2025-2026

Welcome to the 4th Lab session of CO3519 module.

For Two Hours Lab Session:

#	Task	Time (m)	Guidance
A	Please explore the concepts we covered on Monday's lecture about KNN.	10	You can think and contextualise the concepts towards your assessment.
	Please implement/run the code for image classification using KNN. Instead of the code, the screenshots are added so that each code snippet can be run and build basic understanding of training KNN.	40	Please follow the instruction on Page 2 to Page 4, for **Step 1** given there.
	Break	10	
B	Please perform with the second step on the Page 5 which is a step towards using this code for FER.	50	The second step will help you in your assessment. Build your deep understanding, explore the concepts in depth.

For One Hour Lab Session:

#	Task	Time (m)	Guidance
1	Please perform the task **A** from two hours sessions.	50	Think and contextualise the concepts. Then follow the instructions on Page 2 to Page 5 for step1.

Image Classification

In this code, we implement a basic image classification system to distinguish between cats and dogs using the K-Nearest Neighbors (KNN) algorithm. Images of cats and dogs are loaded from specified folders, resized to a uniform size, and flattened into one-dimensional arrays for model compatibility. The data is labeled (0 for cats, 1 for dogs), combined, and split into training and testing sets. The KNN classifier is trained on the processed data to predict whether new images are cats or dogs. Model performance is evaluated using metrics like accuracy, a confusion matrix, and a classification report. Additionally, the system includes functionality to preprocess a new image, make predictions, and visually display the image along with its predicted label.

1. Please run the following code for classification/recognition of cat/dog:

Import necessary libraries. **os**: For navigating the filesystem and accessing image files in specified folders. **numpy**: For numerical operations, including array manipulation for image data. **train_test_split**: Splits the dataset into training and testing subsets. **KNeighborsClassifier**: Implements the K-Nearest Neighbors (KNN) classification algorithm. **accuracy_score**: Measures the accuracy of predictions. **PIL.Image**: For loading and resizing image data.

```
1 import os
2 import numpy as np
3 from sklearn.model_selection import train_test_split
4 from sklearn.neighbors import KNeighborsClassifier
5 from sklearn.metrics import accuracy_score
6 from PIL import Image
```

Please do mounting and access the drive.

```
1 from google.colab import drive
2 drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
```

(Load images and labels) Loads images from a folder and assigns a label to each image (e.g., 0 for cats, 1 for dogs).
os.listdir(folder): Iterates over all image filenames in the given folder.
Image.open: Opens each image file.
img.resize(image_size): Resizes the image to a consistent size (64x64 pixels here).
np.array(img).flatten(): Converts the image into a 1D array (flattening) for use as input to the model.
Appends the processed image data to the images list and the corresponding label to the labels list.
Return Value: A list of flattened image arrays and their respective labels.

```python
def load_images_from_folder(folder, label, image_size=(64, 64)):
    images = []
    labels = []
    for filename in os.listdir(folder):
        img = Image.open(os.path.join(folder, filename))
        img = img.resize(image_size)
        img_array = np.array(img).flatten()  # Flatten the image
        images.append(img_array)
        labels.append(label)
    return images, labels
```

Load both cats and dog's data. Then combine them.
Uses the load_images_from_folder function to load and label:
Cat images (label 0) from the data/cat directory.
Dog images (label 1) from the data/dog directory.
The data can be found under the **Week4** in data folder.
Combines cat and dog images into a single dataset (X) and their corresponding labels into a single label array (y).
Converts both lists into NumPy arrays for compatibility with scikit-learn.

```python

cat_images, cat_labels = load_images_from_folder('/content/drive/My Drive/data/cat', 0)
dog_images, dog_labels = load_images_from_folder('/content/drive/My Drive/data/dog', 1)


X = np.array(cat_images + dog_images)
y = np.array(cat_labels + dog_labels)
```

Split into training and testing.
Divides the dataset into **Training set** (80%): Used to train the KNN model.
Testing set (20%): Used to evaluate the model's performance on unseen data.
random_state=42 ensures reproducibility.

```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

```

Initialize and train the Model
Creates a KNN model with k=5 (considers the 5 nearest neighbors for classification).
Trains the model on the training data (X_train, y_train).

```python

k = 5  # Set number of neighbors
knn = KNeighborsClassifier(n_neighbors=k)


knn.fit(X_train, y_train)
```

```
KNeighborsClassifier
KNeighborsClassifier()
```

Predicts the labels for the testing dataset (X_test).

Calculates the accuracy of the model using accuracy_score:

Accuracy is the percentage of correctly classified images.

Confusion Matrix:

Shows the counts of correct and incorrect predictions for each class.

Rows represent actual labels; columns represent predicted labels.

Classification Report:

Includes precision, recall, F1-score, and support for each class.

target_names maps numerical labels (0 for cats, 1 for dogs) to readable class names.

```python
from sklearn.metrics import confusion_matrix, classification_report

# Predict on the test data
y_pred = knn.predict(X_test)

# Calculate test accuracy
test_accuracy = accuracy_score(y_test, y_pred)
print(f"Test Accuracy: {test_accuracy * 100:.2f}%")

# Generate the confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(conf_matrix)

# Generate the classification report
class_report = classification_report(y_test, y_pred, target_names=["Cat", "Dog"])
print("Classification Report:")
print(class_report)
```

Testing Single Image

Once we train the mode, now we will prepare the image for prediction by:

Resizing the image to the same size used during training (64x64).

Flattening the image to a 1D array.

Reshaping it to a 2D array, as required by the predict method of scikit-learn models.

```python
def preprocess_image(image_path, image_size=(64, 64)):
    img = Image.open(image_path)
    img = img.resize(image_size)
    img_array = np.array(img).flatten()  # Flatten to 1D array
    return img_array.reshape(1, -1)  # Reshape to 2D array for prediction

```

Make sure that the path is correct.

Display Function:

Displays the new image along with the predicted label (Cat or Dog).

Prediction Pipeline:

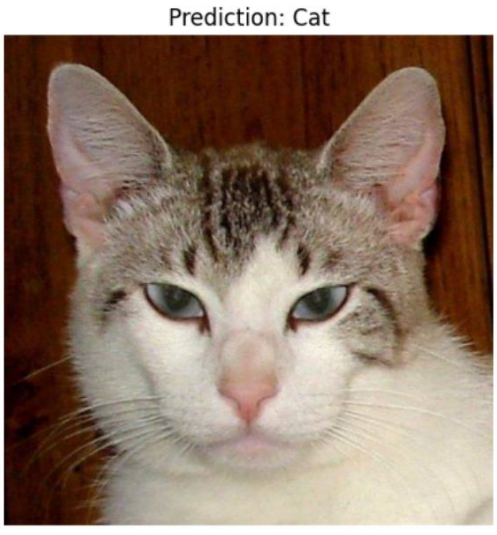
Preprocesses the image using the preprocess_image function.

Predicts the label using the trained KNN model.

Calls display_image_with_label to show the image and prediction.

```
import matplotlib.pyplot as plt

def display_image_with_label(image_path, label):
    img = Image.open(image_path)
    plt.imshow(img)
    plt.axis('off')  # Hide axis
    plt.title(f"Prediction: {'Cat' if label == 0 else 'Dog'}")
    plt.show()
new_image_path = '/content/drive/My Drive/testt/cat.255.jpg'
new_image = preprocess_image(new_image_path)
prediction = knn.predict(new_image)[0]
display_image_with_label(new_image_path, prediction)
```



Note: Running all the above code. You will learn how to KNN works for classification. At the end, you can test a single image. Similarly, the accuracy obtained night be good. In this way, think on how you can improve it.

2. After running the code above, apply this to the assessment.

To apply this, you will add the required number of classes. Update the labels and class information. Similarly, there is also a file located in the Week4 folder (Assignment_help>KNN2) which takes the training and test data separately from a folder. You can modify that code for this purpose and get the facial expression dataset from the Dataset folder.