]] += 1

    return max(count\_labels\_dict, key=lambda k: count\_labels\_dict[k])

def predictLabelKNN(x\_train\_flatten, y\_train, img, k=3, diff='L2'):

    predictedLabel = -1

    predictions = []  # list to save the scores and associated labels as pairs  (score, label)

    score = 0

    for idx, imgT in enumerate(x\_train\_flatten):

        difference = abs(np.subtract(img, imgT))

        if diff == 'L2':

            difference = difference \*\* 2

        score = np.sum(difference)

        predictions.append([score, y\_train[idx][0]])

    predictions = sorted(predictions, key=lambda x: x[0])

    top\_k\_predictions = predictions[:k]

    print(top\_k\_predictions)

    predictedLabel = most\_frequent(top\_k\_predictions)

    return predictedLabel

def do\_every\_combination(N, x\_train\_flatten, y\_train, x\_test\_flatten, y\_test, diff\_type='L1'):

    info = {}

    for i in [1, 3, 5, 10, 20, 50]:

        info[i] = {}

        info[i]['correct\_predicted'] = 0

        info[i]['last\_predicted'] = 0

        info[i]['accuracy'] = 0

    for idx, img in enumerate(x\_test\_flatten[0:N]):

        print(f"Make a prediction for image {idx}")

        # TODO - Application 1 - Step 3 - Call the predictLabelNN function

        for key in info:

            info[key]['last\_predicted'] = predictLabelKNN(x\_train\_flatten, y\_train, img, k=key, diff=diff\_type)

        # TODO - Application 1 - Step 4 - Compare the predicted label with the groundtruth (the label from y\_test).

        #  If there is a match then increment the contor numberOfCorrectPredictedImages

        for key in info:

            if info[key]['last\_predicted'] == y\_test[idx]:

                info[key]['correct\_predicted'] += 1

    with open(f"./results\_{diff\_type}.txt", "w") as f:

        for key in info:

            info[key]['accuracy'] = N \* info[key]['correct\_predicted'] / 100

            f.write(f"{diff\_type} - k={key} ========= {info[key]['accuracy']}\n")

    return info

def exercises\_4\_5(diff\_type='L1'):

    (x\_train, y\_train), (x\_test, y\_test) = keras.datasets.cifar10.load\_data()

    print(x\_train.shape)

    print(y\_train.shape)

    print(x\_test.shape)

    print(y\_test.shape)

    x\_train\_flatten = x\_train.reshape(x\_train.shape[0], 32 \* 32 \* 3)

    x\_test\_flatten = x\_test.reshape(x\_test.shape[0], 32 \* 32 \* 3)

    N = 100

    do\_every\_combination(N, x\_train\_flatten, y\_train, x\_test\_flatten, y\_test, diff\_type=diff\_type)

if \_\_name\_\_ == '\_\_main\_\_':

    # main()

    exercises\_4\_5('L1')

We just have to change the argument of the last function call from ‘L1’ to ‘L2’ for exercise 5.

This script takes every photo, and it passes it through each KNN classifier (1, 3, 5, 10, 20, 50), then writes the output to files.

Output from files:

L1 - k=1 ========= 23.0

L1 - k=3 ========= 23.0

L1 - k=5 ========= 26.0

L1 - k=10 ========= 33.0

L1 - k=20 ========= 33.0

L1 - k=50 ========= 35.0

L2 - k=1 ========= 29.0

L2 - k=3 ========= 27.0

L2 - k=5 ========= 24.0

L2 - k=10 ========= 24.0

L2 - k=20 ========= 24.0

L2 - k=50 ========= 25.0

We can see that when using L2, the accuracy actually stayed the same or even decreased when we increased the number of neighbors.

I’ve changed the most\_frequency function a bit since I understand it better this way, I hope this is not a problem. Basically, we create a key for each new discovered label and initiate the value with 1. Each time we encounter this label again, we just increment the value of its key.

This process could’ve been sped up with the multiprocessing library.